МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ГРАЖДАНСКОЙ АВИАЦИИ

Н.И. Черкасова

# ОПЕРАЦИОННЫЕ СИСТЕМЫ

Москва
2017

**ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)**

---

**Кафедра вычислительных машин, комплексов, систем и сетей**

Н.И. Черкасова

# ОПЕРАЦИОННЫЕ СИСТЕМЫ

Тексты лекций

Утверждено Редакционно-
издательским советом МГТУ ГА
в качестве учебного пособия

**Москва-2017**

УДК 004.451
ББК 6Ф7.3
Ч-48

Печатается по решению редакционно-издательского совета
Московского государственного технического университета ГА

Рецензенты: канд. техн. наук, доц. Л.А. Вайнейкис (МГТУ ГА);
канд. физ.-мат. наук, доц. В.Р. Соловьев (ФПФЭ МФТИ)

Черкасова Н.И.

Ч-48 Операционные системы: учебное пособие. – М.: МГТУ ГА, 2017. – 84 с., лит.: 9 наим., 21 рис., 3 табл.

ISBN 978-5-86311-988-5

Данные тексты лекций содержат материалы учебно-методического характера, необходимые для освоения знаний и умений по предмету «Операционные системы». Содержит 7 лекций курса, в которых рассматриваются вопросы принципов построения современных операционных систем CP/M, ОС MS DOS, OS Windows 9x, ОС IBM OS/2 Warp, ОС Windows NT, ОС Unix.

Данное учебное пособие издается в соответствии с рабочей программой учебной дисциплины «Операционные системы» по Учебному плану для студентов II курса направления 09.03.01 очной формы обучения.

Рассмотрено и одобрено на заседании кафедры 19.05.15 г. и методического совета 19.05.15 г.

ББК 6Ф7.3
Св. тем. план 2017 г.
поз. 33

ЧЕРКАСОВА Наталья Ивановна

ОПЕРАЦИОННЫЕ СИСТЕМЫ
Тексты лекций

| | | |
|---|---|---|
| | Подписано в печать 01.02.2017 г. | |
| Печать офсетная | Формат 60x84/16 | 4,43 уч.-изд. л. |
| 4,85 усл.печ.л. | Заказ № 1725/133 | Тираж 30 экз. |

*Московский государственный технический университет ГА*
*125993 Москва, Кронштадтский бульвар, д.20*

*ООО «ИПП «ИНСОФТ»*
*107140, Москва, 3-й Красносельский переулок, д. 21, стр. 1*

ISBN 978-5-86311-988-5

## Содержание

**Лекция 10. Операционные системы семейства CP/M. Базовая система ввода-вывода. Базовая дисковая операционная система. Командный процессор. Операционная система MS DOS. Ядро ОС. Командные файлы**

Операционная система (ОС) CP/M положила начало созданию операционных систем для микроЭВМ. Она была разработана в 1974г. для 8-разрядных машин. В рамках этой ОС было создано программное обеспечение значительного объёма, включающее трансляторы с языков бейсик, паскаль, си, фортран, кобол, ада и других, текстовые и табличные процессоры, СУБД, графические пакеты и другие приложения.

Система предельно проста и компактна, имеет возможность настройки на разные конфигурации ЭВМ. Первая версия занимала всего 4 Кбайт, в дальнейшем система была реконструирована с целью обеспечения большей независимости от состава и параметров внешних устройств. В последних версиях этой ОС настройка на конкретную архитектуру машины обеспечивается с помощью специального файла, описывающего конфигурацию подключенных внешних устройств.

ОС CP/M постоянно хранится на диске и состоит из трех частей: базовой системы ввода/вывода (БСВВ), базовой дисковой операционной системы (БДОС) и командного процессора (КП), которые содержатся в специальных файлах с именами BIOS, BDOS и CPP. В начале работы указанные файлы загружаются в оперативную память, располагаясь при этом на старших адресах, оставляя младшие для программ пользователя. Требуемая для работы память невелика – в пределах 16 Кбайт.

Базовая система ввода/вывода.

БСВВ содержит в основном подпрограммы - драйверы внешних устройств. Это единственная часть ОС, которая требует настройки на конкретное оборудование. Таблицы, описывающие характеристики используемых устройств, формируются при генерации ОС, во время которой происходит перекомпиляция БСВВ с использованием файла конфигурации CONFIG.CPM. При установке нового устройства достаточно ввести в файл конфигурации соответствующую информацию, скомпилировать новый вариант БСВВ, и система оказывается готовой к работе.

Базовая дисковая операционная система.

БДОС содержит функции управления файловой системой и общего управления машиной (таких функций около 40). Файловая система довольно ограничена. Имена всех файлов хранятся в одном большом каталоге, который можно разделить на несколько нумерованных пользовательских областей.

Такая организация файловой системы не очень удобна, так как не позволяет структурировать хранимую информацию. Весьма ограничены возможности улавливания ошибочных ситуаций, диагностические сообщения очень лаконичны, что связано в основном с компактностью системы.

Командный процессор.

КП имеет средства для обработки лишь нескольких встроенных команд. Сюда относятся команды настройки на рабочий диск, команда USER – настройка на нового пользователя, DIR - выдача каталога диска, TYPE - вывод содержимого файла на дисплей, REN - переименование файла, ERA -.удаление файла. Это минимальный набор команд, которыми приходится пользоваться при работе с ОС. Другие команды реализуются в виде независимых программ, работающих под управлением ОС.

Некоторые из служебных программ поставляются вместе с системой. К ним относятся: программа PIP, обеспечивающая пересылку содержимого файлов, программа STAT, позволяющая просматривать и изменять атрибуты файлов, FORMAT - программа форматирования дисков и некоторые другие. Программы запускаются путем ввода имени программы и её параметров в ответ на «приглашение» системы. Имеется возможность объединения нескольких команд в одном файле и последующего исполнения такого командного файла. При этом исполняются все записанные в ней команды.

Система CP/M является однопользовательской, однопрограммной и 8-разрядной операционной системой, но послужила образцом для создания целого семейства ОС как для 8-, так и для 16-разрядных ПЭВМ. Многопользовательская версия этой системы называется MP/M, а для 16-разрядных ПЭВМ соответствующие версии называются CP/M-86 и MP/И-86. Эти версии по всем функциям похожи на CP/M и MP/M, но ориентированы на работу с процессором 8086. Ещё две версии этой системы – Concurrent CPM и Concurrent DOS – поддерживают многозадачный режим работы.

Операционные системы CP/M оказали заметное влияние на техническую политику многих разработчиков операционных систем и являются лучшими операционными системами для 8-разрядных компьютеров.

Операционная система MS DOS.

На 16-разрядных персональных компьютерах безусловно доминирующей является операционная система MS DOS (Microsoft Disk Operating System) или её аналог PC DOS (Personal Computer Disk Operating System). К основным её достоинствам, по сравнению с ОС CP/M, следует отнести следующее:

1) развитый командный язык;
2) возможность организации многоуровневых каталогов;
3) возможность работы со всеми последовательными устройствами как с файлами;
4) возможность подключение пользователем дополнительных драйверов внешних устройств;
5) возможность запуска фоновых задач одновременно с диалоговой работой пользователя, хотя MS DOS остается при этом однозадачной операционной системой.

Применительно как к аппаратной, так и к программной части компьютера широко используется понятие модуля. Модуль - унифицированная самостоятельная функциональная часть системы, имеющая законченное

оформление и средства сопряжения с другими функциональными узлами и модулями.

Структуру операционной системы MS DOS образуют модули:

1) BIOS (Basic Input/Output System) - базовая система ввода-вывода;
2) модуль расширения - EM BIOS (Extension Module BIOS) в виде файла с именем I0.SYS;
3) базовый модуль (BM - Basic Module) дисковой операционной системы (БДОС) виде файла с именем MSDOS.SYS;
4) командный процессор или интерпретатор команд (Command Interpreter) в виде файла с именем COMMAND.COM;
5) внешние команды и драйверы, утилиты - файлы с расширением .COM, .EXE. .SYS:
6) системный загрузчик (SB - System Bootstrap).

**Алгоритм загрузки операционной системы**

Операционная система хранится во внешней памяти обычно на жестком диске, реже - на гибком. Для нормальной работы компьютера необходимо, чтобы основные модули операционной системы находились в оперативной памяти. Поэтому после включения компьютера организована автоматическая перезапись (загрузка) операционной системы с диска в оперативную память. Наиболее важные аспекты этой загрузки отражены в виде алгоритма на рис. 1.

После успешно выполненной загрузки в оперативную память модуля расширения IO.SYS и базового модуля MSDOS.SYS загружается командный процессор COMMAND.COM и обрабатывается файл конфигурации CONFIG.SYS, который содержит команды подключения необходимых драйверов. Этот файл может отсутствовать, если пользователя устраивает базовый вариант операционной системы.

Затем выполняется обработка командного файла AUTOEXEC.BAT.

Если файлы конфигурации не найдены, компьютер загружает DOS в конфигурации, заданной по умолчанию, и не устанавливает никаких дополнительных драйверов. То же самое произойдёт, если DOS дано указание проигнорировать CONFIG.SYS.

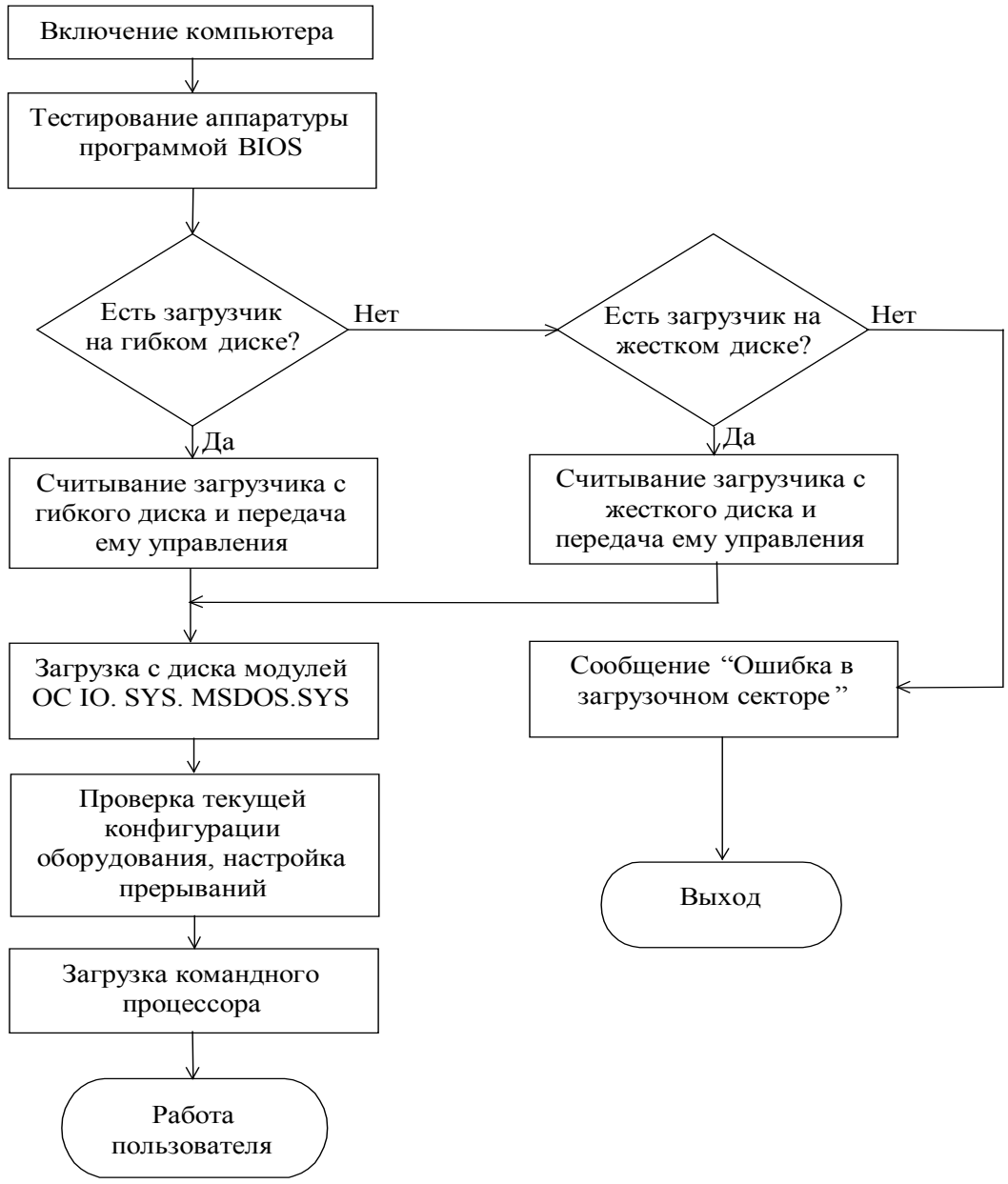


Рис. 1. Алгоритм загрузки операционный системы в оперативную память

**Функции и назначение базовой системы ввода-вывода BIOS**

Базовая система ввода-вывода BIOS - самый близкий к аппаратуре компонент DOS. BIOS находится в постоянной памяти, которая входит в комплект поставки персонального компьютера. Тип операционной системы может изменяться, а BIOS остается постоянным. Поэтому BIOS, являясь неизменяемой частью персонального компьютера, с одной стороны, может рассматриваться как компонент аппаратной части, а с другой стороны, как

компонент любой операционной системы, в том числе и MS DOS. Строго говоря, IOS не входит в состав MS DOS.

Основная функция BIOS реализуется в процессе нормальной работы персонального компьютера. Эта функция осуществляет прежде всего управление стандартными внешними (периферийными) устройствами, входящими в состав комплекта персонального компьютера конкретной модели, а именно: дисплеем, клавиатурой, дисководами, принтером, таймером. Выделение BIOS в отдельный аппаратно-программный модуль позволяет обеспечить независимость программного обеспечения от специфики конкретной модели персонального компьютера.

Вспомогательные функции BIOS реализуются при включении персонального компьютера на этапе загрузки и состоят в следующем:

- поиск сначала на гибком, а затем на жестком диске программы-загрузчика операционной системы и загрузка с диска в оперативную память;
- тестирование аппаратной части, в том числе и оперативной памяти, а при обнаружении неисправности индикация сообщения;
- инициализация векторов прерываний нижнего уровня – тех, которые требуют непосредственного управления аппаратными компонентами (этим прерываниям присвоены номера с 0 по 31).

BIOS содержит: специальные программы (драйверы) по управлению работой стандартными внешними устройствами; тестовые программы для контроля работоспособности аппаратуры; программу начальной загрузки операционной системы.

**Функции и назначение модуля расширения базовой системы ввода-вывода**

Модуль расширения базовой системы ввода-вывода EM BIOS придает гибкость операционной системе при обращении к внешним устройствам, а при необходимости и перекрывает (блокирует) функции постоянного модуля BIOS. Он хранится на диске в виде файла IO.SYS после главного каталога в заранее выделенном фиксированном месте. Объем этого файла небольшой, например, для MS DOS версии 6.22 он равен 40 Кбайт.

Наличие модуля расширения позволяет легко провести модификацию параметров операционной системы, используя файл конфигурации CONFIG.SYS, который хранится в главном каталоге. При подключении новых внешних устройств в этом файле указываются имена новых драйверов, управляющих их работой. Сами драйверы в виде файлов размещаются на диске. После загрузки операционной системы в оперативную память осуществляется поиск на диске файла CONFIG.SYS, где должны быть указаны необходимые драйверы, и модуль расширения осуществляет их подключение.

Основная функция модуля расширения в процессе нормальной работы компьютера - это увеличение возможностей BIOS.

Функции модуля расширения на этапе загрузки состоят в следующем:

определение состояния оборудования;

конфигурирование MS DOS по указаниям в файле CONFIG.SYS;
инициализация и переустановка некоторых векторов прерываний нижнего уровня.

Кроме рассмотренных выше функций, на модуль расширения возлагается задача завершения загрузки MS DOS в оперативную память. С этой целью модуль расширения сначала передаёт управление на загруженный к этому времени в память базовый модуль, называемый также модулем обработки прерываний, в котором устанавливаются внутренние рабочие таблицы, инициируются прерывания верхнего уровня и производится подготовка к загрузке командного процессора. После этого управление возвращается в модуль расширения, который производит загрузку командного процессора и передает ему управление.

**Функции и назначение базового модуля дисковой операционной системы**

Базовый модуль располагается в виде файла MSDOS.SYS на системном диске в специально выделенном для него месте вслед за файлом модуля расширения IO.SYS. Объем файла MSDOS.SYS для версии 6.22 - 38 Кбайт. Базовый модуль не имеет жесткой привязки к аппаратной части, и при необходимости может быть заменён на другой файл.

Основная функция базового модуля в прицеле нормальной работы компьютера - управление ресурсами компьютера, файловой системой на дисковом пространстве и управление работой программ при помощи системы прерываний.

Функциями базового модуля на этапе загрузки являются: считывание в память и запуск командного процессора, инициализация векторов прерываний верхнего уровня (табл. 1.).

Таблица 1

ПРЕРЫВАНИЯ MS-DOS

| Прерывание | Описание |
|---|---|
| Int20H | ОБЩЕЕ ПРЕРЫВАНИЕ ПРОГРАММЫ. |
| Int21H | ДИСПЕТЧЕР ФУНКЦИЙ MS-DOS. |
| Int22H | АДРЕС ПРЕРЫВАНИЯ ПРОГРАММЫ |
| Int23H | АДРЕС Ctrl-Break. |
| Int24H | ССЫЛКА НА УПРАВЛЯЮЩУЮ ЗАПИСЬ КРИТИЧЕСКОЙ ОШИБКИ. |
| Int25H | АБСОЛЮТНОЕ ЧТЕНИЕ ДИСКА. |
| Int26H | АБСОЛЮТНАЯ ЗАПИСЬ НА ДИСК. |
| Int27H | ЗАВЕРШЕНИЕ ПРОГРАММЫ, ОСТАВЛЯЮЩЕЕ ЕЁ РЕЗИДЕНТНОЙ |
| Int30H-3FH | РЕЗЕРВИРУЮТСЯ ОПЕРАЦИОННОЙ СИСТЕМОЙ |

В табл. 1 приведен общий перечень прерываний DOS. При этом следует отметить, что за одним из прерываний (десятичный номер 33, шестнадцатеричный – 21) скрывается множество функций или операций DOS по обслуживанию стандартных устройств и файловой системы.

Деление сервисных функций DOS на два уровня было обусловлено воображением модульности и развития системы. Кроме того, значительная часть прерываний верхнего уровня также была зарезервирована для развития DOS.

Функции, реализуемые данным модулем DOS, в первую очередь используются командами системы (обрабатываемыми командным процессором), но могут вызываться и прикладными программами.

**Функции и назначение командного процессора**

Командный процессор, иногда называемый процессором консольных команд, предназначен для поддержки пользовательского интерфейса DOS. Он представляет собой обычный файл COMMAND.COM и располагается на системном диске в любом месте пространства, выделенного под файлы. Так, для MS DOS версии 6.22 объем COMMAND.COM равен 55 Кбайтам.

Командный процессор состоит из двух модулей - резидентного и транзитного.

Резидентный модуль хранится в оперативной памяти постоянно после загрузки операционной системы. Транзитный модуль может вытесняться из оперативной памяти на диск прикладной программой, если ей для работы не хватает памяти. После окончания работы такой программы транзитный модуль вновь восстанавливается на прежнем месте оперативной памяти путем считывания его с диска. Транзитный модуль содержит исполнитель внутренних команд и загрузчик программ в оперативную память для выполнения. Взаимодействие с командным процессором осуществляется при помощи команд. Под командой понимается указание на выполнение некоторого действия. Команды бывают двух типов: резидентные (внутренние) и транзитные (внешние). Резидентные команды входят в состав самого командного процессора. Транзитные команды являются файлами типа EXE или COM, входящими в состав операционной системы DOS и хранящимися в обычном каталоге, как правило, с именем DOS.

Основные функции командного процессора в процессе нормальной работы компьютера состоят в следующем:

1) приём и анализ команд, введенных с клавиатуры или из командного файла;
2) выполнение внутренних команд;
3) загрузка программ в память для выполнения;
4) обработка прерываний по завершении задачи.

Основная функция командного процессора на этапе загрузки - это выполнение файла автонастройки AUTOEXEC.BAT.

При нормальном функционировании операционной системы командный процессор выдает на экран приглашение к работе, например C:\. В ответ на это приглашение вы вводите имя программы или команды, а командный процессор расшифровывает символы введенного имени и продолжает работу по одному из следующих вариантов:

1) в случае резидентной команды он сразу приступает к ее выполнению;
2) в случае транзитной команды или любой другой программы он загружает ее в оперативную память, подключая для этого два других модуля операционной системы: базовый модуль БДОС и модуль расширения BIOS, и передает этой программе или команде управление.

После окончания работы введенной команды (программы) управление вновь возвращается командному процессору.

Таким образом, исходя из изложенного выше, все виды обращений к DOS можно разделить на три типа:

1) обращение из программ по прерываниям 0 – 31 через BIOS, модуль расширения BIOS или блок начальной загрузки;
2) обращение из программ по прерываниям 32 – 63 через базовый модуль БДОС;
3) команды пользователя и командные файлы через командный процессор.

**Назначение загрузчика**

Загрузчик BOOT RECORD (модуль начальной загрузки) всегда размещается на диске в нулевом секторе и занимает объем 512 байт. Основное назначение этой небольшой программы состоит в поиске и перезаписи (загрузке) с диска в оперативную память двух файлов - IO.SYS и MSDOS.SYS. Поиск этих модулей и их загрузка в оперативную память осуществляются в определенном порядке, поэтому на диске и в оперативной памяти они занимают фиксированное место и следуют один за другим. Если блок начальной загрузки не обнаружит этих модулей на диске, то он выдает соответствующее сообщение и работа компьютера приостанавливается. Кроме того, функцией загрузчика является запуск модуля расширения BIOS.

**Утилиты, внешние команды и драйверы**

Утилиты, внешние команды н драйверы представляют собой программы, хранящиеся во многих случаях в каталоге системного диска в виде файлов типа .COM, .EXE, .SYS. Внешнее различие между ними весьма условное, и связывают его с интерфейсом взаимодействия с пользователем.

Внешней командой принято считать программу, выдающую пользователю ряд простых запросов или выполняющуюся автоматически без специально организованного интерфейса с пользователем. MS DOS имеет определенный перечень внешних команд.

Внешние драйверы, как правило, выполняются без диалога и поставляются отдельно от MS DOS либо совместно с внешним устройством, либо самостоятельно.

Утилиты - обслуживающие программы, которые предоставляют пользователю сервисные услуги. Они, как правило, имеют полноэкранный, организованный в виде меню интерфейс взаимодействия с пользователем. Реже интерфейс организован в виде запросов.

## Командные файлы

Операционная система в своем арсенале инструментальных программных средств содержит специальный программный механизм для автоматизации работы с командами. Он позволяет не только облегчить работу программиста за счет автоматизации часто повторяющейся совокупности команд, но и освободить пользователя от необходимости знания многих тонкостей форматов команд.

Последовательность автоматически выполняемых операций обработки в операционной системе получила название пакетной обработки (batch processing). Инструментальным средством пакетной обработки является командный (пакетный) файл.

Командный файл - файл, позволяющий автоматизировать работу в операционной системе. Понятие "командный файл" используется очень широко. Практически во всех прикладных программных средах вы найдете соответствующий программный инструментарий для его создания. Однако там его называют иначе, например макрос. Различие в названиях появилось для того, чтобы подчеркнуть прикладную, а не системную сферу воздействия макроса, а также потому, что он состоит из команд, действующих только в конкретной прикладной программной среде.

Командный файл, работающий в операционной среде MS DOS, имеет тип .BAT (от англ. Batch - пачка). Макрос имеет тип, который определяет его принадлежность к определенной прикладной среде.

Командный файл создается как текстовый файл в любом текстовом редакторе. Сам текст представляет собой последовательность конструкций команд операционной системы, имен файлов запуска прикладных систем, различных сервисных утилит. Запускается командный файл на выполнение так же, как и команды MS DOS или файлы запуска прикладных программных систем, имеющие тип .COM или .EXE.

