Профессор
Игорь Н. **Бекман**

КОМПЬЮТЕРНЫЕ НАУКИ

Курс лекций

Лекция 9. ОПЕРАЦИОННЫЕ СИСТЕМЫ

Содержание

Операционная система - комплекс программ, обеспечивающий выполнение других программ, распределение ресурсов, планирование, ввод-вывод данных; управление данными, взаимодействие с оператором. Операционную систему составляют: монитор, загрузчик, супервизор, планировщик и набор системных обслуживающих программ (утилит).

В данной лекции мы рассмотрим основные аспекты программного обеспечения компьютера, существующие операционные системы (в том числе – операционные системы реального времени). В заключительной части лекции мы более подробно остановимся на наиболее исзевестных операционных системах (Unix, Dos, Windows, OS/2, Linux и др.).

1. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОМПЬЮТЕРА

Програ́ммное обеспече́ние - совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ (ГОСТ 19781-90).

Программное обеспечение является одним из видов обеспечения вычислительной системы, наряду с техническим (аппаратным), математическим, информационным, лингвистическим, организационным и методическим обеспечением. В компьютерном сленге часто используется слово софт (от *software*), которое в этом смысле впервые применил в статье в *American Mathematical Monthly* математик из Принстонского университета Джон Тьюки в 1958.

Программное обеспечение представляет собой совокупность компьютерных инструкций. Оно охватывает программы, подпрограммы (разделы программы) и данные. Таким образом, программное обеспечение указывает компьютеру, что делать, как, когда, в какой последовательности и как часто. Нередко программное обеспечение называют просто программой. Компьютерные программы состоят из перечней команд, которые заставляют компьютер выполнять нужную работу. Компьютер должен получать исчерпывающие конкретные команды. Часто компьютерные программы имеют вид стенограммы.

Программное обеспечение (ПО) принято по назначению подразделять на **системное**, **прикладное** (пакеты прикладных программ, ППП) и **инструментарий технологии программирования**, а по способу распространения и использования на коммерческое, открытое и свободное. Свободное программное обеспечение может распространяться, устанавливаться и использоваться на любых компьютерах дома, в офисах, школах, вузах, а также коммерческих и государственных учреждениях без ограничений. По способу распространения (доставки, оплаты, ограничения в использовании) бывает: *Commercial Software, Freeware, Shareware, Abandonware, Adware, Free Software, Careware* и др. По назначению ПО разделяется на системное, прикладное и инструментальное.

В **системное ПО** входит: операционная система, ПО общего назначения, ПО реального времени, сетевая, встраиваемая, загрузчик операционной системы, драйвер устройства, программный кодек, утилита, и др. Системные программы имеют дело с взаимодействием между различными компонентами компьютера. Например, операционная система *Windows* представляет собой программу или набор программ, указывающих центральному процессору, как передавать данные и команды внутри процессора, между внутренней памятью компьютера, накопителем на диске и устройствами ввода-вывода, такими, как мониторы, принтеры, модемы, датчики и т.п. Она выполняет сервисные функции, такие, как отслеживание места хранения прикладных программ на гибком диске, с которым взаимодействует компьютер. Лучшие системные программы - это программы, которые позволяют компьютеру делать свою работу, не требуя от оператора, чтобы он был с ней знаком.

Программные средства защиты включают: криптошлюз, средство аутентификации, средство мониторинга и аудита, сканер защищённости, средство разграничения доступа, система криптографической защиты, шифрования и ЭЦП, антивирусная программа, антисапановая программа, межсетевой экран и др.

Инструментарий технологии программирования - совокупность программ, обеспечивающих технологию разработки, отладки и внедрения программных продуктов. Делится на два больших класса инструментальных средств: для создания отдельных приложений (программ) и для создания информационных систем и технологий. Средства для создания отдельных приложений включают локальные средства (языки программирования, системы программирования, инструментальные среды пользователя) и интегрированные среды разработки программ, основное назначение которых - повышение производительности труда программистов за счет автоматизации создания кодов программ, обеспечивающих интерфейс пользователя графического типа, а также автоматизации разработки запросов и отчетов (например, *Delphi*).

В свою очередь языки программирования делятся на следующие виды: **операторные**, которые используются для кодирования алгоритмов, а потому называются алгоритмическими, имеют в составе: машинно-зависимые (ассемблер), применяются для написания программ, явно использующих специфику конкретной аппаратуры; **машинно-ориентированные** (язык *C*, объединяющий идеи ассемблера и алгоритмического языка); **универсальные** (Турбо-Паскаль, Бэйсик), которые приближены максимально, насколько это возможно, к естественному английскому языку: название каждой команды – английское слово; **функциональные**, которые применяются для машинного моделирования той или иной проблематики и имеют в составе: *проблемно-ориентированные (GPSS)* - моделируют систему с помощью последовательности событий. Применяются, в частности, при проектировании вычислительных комплексов; *объектно-ориентированные* (Форт) - имеют встроенные средства для моделирования новых объектов программирования; логико-ориентированные (*Prolog*). Отдельно описываются правила предметной области, по которым затем выводятся новые факты.

Системы программирования включают: интегрированную среду разработчика программы, состоящую, в частности, из текстового редактора, позволяющего создавать и корректировать исходные тексты программ, средств поддержки интерфейса программиста с системными средствами для выполнения различных сервисных функций (например, сохранения или открытия файла); транслятор – программу, переводящую исходный текст во внутреннее представление компьютера; отладчик – программу для трассировки и анализа выполнения прикладных программ. Позволяет отслеживать выполнение программы в пооператорном режиме, идентифицировать место и вид ошибок в программе, наблюдают за изменением значений переменных, выражений и т.д.; компоновщик – программа для подготовки прикладной программы к работе в конкретных адресах основной памяти компьютера; справочные системы.

Инструментальная среда пользователя – это специальные программные средства, встроенные в ППП: библиотеки функций, процедур, объектов и методов обработки; макрокоманды; программные модули-вставки; конструкторы экранных форм и отчетов; языки запросов высокого уровня. Средства для создания информационных систем и технологий поддерживают полный цикл проектирования сложной информационной системы или технологии от исследования объекта автоматизации до оформления проектной и прочей документации на информационную систему или технологию. Они позволяют вести коллективную работу над проектом за счет возможности работы в локальной сети, экспорта – импорта любых фрагментов проекта, организации управления проектом. Инструментальное ПО включает: средство разработки программного обеспечения, среда разработки, *RAD, SDK*, система управления базами данных (СУБД), реляционная (*DB2, Informix*, *Interbase, Firebird, Microsoft SQL Server, MySQL, Oracle, Postgre, PostgreSQL*, ЛИНТЕР), объекто-ориентированная (*Cache*), иерархическая и сетевая.

Прикладная программа представляет собой набор команд для решения внешних задач, отличных от задач основной внутренней работы компьютера. Примером прикладной программы может служить программа обработки текстов или управления базой данных.

Пакет прикладных программ - комплекс взаимосвязанных программ для решения задач определенного класса. Выделяются следующие виды ППП: *проблемно-ориентированные*, которые используются для тех проблемных областей, в которых возможна типизация функций управления, структур данных и алгоритмов обработки. Например, это ППП автоматизации бухучета, финансовой деятельности, управления персоналом и т.д.; *автоматизации проектирования* (или САПР), которые используются в работе конструкторов и технологов, связанных с разработкой чертежей, схем, диаграмм; *общего назначения*, поддерживающие компьютерные технологии конечных пользователей и включают текстовые и табличные процессоры, графические редакторы, системы управления базами данных (СУБД); *офисные*, обеспечивающие организационное управление деятельностью офиса и включающие органайзеры (записные и телефонные книжки, календари, презентации и т.д.), средства распознавания текста и др.; *настольные издательские системы* – более функционально мощные текстовые процессоры; системы искусственного интеллекта, использующие в работе некоторые принципы обработки информации, свойственные человеку. Включают информационные системы, поддерживающие диалог на естественном языке; экспертные системы, позволяющие давать рекомендации пользователю в различных ситуациях; интеллектуальные пакеты прикладных программ, позволяющие решать прикладные задачи без программирования.

В прикладное ПО входит: офисное приложение, текстовый редактор, текстовый процессор, табличный процессор, редактор претензий, корпоротивная информационная система (аудиторская программа, бухгалтерская программа, системы: *MRP, MRP II, ERP, CRM, PO*), система *SCM*, система управления проектами (*Project Management*), система автоматизации документооборота (*EDM*), финансово-аналитическая система, система управления архивами документов (*DWM*), корпоративный портал, система проектирования и производства (система автоматизации проектных работ, САПР, CAD, системы: *CAE, CAM, PDM, PLM, SCADA, MES*), система логистической поддержки изделий (система анализа логической поддержки, *LSA*, система создания ИТЭР, (*IETM)*), система обработки и хранения медицинской информации (система передачи, обработки и архивации изображений, радиологическая информационная сеть, РИС, госпитальная информационная сеть (ГИС).

Научное ПО – это: система математического и статистического расчёта и анализа, система компьютерного моделирования.

Информационные системы: геоинформационная система, ГИС, система поддержки принятия решений, СППР, система управления *IT*-инфраструктурой, справочно-правовая система, СПС. Клиент для доступа к интернет-сервисам: электронная почта, Веб-браузер, система мгновенного обмена сообщениями, *IRC*, *IP*-телефония, пиринговая сеть, потоковое мультимедиа, банк-клиент, мультимедиа, компьютерная игра, музыкальный редактор, видео-редактор, аудиоредактор, медиа-проигрыватель.

Расширение производства и применения персональных компьютеров существенно ускорило разработку т. н. беспрограммного программного обеспечения. В этом случае пользователь может посредством управляющих элементов компьютера взаимодействовать с дисплеем, изображающим логическую или визуальную структуру некоторого вида. Пользователь может ввести с клавиатуры в любой точке дисплея необходимую информацию, а затем перейти к следующей точке. Эта структура может использоваться многократно (если необходимо, каждый раз с другими данными). Чтобы работать в такой программе, от пользователя не требуется знать что-либо о программировании. Многие программы электронных таблиц и баз данных обладают указанными беспрограммными характеристиками. Основные средства организации запросов в больших базах данных универсальных компьютеров базируются на *SQL* (язык структурированных запросов, «эскьюэль»), в котором пользователь запрашивает информацию из базы данных, используя синтаксис, во многом похожий на обычный английский. Стало популярным связывать *SQL* с базами данных персональных компьютеров.

Графические интерфейсы пользователя (ГИП). Компьютерные дисплеи прошли эволюцию от изображения, основанного на знаках, к экстенсивной растровой графике. Это развитие облегчило разработку программ - особенно операционных систем, включая графику и разнообразные изобразительные методы. Многие программы могут быть выведены на экран, а прикладные программы могут выполняться одновременно. Серия компьютеров «Макинтош», выпускаемых фирмой Эппл, положила начало широкому распространению ГИП для персональных компьютеров. Программы системы *Windows* (фирмы Майкрософт), *OS*/2 *Presentation Manager* (фирмы ИБМ), *New Wave* (фирмы Хьюлетт-Паккард) и большая часть программных средств для дисплеев рабочих станций используют ГИП.

2. ОПЕРАЦИОННАЯ СИСТЕМА

Действия, которые пользователь должен выполнить для того, чтобы «объяснить» компьютеру, что тот должен сделать – откуда взять данные, как их обработать и что выдать – зависят от того, с какими программами пользователь работает. Если говорить о взаимодействии не с конкретными программами, а с компьютером в целом, то речь идет о взаимодействии с операционной системой (ОС).

Операционная система – это комплекс управляющих программ, который обеспечивает согласованную работу всех компонентов компьютера для выполнения базовых команд. Операционные системы управляют файловой системой, обеспечивают ввод/вывод информации и передачу данных между разными устройствами и т. д.

Операцио́нная систе́ма, ОС (*operating system*) - базовый комплекс компьютерных программ, обеспечивающий интерфейс с пользователем, управление аппаратными средствами компьютера, работу с файлами, ввод и вывод данных, а также выполнение прикладных программ и утилит.

ОС позволяет абстрагироваться от деталей реализации аппаратного обеспечения, предоставляя разработчикам программного обеспечения минимально необходимый набор функций. С точки зрения обычных пользователей компьютерной техники ОС включает в себя и программы пользовательского интерфейса.

Основные функции (простейшие ОС): загрузка приложений в оперативную память и их выполнение; стандартизованный доступ к периферийным устройствам (устройства ввода-вывода); управление оперативной памятью (распределение между процессами, виртуальная память); управление доступом к данным на энергонезависимых носителях (таких как жёсткий диск, компакт-диск и т.д.), организованным в той или иной файловой системе; пользовательский интерфейс; сетевые операции, поддержка стека протоколов.

Дополнительные функции: параллельное или псевдопараллельное выполнение задач (многозадачность); взаимодействие между процессами: обмен данными, взаимная синхронизация; защита самой системы, а также пользовательских данных и программ от действий пользователей (злонамеренных или по незнанию) или приложений; разграничение прав доступа и многопользовательский режим работы (аутентификация, авторизация.).

Объекты ядра ОС: процессы и файлы.

События: потоки, семафоры, мьютексы, каналы, файлы, проецируемые в память.

Существуют две группы определений ОС: «совокупность программ, управляющих оборудованием» и «совокупность программ, управляющих другими программами». Обе они имеют свой точный технический смысл, который, однако, становится ясен только при более детальном рассмотрении вопроса о том, зачем вообще нужны операционные системы.

