Операционная система. Архитектура операционной системы. Эволюция операционных систем

Современный компьютер – сложнейшая аппаратно-программная система.

Написание программ для компьютера, их отладка и последующее выполнение представляет собой сложную трудоемкую задачу.

Основная причина этого – огромная разница между тем, что удобно для людей, и тем, что удобно для компьютеров.

Компьютер понимает только свой, машинный язык (назовем его Я0), а для человека наиболее удобен разговорный или хотя бы язык описания алгоритмов – алгоритмический язык.

Проблему можно решить двумя способами.

Оба способа связаны с разработкой команд, которые были бы более удобны для человека, чем встроенные машинные команды компьютера.

Эти новые команды в совокупности формируют некоторый язык, который назовем Я1.

Упомянутые два способа решения проблемы различаются тем, каким образом компьютер будет выполнять программы, написанные на языке Я1.

Первый способ – замена каждой команды языка Я1 на эквивалентный набор команд в языке Я0. В этом случае компьютер выполняет новую программу, написанную на языке Я0, вместо программы, написанной на языке Я1.

Эта технология называется трансляцией.

Второй способ – написание программы на языке Я0, которая берет программы, написанные на языке Я1, в качестве входных данных, рассматривает каждую команду по очереди и сразу выполняет эквивалентный набор команд языка Я0.

Эта технология не требует составления новой программы на Я0.

Она называется интерпретацией, а программа, которая осуществляет интерпретацию, **называется интерпретатором**.

В подобной ситуации проще представить себе существование гипотетического компьютера или виртуальной машины, для которой машинным языком является язык Я1, чем думать о трансляции и интерпретации.

Назовем такую виртуальную машину М1, а виртуальную машину с языком Я0 – М0. Для виртуальных машин можно будет писать программы, как будто они (машины) действительно существуют.

Очевидно, можно пойти дальше – создать еще набор команд, который в большей степени ориентирован на человека и в меньшей степени на компьютер, чем Я1.

Этот набор формирует язык Я2 и, соответственно, виртуальную машину М2. Так можно продолжать до тех пор, пока не дойдем до подходящего нам языка уровня n.

Большинство современных компьютеров состоит из двух и более уровней.

Уровень 0 – аппаратное обеспечение машины. Электронные схемы этого уровня выполняют программы, написанные на языке уровня 1.

Следующий уровень – микроархитектурный уровень.

На этом уровне можно видеть совокупности 8 или 32 (иногда и больше) регистров, которые формируют локальную память и АЛУ (арифметико-логическое устройство).

Следующий (второй) уровень составляет уровень архитектуры системы команд. Команды используют регистры и другие возможности аппаратуры. Команды формируют уровень ISA (Instruction Set Architecture), называемый машинным языком.

Следующий (третий) уровень обычно – гибридный.

Четвертый уровень представляет собой символическую форму одного из языков низкого уровня (обычно ассемблер).

Уровни с пятого и выше предназначены для прикладных программистов, решающих конкретные задачи на языках высокого уровня (C, C++, C#, VBA и др.).

Большинство пользователей компьютеров имеют опыт общения с операционной системой, по крайней мере, в той степени, чтобы эффективно выполнять свои текущие задачи. Однако они испытывают затруднения при попытке дать определение операционной системе. В известной степени проблема связана с тем, что операционные системы выполняют две основные, но практически не связанные между собой функции: **расширение возможностей компьютера и управление его ресурсами**.

Однако концепция, рассматривающая операционную систему прежде всего как удобный интерфейс пользователя, – это взгляд сверху вниз. Альтернативный взгляд, снизу вверх, дает представление об операционной системе как о механизме, присутствующем в компьютере для управления всеми компонентами этой сложнейшей системы. В соответствии с этим подходом работа операционной системы заключается в обеспечении организованного и контролируемого распределения процессоров, памяти, дисков, принтеров, устройств ввода-вывода, датчиков времени и т.п. между различными программами, конкурирующими за право их использовать.

