

---

# Вводная лекция

---

## Лекция

Ревизия: 0.3

## **История изменений**

28.10.2010 – Версия 0.1. Первичный документ. Ковтун В.Ю.

06.03.2011 – Версия 0.2. Обновлено рисунки. Ковтун В.Ю., Матвеев Д.

24.02.2011 – Версия 0.3. Обновлено раздел посвященный истории развития ОС. Влад Ковтун.

## Содержание

История изменений	2
Содержание	3
Лекция 1. Вводная лекция	4
Вопросы	4
Введение	4
Расширение возможностей машины	5
Менеджер ресурсов	5
История ОС	6
Краткое отступление	6
Первое поколение (1945-55): электронные лампы и коммутационные панели	6
Второе поколение (1955-65): транзисторы и системы пакетной обработки	7
Третье поколение (1965-1980): интегральные схемы и многозадачность	8
Четвертое поколение (с 1980 года по наши дни): персональные компьютеры	13
Классификация ОС	14
Литература	15

# Лекция 1. Вводная лекция

## Вопросы

1. Введение.
2. История ОС.
3. Классификация ОС.

## Введение

**Вычислительная система** - взаимосвязанная совокупность аппаратных средств вычислительной техники и программного обеспечения, предназначенная для обработки информации.

**Операционная система (ОС)** - комплекс системных и управляющих программ, предназначенных для наиболее эффективного использования всех ресурсов вычислительной системы и удобства работы с ней.

**Назначение ОС** - организация вычислительного процесса в вычислительной системе, рациональное распределение вычислительных ресурсов между отдельными решаемыми задачами; предоставление пользователям многочисленных сервисных средств, облегчающих процесс программирования и отладки задач.

Расположение ОС в структуре компьютера, представлено на рис. 1.

Текстовый редактор	Игра	Почтовый клиент	Web-браузер	Пользовательские приложения (Уровень пользователя)
Компиляторы	Утилиты	Интерпретаторы команд: Java, .NET, Mono И др.		Системные программы (Системный уровень)
Операционная система				
Машинный язык				Оборудование (Аппаратный уровень)
Микроархитектура				
Устройство				

Рис. 1. Структура компьютера (иерархия ПО)

Коротко остановимся на каждом из уровней.

**Оборудование** – электроника: микросхем, проводников, источников питания, трансформаторов и т.д.

**Микроархитектурный уровень** – физические устройства рассматриваются как функциональные единицы, другими словами существуют некие команды управления такими устройствами, кроме того внутри этих устройств также реализованы некие **микропрограммы**, которые реализуют соответствующую функциональность. Например: ЦПУ, АЛУ – содержат набор команд, с помощью которых их можно запрограммировать на языке Ассемблера (ISA – Instruction Set Architecture).

Например, жесткому диску можно дать команду чтения, записав в его регистры, через соответствующие порты ввода-вывода, адрес места на диске, адрес в основной памяти, число байтов для чтения и направление действия (чтение/запись). Реально приходится передавать множество параметров, а статус (результат) выполнения операция – сложен, т.к. возвращается несколько байт, каждый бит либо их комбинация означают некий параметр и после каждого выполнения команды приходится заниматься разбором статуса команды.

**Операционная система позволяет скрыть от программ (программиста-пользователя)** большинство сложностей – общения с аппаратурой напрямую и предоставляет вместо этого упрощенную систему команд (ориентированную на решение наиболее часто возникающих прикладных задач).

Над ОС находится уровень **системных утилит**, которые не являются частью ОС, а предназначены лишь для упрощения выполнения наиболее часто возникающих задач в рамках ОС: дефрагментация дисков, управления файлами и папками и т.д.

**Операционная система** – ПО, которое функционирует в **режиме ядра (режим супервизора)**, которое защищено от вмешательства пользователя с помощью аппаратных средств. Все остальное ПО запускается в **пользовательском режиме**. Отметим, что для современных CPU (архитектуры 0x86), существует несколько уровней изоляции:

- 0 – ядро;
- 1 – драйвер;
- 2 – системный сервис (утилита);
- 3 – пользовательский уровень.

Если пользователю не нравится какой-либо компилятор, он при желании может написать свой собственный, но он не может написать собственный обработчик прерываний системных часов, являющийся частью ОС и обычно защищенный аппаратно от попыток его модифицировать.

Существуют системы, в которых это различие размыто. К ним относятся встроенные системы, они могут не иметь режима ядра, или интерпретируемые системы.

Над системными утилитами находится уровень пользовательских приложений (выполняются в пользовательском режиме) – текстовые процессоры, игры и прочее.

Большинство пользователей компьютеров имеют некоторый опыт общения с ОС, но обычно они испытывают затруднения при попытке дать определение ОС. В известной степени проблема связана с тем, что ОС выполняют **две основные**, но практически не связанные между собой функции:

- расширение возможностей машины;
- управление ее ресурсами.

В зависимости от того, какому пользователю вы зададите вопрос, вы услышите в ответ больше или об одной функции, или о другой. Давайте рассмотрим обе функции.

## Расширение возможностей машины

Как было упомянуто ранее, **архитектура** (система команд, организация памяти, ввод-вывод данных и структура шин) большинства компьютеров на уровне машинного языка примитивна и неудобна для работы с программами, особенно в процессе ввода-вывода данных.