## Основные свойства командного файла

Командный файл состоит из команд операционной системы MS DOS, имен файлов запуска, сервисных программных средств и команд, специально созданных для управления работой командного файла. Каждая команда занимает отдельную строку.

Имя командного файла - уникальное в пределах того каталога, где он находится. Тип - всегда .BAT.

В конструкции команд могут быть как строчные, так и прописные буквы.

В командном файле используются любые команды операционной системы и ее сервисного окружения. Кроме того, имеется ряд команд, специально созданных для управления работой командного файла. Командный файл часто входит в

состав пакета прикладных программ и используется при его загрузке для того, чтобы автоматизировать процесс настройки на конкретного пользователя.

**Выводы**

1. Система CP/M является однопользовательской, однопрограммной и 8-разрядной операционной системой, с возможностью настройки на разные конфигурации ЭВМ, и служит образцом для создания целого семейства ОС как для 8-, так и для 16-разрядных ПЭВМ.
2. Операционная система MS DOS является однозадачной 16-разрядной ОС, к основным особенностям которой следует отнести следующее:
   1) развитый командный язык;
   2) возможность организации многоуровневых каталогов;
   3) возможность работы со всеми последовательными устройствами как с файлами;
   4) возможность подключение пользователем дополнительных драйверов внешних устройств;
   5) возможность запуска фоновых задач одновременно с диалоговой работой пользователя.
3. Основным механизмом функционирования MS DOS является система прерываний.
4. Все виды обращений к DOS можно разделить на три типа:

- обращение из программ по прерываниям 0–31 через BIOS, модуль расширения BIOS или блок начальной загрузки;
- обращение из программ по прерываниям 32 – 63 через базовый модуль БДОС;
- команды пользователя и командные файлы через командный процессор.

## Лекция 11. Система прерываний – основной механизм функционирования ОС MS DOS. Особенности обработки аппаратных прерываний. Маскирование прерываний

Основным механизмом функционирования MS DOS является система прерываний. Рассмотрим его подробно.

Программы могут сами вызывать прерывания с заданным номером. Для этого они используют команду INT. Программные прерывания не являются асинхронными, так как вызываются из программы. Программные прерывания удобно использовать для организации доступа к отдельным, общим для всех программ модулям. Например, программные модули операционной системы доступны прикладным программам именно через прерывания, и нет необходимости при вызове этих модулей знать их текущий адрес в памяти. Прикладные программы могут сами устанавливать свои обработчики прерываний для их последующего использования другими программами. Для этого встраиваемые обработчики прерываний должны быть резидентными в памяти.

Аппаратные прерывания вызываются физическими устройствами и происходят асинхронно. Эти прерывания информируют систему о событиях, связанных с работой устройств.

Некоторые прерывания (первые пять в порядке номеров) зарезервированы для использования самим центральным процессором на случай каких-либо особых событий вроде попытки деления на ноль, переполнения и т.п.

Иногда желательно сделать систему нечувствительной ко всем или отдельным прерываниям. Для этого используют так называемое маскирование прерываний. Но некоторые прерывания замаскировать нельзя, это немаскируемые прерывания.

Обработчики прерываний могут сами вызывать программные прерывания, например, для получения доступа к сервису BIOS или DOS (сервис BIOS также доступен через механизм программных прерываний).

Составление собственных программ обработки прерываний и замена стандартных обработчиков DOS и BIOS является ответственной и сложной работой. Необходимо учитывать все тонкости работы аппаратуры и взаимодействия программного и аппаратного обеспечения. При отладке возможно разрушение операционной системы с непредсказуемыми последствиями.

В IBM-совместимом компьютере имеется 256 различных прерываний с номерами от 0 до OFFH (номера представлены в виде шестнадцатеричных цифр). Для того чтобы связать адрес обработчика прерывания с номером прерывания, используется таблица векторов прерываний, занимающая первый килобайт оперативной памяти - адреса от 0000:0000 до 0000:03FF. Таблица состоит из 256 элементов - FAR-адресов обработчиков прерываний. Эти элементы называются векторами прерываний. В первом слове элемента таблицы записано смещение, а во втором - адрес сегмента обработчика прерывания.

Прерыванию с номером 0 соответствует адрес 0000:0000, прерыванию с номером 1 - 0000:0004 и т.д. Для программиста, использующего язык Си, таблицу можно описать следующим образом:

```
void (* interrupt_table[256])();.
```

При возникновении любого прерывания значения регистра флагов процессора и текущее значение счетчика команд CS:IP автоматически сохраняются в стеке, прерывания запрещаются, и выполняется переход по вектору прерывания.

Векторная система прерываний обеспечивает большую гибкость: программист всегда имеет возможность переключить любое прерывание на свою процедуру. Для этого необходимо только изменить в памяти соответствующий вектор так, чтобы он на нее указывал. Это называется перехватом прерывания.

Аппаратные прерывания относятся к прерываниям низшего уровня, им присвоены младшие номера, и обслуживает их базовая система ввода-вывода.

Логические и программные прерывания относят к верхнему уровню, они имеют большие номера, и их обслуживает в основном базовый модуль DOS.

**Маскирование прерываний**

Часто при выполнении критических участков программ, для того чтобы гарантировать выполнение определенной последовательности команд целиком, приходится запрещать прерывания. Это выполняется командой CLI. Она помещается в начало критической последовательности команд, а в конце располагается команда STI, разрешающая процессору воспринимать прерывания. Команда CLI запрещает только маскируемые прерывания, немаскируемые всегда обрабатываются процессором.

Прерывания нельзя отключать на длительный период времени, так как это может привести к нежелательным последствиям. Например, будут отставать часы. Если запрещаются не все прерывания, а только некоторые, например, от клавиатуры, то осуществляет их контроллер прерываний.

Прикладная программа может изменить программу обработки прерывания. Для этого программа должна переназначить вектор на свой обработчик. Это можно сделать, изменив содержимое соответствующего элемента таблицы векторов прерываний.

Очень важно не забыть перед завершением работы восстановить содержимое измененных векторов в таблице прерываний, т.к. после завершения работы программы память, которая была ей распределена, считается свободной и может быть использована для загрузки другой программы.

Кроме того, операция изменения вектора прерывания должна быть непрерывной в том смысле, что во время изменения не должно произойти прерывание с номером, для которого производится замена программы обработки.

Для облегчения работы по замене прерывания DOS предоставляет специальные функции для чтения элемента таблицы векторов прерывания и для записи в нее нового адреса. Рассмотрим возможности библиотеки C, предназначенные для работы с прерываниями.

Модификатор interrupt описывает функцию, которая является обработчиком прерывания. Такая функция завершается командой возврата из обработки прерывания IRET, и для нее автоматически генерируются команды сохранения регистров на входе и их восстановления при выходе из обработчика прерывания. Пример использования модификатора для описания функции обработки прерывания:

```
void interrupt far int_funct(void) {
    // Тело обработчика прерывания
}
```

Функция обработки прерывания должна быть FAR-функцией, т.к. таблица векторов прерываний содержит полные адреса в виде сегмент:смещение.

Ключевое слово interrupt используется также для описания переменных, предназначенных для хранения векторов прерываний:
void (_interrupt _far *oldvect)(void);
Модификаторы _interrupt и _far для Quick C 2.5 и C 6.0 являются синонимами соответственно interrupt и far.
Какие требования необходимо предъявлять к программе обработки прерывания?
Если прерывания происходят часто, то их обработка может сильно замедлить работу прикладной программы. Поэтому обработчик прерывания должен быть короткой, быстро работающей программой, которая выполняет только самые необходимые действия. Например, считать очередной символ из порта принтера и поместить его в буфер, увеличить значение какого-либо глобального счетчика прерываний и т.п.
Для установки своего обработчика прерываний используется функция setvect(). Эта функция имеет два параметра - номер прерывания и указатель на новую функцию обработки прерывания. Например:
setvect(0x16, my_key_intr);
В этом примере для клавиатурного прерывания с номером 16h устанавливается новый обработчик прерывания my_key_intr.
Если вам надо узнать адрес старого обработчика прерывания по его номеру, лучше всего воспользоваться функцией _dos_getvect, которая принимает в качесте параметра номер прерывания и возвращает указатель на соответствующий этому номеру в таблице векторов прерываний обработчик. Например:
old_vector = _dos_getvect(0x16);
Для организации цепочки прерываний используется функция _chain_intr. В качестве параметра эта функция принимает адрес старого обработчика прерываний.

В листинге 1 представлена программа, иллюстрирующая применение всех трех функций, предназначенных для работы с прерываниями. Эта программа встраивает собственный обработчик прерывания таймера, который будет вызываться примерно 18,2 раза в секунду. Встраиваемый обработчик прерывания считает тики таймера и, если значение счетчика кратно 20, на динамик компьютера выдается звуковой сигнал. В конце работы новая программа обработки прерывания таймера вызывает старый обработчик с помощью функции _chain_intr.

## Особенности обработки аппаратных прерываний

Аппаратные прерывания вырабатываются устройствами компьютера, когда возникает необходимость их обслуживания. Например, по прерыванию таймера соответствующий обработчик прерывания увеличивает содержимое ячеек памяти, используемых для хранения времени. В отличие от программных прерываний, вызываемых запланировано самой прикладной программой,

аппаратные прерывания всегда происходят асинхронно по отношению к выполняющимся программам. Кроме того, может возникнуть одновременно сразу несколько прерываний!

Для обслуживания прерываний существует специальная схема приоритетов. Каждому прерыванию назначается свой уникальный приоритет. Если происходит одновременно несколько прерываний, то система отдает предпочтение самому высокоприоритетному, откладывая на время обработку остальных прерываний.

Система приоритетов реализована на двух микросхемах Intel 8259 (для машин класса XT - на одной такой микросхеме). Каждая микросхема обслуживает до восьми приоритетов. Микросхемы можно объединять (каскадировать) для увеличения количества уровней приоритетов в системе.
Уровни приоритетов обозначаются сокращенно IRQ0 - IRQ15 (для машин класса XT существуют только уровни IRQ0 - IRQ7).
Для машин XT приоритеты линейно зависели от номера уровня прерывания. IRQ0 соответствовало самому высокому приоритету, за ним шли IRQ1, IRQ2, IRQ3 и так далее. Уровень IRQ2 в машинах класса XT был зарезервирован для дальнейшего расширения системы и, начиная с машин класса AT, IRQ2 стал использоваться для каскадирования контроллеров прерывания 8259. Добавленные приоритетные уровни IRQ8 - IRQ15 в этих машинах располагаются по приоритету между IRQ1 и IRQ3.
В табл. 2 приведены аппаратные прерывания, расположенные в порядке приоритета.

Таблица 2

Аппаратные прерывания

| Номер | Описание |
|---|---|
| 8 | IRQ0 - прерывание интервального таймера, возникает 18,2 раза в секунду |
| 9 | IRQ1 - прерывание от клавиатуры. Генерируется при нажатии и при отжатии клавиши. Используется для чтения данных с клавиатуры. |
| A | IRQ2 - используется для каскадирования аппаратных прерываний в машинах класса AT |
| 70 | IRQ8 - прерывание от часов реального времени |
| 71 | IRQ9 - прерывание от контроллера EGA |
| 72 | IRQ10 - зарезервировано |
| 73 | IRQ11 - зарезервировано |
| 74 | IRQ12 - зарезервировано |

Продолжение табл. 2

| | |
|---|---|
| 75 | IRQ13 - прерывание от математического сопроцессора |
| 76 | IRQ14 - прерывание от контроллера жесткого диска |
| 77 | IRQ15 - зарезервировано |
| B | IRQ3 - прерывание асинхронного порта COM2 |
| C | IRQ4 - прерывание асинхронного порта COM1 |
| D | IRQ5 - прерывание от контроллера жесткого диска для XT |
| E | IRQ6 - прерывание генерируется контроллером флоппи-диска после завершения операции |
| F | IRQ7 - прерывание принтера. Генерируется принтером, когда он готов к выполнению очередной операции. Многие адаптеры принтера не используют это прерывание |

Из табл. 2 видно, что самый высокий приоритет у прерываний от интервального таймера, затем идет прерывание от клавиатуры.

Для управления схемами приоритетов необходимо знать внутреннее устройство контроллера прерываний 8259. Поступающие прерывания запоминаются в регистре запроса на прерывание IRR. Каждый бит из восьми в этом регистре соответствует прерыванию. После проверки на обработку в настоящий момент другого прерывания запрашивается информация из регистра обслуживания ISR. Перед выдачей запроса на прерывание в процессор проверяется содержимое восьмибитового регистра маски прерываний IMR. Если прерывание данного уровня не замаскировано, то выдается запрос на прерывание.

Наиболее интересными с точки зрения программирования контроллера прерываний являются регистры маски прерываний IMR и управляющий регистр прерываний.

В машинах класса XT регистр маски прерываний имеет адрес 21h, управляющий регистр прерываний - 20h. Для машин AT первый контроллер 8259 имеет такие же адреса, что и в машинах XT, регистр маски прерываний второго контроллера имеет адрес A1h, управляющий регистр прерываний - A0h.

Разряды регистра маски прерываний соответствуют номерам IRQ. Для того чтобы замаскировать аппаратное прерывание какого-либо уровня, надо заслать в регистр маски байт, в котором бит, соответствующий этому уровню, установлен в 1. Например, для маскирования прерываний от НГМД в порт 21h надо заслать двоичное число 01000000.

**Выводы**

1. Аппаратные прерывания вызываются физическими устройствами и происходят асинхронно.
2. Программные прерывания не являются асинхронными, так как вызываются из программы.
3. Для того чтобы связать адрес обработчика прерывания с номером прерывания, используется таблица векторов прерываний.
4. Для обслуживания прерываний существует специальная схема приоритетов. Каждому прерыванию назначается свой уникальный приоритет.

## Лекция 12. Операционная система IBM OS/2. Состав и особенности IBM OS/2 Warp. Интерфейс Shell. Приложения и объекты IBM OS/2 Warp. Совместимость с MS DOS и MS Windows. Сетевые средства OS/2

Первая версия операционной системы IBM OS/2 Standart Edition 1.0 появилась в 1987 году для замещения связки MS DOS и Windows 3.11 в связи с возможностью использования большей оперативной памяти и для устранения причин зависания системы. Через полгода версия Extended Edition 1.0 была дополнена Менеджером коммуникаций и Менеджером баз данных, а в версии Extended Edition 1.1 появился графический интерфейс Presentation Manager. При этом OS/2 стала использоваться в качестве ОС для машин, занимающих промежуточное положение между ПК и большими базовыми ЭВМ. С её помощью осуществлялись удалённый доступ к базам данных, поддержка локальных сетей, подключение к крупным и средним ЭВМ и т. д.

В 1992 году с появление версии OS/2 2.0 система вышла на качественно другой уровень. Устаревшая Presentation Manager была заменена графической оболочкой Workplase Shell или просто WPS. Причем здесь новым является прежде всего способ работы с компьютером – объектно-ориентированный подход.

В 1994 году система была коренным образом переписана и получила собственное имя Warp. Именно эта система и является операционной системой OS/2, используемой и в настоящее время с различными модификациями.

Основные особенности ОС OS/2 Warp:

1. ОС OS/2 Warp является 32- разрядной ОС.
2. Снимаются ограничения на длину имён файлов и количества памяти, доступной для приложений.
3. ОС даёт возможность приложениям выполняться параллельно через механизм вытесняющей многозадачности и многопоточности.
4. Наряду с FAT используется файловая система HPFS, которая автоматически дефрагментирует файлы.
5. Включены меры защиты от программ, разрушающих другие программы или систему.

6. ОС делает ПК быстрее и дружественнее, чем ОС DOS, задействовав для этого графику, управление мышью, меню, окна, объекты и другие механизмы.
7. Обеспечивается большой набор функций и услуг для программистов, разрабатывающих новые приложения.

Все перечисленные свойства относятся как к ОС OS/2 Warp, так и к предыдущим версиям операционной системы, однако, ОС OS/2 Warp имеет ряд дополнительных возможностей и некоторые особенности. К ним прежде всего относятся следующие:

1) оболочка отображает 256 цветов вместо16 и имеет больше пиктограмм;
2) панель запуска заменена на инструментальную панель;
3) в ОС включена система распознавания речи VoiceType, которая даёт возможность перемещаться по ОС, а также отдавать речевые команды, хотя, безусловно, данная система далека от совершенства;
4) в ОС введен инструментальный комплекс Java Development Kit и транслирующий исполняемый модуль, позволяющий программам Java работать в среде ОС независимо от браузера;
5) в состав ОС включена новая версия браузера, совместимого с Java;
6) к основным архитектурным усовершенствованиям относится появление асинхронной очереди сообщений, которая избавляет ОС от необходимости ожидать сообщений от конкретной программы, что иногда приводит к зависанию системы.

Итак, как было отмечено ранее, ОС обладает повышенной надежностью, жестко контролируя системные ресурсы и процессы. Она даёт корректное разделение их между прикладными задачами и обрабатывает все исключительные и аварийные ситуации. В случае возникновения подобных ситуаций система старается локализовать источник потенциальной опасности и защитить системные процессы и установки, более того, ОС запускает аппарат, позволяющий корректно выйти из критической ситуации. В этом случае она сама запустит системный процесс закрытия файлов данных и аварийного завершения исполняемых программ. Существует также возможность прервать аварийное выполнение любой программы, попросту перезагрузив систему.

В тоже время, несмотря на жесткость работы, ОС весьма гибко настраивается на конкретную задачу. Весьма велики и возможности оптимизации. Здесь используется собственная система управления памяти и улучшенная система виртуальной памяти. Широко применяется кэширование жесткого диска. Для ускорения доступа к внешним устройствам и облегчения обработки ошибок при обращении к ним эти устройства виртуализируются. Системные процессы сделаны максимально возможно малозатратными, в целях экономии процессорного времени. Введен режим пассивности, т.е. загрузка процессора резко уменьшается при уменьшении активности прикладных программ.

## Интерфейс

ОС ориентируется на ввод с помощью мыши, но можно работать лишь с клавиатурой. В системе используется принцип окон и пиктограмм. WPS – рабочий стол – применяется для достижения практически всех механизмов ОС. Он позволяет использовать механизм транспортировки (drag and drop), т.е. перетаскивать мышью практически всё и везде. Следует отметить, что объекты не просто программы – они разработаны для взаимодействия друг с другом, поэтому открыть объект на рабочем столе из командной строки невозможно.

Для запуска наиболее часто используемых функций системы и объектов при помощи кнопок используется специальный объект на рабочем столе – настраиваемая полоса инструментов (Launch Pad или WarpCenter для OS\2 Warp 4.0). По умолчанию там находятся пиктограммы завершения (Shutdown), поиск объектов (Find object), блокировка системы (Locup), справочная система (Assistance Center) и другие.

Система управления файлами проста. Выбранные из какого-либо каталога файлы располагаются в открывшемся окне как набор «понятных» значков, соответствующих их характеристикам. Некоторые основные объекты помещаются в рабочую область непосредственно при установке системы, другие группируются в специальные папки – фолдеры, которые в свою очередь могут объединяться в другие фолдеры, создавая тем самым иерархическую структуру. В системном фолдере содержатся фолдеры, относящиеся к файлам дисков, работе различных сессий, набору вспомогательных программ, средствам конфигурации и т.д., например фолдер Prodactiviti содержит информацию о вспомогательных программах и играх. К основным фолдерам относится также информационный, в котором содержится вся документация, обучающая программа и справочник по системным командам.

Для поддержки конкретной работы и внешнего вида системы используется небольшое количество ключевых файлов. Файл Config.sys содержит информацию о настройке системы и подключённых устройствах. Редактирование данного файла производится с помощью System Editor. Все пакетные файлы имеют расширение “.cmd “, а не “.bat”, хотя последние можно переименовать для работы в OS\2. Файл Startup.cmd выполняет ту же функцию, что и файл автозапуска в DOS. Файлы “.ini” используются для хранения информации о системных настройках (основные – Os2.ini и Os2sys.ini), но не могут редактироваться непосредственно, т.к. не являются базированными на тексте.

ОС создаёт виртуальную память, используя файл Swapper.dat.
Когда система нуждается в памяти, а доступной нет, она будет выгружать информацию в данный файл. Изменение размера и расположения файла Swapper.dat управляется оператором SWAPPATH файла Config.sys. В последних версиях ОС основные системные файлы могут помещаться в Swapper.dat, тогда как прежде они хранились в оперативной памяти. Кроме этого 5 системных DLL( Dynamic Link Library) всегда выгружаются в

Swapper.dat. Наличие библиотек динамической компоновки представляет ряд преимуществ разработчикам программного обеспечения для ОС. В частности, они позволяют повторное использование кода в приложениях. Для использования общих функций в различных приложениях необходимо поместить их в DLL и ОС при выполнении будет хранить в памяти только одну копию кода.

Кроме утилит, входящих в состав ОС, каждая копия системы включает коллекцию приложений BonusPak. BonusPak отделён от базовой системы для того, чтобы фирма могла обновлять каждую часть системы. Примерный перечень приложений – следующий:

1. FaxWorks – пакет для посылки и приёма факсов.
2. HyperACCESS Lite – коммуникационная программа.
3. CompuServe Information Manager – программа для доступа к CompuServe.
4. IBM Internet Connection – пакет приложений для Internet.
5. IBM Works and Personal Information Manager – пакет деловых приложений, включающий текстовой процессор, электронные таблицы, картографическую программу и другие.
6. Multimedia Viewer – программа для просмотра ауди- и видео- файлы, поддерживающая множество форматов.
7. System Information Tool – программа для сбора информации об установленном аппаратном обеспечении.
8. Person to Person – программа совместного использования по коммуникационной связи информационных буферов.
9. Video IN – программа для записи видео-сегментов.

## Архитектура OS\2 Warp 4.0.

Разработчики ОС OS\2 поставили стабильность системы выше производительности. Модель защиты, принятая в Intel, позволяет программам работать на одном из 4 уровней привилегий, называемых кольцами. Самый высокий уровень привилегий имеет кольцо 0, самый низкий – кольцо 3. Отметим, что OS\2 Warp 4.0 ориентирована исключительно на процессоры Intel, т. к. в системе отсутствует фундаментальное микроядро или уровень аппаратной абстракции, обеспечивающий интерфейс между ОС и аппаратными средствами.

Аппаратные средства не дают возможности программам, работающим в менее привилегированных кольцах, переписать содержимое участков памяти, контролируемых более привилегированными программами.

Кольцо 0 обычно называют режимом ядра, и именно здесь в основном работает ОС и все драйверы устройств. Все программы, поставляемые с ОС, работают в кольце 0 или кольце 3, причем прикладные программы выполняются в кольце 3, где им выделяются отдельные, защищённые виртуальные адресные пространства. Отсюда они обращаются к службам ОС, размещённым в кольце 0. Однако переходы через границы колец сопряжены с непроизводительными

расходами, поэтому авторы некоторых ОС предпочитают перенести часть системных программ в кольцо 3, ближе к прикладным программам. Это повышает производительность, но ставит под угрозу целостность системы. В OS\2 реализованы компромиссные решения: основная часть, в том числе ядро, работает исключительно в кольце 0, но некоторые подпрограммы, в том числе пользовательский интерфейс, работают в кольце 3 ради повышения производительности. Поэтому в WPS предусмотрена возможность повторной самоперезагрузки. Shell перезапускает себя без участия пользователя, и, обычно, все программы и окна возобновляют свою работу с того места, где они были прерваны.