Операционная система, среда и операционная оболочка

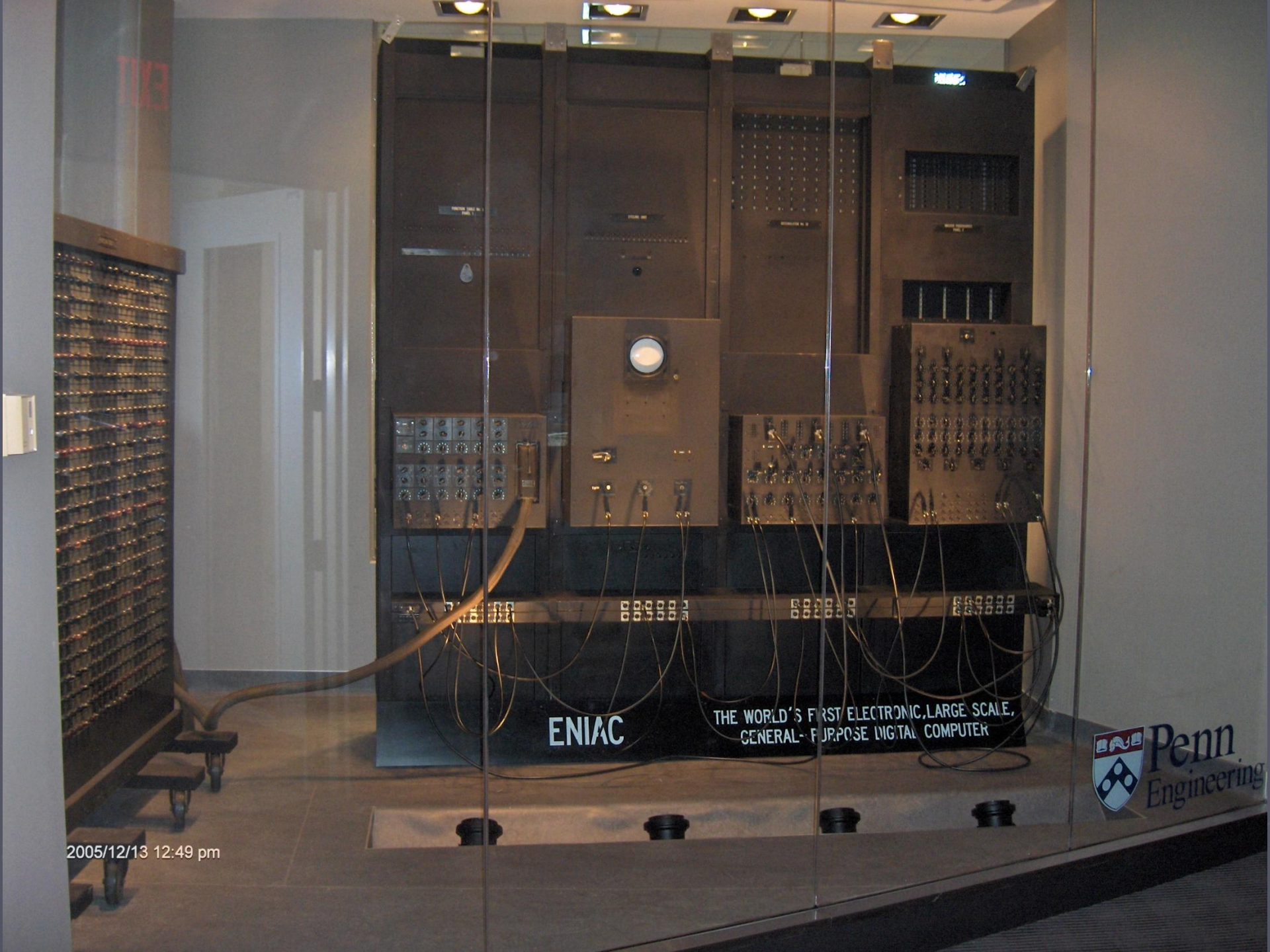
Операционные системы (ОС) в современном их понимании (их назначении и сущности) появились значительно позже первых компьютеров (правда, по всей видимости, и исчезнут в этой сущности в компьютерах будущего).

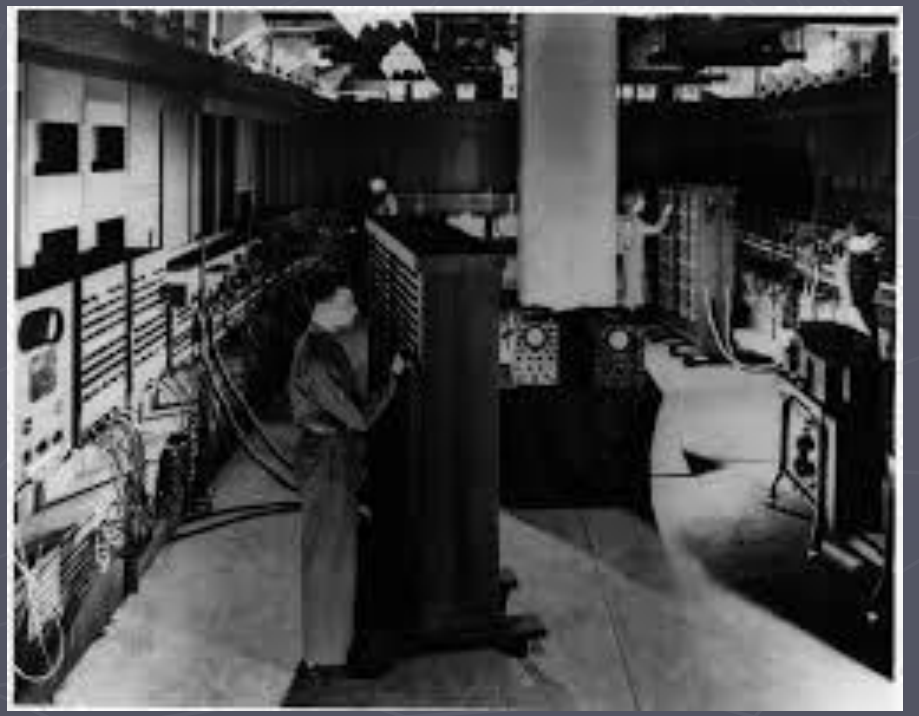
Почему и когда появились ОС?