Есть приложения вычислительной техники, для которых ОС излишни. Например, встроенные микрокомпьютеры содержатся сегодня во многих бытовых приборах, автомобилях, сотовых телефонах и т. п. Зачастую такой компьютер постоянно исполняет лишь одну программу, запускающуюся по включении. И простые игровые приставки - также представляющие собой специализированные микрокомпьютеры - могут обходиться без ОС, запуская при включении программу, записанную на вставленном в устройство «картридже» или компакт-диске. Тем не менее, некоторые микрокомпьютеры и игровые приставки всё же работают под управлением особых собственных ОС. В большинстве случаев, это *UNIX*-подобные системы (последнее особенно верно в отношении программируемого коммутационного оборудования: файрволов, маршрутизаторов).

Операционные системы нужны, если вычислительная система используется для различных задач, причём программы, исполняющие эти задачи, нуждаются в сохранении данных и обмене ими. Из этого следует необходимость универсального механизма сохранения данных; в подавляющем большинстве случаев ОС отвечает на неё реализацией файловой системы. Современные ОС, кроме того, предоставляют возможность непосредственно «связать» вывод одной программы с вводом другой, минуя относительно медленные дисковые операции; различные программы нуждаются в выполнении одних и тех же рутинных действий. Например, простой ввод символа с клавиатуры и отображение его на экране может потребовать исполнения сотен машинных команд, а дисковая операция - тысяч. Чтобы не программировать их каждый раз заново, ОС предоставляют системные библиотеки часто используемых подпрограмм (функций); между программами и пользователями системы необходимо распределять полномочия, чтобы пользователи могли защищать свои данные от несанкционированного доступа, а возможная ошибка в программе не вызывала тотальных неприятностей; необходима возможность имитации «одновременного» исполнения нескольких программ на одном компьютере (даже содержащем лишь один процессор), осуществляемой с помощью

приёма, известного как «разделения времени». При этом специальный компонент, называемый планировщиком, «нарезает» процессорное время на короткие отрезки и предоставляет их поочередно различным исполняющимся программам (процессам); наконец, оператор должен иметь возможность, так или иначе, управлять процессами выполнения отдельных программ. Для этого служат операционные среды, одна из которых - оболочка и набор стандартных утилит - является частью ОС (прочие, такие, как графическая операционная среда, образуют независимые от ОС прикладные платформы). Таким образом, современные универсальные ОС можно охарактеризовать, прежде всего, как использующие файловые системы (с универсальным механизмом доступа к данным),
многопользовательские (с разделением полномочий), многозадачные (с разделением времени).

Многозадачность и распределение полномочий требуют определённой иерархии привилегий компонентов самой ОС. В составе ОС различают три группы компонентов: ядро, содержащее планировщик; драйверы устройств, непосредственно управляющие оборудованием; сетевую подсистему, файловую систему; системные библиотекм и оболочка с утилитами. Большинство программ, как системных (входящих в ОС), так и прикладных, исполняются в непривилегированном («пользовательском») режиме работы процессора и получают доступ к оборудованию (и, при необходимости, к другим ядерным ресурсам, а также ресурсам иных программ) только посредством системных вызовов. Ядро исполняется в привилегированном режиме: именно в этом смысле говорят, что ОС (точнее, её ядро) управляет оборудованием. В определении состава ОС значение имеет критерий операциональной целостности (замкнутости): система должна позволять полноценно использовать (включая модификацию) свои компоненты. Поэтому в полный состав ОС включают и набор инструментальных средств (от текстовых редакторов до компиляторов, отладчиков и компоновщиков).

Предшественником ОС следует считать служебные программы (загрузчики и мониторы), а также библиотеки часто используемых подпрограмм, начавшие разрабатываться с появлением универсальных компьютеров 1-го поколения (конец 1940-х). Служебные программы минимизировали физические манипуляции оператора с оборудованием, а библиотеки позволяли избежать многократного программирования одних и тех же действий (осуществления операций ввода-вывода, вычисления математических функций и т. п.). В 1950-60-х сформировались и были реализованы основные идеи, определяющие функциональность ОС: пакетный режим, разделение времени и многозадачность, разделение полномочий, реальный масштаб времени, файловые структуры и файловые системы.

Необходимость оптимального использования дорогостоящих вычислительных ресурсов привела к появлению концепции «**пакетного режима**» исполнения программ. Пакетный режим предполагает наличие очереди программ на исполнение, причём ОС может обеспечивать загрузку программы с внешних носителей данных в оперативную память, не дожидаясь завершения исполнения предыдущей программы, что позволяет избежать простоя процессора.

Уже пакетный режим в своём развитом варианте требует разделения процессорного времени между выполнением нескольких программ. Необходимость в разделении времени (**многозадачности**, мультипрограммировании) проявилась ещё сильнее при распространении в качестве устройств ввода-вывода телетайпов (а позднее, терминалов с электронно-лучевыми дисплеями) (1960-е годы). Поскольку скорость клавиатурного ввода (и даже чтения с экрана) данных оператором много ниже, чем скорость обработки этих данных компьютером, использование компьютера в «монопольном» режиме (с одним оператором) могло привести к простою дорогостоящих вычислительных ресурсов. Разделение времени позволило создать «многопользовательские» системы, в которых один центральный процессор и блок оперативной памяти соединялся с многочисленными терминалами. При этом часть задач (таких, как ввод или редактирование данных оператором) могла исполняться в режиме диалога, а другие задачи (такие, как массивные вычисления) – в пакетном режиме.

Распространение многопользовательских систем потребовало решения задачи **разделения полномочий**, позволяющей избежать возможности модификации исполняемой программы или данных одной программы в памяти компьютера другой (содержащей ошибку или злонамеренно подготовленной) программы, а также модификации самой ОС прикладной программой. Реализация разделения полномочий в ОС была поддержана разработчиками процессоров, предложивших архитектуры с двумя режимами работы процессора - «реальным» (в котором исполняемой программе доступно всё адресное пространство компьютера) и «защищённым» (в котором доступность адресного пространства ограничена диапазоном, выделенном при запуске программы на исполнение).

Применение универсальных компьютеров для управления производственными процессами потребовало реализации «реального масштаба времени» («**реального времени**») - синхронизации исполнения программ с

внешними физическими процессами; включение функции реального масштаба времени в ОС позволило создавать системы, одновременно обслуживающие производственные процессы и решающие другие задачи (в пакетном режиме и (или) в режиме разделения времени); постепенная замена носителей с последовательным доступом (перфолент, перфокарт и магнитных лент) накопителями произвольного доступа (на магнитных дисках).

Упомянем теперь существующие операционные системы: *Unix*, стандартизация ОС и *POSIX*.

К концу 1960-х годов отраслью и научно-образовательным сообществом был создан целый ряд ОС, реализующих все или часть очерченных выше функций. К ним относятся *Atlas* (Манчестерский университет), *CTTS* и *ITSS* (Массачусетский технологический институт, *MIT*), *THE* (Эйндховенский технологический университет), *RS*400 (Университет Орхуса) и др. (всего эксплуатировалось более сотни различных ОС). Наиболее развитые ОС, такие как *OS*/360 (*IBM*)), *SCOPE (CDC)* и завершённый уже в 1970-х *MULTICS* (МТИ и *Bell Labs*), предусматривали возможность исполнения на многопроцессорных компьютерах. Эклектичный характер разработки ОС привёл к нарастанию кризисных явлений, прежде всего, связанных с чрезмерными сложностью и размерами создаваемых систем. ОС были плохо масштабируемыми (более простые не могли использовать все возможности крупных вычислительных систем; более развитые неоптимально исполнялись на малых или не могли исполняться на них вовсе) и тотально несовместимыми между собой, их разработка и совершенствование затягивались.

Задуманная и реализованная в 1969 К. Томсоном при участии нескольких коллег (включая Д. Ричи и Б. Кернигана), ОС *Unix* (первоначально *UNICS*, что обыгрывало название *MULTICS*) вобрала в себя многие черты более ранних ОС, но обладала целым рядом свойств, отличающих её от большинства предшественниц: простая метафорика (два ключевых понятия: вычислительный процесс и файл); компонентная архитектура: принцип «одна программа - одна функция» плюс мощные средства связывания различных программ для решения возникающих задач («оболочка»); минимизация ядра (кода, выполняющегося в «реальном» (привилегированном) режиме процессора) и количества системных вызовов; независимость от аппаратной архитектуры и реализация на машиннонезависимом языке программирования (язык программирования Си стал побочным продуктом разработки *Unix*); унификация файлов. Unix, благодаря своему удобству прежде всего в качестве инструментальной среды (среды разработки), была тепло принята сначала в университетах, а затем и в отрасли, получившей прототип единой ОС, которая могла использоваться на самых разных вычислительных системах и, более того, могла быть быстро и с минимальными усилиями перенесена на любую вновь разработанную аппаратную архитектуру.

В конце 1970-х годов сотрудники Калифорнийского университета в Беркли внесли ряд усовершенствований в исходные коды *UNIX*, включая работу с протоколами *TCP/IP*. Их разработка стала известна под именем *BSD* (*Berkeley Software Distribution*). Задачу разработать независимую (от авторских прав *Bell Labs*) реализацию той же архитектуры поставил и Р. Столлмен, основатель проекта GNU. Благодаря конкурентности реализаций архитектура ОС *Unix* стала вначале фактическим отраслевым стандартом, а затем обрела статус и стандарта юридического – *ISO/IEC* 9945.

ОС, следующие стандарту или опирающиеся на него, называют «*POSIX*-совместимыми» (чаще встречается словоупотребление «*Unix* – подобные» или «семейство *Unix»*, но оно противоречит статусу торгового знака «*Unix*», принадлежащего консорциуму *The Open Group* и зарезервированному для обозначения ОС, строго следующих стандарту) благодаря названию стандарта - *POSIX*. Сертификация на совместимость со стандартом стоит некоторых денег, из-за чего некотрые системы не проходили этот процесс, однако считаются *POSIX*-совместимыми, просто потому что это так. К *Unix*-подобным ОС относятся системы, базирующиеся на последней версии *Unix*, выпущенной *Bell Labs* (*System* V), на разработках университета Беркли (*FreeBSD*, *OpenBSD, NetBSD*), а также ОС *GNU/Linux*, разработанная в части утилит и библиотек проектом *GNU* и в части ядра - сообществом, возглавляемым Л. Торвальдсом.

Стандартизация ОС гарантирует возможность безболезненной замены самой ОС и/или оборудования при развитии вычислительной системы или сети и дешёвого переноса прикладного программного обеспечения (строгое следование стандарту предполагает полную совместимость программ на уровне исходного текста; из-за профилирования стандарта и его развития некоторые изменения бывают всё же необходимы, но перенос программы между *POSIX*-совместимыми системами обходится на порядки дешевле, чем между альтернативными), а также преемственность опыта пользователей.

Самым заметным эффектом существования этого стандарта стало эффективное разворачивание Интернета в 1990-х годах.

Коротко остановимс теперь на «*Post Unix*»-архитектурах ОС.

Коллектив, создавший ОС Юникс, развил концепцию унификации объектов ОС, включив в исходную концепцию *UNIX* «устройство - это тоже файл» также и процессы, и любые другие системные, сетевые и прикладные сервисы, создав новую концепцию: «что угодно - это файл». Новый концепт получил название

*Plan*9 (название было позаимствовано из фантастического триллера «План 9 из открытого космоса» Эдварда Вуда-младшего), и сменил «рабочую лошадку» *UNIX System* V на компьютерах сети *Bell Labs* в 1992 году.

Кроме реализации всех объектов ОС в виде файлов и размещения их на едином и персональном для каждого терминала вычислительной сети пространстве (namespace), были пересмотрены другие архитектурные решения *UNIX*. Например, в *Plan*9 отсутствует понятие «суперпользователь», и, соответственно, исключаются любые нарушения режима безопасности, связанные с нелегальным получением прав суперпользователя в системе. Для представления (хранения, обмена) информации Р. Пайк и К. Томпсон разработали универсальную кодировку *UTF*-8, на сегодняшний день ставшую стандартом де-факто. Для доступа к файлам используется единый универсальный протокол 9*P*, по сети работающий поверх сетевого протокола (*TCP* или *UDP*). Таким образом для прикладного ПО сети не существует - доступ к локальным и к удалённым файлам единообразен. 9*P* - байт-ориентированный протокол, в отличие от других подобных протоколов, являющихся блок-ориентированными. Это также результат работы концепции: доступ побайтно - к унифицированным файлам, а не поблочно - к разнообразным и сильно изменяющимися с развитием технологий устройствам. Для контроля доступа к объектам не требуется иных решений, кроме уже существующего в ОС контроля доступов к файлам. Новая концепция системы хранения избавила администратора системы от изнурительного труда по сопровождению архивов и предвосхитила современные системы управления версиями файлов.

3. ОПЕРАЦИОННАЯ СИСТЕМА РЕАЛЬНОГО ВРЕМЕНИ

Операционная система реального времени, *ОСРВ* (*Real-Time Operating System*) - тип операционной системы. Это: 1) Операционная система, в которой успешность работы любой программы зависит не только от её логической правильности, но и от времени, за которое она получила этот результат. Если система не может удовлетворить временным ограничениям, должен быть зафиксирован сбой в её работе; 2) Реальное время в операционных системах - способность операционной системы обеспечить требуемый уровень сервиса в определённый промежуток времени»; 3) Операционная система, реагирующая в предсказуемое время на непредсказуемое появление внешних событий; 4) Интерактивные системы постоянной готовности. В категорию ОСРВ их относят, исходя из маркетинговых соображений, и если интерактивную программу называют «работающей в реальном времени», то это лишь означает, что запросы от пользователя обрабатываются с задержкой, незаметной для человека.

Операционные системы реального времени бывают двух типов - системы жесткого реального времени и системы мягкого реального времени. Операционная система, которая может обеспечить требуемое время выполнения задачи реального времени даже в худших случаях, называется операционной системой жёсткого реального времени. Операционная система может обеспечить требуемое время выполнения задачи реального времени в среднем, называется операционной системой мягкого реального времени.