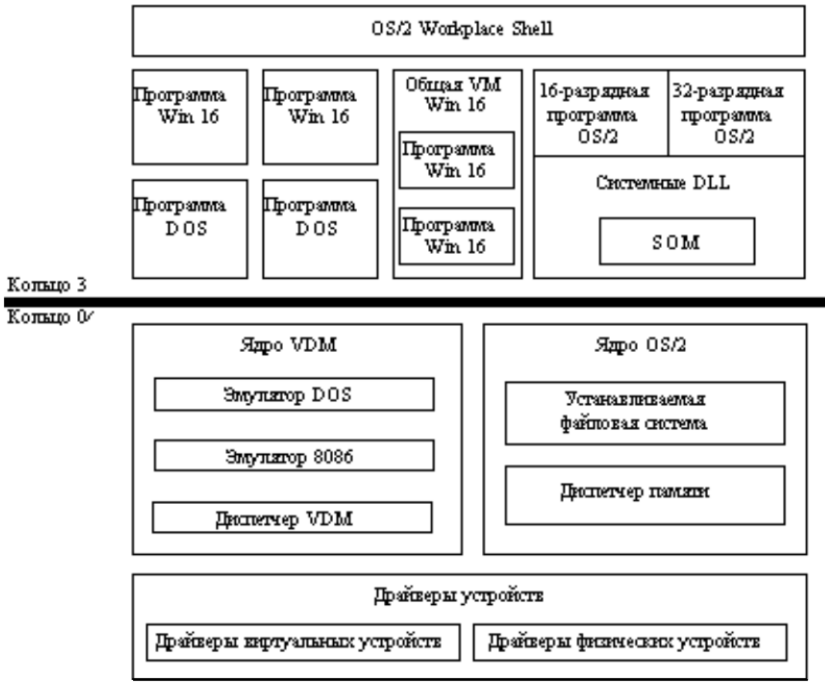


Рис. 2. Архитектура ОС OS\2 Warp 4.0

На рис. 2 представлена структура ОС OS\2 Warp. Система содержит множество видов VM (виртуальных машин) для прикладных программ. 16- и 32-разрядные программы OS\2 выполняются на отдельных VM в режиме приоритетной многозадачности и могут общаться между собой. Прикладные программы Dos и Win16 могут запускаться на отдельных VM в многозадачном режиме. Причем последние на отдельных VM поддерживают полноценные связи друг с другом и с 32-разрядными программами OS\2. Кроме того, можно запускать несколько программ Win16 на общей VM Win16 в режиме коллективной многозадачности. Разнообразные сервисные функции обеспечиваются с помощью системных библиотек, к которым можно обращаться без требующих затрат времени переходов между кольцами. Ядро системы представляет многие базовые сервисные функции, обеспечивает поддержку файловой системы, управление памятью и имеет Диспетчер

аппаратных прерываний. Диспетчер инсталлируемой файловой системы (IFS) теоретически позволяет любой прикладной программе работать с любой файловой системой.

В ядре виртуальных Dos-машин осуществляется эмуляция Dos и процессора 8086. Драйверы виртуальных устройств обеспечивают уровень аппаратной абстракции, а драйверы физических устройств напрямую взаимодействуют с аппаратурой.

## Сетевые средства OS/2

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 версии OS/2 - Warp Connect входят редиректоры для операционных систем NetWare 3.x и 4.1 и OS/2 LAN Server. Версия OS/2 Warp Connect работает с протоколами IPX и NetBIOS, а также реализацией протоколов TCP/IP. Этот комплект устанавливает двухточечное соединение по протоколу PPP вместо соединений SLIP, предусмотренных в базовом пакете OS/2 Warp. Этот комплект понижает нагрузку на центральный процессор и обеспечивает одновременный доступ к локальной сети и сети Internet.

Кроме того, Warp Connect предоставляет средства одноранговой сетевой связи. В эту версию входит большое число собственных драйверов, которые работают 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 в серверу LAN Server 4.0.

Warp Connect содержит также программу Lan Distance фирмы IBM, которая позволяет соединяться через связной сервер с любым подключенным к сети устройством. ОС Warp Connect не содержит 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 поставляетсяся в двух версиях: одна без Windows-библиотек, другая, подобно Full Pack, с библиотеками Win-OS/2.

На рис. 3 изображены сетевые средства OS/2 Warp Connect. Они делятся на четыре уровня. Прикладной уровень включает программные интерфейсы приложений операционной системы. Компоненты на уровне файловой системы отвечают за выполнение файловых операций. Транспортный уровень реализует коммуникационные протоколы. Имеется компонента Общая транспортная семантика (Common Transport Semantic), которая позволяет использовать любую файловую систему (а точнее ее редиректор) в сочетании с любым протоколом транспортного уровня.

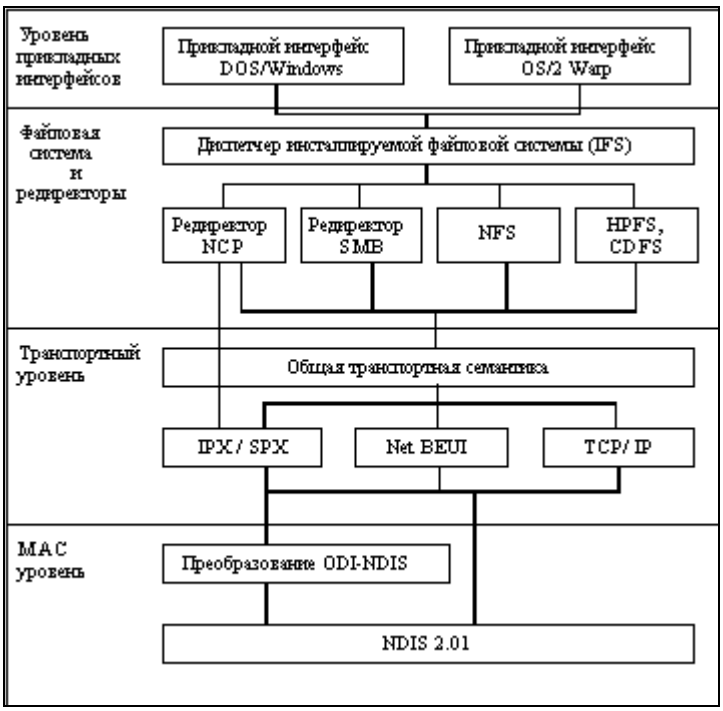


Рис. 3. Структура сетевых средств OS/2 Warp Connect

Программное обеспечение MAC-уровня включает драйверы сетевых адаптеров и диспетчерский слой в стандарте NDIS 2.01, который позволяет различным сетевым протоколам работать через один адаптер, и различным адаптерам связываться через общий протокол. Существует модуль преобразования ODI-NDIS, который позволяет использовать модули транспортных протоколов, реализованные в расчете на работу с диспетчерским слоем ODI компании Novell.

**Совместимость с DOS и MS Windows**

16-разрядные приложения для DOS и MS Windows могут выполняться в системе в их естественном виде, рядом с программами OS\2 в рамках одного «рабочего стола». Однако некоторым Dos-программам из соображений безопасности ОС не позволит работать. К таким программам относятся антивирусные программы, диагностические программы, программы управления памятью и другие. Для таких программ в версию 4.0 введена новая возможность, получившая название True Mode (истинный режим). Доступный только в тех системах, где ОС установлена поверх Dos, истинный режим даёт возможность системе автоматически сохранить на диске содержимое памяти, загрузить Dos для выполнения нужной программы, а затем перезагрузить дисковый файл в память так, чтобы ОС смогла возобновить работу с того места, где она была прервана.

32-разрядные приложения Windows не могут запускаться в ОС OS\2 Warp, т.к. данная версия содержит подмножество Win32s интерфейса

прикладного программирования Win32, но не имеет полного набора функций Win32. Однако для запуска программ Win32s в среде OS\2 Warp не требуется вносить в них каких-либо изменений.

**Выводы**

1. ОС OS/2 Warp является 32-разрядной ОС, использующей механизм вытесняющей многозадачности и многопоточности.
2. В ОС включены меры защиты от программ, разрушающих другие программы или систему.
3. ОС имеет дружественный интерфейс, используя для этого графику, управление мышью, меню, окна, объекты и другие механизмы.
4. Кроме утилит, входящих в состав ОС, каждая копия системы включает коллекцию приложений BonusPak.
5. Стабильность системы определяется в том числе и использованием модели защиты, принятой в Intel, которая позволяет программам работать на одном из 4 уровней привилегий, называемых кольцами. Самый высокий уровень привилегий имеет кольцо 0, самый низкий – кольцо 3. Аппаратные средства не дают возможности программам, работающим в менее привилегированных кольцах, переписать содержимое участков памяти, контролируемых более привилегированными программами.
6. 16-разрядные приложения для DOS и MS Windows могут выполняться в системе в их естественном виде, рядом с программами OS\2 в рамках одного «рабочего стола».
7. 32-разрядные приложения Windows не могут запускаться в ОС OS\2 Warp, т.к. данная версия не имеет полного набора функций Win32. Однако программы Win32s в среде OS\2 Warp выполняются в полном объёме.

## Лекция 13. Операционная система Windows 9x. Основные особенности и возможности. Настройка и оптимизация Windows 9x. Взаимозависимость, совместимость файловых систем. Архитектура ОС Windows 9x. Приложения для Windows 9x

Операционная система Windows ориентирована на организацию удобной среды работы пользователя на персональном компьютере. До ее появления любая операционная система требовала от пользователя знания языка команд по управлению компьютером. Windows позволила изменить облик системной среды и правила работы в ней. Появился удобный для пользователя графический интерфейс с достаточно простыми правилами работы. Оставаясь невостребованной в течение 1985 - 1990 гг., среда Windows изменила лицо компьютерного мира в последующее пятилетие, пройдя путь от графической оболочки операционной системы MS DOS в первых версиях до полноценной операционной системы в последующих версиях Windows 9x .

К работе над графической средой для персональных компьютеров IBM PC компания Microsoft приступила в 1981 г., сразу после выпуска операционной системы MS DOS 1.0. Первый этап работы завершился к апрелю 1983 г. выпуском программы Interface Manager. Однако данная программа, несмотря на серию проведенных доработок, в течение двух лет не была принята. Только в июне 1985 г. проект Interface Manager увидел свет, но под названием Windows 1,0. Этот вариант так же, как и предыдущие, не нашел поддержки, однако работа в этом направлении была продолжена.

В октябре 1987 г. вышла в свет и получила признание версия Windows 2.0. Для Windows 2.0 был разработан табличный процессор Microsoft Excel и ряд текстовых процессоров, в том числе Word 1.0. Версия Windows для компьютеров с микропроцессором 80286 предоставила возможность пользователям использовать расширенную память, а в версии Windows для компьютеров с микропроцессором 80386 впервые реализована многозадачность.

В мае 1990 г. вышла в свет Windows 3.0 - графическая оболочка операционной системы MS DOS. Эта версия Windows быстро завоевала признание среди пользователей. Windows 3.0 имела не только полноценный графический интерфейс, но и поддерживала режим многозадачности. Одно из важнейших достоинств системы Windows - унификация работы с внешними устройствами: эта система берет на себя организацию работы внешних устройств, предоставляя приложениям типовой интерфейс. При этом не надо писать драйверы под все прикладные программы, как это было в MS DOS; достаточно, чтобы был один драйвер под Windows, тогда все приложения получают доступ к устройству. Следует отметить также появление Диспетчера программ (Program Manager), который позволил перейти на более удобный способ управления компьютером с помощью мыши.

Однако надежность работы Windows 3.0 была все-таки невысокой. Этот недостаток был устранен в версии Windows 3.1, выпущенной в апреле 1992г. Начиная с этой версии, компания Microsoft реализовала заложенную в процессоре 80386 и выше возможность организации виртуальной памяти, когда на жестком диске создается "файл подкачки" для более эффективной работы оперативной памяти. Недостаточный объем оперативной памяти компьютера перестал служить основным препятствием для запуска приложений и загрузки документов, а проявлялся лишь в некотором замедлении работы системы, тем не менее, Windows по-прежнему не была операционной системой, а оставалась только удобной графической надстройкой над MS DOS. Все недостатки и ограничения MS DOS продолжали наследоваться Windows.

Дальнейшее развитие Windows происходило в направлении организации работы в сетях. Можно сказать, что магистральным направлением развития Windows становятся сетевые средства, которые с каждой новой версией приобретают все больший вес. Так, версия Windows для рабочих групп 3.11, выпущенная в октябре 1993 г., решила проблемы малых. предприятий,

связанные с необходимостью подбора, установки и настройки разнородного программного обеспечения для обслуживания одноранговых локальных сетей.
Выпущенная в сентябре 1995 г. операционная система Microsoft Windows 95 стала первой графической операционной системой для компьютерной платформы IBM PC. Однако при эксплуатации этой системы было обнаружено много недостатков и в первую очередь невысокий уровень надежности работы. Многие из них были устранены в модификации Windows 95, известной как OSR2.

Операционная система Windows 95 появилась в результате слияния операционной системы MS DOS и ее графической оболочки Windows 3.1 (3.11) и получила дальнейшее развитие в виде операционных систем Windows 9x. Основными характеристиками операционных систем линейки Windows 9x являются:

1. Полностью интегрированная 32-разрядная архитектура ОС защищенного режима, включающая управление памятью, диспетчеризацию процессов и разрешение конфликтов между ними, что устраняет необходимость в отдельных копиях DOS для процесса.
2. Вытесняющая многозадачность и многопоточность, что улучшает реакцию приложений на действие пользователя и уменьшает влияние на работу приложений фоновых процессов, таких, например, как высокоскоростной обмен данных.
3. Графический пользовательский интерфейс.
4. Динамические 32-разрядные драйверы защищённого режима доступны всем запущенным в системе процессам. Драйверы загружаются в память при необходимости и выгружаются, если их не использует ни один процесс. Приложения DOS могут использовать все 640 Кб доступной им памяти, т. к. теперь нет необходимости в драйверах реального режима DOS.
5. Использование виртуальной памяти.
6. Подключение новых периферийных устройств по технологии Plug and Play.
7. Совместимость с ранее созданным программным обеспечением DOS.
8. Устанавливаемые 32-разрядные файловые системы имеют гораздо более высокую производительность, чем аналогичные 16-разрядные. Возможно установление других файловых систем.
9. Динамическая настройка среды устраняет необходимость организации постоянных файлов вытеснения виртуальной памяти и настройки системных параметров.
10. Наличие коммуникационных программных средств.
11. Наличие средств мультимедиа.

## Настройка и оптимизация Windows 9x. Технология Plug and Play

Windows 9x поддерживает технологию Plug and Play. Стандарты этой технологии, впервые предложенные в январе 1993 года, предназначены для того, чтобы сделать конфигурирование устройств полностью автоматическим и прозрачным. Конечная цель данной технологии – исключить конфликты между устройствами, сделать ненужным составление файлов конфигурации и загрузку драйверов устройств. Технология Plug and Play также требует, чтобы драйверы устройств обеспечивали динамическое реконфигурирование.

Технология Plug and Play ( PnP ) включает в себя три компонента:

1. Версия BIOS PnP, которая содержит основные инструкции для определения устройств, необходимых для загрузки компьютера.
2. Операционная система PnP (Windows 95 является первой ОС PnP).
3. Аппаратные средства PnP, т.е. устройства PC, которые автоконфигурируемы ОС PnP.

После включения компьютера, соответствующего стандарту PnP, операционная система выполняет последовательно следующие операции:

1) системный BIOS идентифицирует устройства на материнской плате, включая тип шины, а также внешние устройства;
2) системный BIOS определяет требования ресурсов каждого устройства, причем на этом же этапе BIOS определяет какие из устройств имеют фиксированные значения ресурсов, а какие являются устройствами PnP, чьи значения ресурсов могут быть реконфигурированы;
3) ОС предоставляет ресурсы, оставшиеся после размещения фиксированных ресурсов, каждому устройству PnP, но если имеется несколько устройств, то может понадобиться несколько итераций процесса размещения ресурсов для исключения всех ресурсных конфликтов;
4) ОС создаёт конечную системную конфигурацию и сохраняет данные размещения ресурсов для этой конфигурации в реестре;
5) ОС отыскивает, требуемые для устройств, драйверы, но если соответствующий драйвер не найден, система требует установить его пользователю, после чего он загружается в память, и начальные операции заканчиваются.

## Системный Реестр в ОС MS Windows 9x

Системный Реестр Windows (СР) – это большая база данных, в которых записаны настройки, как самой операционной системы, так и приложений, в ней установленных.

Реестр Windows решает проблему централизованного хранения всех параметров настройки и инициализации системы. Реестр использует только два файла: USER.DAT и SYSTEM.DAT. Назначение этих файлов различно: первый из них предназначен для хранения информации о конкретных пользователях, а

второй - для хранения параметров системы. При запуске Windows происходит автоматическое резервное копирование текущих файлов реестра. Файлы резервных копий получают имена SYSTEM.DA0 и USER.DA0. Эта процедура предоставляет вам виртуальную гарантию того, что при следующей перезагрузке системы Windows будет иметь «хорошие» копии файлов реестра.

Сами по себе файлы реестра являются двоичными. Однако программа «Редактор реестра» представляет реестр в виде, удобном для восприятия пользователем. Представленная структура аналогична файловой системе. Главные её элементы – разделы, представленные в левом окне редактора и параметры в правом. Разделы соответствуют папкам (директориям) и могут содержать другие разделы (подразделы) и параметры. Директория на жестком диске является структурным элементом и не несет информации. Информация хранится в файлах. То же самое в структуре реестра: разделы обладают именем, тогда как параметры имеют и имя, и значение. Собственно в значениях параметров и хранится информация. Параметры могут быть трех типов: строковые, двоичные и типа «Dword». У подраздела может быть ( а может и не быть) значения только того типа, который для него допустим. Каждый раздел, помимо параметров, обладающих именем, содержит один параметр «по умолчанию», который добавляется автоматически при создании раздела и не переименовывается. Разделы обозначают категории (например, название устройства или программы), к которым относятся находящиеся внутри них параметры.

На верхнем уровне реестра находятся шесть разделов:
HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS
HKEY_CURRENT_CONFIG
HKEY_DYN_DATA.
(Обычно их называют по первым буквам: HKCR, HKCU, HKLM, HKU, HKCC, и HKDD.) Разделы верхнего уровня нельзя переименовать.

## Архитектура ОС Windows 9x

Windows 9x представляет собой пример эволюционного развития архитектуры Windows., но заложенные в ней архитектурные анахронизмы могут приводить к неожиданному краху системы

Windows 9x обладает некоторыми уникальными возможностями по сравнению с Windows3.1. Она обеспечивает хорошую поддержку мобильных пользователей, спектр ее коммуникационных средств чрезвычайно широк и гибок, она наконец предоставляет некоторое решение проблемы недостатка памяти для выполнения приложений. Недостаток памяти Windows 3.1 был напрямую связан с фиксированным объёмом памяти локальных «хипов» (64-Кбайтных сегментах памяти, адресуемых с помощью 16-разрядных

указателей), принадлежащих модулям USER и GDI. При этом система не освобождает ресурсы после завершения программ, использующих данные ресурсы, но не освободившие их после выполнения.

В Windows 9x многие структуры данных вынесены из «хипов» 16-разрядных модулей USER и GDI и перенесены в 32-разрядные «хипы», размер которых практически не ограничен. В дополнении к обеспечению расширенного пространства для «хипа» система освобождает ресурсы за программами, которые не могут делать этого самостоятельно. Слежение за ресурсами идёт по потокам. По завершении потока любые ресурсы, которые он не смог освободить, автоматически освобождаются операционной системой. Однако здесь может возникнуть проблема совместимости с 16-разрядными приложениями. Это связано с наличием 16-разрядного кода в ОС.

Наконец, переделка средств управления оперативной памятью позволит пользователям загрузить больше приложений прежде, чем система выдаст сообщение о нехватке памяти. Например, в Windows 3.1 обычно можно загрузить от 3 до 5 приложений, а для Windows 9x эти границы составляют 6 - 12 приложений.

Одной из проблем Windows 3.1 является способность приложения вызвать крах системы, вынудив сделать перезагрузку. В Windows 9x осталось много старого 16-разрядного кода, с помощью которого осуществляется выполнение приложений. Например, такие критические компоненты операционной системы, как USER и GDI, которые соответственно обеспечивают управление окнами и предоставляют средства графического интерфейса, являются по-прежнему 16-разрядными и работают в том же адресном пространстве, что и 16-разрядные приложения. Поэтому 16-разрядное приложение, содержащее ошибки, может потенциально "подвесить" виртуальную машину, на которой работают подсистемы USER и GDI, или, что еще хуже, заставить USER или GDI неверно работать, что может привести к краху всей ОС. Даже 32-разрядные приложения могут вызвать останов системы.

Большая часть нижней памяти размером в 1 Мбайт, принадлежащая адресному пространству системного кода Windows 9x (то есть системной виртуальной машине System VM), открыта для операций приложения Win32.

Многозадачность - это еще одно потенциально слабое место. Windows 9x пересылает все вызовы USER API через 16-разрядную системную виртуальную машину System VM, которая размещается там же, где и выполняемое 16-разрядное приложение. Если 16-разрядное приложение «подвешивает» машину System VM, отказываясь обрабатывать сообщение (встречающийся чаще всего тип ошибки в существующих приложениях Windows), то все остальные процессы приостанавливаются. Пока пользователь не завершит в принудительном порядке зависшее 16-разрядное приложение (в Windows 9x есть хорошее средство для выполнения этой операции) и тем самым не

освободит машину System VM, другие выполняемые программы, даже 32-разрядные, будут заблокированы.

Рисунки 4 и 5 дают возможность сравнить архитектурные решения, положенные в основу операционных систем Windows 3.1, Windows 9x.

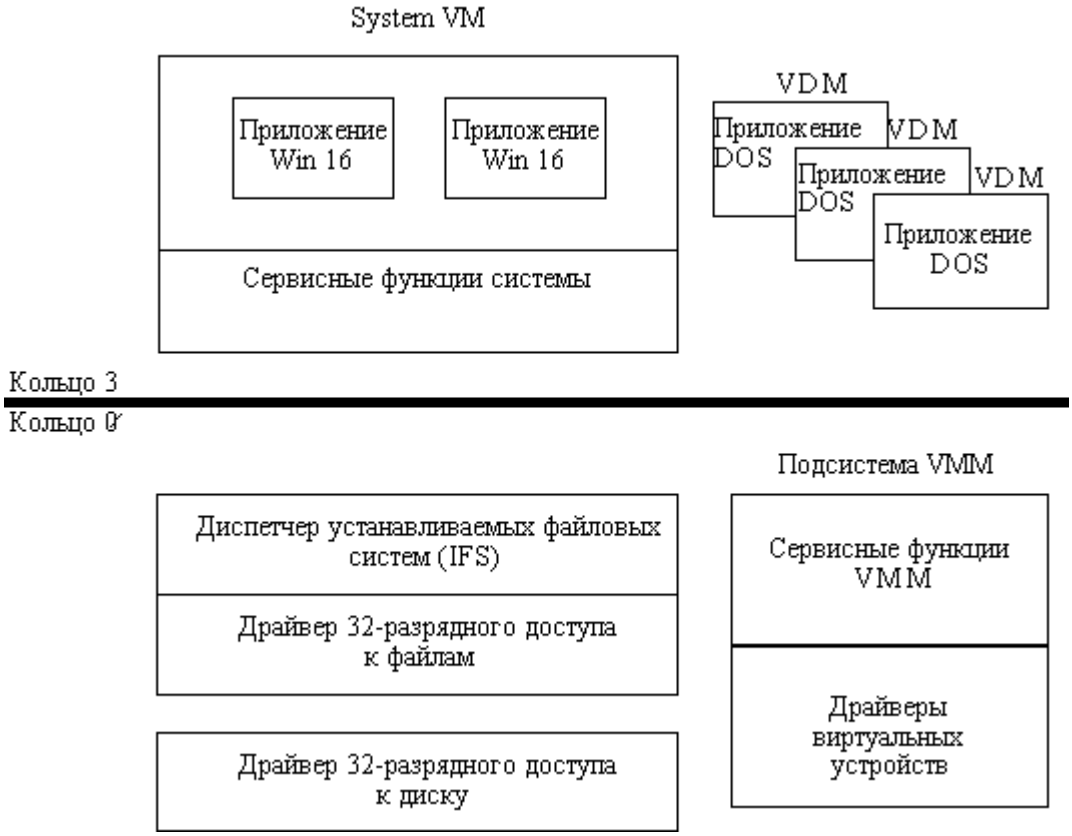


Рис. 4. Архитектура ОС Windows 3.1

В состав операционной системы Windows 3.1 входит системная виртуальная машина System VM, внутри которой размещаются все 16-разрядные приложения Win16, а также код и данные системных DLL, которые обеспечивают выполнение сервисных функций ОС. Программы Win16 выполняются в режиме невытесняющей (коллективной) многозадачности. Системные библиотеки USER, GDI и KERNEL предоставляют сервисные функции операционной системы приложениям и отображаются в адресное пространство, совместно используемое приложениями Win16. Приложения DOS запускаются на отдельных виртуальных DOS-машинах (VDM), работающих в режиме вытесняющей многозадачности. Диспетчер устанавливаемых файловых систем (IFS) и драйвер 32-разрядного доступа к файлам (только в Windows for Workgroups 3.11) осуществляют большинство файловых операций в защищенном режиме, что ускоряет доступ к файлам. Драйвер 32-разрядного доступа к диску управляет обменом с диском на физическом уровне.