Считается что первая цифровая вычислительная машина ENIAC (Electronic Numerical Integrator and Computer) была создана в 1946 году по проекту "Проект PX" Министерства обороны США.

На реализацию проекта затрачено 500 тыс. долларов. Компьютер содержал 18000 электронных ламп, массу всякой электроники, включал в себя 12 десятиразрядных сумматоров, а для ускорения некоторых арифметических операций имел умножитель и "делитель-извлекатель" квадратного корня.

Программирование сводилось к связыванию различных блоков проводами. Конечно, никакого программного обеспечения и тем более операционных систем тогда еще не существовало.


ENIAC
THE WORLD'S FIRST ELECTRONIC, LARGE SCALE,
GENERAL-PURPOSE DIGITAL COMPUTER
Penn
Engineering
2005/12/13 12:49 pm



Интенсивное создание различных моделей ЭВМ относится к началу 50-х годов прошлого века.

В эти годы одни и те же группы людей участвовали и в проектировании, и в создании, и в программировании, и в эксплуатации ЭВМ.

Программирование осуществлялось исключительно на машинном языке (а затем на ассемблере), не было никакого системного программного обеспечения, кроме библиотек математических и служебных подпрограмм.

Операционные системы еще не появились, а все задачи организации вычислительного процесса решались вручную каждым программистом с примитивного пульта управления ЭВМ.

С появлением полупроводниковых элементов вычислительные возможности компьютеров существенно выросли.

Выполнение программ усложнилось и включало в себя следующие основные действия:

загрузка нужного транслятора (и др.);

запуск транслятора и получение программы в машинных кодах;

связывание программы с библиотечными подпрограммами;

загрузка программы в оперативную память;

запуск программы;

вывод результатов работы программы на печатающее или другое периферийное устройство.

Для организации эффективной загрузки всех средств компьютера в штаты вычислительных центров ввели должности специально обученных операторов.

Считается, что первую операционную систему создала в 1952 году для своих компьютеров IBM-701 исследовательская лаборатория фирмы General Motors.

В 1955 году эта фирма и North American Aviation совместно разработали ОС для компьютера IBM-704.

В конце 50-х годов прошлого века ведущие фирмы изготовители поставляли операционные системы со следующими характеристиками:

пакетная обработка одного потока задач;

наличие стандартных программ ввода-вывода;

возможности автоматического перехода от программы к программе;

средства восстановления после ошибок;

языки управления заданиями, предоставляющие пользователям возможность описывать свои задания и ресурсы, требуемые для их выполнения.

Иерархическая структура программно-аппаратных средств компьютера

Операционная система предназначена для того, чтобы скрыть все эти сложности. Конечный пользователь обычно не интересуется деталями устройства аппаратного обеспечения компьютера. Компьютер ему видится как набор приложений. Приложение может быть написано программистом на каком-либо языке программирования.

Операционная система выступает в роли посредника, облегчая программисту, пользователям и программным приложениям доступ к различным службам и возможностям компьютера.

Таким образом, операционная система – это набор программ, контролирующих работу прикладных программ и системных приложений и исполняющих роль интерфейса между пользователями, программистами, прикладными программами, системными приложениями и аппаратным обеспечением компьютера.

Образно можно сказать, что аппаратура компьютера предоставляет "сырую" вычислительную мощность, а задача операционной системы заключается в том, чтобы сделать использование этой вычислительной мощности доступным и по возможности удобным для пользователя.

Таким образом, операционная среда – это программная среда, образуемая операционной системой, определяющая интерфейс прикладного программирования (API) как множество системных функций и сервисов (системных вызовов), которые предоставляются прикладным программам.

Еще одно важное понятие, связанное с операционной системой, относится к реализации пользовательских интерфейсов.

Как правило, любая операционная система обеспечивает удобную работу пользователя за счет средств пользовательского интерфейса.

В общем случае под оболочкой операционной системы понимается часть операционной среды, определяющая интерфейс пользователя, его реализацию (текстовый, графический и т.п.), командные и сервисные возможности пользователя по управлению прикладными программами и компьютером.

Перейдем к рассмотрению эволюции операционных систем.

Рассматривая эволюцию ОС, следует иметь в виду, что разница во времени реализации некоторых принципов организации отдельных операционных систем до их общего признания, а также терминологическая неопределенность не позволяют дать точную хронологию развития ОС. Однако сейчас уже достаточно точно можно определить основные вехи на пути эволюции операционных систем.

Существуют также различные подходы к определению поколений ОС. Известно разделение ОС на поколения в соответствии с поколениями вычислительных машин и систем.

Другая точка зрения не связывает поколение ОС с соответствующими поколениями ЭВМ. Так, например, известно определение поколений ОС по уровням входного языка ЭВМ, режимам использования центральных процессоров, формам эксплуатации систем и т.п.

Видимо, наиболее целесообразным следует считать выделение этапов развития ОС в рамках отдельных поколений ЭВМ и ВС.