Системы жёсткого реального времени не допускают задержек реакции системы, так как это может привести к: потере актуальности результатов; большим финансовым потерям; авариям и катастрофам. Если не выполняется обработка критических ситуаций либо она происходит недостаточно быстро, система жёсткого реального времени прерывает операцию и блокирует её, чтобы не пострадала надёжность и готовность остальной части системы. Примерами систем жёсткого реального времени могут быть - системы управления бортового оборудования, системы аварийной защиты, регистраторы аварийных событий.

Системы мягкого реального времени характеризуются возможностью задержки реакции, что может привести к увеличению стоимости результатов и снижению производительности системы в целом. Примером может служить работа компьтерной сети. Если система не успела обработать очередной принятый пакет, это приведет к остановке на передающей стороне и повторной посылке (в зависимости от протокола). Данные при этом не теряются, но производительность сети снижается.

Основное отличие системам жёсткого и мягкого реального времени можно охарактеризовать так: система жёсткого реального времени никогда не опоздает с реакцией на событие, система мягкого реального времени - не должна опаздывать с реакцией на событие. Обозначим операционной системой реального времени такую систему, которая может быть использована для построения систем жёсткого реального времени. Это определение выражает отношение к ОСРВ как к объекту, содержащему необходимые инструменты, но также означает, что эти инструменты ещё необходимо правильно использовать. Большинство программного обеспечения ориентировано на «мягкое» реальное время. Для подобных систем характерно: гарантированное время реакции на внешние события (прерывания от оборудования); жёсткая подсистема планирования процессов (высокоприоритетные задачи не должны вытесняться низкоприоритетными, за некоторыми исключениями); повышенные требования к времени реакции на

внешние события или реактивности (задержка вызова обработчика прерывания не более десятков микросекунд, задержка при переключении задач не более сотен микросекунд)

Классическим примером задачи, где требуется ОСРВ, является управление роботом, берущим деталь с ленты конвейера. Деталь движется, и робот имеет лишь маленький промежуток времени, когда он может её взять. Если он опоздает, то деталь уже не будет на нужном участке конвейера, и следовательно, работа не будет сделана, несмотря на то, что робот находится в правильном месте. Если он подготовится раньше, то деталь ещё не успеет подъехать, и он заблокирует ей путь.

Табл. 1. Сравнение ОСРВ и обычных операционных систем.

	ОС реального времени	ОС общего назначения
Основная задача	Успеть среагировать на события, происходящие на оборудовании	Оптимально распределить ресурсы компьютера между пользователями и задачами
На что ориентирована	Обработка внешних событий	Обработка действий пользователя
Как позиционируется	Инструмент для создания конкретного аппаратно-программного комплекса реального времени	Воспринимается пользователем как набор приложений, готовых к использованию
Кому предназначена	Квалифицированный разработчик	Пользователь средней квалификации

В своем развитии ОСРВ строились на основе следующих архитектур.

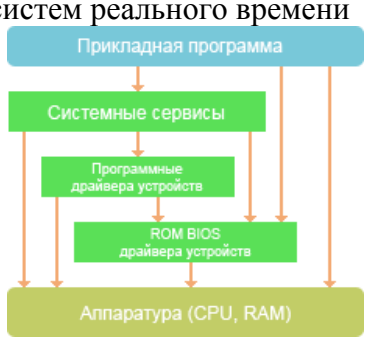
Монолитная архитектура. ОС определяется как набор модулей, взаимодействующих между собой внутри ядра системы и предоставляющих прикладному ПО входные интерфейсы для обращений к аппаратуре. Основной недостаток этого принципа построения ОС заключается в плохой предсказуемости её поведения, вызванной сложным взаимодействием модулей между собой.

Уровневая (слоевая) архитектура. Пример – MS-DOS. Прикладное ПО имеет возможность получить доступ к аппаратуре не только через ядро системы и ее сервисы, но и напрямую. По сравнению с монолитной такая архитектура обеспечивает значительно большую степень предсказуемости реакций системы, а также позволяет осуществлять быстрый доступ прикладных приложений к аппаратуре. Главным недостатком таких систем является отсутствие многозадачности.

Архитектура «клиент–сервер». Основной её принцип заключается в вынесении сервисов ОС в виде серверов на уровень пользователя и выполнении микроядром функций диспетчера сообщений между клиентскими пользовательскими программами и серверами – системными сервисами. Преимущества такой архитектуры: Повышенная надежность, т. к. каждый сервис является, по сути, самостоятельным приложением и его легче отладить и отследить ошибки; Улучшенная масштабируемость, поскольку ненужные сервисы могут быть исключены из системы без ущерба к ее работоспособности; Повышенная отказоустойчивость, т. к. «зависший» сервис может быть перезапущен без перезагрузки системы.

Табл. 2. Архитектуры операционных систем реального времени

Монолитная архитектура



Уровневая (слоевая) архитектура

Архитектура «клиент–сервер»

Ядро операционной системы обеспечивает функционирование промежуточного абстрактного уровня ОС, который скрывает от прикладного ПО специфику технического устройства процессора (нескольких процессоров) и связанного с ним аппаратного обеспечения.

Указанный абстрактный уровень предоставляет для прикладного программного обеспечения пять основных категорий сервисов.

Управление задачами. Самая главная группа сервисов. Позволяет разработчикам приложений проектировать программные продукты в виде наборов отдельных программных фрагментов, каждый из

которых может относиться к своей тематической области, выполнять отдельную функцию и иметь свой собственный квант времени, отведенный ему для работы. Каждый такой фрагмент называется задачей. Сервисы в рассматриваемой группе обладают способностью запускать задачи и присваивать им приоритеты. Основной сервис здесь - планировщик задач. Он осуществляет контроль за выполнением текущих задач, запускает новые в соответствующий период времени и следит за режимом их работы.

Динамическое распределение памяти. Многие (но не все) ядра ОСРВ поддерживают эту группу сервисов. Она позволяет задачам заимствовать области оперативной памяти для временного использования в работе приложений. Часто эти области впоследствии переходят от задачи к задаче, и посредством этого осуществляется быстрая передача большого количества данных между ними. Некоторые очень малые по размеру ядра ОСРВ, которые предполагается использовать в аппаратных средах с строгим ограничением на объем используемой памяти, не поддерживают сервисы динамического распределения памяти.

Управление таймерами. Так как встроенные системы предъявляют жесткие требования к временным рамкам выполнения задач, в состав ядра ОСРВ включается группа сервисов, обеспечивающих управление таймерами для отслеживания лимита времени, в течение которого должна выполняться задача. Эти сервисы измеряют и задают различные промежутки времени (от 1 мкс и выше), генерируют прерывания по истечении временных интервалов и создают разовые и циклические будильники.

Взаимодействие между задачами и синхронизация. Сервисы данной группы позволяют задачам обмениваться информацией и обеспечивают ее сохранность. Они так же дают возможность программным фрагментам согласовывать между собой свою работу для повышения эффективности. Если исключить эти сервисы из состава ядра ОСРВ, то задачи начнут обмениваться искаженной информацией и могут стать помехой для работы соседних задач.

Контроль устройства ввода/вывола. Сервисы этой группы обеспечивают работу единого программного интерфейса, взаимодействующего со всем множеством драйверов устройств, которые являются типичными для большинства встроенных систем.

В дополнение к сервисам ядра, многие ОСРВ предлагают линейки дополнительных компонентов для организации таких высокоуровневых понятий, как файловая система, сетевое взаимодействие, управление сетью, управление базой данных, графический пользовательский интерфейс и т. д. Хотя многие из этих компонентов намного больше и сложнее, чем само ядро ОСРВ, они, тем не менее, основываются на его сервисах. Каждый из таких компонентов включается во встроенную систему, только если ее сервисы необходимы для выполнения встроенного приложения и только для того, чтоб свести расход памяти к минимуму.

Многие операционные системы общего назначения также поддерживают указанные выше сервисы. Однако ключевым отличием сервисов ядра ОСРВ является детерминированный, основанный на строгом контроле времени, характер их работы. В данном случае под детерминированностью понимается то, что для выполнения одного сервиса операционной системы требуется временной интервал заведомо известной продолжительности. Теоретически это время может быть вычислено по математическим формулам, которые должны быть строго алгебраическими и не должны включать никаких временных параметров случайного характера. Любая случайная величина, определяющая время выполнения задачи в ОСРВ может вызвать нежелательную задержку в работе приложения, тогда следующая задача не уложится в свой квант времени, что послужит причиной для ошибки.

В этом смысле операционные системы общего назначения не являются детерминированными. Их сервисы могут допускать случайные задержки в своей работе, что может привести к замедлению ответной реакции приложения на действия пользователя в заведомо неизвестный момент времени. При проектировании обычных операционных систем разработчики не акцентируют свое внимание на математическом аппарате вычисления времени выполнения конкретной задачи и сервиса. Это не является критичным для подобного рода систем.

Большинство ОСРВ выполняют планирование задач, руководствуясь следующей схемой. Каждой задаче в приложении ставится в соответствие некоторый приоритет. Чем больше приоритет, тем выше должна быть реактивность задачи. Высокая реактивность достигается путем реализации подхода приоритетного вытесняющего планирования (preemptive priority scheduling), суть которого заключается в том, что планировщику разрешается останавливать выполнение любой задачи в произвольный момент времени, если установлено, что другая задача должна быть запущена незамедлительно. Описанная схема работает по следующему правилу: если две задачи одновременно готовы к запуску, но первая обладает высоким приоритетом, а вторая низким, то планировщик отдаст предпочтение первой. Вторая задача будет запущена только после того, как завершит свою работу первая. Возможна ситуация, когда задача с низким

приоритетом уже запущена, а планировщик получает сообщение, что другая задача с более высоким приоритетом готова к запуску. Причиной этому может послужить какое-либо внешнее воздействие (прерывание от оборудования), как, например, изменение состояния переключатля устройства, управляемого ОСРВ. В такой ситуации планировщик задач поведет себя согласно подходу приоритетного вытесняющего планирования следующим образом. Задаче с низким приоритетом будет позволено выполнить до конца текущую ассемблерную команду (но не команду, описанную в исходнике программы языком высокого уровня), после чего выполнение задачи останавливается. Далее запускается задача с высоким приоритетом. После того, как она прорабатывает, планировщик запускает прерванную первую задачу с ассемблерной команды, следующей за последней выполненой.

Каждый раз, когда планировщик задач получает сигнал о наступлении некоторого внешнего события (триггер), причина которого может быть как аппаратная, так и программная, он действует по следующему алгоритму: Определяет, должна ли текущая выполняемая задача продолжать работать; Устанавливает, какая задача должна запускаться следующей; Сохраняет контекст остановленной задачи (чтобы она потом возобновила работу с места останова); Устанавливает контекст для следующей задачи; Запускает эту задачу. Эти пять шагов алгоритма также называются переключением задач. В обычных ОСРВ задача может находиться в 3-х возможных состояниях: Задача выполняется; Задача готова к выполнению; Задача заблокирована. Большую часть времени основная масса задач заблокирована. Только одна задача может выполняться на центральном процессоре в текущий момент времени. В примитивных ОСРВ список готовых к исполнению задач, как правило, очень короткий, он может состоять не более чем из двух-трех наименований. Основная функция администратора ОСРВ заключается в составлении такого планировщика задач. Если в списке готовых к выполнению задач последних имеется не больше двух-трех, то предполагается, что все задачи расположены в оптимальном порядке. Если же случаются такие ситуации, что число задач в списке превышает допустимый лимит, то задачи сортируются в порядке приоритета.

Для решения задачи эффективного планирования в ОСРВ наиболее интенсивно применяются два подхода:

Статические алгоритмы планирования (*RMS - Rate Monotonic Sc*heduling). Используют приоритетное вытесняющее планирование. Приоритет присваивается каждой задаче до того, как она начала выполняться. Преимущество отдается задачам с самыми короткими периодами выполнения.

Динамические алгоритмы планирования (*EDF - Earliest Deadline First Scheduling*). Приоритет задачам присваивается динамически, причем предпочтение отдается задачам с наиболее ранним предельным временем начала (завершения) выполнения.

При больших загрузках системы *EDF* более эффективен, нежели *RMS*.

Многозадачным системам необходимо распределять доступ к ресурсам. Одновременный доступ двух и более процессов к какой либо области памяти или другим ресурсам представляет определённую угрозу. Существует 3 способа решения этой проблемы: Временное блокирование прерываний; Двоичные семафоры; Посылка сигналов. ОСРВ обычно не используют первый способ, потому что пользовательское приложение не может контролировать процессор столько, сколько хочет. Однако, во многих встроенных системах и ОСРВ позволяется запускать приложения в режиме ядра для доступа к системным вызовам и дается контроль над окружением исполнения без вмешательства ОС.

На однопроцессорных системах наилучшим решением является приложение запущенное в режиме ядра, которому позволено блокирование прерываний. Пока прерывание заблокировано приложение использует ресурсы процесса единолично, никакая другая задача или прерывание не может выполнятся. Таким образом защищаются все критичные ресурсы. После того как приложение завершит критические действия, оно должно разблокировать прерывания, если таковые имеются. Временное блокирование прерывания позволено только тогда, когда самый долгий промежуток выполнения критической секции меньше, чем допустимое время реакции на прерывание. Обычно этот метод защиты используется только когда длина критического кода не превышает нескольких строк и не содержит циклов. Этот метод идеально подходит для защиты регистров. Когда длина критического участка больше максимальной или содержит циклы, программист должен использовать механизмы идентичные или имитирующие поведение систем общего назначения, такие как семафоры и посылка сигналов.

