

Кафедра информатики и программирования

# Операционные системы

s\_nazarov@mail.ru

Лектор : доктор технических наук,  
профессор Назаров С.В.



## Объем дисциплины и виды учебной работы

Таблица 1

| Вид учебной работы                   | Часы       | Семестры<br>(триместры, модули) |            |
|--------------------------------------|------------|---------------------------------|------------|
|                                      |            | 4                               | 5          |
| <b>Общая трудоёмкость дисциплины</b> | <b>216</b> | <b>108</b>                      | <b>108</b> |
| <i>Аудиторные занятия</i>            | <b>102</b> | <b>51</b>                       | <b>51</b>  |
| Лекции (Л)                           | <b>34</b>  | <b>17</b>                       | <b>17</b>  |
| Практические занятия (ПЗ)            | <b>68</b>  | <b>34</b>                       | <b>34</b>  |
| <i>Самостоятельная работа</i>        | <b>114</b> | <b>57</b>                       | <b>57</b>  |
| В семестре                           | <b>114</b> | <b>57</b>                       | <b>39</b>  |
| В сессию                             | <b>72</b>  |                                 | <b>18</b>  |



# Литература

## Основная

- Назаров С.В. Операционные среды, системы и оболочки. Основы структурной и функциональной организации. – М.: КУДИЦ-ПРЕСС, 2007*
- Назаров С.В., Широков А.И. Современные операционные системы. . – М.: Интернет-Университет Информационных технологий: Бином. Лаборатория знаний, 2010*
- Назаров С.В., Гудыно Л.П., Кириченко А.А. Операционные системы. Практикум. Учеб. пособие. – М.: КНОРУС, 2011*
- Назаров С.В., Гудыно Л.П., Кириченко А.А. Операционные системы. Практикум. Учеб. пособие. – М.: КУДИЦ-ПРЕСС, 2008.*

## Дополнительная

1. Олифер В.Г., Олифер Н.А. Сетевые операционные системы. СПб.: Питер, 2008
2. Столингс В. Операционные системы. М.: Вильямс, 2006
3. Таненбаум Э. Современные операционные системы. Изд-е 3. СПб., Питер, 2010
4. Чекмарев А.Н. Microsoft Windows 7. Руководство администратора. – СПб.: БХВ-Петербург, 2010
5. Райтман М.А. Установка и настройка Windows 7 для максимальной производительности. – СПб.: БХВ-Петербург, 2010
6. Руссинович М., Соломон Д. Внутреннее устройство Microsoft Windows: Windows Server 2003, Windows XP и Windows 2000. Мастер-класс. / Пер. с англ. – 4-е изд. – М.: Издательство «Русская редакция»; Спб.: Питер, 2006.



# **Тема 1. Введение. Назначение, функции и архитектура операционных систем.**

## **Основные определения и понятия**

- 1.1. Определение операционной системы (ОС). Место ОС в программном обеспечении вычислительных систем**
- 1.2. Эволюция операционных систем**
- 1.3. Назначение, состав и функции ОС**
- 1.4. Архитектуры операционных систем**
- 1.5. Классификация операционных систем**
- 1.6. Эффективность и требования, предъявляемые к ОС**
- 1.7. Множественные прикладные среды. Совместимость**
- 1.8. Способы работы с программами разных операционных систем на одном компьютере**
- 1.9. Виртуализация от Microsoft**
- 1.10. Технология Virtuozzo**
- 1.11. Открытая платформа виртуализации VirtualBox**
- 1.12. Установка и конфигурирование операционных систем**



# 1.1. Определение операционной системы (ОС). Место ОС в программном обеспечении вычислительных систем

**1946 г. – ENIAC (Electronic Numerical Integrator and Computer) – полное отсутствие какого-либо ПО, программирование путем коммутации устройств.**

Начало 50-х г. – появление алгоритмических языков и системного ПО.

**Усложнение процесса выполнения программ:**

1. Загрузка нужного транслятора.
2. Запуск транслятора и получение программы в машинных кодах.
3. Связывание программы с библиотечными подпрограммами.
4. Запуск программы на выполнение.
5. Вывод результатов работы на печатающее или другое устройство.

Для повышения эффективности использования ЭВМ вводятся операторы, затем разрабатываются управляющие программы – мониторы - прообразы операционных систем.

**1952 г. – Первая ОС создана исследовательской лабораторией фирмы General Motors для IBM-701.**

**1955 г. – ОС для IBM-704. Конец 50-х годов: язык управления заданиями и пакетная обработка заданий.**



**1963 г. – ОС MCP (Главная управляющая программа) для компьютеров B5000 фирмы Burroughs: мультипрограммирование, мультипроцессорная обработка, виртуальная память, возможность отладки программ на языке исходного уровня, сама ОС написана на языке высокого уровня.**

**1963 г. – ОС CTSS (Compatible Time Sharing System – совместимая система разделения времени для компьютера IBM 7094 – Массачусетский технологический институт.**

**1963 г. – ОС MULTICS (Multiplexed Information and Computing Service) – Массачусетский технологический институт.**

**1974 г. – (UNICS) UNIX (Uniplexed Information and Computing Service) для компьютера PDP-7, публикация статьи Ритчи (С) и Томпсона.**