Первым этапом развития системного программного обеспечения можно считать использование библиотечных программ. Концепция библиотек подпрограмм является наиболее ранней и восходит к 1949 году. Эти средства применялись в ЭВМ первого поколения, когда операционных систем как таковых еще не существовало.

Возникли специальные программы методов доступа. Таким образом было создано базовое системное программное обеспечение.

С улучшением характеристик ЭВМ и ростом их производительности стало ясно, что существующего базового программного обеспечения (ПО) недостаточно. Появились операционные системы ранней пакетной обработки – мониторы.

Началось интенсивное развитие методов управления данными, возникала такая важная функция ОС, как реализация ввода-вывода без участия центрального процесса – (от англ. SPOOL – Simultaneous Peripheral Operation on Line).

Появление новых аппаратных разработок (1959-1963 гг.) – систем прерываний, таймеров, каналов – стимулировало дальнейшее развитие ОС. Возникли исполнительные системы, которые представляли собой набор программ для распределения ресурсов ЭВМ, связей с оператором, управления вычислительным процессом и управления вводом-выводом. Такие исполнительные системы позволили реализовать довольно эффективную по тому времени форму эксплуатации вычислительной системы – однопрограммную пакетную обработку.

Однако однопрограммная пакетная обработка с ростом производительности ЭВМ не могла обеспечить экономически приемлемый уровень эксплуатации машин. Решением стало мультипрограммирование – способ организации вычислительного процесса, при котором в памяти компьютера находится несколько программ, попеременно выполняющихся одним процессором.

Теория построения операционных систем в этот период обогатилась рядом плодотворных идей. Появились различные формы мультипрограммных режимов работы, в том числе разделение времени – режим, обеспечивающий работу многотерминальной системы. Режим разделения времени позволил пользователю интерактивно взаимодействовать со своими программами.

Одной из первых ОС, использующих эти новейшие решения, была операционная система MCP (главная управляющая программа), созданная фирмой Burroughs для своих компьютеров B5000 в 1963 году.

В этой ОС были реализованы многие концепции и идеи, ставшие впоследствии стандартными для многих операционных систем:

мультипрограммирование;

мультипроцессорная обработка;

виртуальная память;

возможность отладки программ на исходном языке;

написание операционной системы на языке высокого уровня.

CTSS (Compatible Time Sharing System) – совместимая система разделения времени, разработанная в Массачусетском технологическом институте (1963 год) для компьютера IBM-7094.

Эта система была использована для разработки в этом же институте совместно с Bell Labs и General Electric системы разделения времени следующего поколения MULTICS (Multiplexed Information And Computing Service).

Одним из важнейших событий в истории операционных систем считается появление в 1964 году семейства компьютеров под названием System/360 фирмы IBM, а позже – System/370.

Это было первой в мире реализацией концепции семейства программно и информационно совместимых компьютеров.

ОС добавились модули, реализующие протоколы связи.

Операционная система становится "неотъемлемой частью ЭВМ".

В процессорах появился привилегированный режимы работы, мощная система прерываний, защита памяти, специальные регистры для быстрого переключения программ, средства поддержки виртуальной памяти и др.

Кроме того, современные ОС имеют достаточно большой набор средств и способов диагностики и восстановления работоспособности системы.

Сюда относятся:

диагностические программы для выявления ошибок в конфигурации ОС;

средства восстановления последней работоспособной конфигурации;

средства восстановления поврежденных и пропавших системных файлов и др.

Следует отметить еще одно назначение ОС.

Возможность развития. Современные ОС организуются таким образом, что допускают эффективную разработку, тестирование и внедрение новых системных функций, не прерывая процесса нормального функционирования вычислительной системы.

Большинство операционных систем постоянно развиваются (нагляден пример Windows). Происходит это в силу следующих причин.

Обновление и возникновение новых видов аппаратного обеспечения.

Новые сервисы. Для удовлетворения пользователей или нужд системных администраторов ОС должны постоянно предоставлять новые возможности.

Исправления. В каждой ОС есть ошибки. Время от времени они обнаруживаются и исправляются. Важную роль играет хорошая и полная документированность системы.

Перейдем к рассмотрению состава компонентов и функций ОС.

Современные операционные системы содержат сотни и тысячи модулей (например, Windows 2000 содержит 29 млн строк исходного кода на языке С).

Наиболее важными подсистемами управления ресурсами являются **подсистемы управления процессами, памятью, файлами и внешними устройствами**, а подсистемами, общими для всех ресурсов, являются подсистемы пользовательского интерфейса, защиты данных и администрирования.

Управление процессами. Подсистема управления процессами непосредственно влияет на функционирование вычислительной системы. Для каждой выполняемой программы ОС организует один или более процессов.

Каждый такой процесс представляется в ОС информационной структурой (таблицей, дескриптором, контекстом процессора), содержащей данные о потребностях процесса в ресурсах, а также о фактически выделенных ему ресурсах (область оперативной памяти, количество процессорного времени, файлы, устройства ввода-вывода и др.).