Следующим проблемам выделения памяти в ОСРВ уделяется больше внимания, нежели в операционных системах общего назначения: 1) Скорост выделения памяти. Стандартная схема выделения памяти предусматривает сканирование списка неопределенной длины для нахождения свободной области памяти заданного размера, а это неприемлемо, так как в ОСРВ выделение памяти должно происходить за фиксированное время. 2) Память может стать фрагментированной в случае разделения свободных ее

участков уже запущенными процессами. Это может привести к остановке программы из-за её неспособности задействовать новый участок памяти. Алгоритм выделения памяти, постепенно увеличивающий фрагментированность памяти, может успешно работать на настольных системах, если те перезагружаются не реже одного раза в месяц, но является неприемлемым для встроенных систем, которые работают годами без перезагрузки. Простой алгоритм с фиксированной длиной участков памяти очень хорошо работает в несложных встроенных системах. Также этот алгоритм отлично функционирует и в настольных системах, особенно тогда, когда во время обработки участка памяти одним ядром следующий участок памяти обрабатывается другим ядром. Такие оптимизированные для настольных систем *OCPB* как *Unision Operating System* или *DSPnano RTOS* предоставляют указанную возможность.

К операционным системам реального времени относятся: Свободные (*XOberon, RTLinux*, *RTEMS, eCos, Fiasco, OSA, FreeRTOS, KURT, Phoenix-RTOS, Nut/OS, Prex, RTAI, scmRTOS, SHaRK, TRON Project, Xenomai, BeRTOS*), Проприетарные (*Automation Runtime, QNX, RTOS-32, Ardence RTX, ChorusOS, DNIX, DMERT, DSOS, embOS (Segger), HP-1000/RTE, INTEGRITY, ITRON, LynxOS, MERT, MicroC/OS-II, MQX RTOS, Nucleus, OS-9, OSE, OSEK/VDX, OSEKtime, PDOS, Phar Lap ETS, PikeOS, Portos, pSOS, REX, RMX, RSX-11, RT-11, RTOS-UH, RTXC, Salvo RTOS, SINTRAN III, Symbian OS, ThreadX, VRTX, VxWorks/Tornado, Windows CE, µnOS, UNIX-RTR, Virtuoso, DSP, eCos*, Багет).

Операционные системы могут быть классифицированы по базовой технологии (*UNIX*-подобные, пост-*UNIX*/потомки *UNIX*), типу лицензии (проприетарная или открытая), развивается ли в настоящее время (устаревшие или современные), по назначению (универсальные, ОС встроенных систем, ОС *PDA*, ОС реального времени, для рабочих станций или для серверов), а также по множеству других признаков.

4. НЕКОТОРЫЕ ОПЕРАЦИОННЫЕ СИСТЕМЫ

4.1 UNIX

UNIX - группа переносимых, многозадачных и многопользовательских операционных систем..

Первая система *UNIX* разработана в 1969 в подразделении *Bell Labs* компании *AT&T*. С тех пор было создано большое количество различных *UNIX*-систем. Юридически лишь некоторые из них имеют полное право называться «*UNIX*»; остальные же, хотя и используют сходные концепции и технологии, объединяются термином «*UNIX*-подобные».

Некоторые отличительные признаки *UNIX*-систем включают в себя: использование простых текстовых файлов для настройки и управления системой; широкое применение утилит, запускаемых в командной строке; взаимодействие с пользователем посредством виртуального устройства - терминала; представление физических и виртуальных устройств и некоторых средств межпроцессового взаимодействия как файлов; использование конвейеров из нескольких программ, каждая из которых выполняет одну задачу.

В настоящее время *UNIX* используются в основном на серверах, а также как встроенные системы для различного оборудования. На рынке ОС для рабочих станций и домашнего применения *UNIX* уступили другим операционным системам, таким как *Microsoft Windows* и *Mac OS*, хотя существующие программные решения для Unix-систем позволяют реализовать полноценные рабочие станции как для офисного, так и для домашнего использования.

UNIX-системы имеют большую историческую важность, поскольку благодаря им распространились некоторые популярные сегодня концепции и подходы в области ОС и программного обеспечения. Также, в ходе разработки *Unix*-систем был создан язык Си.

Коротко остановимся на истории создания *UNIX*

В 1957 в *Bell Labs* была начата работа по созданию операционной системы для собственных нужд. Под руководством Виктора Высотского создана система *BESYS*. Впоследствии он возглавил проект *Multics*, а затем стал главой информационного подразделения *Bell Labs*. В 1964 появились компьютеры третьего поколения, для которых возможности *BESYS* уже не подходили. *UNIX* была разработана сотрудниками *Bell Labs* К. Томсоном, Д. Ритчи и Д. МакИлроем. В 1969 Кен Томпсон, стремясь реализовать идеи, которые были положены в основу MULTICS, но на более скромном аппаратном обеспечении (*DEC PDP*-7), написал первую версию новой операционной системы, а Брайан Кернинган придумал для неё название - *UNICS* (*UNIplexed Information and Computing System*) - в противовес *MULTICS* (*MULTIplexed Information and Computing Service*). Позже это название сократилось до *UNIX*. В 1971 вышла версия для *PDP*-11, наиболее успешного семейства миникомпьютеров 1970-х (в СССР оно было известно как СМ ЭВМ). Эта версия получила название «первая редакция» (*Edition* 1). Первые версии UNIX были написаны на ассемблере и не имели встроенного компилятора с языка высокого уровня. В 1969 К. Томпсон при содействии Д. Ритчи разработал и реализовал язык Би (*B*), представлявший собой упрощённый (для реализации на

миникомпьютерах) вариант разработанного в 1966 языка *BCPL*. Би, как и *BCPL*, был интерпретируемым языком. В 1972 выпущена вторая редакция *UNIX*, переписанная на языке Би. В 1969—1973 на основе Би разработан компилируемый язык, получивший название Си (*C*). В 1973 вышла третья редакция *UNIX*, со встроенным компилятором языка Си. 15 октября того же года появилась четвёртая редакция, с переписанным на Си системным ядром, а в 1975 - пятая редакция, полностью переписанная на Си. С 1974 *UNIX* стал бесплатно распространяться среди университетов и академических учреждений.

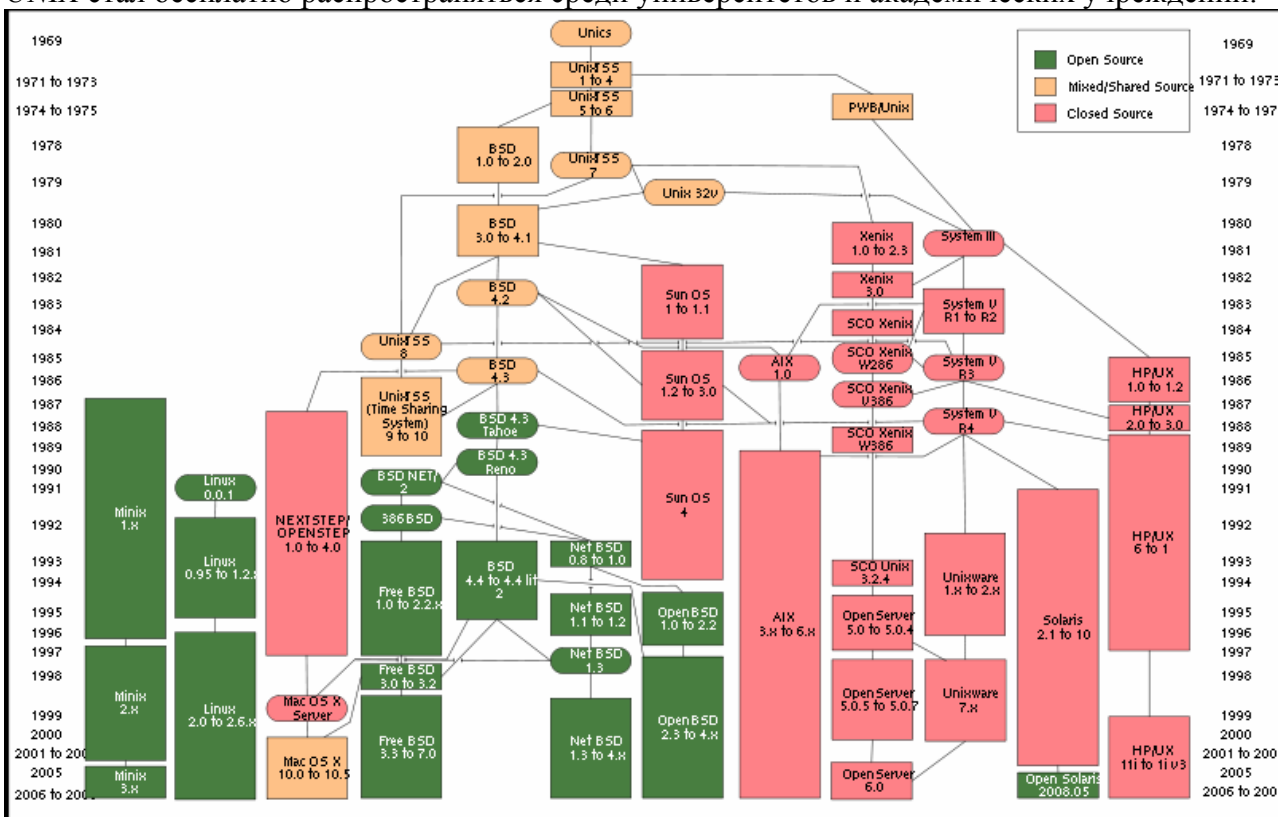


Рис. 1 . Генеалогическое древо *UNIX*-систем

С 1975 началось появление новых версий, разработанных за пределами *Bell Labs*, и рост популярности системы. *Bell Labs* выпустила шестую редакцию, известную по широко разошедшимся комментариям Джона Лайонса. Седьмая редакция была последней единой версией *UNIX*. Именно в ней появился близкий к современному интепретатор командной строки *Bourne shell*. С 1978 начинает свою историю *BSD UNIX*, созданный в университете Беркли. В 1979 выпущена версия 3*BSD (* седьмая редакция). *BSD* поддерживал такие свойства, как виртуальную память и замещение страниц по требованию (автор Б.Джой). В начале 1980-х компания *AT&T*, которой принадлежали *Bell Labs*, осознала ценность *UNIX* и начала создание коммерческой версии *UNIX*. Эта версия, поступившая в продажу в 1982, носила название *UNIX System* III и была основана на седьмой версии системы. Было предложено два интерфейса программирования сетевых приложений: *Berkley sockets* и интерфейс транспортного уровня *TLI* (*Transport Layer Interface*). Интерфейс *Berkley sockets* разработан в университете Беркли и использовал стек протоколов *TCP/IP*. *TLI* был создан *AT&T* в соответствии с определением транспортного уровня модели *OSI* и появился в системе *System* V версии 3. Реализация *TCP/IP* включена в базовую поставку *System* V версии 4. Это вызвало размежевание между двумя ветвями *UNIX* - *BSD* (университета Беркли) и *System* V (коммерческая версия от *AT&T*). Впоследствии, многие компании, лицензировав *System* V у *AT&T*, разработали собственные коммерческие разновидности *UNIX*, такие, как *AIX, CLIX, HP-UX, IRIX, Solaris*.

В середине 1983 была выпущена версия 4.2BSD, поддерживающая работу в сетях *Ethernet* и *Arpanet*. В 1983-1990 в BSD было добавлено много новых возможностей, причём существенно улучшены возможности работы с файловыми сетями. Тем временем AT&T выпускала новые версии своей системы, названной *System* V. В 1983 вышла версия 1 (*SVR1 - System V Release* 1). Версия 2 (*SVR*2, 1984), реализовывала монопольный доступ к файлам, доступ к страницам по требованию, копирование при записи. Версия 3 вышла в 1987 и включала *TLI*, а также систему поддержки удалённых файловых систем *RFS*. Версия 4 (*SVR*4,1988), поддерживала многие возможности *BSD*, в частности *TCP/IP*, сокеты, новый командный интерпретатор *csh*. Современные реализации *UNIX* не являются системами V или *BSD* в чистом виде. Они реализуют возможности как *System* V, так и *BSD*.

В 1983 Р. Столлмэн объявил о создании проекта *GNU* - свободной *UNIX*-подобной операционной системы с нуля, без использования оригинального исходного кода. Большая часть программного обеспечения, разработанного в рамках данного проекта, - такого, как GNU toolchain, *Glibc* (стандартная библиотека языка Си) и *Coreutils* - играет ключевую роль в других свободных операционных системах. Операционная система *GNU* и ядро *Linux* вместе составляют ОС, известную, как *GNU/Linux*. Дистрибутивы этой системы (такие как *Red Hat* и *Debian*), включающие ядро, утилиты *GNU* и дополнительное программное обеспечение стали популярными как среди любителей, так и среди представителей бизнеса. В 1992 вышел дистрибутив 386/*BSD*, основанный на *Networking Release* 2, распространяемый компанией *BSDI*. В 1993 появился другой дистрибутив - *FreeBSD*, нацеленный на простых пользователей. В 2005 был открыт исходный код операционной системы *Solaris*. Этот проект, как и созданная на его основе операционная система, получили название *OpenSolaris*. Затем был создан дистрибутив *SchilliX*. В 2008 появился первый официальный дистрибутив *OpenSolaris* 2008.05. Существует более десяти дистрибутивов на основе *OpenSolaris*, наиболее известные из которых *BeleniX* и *Nexenta OS*. В настоящий момент *GNU/Linux* и представители семейства *BSD* быстро отвоёвывают рынок у коммерческих *UNIX*-систем и одновременно проникают как на настольные компьютеры конечных пользователей, так и на мобильные и встраиваемые системы.

После разделения компании *AT&T*, товарный знак *UNIX* и права на оригинальный исходный код неоднократно меняли владельцев, в частности, длительное время принадлежали компании *Novell*. В 1993 *Novell* передала права консорциуму *X/Open*, который затем объединился с *Open Software Foundation*, образовав консорциум *The Open Group*. Он объединяет ведущие компьютерные корпорации и государственные организации, в том числе *IBM, Hewlett-Packard*, *Sun, NASA* др. Консорциум занимается разработкой открытых стандартов в области операционных систем, самым важным из которых является *Single UNIX Specification*, ранее известный как *POSIX*. В 1995 *Novell* продала права на лицензии и дальнейшую разработку System V компании *Santa Cruz Operation*. В 2000 *Santa Cruz Operation* продала свой *UNIX*-бизнес компании *Caldera*, которая затем была переименована в *SCO Group*.