Программа, скрывающая истину об аппаратном обеспечении и представляющая простой список поименованных файлов, которые можно читать и записывать, и **является ОС**. ОС не только устраняет необходимость работы непосредственно с дисками и предоставляет простой, ориентированный на работу с файлами интерфейс, но и скрывает множество неприятной работы с прерываниями, счетчиками времени, организацией памяти и другими элементами низкого уровня. В каждом случае абстракция, предлагаемая ОС, намного проще и удобнее в обращении, чем то, что может предложить нам непосредственно основное оборудование.

## Менеджер ресурсов

Концепция, рассматривающая ОС прежде всего как удобный интерфейс пользователя, — это взгляд сверху вниз. Альтернативный взгляд, снизу вверх, дает представление об ОС как о механизме, присутствующем в устройстве компьютера для управления всеми частями этой сложнейшей машины. Современные компьютеры состоят из CPU, памяти, датчиков времени, дисков, мыши, сетевого интерфейса, принтеров и огромного количества других устройств. В соответствии со вторым подходом работа ОС заключается в обеспечении организованного и контролируемого распределения CPU, памяти и устройств ввода-вывода между различными программами, состоящими за право их использовать.

Когда компьютером (или сетью) пользуются несколько пользователей, необходимость в управлении памятью, устройствами ввода-вывода, другими ресурсами и их защите

сильно возрастает, поскольку пользователи могут обращаться к ним в абсолютно непредсказуемом порядке. К тому же часто приходится распределять между пользователями не только оборудование, но и информацию (файлы, базы данных и т. д.). С этой точки зрения основная задача ОС заключается в отслеживании того, кто и какой ресурс использует, в обработке запросов на ресурсы, в подсчете коэффициента загрузки и разрешении проблем конфликтующих запросов от различных программ и пользователей.

Управление ресурсами включает в себя их **мультиплексирование** (распределение) двумя способами:

- во времени;
- в пространстве.

Когда **ресурс распределяется во времени**, различные пользователи и программы используют его по очереди. Сначала один из них получает доступ к использованию ресурса, потом другой и т. д. Например, несколько программ хотят обратиться к CPU. В этой ситуации ОС сначала разрешает доступ к CPU одной программе, затем, после того как она поработала достаточное время, другой программе, затем следующей и, в конце концов, опять первой. Определение того, как долго ресурс будет использоваться во времени, кто будет следующим и на какое время ему предоставляется ресурс — это задача ОС. Еще один пример временного мультиплексирования — распределение заданий, посылаемых для печати на принтер. Когда задания выстраиваются в очередь для печати на одном принтере, ОС каждый раз нужно принимать решение о том, которое из них будет печататься следующим.

Другой вид распределения — это **пространственное мультиплексирование**. Вместо поочередной работы каждый клиент получает часть ресурса. Обычно оперативная память разделяется между несколькими работающими программами, так что все они одновременно могут постоянно находиться в памяти (например, используя центральный процессор по очереди). Если предположить, что памяти достаточно для того, чтобы хранить несколько программ, эффективнее разместить в памяти сразу несколько программ, чем выделить всю память одной программе, особенно если ей нужна лишь небольшая часть имеющейся памяти. Конечно, при этом возникают проблемы справедливого распределения, защиты памяти и т. д., и для разрешения подобных вопросов существует ОС. Другой ресурс, распределяемый пространственно, — это диск (жесткий). Во многих системах один диск в одно и то же время может содержать файлы нескольких пользователей. Распределение дискового пространства и отслеживание того, кто какие блоки диска использует, является типичной задачей управления ресурсами, которую также выполняет ОС.

## История ОС

### Краткое отступление

История развития ОС насчитывает уже много лет. Так как ОС появились и развивались в процессе конструирования компьютеров, то эти события исторически тесно связаны. Поэтому чтобы представить, как выглядели ОС, рассмотрим следующие друг за другом поколения компьютеров. Такая схема взаимосвязи поколений ОС и компьютеров довольно груба, но она обеспечивает некоторую структуру, без которой ничего не было бы понятно.

Первый настоящий цифровой компьютер был изобретен английским математиком Чарльзом Бэббиджем (Charles Babbage, 1792-1871). Хотя большую часть жизни Бэббидж посвятил попыткам создания своей «аналитической машины», он так и не смог заставить ее работать должным образом. Это была чисто механическая машина, а технологии того времени не были достаточно развиты для изготовления многих деталей и механизмов высокой точности. Не стоит и говорить, что его аналитическая машина не имела ОС.

Интересный исторический факт: Бэббидж понимал, что для аналитической машины ему необходимо ПО, поэтому он нанял молодую женщину по имени Ада Лавлейс (Ada Lovelace), дочь знаменитого британского поэта Лорда Байрона. Она и стала первым в мире программистом, а язык программирования Ada® назван в ее честь.

### Первое поколение (1945-55): электронные лампы и коммутационные панели

На первых машинах использовались механические реле, но они были очень медлительны, длительность такта составляла несколько секунд. Позже реле заменили

электронными лампами. Машины получались громоздкими, заполняющими целые комнаты, с десятками тысяч электронных ламп, но все равно они были в миллионы раз медленнее, чем даже самый дешевый современный персональный компьютер.