В современных мультипрограммных ОС может существовать одновременно несколько процессов, порожденных по инициативе пользователей и их приложений, а также инициированных ОС для выполнения своих функций (системные процессы).

Управление памятью. Подсистема управления памятью производит распределение физической памяти между всеми существующими в системе процессами, загрузку и удаление программных кодов и данных процессов в отведенные им области памяти, настройку адресно-зависимых частей кодов процесса на физические адреса выделенной области, а также защиту областей памяти каждого процесса.

Одним из наиболее популярных способов управления памятью в современных ОС является виртуальная память. Реализация механизма виртуальной памяти позволяет программисту считать, что в его распоряжении имеется однородная оперативная память, объем которой ограничивается только возможностями адресации, предоставляемыми системой программирования.

Важная функция управления памятью – защита памяти.

Нарушения защиты памяти связаны с обращениями процессов к участкам памяти, выделенной другим процессам прикладных программ или программ самой ОС.

Управление файлами.

Функции управления файлами сосредоточены в файловой системе ОС.

Операционная система виртуализирует отдельный набор данных, хранящихся на внешнем накопителе, в виде файла – простой неструктурированной последовательности байтов, имеющих символьное имя.

Управление внешними устройствами.

Функции управления внешними устройствами возлагаются на подсистему управления внешними устройствами, называемую также подсистемой ввода-вывода.

Она является интерфейсом между ядром компьютера и всеми подключенными к нему устройствами.

Программа, управляющая конкретной моделью внешнего устройства и учитывающая все его особенности, называется **драйвером**.

Наличие большого количества подходящих драйверов во многом определяет успех ОС на рынке. ОС должна поддерживать четко определенный интерфейс между драйверами и остальными частями ОС.

Защита данных и администрирование.

Безопасность данных вычислительной системы обеспечивается средствами отказоустойчивости ОС, направленными на защиту от сбоев и отказов аппаратуры и ошибок программного обеспечения, а также средствами защиты от несанкционированного доступа.

Важным средством защиты являются функции аудита ОС, заключающегося в фиксации всех событий, от которых зависит безопасность системы.

Интерфейс прикладного программирования.

Прикладные программисты используют в своих приложениях обращения к операционной системе, когда для выполнения тех или иных действий им требуется особый статус, которым обладает только ОС. Возможности операционной системы доступны программисту в виде набора функций, который называется **интерфейсом прикладного программирования (Application Programming Interface, API).**

Способ реализации системных вызовов зависит от структурной организации ОС, особенностей аппаратной платформы и языка программирования.

В ОС UNIX системные вызовы почти идентичны библиотечным процедурам. Ситуация в Windows иная (более подробно это рассмотрим далее).

Пользовательский интерфейс. ОС обеспечивает удобный интерфейс не только для прикладных программ, но и для пользователя (программиста, администратора).

Современные ОС поддерживают развитые функции пользовательского интерфейса для интерактивной работы за терминалами двух типов: алфавитно-цифрового и графического.

Программный модуль ОС, ответственный за чтение отдельных команд или же последовательности команд из командного файла, иногда называют командным интерпретатором (в MS-DOS – командным процессором).

Вычислительные системы, управляемые из командной строки, например UNIX-системы, имеют командный интерпретатор, называемый оболочкой (Shell). Она, собственно, не входит в состав ОС, но пользуется многими функциями операционной системы. Когда какой-либо пользователь входит в систему, запускается оболочка. Стандартным терминалом для нее является монитор с клавиатурой. Оболочка начинает работу с печати приглашения (prompt) – знака доллара (или иного знака), говорящего пользователю, что оболочка ожидает ввода команды (аналогично управляется MS-DOS). Если теперь пользователь напечатает какую-либо команду, оболочка создает системный вызов и ОС выполнит эту команду. После завершения оболочка опять печатает приглашение и пытается прочесть следующую входную строку.

Ввод команд может быть упрощен, если операционная система поддерживает графический пользовательский интерфейс.

Архитектура операционной системы

Под архитектурой операционной системы понимают структурную и функциональную организацию ОС на основе некоторой совокупности программных модулей.

В состав ОС входят исполняемые и объектные модули стандартных для данной ОС форматов, программные модули специального формата (например, загрузчик ОС, драйверы ввода-вывода), конфигурационные файлы, файлы документации, модули справочной системы и т.д.

Первая версия ОС OS/360 была создана коллективом из 5000 человек за 5 лет и содержала более 1 млн строк кода. Разработанная несколько позже операционная система Mastics содержала к 1975 году уже 20 млн строк.

Стало ясно, что разработка таких систем должна вестись на основе модульного программирования.

Большинство современных ОС представляют собой хорошо структурированные модульные системы, способные к развитию, расширению и переносу на новые платформы. Какой-либо единой унифицированной архитектуры ОС не существует, но известны универсальные подходы к структурированию ОС.