Идеи, заложенные в основу *UNIX,* оказали огромное влияние на развитие компьютерных операционных систем. Как и *Multics, UNIX* была написана на языке высокого уровня, а не на ассемблере (доминировавшем в то время). Она содержала значительно упрощённую, по сравнению с современными ей операционными системами, файловую модель. Файловая система включала как службы, так и устройства (такие как принтеры, терминалы и жёсткие диски) и предоставляла внешне единообразный интерфейс к ним, но дополнительные механизмы работы с устройствами не вписывались в простую модель «поток байтов». *UNIX* популяризовала предложенную в *Multics* идею иерархической файловой системы с произвольной глубиной вложенности. Другие операционные системы того времени позволяли разбивать дисковое пространство на каталоги или разделы, но число уровней вложенности было фиксировано и уровень вложенности был только один. Позднее все основные фирменные операционные системы обрели возможность создания рекурсивных подкаталогов, заимствованную из *Multics*.

То, что интерпретатор команд стал просто одной из пользовательских программ, а в качестве дополнительных команд выступают отдельные программы, является ещё одной инновацией *Multics*. Язык командной оболочки *UNIX* используется пользователем как для интерактивной работы, так и для написания скриптов, т. е. не существует отдельного языка описания заданий, как, например, в системе *JCL* фирмы *IBM*. Так как оболочка и команды операционной системы являются обычными программами, пользователь может выбирать их в соответствии со своими предпочтениями, или даже написать собственную оболочку. Наконец, новые команды можно добавлять к системе без перекомпиляции ядра. Предложенный в командной строке *UNIX*, способ создания цепочек программ, последовательно обрабатывающих данные, способствовал использованию параллельной обработки данных.

Существенными особенностями *UNIX* были полная ориентация на текстовый ввод-вывод и предположение, что размер машинного слова кратен восьми битам. Первоначально в *UNIX* не было даже редакторов двоичных файлов - система полностью конфигурировалась с помощью текстовых команд. Наибольшей и наименьшей единицей ввода-вывода служил текстовый байт, что полностью отличало ввод-вывод *UNIX* от ввода-вывода других операционных систем, ориентированного на работу с записями. Ориентация на использование текста для представления всего, что только можно, сделала полезными т. н. конвейеры (pipelines). Ориентация на текстовый восьмибитный байт сделала *UNIX б*олее масштабируемой и переносимой, чем другие операционные системы. Со временем текстовые приложения одержали победу и в других областях, например, на уровне сетевых протоколов, таких как *Telnet, FTP, SMTP, HTTP* и других.

UNIX способствовала широкому распространению регулярных выражений, которые были впервые реализованы в текстовом редакторе *ed* для *UNIX*. Возможности, предоставляемые *UNIX*-программам, стали основой стандартных интерфейсов операционных систем (*POSIX*). Широко используемый в системном программировании язык Си, созданный изначально для разработки *UNIX*, превзошёл *UNIX* по популярности. Си был первым «веротерпимым» языком, который не пытался навязать программисту тот или иной стиль программирования. Си был первым высокоуровневым языком, предоставляющим доступ ко всем возможностям процессора, таким как ссылки, таблицы, битовые сдвиги, приращения и т. п. С другой стороны, свобода Си приводила к ошибкам переполнения буфера в таких функциях стандартной библиотеки Си, как *gets* и *scanf*. Результатом стали многие печально известные уязвимости, например, та, что эксплуатировалась в знаменитом черве Морриса.

Первые разработчики *UNIX* способствовали внедрению принципов модульного программирования и повторного использоания в инженерную практику. *UNIX* предоставлял возможность использования протоколов *TCP/IP* на сравнительно недорогих компьютерах, что привело к быстрому росту Интерета. Это, в свою очередь, способствовало быстрому обнаружению нескольких крупных уязвимостей в системе безопасности, архитектуре и системных утилитах *UNIX*. Со временем ведущие разработчики *UNIX* разработали культурные нормы разработки программного обеспечения, которые стали столь же важны, как и сам *UNIX*

Особенности *UNIX*, отличающие данное семейство от других ОС: Файловая система древовидная, чувствительная к регистру символов в именах, очень слабые ограничения на длину имён. Нет поддержки структурированных файлов ядром ОС, на уровне системных вызовов файл есть поток байт. Командная строка находится в адресном пространстве запускаемого процесса, а не извлекается системным вызовом из процесса интерпретатора команд. Понятие «переменных окружения». Запуск процессов вызовом *fork*, т. е. возможность клонирования текущего процесса со всем состоянием. Понятия *stdin/stdout/stderr*. Ввод/вывод только через дескрипторы файлов. Традиционно крайне слабая поддержка асинхронного ввода/вывода, по сравнению с *VMS* и *Windows NT*. Интерпретатор команд есть обыкновенное приложение, общающееся с ядром обыкновенными системными вызовами. Команда командной строки есть не более чем имя файла программы, не требуется специальная регистрация и специальная разработка программ как команд. Не принят подход с программой, задающей пользователю вопросы о режимах своей работы, вместо этого используются параметры командной строки. Пространство имён устройств на диске в каталоге */dev*, поддающееся управлению администратором, в отличие от подхода *Windows*, где это пространство имен размещается в памяти ядра, и администрирование этого пространства (например, задание прав доступа) крайне затруднено из-за отсутствия его постоянного хранения на дисках (строится каждый раз при загрузке). Широкое использование текстовых файлов для хранения настроек, в отличие от двоичной базы данных настроек, как, например, в *Windows*. Широкое использование утилит обработки текста для выполнения повседневных задач под управлением скриптов. «Раскрутка» ОС после загрузки ядра путём исполнения скриптов стандартным интерпретатором команд. Широкое использование конвейеров (*pipe*). Все процессы, кроме init, равны между собой, не бывает «специальных процессов». Адресное пространство делится на глобальное для всех процессов ядро и на локальную для процесса части, нет «групповой» части адресного пространства, как в *VMS* и *Windows NT*, как и возможности загрузки туда кода и его исполнения там. Использование двух уровней привилегий процессора вместо четырёх в *VMS*. Отказ от использования оверлеев в пользу деления программы на несколько программ поменьше, общающихся через конвейеры или временные файлы. Отсутствие APC и аналогов, то есть произвольных (а не жестко перечисленных в стандартном множестве) сигналов, не доставляемых до явного пожелания процесса их получить. Концепция сигнала уникальна для *UNIX*, и крайне сложна в переносе на другие ОС, такие, как *Windows*.

Большое количество разных вариантов системы UNIX привело к необходимости стандартизовать её средства, чтобы упростить переносимость приложений и избавить пользователя от необходимости изучать особенности каждой разновидности UNIX. С этой целью ещё в 1980 была создана пользовательская группа /usr/group. Самые первые стандарты были разработаны в 1984-1985 гг. В настоящее время наиболее важными являются следующие стандарты: *POSIX* 1003.2-1992, определяющий поведение утилит, в том числе командного интерпретатора. *POSIX* 1003.1*b*-1993, дополняющий *POSIX* 1003.1-1988. Определяет поддержку систем реального времени. *POSIX* 1003.1c-1995, дополняющий *POSIX* 1003.1-1988. Определяет нити (*threads*). Все стандарты *POSIX* объединены в документе *IEEE* 1003.

В начале 1990-х годов *The Open Group* предложила другой, похожий на *POSIX* стандарт - *Common API Specification*, или *Spec* 1170. Стандарт приобрёл большую популярность, чем *POSIX*, поскольку был доступен бесплатно. В 1998 были начаты работы по объединению данных стандартов. Благодаря этому в

настоящее время эти стандарты почти идентичны. Совместный стандарт называется *Single UNIX Specification Version* 3 и доступен бесплатно в интернете. В целях совместимости несколько создателей *UNIX*-систем предложили использовать *ELF*-формат систем *SVR*4 для двоичных и объектных файлов. Единый формат полностью обеспечивает соответствие двоичных файлов в рамках одной компьютерной архитектуры. Структура каталогов некоторых систем, в частности, *GNU/Linux*, определена в стандарте *Filesystem Hierarchy Standard*.

4.2 DOS

DOS (*Disk Operating System* - дисковая операционная система, ДОС) - семейство операционных систем для персональных компьютеров. Ориентированно на использование дисковых накопителей, таких как жёсткий диск и дискета.

Существовали операционные системы с таким названием для больших ЭВМ производства *IBM* и их клонов в 1960-80-х годах. Операционная система *MS-DOS* - разработанная корпорацией Microsoft однопользовательская дисковая операционная система с текстовым интерфейсом, обеспечивающая поддержку и ведение файлов в структуре многоуровневых каталогов.

Операционная система - комплекс программ, обеспечивающий: выполнение других программ; распределение ресурсов; планирование; ввод-вывод данных; управление данными; взаимодействие с оператором. Операционную систему составляют: монитор; загрузчик; супервизор; планировщик; набор системных обслуживающих программ (утилит).

DOS в основном предназначался для *IBM*- совместимых компьютеров. *DOS* является однозадачной операционной системой. После запуска управление передаётся прикладной прграммой, которая получает в своё распоряжение все ресурсы компьютера и может осуществлять ввод/вывод посредством как функций предоставляемых операционной системой, так и функций базовой системы ввода/вывода (*BIOS*), а также работать с устройствами напрямую.

BIOS (*Basic Input-Output System* - базовая система ввода-вывода) - небольшая программа, находящаяся в ПЗУ (постоянное запоминающее устройство) и отвечающая за самые базовые функции интерфейса и настройки оборудования, на котором она установлена. Наиболее широко среди пользователей компьютеров известна *BIOS* материнской платы, но *BIOS* присутствуют почти у всех компонентов компьютера: у видеоадаптеров, сетевых адаптеров, модемов, дисковых контроллеров, принтеров. Обозначение подобного базового ПО термином «*BIOS*» присуще для персональных компьютеров на базе процессоров с архитектурой x86. Для компьютеров на базе процессоров других типов для обозначения ПО, выполняющего подобные функции, используются другие термины, например, базовое ПО машин с процессором архитектуры *SPARC* называется *PROM*.

DOS имеет консольную систему ввода/вывода и поддерживает три стандартных потока: stdin, stdout и stderr. *DOS* - 16-битная операционная система, работающая в реальном режиме, поэтому для расширения возможностей и преодоления ограничений реального режима были созданы так называемые расширители *DOS*. Они запускают программы в защищённом 32-битном режиме и эмулируют исходные сервисы операционной системы. Обычно они поддерживают стандарт *DOS Protected Mode Interface (DPMI*). Самый известный и широко используемый (в компьютерных играх) расширитель - *DOS/4GW*. Существует несколько ветвей ДОС для ПК. Все они схожи по наборам команд и базовой функциональности, но отличаются производительностью, стабильностью работы и дополнительными функциями.

DR-DOS (Novell DOS, Caldera DR-DOS) - выпущена *Digital Research* в 1991, перекуплена компанией *Novell* в 1993, затем компанией *Caldera*. *MS-DOS* выпущена компанией *Microsoft* в 1981. PC DOS выпущена компанией *IBM* в 1981. *PTS-DOS* выпущена компанией ФизТехСофт в 1991. Paragon DOS Pro. FreeDOS выпущена в 1994. Свободная ДОС, изначально называлась *PD-DOS*. *FreeDOS*-32 - свободная 32-битная ДОС. Не требует расширителей для запуска 32-битных приложений. Появление *FreeDOS*, а также развитие свободного программного, привело к появлению полностью свободного дистрибутива ДОС *GNU/DOS*. В его состав входят *GNU*-приложения, такие как vim (текстовый редактор), *Arachne* (веб-браузер, почтовый клиент и файловый менеджер), *OpenGEM* (графический пользовательский интерфейс), различные средства разработки программного обеспечения для ДОС. Его объём составляет более 70 Мб двоичных программ, а также более 200 Мб двоичных программ и их исходников. *GNU/DOS* может пригодиться пользователям старых компьютеров, желающим пользоваться самыми современными версиями программ, а также разработчикам, которым необходимо полное управление оборудованием компьютера.

4.3 MS-DOS

MS-DOS (*Microsoft Disk Operating System* - дисковая ОС от *Microsoft*) - коммерческая операционная система фирмы *Microsoft* для персональных компьютеров. *MS-DOS* - самая известная ОС из семейства *DOS*, ранее устанавливаемая на

большинство *IBM PC*-совместимых компьютеров. Со временем она была вытеснена ОС семейства *Windows* 9*x* и *Windows NT*.

MS-DOS была создана в 1981 и, в ходе её развития, было выпущено восемь крупных версий (1.0, 2.0 и т. д.) и два десятка промежуточных (3.1, 3.2 и т. п.), пока в 2000 *Microsoft* не прекратила её разработку. Это был ключевой продукт фирмы, дававший ей существенный доход и маркетинговый ресурс, в ходе развития *Microsoft* от разработчика языка программирования до крупной компании, производящей самое разнообразное программное обеспечение. Последняя официальная версия 6.22. Однако существует версия 7.1 в виде ядра *Windows* 98, которая загружается на начальном этапе загрузки системы.