В те времена каждую машину и разрабатывала, и строила, и программировала, и эксплуатировала, и поддерживала в рабочем состоянии одна команда. Все программирование выполнялось на абсолютном машинном языке, управления основными функциями машины осуществлялось просто при помощи соединения коммутационных панелей проводами. Тогда еще не были известны языки программирования (даже ассемблера не было). Об ОС никто и не слышал. Обычный режим работы программиста был таков: записаться на определенное время на специальном стенде, затем спуститься в машинную комнату, вставить свою коммутационную панель в компьютер и провести несколько следующих часов в надежде, что во время работы ни одна из двадцати тысяч электронных ламп не выйдет из строя. Фактически тогда на компьютерах занимались только прямыми числовыми вычислениями, например расчетами таблиц синусов, косинусов и логарифмов.

К началу 50-х, с выпуском перфокарт, установившееся положение несколько улучшилось. Стало возможно вместо использования коммутационных панелей записывать и считывать программы с карт, но во всем остальном процедура вычислений оставалась прежней.

## Характеристика

Компьютеры первого поколения были предназначены для решения задач с ядерным оружием. Компьютер состоял из оперативной памяти, CPU, устройства печати данных на узкой ленте. Эти машины имели только **однопользовательский режим** работы. Программа и данные вводились в оперативную память, потом запускались. Пользователь вводил в память машины команды, результаты выполнения программы появлялись на печати. В случае ошибки машина останавливалась, на лампочках загоралась ошибка.

Средство программирования – **машинный язык** (машинный код). Пользователь должен был знать машинные команды и уметь программировать на машинном коде. В таких машинах аппаратные сбои происходили достаточно часто. Позднее аппаратный загрузчик упрощал ввод данных в оперативную память. Появились первые решения в области операций программирования: ассемблеры и автокоды (служебные программы, которые переводили программы с языка Ассемблер на язык машинного кода).

### Что нужно запомнить о компьютерах первого поколения:

- Зарождение класса сервисных, управляющих программ
- Зарождение языков программирования
- Однопользовательский, персональный режим

## Второе поколение (1955-65): транзисторы и системы пакетной обработки

В середине 50-х изобретение и применение транзисторов радикально изменило всю картину. Компьютеры стали достаточно надежными, появилась высокая вероятность того, что машина будет работать довольно долго, выполняя при этом полезные функции. Впервые сложилось четкое разделение между проектировщиками, сборщиками, операторами, программистами и обслуживающим персоналом.

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

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

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

машине. **Системы пакетной обработки** стали прообразом современных ОС, они были первыми системными программами, предназначенными для управления вычислительным процессом.

## Характеристики

Компьютеры второго поколения потребляли меньше электроэнергии и были более производительными: они выполняли несколько миллионов операций в секунду. Появились новые внешние устройства, усовершенствовалось ПО.

Вначале появилась возможность выполнять программы не только по одной, но и пакетами. Внутри пакета могло быть несколько программ, предназначенных для последовательного выполнения. Этот пакет мог представляться как стопка перфокарт, данные на магнитной ленте, перфоленты. Соответственно, появилась специальная управляющая программа, позволявшая координировать обработку заданий из пакета и определять момент, когда задание могло начать выполняться. В случае ошибки управление также передавалось на управляющую программу **мониторной системой**.

Следующий этап – появление компьютеров, в которых поддерживался **мультипрограммный режим**. На обработке могло находиться сразу несколько программ. CPU выполнялась одна программа, другие ожидали или занимались обменом. С появлением таких компьютеров появились ОС. Расширился спектр задач, для решения которых использовались компьютеры.

Появилась проблема **дружественности программных объектов** (интерфейсов). Командные языки (языки управления заданиями) упростили работу пользователя с системой.

Развились средства программирования, доступные для пользователя, в частности, появились языки высокого уровня. Появилась тенденция к аппаратной независимости команд языка программирования. Появилась новая задача: упростить процесс программирования посредством использования языков высокого уровня. Появились проблемно-ориентированные языки программирования.

Появились первые прообразы файловых систем. Нужно было хранить данные вне оперативной памяти. Появление файловых систем упростило процесс организации и хранения данных на внешних устройствах. Появилось понятие именованного набора данных (прообраз виртуального устройства), что дало возможность абстракции пользователя от внешних устройств.

### Что нужно запомнить о компьютерах второго поколения:

- Пакетная обработка заданий.
- Мультипрограммирование.
- Языки управления заданиями.
- Файловые системы.
- Виртуальные устройства.
- Операционные системы.

### Третье поколение (1965-1980): интегральные схемы и многозадачность

В это время в технической базе произошел переход от отдельных полупроводниковых элементов типа транзисторов к интегральным микросхемам. Вычислительная техника становится более надежной и дешевой. Растет сложность и количество задач, решаемых компьютерами. Повышается производительность CPU.

К началу 60-х годов большинство производителей компьютеров имело две отдельные, полностью несовместимые производственные линии. С одной стороны, существовали научные крупномасштабные компьютеры с **пословной** обработкой текста типа IBM 7094, использовавшиеся для числовых вычислений в науке и технике. С другой стороны — коммерческие компьютеры с **посимвольной** обработкой, такие как IBM 1401, широко используемые банками и страховыми компаниями для сортировки и печатания данных.