Принципиально важными универсальными подходами к разработке архитектуры ОС являются:

модульная организация;

функциональная избыточность;

функциональная избирательность;

параметрическая универсальность;

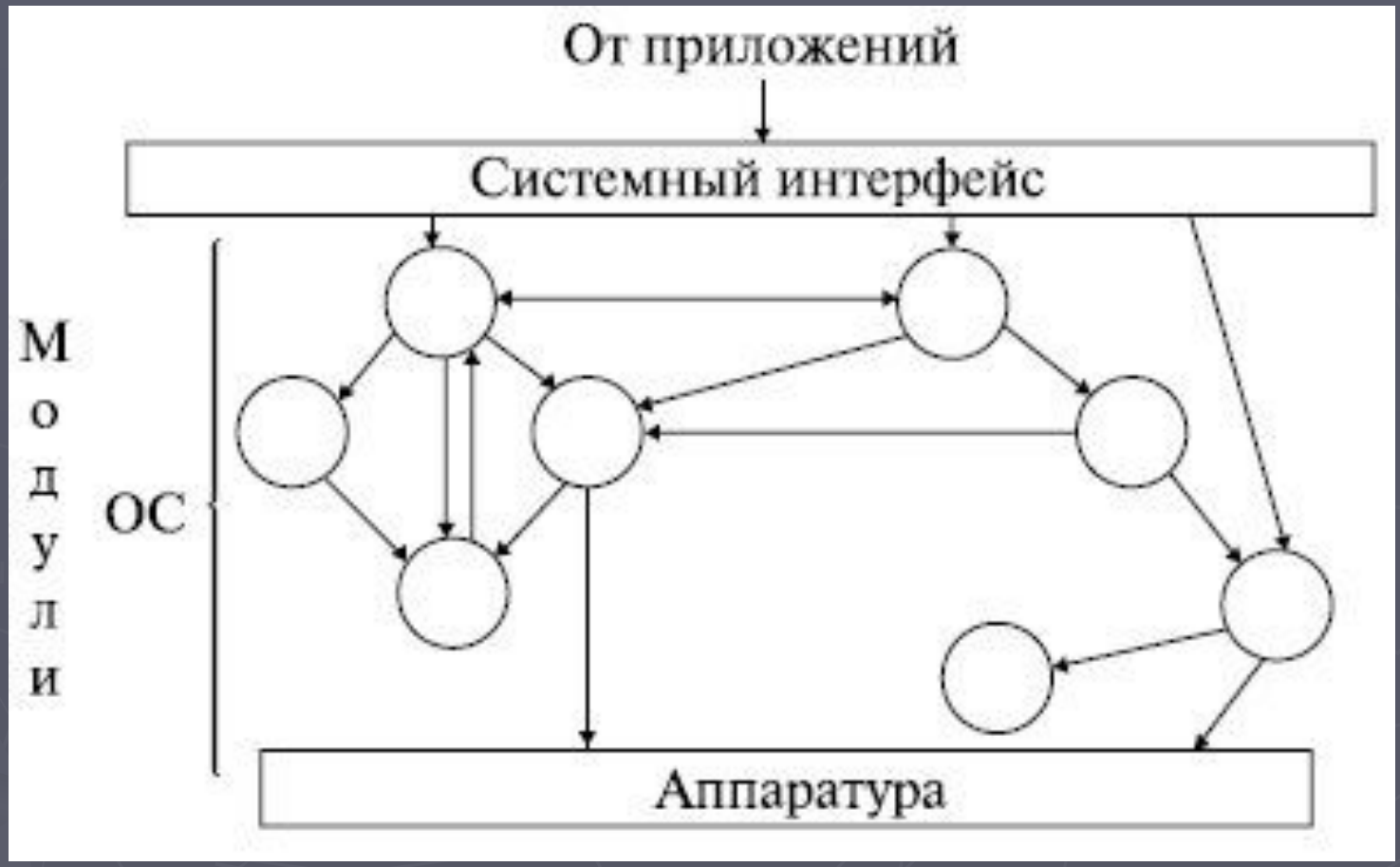
концепция многоуровневой иерархической вычислительной системы;

разделение модулей на две группы по функциям;