В 1980 Т. Патерсоном из *Seattle Computer Products* была создана *QDOS (Quick and Dirty Operating System*). Она продавалась *SCP* под названием 86-*DOS*), так как была создана для процессора Intel 8086. В основном *QDOS* базировалась на наиболее известной ОС того времени - *CP/M*, созданной компанией *Digital Research*, однако использовала другую файловую систему. *Microsoft* приобрела лицензию 86-*DOS*. *MS-DOS* работает в реальном режиме x86-процессора и поддерживает исполнение только одной программы одновременно. Ядро системы устанавливает прерывание *INT 21h* для системных сервисов - таких, как открытие файла, запись в файл и подобных. Для *MS-DOS*, предоставляющей пользователю лишь интерфейс командной строки, был создан целый ряд оболочек, т. е. программ, которые позволяют сделать работу с файлами более наглядной и удобной.

Наиболее известные оболочки:

Norton Commander - наиболее популярный в России коммерческий файловый менеджер. Все операции с файлами производятся на двух панелях при помощи горячих клавиш и, позднее, мыши и меню. Последние версии включают множество плагинов, значительно расширяющих функциональность. По образу Norton Commander позже было создано множество интерфейсов файловых менеджеров и других программ для различных операционных систем.

Volkov Commander - клон *Norton Commander*. В отличие от *Norton Commander*, поддерживает длинные имена файлов. Очень компактен. Базовый комплект включает только сам файловый менеджер с минимальным, но достаточным набором функций, и занимает на диске 64 Кбайт. Функциональность расширяется подключением других приложений.

DOS Navigator - дальнейшее развитие идеи *Norton Commander*. Большая функциональность. Большее количество панелей. Переключение между окнами. Расширение за счёт лёгкого и удобного подключения плагинов и приложений сторонних разработчиков.

DOS Shell (DOSSHELL) - начиная с версии *MS-DOS* 5.0 входит в состав дистрибутива *MS-DOS*. Оболочка использует «двухпанельный» принцип, но уже с псевдографическим интерфейсом и манипулятором «мышь». Позволяет запускать одновременно несколько приложений, но только одно из них активно. Была перенесена из дистрибутива *MS-DOS* 6.22 в дополнительный пакет *MS-DOS Resource Kit*, который можно было приобрести по прилагавшемуся купону.

Заметим, что в операционных системах *MS Windows* начиная с версии 9x не всегда удаётся запустить приложение, написанное для *MS-DOS*. Ещё одна проблема, с которой сталкиваются пользователи при работе с приложениями *MS-DOS* на современных компьютерах - значительная разница в быстродействии. За последние годы быстродействие компьютеров значительно возросло. Поэтому многие игры для *MS-DOS* на современном компьютере работают слишком быстро, так что пользователь не успевает увидеть происходящее на экране и проанализировать игровую ситуацию. Причина - в использовании циклов для формирования задержек. Современные процессоры выполняют их слишком быстро, а часто и вообще игнорируют. По этой же причине некоторые приложения прекращают работу, выводя ошибку деления на ноль. Для решения вышеназванных и целого ряда других проблем работы с приложениями *MS-DOS* под управлением *Windows NT* и *Unix-подобных* ОС применяются специальные эмуляторы. На данный момент наиболее известный из них - *DOSBox*, позволяющий настраивать индивидуальные параметры запуска каждого *MS-DOS*-приложения: быстродействие эмулируемого компьютера, эмулируемая звуковая и видеокарта и т. п.

Минимальный набор файлов операционной системы *MS-DOS: IO.SYS* - расширение *BIOS*; *MSDOS.SYS* - обработка прерываний; *COMMAND.COM* - командный процессор (поддержка интерфейса командной строки). *COMMAND.COM* не является необходимым. Его можно заменить любым другим приложением, способным выполнять нужные вам команды. Делается это добавлением в *CONFIG.SYS* строки *shell=c:\my\myprog.com*. В своё время сторонними разработчиками было выпущено множество командных процессоров. Наиболее распространённый - *NDOS.COM* (лицензированный 4*DOS*) из пакета *Norton Utilities* фирмы *Symantec*. Файлы конфигурации: *CONFIG.SYS* - конфигурирование системы и

загрузка драйверов устройств на этапе инициализации *MSDOS.SYS; AUTOEXEC.BAT* - стартовый пакетный файл. Выполняется при запуске *COMMAND.COM* во время загрузки *MS-DOS*.

Некоторые файлы и их функциональное назначение: *ANSI.SYS* - расширенный драйвер консоли (экрана и клавиатуры). *HIMEM.SYS* - драйвер дополнительной (*extended memory*) и *HMA*-памяти. *EMM386.EXE* - драйвер расширенной памяти (*expanded memory*); *RAMDRIVE.SYS* - драйвер электронного диска. *KEYB.COM* - драйвер переключения языковых раскладок клавиатуры. *KEYBOARD.SYS* - файл с описаниями языковых раскладок клавиатуры, оформленный как драйвер. *COUNTRY.SYS* - Файл с таблицами локализации, алфавитами сортировки. *DISPLAY.SYS* - драйвер дисплея; в частности, загружает локализованные шрифты. **.CPI* - загружаемые шрифты кодовых страниц экрана и клавиатуры. *MODE.COM* - программа настройки ряда параметров экрана и портов ввода-вывода системы: последовательного, параллельного и т.д.

4.4 Windows

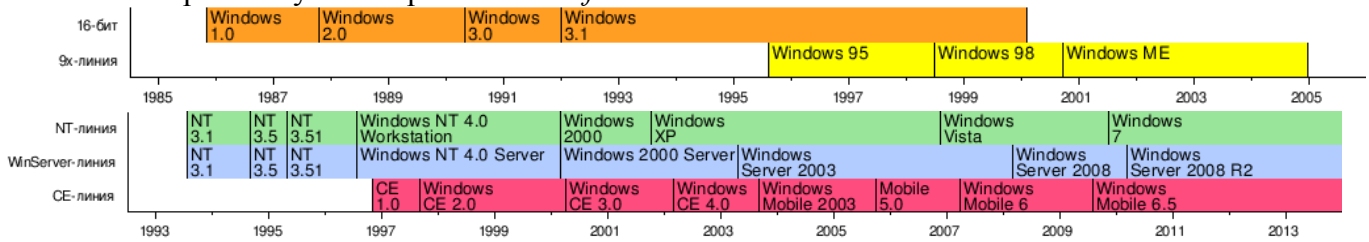
Microsoft Windows - семейство проприетарных операционных систем корпорации Майкрософт (*Microsoft*), базирующихся на основе графического интерфейса пользователя. Появление их явилось решающим шагом в широком продвижении и развитии перспективных способов взаимодействия систем человек-машина и машина-машина, создания дружественной среды для взаимодействия как пользователя с компьютерными приложениями, так и аппаратных средств внутри вычислительного комплекса.

Проприета́рное программное обеспечение (*proprietary oftware частное, патентованное, в составе собственности + программное обеспечение*) – программное обеспечение, являющееся частной собственностью авторов или правообладателей и не удовлетворяющее критериям свободного (не просто с открытым кодом). С позиции Фонда свободного ПО оно, к тому же не является *полусвободным*. Правообладатель сохраняет за собой монополию на его использование, копирование и модификацию, полностью или в существенных моментах. Часто проприетарным называют любое несвободное ПО, включая *полусвободное*.

Рис. 2. Скриншот *Windows* 7

В настоящее время под управлением операционных систем семейства *Windows* работает более 90% всего парка вычислительных машин в мире и около 95% процентов персональных компьютеров. Операционные системы *Windows* работают на платформах *x*86, *x*86-64, *IA*-64, *ARM*. Существовали также версии для *DEC Alpha, MIPS, PowerPC* и *SPARC*.

Табл. 3. История выпусков версий *Microsoft Windows*



Дата выхода	Название	Последняя версия	Дата прекращения поддержки	Последняя совместимая версия Internet Explorer
20.11.1985	Windows 1.0	1.04 (1.04.1987)	31.12.2001	
1.11.1987	Windows 2.0	2.11 (13.03.1989)	31.12.2001	
22.05.1990	Windows 3.0	3.00a (31.10.1990)	31.12.2001	
18.03.1992	Windows 3.1	3.1	31.12.2001	
10.1992	Windows For Workgroups 3.1	3.11 (31.12.1993)	31.12.2001	5.0
27.07.1993	Windows NT 3.1	3.10.528 SP3 (10.10.1994)	31.12.2000	5.0
21.09.1994	Windows NT 3.5	3.50.807 SP3	31.12.2000	

		(21.06.1995)		
30.05.1995	Windows NT 3.51	3.51.1057 SP5 (19.09.1996)	31.12.2001	
24.08.1995	Windows 95	4.00.950C (26.10.1997)	31.12.2000; 31.12.2001 (SBL)	5.5
29.07.1996	Windows NT 4.0	4.00.1381 / SP6a SRP (26.07.2001)	20.06.2002; 30.06.2003 (SBL); 31.12.2004 (ext)	6.0
25.06.1998	Windows 98	4.10.1998 (25.06.1998)	30.06.2002 (retail); 30.10.2003 (SBL); 11.07.2006 (ext)	6.0
5.05.1999	Windows 98 SE	4.10.2222A (5.06.1999)	30.06.2002 (retail); 31.03.2004 (SBL); 11.07.2006 (ext)	6.0
17.02.2000	Windows 2000	5.0.2195 / 5.0 SP4 Rollup 1 v2 (13.09.2005)	31.03.2004 (retail); 31.03.2005 (SBL); 13.06/2010 (ext)	6.0
14.09.2000	Windows Me	4.90.3000 (14.09.2000)	31.12.2004 (retail); 30.06.2004 (SBL); 11.07.2006 (ext)	6.0
25.10.2001(RTM) 31.12.2001 (продажи)	Windows XP	5.1.2600.5112 SP3 (19.02.2008)	30.09.2004 (RTM); 10.10. 2006 (SP1/SP1a); 30.05.2008 (retail); 31.01.2009 (SBL); 14.04.2009 (SP3); 8.04.2014 (ext)	8
28.03. 2003	Windows XP 64-bit Edition	5.2.3790	25.06.2005	8
24.04.2003	Windows Server 2003	5.2.3790.3959 SP2 (13.03. 2007)	13.07.2010	8
25.04.2005	Windows XP Professional x64 Edition	5.2.3790.3959 SP2 13.03. 2007	30.06.2008 (retail); 31.01.2009 (SBL)	8
8.07. 2006	Windows Fundamentals for Legacy PCs	5.1.2600 RTM (8.07.2006)	8.07.2008(retail), 12.07.2010(Service Pack)	8
8.11.2006 (RTM) 30.01.2007 (продажи)	Windows Vista	6.0.6001 / SP2 Build 6002 (25.05.2009)	10.04.2012(retail), 13.04.2010(SP2), N/A(ext)	8
16.07.2007	Windows Home Server	5.2.1500 (16.07.2007)	8.01.2013(retail)	
27.02.2008	Windows Server 2008	6.0.6002 / SP2 build 6002 (25 мая 2009)	9.07.2015(retail),10.07.2018(ext), 12 .07.2011(SP2)	8
13.07.2009 (RTM) 22.10.2009 (продажи)	Windows 7	6.1.7600 (22.10. 2009)	13.01.2015(retail), 14 января 2020(ext)	8
13.07.2009 (RTM) 22.10.2009 (продажи)	Windows Server 2008 R2 (ранее известна как Windows Server 7)	6.1.7600 (22.10. 2009)	9.07.2015(retail),10.07.2018(ext)	8

В столбце «Дата прекращения поддержки» словом «*retail*» помечается дата окончания продаж конечному пользователю; аббревиатурой «*SBL*» помечена дата окончания выдачи *System Builder* лицензии; «*ext*» - окончание срока продления поддержки

Обычно все версии Windows делят на несколько «групп».

Графические интерфейсы и расширения для DOS. Эти версии *Windows* не были полноценными операционными системами, а являлись надстройками к операционной системе *MS-DOS* и являлись по сути операционными оболочками, обеспечивая стандартизацию интерфейсов аппаратного обеспечения и единообразие для ползовательских интерфесов программ. Предоставляли встроенные средства (*GDI*) для создания графического интерфейса пользователя. Они работали с процессорами начиная с *Intel* 8086. *Windows* 1.0 (1985); *Windows* 2.0 (1987); *Windows* 2.1 (*Windows* 386) (1987) - в системе появилась возможность запуска *DOS*-приложений в графических окнах, причём каждому приложению предоставлялись полные 640 Кб памяти. Полная поддержка процессора 80286. *Windows* 3.0 (1990) -

появилась поддержка процессоров 80386 и защищённого режима. *Windows* 3.1 (1992) - серьёзно переработанная *Windows* 3.0; устранены *UAE* (*Unrecoverable Application Errors* - фатальные ошибки прикладных программ), добавлен механизм *OLE*, печать в режиме *WYSIWYG* («что видите, то и получите»), шрифты *TrueType*, изменён Проводник (диспетчер файлов), добавлены мультимедийные функции. Windows для рабочих групп (*Windows for Workgroups*) 3.1/3.11 - первая версия ОС семейства с поддержкой локальных сетей. В *WFWG* 3.11 также испытывались отдельные усовершенствования ядра, применённые позднее в *Windows* 95.

Семейство *Windows* 9x включает в себя *Windows* 95, *Windows* 98 и *Windows Me*. *Windows* 95 была выпущена в 1995. Её отличительными особенностями являются новый пользовательский интерфейс, поддержка длинных имён файлов, автоматическое определение и конфигурация периферийных устройств *Plug* and *Play*, и способность исполнять 32-битные приложения. *Windows* 95 использует вытесняющую многозадачность и выполняет каждое 32-битное приложение в своём адресном пространстве. Операционные системы этого семейства не являлись безопасными многопользовательскими системами как *Windows NT*, поскольку строгое разделение исполняющихся приложений не было реализовано в ядре. Программный интерфейс был подмножеством *Win32 API* поддерживаемым *Windows NT*, но имел поддержку юникода в очень ограниченном объёме. Также в нём не было должного обеспечения безопасности. В составе *Windows* 95 присутствовал *MS-DOS* 7.0, однако его роль сводилась к обеспечению процесса загрузки и исполнению 16-битных *DOS* приложений.