Подсистема управления виртуальными машинами (VM Manager, VMM) предоставляет сервисные функции низкого уровня, такие как распределение процессорного времени между VM и управление виртуальной памятью. Сюда также относятся драйверы виртуальных устройств (VxD) для аппаратуры.

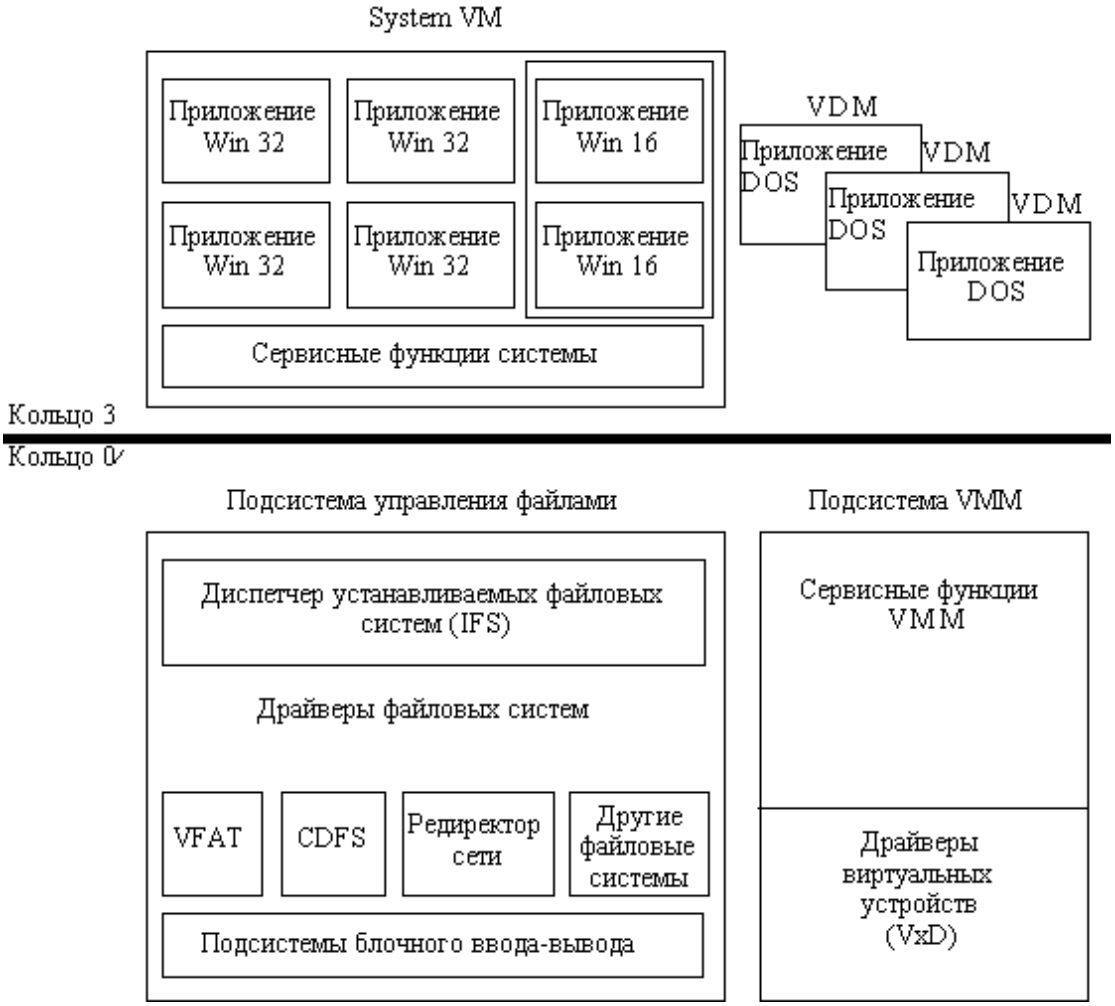


Рис. 5 Архитектура ОС Windows 9x

Архитектура Windows 9x представляет собой улучшенную версию архитектуры Windows 3.1. Внутри системной VM выполняются приложения Win16 и Win32. Большая часть кода операционной системы и данных также размещается здесь. Приложения Win32 работают на основе алгоритма вытесняющей многозадачности в отдельных адресных пространствах. Все приложения Win16 выполняются как единый процесс в общем адресном пространстве на основе алгоритма невытесняющей многозадачности. Библиотеки динамической компоновки USER, USER32, GDI, GDI32, KERNEL и KERNEL32, которые предоставляют системные сервисы всем приложениям, загружаются в системную VM и отображаются в адресные пространства

каждого прикладного процесса. Это повышает производительность за счет устранения затрат времени на переходы между кольцами защиты при вызове системных функций. Однако с другой стороны, это также ставит под угрозу целостность системы, открывая доступ к частям ОС для прикладных программ. На виртуальных DOS-машинах (VDM) выполняются DOS-программы. Они работают в режиме вытесняющей многозадачности.

Подсистема управления файлами Windows 9x работает в нулевом кольце защиты и обрабатывает все вызовы, связанные с вводом-выводом. Большинство вызовов обрабатывается в защищенном режиме, но некоторые по-прежнему приводят к переключению в режим Virtual 86 и обрабатываются в реальном режиме DOS.

Диспетчер устанавливаемых файловых систем IFS передает вызовы файлового ввода-вывода драйверу соответствующей файловой системы. Драйвер файловой системы VFAT реализует собственную VFAT-систему Windows 9x. Драйвер CDFS заменяет MSCDEX и управляет операциями по вводу данных с накопителей CD ROM. Редиректор, выполненный в виде драйвера файловой системы, обеспечивает обращение к сетевым накопителям. Можно устанавливать дополнительные драйверы файловых систем. Подсистема блочного ввода-вывода выполняет соответствующие операции на физическом уровне в ответ на запросы драйверов файловых систем.

Подсистема управления виртуальными машинами (VMM) предоставляет низкоуровневые сервисные функции, например, планирование нитей и управление памятью. Сюда также относятся драйверы виртуальных устройств (VxD) для аппаратуры. Перенос Win16 приложений на 32-разрядную платформу сопряжен с большими техническими сложностями. В этом случае необходимо решить две фундаментальные задачи по работе с целыми числами и по адресации памяти.

**Выводы**

1. Операционная система Windows 9x ориентирована на организацию удобной графической среды работы пользователя на персональном компьютере.
2. 32-разрядная архитектура ОС защищенного режима, включает управление памятью, диспетчеризацию процессов и разрешение конфликтов между ними.
3. В системе используется механизм вытесняющей многозадачности и многопоточности наряду с механизмом коллективной многозадачности, используется виртуальная память.
5. Устанавливаемые 32-разрядные файловые системы имеют гораздо более высокую производительность, чем аналогичные 16-разрядные. Возможно установление других файловых систем.
6. В ОС Windows 9x в зависимости от версии имеется в наличие набор коммуникационных программных средств и средств мультимедиа.
7. Windows 9x поддерживает технологию Plug and Play.

## Лекция 14. Файловые системы FAT, HPFS, NTFS. Взаимозависимость, совместимость файловых систем

Наряду с системой управления процессами, файловая система является одной из основных составляющих любой операционной системы и имеет свои особенности для каждой ОС.

Файловая система - это компонент операционной системы, обеспечивающий организацию создания, хранения и доступа к именованным наборам данных. Эти именованные наборы данных называются файлами, т.е. файл - некоторый набор данных, с которым можно работать через имя.

ОС однозначно определяет интерфейс взаимодействия с пользователем и имеет стандартный набор функций:

1. Открытие файла. Средство, устанавливающее связь между файлом на диске и ОС. Данная операция необходима, для того чтобы объявить ОС, что файл будет работать с конкретным процессом (только после открытия файла из него можно читать и записывать в него). ОС, исходя из некоторых соображений, может принимать решение по поводу работы с файлом (например, блокирование доступа в этот файл для других процессов).
2. Закрытие файла - завершение работы с файлом. Доступ к файлу может быть изменен, с ним могут работать другие процессы.
3. Операция чтения/записи. Файл состоит из блоков, обмен осуществляется блоками. Допустим обмен блоков не ограниченного размера.
4. Создание нового файла. С новым именем будет ассоциироваться место на диске, где после работы с файлом будет храниться информация.
5. Управление файловым указателем. Практически с каждым открытым файлом связывается понятие файлового указателя. Этот указатель, стоит на начале файла. После чтения указатель переносится на позицию прочитанного блока. Для организации работы с файлом требуется уметь управлять этим указателем. Указатель есть некоторая переменная, доступная программе, которая связана с функцией открытия файла (создающей эту переменную).
6. Защита данных (верификация доступа). Многопользовательские системы могут верифицировать своих пользователей.

### Подходы к реализации ФС

Одноуровневая организация файлов непрерывными сегментами.

Термин «одноуровневая» означает, что система обеспечивает работу с файлами уникально именованными.

Схема организации состоит в том, что на некотором пространстве внешней памяти выделяется некоторая непрерывная область памяти для хранения данных и область данных которая называется каталог. Каталог имеет следующую структуру:

| имя | начало файла | конец файла |
|---|---|---|
| ... | ... | ... |
| .. | ... | ... |

Рис.6. Одноуровневая организация файлов непрерывными сегментами

Каждый файл имеет свою строку в этой таблице. Если открываем новый файл, то появляется новая строка, при удалении эта строка очищается. Здесь есть физические ограничения: количество строк ограничено. При работе с файлом быстро осуществляются операции чтения/записи, т.к. мы знаем где начало и конец файла. Повысить эффективность можно разместив каталог в ОП. Все выполняется быстрее за счет уменьшения количества обменов. Большим достоинством является - быстрый поиск при обмене информации. К недостаткам прежде всего следует отнести выделение непрерывной области на внешнем носителе. При создании нового файла необходимо знать максимально допустимый размер. Если не хватает места, то либо выдается ошибка, либо осуществляется алгоритм поиска свободного места (непрерывного и равного нужного объема). Во втором случае гарантии нет, т.к. такого объёма свободного пространства может и не быть. Пропадает много пустого места.

**Файловая система с блочной организацией**

На пространстве внешней памяти выделяется непрерывная область данных, в которую помещается каталог. Все остальное пространство разбивается на блоки, удобные для осуществления обменных операций. Каждая строка таблицы соответствует некоторому определенному блоку ФС. Каждый файл занимает как минимум один блок. Максимальное количество блоков, занятых одним файлом, ограничено лишь структурой ФС. Таблица имеет вид, показанный на рис. 7

| № | имя файла | атрибуты |
|---|---|---|
| ... | ... | ... |
| ... | ... | ... |

Рис. 7. Файловая система с блочной организацией

В «атрибутах» содержатся некоторые подполя, например, № части файла, расположенного в этом блоке, который относится к одному файлу. Физически блоки могут располагаться в произвольном порядке.

Такая организация позволяет уйти от проблемы, связанной с размером файла, которая требовалась в предыдущем случае. При создании нового файла его размер равен размеру одного блока. Если при записи размер файла превышает размер блока, то появляется новая строка в таблице с новым блоком (тем же именем файла). Не заполненным строкам таблицы соответствует

список свободных блоков памяти. При такой организации снимается проблема внешней фрагментации, но остается проблема внутренней фрагментации (связанная с размером блока).

Эффективность работы подобной ФС ниже, чем при работе с непрерывными сегментами (каталог может быть большого размера, следовательно, он расположен не в ОП, следовательно затраты времени на обмен увеличиваются). В такой системе требуется уникальность имен лишь среди файлов одного пользователя.

**Иерархическая файловая система**

В основе подобной структуры лежит дерево. В корне дерева находится так называемый корень файловой системы. Лист - пустой каталог, либо файл. Узлы дерева, отличные от листа, являются файлами-каталогами. Создавать каталог можно в любой момент времени.

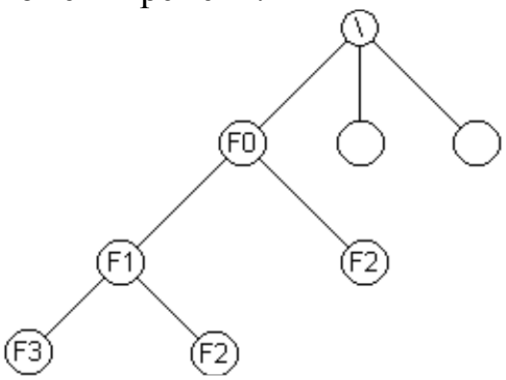


Рис. 8. Иерархическая файловая система

Для именования файлов используется механизм, основанный на понятии имени файла и полного пути. Таким образом, получается, что имена файлов уникальны в пределах одного каталога, т.е. частично снимается проблема именования. Здесь существует понятие текущий каталог, на который настроена ФС. Такая организация хороша тем, что она позволяет работать как с коротким именем файла (если системно подразумевается, что мы работаем в данном каталоге), так и с полным именем файла. Есть возможность смены текущего каталога. Полные имена файлов есть пути, а в любом дереве от его корня до любого узла существует единственный путь.

Согласно этой иерархии, каждому из файлов можно привязывать какие-то атрибуты, связанные с правами доступа. Правами доступа могут обладать как пользовательские файлы, так и каталоги. Структура этой системы хороша для организации многопользовательской работы, за счет отсутствия проблемы именования, и такая система может очень хорошо наращиваться. Но она менее эффективна, чем одноуровневая ФС с непрерывными сегментами. Безусловно, что все файловые структуры современных ОС являются иерархическими ФС.
Перейдем к рассмотрению конкретных ФС.

Рассмотрим основные понятия, связанные с файловыми системами. Во-первых, «кластеры» - это единица размещения (блок) данных на диске, которой

оперирует любая файловая система в DOS или в Windows. В разделе жесткого диска доступное пространство равномерно разделено на кластеры, причем каждый кластер может или быть незанятым, доступным, т. е. предназначенным для использования файлом, или дефектным, непригодным для использования. Состояние каждого кластера записывается и сохраняется в FAT раздела жесткого диска при помощи 12-разрядных, или 16-разрядных записей (элементов описания). Поскольку каждая запись в FAT содержит указатель на следующий кластер в «цепочке кластеров», то чем больше размер записи в FAT, тем больше максимально возможное число значений указателей и, следовательно, максимально возможное число кластеров, которые содержит в себе раздел жесткого диска. Например, 12-разрядная FAT может контролировать состояние не более 4.086 кластеров, а 16-разрядная FAT способна управиться с 65.526 кластерами. Во-вторых, по причинам, таящимся глубоко в недрах DOS, за основание системы исчисления размеров кластеров принят 512-байтный сектор диска, и кластер должен иметь размер, равный основанию (512 байт), умноженному на 2 в степени n, а именно 512 байт, 1 Кбайт, 2 Кбайт, 4 Кбайт, 8 Кбайт, 16 Кбайт или 32 Кбайт, т. е . размер сектора в FAT зависит от размера жесткого диска в отличие от файловых систем NTFS и HPFS, где регламентировано не количество секторов, а размер кластера - 512 байт.

**Различные варианты FAT**

Сокращение FAT означает «таблица размещения файлов» (File Allocation Table). Этот термин относится к линейной табличной структуре со сведениями о файлах — именами файлов, их атрибутами и другими данными, определяющими местонахождение файлов (или их фрагментов) в среде FAT. Элемент FAT определяет фактическую область диска, в которой хранится начало физического файла.

**Исходная FAT**

Файловая система FAT изначально использовалась в DOS. Ее история прослеживается практически до самых ранних моделей PC на базе DOS. Таблица размещения файлов, по имени которой была названа FAT, представляет собой простую табличную структуру. Когда пользователь обращается к каталогам или находящимся в них файлам, эта структура поэлементно просматривается от начала к концу. Невзирая на свою простоту и даже примитивность, FAT была распространена достаточно широко и встречается в различных версиях DOS, Windows 3.x, Windows NT, Macintosh и многих разновидностях Unix. Ряд других, более экзотических операционных систем также поддерживали файловую структуру FAT. По этой причине рекомендуется использовать тома FAT в Windows NT Server или Workstation в тех случаях, когда различные типы клиентов должны обмениваться данными через единую файловую систему.

Одной из важнейших характеристик исходной FAT было использование имен файлов формата 8.3. Это означает, что имя файла имеет длину до восьми

символов, перед расширением ставится точка, а само расширение имеет длину не более трех символов. Многие пользователи считают это ограничение отличительным признаком DOS, но на самом деле оно связано с файловой системой FAT.

Поскольку размеры файлов и разделов, находящихся под управлением FAT, росли с каждой новой версией DOS, в таблицах размещения FAT используется два типа указателей. Для разделов объемом менее 50 Мбайт используются 12-разрядные указатели, а для больших разделов — 16-разрядные. Вот почему тома FAT иногда обозначаются FAT12 (для томов с 12-разрядными элементами) или FAT16 (для томов с 16-разрядными элементами В DOS версий от 3.0 до 6.22 использовались 16-разрядные драйверы FAT (хотя это никак не отражается на размере элементов FAT — пометка FAT12 или FAT16 определяется исключительно размерами тома FAT).

В дальнейшем к стандартной FAT добавились еще две разновидности, используемые в современных операционных системах Microsoft (конкретно — в Windows 9x и Windows NT): VFAT (виртуальная FAT) и FAT32.

Таблица 3

Основные характеристики файловых систем

| Файловая система | FAT | VFAT | FAT32 | NTFS | HPFS |
|---|---|---|---|---|---|
| Максимальный размер тома | 2 Гбайт | 4 Гбайт | 4 Тбайт | 16 Эбайт | 2 Тбайт |
| Максимальный размер файла | 2 Гбайт | 4 Гбайт | 4 Тбайт | 16 Эбайт | 2 Тбайт |
| Максимальное количество файлов в корневом каталоге | 512 | 512 | Неограниченно | Неограниченно | Неограниченно |
| Максимальное количество файлов в некорневом каталоге | 65535 | Неограниченно | Неограниченно | Неограниченно | Неограниченно |
| Безопасность на уровне файлов | Нет | Нет | Нет | Да | Да |
| Поддержка длинных имен файлов | Нет | Да | Да | Да | Да |
| Самовосстановление | Нет | Нет | Да | Да | Да |
| Ведение журналов транзакций | Нет | Нет | Нет | Да | Нет |
| Сжатие на уровне файлов | Нет | Нет | Нет | Да | Да |
| Соответствие стандарту POSIX | Нет | Нет | Нет | Да | Нет |

## VFAT (виртуальная FAT)

Файловая система VFAT впервые появилась в Windows for Workgroups 3.11 и была предназначена для выполнения файлового ввода/вывода в защищенном режиме. С выходом Windows 9x в VFAT добавилась поддержка длинных имен файлов (LFN). Тем не менее VFAT сохраняет совместимость с исходным вариантом FAT; это означает, что наряду с длинными именами в ней поддерживаются имена формата 8.3, а также существует специальный механизм для преобразования имен 8.3 в длинные имена, и наоборот. Именно файловая система VFAT поддерживается исходными версиями Windows 9x, Windows NT 3.51 и Windows NT 4. При работе с VFAT крайне важно использовать файловые утилиты, поддерживающие VFAT вообще и длинные имена в частности. Дело в том, что более ранние файловые утилиты DOS запросто модифицируют то, что кажется им исходной структурой FAT. Это может привести к потере или порче длинных имен из таблицы FAT, поддерживаемой VFAT (или FAT32). Следовательно, для томов VFAT необходимо пользоваться файловыми утилитами, которые понимают и сохраняют файловую структуру VFAT. В исходной версии Windows 95 основной файловой системой была 32-разрядная VFAT. VFAT может использовать 32-разрядные драйверы защищенного режима или 16-разрядные драйверы реального режима. При этом элементы FAT остаются 12- или 16-разрядными, поэтому на диске используется та же структура данных, что и в предыдущих реализациях FAT. VFAT обрабатывает все обращения к жесткому диску и использует 32-разрядный код для всех файловых операций с дисковыми томами.

Как видно из табл. 3, VFAT отличается от исходной FAT в первую очередь увеличением максимального размера тома и отдельных файлов, увеличением количества файлов в некорневых каталогах и поддержкой длинных имен наряду с именами формата 8.3. VFAT также поддерживает 16-и 32-разрядные обращения, тогда как исходная FAT ограничивалась 16-разрядными.

## Файловая система FAT32

32-разрядная файловая система FAT32 пришла на смену VFAT в Microsoft Windows 95 OEM Service Release 2. FAT32 является полностью самостоятельной 32-разрядной файловой системой (как NTFS) и содержит многочисленные усовершенствования и дополнения по сравнению с предыдущими реализациями FAT. Самое принципиальное отличие заключается в том, что FAT32 намного эффективнее расходует дисковое пространство. FAT32 использует дисковые кластеры меньшего размера по сравнению с предыдущими версиями, которые ограничивались 65 535 кластерами на том (соответственно с увеличением размера диска приходилось увеличивать и размер кластеров). Следовательно, даже для дисков размером до 8 Гбайт FAT32 может использовать 4-килобайтные кластеры. В результате по сравнению с дисками FAT16 экономится в среднем 10-15% дискового пространства.

FAT32 также может перемещать корневой каталог и использовать резервную копию FAT вместо стандартной. Расширенная загрузочная запись FAT32 позволяет создавать копии критических структур данных; это повышает устойчивость дисков FAT32 к нарушениям структуры FAT по сравнению с предыдущими версиями. Корневой каталог в FAT32 представлен в виде обычной цепочки кластеров. Следовательно, корневой каталог может находиться в произвольном месте диска, что снимает действовавшее ранее ограничение на размер корневого каталога (512 элементов).

Поскольку прежние утилиты FAT (для FAT32 в эту категорию входят обе файловые системы, FAT и VFAT) могут повредить или уничтожить важную служебную информацию, для томов FAT32 нельзя пользоваться никакими файловыми утилитами, кроме утилит FAT32.

Кроме повышения емкости FAT до 4 Тбайт для томов и отдельных файлов FAT32 вносит ряд необходимых усовершенствований в структуру корневого каталога. Предыдущие реализации требовали, чтобы вся информация корневого каталога FAT находилась в определённом месте дискового пространства (обычно объём этого пространства составляет 32 кластера). При этом корневой каталог мог содержать не более 512 файлов, т. к. имени файла в формате 8.3 соответствует 32-байтный элемент каталога. Появление длинных имен фактически привело к дальнейшему уменьшению количества файлов, находящихся в корневом каталоге. Поскольку длинное имя может содержать до 256 символов, всего один файл с полным длинным именем занимает до 25 элементов FAT (1 для имени 8.3 и еще 24 для самого длинного имени). Таким образом, количество элементов корневого каталога VFAT уменьшается до 21, если все имена содержат 256 символов. Следует избегать длинных имен в корневых каталогах FAT при отсутствии FAT32.

Длина полной файловой спецификации, включающей путь и имя файла (длинное или 8.3), тоже ограничивается 260 символами. FAT32 успешно справляется с проблемой длинных имен в корневом каталоге, но проблема с ограничением длины полной файловой спецификации остается. По этой причине рекомендуется ограничивать длинные имена 75-80 символами, чтобы оставить достаточно места для пути (180-185 символов).