Развитие и поддержка двух совершенно разных производственных линий для изготовителей были достаточно дорогим удовольствием. Кроме того, многим покупателям изначально требовалась небольшая машина, однако позже ее возможностей становилось недостаточно и требовался более мощный компьютер, который работал бы с теми же самыми программами, но быстрее.

Фирма IBM попыталась решить эти проблемы разом, выпустив серию машин IBM/360. 360-е были серией программно-совместимых машин, варьирующихся от компьютеров размером с IBM 1401 до машин, значительно более мощных, чем IBM 7094. В последующие годы, используя более современные технологии, корпорация IBM выпустила компьютеры, совместимые с 360, эти серии известны под номерами 370, 4300, 3080 и 3090.

360-е стали первой основной линией компьютеров, на которой использовались мелкомасштабные интегральные схемы, дававшие преимущество в цене и качестве по сравнению с машинами второго поколения, созданными из отдельных транзисторов. Корпорация IBM добилась мгновенного успеха, а идею семейства совместимых компьютеров скоро приняли и все остальные основные производители. В компьютерных центрах до сих пор можно встретить потомков этих машин. В настоящее время они часто используются для управления огромными БД (например, для систем бронирования и продажи билетов на авиалиниях) или как серверы узлов Интернета, которые должны обрабатывать миллиарды запросов в секунду.

Основное преимущество «одного семейства» оказалось одновременно и величайшей его слабостью. По замыслу его создателей все программное обеспечение, включая ОС OS/360, должно было одинаково хорошо работать на всех моделях компьютеров: и в небольших системах, которые часто заменяли 1401-е и применялись для копирования перфокарт на магнитные ленты, и на огромных системах, заменяющих 7094-е и использовавшихся для расчета прогноза погоды и других сложных вычислений. Кроме того, предполагалось, что одну ОС можно будет использовать в системах как с несколькими внешними устройствами, так и с большим их количеством; а также как в коммерческих, так и в научных областях. Но самым важным было, чтобы это семейство машин давало результаты независимо от того, кто и как его использует.

Ни IBM, ни кто-либо еще не мог написать ПО, удовлетворяющего всем этим противоречивым требованиям. В результате появилась огромная и необычайно сложная ОС. Она состояла из миллионов строк, написанных на ассемблере тысячами программистов, содержала тысячи и тысячи ошибок, что повлекло за собой непрерывный поток новых версий, в которых пытались исправить эти ошибки. В каждой новой версии устранялась только часть ошибок, вместо них появлялись новые, так что общее их число, вероятно, оставалось постоянным.

При сложных научных вычислениях и ограниченных возможностях CPU устройства ввода-вывода задействовались довольно редко, так что это потраченное впустую время не играло существенной роли. Но при коммерческой обработке данных время ожидания устройства ввода-вывода могло занимать 80% или 90% всего рабочего времени, поэтому необходимо было что-нибудь сделать во избежание длительного простаивания весьма дорогостоящего процессора.

Решение этой проблемы заключалось в **разбиении памяти на несколько частей**, называемых **разделами**, каждому из которых давалось отдельное задание, как показано на рис. 1.4.

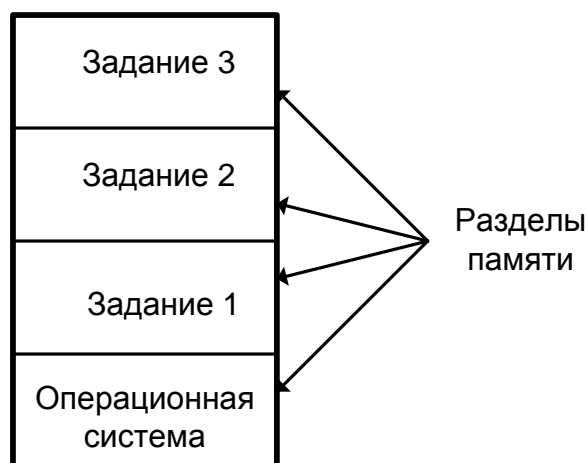


Рис. 2. Многозадачная система с тремя задачами в памяти

Пока одно задание ожидало завершения работы устройства ввода-вывода, другое могло использовать CPU. Если в оперативной памяти содержалось достаточное

количество заданий, CPU мог быть загружен почти на все 100% по времени. Множество одновременно хранящихся в памяти заданий требовало наличия специального оборудования для защиты каждого задания от возможного любопытства и ущерба со стороны остальных заданий. 360-я и другие системы третьего поколения были снабжены подобными аппаратными средствами.

**Другим важным плюсом ОС третьего поколения** стала способность считывать задание с перфокарт на диск по мере того, как их приносили в машинный зал. Всякий раз, когда текущее задание заканчивалось, ОС могла загружать новое задание с диска в освободившийся раздел памяти и запускать его. Этот технический прием называется «подкачкой» данных.

Хотя ОС третьего поколения вполне подходили для больших научных вычислений и справлялись с крупными коммерческими обработками данных, они все еще, по существу, представляли собой разновидности системы пакетной обработки. Многие программисты тосковали по первому поколению машин, когда они могли распоряжаться всей машиной в течение нескольких часов и имели возможность быстро отлаживать свои программы. При системах третьего поколения временной промежуток между передачей задания и возвращением результатов часто составлял несколько часов, так что единственная неуместная запятая могла стать причиной сбоя при компиляции, и получалось, что программист тратил впустую половину дня.