Семейство *Windows NT*. Операционные системы этого семейства в настоящее время работают на процессорах с архитектурами *x*86, *x*64, и *Itanium*. Ранние версии (до 4.0 включительно) также поддерживали некоторые *RISC*-процессоры: *Alpha, MIPS*, и *Power PC*. Все операционные системы этого семейства являются полностью 32-битными операционными системами, и не нуждаются в *MS-DOS* даже для загрузки. Только в этом семействе представлены операционные системы для серверов. До версии Windows 2000 включительно они выпускались под тем же названием что и аналогичная версия для рабочих станций, но с добавлением суффикса, например «*Windows NT* 4.0 *Server*» и «*Windows 2000 Datacenter Server*». Начиная с Windows Server 2003, серверные операционные системы называются по-другому. *Windows NT* 3.1 (1993); *Windows NT* 3.5 (1994); *Windows NT* 3.51 (1995); *Windows NT* 4.0 (1996); *Windows* 2000 (2000) - *Windows NT* 5.0; *Windows XP* (2001) - *Windows NT* 5.1; *Windows XP* 64-*bit Edition* (2006) - *Windows NT* 5.2; *Windows Server* 2003 (2003) - *Windows NT* 5.2; *Windows Vista* (2006) - *Windows NT* 6.0; *Windows Home Server* (2007) - *Windows NT* 5.2; *Windows Server* 2008 (2008) - *Windows NT* 6.0; *Windows Small Business Server* (2008) - *Windows NT* 6.0; *Windows* 7 - *Windows NT* 6.1 (2009); *Windows Server* 2008 *R*2 - *Windows NT* 6.1 (2009).

В основу семейства *Windows NT* положено разделение адресных пространств между процессами. Каждый процесс имеет возможность работать с выделенной ему памятью. Однако он не имеет прав для записи в память других процессов, драйверов и системного кода. Семейство Windows NT относится к операционным системам с вытесняющей многозадачностью. Разделение процессорного времени между потоками происходит по принципу «карусели». Ядро операционной системы выделяет квант времени (в *Windows* 2000 квант равен 20 мс) каждому из потоков по очереди при условии, что все потоки имеют одинаковый приоритет. Поток может отказаться от выделенного ему кванта времени. В этом случае система перехватывает у него управление (даже если выделенный квант времени не закончен) и передаёт управление другому потоку. При передаче управления другому потоку система сохраняет состояние всех регистров процессора в особой структуре в оперативной памяти. Эта структура называется контекстом потока. Сохранение контекста потока достаточно для последующего возобновления его работы.

Семейство ОС *Windows Mobile* для карманных компьютеров. Это семейство операционных систем реального времени было специально разработано для встраиваемых систем. Поддерживаются процессоры *ARM*, *MIPS, SuperH* и *x*86. В отличие от остальных операционных систем *Windows*, операционные системы этого семейства продаются только в составе готовых устройств, таких как смартфоны, карманные компьютеры, *GPS* навигаторы, MP3 проигрыватели, и другие. В настоящее время под термином «*Windows CE*» понимают только ядро операционной системы. Например, *Windows Mobile* 5.0 включает в себя ядро *Windows CE* 5.0, хотя в некоторых устройствах ядро *Windows* CE используется и без *Windows Mobile*.

Семейство встраиваемых ОС *Windows Embedded* - это семейство операционных систем реального времени, было специально разработано для применения в различных встраиваемых системах. Ядро системы общее с семейством ОС *Windows CE* и поддерживает процессоры *ARM*, *MIPS, SuperH* и *x*86. *Windows Embedded* включает дополнительные функции по встраиванию, среди которых фильтр защиты от записи (*EWF* и *FBWF*), загрузка с флеш-памяти, *CD-ROM*, сети, использование собственной оболочки системы и т.п. В отличие от остальных операционных систем *Windows*, операционные системы этого семейства

продаются только в составе готовых устройств, таких как: банкоматы, медицинские приборы, навигационное оборудование, «тонкие» клиенты, *VoIP*-терминалы, медиа-проигрыватели, цифровые рамки (альбомы), кассовые терминалы, платёжные терминалы, роботы, игровые автоматы, музыкальные автоматы, и другие. В настоящее время выпускаются следующие варианты ОС *Windows Embedded*: *Windows Embedded CE, Windows Embedded Standard, Windows Embedded POSReady, Windows Embedded Enterprise, Windows Embedded NavReady, Windows Embedded Server.*

Интегрированные программные продукты. Пакет *Microsoft Windows* включает в себя стандартные приложения, такие как браузер (*Internet Explorer*), почтовый клиент (*Outlook Express* или *Windows Mail*), музыкальный и видео проигрыватель (*Windows Media Player*). С помощью технологий *COM* и *OLE* их компоненты могут быть использованы в приложениях сторонних производителей. Эти продукты бесплатны, и могут быть свободно скачаны с официального сайта *Microsoft*, однако для установки некоторых из них необходимо иметь лицензионную версию *Microsoft Windows*. Запуск этих программ под другими операционными системами возможен только с помощью эмуляторов среды *Windows* (*Wine*), хотя такое их использование нарушает пользовательское соглашение.

В настоящее время *Microsoft Windows* установлена примерно на 93% персональных компьютеров и рабочих станций. Кроме того, набирает оборот (около 5 %) и её конкурент - *Mac OS X*. Также заметен рост пользователей операционной системы *GNU/Linux* (по данным *Net Applications* - с 0,68% в мае 2008 года до 0,93% в августе 2009, а по данным *w3schools* - с 3,6% до 4,2% за тот же период), занимающей третье место в рейтингах популярности.

4.5 OS/2

OS/2 – операционная система фирмы *IBM.*

Параллельно с разработкой *Windows* корпорация Microsoft совместно с *IBM* вела активную работу по созданию системы OS/2. 1.08.1984 *IBM* объявила о выпуске нового поколения персональных компьютеров - *IBM PC/AT*. Совместно с *Microsoft IBM* приступила к разработке новой операционной системы для компьютеров *IBM PC AT*. Новая ОС должна была преодолеть ограничение *DOS* на 640 Кб памяти для прикладных программ и реализовать поддержку режима многозадачности. В начале 1990-х годов пути двух гигантов *IT*-индустрии разошлись. *Microsoft* независимо от *IBM* начинает разработку *Windows* 3.0. *IBM*, независимо от *Microsoft*, разворачивает работу над облегчённой версией *OS*/2, которая требовала бы меньше ресурсов, чем *OS*/2 1.2. Было полностью переписано ядро и драйверы, добавлены *TCP/IP*- и *USB*-стеки. *OS*/2 *v*0.99 – бета-версия. Предназначена для отладочных целей, поставлялась почти без драйверов. Цель создания - конкуренция на рынке многозадачных оболочек для *DOS*, основным конкурентом считалась *DESQview*. Включает поддержку кооперативной многозадачности и некоторых функций *API DESQview*. *OS*/2 v1.0 (декабрь 1987) - первая официальная версия.

После того, как *IBM* и *Microsoft* разошлись в разные стороны, *Microsoft* переделала свою версию *OS*/2 в *Windows NT*, а сама *OS*/2 продолжала разрабатываться в фирме *IBM*, которая всё же не уделяла этой операционной системе должного внимания. 21.05.1990 вышла *Windows* 3.0. За первый месяц её копий было продано больше, чем копий *OS*/2 за целых три года. *Microsoft* занимает доминирующее положение на рынке офисного ПО для *Windows*. В 1991 *Microsoft* прекратила участие в разработке *OS*/2, разрабатывающаяся аппаратно-независимая ОС переименована из «*OS*/2, *Version* 3» в «*Windows NT*». 26.10.1996 вышла следующая версия - *OS*/2 *Warp* 4.0 (Мерлин). В 1999 появляется *OS*/2 *Warp Server for e-business* (кодовое название «Аврора», версия системы - 4.5).

OS/2 существует до сих пор и приобрела некоторую популярность в среде корпоративных клиентов и сетевиков. И сегодня многие крупнейшие корпорации в Европе доверяют *OS*/2 управление своими компьютерными сетями, однако в России *OS*/2 не получила широкого распространения. Особой популярностью в качестве домашней операционной системы *OS*/2 никогда не пользовалась, оставаясь в тени *Windows*, и, позже *Windows NT*. Тем не менее усилия как самой *IBM*, так и множества корпоративных и независимых разработчиков программного обеспечения не прошли даром - *OS*/2 является стабильной системой с предсказуемым поведением и хорошим набором системных и прикладных программ. При этом *OS*/2 представляет собой самостоятельную линию развития операционных систем, отличаясь от *Windows NT* существенно меньшими требованиями к аппаратным средствам, а от *GNU/Linux* - лучшей поддержкой программ для *DOS* и *Win*16.

Сервер «*Aurora*» (*OS*/2 *WSeB* 4.5*x*) популярна в качестве файлового сервера из-за производительности, надёжности и набора возможностей. Используется, например, в качестве сервера приложений и контроллера домена, а также позволяет использовать разнообразные интернет-сервисы вроде

серверов *HTTP, FTP, SMTP/POP3*, файрвола, прокси-сервера, сервера точного времени и т. п. *OS*/2 встречалось в 1990-х годах в институтских лабораториях - там, где много управляющих программ для «экзотических» платформ типа *Windows* 2.x или *GeoWorks*, где от системы требуется хорошая многозадачность. *OS*/2 обрабатывает многомегабайтные потоки информации. Это было одной из причин популярности *OS*/2 в Фидонете, где крупные узлы ежедневно обрабатывают десятки и сотни мегабайт почты. *OS*/2 была популярна в конце 1990-х годах для разработки программ на *Java*, учитывая уважительное отношение *IBM* к этому языку и самую быструю реализацию из существующих для платформы *x*86 *Java*-машин. Также разрабатывалась серия продуктов *IBM VisualAge* (*C++*, *Java* и *SmallTalk*).

Некоторые особенности системы

Файловая система. В *OS*/2 реализован механизм подключаемых файловых систем (*Installable File System, IFS*). Это означает, что для работы с той или иной файловой системой нужно просто загрузить соответствующий драйвер. Штатная «высокопроизводительная файловая система» (*HPFS - High Performance File System*) поддерживает разделы диска до 64 гигабайт (это ограничение драйвера, сама файловая система поддерживает до двух терабайт) и позволяет использовать имена файлов длиной до 255 символов. *HPFS* экономно расходует дисковое пространство (размер сектора составляет 512 байт), крайне мало подвержена фрагментации и отличается стабильностью. В последних версиях *OS*/2 имеется менеджер логических дисков (*LVM*), позволяющий объединять несколько физических разделов (в том числе находящихся на разных дисках) в единый том, и включена поддержка более быстрой журналируемой файловой системы *JFS*, поддерживающей тома больших объёмов (до двух терабайт). Помимо входящих в поставку *IFS* для *FAT*, *HPFS*, *JFS*, *ISO*9660 (*CDFS*) и *UDF* существуют также монтируемые файловые системы сторонних производителей для *VFAT, FAT32, EXT2, NTFS, HFS, AEFS* и др.

Графический интерфейс пользователя. В *OS*/2 в качестве штатного интерфейса используется *Workplace Shell* (*WPS*). В отличие от *Microsoft Windows*, где графические и текстовые программы используют две разные кодировки («кодировка *DOS*» и «кодировка *Windows*»), в *OS*/2 везде используется кодировка *DOS* (для русской локализации - *CP*866). Исключение составляют приложения *Windows*, запускаемые под *OS*/2. Поддержка Юникода, однако, в «родных» приложениях *OS*/2 практически отсутствует - в частности, в именах файлов, хотя файловая система *JFS* хранит их в *Unicode*.

Командная строка. В отличие от *Microsoft Windows*, *GUI* в *OS*/2 можно не загружать, получая при этом работоспособную систему в режиме командной строки. Штатная оболочка может быть заменена на более продвинутый аналог (4*os*2 или портированные из юниксов *sh, bash* и т. п.). Это позволяет получить работоспособную систему на одной-двух дискетах.

REXX - язык для написания скриптов. Это язык с весьма несложным синтаксисом, разработанный в *IBM*. Версия *REXX* для *OS*/2 позволяет писать как консольные, так и графические приложения, выполнять команды *OS*/2, а также обращаться к *API OS*/2. Помимо этого, многие программы имеют *REXX-API*, позволяющее создавать скрипты для управления работой этих программ.

TCP/IP *OS*/2 имеет юниксоподобный стек *TCP/IP*, с привычными для пользователей юниксов утилитами и демонами типа *arp, ifconfig, netstat, ppp, telnetd, sendmail* и т. д. Стек *TCP/IP* в *OS*/2 содержит клиента *SOCKS* и портированный из *AIX* файрвол.

OS/2 поддерживает приложения *DOS* и *Win*16. К примеру, *OS*/2 позволяет загружать с дискеты или её образа на диске произвольную версию *DOS* или передавать в пользование программе 736 Кб основой памяти. Для особо капризных программ существует различные параметры настройки, контролирующие практически все аспекты работы сессии *DOS*. Приложения *Windows* могут быть запущены как в полноэкранной сессии *Win-OS*/2 (ничем не отличается от *Windows* 3.1), так и поверх рабочего стола *OS*/2. По тестам *VolanoMark* 2.1.2, *IBM JDK* 1.1.7 for *OS*/2 является самой быстрой *Java*-машиной на платформе *x*86. *Java*-машина для *OS*/2 бесплатна и поставляется вместе с системой.