Наконец, FAT32 повышает отказоустойчивость FAT. Во-первых, в загрузочных записях FAT32 хранятся важнейшие данные файловой системы (например, сведения о таблице разделов). Во-вторых, в FAT32 можно отключить зеркальное копирование FAT, чтобы для поиска файлов и работы с ними использовалась вторая копия FAT. Но средства самовосстановления FAT при всей полезности уступают своим аналогам в NTFS.

### Файловая система HPFS

HPFS - сокращенное название высокопроизводительной файловой системы (high performance file system), совместно разработанной в 1989 году корпорациями IBM и Microsoft.

Эта система была разработана, чтобы преодолеть некоторые недостатки FAT, к числу которых относятся:

1) ограничения, налагаемые на размер файлов и дискового пространства;
2) ограничение длины имени файла;
3) фрагментация файлов, приводящая к снижению быстродействия системы и износу оборудования;
4) непроизводительные затраты памяти, вызванные большими размерами кластеров;
5) подверженность потерям данных.

В OS/2 применяется метод хранения расширенных атрибутов (extended attributes). В разделе FAT файл, содержащий единственный символ, занял бы целый кластер для размещения собственно файла и еще один кластер для расширенных атрибутов.

Так как расширенные атрибуты почти всегда имеют объем меньше 300 байт, размер теряемого впустую дискового пространства изменяется от примерно половины кластера при использовании малых разделов до львиной доли объема кластер при больших разделах. В сумме на каждом файле теряется примерно кластер.

Переход на HPFS позволил сэкономить дисковое пространство. HPFS распределяет пространство, основываясь на физических 512-байтовых секторах, а не на кластерах, независимо от размера раздела. Система HPFS позволяет уменьшить и непроизводительные потери, так как в ней предусмотрено хранение до 300 байт расширенных атрибутов в F-узле файла, без захвата для этого дополнительного сектора.

Файловая система HPFS обеспечивает низкий уровень фрагментации. Хотя избавиться полностью от нее не удается, снижение производительности, возникающее по этой причине, почти незаметно для пользователя.

Первые 16 секторов раздела HPFS составляют загрузочный блок. Эта область содержит метку диска и код начальной загрузки системы. Сектор 16, известный под названием суперблок, содержит много общей информации о файловой системе в целом: размер раздела, указатель на корневой каталог, счетчик элементов каталогов, номер версии HPFS, дату последней проверки и исправления раздела при помощи команды CHKDSK, а также дату последнего выполнения процедуры дефрагментации раздела. Он также содержит указатели на список испорченных блоков на диске, таблицу дефектных секторов и список доступных секторов.

Сектор 17 носит название SpareBlock (запасной блок). Он содержит указатель на список секторов, которые можно использовать для «горячего» исправления ошибок, счетчик доступных секторов для «горячего» исправления ошибок, указатель на резерв свободных блоков, применяемых для управления деревьями каталогов, и информацию о языковых наборах символов.

Система HPFS использует информацию о языковых наборах, чтобы дать возможность пересылать файлы, составленные на разных языках, даже в том

случае, когда имена файлов содержат уникальные для какого-либо языка символы. SpareBlock также содержит так называемый «грязный» флаг. Этот новый флаг сообщает операционной системе о том, было ли завершение предыдущего сеанса работы нормальным, либо произошло в результате сбоя электропитания, либо файлы не были закрыты должным образом по какой-то другой причине. Если этот флаг обнаружен во время начальной загрузки, то операционная система автоматически запускает утилиту CHKDSK, пытаясь обнаружить и исправить все ошибки, внесенные в файловую систему из-за неправильного выключения системы.

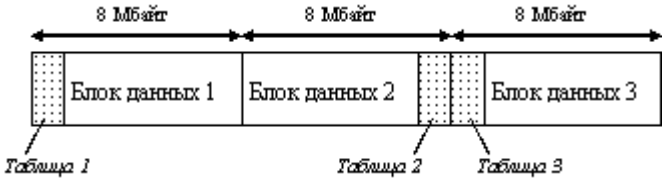


Рис.9. Прием увеличения доступного непрерывного пространства

Во время форматирования раздела HPFS делит его на полосы по 8 Мбайт каждая. Каждая полоса - ее можно представить себе как виртуальный "мини-диск" - имеет отдельную таблицу объемом 2 Кбайт, в которой указывается. какие секторы полосы доступны, а какие заняты. Чтобы максимально увеличить протяженность непрерывного пространства для размещения файлов, таблицы попеременно располагаются в начале и в конце полос (рис.9.). Этот метод позволяет файлам размером до 16 Мбайт (минус 4 Кбайта, отводимые для размещения таблицы) храниться в одной непрерывной области.

Затем файловая система HPFS оценивает размер каталога и резервирует необходимое пространство в полосе, расположенной ближе всего к середине диска. Сразу же после форматирования объем диска в HPFS кажется меньше, чем в FAT, так как заранее резервируется место для каталогов в центре диска. Место резервируется в середине диска для того, чтобы физические головки, считывающие данные, никогда не проходили более половины ширины диска.

Тот факт, что все пространство заранее распределено, также позволяет HPFS использовать специально оптимизированное программное обеспечение для более быстрой и эффективной работы с каталогами. Сравните это с системой FAT, где головкам требуется пройти весь путь к началу диска и прочитать таблицу размещения файлов, затем найти кластер, вновь пройти к началу диска, чтобы определить в FAT местонахождения следующего кластера, и так далее. Эта процедура становится еще более неудобной по мере нарастания фрагментации. Поэтому ясно, что размещение каталогов в середине диска повышает производительность системы. Вместе с тем такое предварительное распределение не накладывает ограничений на число файлов, которые могут быть размещены на жестком диске. В редких случаях, когда системе HPFS потребуется больше пространства, чем изначально было отведено под каталоги,

она может выделить дополнительное пространство из любой доступной области диска.

Число файлов в каждом блоке каталога - переменная величина, зависящая от длины имен файлов, которые содержатся в нем. Имена файлов в HPFS могут иметь длину до 254 символов, они сортируются в порядке, определяемом последовательностью символов в текущей кодовой странице системы.

HPFS использует для хранения элементов каталогов структуру данных, называемую B-деревом. Каждый элемент каталога начинается с числа, представляющего длину элемента, которая изменяется в зависимости от длины имени файла. Затем следуют время и дата создания файла, его размер и атрибуты (только для чтения, архивный, скрытый и системный), а также указатель на F-узел файла. Каждый файл (и каталог) имеет F-узел - структуру данных, занимающую один сектор и содержащую принципиально важную информацию о файле.

F-узел содержит указатель на начало файла, первые 15 символов имени файла, дополнительные временные маркеры последней записи и последнего доступа, журнал, хранящий информацию о предыдущих обращениях к файлу, структуру распределения, описывающую размещение файла на диске, и первые 300 байт расширенных атрибутов файла. (Расширенные атрибуты редко занимают более 300 байт, что фактически означает, что HPFS для получения этой информации приходится читать на один сектор меньше, чем FAT).

Программы LAN Server и LAN Manager фирмы IBM также сохраняют в F-узле информацию об управлении пользовательским доступом (Access Control). F-узлы хранятся в смежных с представляемыми ими файлами секторах, поэтому, когда файл открывается, то четыре автоматически считываемых в кэш сектора содержат F-узел и три первых сектора файла.

Структура размещения HPFS имеет дополнительные преимущества по сравнению с FAT благодаря техническому приему, называемому кодированием по длине выполнения (Run Length Encoding, RLE). Вместо того, чтобы определять в таблице каждый используемый сектор, HPFS сохраняет указатель на первый сектор и число последовательно расположенных используемых секторов. Каждая область дискового пространства, описываемая парой (сектор, длина), называется экстентом. Хотя HPFS и сводит фрагментацию к минимуму, файлы все же могут быть в некоторой степени фрагментированными. В таких ситуациях пары, описывающие экстенты, добавляются к F-узлу файла. Один F-узел может хранить до 8 экстентов, обеспечивая достаточное пространство для большинства файлов.

Если все же потребуется еще большее пространство, то HPFS изменяет структуру таким образом, что F-узел становится корнем B+-дерева секторов размещения. B+-дерево является вариантом бинарного B-дерева. Созданное как структура для более быстрого обнаружения данных по сравнению с методом последовательного перебора, бинарное дерево состоит из ветвей, каждая из которых представляет выбор одного из двух возможных продолжений.

## NTFS (New Technology File System)

В название файловой системы NTFS не зря входят слова «New Technology», то есть «новая технология» — NTFS содержит ряд значительных усовершенствований и изменений, специфических для Windows NT. С точки зрения пользователей, файлы по-прежнему хранятся в каталогах. Однако в NTFS и HPFS, в отличие от FAT, не существует ни особых свойств корневых каталогов, ни ограничений, связанных с аппаратурой (например, возможности обращения к максимальному количеству дисковых секторов или кластеров). На томах NTFS не существует специальных областей наподобие таблицы размещения файлов, по которой FAT получила свое имя. При проектировании NTFS особое внимание было уделено следующим характеристикам:

1. Надежность. Высокопроизводительные компьютеры и системы совместного пользования должны обладать повышенной надежностью, которая является ключевым элементом структуры и поведения NTFS.
2. Расширенная функциональность. NTFS проектировалась с учетом возможного расширения. В ней были воплощены многие дополнительные возможности — усовершенствованная отказоустойчивость, эмуляция других файловых систем, мощная модель безопасности, параллельная обработка потоков данных и создание файловых атрибутов, определяемых пользователем.
3. Поддержка POSIX. Поскольку правительство США требует, чтобы все закупаемые им системы хотя бы в минимальной степени соответствовали стандарту POSIX, такая возможность была предусмотрена и в NTFS. К числу базовых средств файловой системы POSIX относится необязательное использование имен файлов с учетом регистра, хранение времени последнего обращения к файлу и механизм так называемых «жестких ссылок» — альтернативных имен, позволяющих ссылаться на один и тот же файл по двум и более именам.
4. Гибкость. Модель распределения дискового пространства в NTFS отличается чрезвычайной гибкостью. Размер кластера может изменяться от 512 байт до 64 Кбайт; он представляет собой число, кратное внутреннему кванту распределения дисковой аппаратуры. NTFS также поддерживает длинные имена файлов, набор символов Unicode и альтернативные имена формата 8.3 для совместимости с FAT.

### Физическая структура NTFS

Раздел NTFS теоретически может быть почти какого угодно размера. Максимальный размер раздела NTFS в данный момент ограничен лишь размерами жестких дисков.

### Структура раздела

Как и любая другая система, NTFS делит все полезное место на кластеры - блоки данных, используемые единовременно. NTFS поддерживает почти

любые размеры кластеров - от 512 байт до 64 Кбайт, неким стандартом же считается кластер размером 4 Кбайт.

Диск NTFS условно делится на две части. Первые 12% диска отводятся под так называемую MFT зону - пространство, в которое растет метафайл MFT. Запись каких-либо данных в эту область невозможна. MFT-зона всегда держится пустой - это делается для того, чтобы самый главный, служебный файл (MFT) не фрагментировался при своем росте. Остальные 88% диска представляют собой обычное пространство для хранения файлов.

Свободное место диска, однако, включает в себя всё физически свободное место - незаполненные куски MFT-зоны туда тоже включаются. Механизм использования MFT-зоны таков: когда файлы уже нельзя записывать в обычное пространство, MFT-зона просто сокращается (в текущих версиях операционных систем ровно в два раза), освобождая таким образом место для записи файлов. При освобождении места в обычной области MFT зона может снова расширится. При этом не исключена ситуация, когда в этой зоне остались и обычные файлы: никакой аномалии тут нет.

## MFT и его структура

Файловая система NTFS представляет собой выдающееся достижение структуризации: каждый элемент системы представляет собой файл - даже служебная информация. Самый главный файл на NTFS называется MFT, или Master File Table - общая таблица файлов. Именно он размещается в MFT зоне и представляет собой централизованный каталог всех остальных файлов диска и себя самого.

MFT поделен на записи фиксированного размера (обычно 1 Кбайт), и каждая запись соответствует какому- либо файлу (в общем смысле этого слова). Первые 16 файлов носят служебный характер и недоступны операционной системе - они называются метафайлами, причем самый первый метафайл - сам MFT. Эти первые 16 элементов MFT - единственная часть диска, имеющая фиксированное положение. Вторая копия первых трех записей для надежности хранится ровно посередине диска. Остальной MFT-файл может располагаться, как и любой другой файл, в произвольных местах диска - восстановить его положение можно с помощью его самого, «зацепившись» за самую основу - за первый элемент MFT.

## Метафайлы

Первые 16 файлов NTFS (метафайлы) носят служебный характер. Каждый из них отвечает за какой-либо аспект работы системы. Преимущество настолько модульного подхода заключается в поразительной гибкости - например, на FAT-е физическое повреждение в самой области FAT фатально для функционирования всего диска, а NTFS может сместить, даже фрагментировать по диску, все свои служебные области, обойдя любые неисправности поверхности - кроме первых 16 элементов MFT.

Метафайлы находятся корневом каталоге NTFS диска - они начинаются с символа имени “$”, хотя получить какую-либо информацию о них стандартными средствами сложно. И для этих файлов указан вполне реальный размер - можно узнать, например, сколько операционная система тратит на каталогизацию всего вашего диска, посмотрев размер файла $MFT. Далее приведены используемые метафайлы и их назначение:

1. $MFT сам MFT.
2. $MFTmirr копия первых 16 записей MFT, размещенная посередине диска.
3. $LogFile файл поддержки журналирования.
4. $Volume служебная информация - метка тома, версия файловой системы. $AttrDef список стандартных атрибутов файлов на томе$. корневой каталог.
5. $Bitmap карта свободного места тома$Boot загрузочный сектор (если раздел загрузочный.
6. $Quota файл, в котором записаны права пользователей на использование дискового пространства (начал работать лишь в NT5).
7. $Upcase файл - таблица соответствия заглавных и прописных букв и имен файлов на текущем томе. Нужен в основном потому, что в NTFS имена файлов записываются в Unicode, что составляет 65 тысяч различных символов, искать большие и малые эквиваленты которых очень нетривиально.

**Файлы и потоки**

Понятие файла в NTFS – это, прежде всего, обязательный элемент - запись в MFT, так как все файлы диска упоминаются в MFT. В этом месте хранится вся информация о файле, за исключением собственно данных. Имя файла, размер, положение на диске отдельных фрагментов и т.д. Если для информации не хватает одной записи MFT, то используются несколько, причем не обязательно подряд. Опциональный элемент - потоки данных файла. Во-первых, файл может не иметь данных - в таком случае на него не расходуется свободное место самого диска. Во-вторых, файл может иметь не очень большой размер. Тогда идет в ход довольно удачное решение: данные файла хранятся прямо в MFT, в оставшемся от основных данных месте в пределах одной записи MFT. Файлы, занимающие сотни байт, обычно не имеют своего «физического» воплощения в основной файловой области - все данные такого файла хранятся в одном месте - в MFT.

Интересно обстоит дело и с данными файла. Каждый файл на NTFS имеет несколько абстрактное строение - у него нет как таковых данных, а есть потоки (streams). Один из потоков и носит привычный нам смысл - данные файла. Но большинство атрибутов файла - тоже потоки. Таким образом, получается, что базовая сущность у файла только одна - номер в MFT, а всё остальное опционально. Данная абстракция может использоваться для создания довольно удобных вещей - например, файлу можно «прибавить» еще один поток, записав

в него любые данные - например, информацию об авторе и содержании файла, как это сделано в Windows 2000 (самая правая закладка в свойствах файла, просматриваемых из проводника). Интересно, что эти дополнительные потоки не видны стандартными средствами: наблюдаемый размер файла - это лишь размер основного потока, который содержит традиционные данные. Можно, к примеру, иметь файл нулевой длинны, при стирании которого освободится 1 Гбайт свободного места - просто потому, что какая-нибудь программа или технология прикрепила в нему дополнительный поток (альтернативные данные) гигабайтового размера. Но на самом деле в текущий момент потоки практически не используются, хотя гипотетически они возможны.

Файл на NTFS - это более глубокое и глобальное понятие, чем можно себе представить просто просматривая каталоги диска. Имя файла может содержать любые символы, включая полый набор национальных алфавитов, так как данные представлены в Unicode - 16-битном представлении, которое дает 65535 разных символов. Максимальная длина имени файла - 255 символов.

## Каталоги

Каталог на NTFS представляет собой специфический файл, хранящий ссылки на другие файлы и каталоги, создавая иерархическое строение данных на диске. Файл каталога поделен на блоки, каждый из которых содержит имя файла, базовые атрибуты и ссылку на элемент MFT, который уже предоставляет полную информацию об элементе каталога. Внутренняя структура каталога представляет собой бинарное дерево. Это означает, что для поиска файла с данным именем в линейном каталоге, таком, например, как у FAT-а, операционной системе необходимо просматривать все элементы каталога, пока она не найдет нужный. Бинарное же дерево располагает имена файлов таким образом, чтобы поиск файла осуществлялся более быстрым способом - с помощью получения двухзначных ответов на вопросы о положении файла. Вопрос, на который бинарное дерево способно дать ответ, таков: в какой группе, относительно данного элемента, находится искомое имя - выше или ниже? Мы начинаем с такого вопроса к среднему элементу, и каждый ответ сужает зону поиска в среднем в два раза. Файлы, скажем, просто отсортированы по алфавиту, и ответ на вопрос осуществляется очевидным способом - сравнением начальных букв. Область поиска, суженная в два раза, начинает исследоваться аналогичным образом, начиная опять же со среднего элемента.

Это означает, что для поиска одного файла среди 1000, например, FAT придется осуществить в среднем 500 сравнений (наиболее вероятно, что файл будет найден на середине поиска), а системе на основе дерева - всего около 12-ти (2^10 = 1024). Экономия времени поиска существенна. Но стоит обратить внимание, что в традиционных системах (FAT) не всё так плохо: во-первых, поддержание списка файлов в виде бинарного дерева довольно трудоемко, а во-вторых - даже FAT в исполнении последних версий системы (Windows2000 или

Windows98) использует сходную оптимизацию поиска. Какую информацию можно получить, просто прочитав файл каталога? Ровно то, что выдает команда dir. Для выполнения простейшей навигации по диску не нужно лазить в MFT за каждым файлом, надо лишь читать самую общую информацию о файлах из файлов каталогов. Главный каталог диска - корневой - ничем не отличается об обычных каталогов, кроме специальной ссылки на него из начала метафайла MFT.

## Журналирование

NTFS - отказоустойчивая система, которая может привести себя в корректное состояние при практически любых реальных сбоях. Любая современная файловая система основана на таком понятии, как транзакция - действие, совершаемое целиком и корректно или не совершаемое вообще. У NTFS просто не бывает промежуточных (ошибочных или некорректных) состояний - квант изменения данных не может быть поделен на до и после сбоя, принося разрушения и путаницу - он либо совершен, либо отменен.

Механизм системы - журнал транзакций. Система, начиная запись на диск, пометила в метафайле $LogFile это свое состояние. При перезагрузке этот файл изучается на предмет наличия незавершенных транзакций, которые были прерваны аварией и результат которых непредсказуем, причем, все эти транзакции отменяются: Место, в которое осуществлялась запись, помечается снова как свободное, индексы и элементы MFT приводятся в состояние, в котором они были до сбоя, и система в целом остается стабильна. В случае, если ошибка произошла при записи в журнал, транзакция либо еще и не начиналась (идет только попытка записать намерения её произвести), либо уже закончилась, то есть идет попытка записать, что транзакция на самом деле уже выполнена. В последнем случае при следующей загрузке система сама вполне разберется, что на самом деле всё и так записано корректно, и не обратит внимания на «незаконченную» транзакцию.

Журналирование - это средство, позволяющее существенно сократить число ошибок и сбоев системы. Важно понимать, что система восстановления NTFS гарантирует корректность файловой системы, а не данных. Если при записи на диск произошла авария - данные могут и не записаться.

## Сжатие

Файлы NTFS имеют один довольно полезный атрибут – «сжатый». Дело в том, что NTFS имеет встроенную поддержку сжатия дисков - то, для чего раньше приходилось использовать Stacker или DoubleSpace. Любой файл или каталог в индивидуальном порядке может хранится на диске в сжатом виде - этот процесс совершенно прозрачен для приложений. Сжатие файлов имеет очень высокую скорость и только одно большое отрицательное свойство - огромная виртуальная фрагментация сжатых файлов, которая, правда, никому особо не мешает. Сжатие осуществляется блоками по 16 кластеров и

использует так называемые «виртуальные кластеры» - опять же предельно гибкое решение, позволяющее добиться интересных эффектов - например, половина файла может быть сжата, а половина – нет.

Безопасность

NTFS содержит множество средств разграничения прав объектов. Права, назначаемые любому объекту и однозначно соблюдаемые системой, эволюционируют - крупные изменения и дополнения прав осуществлялись уже несколько раз и к уже Windows 2000 все-таки они пришли к достаточно разумному набору.

Права файловой системы NTFS неразрывно связаны с самой системой - то есть они, вообще говоря, необязательны к соблюдению другой системой, если ей дать физический доступ к диску. Для предотвращения физического доступа в Windows2000 (NT5) всё же ввели стандартную возможность. Система прав в своем текущем состоянии достаточно сложна.

## Особенности NTFS. Hard Links

Hard Links была в NTFS с незапамятных времен, но использовалась очень редко. Hard Link - это фунуция, когда один и тот же файл имеет два имени (несколько указателей файла-каталога или разных каталогов указывают на одну и ту же MFT запись). Допустим, один и тот же файл имеет имена 1.txt и 2.txt: если пользователь сотрет файл 1, останется файл 2. Если сотрет 2 - останется файл 1, то есть оба имени с момента создания совершенно равноправны. Файл физически стирается лишь тогда, когда будет удалено его последнее имя.

## Symbolic Links (NT5)

Данная функция – это гораздо более практичная возможность, позволяющая делать виртуальные каталоги - ровно так же, как и виртуальные диски командой subst в DOSe. Применения достаточно разнообразны: во-первых, упрощение системы каталогов. Если не подходит каталог «Documents and settings\Administrator\Documents», можно прилинковать его в корневой каталог - система будет по прежнему общаться с каталогом с длинным именем, а пользователь - с гораздо более коротким именем, полностью ему эквивалентным. Для создания таких связей можно воспользоваться программой junction (junction.zip (15 Kb), 36 кб), которую написал известный специалист Mark Russinovich (http://www.sysinternals.com). Программа работает только в начиная с NT5 (Windows 2000), как и сама возможность.

Для удаления связи можно воспользоваться стандартной командой rd. Однако попытка удаления связи с помощью проводника или других файловых менеджеров, не понимающих виртуальную природу каталога (например, FAR), приведет к удалению данных, на которые дается ссылка.

## Шифрование (NT5)

Данная возможность полезна, так как каждый файл или каталог может быть зашифрован, что не даст возможность прочесть его другой инсталляцией NT. В сочетании со стандартным и сложным паролем на загрузку самой системы эта возможность обеспечивает достаточную для большинства применений безопасность избранных вами важных данных.

## Отличия FAT и NTFS и HPFS

Если говорить о накладных расходах на хранение служебной информации, FAT отличается от NTFS и HPFS большей компактностью и меньшей сложностью. В большинстве томов FAT на хранение таблицы размещения, содержащей информацию обо всех файлах тома, расходуется менее 1 Мбайт. Столь низкие накладные расходы позволяют форматировать в FAT жесткие диски малого объема и флоппи-диски. С другой стороны, в NTFS служебные данные занимают больше места, чем в FAT, отчасти из-за того, что каждый элемент каталога занимает 2 Кбайт (впрочем, это имеет и свои преимущества, так как содержимое файлов объемом 1500 байт и менее может полностью храниться в элементе каталога).

Система NTFS не может использоваться для форматирования флоппи-дисков. Не стоит пользоваться ею для форматирования разделов объемом менее 50 Мбайт. Относительно высокие накладные расходы приводят к тому, что для малых разделов служебные данные могут занимать до 25% объема носителя.