Появление мультипрограммирования требует настоящей революции в строении вычислительной системы. Особую роль здесь играет аппаратная поддержка (многие аппаратные новшества появились еще на предыдущем этапе эволюции), наиболее существенные особенности которой перечислены ниже.

- **Реализация защитных механизмов.** Программы не должны иметь самостоятельного доступа к распределению ресурсов, что приводит к появлению привилегированных и непривилегированных команд. **Привилегированные команды**, например команды ввода-вывода, могут исполняться только ОС. Говорят, что она работает в **привилегированном режиме**. Переход управления от прикладной программы к ОС сопровождается контролируемой сменой режима. Кроме того, это **защита памяти**, позволяющая изолировать конкурирующие пользовательские программы друг от друга, а ОС – от программ пользователей.
- **Наличие прерываний.** Внешние прерывания оповещают ОС о том, что произошло асинхронное событие, например завершилась операция ввода-вывода. **Внутренние прерывания** (сейчас их принято называть **исключительными ситуациями**) возникают, когда выполнение программы привело к ситуации, требующей вмешательства ОС, например деление на ноль или попытка нарушения защиты.
- **Развитие параллелизма в архитектуре.** Прямой доступ к памяти и организация каналов ввода-вывода позволили освободить CPU от рутинных операций.

Не менее важна в организации мультипрограммирования роль ОС. Она отвечает за следующие операции.

- Организация интерфейса между прикладной программой и ОС при помощи системных вызовов.
- Организация очереди из заданий в памяти и выделение процессора одному из заданий потребовало планирования использования процессора.
- Переключение с одного задания на другое требует сохранения содержимого регистров и структур данных, необходимых для выполнения задания, иначе говоря, контекста для обеспечения правильного продолжения вычислений.
- Поскольку память является ограниченным ресурсом, нужны стратегии управления памятью, то есть требуется упорядочить процессы размещения, замещения и выборки информации из памяти.
- Организация хранения информации на внешних носителях в виде файлов и обеспечение доступа к конкретному файлу только определенным категориям пользователей.
- Поскольку программам может потребоваться произвести санкционированный обмен данными, необходимо их обеспечить средствами коммуникации.
- Для корректного обмена данными необходимо разрешать конфликтные ситуации, возникающие при работе с различными ресурсами и предусмотреть координацию программами своих действий, т.е. снабдить систему средствами синхронизации.

Мультипрограммные системы обеспечили возможность более эффективного использования системных ресурсов (например, CPU, памяти, периферийных устройств), но они еще долго оставались **пакетными**. Пользователь не мог непосредственно взаимодействовать с заданием и должен был предусмотреть с помощью управляющих карт все возможные ситуации. Отладка программ по-прежнему занимала много времени и требовала изучения многостраничных распечаток содержимого памяти и регистров или использования отладочной печати.

Появление электронно-лучевых дисплеев и переосмысление возможностей применения клавиатур поставили на очередь решение этой проблемы. Логическим расширением систем мультипрограммирования стали **time-sharing системы**, или **системы разделения времени**). В них CPU переключается между задачами не только на время операций ввода-вывода, но и просто по прошествии определенного времени. Эти переключения происходят так часто, что пользователи могут взаимодействовать со своими программами во время их выполнения, то есть интерактивно. В результате появляется возможность одновременной работы нескольких пользователей на одной компьютерной системе. У каждого пользователя для этого должна быть хотя бы одна программа в памяти. Чтобы уменьшить ограничения на количество работающих пользователей, была внедрена идея неполного нахождения исполняемой программы в оперативной памяти. Основная часть программы находится на диске, и фрагмент, который необходимо в данный момент выполнять, может быть загружен в оперативную память, а ненужный – выкачан обратно на диск. Это реализуется с помощью механизма **виртуальной памяти**. Основным достоинством такого механизма является создание иллюзии неограниченной оперативной памяти ЭВМ.

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

Параллельно внутренней эволюции вычислительных систем происходила и внешняя их эволюция. До начала этого периода вычислительные комплексы были, как правило, несовместимы. Каждый имел собственную ОС, свою систему команд и т. д.

В результате программу, успешно работающую на одном типе машин, необходимо было полностью переписывать и заново отлаживать для выполнения на компьютерах другого типа.

В начале третьего периода появилась идея создания семейств программно совместимых машин, работающих под управлением одной и той же ОС. Первым семейством программно совместимых компьютеров, построенных на интегральных микросхемах, стала серия машин IBM/360. Разработанное в начале 60-х годов, это семейство значительно превосходило машины второго поколения по критерию цена/производительность.

За ним последовала линия компьютеров PDP, несовместимых с линией IBM, и лучшей моделью в ней стала PDP-11.