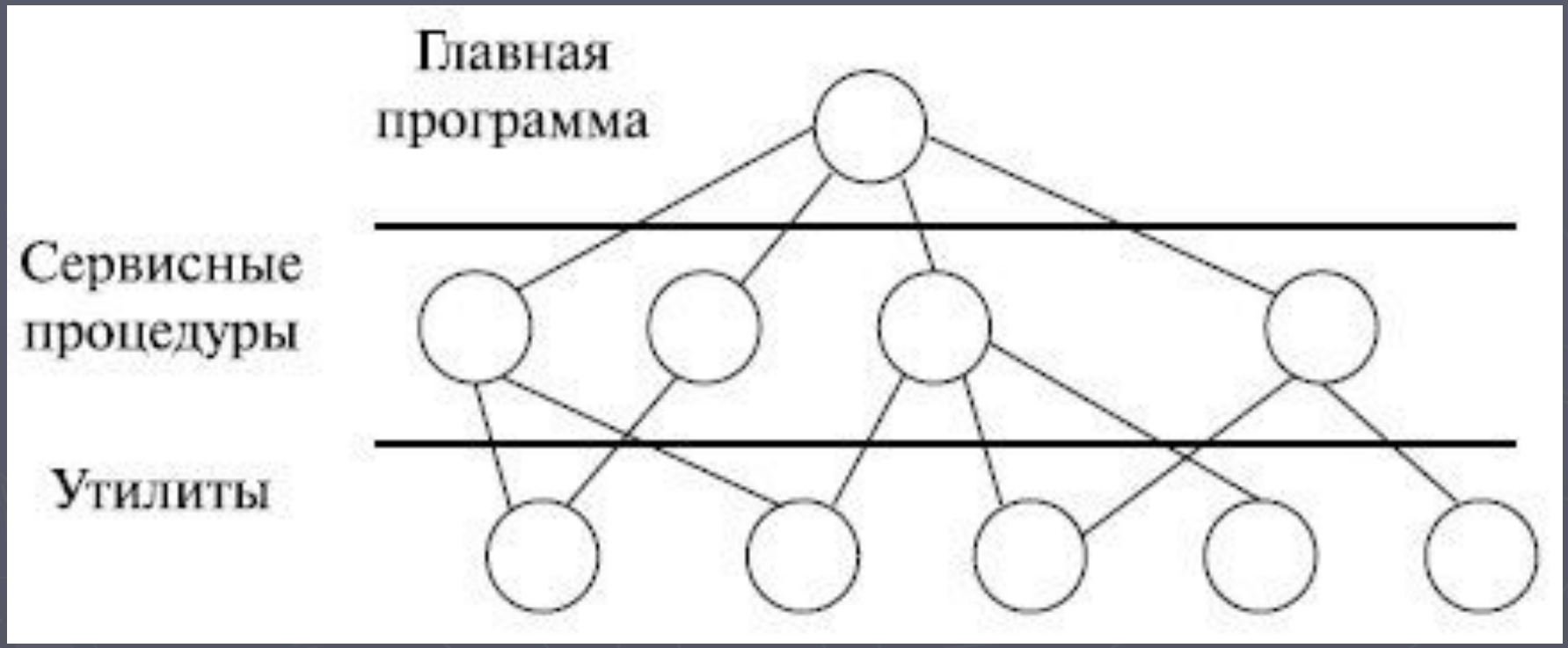
разделение модулей ОС на две группы по размещению в памяти вычислительной системы;

реализация двух режимов работы вычислительной системы: привилегированного режима (Kernel mode), или режима супервизора (supervisor mode), и пользовательского режима (user mode), или режима задачи (task mode);

ограничение функций ядра.



Монолитная архитектура



Структурированная архитектура

Такая организация ОС предполагает следующую структуру:

главная программа, которая вызывает требуемые сервисные процедуры;

набор сервисных процедур, реализующих системные вызовы;

набор утилит, обслуживающих сервисные процедуры.

Вспомогательные модули обычно подразделяются на группы:

утилиты – программы, выполняющие отдельные задачи управления и сопровождения вычислительной системы;

системные обрабатывающие программы – текстовые и графические редакторы (Paint, Imaging в Windows 2000), компиляторы и др.;

программы предоставления пользователю дополнительных услуг (специальный вариант пользовательского интерфейса, калькулятор, игры, средства мультимедиа Windows 2000);