**1981 г. – PC (IBM), DOS (Seattle Computer Products) – MS DOS (Б. Гейтс).**

**1983г. – Apple, Lisa с Apple, Lisa с GUI (Даг Энгельбарт – Стэнфорд).**

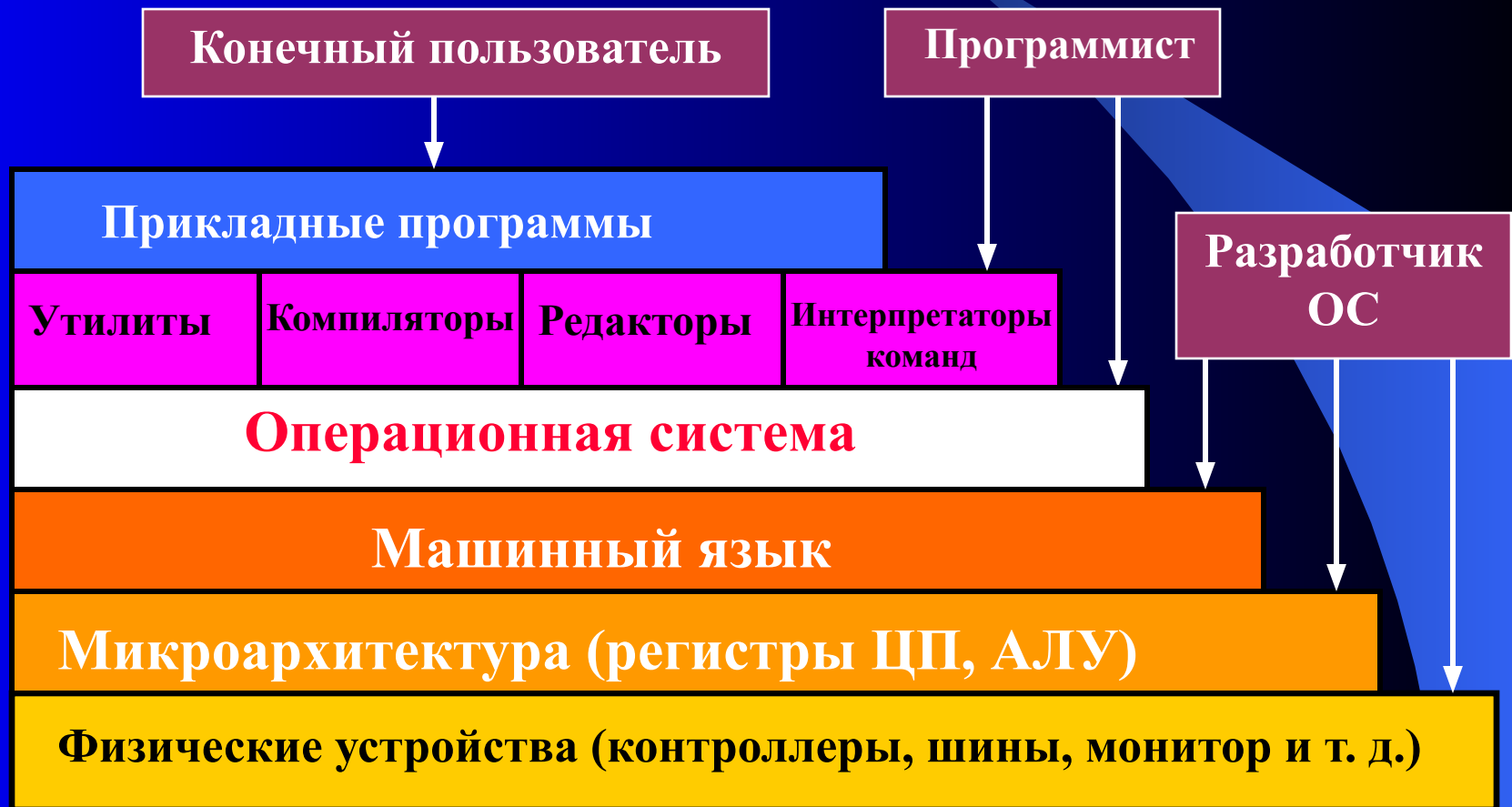
**1985 г. – Windows, X Windows и Motif (для UNIX).**

**1987 г. – MINIX (Э. Таненбаум) – 11800 стр. С и 800 ассемблер (микроядро – 1600 С и 800 ассемблер)**

**1991 г. – Linux (Линус Торвальдс).**



# Расположение ОС в иерархической структуре программного и аппаратного обеспечения компьютера



# ОПЕРАЦИОННАЯ СИСТЕМА

- это набор программ, контролирующих работу прикладных программ и системных приложений и исполняющих роль интерфейса между пользователями, программистами, приложениями и аппаратным обеспечением компьютера.

# ОПЕРАЦИОННАЯ СРЕДА

- программная среда, образуемая операционной системой, определяющая интерфейс прикладного программирования (API) как множество системных функций и сервисов (системных вызовов), предоставляемых прикладным программам.

# ОПЕРАЦИОННАЯ ОБОЛОЧКА

- часть операционной среды, определяющая интерфейс пользователя, его реализацию (текстовый, графический и т.п.), командные и сервисные возможности пользователя по управлению прикладными программами и компьютером





# 1.2. Эволюция операционных систем



- 1970 Динамическое распределение основной памяти  
Разделение времени, многотерминальные системы  
UNIX (PDP-7), Ken Thompson
- 1965 Управляемое мультипрограммирование  
Классическое мультипрограммирование, OS/360  
ОС CTSS (1963), MULTICS (начало работ)  
Оверлейные структуры  
Логическая система управления вводом-выводом
- 1960 Системы прерываний, контрольные точки  
Управление файлами, таймеры  
Спулинг (SPOOL)  
Мониторы
- 1955 Методы доступа, полибуферизация  
Загрузчики, редакторы связей
- 1950 Диагностические программы  
Ассемблеры, макрокоманды  
Библиотеки подпрограмм
- 1946 Первый компьютер





- 2012 Windows 8
- 2008 Windows Server 2008
- 2007 Windows Vista, Windows 7
- 2005 Windows 2003, 64-разрядная
- 2003 Windows 2003 .NET Framework, MAC OS X
- 2000 Windows 2000
- Windows 4.0 – 1996
- 1995 Windows 95
- Корпоративные информационные системы
- NetWare 4.0 – 93, Windows NT 3.1 – 93
- Linux 0.01 - 1993
- 1990 MINIX – 87 (11800 стр. C + 800 стр. Asm.)
- OS/2 - 87
- 1985 OS-Net (Novell) - 83, MS-Net - 84, Windows 1.0 – 85
- Интернет (1983), Персональные компьютеры (1981)
- MS DOS 1.0 – (1981)
- 1980 Сети ЭВМ, UNIX, TCP/IP
- Локальные сети
- 1975 SNA (System Network Architecture), MULTICS
- Протокол X.25, телеобработка, базы данных
- 1965 Виртуальная ЭВМ, виртуальная память



# Операционные системы IBM

1. BPS/360 (Базовая программная поддержка)
2. BOS/360 (Базовая операционная система)
3. TOS/360 (Ленточная операционная система)
4. DOS/360 (Дисковая операционная система)
5. OS/360 – PCP (Первичная управляющая программа)
6. OS/360 – MFT (Мультипрограммирование с фиксированным числом задач)
7. OS/360 – MVT (Мультипрограммирование с переменным числом задач)
8. OS/360 – VMS (Система с переменной памятью)
9. CP-67/CMS (Управляющая программа 67/ диалоговая мониторная система)
10. DOS/VS (Дисковая виртуальная система)
11. OS/VS1 (Виртуальная система 1)
12. OS/VS2 (Виртуальная система 2)
13. VM/370 (Виртуальная машина)



# 1.3. Назначение, состав и функции ОС

## Назначение

### 1. Обеспечение удобного интерфейса [приложения, пользователь] - компьютер за счет предоставляемых сервисов:

- 1.1. Инструменты для разработки программ
- 1.2. Автоматизация исполнения программ
- 1.3. Единообразный интерфейс доступа к устройствам ввода-вывода
- 1.4. Контролируемый доступ к файлам
- 1.5. Управление доступом к совместно используемой ЭВМ и ее ресурсам
- 1.6. Обнаружение ошибок и их обработка
- 1.7. Учет использования ресурсов

### 2. Организация эффективного использования ресурсов ЭВМ

- 2.1. Планирование использования ресурса
- 2.2. Удовлетворение запросов на ресурсы
- 2.3. Отслеживание состояния и учет использования ресурса
- 2.4. Разрешение конфликтов между процессами, претендующими на одни и те же ресурсы



### **3. Облегчение процессов эксплуатации аппаратных и программных средств вычислительной системы**

**3.1. Широкий набор служебных программ (утилит), обеспечивающих резервное копирование, архивацию данных, проверку, очистку, дефрагментацию дисковых устройств и др.**