**Выводы**

1. Термин FAT относится к линейной табличной структуре со сведениями о файлах — именами файлов, их атрибутами и другими данными, определяющими местонахождение файлов (или их фрагментов) в среде FAT.
2. Элемент FAT определяет фактическую область диска, в которой хранится начало физического файла.
3. Состояние каждого кластера записывается и сохраняется в FAT раздела жесткого диска при помощи 12-разрядных, или 16-разрядных записей и размер каждого кластера зависит от размеров жесткого диска, причем количество кластеров регламентировано.
4. Размер кластера в FAT32, NTFS и HPFS не зависит от размера жесткого диска, т.к. в этих системах регламентировано не количество секторов, а размер кластера - 512 байт.
5. Поскольку в основу структуры каталогов NTFS и HPFS заложена эффективная структура данных, называемая «бинарным деревом», время поиска файлов в этих файловых системах не связано линейной зависимостью с их количеством.

## Лекция 15. Windows NT. Состав и особенности ОС Windows NT. Операционная система Windows 2000. Особенности развития операционных систем Windows. Концепции Windows Runtime (WinRT) и Metro UI

В конце 1988 года Microsoft начала разработку нового проекта в области программного обеспечения: создание новой ОС фирмы Microsoft для 90-х годов. Была собрана команда инженеров для разработки ОС новой технологии (New Technology - NT). Первоначально планировалось разработать NT с пользовательским и программным (API) интерфейсами в стиле OS/2, однако, OS/2 плохо продавалась, а Windows 3.0 имела большой и постоянный успех на рынке. Увидев рыночные ориентиры и сложности, связанные с развитием и поддержкой двух несовместимых систем, Microsoft решила изменить свой курс и направить своих инженеров в сторону стратегии единой цельной операционной системы. Эта стратегия состоит в том, чтобы разрабатывать семейство базирующихся на Windows операционных систем, которые охватывали бы множество типов компьютеров, от самых маленьких ноутбуков до самых больших мультипроцессорных рабочих станций. Windows NT, как было названо следующее поколение Windows-систем, занимает самое высокое место в семействе Windows. Она поддерживает графический интерфейс (GUI) пользователя Windows, а также является первой базирующейся на Windows операционной системой фирмы Microsoft, поддерживающей Win32 API, 32-битный программный интерфейс для разработки новых приложений. Win32 API делает доступными для приложений улучшенные свойства ОС, такие как многонитеевые процессы, синхронизацию, безопасность, I/O, управление объектами. В июле 1993 года появились первые ОС семейства NT - Windows NT 3.1 и Windows NT Advanced Server 3.1.

### Концепции Windows NT

### NT executive и защищенные подсистемы

При разработке структуры Windows NT была в значительной степени использована концепция микроядра и модель клиент-сервер. В соответствии с этой идеей ОС разделена на несколько подсистем, каждая из которых выполняет отдельный набор сервисных функций (серверов), например, сервис памяти, сервис по созданию процессов, или сервис по планированию процессов.

Каждый сервер выполняется в пользовательском режиме, выполняя цикл проверки запроса от клиента на одну из его сервисных функций. Клиент, которым может быть либо другая компонента ОС, либо прикладная программа, запрашивает сервис, посылая сообщение на сервер. Ядро ОС (или микроядро), работая в привилегированном режиме, доставляет сообщение нужному серверу, затем сервер выполняет операцию, после этого ядро возвращает результаты клиенту с помощью другого сообщения.

Структурно Windows NT может быть представлена в виде двух частей: часть операционной системы, работающая в режиме пользователя, и часть операционной системы, работающая в режиме ядра (рис. 10).

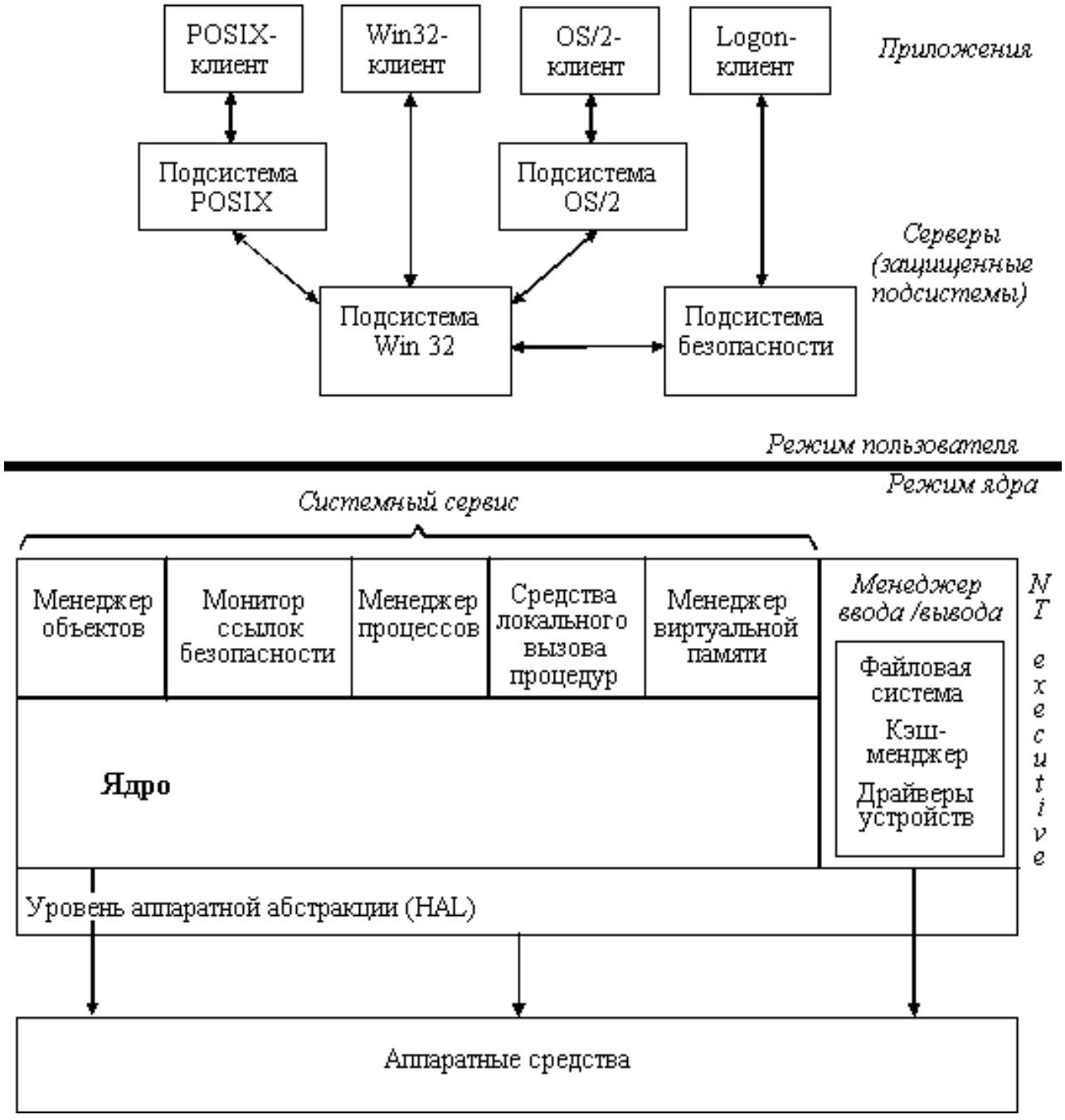


Рис. 10. Структура ОС Windows NT

Часть Windows NT, работающая в режиме ядра, называется executive - исполнительной частью. Она включает ряд компонент, которые управляют виртуальной памятью, объектами (ресурсами), вводом-выводом и файловой системой (включая сетевые драйверы), взаимодействием процессов и частично системой безопасности. Эти компоненты взаимодействуют между собой с помощью межмодульной связи. Каждая компонента вызывает другие с помощью набора тщательно специфицированных внутренних процедур.

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

Поддержку защищенных подсистем обеспечивает исполнительная часть - Windows NT executive, которая работает в пространстве ядра и никогда не сбрасывается на диск. Ее составными частями являются:

1. Менеджер объектов. Создает, удаляет и управляет объектами NT executive - абстрактными типами данных, используемых для представления ресурсов системы.
2. Монитор безопасности. Устанавливает правила защиты на локальном компьютере. Охраняет ресурсы операционной системы, выполняет защиту и регистрацию исполняемых объектов.
3. Менеджер процессов. Создает и завершает, приостанавливает и возобновляет процессы и нити, а также хранит о них информацию.
4. Менеджер виртуальной памяти.
5. Подсистема ввода-вывода.

Подсистема ввода-вывода включает в себя следующие компоненты:

1) менеджер ввода-вывода, предоставляющий средства ввода-вывода, независимые от устройств;
2) файловые системы - NT-драйверы, выполняющие файл-ориентированные запросы на ввод-вывод и транслирующие их в вызовы обычных устройств; сетевой редиректор и сетевой сервер - драйверы файловых систем, передающие удаленные запросы на ввод-вывод на машины сети и получающие запросы от них;
3) драйверы устройств NT executive - низкоуровневые драйверы, которые непосредственно управляют устройством; менеджер кэша, реализующий кэширование диска.

Исполнительная часть, в свою очередь, основывается на службах нижнего уровня, предоставляемых ядром (его можно назвать и микроядром) NT. В функции ядра входит:

1) планирование процессов;
2) обработка прерываний и исключительных ситуаций;
3) синхронизация процессоров для многопроцессорных систем;
4) восстановление системы после сбоев.

Ядро работает в привилегированном режиме и никогда не удаляется из памяти. Обратиться к ядру можно только посредством прерывания. Ядро расположено над уровнем аппаратных абстракций (Hardware Abstraction Level HAL), который концентрирует в одном месте большую часть машинно-зависимых процедур. HAL располагается между NT executive и аппаратным обеспечением и скрывает от системы такие детали, как контроллеры прерываний, интерфейсы ввода/вывода и механизмы взаимодействия между процессорами. Такое решение позволяет легко переносить Windows NT с одной платформы на другую путем замены только слоя HAL.

Отметим, что ОС Windows NT не является системой, основанной на концепции ядра в полном смысле этого слова. Микроядро NT, как указывалось выше, служит, главным образом, средством поддержки для переносимой основной части ОС - набора пользовательских сред. Концентрация машинно-зависимых программ внутри микроядра делает перенос NT на разнообразные процессоры относительно легким. Но в то время, как некоторые микроядра (Mach и Chorus) предполагается поставлять в качестве самостоятельного программного продукта, из операционной системы Windows NT ядро вряд ли может быть вычленено для отдельного использования. Это является одной из причин того, что некоторые специалисты не считают Windows NT истинно микроядерной ОС в том смысле, в котором таковыми являются Mach и Chorus. Те же критики отмечают также, что NT не исключает, как это положено, все надстроенные службы из пространства ядра и что драйверы устройств в NT по минимуму взаимодействуют с ядром, предпочитая работать непосредственно с лежащим ниже слоем аппаратной абстракции HAL.

При создании NT разработчики руководствовались задачами улучшения производительности и сетевых возможностей, а также требованием поддержки определенного набора прикладных сред. Эта цель была достигнута продуманным разделением функций между модулями ядра и остальными модулями. Например, передача данных в файловую систему и по сети производится быстрее в пространстве ядра, поэтому внутри ядра NT выделены буфера для небольших по объему (от 16 до 32 Кб) операций чтения и записи, являющихся типичными для приложений клиент-сервер и распределенных приложений. Размещение этих функций ввода-вывода внутри ядра может и портит академическую чистоту микроядра NT, но соответствует цели создания NT.

Защищенные подсистемы Windows NT создаются во время загрузки операционной системы. Сразу после создания они начинают бесконечный цикл своего выполнения, отвечая на сообщения, поступающие к ним от прикладных процессов и других подсистем. Среди этих подсистем можно выделить подкласс, называемый подсистемами окружения. Подсистемы окружения реализуют интерфейсы приложений операционной системы (API). Другие типы подсистем, называемые интегральными подсистемами, исполняют необходимые операционной системе задачи. Например, большая часть системы безопасности Windows NT реализована в виде интегральной подсистемы, сетевые серверы также выполнены как интегральные подсистемы.

Наиболее важной подсистемой окружения является Win32 - подсистема, которая обеспечивает доступ для приложений к 32-битным Windows API. Дополнительно эта система обеспечивает графический интерфейс с пользователем и управляет вводом/выводом данных пользователя. Также поддерживаются подсистемы POSIX, OS/2,16-разрядная Windows и MS-DOS.
Как уже отмечалось выше, каждая защищенная подсистема работает в режиме пользователя, вызывая системный сервис NT executive для выполнения привилегированных действий в режиме ядра. Но сетевые серверы могут выполняться как в режиме пользователя, так и в режиме ядра, в зависимости от того, как они разработаны.

Так как подсистемы связываются между собой путем передачи сообщений, то когда, например, пользовательское приложение вызывает какую-нибудь API-процедуру, подсистема окружения, обеспечивающая эту процедуру, получает сообщение и выполняет ее либо обращаясь к ядру, либо посылая сообщение другой подсистеме. После завершения процедуры подсистема окружения посылает приложению сообщение, содержащее возвращаемое значение. Посылка сообщений и другая деятельность защищенных подсистем невидима для пользователя.

Основным средством, скрепляющим все подсистемы Windows NT в единое целое, является механизм вызова локальных процедур (Local Procedure Call - LPC). LPC представляет собой оптимизированный вариант более общего средства - удаленного вызова процедур (RPC), которое используется для связи клиентов и серверов, расположенных на разных машинах сети.

Средства LPC поддерживают несколько способов передачи данных между клиентами и серверами: один обычно используется для передачи коротких сообщений, другой - для длинных сообщений, а третий оптимизирован специально для использования подсистемой Win32. Каждая подсистема устанавливает порт - канал связи, посредством которого с ней могут связываться другие процессы. Порты реализуются как объекты.
Windows NT использует защищенные подсистемы для того, чтобы:

1. Обеспечить несколько программных интерфейсов (API), по возможности не усложняя при этом базовый программный код (NT executive).

2. Изолировать базовую операционную систему от изменений или расширений в поддерживаемых API.
3. Объединить часть глобальных данных, требующихся всем API, и в то же время отделить данные, использующиеся каждым отдельным API от данных, использующихся другими API.
4. Защитить окружение каждого API от приложений, а также от окружений других API, и защитить базовую операционную систему от различных окружений.
5. Позволить операционной системе расширяться в будущем за счет новых API.

Таким образом, реализация частей ОС в виде серверов, выполняющихся в режиме пользователя, является важнейшей частью проекта Windows NT и оказывает глубокое воздействие на все функционирование системы.

### Множественные прикладные среды

При разработке NT важнейшим рыночным требованием являлось обеспечение поддержки по крайней мере двух уже существующих программных интерфейсов OS/2 и POSIX, а также возможности добавления других API в будущем.

Заметим, что для того, чтобы программа, написанная для одной ОС, могла быть выполнена в рамках другой ОС, недостаточно лишь обеспечить совместимость API. Кроме этого, необходимо обеспечить ей "родное" окружение: структуру процесса, средства управления памятью, средства обработки ошибок и исключительных ситуаций, механизмы защиты ресурсов и семантику файлового доступа. Отсюда ясно, что поддержка нескольких прикладных программных сред является очень сложной задачей, тесно связанной со структурой операционной системы. Эта задача была успешно решена в Windows NT.

Windows NT поддерживает пять прикладных сред операционных систем: MS-DOS, 16-разрядный Windows, OS/2 1.x, POSIX и 32-разрядный Windows (Win32). Все пять прикладных сред реализованы как подсистемы окружения.
16-битовые приложения DOS и Windows работают на VDM (virtual DOS machines - виртуальные машины DOS), каждая из которых эмулирует полный 80x86 процессор с MS-DOS. В NT VDM является приложением Win32, значит, как и обычные модули прикладных сред для UNIX, приложения DOS и 16-битовой Windows расположены в слое непосредственно над подсистемой Win32.

Подсистемы OS/2 и POSIX построены по-другому. В качестве полноценных подсистем NT они могут взаимодействовать с подсистемой Win32 для получения доступа к вводу и выводу, но также могут обращаться непосредственно к исполнительной системе NT за другими средствами операционной системы. Подсистема OS/2 может выполнять многие имеющиеся

приложения OS/2 символьного режима, включая OS/2 SQL Server, и поддерживает именованные каналы и NetBIOS.

Однако возможности подсистемы POSIX весьма ограничены, несмотря на непосредственный доступ ее к службам ядра. Приложения POSIX должны быть откомпилированы специально для Windows NT. NT не поддерживает двоичный код, предназначенный для других POSIX-совместимых систем, таких как UNIX. К тому же подсистема POSIX NT не поддерживает непосредственно печать, не поддерживает сетевой доступ, за исключением доступа к удаленным файловым системам, и не поддерживает многие средства Win32, например, отображение на память файлов и графику.

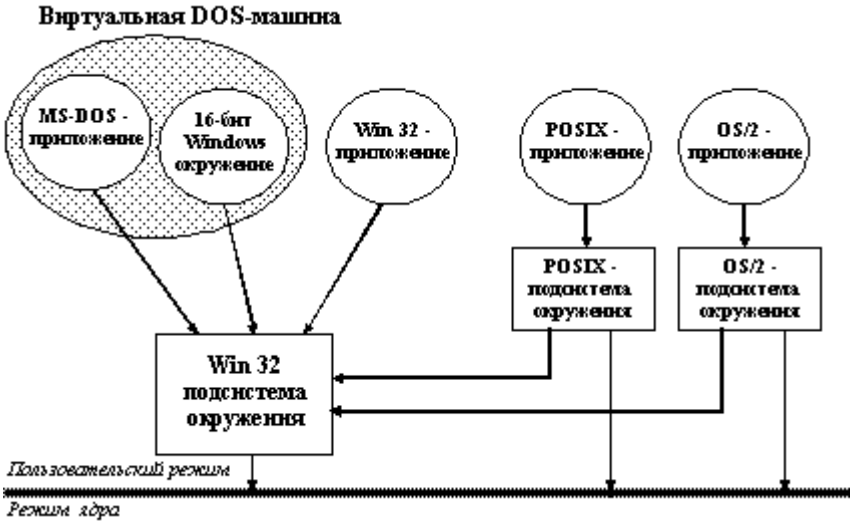


Рис. 11. Реализация множественных прикладных сред в Windows NT

На рис. 11 показана структура, обеспечивающая в Windows NT поддержку множественных прикладных сред.

NT executive выполняет базовые функции операционной системы и является той основой, на которой подсистемы окружения реализуют поддержку своих приложений. Все подсистемы равноправны и могут вызвать "родные" функции NT для создания соответствующей среды для своих приложений. Каждая подсистема окружения имеет свое представление о том, что такое, например, процесс или описатель файла, поэтому структуры данных, используемые в каждом окружении, могут не совпадать. Следовательно, как только подсистема Win32 передала прикладной процесс другой подсистеме окружения, данное приложение становится клиентом этой подсистемы вплоть до завершения процесса. При этом подсистема Win32 перенаправляет входные сообщения от пользователя этому приложению, а также отображает вывод приложения на экране.

## Производительность системы

ОС Windows NT является надежной операционной системой. Однако за надежность надо платить прежде всего за счет производительности. Когда программа обращается к функциям подсистемы (это может происходить по несколько сот раз в секунду), ОС необходимо выполнить переключение из вызывающего процесса в процесс подсистемы и обратно, что, естественно, заметно сказывается на производительности.

Хотя переходы между кольцами нежелательны, они неизбежны, так как прикладные программы, выполняемые в пользовательском режиме, используют средства вызова локальных процедур для обмена данными с подсистемами, а также потому, что подсистемам приходится обращаться к ядру, чтобы вызвать службы исполнительного модуля.

Еще одним недостатком размещения служб в подсистемах, работающих отдельно от прикладных программ, является то, что для передачи данных между процессами требуется память. Для каждого прикладного потока, который обращается к функциям, система создает соответствующий серверный поток в подсистеме, представляющий собой часть высокопроизводительного механизма обмена данными между процессами. Парные потоки повышают производительность, но для его организации требуется память, тем более, что каждой паре потоков выделятся совместный буфер памяти, доступный обоим процессам, который служит окном для пересылки данных.

Определим основные характеристики ОС Windows NT. Операционная система:

1) является истинно 32-разрядной и реентерабельной;
2) поддерживает вытесняющую многозадачность и работу с виртуальной памятью;
3) работает на разных аппаратных платформах;
4) хорошо масштабируется в системах с симметричной мультипроцессорной обработкой;
5) является распределённой вычислительной платформой, способной выступать в роли как клиента сети, так и сервера;
6) поддерживает большинство 16-разрядных приложений и 32-разрядных приложений других операционных систем;
7) отвечает требованиям к соответствию POSIX 1003.1;
8) отвечает требованиям к безопасности ОС;
9) поддерживает Unicode.

Однако эта операционная система является стратегической платформой на будущее – не только для серверов и офисных клиентских компьютеров, но и для домашних (бытовых) систем. Ниже перечислены некоторые архитектурные отличия и преимущества ОС Windows NT в сравнении с потребительскими версиями (ОС Windows 9x):

1. ОС поддерживает многопроцессорные системы, а потребительские версии Windows – нет.

2. Файловая система поддерживает защиту (например, управление избирательным доступом), отсутствующую в файловых системах потребительских версий.
3. ОС – полностью 32-разрядная система, которая не содержит 16-разрядный код кроме необходимого для поддержки 16-разрядных приложений. Потребительские версии содержат большой объем 16-разрядного кода.
4. ОС полностью реентерабельна, тогда как потребительские версии нереентерабельны (в основном, из-за 16-разрядного кода).

К нереентерабельному коду относится большинство функций управления графикой и окнами. Когда 32-разрядное приложение в потребительской версии пытается вызвать системный сервис в виде 16-разрядного нереентерабельного кода, ему сначала нужно использовать общесистемную блокировку (mutex), чтобы предотвратить вхождение других потоков в зону нереентерабельного кода. Более того, блокировка сохраняется на все время выполнения 16-разрядных приложений. В итоге, несмотря на то, что потребительские версии включают 32-разрядный планировщик, поддерживающий многопоточность и вытесняющий многозадачность, приложения часто выполняются как однопоточные.

Система предусматривает возможность выполнения 16-разрядных приложений в раздельных адресных пространствах, а потребительские версии выполняют такие приложения исключительно в выделенном им общем адресном пространстве, где они могут испортить не только свои данные, но и данные других 16-разрядных приложений.

Разделяемая память системы видима только тем процессам, которые проецируют ее на один и тот же раздел памяти. В Win32 такой раздел называется объектом проекции файла. В потребительский версиях вся разделяемая память видима и доступна для записи любому процессу, т.е. каждый процесс может осуществлять запись в любой объект проекции файла. В потребительских версиях несколько системных страниц памяти доступны коду пользовательских режима для записи, т.е. приложения могут повредить системные данные или даже привести к краху системы.

ОС не способна выполнять все старые 16-разрядные приложения, в отличие от потребительских версий.

## Операционная система Windows 2000

Операционная система Windows 2000 представляет собой продолжение линейки ОС Windows NT к которому программисты подключили практически все функции потребительских версий ( ОС Windows 9x), в частности, технологию РпР. Также в эту систему добавлено множество функций, связанных с мультимедиа. Однако по критерию надёжности Windows 2000 превосходит даже известную своей надёжностью Windows NT 4.0. Хотя ОС поддерживает различные файловые системы, его «родной» является NTFS 5,

которая несовместима с предыдущими версиями NTFS. Достаточно кардинально изменились сетевые настройки. Добавлены дополнительные протоколы к уже имеющимся основным. Однако Windows 2000 естественно требует гораздо больших системных ресурсов, как относительно потребительских версий, так и Windows NT 4.0, особенно система чувствительна к оперативной памяти. Также следует отметить, что ОС наиболее оптимально работает с приложениями, написанными под неё, и практически не может выполнять приложения Dos с прямым обращениям к аппаратуре, а также ряд других приложений.