Желание сократить время ожидания ответа привело к разработке **режима разделения времени**, варианту многозадачности, при котором у каждого пользователя есть свой диалоговый терминал. Если двадцать пользователей зарегистрированы в системе, работающей в режиме разделения времени, и семнадцать из них думают, беседуют или пьют кофе, то CPU по очереди предоставляется трем пользователям, желающим работать на машине. Так как люди, отлаживая программы, обычно выдают короткие команды (например, компилировать процедуру на пяти страницах) чаще, чем длинные (например, упорядочить файл с миллионами записей), то компьютер может обеспечивать быстрое интерактивное обслуживание нескольких пользователей. При этом он может работать над большими пакетами в фоновом режиме, когда CPU не занят другими заданиями. Первая серьезная система с **режимом разделения времени** CTSS (Compatible Time Sharing System — Совместимая система разделения времени) была разработана в Массачусетском технологическом институте (MIT) на специально переделанном компьютере IBM 7094. Однако режим разделения времени не стал действительно популярным до тех пор, пока не получили широкого распространения необходимые технические средства защиты.

После успеха системы CTSS Массачусетский технологический институт, система исследовательских лабораторий Bell Labs и корпорация General Electric (тогда главный изготовитель компьютеров) решили начать разработку «компьютерного предприятия

общественного пользования» — машины, которая должна была поддерживать сотни одновременных пользователей в режиме разделения времени. Образцом для новой машины послужила система распределения электроэнергии. Когда вам нужна электроэнергия, вы просто вставляете штепсель в розетку и получаете энергии столько, сколько вам нужно. Проектировщики этой системы, известной как MULTICS (MULTiplexed Information and Computing Service — мультиплексная информационная и вычислительная служба), представляли себе одну огромную вычислительную машину, воспользоваться услугой которой мог каждый человек в районе Бостона. Мысль о том, что машины, гораздо более мощные, чем их мэйнфрейм GE-645, будут продаваться миллионами по цене тысяча долларов за штуку всего лишь через тридцать лет, казалась чистойшей научной фантастикой, как если бы сегодня кто-либо вздумал проектировать сверхзвуковые трансатлантические подводные поезда.

Успех системы MULTICS был весьма неоднозначен. Эта система разрабатывалась для того, чтобы обеспечить сотни пользователей машиной, немногим более мощной, чем персональный компьютер с процессором Intel 386, хотя при этом имеющей возможность работы со значительно большим количеством устройств ввода-вывода. Это было не так уж безумно, как может показаться, **потому что в те дни люди знали, как писать маленькие, эффективные программы — навык, который впоследствии был утерян.** Существовало много причин, по которым система MULTICS не захватила весь мир. Не последнюю роль сыграл тот факт, что эта система была написана на языке PL/I, а компилятор языка PL/I появился лишь через несколько лет, к тому же первую версию этого компилятора можно было назвать работоспособной с большой натяжкой. Кроме того, система MULTICS была чрезвычайно претенциозна для своего времени, во многом походя на аналитическую машину Чарльза Бэббиджа в девятнадцатом столетии.

Итак, MULTICS подала много конструктивных идей компьютерным теоретикам, но превратить ее в серьезный продукт и добиться коммерческого успеха оказалось намного тяжелее, чем ожидалось. Система исследовательских лабораторий Bell Labs выбыла из проекта, а компания General Electric совсем оставила компьютерный бизнес. Однако Массачусетский технологический институт проявил упорство и со временем получил работающую систему. В конце концов, она была продана как коммерческое изделие компанией Honeywell, купившей компьютерный бизнес General Electric, и установлена примерно в восьмидесяти больших компаниях и университетах по всему миру. Несмотря на свою малочисленность, пользователи системы MULTICS были отчаянно преданы ей. Например, компании General Motors, Ford и Управление национальной безопасности США оставили свои системы MULTICS только в конце 90-х годов, через 30 лет после выхода системы.

К настоящему времени идея компьютерного предприятия общественного пользования выдохлась, но она может благополучно вернуться к жизни в форме массивных централизованных Интернет-серверов (такие услуги предоставляют все компании производители суперкомпьютеров, например NEC), выполняющих основную часть работы, к которым будут присоединены относительно «глупые» пользовательские машины. Мотивировка, вероятно, будет следующей: большинство пользователей не захочет администрировать все более усложняющуюся и привередливую систему компьютера и предпочтет доверить эту работу команде профессионалов, работающих на обслуживающую сервер компанию. Электронная коммерция уже развивается в этом направлении, создаются различные компании, управляющие электронными супермаркетами на многопроцессорных серверах, с которыми соединяются простые машины клиентов. Все это очень напоминает замысел системы MULTICS.

Несмотря на неудачу с точки зрения коммерции, система MULTICS значительно повлияла на последующие операционные системы. Системе MULTICS также посвящен все еще активный web-сайт [www.multicians.org](http://www.multicians.org), с большим количеством информации о системе, ее проектировщиках и пользователях.

Еще одним важным моментом развития во времена третьего поколения был феноменальный рост мини компьютеров, начиная с выпуска машины PDP-1 корпорацией DEC в 1961 году. Компьютеры PDP-1 обладали оперативной памятью, состоящей всего лишь из 4К 18-битовых слов, но стоили они по 120 тысяч долларов за штуку (это меньше 5% от цены IBM 7094) и поэтому расхватывались как горячие пирожки. На некоторых видах нечисловой работы они работали почти с такой же скоростью, как IBM 7094, что дало толчок к появлению новой индустрии. За этой машиной последовала целая серия других PDP (в отличие от семейства IBM, полностью несовместимых), и как кульминация — PDP-11.