**3.2. Средства диагностики и восстановления работоспособности вычислительной системы и операционной системы:**

- диагностические программы для выявления ошибок в конфигурации ОС;**
- средства восстановления последней работоспособной конфигурации;**
- средства восстановления поврежденных и пропавших системных файлов и др.**

### **4. Возможность развития**

**4.1. Обновление и возникновение новых видов аппаратного обеспечения**

**4.2. Новые сервисы**

**4.3. Исправления (обнаружение программных ошибок)**

**4.4. Новые версии и редакции ОС**



# Состав компонентов и функции операционной системы:

1. Управление процессами
2. Управление памятью
3. Управление файлами
4. Управление внешними устройствами
5. Защита данных
6. Администрирование
7. Интерфейс прикладного программирования
8. Пользовательский интерфейс



# 1.4. Архитектуры операционных систем

## ОСНОВНЫЕ ПРИНЦИПЫ РАЗРАБОТКИ АРХИТЕКТУРЫ ОПЕРАЦИОННЫХ СИСТЕМ:

1. Концепция многоуровневой иерархической вычислительной системы (виртуальной машины) с ОС многослойной структуры.
2. Разделение модулей ОС по функциям на две группы: ядро – модули, выполняющие основные функции ОС, и модули, выполняющие остальные (вспомогательные) функции.
3. Разделение модулей ОС по размещению в памяти вычислительной системы: резидентные, постоянно находящиеся в оперативной памяти, и транзитные, загружаемые в оперативную память только на время выполнения своих функций.
4. Реализация двух режимов работы вычислительной системы: привилегированного режима (режима ядра – kernel mode) или режима супервизора (supervisor) и пользовательского режима (user mode) или режима задача (task mode).
5. Ограничение функций ядра (а, следовательно и числа его модулей) до минимально необходимых функций.



6. Модульное строение (однократно используемые – при загрузке ОС) и повторно используемые (привилегированные – не допускают прерываний, реентерабельные – допускают прерывания и повторный запуск, повторновходимые – допускают прерывания после завершения секций).
7. Параметрическая универсальность. Возможность генерации ОС и создания нескольких рабочих конфигураций.
8. Функциональная избыточность.
9. Функциональная избирательность.
10. Открытость, модифицируемость, расширяемость (возможность получения текстов исходных модулей).
11. Мобильность – возможность переноса на различные аппаратные платформы.
12. Совместимость – возможность выполнения приложений, рассчитанных на другие ОС.
13. Безопасность – защита от несанкционированного доступа, защита легальных пользователей друг от друга, аудит, возможность восстановления ОС после сбоев и отказов.





## Модульно – интерфейсный подход (структурный подход)

1. Декомпозиция системы на модули по структурному или функциональному признаку.
2. Модули и их взаимные связи образуют абстракцию системы высокого уровня.
3. Описывается каждый модуль и определяется его интерфейс.
4. Проводится декомпозиция каждого модуля и т. д.

Спецификации модулей и их интерфейсов дают структурную основу для проектирования каждого модуля и всей системы в целом.

Правильное определение и выделение модулей представляет собой сложную задачу. Тесно связанные между собой части системы должны входить в один и тот же модуль.

Разработчики программного обеспечения начинают работу с очень грубого и неполного наброска схемы системы и преждевременно обращают внимание на детали отдельных модулей. Поэтому решения, влияющие на систему глобальным образом, принимаются не из тех предпосылок, из которых нужно и без ясного понимания их последствий.

Преждевременная реализация приводит к неустойчивости программного обеспечения, которая часто требует огромных усилий по поддержанию системы.



## Многослойная (иерархическая) структура операционной системы и метод проектирования «сверху вниз» и «снизу вверх»

1. Операционная система представляется в виде иерархии слоев.
2. Верхний слой определяет виртуальную машину с желаемыми свойствами.
3. Каждый следующий слой детализирует вышележащий, выполняя для него некоторый набор функций.
4. Межслойные интерфейсы подчиняются строгим правилам. Связи внутри слоя могут быть произвольными.
5. Отдельный модуль слоя  $L(i)$  может выполнить работу самостоятельно или последующим вариантам: обратиться только к слою  $L(i-1)$ ; обратиться к некоторой команде определенного слоя  $L(q)$ , который выполняет требуемую функцию ( $i-2 \leq q \leq 0$ ); обратиться к любому последующему слою  $L(s)$ , ( $i-2 \leq s \leq 0$ ).

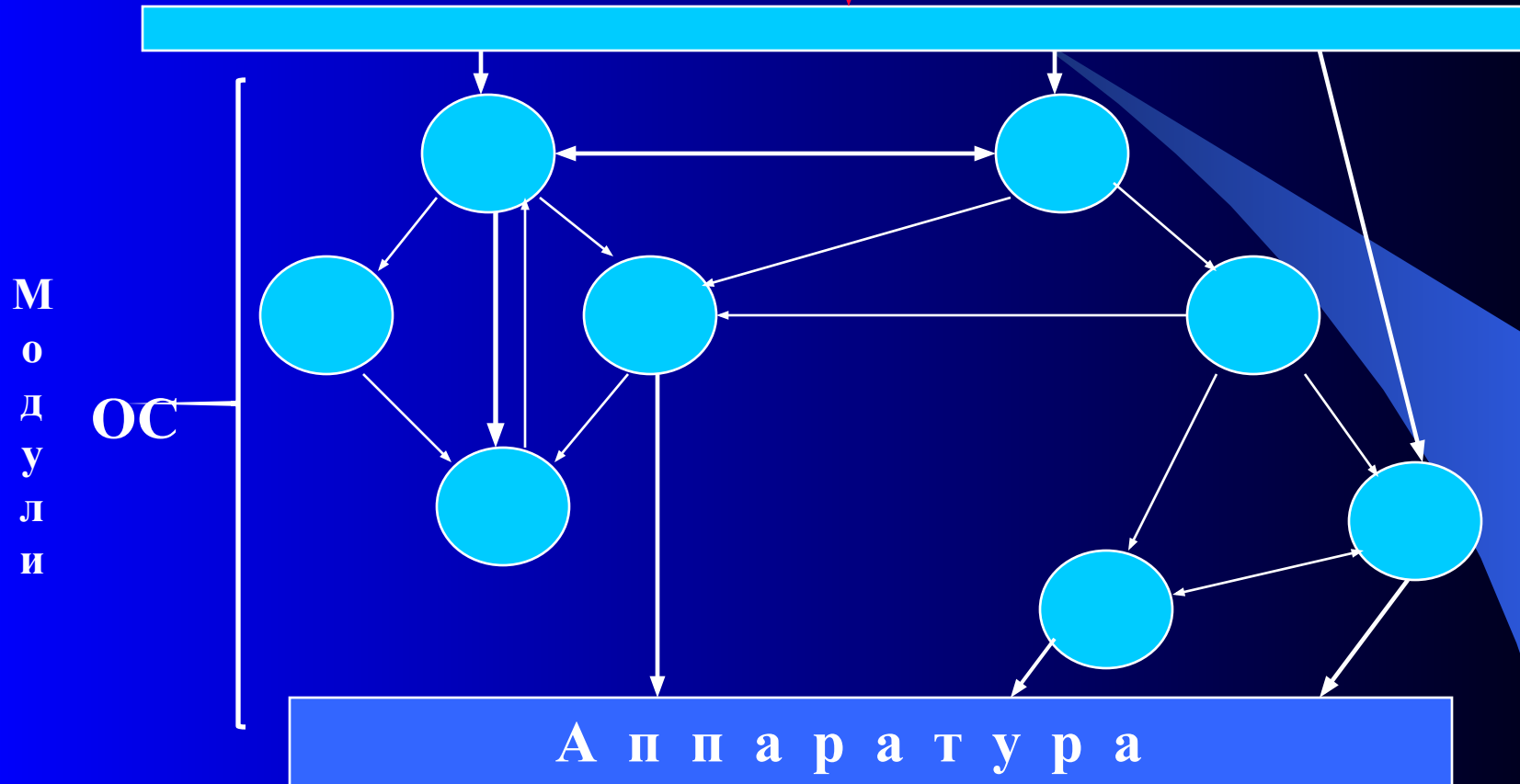
### Достоинства:

1. Между уровнями можно организовать четкий интерфейс.
2. Систему можно спроектировать методом «сверху вниз», а реализовать методом «снизу вверх».
3. Уровни реализуются в соответствии с их порядком, начиная с аппаратуры и далее вверх.
4. Каждую новую виртуальную машину можно детально проверить, после чего продолжать дальнейшую работу.
5. Любой слой достаточно просто модифицировать, не затрагивая другие слои и не меняя межслойные интерфейсы.



# Монолитная архитектура операционной системы

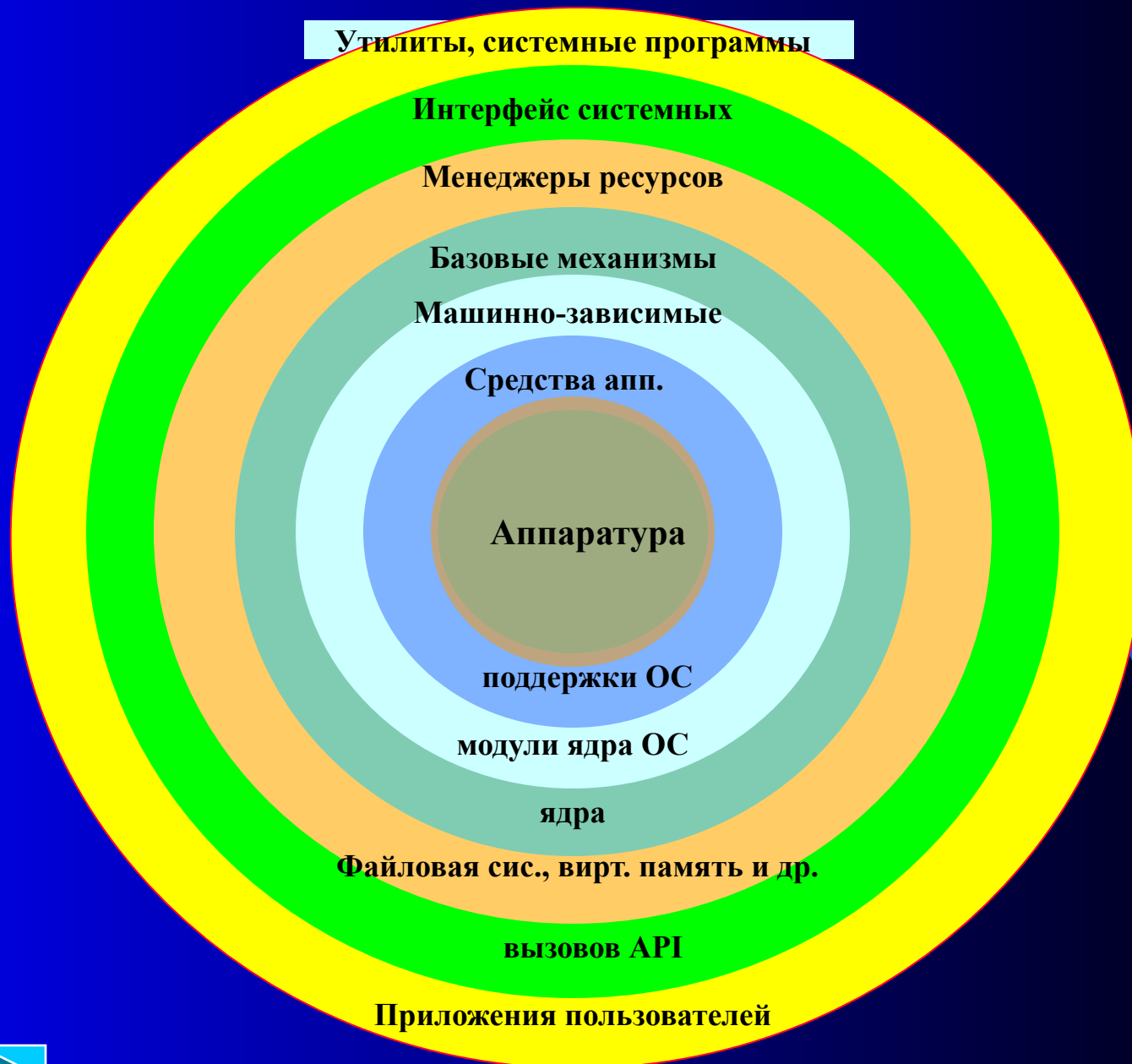
От приложений  
системный интерфейс



**Пример: ранние версии ядра UNIX, Novell NetWare. Каждая процедура имеет хорошо определенный интерфейс в терминах параметров и результатов и может любую другую для выполнения нужной работы.**