## Выпуски Windows 2000

Существует четыре выпуска Windows 2000: Windows 2000 Professional, Windows 2000 Server, Windows 2000 Advanced Server, Windows 2000 Datacenter Server. Они различаются по следующим параметрам:

1) числу поддерживаемых процессоров;
2) объему поддерживаемой физической памяти;
3) возможному количеству одновременных сетевых соединений;
4) наличием в выпусках Server сервисов, не входящих в Professional.

На рисунке 12 представлены характеристики для основных выпусков ОС Windows 2000.

Во всех выпусках базовые системные файлы одинаковы: ядро, библиотеки, драйверы и основные системные утилиты. Однако некоторые из этих компонентов по-разному функционируют в зависимости от выпуска системы. Windows 2000 Server оптимизирован для работы в качестве высокопроизводительных серверов приложений, а Windows 2000 Professional, несмотря на поддержку серверных возможностей, предназначен для персональных систем. Если ядро во всех выпусках одинаково, то, как система определяет, какой именно выпуск загружается? Для этого она проверяет значение параметров ProductType и ProductSuite в разделе реестра HKLM\SYSTEM\CurrentControlSet\Control\ProductOptions. Параметр ProductType используется, чтобы отличить Windows 2000 Professional от Windows 2000 Server любого выпуска (см. рис. 12). Другой параметр, ProductSuite, позволяет различать Windows 2000 Server, а также определять, установленые службы терминала (только для серверных систем). В системах Windows 2000 Professional этот параметр пуст.

| Выпуск | Число процес-соров | Объем физической памяти, Гб | Количество одновремен-ных сетевых соединений | Дополнитель-ные сервисы |
|---|---|---|---|---|
| Windows 2000 Professional | 2 | 4 | 10 | Нет |
| Windows 2000 Server | 4 | 4 | не ограничено | Поддержка работы в качестве контроллера домена, служба ACTIVEDIRECTORY, сервер DHCP, сервер DNS, сервер DFS, службы сертификации, удаленная установка и службы терминала |
| Windows 2000 Advanced Server | 8 | 8 | не ограничено | Двухузловые кластеры |
| Windows 2000 Datacenter Server | 32 | 64 | не ограничено | Четырехузло-вые кластеры |

Рис. 12. Характеристики Windows 2000 для различных выпусков

| Выпуск Windows 2000 | Значение ProductType |
|---|---|
| Windows 2000 Professional | WinNT |
| Windows 2000 Server (контроллер домена) | LanmanNT |
| Windows 2000 Server (только сервер) | ServerNT |

Рис. 13. Значения параметров для различных выпусков ОС

Дальнейшие выпуски ОС Windows NT были выпущены для работы или строго в качестве высокопроизводительных серверов приложений (Windows 2003, Windows 2010), а Windows 7, Windows 8, Windows XP и т.д., несмотря на поддержку серверных возможностей, предназначены исключительно для персональных систем

**Особенности развития операционных систем Windows. Концепции Windows Runtime (WinRT) и Metro UI**

Операционные системы семейства Windows, получившие тотальное распространение с середины девяностых годов, планомерно развивались до середины «нулевых», когда персонализация вычислений получила новый мощный толчок – настоящую мобильность и доступность.

В то же время популярность стал завоевывать touch-интерфейс, когда пользователь работает с приложениями с помощью прикосновений и жестов, непосредственно касаясь пальцами экрана. Другим существенным сдвигом, в огромной степени повлиявшим на умы разработчиков приложений, стало следствие тотальной мобилизации вычислений – упрощение приложений. Пользователям стало необходимо большее количество приложений. Но, как следствие среды и размера устройств, все эти приложения становились простыми, чаще всего направленными на потребление контента или игру. Другими словами, без уменьшения спроса на приложения для созидания, появилась огромная, существенно большая потребность в приложениях для потребления. Одним из характерных явлений стала популярность магазинов приложений. Эта простая идея решила две важнейшие задачи: для пользователя – где взять приложение, для разработчика – где его продавать. Разработчики Windows, еще задолго до появления iPad, начали проект по разработке нового поколения флагманской операционной системы, где решались следующие задачи:

1. Увеличение времени автономной работы тремя путями – общей оптимизацией, переносом на другие архитектуры (ARM) и, наконец, выработкой правил для разработчиков, когда приложение минимизирует потребляемые ресурсы.
2. Оптимизация под touch-интерфейс.

3. Создание экосистемы по продаже и продвижению приложений.

ОС Windows 8 и ее модификации – большая возможность для разработчиков. Ведь у каждого пользователя Windows появляется Windows Store – магазин приложений, который является основным (а в некоторых случаях – единственным) источником приложений для его компьютера, будь то десктоп, лэптоп или планшет.

## Что такое Metro

Основные изменения для пользователя и разработчика в Windows 8 можно описать одним словом – METRO. Этот термин включает в себя две новых концепции: Windows Runtime (WinRT) и Metro UI. Говоря «metro-приложения», мы подразумеваем не только новый интерфейс, но и новую платформу разработки.

Что же представляет собой приложение в стиле metro. Это сплав новой парадигмы интерфейса, нового эффективного API и соответствующей платформы разработки, плюс интеграция с системой и другими приложениями.

У приложений имеется одно окно с поддержкой нескольких представлений

В отличие от классических приложений приложения в стиле Metro содержат одно окно без хрома, по умолчанию занимающее весь экран, чтобы пользователя не отвлекали лишние детали.


Рис. 14. Окно приложения в стиле Metro

Приложение в стиле Metro может поддерживать различные компоновки и представления, чтобы обеспечить динамичное и удобное обслуживание пользователя при различных параметрах конструкции и размерах экрана устройств.

Рис. 15.Различные параметры конструкции и размерах экрана устройств

Приложения в стиле Metro работают с различными источниками входных данных, включая перо, мышь, клавиатуру и сенсорный ввод. Для всех этих источников входных данных используется единый набор событий. В приложения в стиле Metro включается набор стилей по умолчанию, гарантирующий нормальную работу элементов пользовательского интерфейса с сенсорным вводом.

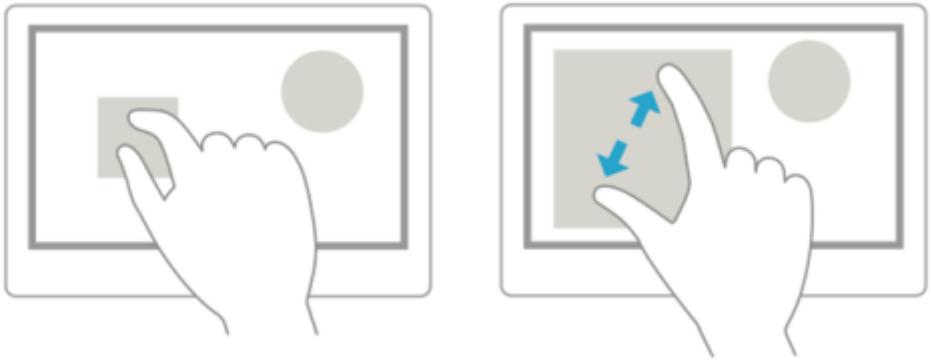

Рис. 16. Сенсорный ввод

Приложения могут взаимодействовать друг с другом. Контакты приложений — это возможность для пользователей с легкостью выполнять поиск по нескольким приложениям и обмениваться содержимым между ними. Они расширяют практическую полезность приложения, устраняя необходимость взаимодействия с различными стандартами API для конкретных приложений, когда нужно получить доступ к данным, хранящимся в другом приложении или созданным им. При этом пользователь остается в среде.

В приложениях доступны новые элементы управления в области интерфейса. Приложения в стиле Metro содержат несколько новых элементов управления, упрощающих организацию эффективного взаимодействия с пользователями. Среди этих элементов можно выделить строку команд (App Bar) приложения и чудо-кнопки (Charms).

## Строка команд приложения(App Bar)

Строка команд приложения размещается вне окна приложения и является его основным командным интерфейсом. Строка команд используется для размещения элементов навигации, команд и инструментов для пользователей. По умолчанию строка команд приложения скрыта и появляется, когда пользователь проводит пальцем в направлении от верхнего или нижнего края экрана. Она является обложкой для содержимого приложения и может быть скрыта пользователем, если он проведет пальцем по краю экрана или будет взаимодействовать с приложением.

## Чудо-кнопки (Charms)

Чудо-кнопки — это определенный и согласованный набор кнопок, используемый во всех приложениях для поиска, отправки, подключения, настройки и запуска. Пользователи могут:

1. Искать содержимое, размещенное в своем или другом приложении. Кроме того, поиск содержимого в приложении возможен из другого приложения.
2. Делиться содержимым из приложения с контактами или службами.
3. Перейти прямо на Начальный экран.
4. Подключаться к устройствам и отправлять содержимое, выполнять потоковую передачу мультимедиа, а также печатать документы.
5. Использовать параметры для настройки приложения по своему усмотрению.

Приложения используют плитки вместо значков (Tiles vs. Icons). Когда пользователь устанавливает приложение, оно появляется на начальном экране как плитка. Если коснуться или щелкнуть плитку, приложение будет запущено.


Рис. 17. Текущий экран Windows 8

Приложение может предоставлять содержимое с помощью плитки, даже когда не работает. Используя динамические плитки (Live Tiles), приложение способно выводить на экран полезные данные в краткой форме при минимальном расходе заряда батареи устройства. Приложение может настроить систему для периодического запроса обновлений в веб-службе вне зависимости от того, работает ли оно. Кроме того, в приложениях доступна настройка служб push-уведомлений Windows (WNS), чтобы отправлять сообщения напрямую из веб-службы в динамическую плитку.

## WinRT

До недавнего времени разработчики под Windows использовали две основных группы API - native через Win32 API и managed через .NET Framework. При этом вторая группа постепенно развивалась, получая различные модели создания пользовательского интерфейса, работы с данными и сервисами, построения исходного кода и архитектуры приложений. Между тем, сама Windows – то есть Win32 API получала не так много настоящих толчков к развитию базовой модели разработки. Последним существенным явлением был COM, но эти годы сами компьютеры не стояли на месте. Появлялись всевозможные сенсоры, сетевые устройства (включая 3G/LTE и т.п.), камеры, чувствительные к прикосновениям экраны. Наконец, энергопотребление становилось все более важным. Таким образом, возникла необходимость разработать и новый API, который будучи родным (native) для операционной системы, станет отвечать неким минимальным требованиям и веяниям времени. Windows Runtime (WinRT).

Особенности WinRT:

1. Это native API, работающий вместе с оптимизированным COM API. При этом возможна работа manage-сред поверх WinRT (и .NET Framework в Windows 8 – тому пример).
2. Это объектно-ориентированный API, включающий универсальный формат метаданных типов. WinRT включает поддержку многих современных свойств персональных компьютеров (сенсоры, камеры и т.п.), а энергосбережение является тут чрезвычайно важным.
3. Это языконезависимый API, изначально поддерживающий различные модели: C++, NET Framework, C# и Visual Basic, HTML5 и JavaScript.
4. Современная декларативная модель разработки интерфейсов на XAML или HTML5 стала частью Windows API, а не надстройкой над ним.

В результате приложения под Windows разделились на 2 группы:

1. Классические desktop-приложения.
2. Новые metro-приложения.

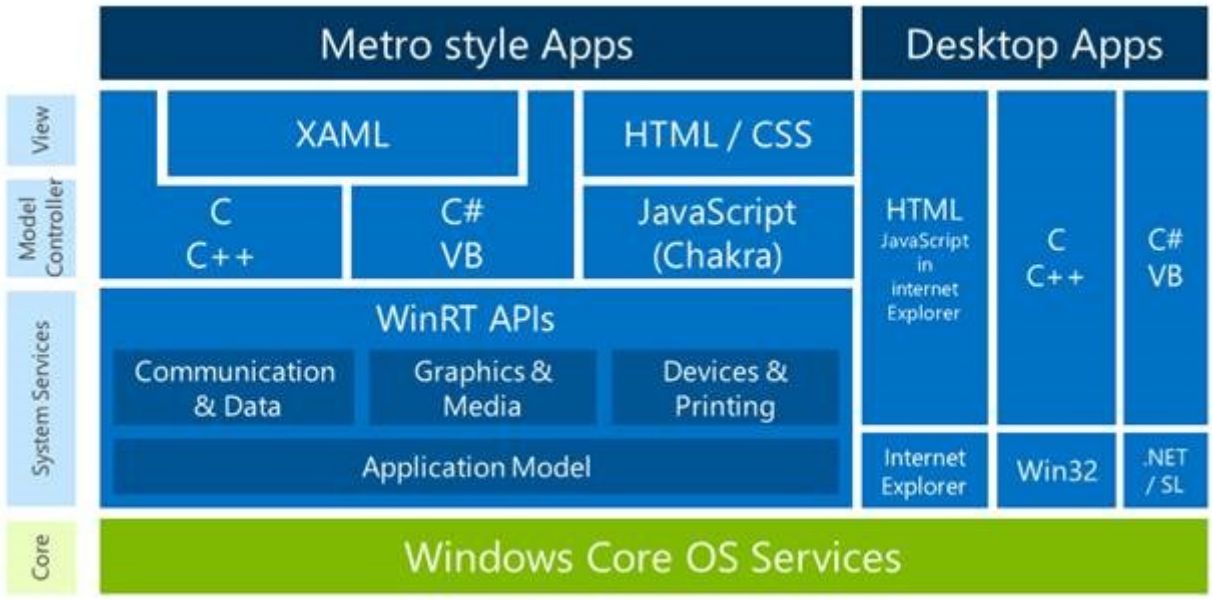


Рис. 18. Группы приложений Windows

Сам WinRT помимо собственно функционала, отвечающего за выполнение тех или иных функций, получил ряд важных компонентов.

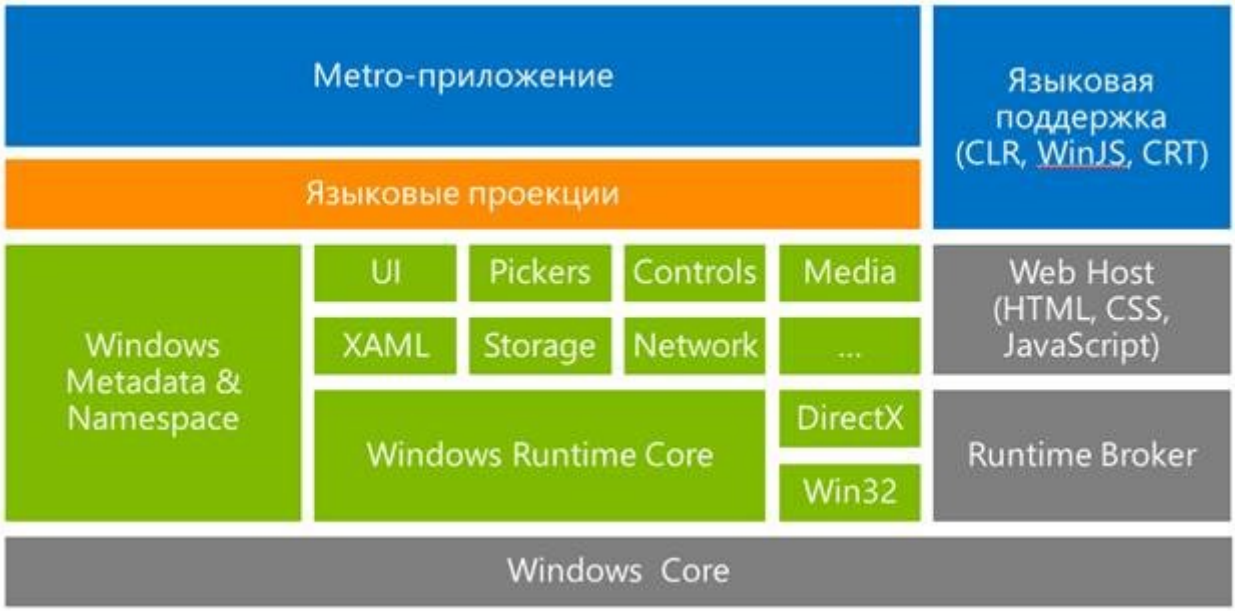


Рис. 19. Компоненты WinRT

Этими компонентами стали:

1. Подсистема метаданных и языковых проекций. Благодаря ей компоненты, написанные на разных языках программирования, могут вызывать друг друга.
2. Брокеры. Эти компоненты несут две важные функции - поддержку обмена данными между приложениями и защиту некоторых важных вызовов через механизм capabilities и явных разрешений.

Есть также еще две важные части Windows API, доступные и используемые metro-приложениями - Application Model и Windows Store API. Последняя группа управляет важными составляющими приложения:

1. Составом компонентов.
2. Возможностями (capabilities), затрагивающим безопасность и функционирование системы, такими как, доступ к местоположению, файловой системе, сети, идентификации пользователя, всевозможным точкам интеграции с системой и так далее.
3. Взаимодействием с Windows Store и информацией о приложении, которое, как мы уже знаем, в подавляющем большинстве случаев будет установлено именно оттуда.

Следует отметить, что DirectX и подмножество Win32 и COM API, доступны для компонентов и приложений, разрабатываемых на C++. Они необходимы, с одной стороны, разработчикам игр и мультимедиа-приложений (DirectX), с другой стороны, - для импортирования существующего кода (подмножество Win32).

## Модель выполнения приложений в WinRT

Отдельно необходимо остановиться на модели выполнения или жизни приложения в Windows 8 и WinRT. Цикл жизни metro-приложения существенно отличается от такового для приложений классических. Его можно разделить на три группы:

1) установка;
2) выполнение;
3) удаление.

С первым и третьим все более-менее понятно, хотя Windows Store дает дополнительные возможности. Но вот цикл выполнения сам по себе требует отдельного внимания. Стадии выполнения приложения точно описываются перечислением ApplicationExecutionState.

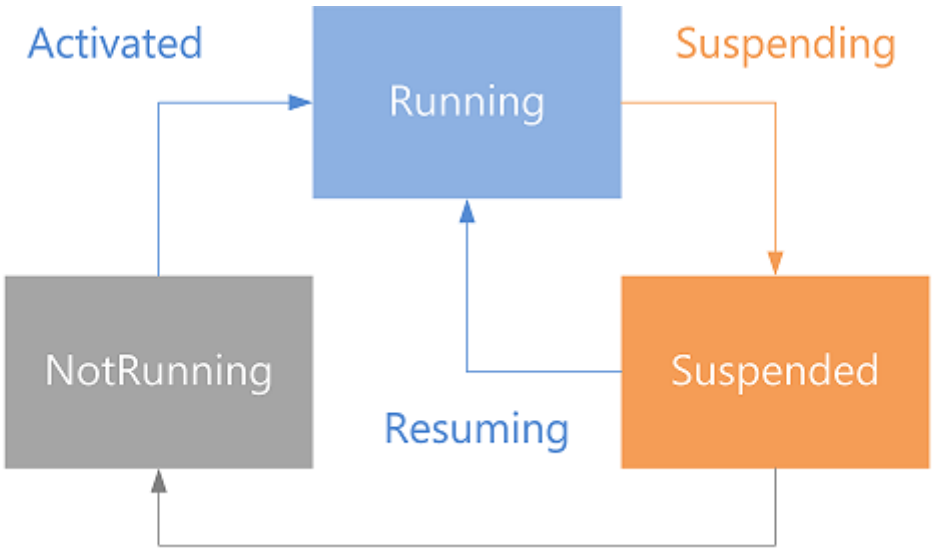


Рис. 20. Стадии выполнения приложения

Если пользователь не работает прямо сейчас с приложением, не работает и само приложение. Оно либо спит (suspended), либо и вовсе выгружается. Однако это не означает, что в WinRT невозможно выполнение фоновых процессов.

**Файловая система и данные приложений в WinRT**

Еще одной особенностью WinRT является порядок работы с файлами и данными. В общем случае данные metro-приложений изолированы друг от друга. Это обеспечивает стабильность и безопасность как самих приложений, так и системы в целом. У всякого metro-приложения есть три типа файлового хранилища:

1) локальное;
2) временное;
3) переносимое (roaming).

Первый тип, как следует из названия, содержит файлы и папки, которые приложение создает на локальном компьютере в специальной области, выделенной этому приложению. Второй тип очень похож на первый с тем исключением, что как только приложение выгружается, система может в любой момент удалить такие временные данные. Третий тип хранилища – roaming, представляет собой файлы и папки, синхронизируемые между всеми устройствами с Windows 8, где зарегистрировался тот же пользователь, который в приложении изначально создал данные. Пользователь определяется по Microsoft ID, который также все знают под именем LiveID.

## Файловая система и разрешения

Для получения приложением доступа к файлам, находящимся на всем остальном пространстве дисков устройства предназначено два механизма: File Pickers и Libraries. Первый механизм позволяет получить доступ к файлам через запрос пользователя на выбор таких файлов. Он их явно выбирает, давая таким образом доступ к этим файлам приложению. Второй механизм дает прямой доступ к файлам *библиотек документов* Windows: Documents, Pictures, Videos и т.д. Этот подход требует соблюдения особых условий.

## Универсальные приложения

Операционные системы семейства Windows первоначально разрабатывались для персональных компьютеров, однако, в связи с развитием различных устройств и операционных систем для них Microsoft также разрабатывает операционную систему для данной линейки, которую можно использовать для различных устройств, к примеру, смартфонов, игровых приставок X-box, планшетов, настольных компьютеров. Поэтому возникает необходимость разработки универсальных приложений. Рассмотрим особенности данных приложений.

Приложения для Windows специально ориентированы на пользователя, который их запускает. В них учтены специальные возможности аппаратного обеспечения, к тому же они могут адаптироваться к контексту, в котором исполняются. Они масштабируются в соответствии с множеством вариантов разрешения и ориентации экрана. Эти приложения без проблем поддерживают управление с помощью мыши и клавиатуры, когда недоступны возможности сенсорного дисплея. Они могут опираться на показания датчиков при определении местоположения пользователя и реагируют на перемещения устройства, на котором выполняются.

Концепция универсального приложения важна в первую очередь для разработчиков. При создании приложений с общим кодом для Windows и Windows Phone приходится использовать разделяемую переносимую библиотеку для выделения общего кода. Также из-за разницы в API Windows 8 и Windows Phone приходилось часть кода делать платформенно-зависимым.
Ранее Windows Phone во многом строилась на Silverlight, теперь платформа приложений для Windows исполняется с помощью WinRT. С использованием техники, которая называется проекцией, WinRT отображает системные прикладные программные интерфейсы на объекты выбранного языка программирования. Скомпилированный код вызывает API напрямую, как если писать программу на низкоуровневом неуправляемом языке C или C++. Здесь не применяется промежуточная трансляция или отображение, API транслируется напрямую в ОС Windows, получая непосредственный доступ к

возможностям операционной системы. В отличие от API Win32, WinRT — это объектно-ориентированный прикладной программный интерфейс. API WinRT предоставляется с помощью той же техники, что и в среде .NET. Общеязыковая инфраструктура (Common Language Infrastructure, CLI) призвана снабжать метаданными о прикладных программных интерфейсах, которые компилятор может использовать для проецирования их сигнатур в выбранный язык и для компиляции в машинный код. Метаданные хранятся в файлах с расширением .winmd (Windows Metadata — метаданные Windows).

Ранее разработчикам приходилось писать два приложения под две разные платформы, даже если работали они одинаково. Теперь можно делать единый проект для смартфонов и больших устройств (планшетов, ноутбуков и ПК). Это означает, что около большая часть системных вызовов между Windows выше 8.1 и Windows Phone являются общими. В среде разработки Visual Studio 2013 Update 2 появился новый шаблон проекта для унифицированных приложений Windows. Этот шаблон создает различные проекты для Windows и Windows Phone, и третий «разделяемый» проект, в котором размещается весь общий код. При этом «разделяемый» проект может содержать не только код, но и XAML-разметку, общие ресурсы, изображения и т.д. Этот проект не компилируется в отдельную библиотеку, а разделяется между двумя платформенными проектами на уровне текстового включения на этапе компиляции. Такой шаблон можно использовать для разработки на C#/XAML, C++/XAML или HTML/JS. Конечным результатом являются две сборки приложения — одна для смартфонов, другая для планшетов и ПК. В магазинах Windows и Windows Phone будут использоваться единые идентификаторы приложений, что позволит реализовать сценарии единой покупки приложения для использования на всех платформах.