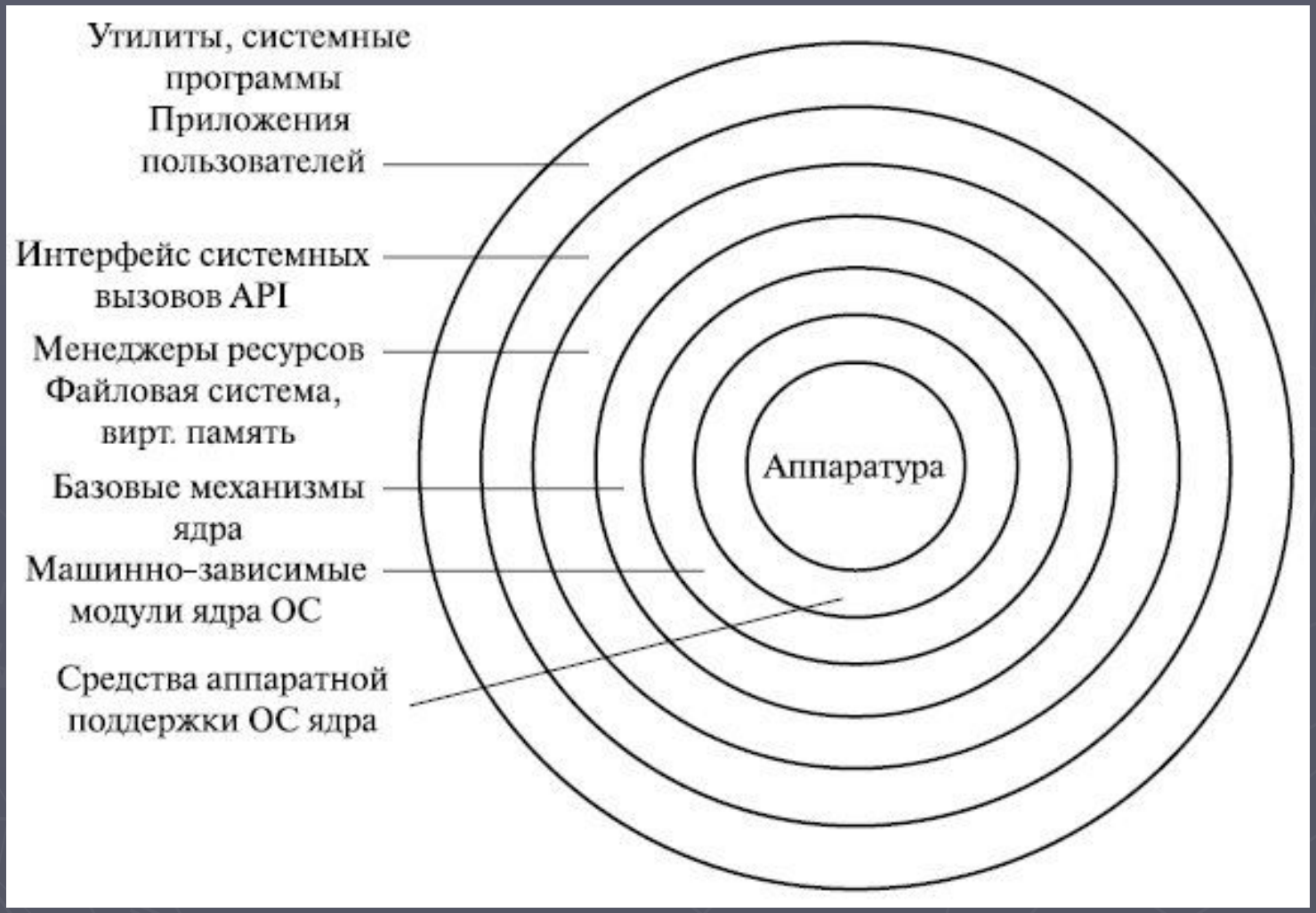
библиотеки процедур различного назначения, упрощения разработки приложений, например, библиотека функций ввода-вывода, библиотека математических функций и т.п.

В концепции многоуровневой (многослойной) иерархической машины структура ОС также представляется рядом слоев.

При такой организации каждый слой обслуживает вышележащий слой, выполняя для него некоторый набор функций, которые образуют межслойный интерфейс.

На основе этих функций следующий верхний по иерархии слой строит свои функции – более сложные и более мощные и т.д.

Кроме того, модули каждого слоя можно изменять без необходимости изменений в других слоях (но не меняя межслойных интерфейсов!).



Многослойная структура ОС

Переход к микроядерной архитектуре

Клиент-серверная архитектура

Обработка системного вызова в микроядерной архитектуре

В то же время признаны следующие достоинства микроядерной архитектуры:

единообразные интерфейсы;
простота расширяемости;
высокая гибкость;
возможность переносимости;
высокая надежность;
поддержка распределенных систем;
поддержка объектно-ориентированных ОС.

Многое зависит от размеров и функциональных возможностей микроядра.

Для возможности представления о размерах микроядер операционных систем в ряде источников приводятся такие данные:

типичное микроядро первого поколения – 300 Кбайт кода и 140 интерфейсов системных вызовов;

микроядро ОС L4 (второе поколение) – 12 Кбайт кода и 7 интерфейсов системных вызовов.

В современных операционных системах различают следующие виды ядер.

Наноядро (НЯ). Крайне упрощённое и минимальное ядро, выполняет лишь одну задачу – обработку аппаратных прерываний, генерируемых устройствами компьютера.

Микроядро (МЯ) предоставляет только элементарные функции управления процессами и минимальный набор абстракций для работы с оборудованием.

Экзоядро (ЭЯ) предоставляет лишь набор сервисов для взаимодействия между приложениями, а также необходимый минимум функций, связанных с защитой

Монолитное ядро (МнЯ) предоставляет широкий набор абстракций оборудования.

Модульное ядро (Мод. Я) – современная, усовершенствованная модификация архитектуры МЯ.

Гибридное ядро (ГЯ) – модифицированные микроядра, позволяющие для ускорения работы запускать "несущественные" части в пространстве ядра. Имеют "гибридные" достоинства и недостатки. Примером смешанного подхода может служить возможность запуска операционной системы с монолитным ядром под управлением микроядра.

Наиболее тесно элементы микроядерной архитектуры и элементы монолитного ядра переплетены в ядре Windows NT. Хотя Windows NT часто называют микроядерной операционной системой, это не совсем так.

Микроядро NT слишком велико (более 1 Мбайт), чтобы носить приставку "микро". Компоненты ядра Windows NT располагаются в вытесняемой памяти и взаимодействуют друг с другом путем передачи сообщений, как и положено в микроядерных операционных системах.

В то же время все компоненты ядра работают в одном адресном пространстве и активно используют общие структуры данных, что свойственно операционным системам с монолитным ядром.