# АРХИТЕКТУРА МНОГОУРОВНЕВОЙ ОПЕРАЦИОННОЙ СИСТЕМЫ



# Смена режимов при выполнении вызова функции ядра

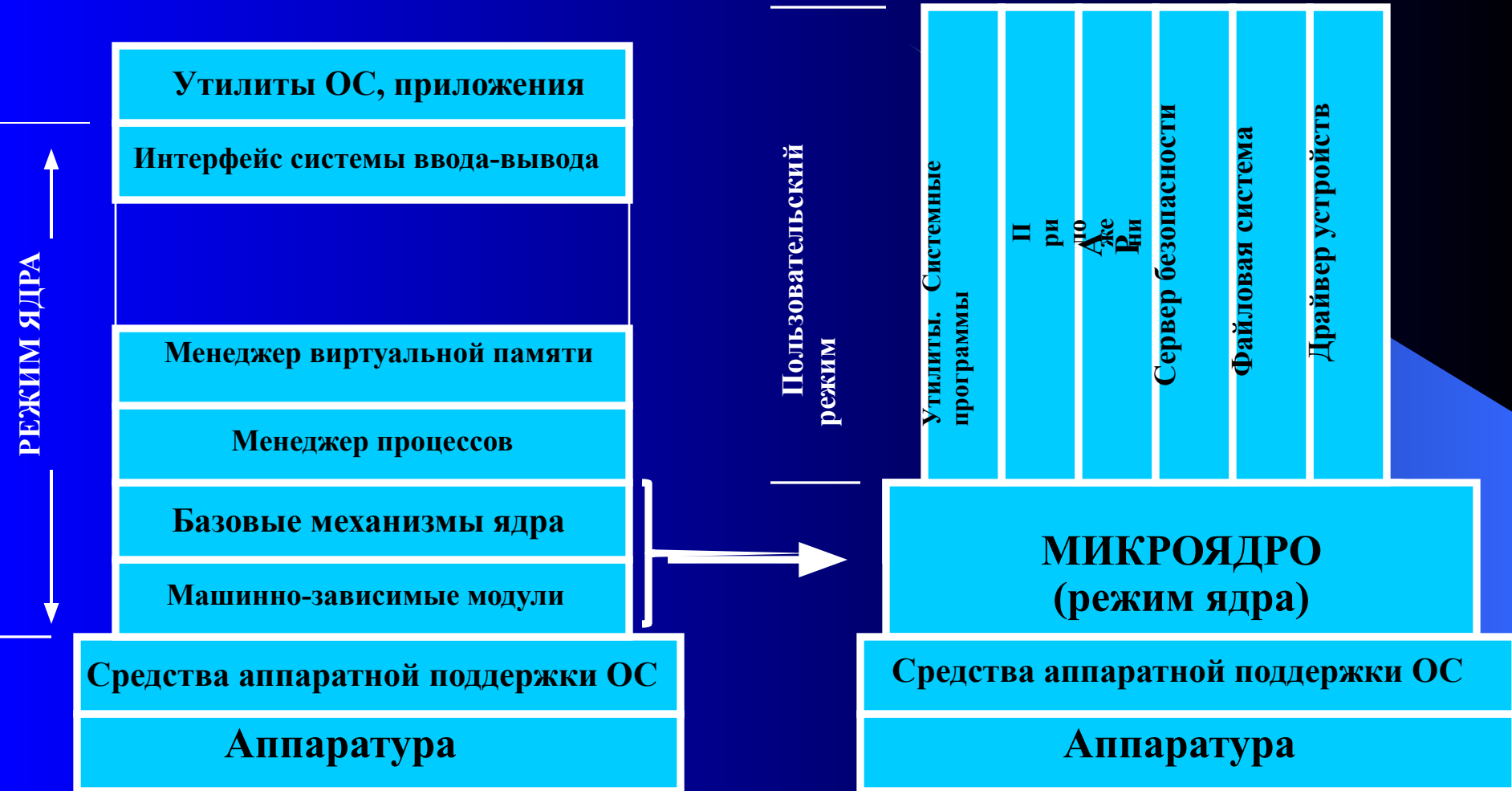


Недостатки иерархической организации ОС:

1. Значительные изменения одного из уровней могут иметь трудно предвидимое влияние на смежные уровни.
2. Многочисленные взаимодействия между соседними уровнями усложняют обеспечение безопасности.



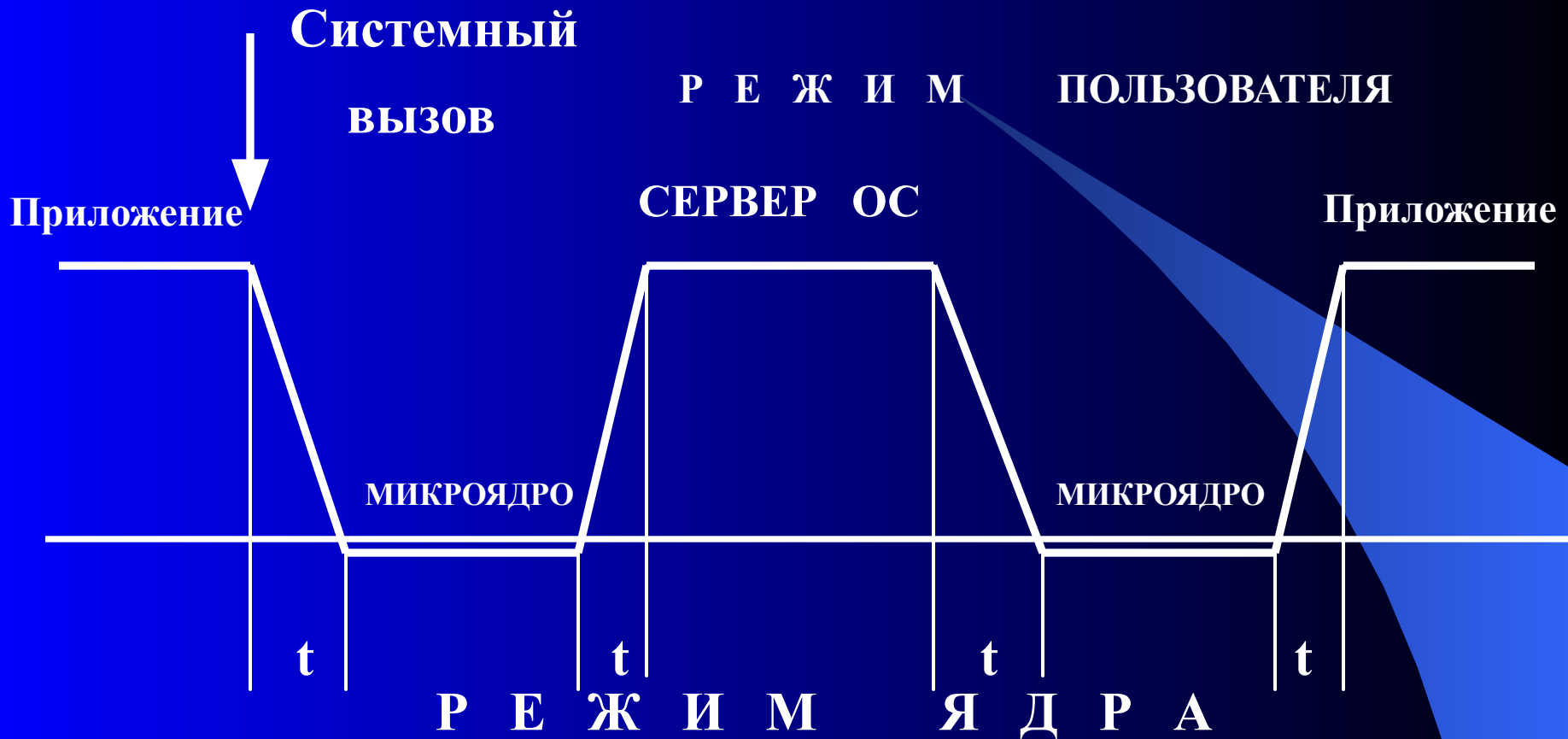
# Микроядерная архитектура ОС



# Структура ОС клиент-сервер



# Смена режимов при выполнении вызова функции микроядра



Достоинства: единообразные интерфейсы, расширяемость, гибкость, переносимость, надежность, поддержка распределенных систем, поддержка объектно-ориентированных ОС.