Кен Томпсон (Ken Thompson), один из специалистов из Bell Labs, работавший над проектом MULTICS, впоследствии воспользовался PDP-7 для написания усеченной однопользовательской версии системы MULTICS. Эта работа позже развилась в ОС **UNIX®**, ставшую популярной в академическом мире, в правительственных управлениях и во многих компаниях.

Достаточно сказать, что по причине широкой доступности исходного кода различные организации создали свои собственные (несовместимые) версии, что привело к хаосу. Были разработаны две главные версии:

**System V** корпорации AT&T;

**BSD** (Berkeley Software Distribution) Калифорнийского университета Беркли.

Эти системы, в свою очередь, распадаются на отдельные разновидности. Чтобы стало возможным писать программы, работающие в любой UNIX-системе, Институт инженеров по электротехнике и электронике IEEE разработал стандарт системы UNIX, называемый **POSIX**, который теперь поддерживают большинство версий UNIX. Стандарт POSIX определяет минимальный интерфейс системного вызова, который должны поддерживать совместимые системы UNIX. Некоторые другие ОС теперь тоже поддерживают интерфейс POSIX.

Желание иметь свободно распространяемую рабочую (в противоположность образовательной) версию MINIX подвигло финского студента Линуса Торвальдса (Linus Torvalds) к написанию системы **Linux**. Эта система была разработана на основе MINIX и первоначально обладала ее характерными особенностями (например, поддерживала ту же файловую систему). С тех пор система Linux была значительно расширена, но она все еще сохраняет большую часть структуры, общей как для системы MINIX, так и для системы UNIX (на которой и была основана система MINIX). Большая часть того, что будет сказано о UNIX, применимо к System V, BSD, MINIX, Linux, а также к другим версиям и клонам UNIX.

## Характеристика

Компьютеры третьего поколения потребляли меньше электроэнергии и были более компактными. Основная их особенность – **унификация программных и аппаратных узлов и устройств**. Появились **семейства** компьютеров. **Семейство ЭВМ** – это серия, состоящая из компьютеров разной комплектации, производительности и стоимости, но объединенная требованием программной преемственности снизу вверх. Появилась возможность аппаратной модификации компьютеров, а также унификация программных интерфейсов. Появление семейств компьютеров создало возможность увеличения сроков жизни программных систем (раньше для каждой новой модели компьютера программное обеспечение переписывалось, теперь же все программы, написанные для младших моделей семейства, работали и на старших).

В ОС появились **драйверы** устройств. Появились **правила разработки драйверов**, а также внесения в систему новых драйверов. Развитие получили концепции виртуальных устройств, управляющие программы которых имели унифицированный интерфейс.

### Что нужно запомнить о компьютерах третьего поколения:

- Аппаратная унификация узлов и устройств.
- Создание семейств компьютеров.
- Унификация компонентов программного обеспечения.

## Четвертое поколение (с 1980 года по наши дни): персональные компьютеры

Следующий период в эволюции операционных систем связан с появлением Больших Интегральных Схем (LSI, Large Scale Integration) — кремниевых микросхем, содержащих тысячи транзисторов на одном квадратном сантиметре. С точки зрения архитектуры персональные компьютеры (первоначально называемые микрокомпьютерами) были во многом похожи на мини-компьютеры класса PDP-11, но, конечно, отличались по цене. Если появление мини-компьютеров позволило отделам компаний и факультетам университетов иметь собственный компьютер, то с появлением микропроцессоров каждый человек получил возможность купить свой собственный персональный компьютер.

С середины 80-х годов начали расти и развиваться сети персональных компьютеров, управляемых сетевыми и распределенными ОС. В сетевой операционной системе

пользователи знают о существовании многочисленных компьютеров, могут регистрироваться на удаленных машинах и копировать файлы с одной машины на другую. Каждый компьютер работает под управлением локальной ОС и имеет своего собственного локального пользователя (или пользователей).

Сетевые ОС несущественно отличаются от однопроцессорных ОС. Ясно, что они нуждаются в сетевом интерфейсном контроллере и специальном низкоуровневом ПО, поддерживающем работу контроллера, а также в программах, разрешающих пользователям удаленную регистрацию в системе и доступ к удаленным файлам. Но эти дополнения, по сути, не изменяют структуры ОС.

Распределенная ОС, напротив, представляется пользователям традиционной single-CPU системой, хотя она и составлена из множества CPU (находящихся на удаленных компьютерах). При этом пользователи не должны беспокоиться о том, где работают их программы или где расположены файлы; все это автоматически и эффективно обрабатывается самой ОС.

Чтобы создать настоящую распределенную ОС, недостаточно просто добавить несколько страниц кода к single-CPU ОС, так как распределенные и централизованные системы имеют существенные различия. Сейчас активно развивается направление кластеризации – организации нескольких отдельных компьютеров через высокопроизводительный сетевой интерфейс в сеть и представляющих, с точки зрения пользователя, единый вычислительный ресурс [6]. Распределенные системы, например, часто позволяют прикладным задачам одновременно обрабатываться на нескольких CPU, поэтому требуется более сложный алгоритм загрузки CPU для оптимизации распараллеливания.