Для облегчения портирования *Unix/Linux*-программ, а также разработки родных приложений в *Unix*-подобном стиле, используется набор библиотек emx, распространяемый в соответствии с лицензией *GNU GP*L. Если *Unix*-программа не завязана на конкретные особенности реализации ядра (часто бывает с приложениями для *Linux*), то в большинстве случаев её можно скомпилировать под *OS*/2. Однако, если программа рассчитана на *POSIX*-совместимость, её компиляция под OS/2 может оказаться проблематичной. Существуют также несколько реализаций *X Window System* для *OS*/2, наиболее распространённой из которых является *XFree*86*OS*/2. В комплект поставки *OS*/2 *Warp Connect* и *OS*/2 *Warp* 4 (*Merlin*) входят клиенты сетей *Novell NetWare*, *Microsoft Network* и *NFS*. В *OS*/2 *Warp* 4.0 и выше реализована программная поддержка *OpenGL*. Универсальный видеодрайвер *Scitechsoft SNAP* реализует «*software optimized*»

поддержку *OpenGL*. Дайверы к самой разнообразной аппаратуре ныне отсутствуют на официальном сайте *IBM*.

В рамках проекта *Core*/2 существуют два действующих направления по развитию *OS*/2: *OS*/4 - создание современного ядра методом реверс-инжиринга и полного переписывания кода на основе существующих ядер. *osFree* - создание всей операционной системы «с нуля» на основе современных микроядерных технологий и активного использования *Open Source* наработок. *MSX* (*Machines with Software eXchangeability*) - название стандарта для бытовых компьютеров 190-х. Он являлся попыткой создания единых стандартов для разработчиков аппаратного обеспечения, Всего в мире было продано 5 миллионов экземпляров *MSX*-совместимых компьютеров.

Nishi предложил стандарт *MSX* в качестве попытки создания единого индустриального стандарта для бытовых компьютеров. Любое устройство или программное обеспечение с логотипом *MSX* совместимо с продукцией стандарта *MSX* других производителей. Стандарт *Nishi* состоял из нескольких уже имеющихся в наличии частей. Это процессор *Zilog* Z80, работающий на частоте 3.58 МГц, видеоконтроллер *TMS*9918 компании *Texas Instruments* с 16 КБ видеопамяти, и микросхема звукогенератора *AY*-3-8910 компании *General Instrument* (*GI*). Эти компоненты, совместно с интерпретатором *MSX BASIC* компании *Microsoft*, сделали MSX конкурентоспособным стандартом, но также и делали стоимость соответствующих ему компьютеров достаточно высокой. Стандарт *MSX* сильно напоминал уже существующий на тот момент бытовой компьютер *SV*-328 компании *Spectravideo*, однако этот компьютер не полностью соответствовал стандарту *MSX*. Впоследствии *Spectravideo* выпустила модель *SV*-728, которая уже являлась стандартным компьютером *MSX*. До появления и последующего большого успеха игровой консоли *Famicom* от *Nintendo*, компьютеры *MSX* были основной домашней платформой для наиболее известных японских компаний - разработчиков видеоигр.

4.6 Mac OS X

Mac OS X - *POSIX*-совместимая операционная система корпорации *Apple*. Основана на микроядре *Mach* и некоторых подсистемах *BSD* 4.4, выпускается для компьютеров *Macintosh* (Макинтош) на базе процессоров *PowerPC* и *Intel*. (Последняя версия *Mac OS X*, 10.6, поддерживает только компьютеры *Mac* на базе процессора *Intel*) Вторая по популярности в мире операционная система (рыночная доля в июле 2009 - 4,86 %).

Основа системы *Darwin* - свободное программное обеспечение. Его ядром является *XNU* (не Юникс»), в котором используется ядро *Mach* и стандартные сервисы *BSD*. Все возможности Unix доступны через консоль. Поверх этой основы в *Apple* разработано много проприетарных компонентов, таких как *API Cocoa* и *Carbon, Quartz*. *Mac OS X* включает множество возможностей, делающих её более стабильной, чем предыдущую версию *Mac OS* 9.

В *Mac OS X* используется вытесняющая многозадачность и защита памяти, позволяющие запускать несколько процессов, которые не могут прервать или повредить друг друга. На архитектуру *Mac OS X* повлияла *OpenSTEP*, которая была задумана как портируемая операционная система. К примеру, *NeXTSTEP* была портирована с оригинальной платформы 68*k* компьютера *NeXT*, до того как *NeXTSTEP* была куплена *Apple*. Так и *OpenSTEP* была портирован на *PowerPC* в рамках проекта *Rhapsody*. Наиболее заметным изменением была тема *Aqua*. Использование закруглённых углов, полупрозрачных цветов и светлых полосок также повлияло и на внешний вид аппаратного обеспечения первых *iMac*. После выхода первой версии *Mac OS X* другие разработчики тоже стали использовать дизайн *Aqua*.

Mac OS X также включает среду разработки программного обеспечения *Xcode,* которая позволяет разрабатывать программы на нескольких языках, включая Си, *C++*, *Objective-C*, *Ruby* и *Java*. Она поддерживает компиляцию в так называемые «универсальные программы» (*Universal* Binary), которые могут запускаться на нескольких платформах (*x*86, *PowerPC*), так же, как «*fat* binaries» использовались для запуска одного приложения на 68*k* и *PowerPC* платформах.

Основами *Mac OS X* являются: Подсистема с открытым кодом - *Darwin* (ядро *Mach*, набор утилит *BSD*). Среда программирования *Core Foundation* (*Carbon API, Cocoa API* и *Java API*). Графическая среда *Aqua* (*QuickTime, Quartz Extreme* и *OpenGL*). Технологии *CoreImage*, *CoreAudio* и *CoreData*. Для обеспечения гладкого перехода с *Mac OS* 9 на *Mac OS X* был создан *Carbon*. Приложение, написанное с помощью *Carbon*, может запускаться на любой из этих ОС. С другой стороны, *Mac OS X* наследует многое из *OpenSTEP*, которая не является обратно-совместимой с другими версиями *Mac OS*. В данный момент *Apple* рекомендует *API,* именуемый *Cocoa*, и там наследие *OpenSTEP* весьма заметно - имена многих классов начинаются с «*NS*» (*NSObject, NSArray*), что является аббревиатурой от *NeXTSTEP*.

Также *Mac OS X* поддерживает *Java*. Это означает, что приложения, написанные на *Java* и использующие *Swing* выглядят так же, как и приложения, использующие *Cocoa*. Традиционно приложения под *Cocoa* разрабатываются на *Objective C*, альтернативе *Java*. Однако 25.07.2007 года *Apple* заявила, что дальнейшие расширения в *Cocoa* не будут портированы на *Java*. В составе *Mac OS X*, начиная с версии 10.5 *Leopard*, поставляется интерпретатор *Ruby* с поддержкой *Cocoa*. В отличие от предшественников, *Mac OS X* является полноценной, сертифицированной *UNIX*'03 операционной системой. Это означает, что большинство программ, написанных для *BSD, GNU/Linux* и других *UNIX*-подобных систем, скомпилируются и будут работать на *Mac OS X* без дополнительных изменений в коде. Для удобной установки таких программ разработаны менеджеры пакетов, такие как *Fink* или *MacPorts*. Начиная с версии 10.3, *Mac OS X* по умолчанию включает в себя *X11.app* - адаптированную версию *X*-сервера. Это позволяет запускать на *Mac OS X* приложения, разработанные для *X*11. Для вывода на экран *X11.app* использует *Quartz*. *Mac OS X* основана на ядре *XNU*, созданном на основе микроядра *Mach* 3.0.

Ранние версии *Mac OS X* поддерживали все компьютеры *Macintosh* (лаптопы, десктопы или серверы) на процессорах *PowerPC G3, G4* и *G*5. Более поздние версии перестают поддерживать старое оборудование: например, версия 10.3 *Panther* не поддерживает старые *G*3, 10.4 *Tiger* не поддерживает системы без портов *FireWire*. Существуют утилиты, такие как *XPostFacto*, и патчи к установочному диску, сделанные сторонними разработчиками, для установки новых версий *Mac OS X* на оборудовании, официально не поддерживаемым *Apple*, включая некоторые до-*G*3 системы. Исключая некоторые возможности, требуемые оборудованием (такие, как графическое ускорение, запись *DVD*), операционная система предлагает одинаковую функциональность на всём поддерживаемом оборудовании.

Версия *Mac OS X* для *PowerPC* остаётся совместимой со старыми *Mac OS* приложениями через эмуляцию так называемой *Classic*, которая позволяет пользователям запускать *Mac OS* 9 как процесс в *Mac OS X*, поэтому многие старые приложения запускаются так, как будто под старой операционной системой. *Classic* не поддерживает компьютеры на процессорах *Intel*. В 2005 появилась версия *Mac OS X* для процессоров *Intel*. Сейчас большинство приложений, которые доступны только для *PowerPC*, поддерживается при помощи эмулятора *Rosetta*. Несмотря на серьезные изменения - например, закрытость платформы и отсутствие рабочего стола - в порте сохранились такие особенности, как *Darwin*, ядро *XNU*. В некоторых сторонних программах сохранились эффекты из настольной версии - например, в программе *Converter*.

4.7 Linux

Linux – ядро операционной системы, соответствующее стандартам POSIX. Разработка была начата финским студентом Линусом Торвальдсом в 1991.

POSIX (***P**ortable **O**perating **S**ystem **I**nterface for Uni**x** - Переносимый интерфейс операционных систем Unix*) - набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой. Стандарт создан для обеспечения совместимости различных *UNIX*-подобных операционных систем и переносимости прикладных программ на уровне исходного кода, но может быть использован и для не-*Unix* систем. Серия стандартов *POSIX* была разработана комитетом 1003 *IEEE*. Международная организация по стандартизации (*ISO*) совместно с Международной электротехнической комиссией (*IEC*) приняли данный стандарт (*POSIX*) под названием ***ISO/IEC* 9945**.

В основном код написан на Си с некоторыми расширениями *gcc* и на ассемблере. Распространяется в основном свободно на условиях *GNU General Public License*. Ядро *Linux* поддерживает многозадачность, виртуальную память, динамические библиотеки, отложенную загрузку, производительную систему управления памятью и многие сетевые протоколы. На сегодняшний день *Linux* - монолитное ядро с поддержкой загружаемых модулей. Драйверы устройств и расширения ядра обычно запускаются на «кольце 0», с полным доступом к оборудованию. В отличие от обычных монолитных ядер, драйверы устройств легко собираются в виде модулей и загружаются или выгружаются во время работы системы.

Рис. 3. Пингвин *Tux* - символ *Linux*.

Не задуманный изначально как многоплатформенное ядро, Linux на данный момент портирован на очень широкий круг архитектур, запускается на широком спектре оборудования от *iPAQ* (карманный компьютер) до *IBM S*/390 (высокопроизводительный мейнфрейм). Системы на основе *Linux* используются в

качестве основных практически на всех суперкомпьютерах (более 80 % списка Top500), в том числе и на самых мощных - *Roadrunner* фирмы *IBM*.

GNU/Linux - общее название *UNIX*- подобных операционных систем на основе одноимённого ядра и собранных для него библиотек и системных программ, разработанных в рамках проекта GNU.

GNU/Linux работает на *PC*-совместимых системах семейства *Intel x*86, а также на *IA*-64, *AMD*64, *PowerPC*, *ARM* и многих других. К операционной системе *GNU/Linux* также часто относят программы, дополняющие эту операционную систему, и прикладные программы, делающие её полноценной многофункциональной операционной средой.

В отличие от большинства других операционных систем, *GNU/Linux* не имеет единой «официальной» комплектации. Вместо этого *GNU/Linux* поставляется в большом количестве так называемых дистрибутивов, в которых программы *GNU* соединяются с ядром *Linux* и другими программами. Наиболее известными дистрибутивами *GNU/Linux* являются *Ubuntu*, *Debian GNU/Linux*, *Red Hat*, *Fedora*, *Mandriva, SuSE*, *Gentoo*, *Slackware*, *Archlinux*. Российские дистрибутивы - *ALT Linux* и *ASPLinux*.

Рис. 4. *GNU/Linux*

В отличие от *Microsoft Windows* (*Windows NT*), *Mac OS* (*Mac OS X*) и коммерческих *Unix*-подобных, *GNU/Linux* не имеет географического центра разработки. Нет и организации, которая владела бы этой системой; нет даже единого координационного центра. Программы для *Linux* - результат работы тысяч проектов. Некоторые из этих проектов централизованы, некоторые сосредоточены в фирмах. Многие проекты объединяют хакеров со всего света, которые знакомы только по переписке. Создать свой проект или присоединиться к уже существующему может любой и, в случае успеха, результаты работы станут известны миллионам пользователей. Пользователи принимают участие в тестировании свободных программ, общаются с разработчиками напрямую, что позволяет быстро находить и исправлять ошибки и реализовывать новые возможности.

GNU/Linux является *UNIX*-совместимой, однако основывается на собственном исходном коде. Именно такая гибкая и динамичная система разработки, невозможная для проектов с закрытым кодом, определяет исключительную экономическую эффективность *GNU/Linux*. Низкая стоимость свободных разработок, отлаженные механизмы тестирования и распространения, привлечение людей из разных стран, обладающих разным видением проблем, защита кода лицензией *GPL* - всё это стало причиной успеха свободных программ. Конечно, такая высокая эффективность разработки не могла не заинтересовать крупные фирмы, которые стали открывать свои проекты. Так появились *Mozilla* (*Netscape, AOL*), *OpenOffice.org* (*Sun*), свободный клон *Interbase (Borland*) - *Firebird, SAP DB (SAP*). *IBM* способствовала переносу *GNU/Linux* на свои мейнфреймы. С другой стороны, открытый код значительно снижает себестоимость разработки закрытых систем для *GNU/Linux* и позволяет снизить цену решения для пользователя. Вот почему *GNU/Linux* стала платформой, часто рекомендуемой для таких продуктов, как СУБД *Oracle, DB2, Informix, SyBase, SAP R3, Domino*. Сообщество *GNU/Linux* поддерживает связь посредством групп пользователей *Linux.*