## Классификация ядер операционных систем

1. **Наноядро (НЯ)** – крайне упрощённое и минимальное ядро, выполняет лишь одну задачу – обработку аппаратных прерываний, генерируемых устройствами компьютера. После обработки посылает информацию о результатах обработки вышележащему программному обеспечению. Концепция наноядра близка к концепции HAL. НЯ используются для виртуализации аппаратного обеспечения реальных компьютеров или для реализации механизма гипервизора.

2. **Микроядро (МЯ)** предоставляет только элементарные функции управления процессами и минимальный набор абстракций для работы с оборудованием. Большая часть работы осуществляется с помощью специальных пользовательских процессов, называемых сервисами. В микроядерной операционной системе можно, не прерывая ее работы, загружать и выгружать новые драйверы, файловые системы и т. д. Микроядерными являются ядра ОС Minix и GNU Hurd и ядро систем семейства BSD.

3. **Экзоядро (ЭЯ)** – предоставляет лишь набор сервисов для взаимодействия между приложениями, а также необходимый минимум функций, связанных с защитой: выделение и высвобождение ресурсов, контроль прав доступа, и т. д. ЭЯ не занимается предоставлением абстракций для физических ресурсов – эти функции выносятся в библиотеку пользовательского уровня (так называемую libOS). В отличие от микроядра ОС, базирующиеся на ЭЯ, обеспечивают большую эффективность за счет отсутствия необходимости в переключении между процессами при каждом обращении к оборудованию.



**4. Монолитное ядро (МЯ)** предоставляет широкий набор абстракций оборудования. Все части ядра работают в одном адресном пространстве. МЯ требуют перекомпиляции при изменении состава оборудования. Компоненты операционной системы являются не самостоятельными модулями, а составными частями одной программы. МЯ более производительны, чем микроядро, поскольку работает как один большой процесс. МЯ являются большинством Unix-систем и Linux. Монолитность ядер усложняет отладку, понимание кода ядра, добавление новых функций и возможностей, удаление ненужного, унаследованного от предыдущих версий, кода. «Разбухание» кода монолитных ядер также повышает требования к объёму оперативной памяти.

**5. Модульное ядро (Мод. Я)** – современная, усовершенствованная модификация архитектуры МЯ. В отличие от «классических» МЯ, модульные ядра не требуют полной перекомпиляции ядра при изменении состава аппаратного обеспечения компьютера. Вместо этого они предоставляют тот или иной механизм подгрузки модулей, поддерживающих то или иное аппаратное обеспечение (например, драйверов). Подгрузка модулей может быть как динамической, так и статической (при перезагрузке ОС после переконфигурирования системы). Мод. Я удобнее для разработки, чем традиционные монолитные ядра. Они предоставляют программный интерфейс (API) для связывания модулей с ядром, для обеспечения динамической подгрузки и выгрузки модулей. Не все части ядра могут быть сделаны модулями. Некоторые части ядра всегда обязаны присутствовать в оперативной памяти и должны быть жёстко «вшиты» в ядро.



**6. 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. Так устроены 4.4BSD и MkLinux, основанные на микроядре Mach. Микроядро обеспечивает управление виртуальной памятью и работу низкоуровневых драйверов. Все остальные функции, в том числе взаимодействие с прикладными программами, осуществляется монолитным ядром. Данный подход сформировался в результате попыток использовать преимущества микроядерной архитектуры, сохраняя по возможности хорошо отлаженный код монолитного ядра.**

**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**



# Средства аппаратной поддержки ОС

- 1. Средства поддержки привилегированного режима: системные регистры процессора, слово состояния процессора, привилегированные команды, привилегированные режимы.**
- 2. Средства трансляции адресов: буферы быстрой трансляции виртуальных адресов, регистры процессора, средства поддержки сегментно-страничных таблиц.**
- 3. Средства переключения процессов: регистры общего назначения, системные регистры и указатели, флаги операций.**
- 4. Система прерываний: регистры и флаги прерываний, регистры масок, контроллеры прерываний.**
- 5. Системный таймер и системные часы.**
- 6. Средства защиты памяти: граничные регистры, ключи.**



# 1.5. Классификация операционных систем

1. Назначение (универсальные, специализированные – управление производством, обучение)
2. Способ загрузки (загружаемые, постоянно находящиеся в памяти)
3. Особенности алгоритмов управления ресурсами
  - 3.1. Многозадачность: однозадачные (MS DOS), невытесняющая многозадачность (Windows 3.x, NewWare), вытесняющая многозадачность (Windows NT, OS/2, Unix)
  - 3.2. Многопользовательский режим: отсутствие (MS DOS, Windows 3.x), имеется (Windows NT, OS/2, Unix)
  - 3.3. Многопроцессорная обработка: отсутствие, асимметричные ОС, симметричные ОС
4. По базовой технологии (Юникс-подобные или подобные Windows)
5. По типу лицензии (проприетарная или открытая)
6. По состоянию развития (устаревшая DOS, NextStep или современные GNU/Linux и Windows)



## **7. Область использования и форма эксплуатации**

**пакетная обработка (OS/360)**

**разделение времени**

**реальное время (VxWorks, QNX)**

## **8. Аппаратная платформа**

**8.1. ОС для смарт-карт (с интерпретатором виртуальной Java-машины)**

**8.2. Встроенные ОС (Palm OS, Windows CE – Consumer Electronics)**

**8.3. ОС для ПК (Windows 9.x, Windows 2000, Linux, Mac OS X)**

**8.4. ОС мини-ЭВМ (RT-11 и RSX-11M для PDP-11, UNIX для PDP-7)**

**8.5. ОС мэйнфреймов (OS/390 – пакетная обработка, разделение времени, обработка транзакций)**

**8.6. Серверные операционные системы для ЛВС, Интранет и Интернет (UNIX, AIX, Windows 2000/2002, Linux)**

**8.7. Кластерные операционные системы (Windows 2000 Cluster Server, Sun Cluster (Solaris))**



# 1.6. Эффективность и требования, предъявляемые к операционным системам

1. Эффективность – степень соответствия своему назначению, техническое совершенство и экономическая целесообразность
2. Надежность и отказоустойчивость
3. Безопасность (защищенность)
4. Предсказуемость
5. Расширяемость
6. Переносимость
7. Совместимость
8. Удобство
9. Масштабируемость



## 1.7. Множественные прикладные среды. Совместимость

**Совместимость – возможность операционной системы выполнять приложения , разработанные для других операционных систем.**

### **Виды совместимости:**

- 1. На двоичном уровне (уровень исполняемой программы).**
- 2. На уровне исходных текстов (уровень исходного модуля).**

### **Вид совместимости определяется:**

- 1. Архитектурой центрального процессора.**
- 2. Интерфейсом прикладного программирования (API).**
- 3. Внутренней структурой исполняемого файла.**
- 4. Наличием соответствующих компиляторов и библиотек.**