Наличие задержек при передаче данных в сетях означает, что эти алгоритмы должны работать с неполной, устаревшей или даже неправильной информацией. Эта ситуация радикально отличается от single-CPU системы, в которой ОС обладает полной информацией относительно состояния системы.

## Характеристика

Появились стандарты аппаратных интерфейсов.

Появились персональные компьютеры, а также встраиваемые системы (компьютеры, встраиваемые в бытовое и технологическое оборудование).

Произошла революция в области программного обеспечения, и, в частности, в области ОС:

- Появилась ОС Unix. При создании Unix был разработан и включен в нее язык C, что дало возможность создание **переносимого программного обеспечения**; позднее, большая часть ядра Unix была переписана на C, что сделало переносимой и саму Unix.
- Были разработаны **персональные ОС**, предназначенные для работы непрофессиональных пользователей с компьютером.
- ОС развивались и унифицировались, происходила дальнейшая **стандартизация программных интерфейсов** и увеличение их «дружественности».

Получили развитие сетевые технологии, появились открытые сетевые технологии. Был создан Интернет. Появились принципиально новые задачи, связанные с безопасностью хранения и передачи данных.

## Что нужно запомнить о компьютерах четвертого поколения:

- «Дружественность» пользовательских интерфейсов
- Сетевые технологии
- Безопасность хранения и передачи данных

## Классификация ОС

В соответствии с условиями применения различают три режима ОС:

- пакетной обработки;
- разделения времени;
- реального времени.

В режиме **пакетной обработки** ОС последовательно выполняет собранные в пакет задания. В этом режиме пользователь не имеет контакта с ЭВМ, получая лишь результаты вычислений. В режиме **разделения времени** ОС одновременно выполняет несколько задач, допуская обращение каждого пользователя к ЭВМ. В режиме **реального времени** ОС обеспечивает управление объектами в соответствии с принимаемыми входными сигналами. Время отклика ЭВМ с ОС реального времени на возмущающее воздействие должно быть минимальным.

Ниже остановимся на классификации ОС, согласно задачам, которые решают ОС.

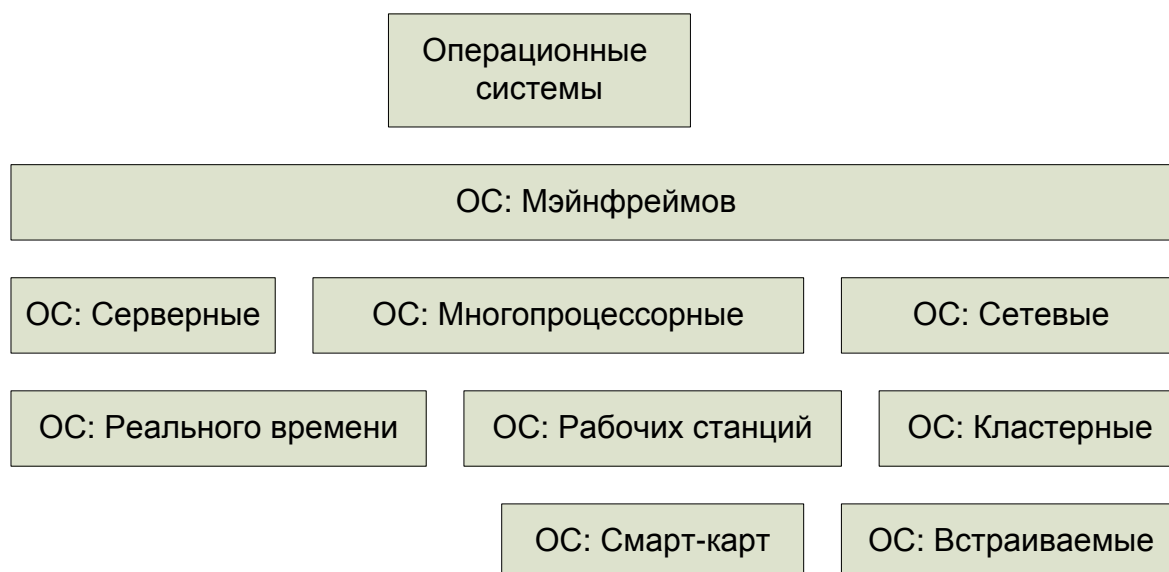


Рис. 3. Основная классификация ОС по их назначению

## Литература

1. Э. Таненбаум. Современные операционные системы. 2-ое изд. –СПб.: Питер, 2002. –1040 с.
2. А. Шоу. Логическое проектирование операционных систем. Пер. с англ. –М.: Мир, 1981. –360 с.
3. С. Кейслер. Проектирование операционных систем для малых ЭВМ: Пер. с англ. –М.: Мир, 1986. –680 с.
4. Э. Таненбаум, А. Вудхалл. Операционные системы: разработка и реализация. Классика CS. –СПб.: Питер, 2006. –576 с.
5. Операционная система OS/2. URL: <http://ru.wikipedia.org/wiki/OS/2>
6. Кластер (группа компьютеров). URL: [http://ru.wikipedia.org/wiki/Кластер\\_\(группа\\_компьютеров\)](http://ru.wikipedia.org/wiki/Кластер_(группа_компьютеров))