WinRT — это единая среда, позволяющая писать код на совершенно разных языках и в рамках разных концепций. Платформой поддерживаются фреймворки XAML, HTML и DirectX, что дает возможность создавать код в:

1. C++ (с дополнениями Metro Extentions);
2. C#, vb.NET — с разметкой XAML;
3. HTML5 и Javascript — с дополнениями для WinRT API.

Модули, написанные с помощью разных фреймворков и на разных языках, могут совмещаться в рамках одного приложения. В нем можно создавать универсальные параметры, которые будут действовать для всех частей, написанных на разных языках. WinRT позволяет работать на аппаратных платформах x86 и ARM.

### Интерфейс универсальных приложений

С технической точки зрения у нового интерфейса WinRT есть несколько новых особенностей. Например, аппаратное ускорение отрисовки интерфейса, позволяющее разгрузить процессор. Собственно, оно появилось еще в Windows 7 — это компонент Direct2D, который заменяет GDI+. В WinRT аппаратное

ускорение задействовано всегда. К тому же система может не перерисовывать статичные участки интерфейса.

Сейчас Microsoft предлагает новую концепцию интерфейса для универсальных приложений WinRT. Ее суть, в том, что интерфейс приложения должен учитывать класс устройства, то есть параметры экрана (размер, ориентацию и пр.) и некоторые другие особенности. В зависимости от этого может меняться внешний вид интерфейса, объем выводимой на экран информации и схема взаимодействия с пользователем.

Таким образом, универсальные приложения должны одинаково работать на любом устройстве с платформой WinRT, однако их интерфейс может различаться в зависимости от класса устройства, размера экрана и других параметров. С одной стороны, делать несколько интерфейсов — дополнительная работа. С другой — такой подход позволяет обеспечить большее удобство работы с приложением на любом устройстве.

**Выводы**

1. ОС Windows NT является истинно 32-разрядной и реентерабельной системой, поддерживающей вытесняющую многозадачность и работу с виртуальной памятью, работает на разных аппаратных платформах.
2. Система реализована в виде распределённой вычислительной платформы, способной выступать в роли как клиента сети, так и сервера.
3. В структуре Windows NT использована концепция микроядра и модель клиент-сервер.
4. Windows NT поддерживает пять прикладных сред операционных систем: MS-DOS, 16-разрядный Windows, OS/2 1.x, POSIX и 32-разрядный Windows (Win32), которые реализованы как подсистемы окружения.
5. Основные изменения для пользователя и разработчика в Windows 8 – METRO. Этот термин включает в себя две новых концепции: Windows Runtime (WinRT) и Metro UI.
6. Приложение в стиле metro – это сплав новой парадигмы интерфейса, нового эффективного API и соответствующей платформы разработки, а также интеграция с системой и другими приложениями.

## Лекция 16. Операционная система UNIX SYSTEM V, RELEASE 4.2. Ядро системы. Режим пользователя и режим ядра. Процессы. Основные утилиты UNIX SYSTEM V

ОС UNIX – одна из старейших (впервые она появилась в1969 году) операционных систем, являющихся одновременно многопользовательской и многозадачной со встроенной сетевой поддержкой, написанной буквально для любых платформ.

UNIX зародился в лаборатории Bell Labs фирмы AT&T более 20 лет назад. Но она уже включала характерную для UNIX файловую систему, основанную на индексных дескрипторах inode, имела подсистему управления

процессами и памятью, а также позволяла двум пользователям работать в режиме разделения времени. Система была написана на ассемблере. Первоначальное имя UNIX (Uniplex Information and Computing Services) UNICS, подчеркивало её отличие от многопользовательской MULTICS. Вскоре UNICS начали называть UNIX.

Большое влияние на судьбу UNIX оказала перепись ее на языке высокого уровня C, разработанного специально для этих целей. Это произошло в 1973 году, UNIX насчитывал к этому времени уже 25 инсталляций, и в Bell Labs была создана специальная группа поддержки UNIX.

UNIX получил широкое распространение в университетах, так как для них он поставлялся бесплатно вместе с исходными кодами на C. Широкое распространение эффективных C-компиляторов сделало UNIX уникальной для того времени ОС из-за возможности переноса на различные компьютеры. Университеты внесли значительный вклад в улучшение UNIX и дальнейшую его популяризацию.

Однако распространение UNIX породило проблему несовместимости его многочисленных версий. Периодически делались и делаются попытки стандартизации UNIX, но они пока имели ограниченный успех. 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 список этот можно еще долго продолжать.

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:

1. Многопользовательский режим со средствами защиты данных от несанкционированного доступа.
2. Реализация мультипрограммной обработки в режиме разделения времени, основанная на использовании алгоритмов вытесняющей многозадачности (preemptive multitasking).
3. Использование механизмов виртуальной памяти и свопинга для повышения уровня мультипрограммирования.
4. Базовые объекты: процесс и файл. Есть некоторые унифицированные средства формирования процессов.

5. Унификация операций ввода-вывода на основе расширенного использования понятия "файл". Грамотно организована работа с внешними устройствами: байт-ориентированные и блок-ориентированные ВУ.
6. Иерархическая файловая система, образующая единое дерево каталогов независимо от количества физических устройств, используемых для размещения файлов. Вся системная информация хранится в текстовых файлах.
7. Переносимость системы за счет написания её основной части на языке С.
8. Разнообразные средства взаимодействия процессов, в том числе и через сеть.
9. Удобный интерфейс СИ и UNIXа. Можно создать свое программное окружение и систему команд.
10. Кэширование диска для уменьшения среднего времени доступа к файлам. ОС UNIX решает проблему сглаживания скорости ЦП и ОП, за счет глубокой многоуровневой буферизации и за счет оптимизации доступа к информации в ФС. Буферизация идет через КЭШ-буфера, следовательно, происходит минимальное количество обменов, снижение нагрузки на механику дисков.

Хотя операционная система и большинство команд написаны на Си, система UNIX поддерживает ряд других языков, таких как Фортран, Бейсик, Паскаль, Ада, Кобол, Лисп и Пролог. Система UNIX может поддерживать любой язык программирования, для которого имеется компилятор или интерпретатор, и обеспечивать системный интерфейс, устанавливающий соответствие между пользовательскими запросами к операционной системе и набором запросов, принятых в UNIX.

Далее мы подробно остановимся на основных концепциях версии UNIX System V Release 4, которая вобрала в себя лучшие черты линий UNIX System V и UNIX BSD.

**Ядро операционной системы**

На рис. 21 изображена архитектура верхнего уровня системы UNIX. Технические средства, показанные в центре диаграммы, выполняют функции, обеспечивающие функционирование операционной системы. Операционная система взаимодействует с аппаратурой непосредственно, обеспечивая обслуживание программ и их независимость от деталей аппаратной конфигурации. Если представить систему, состоящей из пластов, в ней можно выделить системное ядро, изолированное от пользовательских программ.

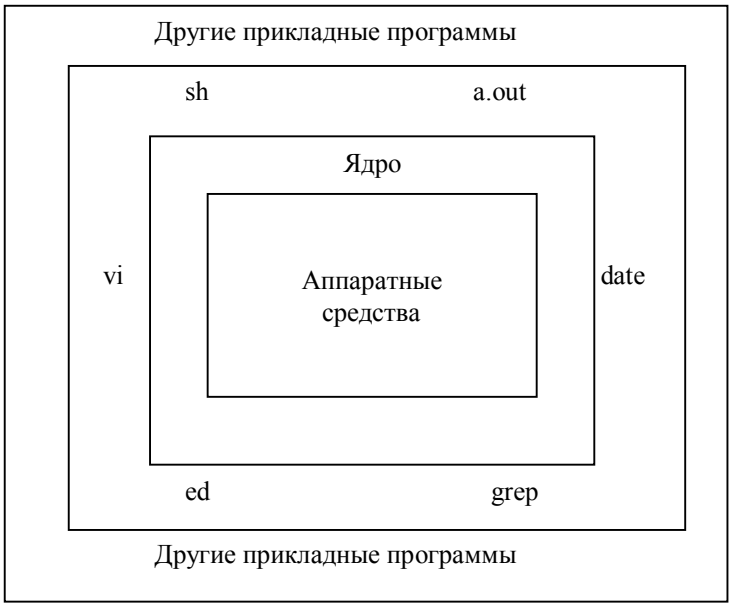


Рис. 21. Архитектура системы UNIX

Поскольку программы не зависят от аппаратуры, их легко переносить из одной системы UNIX в другую, функционирующую на другом комплексе технических средств, если только в этих программах не подразумевается работа с конкретным оборудованием. Например, программы, рассчитанные на определенный размер машинного слова, гораздо труднее переводить на другие машины по сравнению с программами, не требующими подобных установлений.

Программы, подобные командному процессору shell и редакторам (ed и vi) и показанные на внешнем по отношению к ядру слое, взаимодействуют с ядром при помощи хорошо определенного набора обращений к операционной системе. Обращения к операционной системе понуждают ядро к выполнению различных операций, которых требует вызывающая программа, и обеспечивают обмен данными между ядром и программой. Некоторые из программ, приведенных на рисунке, в стандартных конфигурациях системы известны как команды, однако, на одном уровне с ними могут располагаться и доступные пользователю программы, такие как программа a.out, стандартное имя для исполняемого файла, созданного компилятором с языка Си. Другие прикладные программы располагаются выше указанных программ, на верхнем уровне, как это показано на рисунке. Например, стандартный компилятор с языка Си располагается на самом внешнем слое: он вызывает препроцессор для Си, ассемблер и загрузчик (компоновщик), т.е. отдельные программы предыдущего уровня.

Хотя на рисунке приведена двухуровневая иерархия прикладных программ, пользователь может расширить иерархическую структуру на столько

уровней, сколько необходимо. На самом деле стиль программирования, принятый в системе UNIX, допускает разработку комбинации программ, выполняющих одну и ту же, общую задачу.

Многие прикладные подсистемы и программы, составляющие верхний уровень системы, такие как командный процессор shell, редакторы, SCCS (система обработки исходных текстов программ) и пакеты программ подготовки документации, постепенно становятся синонимом понятия "система UNIX". Однако все они пользуются услугами программ нижних уровней и, в конечном счете, ядра с помощью набора обращений к операционной системе. В версии V принято 64 типа обращений к операционной системе, из которых немногим меньше половины используются часто. Они имеют несложные параметры, что облегчает их использование, предоставляя при этом большие возможности пользователю. Набор обращений к операционной системе вместе с реализующими их внутренними алгоритмами составляют "тело" ядра. Короче говоря, ядро реализует функции, на которых основывается выполнение всех прикладных программ в системе UNIX, и им же определяются эти функции.
Ядро ОС - программа, функцией которой является управление базовыми объектами системы. Ядро в своем теле размещает необходимые таблицы данных. Ядро считается некоторой неразделяемой частью ОС. Оно обычно работает в режиме супервизора, все остальные функции ОС могут работать и в других режимах.

Ядро UNIX состоит из двух частей:

1. Управление процессами.
2. Управление устройствами.

Первая часть резервирует ресурсы, определяет последовательности выполнения процессов, принимает запросы на их обслуживание. Вторая - контроль за передачей данных между основной памятью и периферией.

В любой момент времени выполняется либо процесс пользователя, либо команда ОС, следовательно существует два режима: режим пользователя и режим ядра. Разделение этих режимов может поддерживаться как аппаратно, так и программно.

Среди функций ядра можно отметить:

1. Управление выполнением процессов посредством их создания, завершения или приостановки и организации взаимодействия между ними.
2. Планирование очередности предоставления выполняющимся процессам времени центрального процессора (диспетчеризация). Процессы работают с центральным процессором в режиме разделения времени: центральный процессор выполняет процесс, по завершении отсчитываемого ядром кванта времени процесс приостанавливается и ядро активизирует выполнение другого процесса. Позднее ядро запускает приостановленный процесс.

3. Выделение выполняемому процессу оперативной памяти. Ядро операционной системы дает процессам возможность совместно использовать участки адресного пространства на определенных условиях, защищая при этом адресное пространство, выделенное процессу, от вмешательства извне. Если системе требуется свободная память, ядро освобождает память, временно выгружая процесс на внешние запоминающие устройства, которые называют устройствами выгрузки. Если ядро выгружает процессы на устройства выгрузки целиком, такая реализация системы UNIX называется системой со свопингом (подкачкой); если же на устройство выгрузки выводятся страницы памяти, такая система называется системой с замещением страниц.
4. Выделение внешней памяти с целью обеспечения эффективного хранения информации и выборка данных пользователя. Именно в процессе реализации этой функции создается файловая система. Ядро выделяет внешнюю память под пользовательские файлы, мобилизует неиспользуемую память, структурирует файловую систему в форме, доступной для понимания, и защищает пользовательские файлы от несанкционированного доступа.
5. Управление доступом процессов к периферийным устройствам, таким как терминалы, ленточные устройства, дисководы и сетевое оборудование.

Выполнение ядром своих функций довольно очевидно. Например, оно узнает, что данный файл является обычным файлом или устройством, но скрывает это различие от пользовательских процессов. Так же оно, форматируя информацию файла для внутреннего хранения, защищает внутренний формат от пользовательских процессов, возвращая им неформатированный поток байтов. Наконец, ядро реализует ряд необходимых функций по обеспечению выполнения процессов пользовательского уровня, за исключением функций, которые могут быть реализованы на самом пользовательском уровне. Например, ядро выполняет действия, необходимые shell'у как интерпретатору команд: оно позволяет процессору shell читать вводимые с терминала данные, динамически порождать процессы, синхронизировать выполнение процессов, открывать каналы и переадресовывать ввод-вывод.

**Файловая система**

Файловая подсистема управляет файлами, размещает записи файлов, управляет свободным пространством, доступом к файлам и поиском данных для пользователей. Процессы взаимодействуют с подсистемой управления файлами, используя при этом совокупность специальных обращений к операционной системе, таких как open (для того, чтобы открыть файл на чтение или запись),close, read, write, stat (запросить атрибуты файла), chown (изменить запись с информацией о владельце файла) и chmod (изменить права доступа к файлу.

Подсистема управления файлами обращается к данным, которые хранятся в файле, используя буферный механизм, управляющий потоком

данных между ядром и устройствами внешней памяти. Буферный механизм, взаимодействуя с драйверами устройств ввода-вывода блоками, инициирует передачу данных к ядру и обратно.

Драйверы устройств являются такими модулями в составе ядра, которые управляют работой периферийных устройств. Устройства ввода-вывода блоками относятся к типу запоминающих устройств с произвольной выборкой; их драйверы построены таким образом, что все остальные компоненты системы воспринимают эти устройства как запоминающие устройства с произвольной выборкой.

Подсистема управления файлами также непосредственно взаимодействует с драйверами устройств "неструктурированного" ввода-вывода, без вмешательства буферного механизма. К устройствам неструктурированного ввода-вывода, иногда именуемым устройствами посимвольного ввода-вывода (текстовыми), относятся устройства, отличные от устройств ввода-вывода блоками.

В ОС файлы организованы в каталоги, но, в отличие от DOS, эта система более гибкая и управляемая. Т.к. Unix – многопользовательская система, предусматривается некоторая защита, запрещающая другим пользователям работу с файлами без специального разрешения конкретного пользователя. Поэтому каждый файл в системе имеет полномочия на чтение, запись и выполнение, определяющие, кто и что может с этим файлом делать. Файлы являются простыми, неструктурированными байтовыми последовательностями, поэтому записывать и считывать файлы сложной структуры должны сами программы. Файлы и каталоги считываются системой с помощью внутренних номеров – индексных дескрипторов. ОС отображает имена файлов и каталогов в их индексные дескрипторы. При этом система знает, как конкретно индексные дескрипторы соотносятся с теми или иными блоками или дорожками на диске. При вводе имени файла операционная система выполняет следующее:

1) для получения соответствующей таблицы используется маршрут;
2) находится введенное имя в таблице;
3) отмечается соответствующий индексный дескриптор;
4) полученный индексный дескриптор используется для поиска файла на диске;
5) в соответствии с командой происходит работа с данными файла.

Файлы Unix называются простыми или плоскими. Их можно рассматривать как простые байтовые или символьные потоки без какого-либо формата или структуры. Размеры такого файла могут превышать емкость любого диска.

Т.к. фрагмент находящихся в файле простых данных знает только свой уникальный индексный дескриптор, то каждое имя имеет свой уникальный индексный дескриптор. Однако несколько различных имен могут иметь одно и то же значение индексного дескриптора, т.е. может существовать несколько файлов (имен), представляющих одни и те же данные, и при изменениях в

одном файле автоматически изменяются значения другого. Существуют специальные команды, позволяющие связывать один файл с другими именами.

## Процессы

Подсистема управления процессами отвечает за синхронизацию процессов, взаимодействие процессов, распределение памяти и планирование выполнения процессов. Подсистема управления файлами и подсистема управления процессами взаимодействуют между собой, когда файл загружается в память на выполнение: подсистема управления процессами читает в память исполняемые файлы перед тем, как их выполнить.

Примерами обращений к операционной системе, используемых при управлении процессами, могут служить fork (создание нового процесса), exec (наложение образа программы на выполняемый процесс), exit (завершение выполнения процесса), wait (синхронизация продолжения выполнения основного процесса с моментом выхода из порожденного процесса), brk (управление размером памяти, выделенной процессу) и signal (управление реакцией процесса на возникновение экстраординарных событий).

Аппаратный контроль отвечает за обработку прерываний и за связь с машиной. Такие устройства, как диски и терминалы, могут прерывать работу центрального процессора во время выполнения процесса. При этом ядро системы после обработки прерывания может возобновить выполнение прерванного процесса. Прерывания обрабатываются не самими процессами, а специальными функциями ядра системы, перечисленными в контексте выполняемого процесса.

Как следует из описанного выше, основными компонентами Unix являются ядро, shell (оболочка) и утилиты. Shell – это интерфейс, обеспечивающий взаимодействия между ядром и пользователем. Она интерпретирует команды, вводимые пользователем, и посылает их в ядро. Интерфейс shell очень прост, он обычно состоит из приглашения, по которому пользователь вводит команды. Однако shell не только интерпретирует команды, но и создает среду, которую пользователь может конфигурировать и программировать, для чего в оболочке имеется свой язык программирования.

Каждому пользователю операционная система предоставляет свой собственный пользовательский интерфейс или shell, который модифицируется в соответствии с конкретными потребностями. В этом смысле shell пользователя работает как ОС, которой пользователь может управлять по своему усмотрению. Существует несколько разновидностей shell. В основном используются следующие три: Bourne, Korn и C-Shell. В качестве альтернативы командной строке применяется графический пользовательский интерфейс (GUI) X-Window. В этом интерфейсе имеется несколько программ управления окнами (менеджеры окон), которые позволяют пользователю управлять с помощью мыши окнами, пиктограммами и меню. Два самых популярных

менеджера окон следующие: Free Virtual Window Manager (FVWM) и Open Look Window Manager (OLWM).

**Утилиты**

Многие утилиты ОС можно отнести к трем категориям: редакторы, фильтры и коммуникационные программы, хотя есть еще и утилиты, выполняющие работу над файлами или управляющие кодом выполнения программ и т.д.

Во всех версиях Unix существует определенный набор стандартных редакторов, таких как: Ed, Ex – строковые редакторы; Vi, Vmax – экранные редакторы.

К фильтрам относятся утилиты, осуществляющие следующие операции:

1) считывание входной информации, поступающей либо от пользователя, либо из файла;
2) изучение и обработка входной информации;
3) вывод результатов.

Фильтры можно соединять между собой, подавая выходные данные одного фильтра на вход другого и т.д. Можно писать и собственные программы-фильтры, причем для этого существует язык программирования фильтров Awk.

Будучи многопользовательской операционной системой, Unix должна поддерживать контакт со всеми пользователями и отслеживать работу каждого из них. Наличие системы внутреннего контроля и особенности организации файловой системы обуславливают легкость реализации системы электронной почты. С ее помощью можно посылать сообщение любому пользователю системы и получать их, возможна даже одновременная рассылка нескольким пользователям.

Одной из важнейших черт системы является наличие инструментальных средств, предназначенных для работы в Internet. Эти сети создавались и развивались именно на основе Unix-систем. Такие программы, как FTP и Telnet, впервые были реализованы на BSD- версиях Unix.

В системах имеется полный набор инструментария Internet, а также средства прямого доступа к этой сети, например, программа Slip. В последние годы разработчиками предлагаются значительные усилия, расширяющие сетевые возможности Unix-систем.

**Выводы**

1. Операционная система семейства UNIX является системой, работающей в многопользовательском режиме со средствами защиты данных от несанкционированного доступа.
2. Реализация мультипрограммной обработки в режиме разделения времени основана на использовании алгоритмов вытесняющей многозадачности (preemptive multitasking).

3. В ОС используются механизмы виртуальной памяти и свопинг для повышения уровня мультипрограммирования.
4. Базовыми объектами являются процесс и файл.

## Литература

1. Дейтел Х.М., Дейтел П.Дж., Чофнес Д.Р. Операционные системы. - М.: Бином, 2006. – Ч. 1.
2. Дейтел Х.М. , Дейтел П.Дж., Чофнес Д.Р. Операционные системы. - М.: Бином. 2007. – Ч.2.
3. Сван Т. Программирование для Windows в Borland C++. - М: Бином, 1996.
4. Финогенов К.Г. C/C++:Win32. Основы программирование. - М: Диалог. - МИФИ, 2006.
5. Таненбаум Э. Современные операционные системы. - СПб: Питер, 2006.
6. Джонсон М. Харт. Системное программирование в среде WIN 32. - 2-е изд. - М. Вильямс, 2014.
7. Солдатов В.П. Программирование драйверов под Windows. - М.:Бином, 2004.
8. Олифер В.Г., Олифер Н.А. Сетевые операционные системы. - СПб: Питер, 2011.
9. www.microsoft.ru

Приложение

Листинг 1

После установки нового обработчика прерывания таймера основная программа ждет нажатия на клавиатуре любой клавиши, затем она восстанавливает старое содержимое вектора прерывания.

```
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>

// Выключаем проверку стека и указателей

#pragma check_stack( off )
#pragma check_pointer( off )

// Это макро используется для выдачи
//   сигнала на внутренний динамик
//   компьютера. Используется вывод
//   в формате TTY символа BELL (7)
//   через прерывание BIOS 10h

#define BEEP() _asm { \
            _asm mov bx,0      \
            _asm mov ax, 0E07h \
            _asm int 10h       \
                  }

void main(void);


// Объявление программы обработки прерывания

void _interrupt _far timer(void);

// Эта переменная предназначена для хранения
//   старого значения вектора прерывания
//   таймера. Она должна быть глобальной.

void (_interrupt _far *oldvect)(void);

// Переменная для подсчета тиков таймера
```

```
volatile long ticks;


void main(void) {

        ticks=0L;  // Сбрасываем счетчик тиков таймера

        oldvect = _dos_getvect(0x1c); // Запоминаем адрес
                                      // старого обработчика
                                      // прерывания

        _dos_setvect(0x1c, timer);    // Устанавливаем свой
                                      // обработчик

        printf("\nТаймер установлен. Нажмите любую"
                    "клавишу...\n");

        getch(); // Ожидаем нажатия на любую клавишу

        _dos_setvect(0x1c,oldvect);   // Восстанавливаем старый
                                      // обработчик прерывания
                                      // таймера
        exit(0);

}

// Функция обрабатывает прерывания таймера

void _interrupt _far timer(void) {

        ticks++; // Увеличиваем счетчик тиков таймера

        // Если значение счетчика тиков кратно 20,
        // выдаем сигнал на динамик компьютера
        if((ticks % 20) == 0) BEEP();

        // Вызываем старый обработчик прерывания
        _chain_intr(oldvect);
```