### **Способы достижения совместимости:**

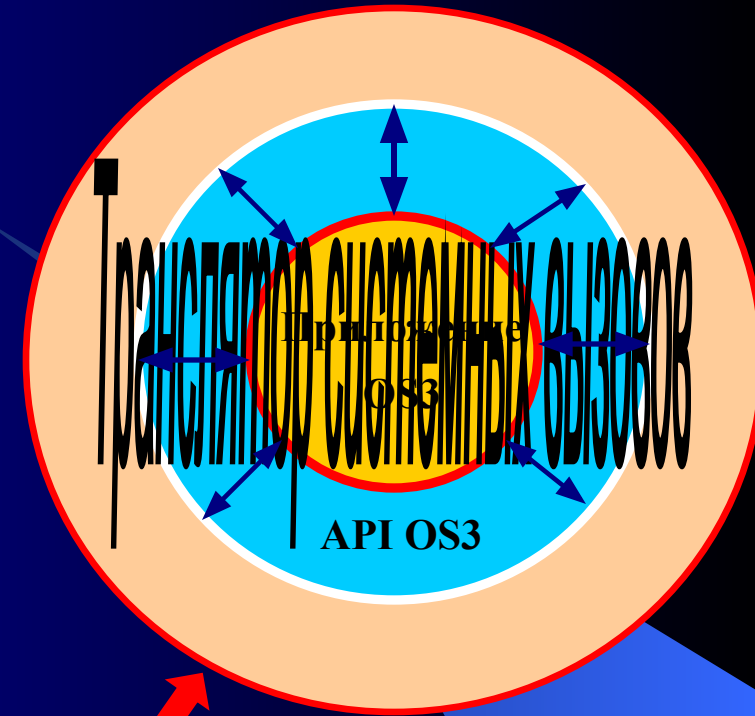
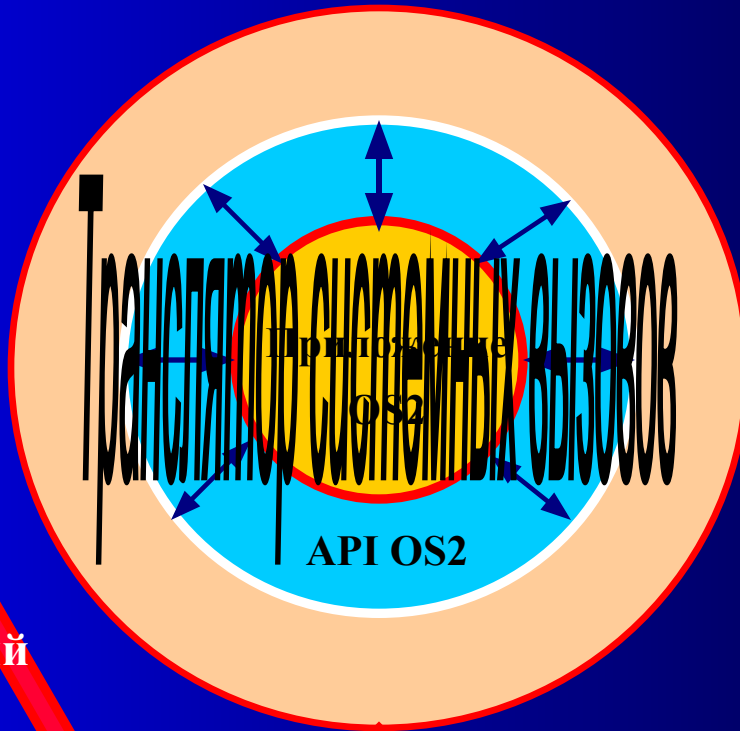
- 1. Эмуляция двоичного кода.**
- 2. Трансляция библиотек.**
- 3. Создание множественных прикладных сред различной архитектуры.**





## Прикладная среда OS2

## Прикладная среда OS3



Пользовательский режим

Привилегированный режим



Приложение OS1

Приложение OS2

Приложение OS3

Пользовательский режим

Привилегированный режим

API OS1

API OS2

API OS3

Менеджеры ресурсов

Базовые механизмы

Машинно-независимые задачи



# Приложения

# Серверы ОС



# Подсистемы среды Windows 2000



# 1.8. Способы работы с программами разных операционных систем на одном компьютере

## Способ №1: многовариантная загрузка

Это самый распространенный способ (до недавнего времени) решения проблемы, который использует подавляющее большинство пользователей. Жесткий диск компьютера разбивается на несколько разделов и на каждый из них устанавливается своя операционная система и программы для нее. Кроме того, настраивается менеджер многовариантной загрузки, позволяющий выбрать операционную систему при загрузке компьютера. При таком подходе невозможно одновременно работать с приложениями разных операционных систем и для смены операционной системы необходимо перезагрузить компьютер. Зато операционные системы и их приложения исполняются без потерь в скорости и надежности. Если операционные системы совместимы по типу файловой системы, то возможно создание общего раздела для обмена файлами между ними.

### **Итоговые оценки по десятибалльной шкале:**

**Одновременная работа: 0**

**Многоплатформенность: 5**

**Производительность: 10**

**Совместимость: 10**



## Способ №2: эмуляция API операционной системы

Обычно приложения работают в изолированном адресном пространстве и взаимодействуют с оборудованием при помощи API, предоставляемым операционной системой. Если две ОС совместимы по своим API (например, Windows 98 и Windows 2000), то приложения, разработанные для одной из них, будут работать и на другой. Если две операционные системы несовместимы по своим API (например, Windows 2000 и Linux), то существует способ перехватить обращения приложений к API и симитировать поведение одной операционной системы средствами другой операционной системы.

При таком подходе можно поставить одну операционную систему и работать одновременно как с ее приложениями, так и с приложениями другой операционной системы. Поскольку весь код приложения исполняется без эмуляции и лишь вызовы API эмулируются, потеря в производительности незначительная. Но из-за того что многие приложения используют недокументированные функции API или обращаются к операционной системе в обход API, даже хорошие эмуляторы API имеют проблемы совместимости.

**Итоговые оценки по десятибалльной шкале:**

**Одновременная работа: 9**

**Многоплатформенность: 0**

**Производительность: 9**

**Совместимость: 3**



## Способ №3: полная или частичная эмуляция

Проекты, выполненные по технологии полной эмуляции работают как интерпретаторы. Они последовательно выбирают код гостевой операционной системы и эмулируют поведение каждой отдельно взятой инструкции. Поскольку при этом полностью эмулируется поведение как процессора, так и всех внешних устройств виртуального Intel x86 компьютера, то существует возможность запускать эмулятор на компьютерах с совершенно другой архитектурой. Скорость работы гостевых приложений может упасть в 100-1000 раз, что означает практическую невозможность нормальной работы с гостевой операционной системой внутри эмулятора. Поэтому полная эмуляция используется редко (низкоуровневых отладчиков для исследования и трассировки операционных систем).

Виртуальная машина эмулирует реальное аппаратное обеспечение, что позволяет использовать в качестве гостевых обычные, немодифицированные операционные системы, а команды, требующие себе особых привилегий, обрабатываются средствами VMM.

### Итоговые оценки по десятибалльной шкале:

**Одновременная работа: 10**

**Многоплатформенность: 9**

**Производительность: 2**

**Совместимость: 9**



## Способ №4: виртуальная машина эмулирует реальное аппаратное обеспечение (квазиэмуляция)

Существует большое количество инструкций, которые будут нормально исполняться в режиме нескольких операционных систем, и некоторое небольшое количество инструкций, которые должны эмулироваться. Технология квазиэмуляции заключается в том, чтобы обнаружить и симитировать поведение второго множества инструкций и исполнять инструкции первого множества без эмуляции.

Виртуальная машина эмулирует реальное аппаратное обеспечение, что позволяет использовать в качестве гостевых обычные, немодифицированные операционные системы, а команды, требующие себе особых привилегий, обрабатываются средствами VMM. В этом случае обеспечиваются основные функции процессора и остальных главных компонентов компьютера. Идея естественной виртуализации: поверх аппаратного уровня (физический сервер) располагается уровень монитора виртуальных машин VMM (гипервизор). Гипервизор полностью эмулирует компьютер, и способен поддерживать выполнение более чем одной операционной системы. На VMM выполняются так называемые гостевые операционные системы (guest OS) виртуальных машин, непосредственно поддерживающие работу приложений.

**Итоговые оценки по десятибалльной шкале:**

**Одновременная работа: 10**

**Многоплатформенность: 5**

**Производительность: 8**

**Совместимость: 8**





**Virtually** - фактически как , практически как, в сущности, поистине.

## Примеры:

**Telnet сеанс** – **ФАКТИЧЕСКИ КАК** работать за консолью удаленного компьютера.

**Сетевой диск** – **ПРАКТИЧЕСКИ КАК** обычный логический диск.

**Виртуальная память** – **ПОИСТИНЕ** как большая оперативная память.

**Виртуализация** - это отделение логического ресурса от физического.

**Виртуализация** повышает эффективность использования физических ресурсов, обеспечивает высокую гибкость их использования и упрощает управление изменениями



## Основные области применения:

- Тестирование программного обеспечения и средств разработки ( тестирование создаваемых приложений, тестирование конфигураций и настроек готового программного обеспечения, а также действий администраторов серверов и сети с целью проверки работоспособности той или иной конфигурации серверного ПО перед началом ввода его в реальную эксплуатацию).
- Хостинг унаследованных приложений. Зачастую наиболее удачные бизнес-приложения эксплуатируются десятилетиями, поэтому вполне может случиться так, что платформа, для которой они написаны, в компании уже практически не применяется из-за отсутствия нормальной технической поддержки со стороны производителей оборудования.
- Консолидация загрузки серверов. Идея консолидации загрузки серверов заключается в создании виртуальных машин с разными операционными системами и программным обеспечением, реализующими выполнение указанных задач, и в размещении одного и того же набора этих виртуальных машин на нескольких физических серверах. Благодаря этому число самих серверов можно уменьшить, да и выход из строя одного из серверов не будет столь критичен для компании, поскольку его нагрузку может взять на себя виртуальная машина на каком-либо другом сервере.
- Моделирование распределенных серверных приложений на одном физическом сервере. Данный способ применения серверных виртуальных машин предназначен для разработчиков, специалистов по тестированию и специалистов по внедрению приложений масштаба предприятия. С его помощью можно создавать распределенные приложения, тестировать их, а также моделировать реальные условия внедрения, используя для этой цели один-единственный компьютер, что позволяет сократить расходы на приобретение аппаратного обеспечения для разработки приложений.



# Преимущества

## Экономическая и практическая выгода

### Снижение затрат на оборудование

Консолидация нескольких приложений и операционных систем на одном сервере.

### Более эффективное использование ресурсов.

Около 70% времени серверы простаивают, изнашиваясь и потребляя электроэнергию.

### Уменьшение административных издержек.

Виртуальные среды, благодаря абстрагированию от физического оборудования, являются более гибкими в управлении. Виртуальные сервера могут быть с лёгкостью перенесены с одного физического оборудования на другое..



# Отказоустойчивость

## Презентации

Виртуализация позволяет построить отказоустойчивую инфраструктуру

## Реальность

Отказоустойчивость – это глобальное понятие, туда входит много компонентов, только одна виртуализация не поможет.

Большинство сервисов уже отказоустойчивы (Exchange, SQL, Web)

Не все знают, где узкие места в их системе, чтобы сделать систему более надежной.



# Безопасность

## Презентации

Виртуализация усиливает безопасность

## Реальность

Дополнительное ПО = дополнительная лазейка

Виртуальную машину проще украсть, это же файл.

Через гипервизор проходит весь трафик виртуальных машин, есть что атаковать.



# Сокращение расходов

## Презентации

Виртуализация позволяет сократить расходы на ИТ-инфраструктуру

## Реальность

Бюджет ДОЛЖЕН БЫТЬ ПОТРАЧЕН ☺

ТСО= CapEx + OpEx – а многие ли знают свои текущие расходы?

Не все производители ПО имеют лояльные политики лицензирования при виртуализации !



# Персонал

## Презентации

Виртуализация позволяет сократить затраты на персонал

## Реальность

Внедрение новой системы – **дополнительная** нагрузка на персонал

Больше экземпляров ОС (рост в связи с виртуализацией)

В наших условиях, обычно не хватает ИТ персонала – на ком сокращать?



# Планирование

## Презентации

Виртуализация позволяет упростить планирование инфраструктуры

## Реальность

*Было:* рост количества серверов – *стало:* рост количества VM

Гипервизор – это больше железо, чем софт.

**ОБЯЗАТЕЛЬНО** очень аккуратно планировать инфраструктуру.

На что обратить внимание – сети, СХД.





# Архитектура Hyper-V R2

## Процессор

х64 сервера, с аппаратной поддержкой виртуализации

AMD AMD-V или Intel VT

Требуется включение Data Execution Prevention (DEP)

AMD (NX no execute bit)

Intel (XD execute disable)

Hyper-V не поддерживает процессоры Itanium (IA-64)

**Data Execution Prevention (DEP)** (англ. Предотвращение выполнения данных) функция безопасности, встроенная в семейство операционных систем операционных систем Windows операционных систем Windows, которая не позволяет приложению операционных систем Windows, которая не позволяет приложению исполнять код из области памяти, помеченной как «только для данных». Она позволит предотвратить некоторые атаки, которые, например, сохраняют код в такой области с помощью переполнения буфера. DEP работает в двух режимах: аппаратном, для процессоров, которые могут помечать страницы как «не для исполнения кода», и программном, для остальных процессоров.





**В качестве примера современного классического решения Type 1 hypervisor можно назвать VMware ESX Server; по существу это еще одна операционная система, действующая непосредственно на аппаратной платформе x86 в чистом виде. Гостевыми операционными системами, работающими на ESX Server, могут быть Linux, Windows, FreeBSD, NetWare и Solaris. Как самостоятельная операционная система, VMware ESX Server интерпретирует аппаратную платформу в качестве пула логических ресурсов и динамически перераспределяет его между гостевыми операционными системами**



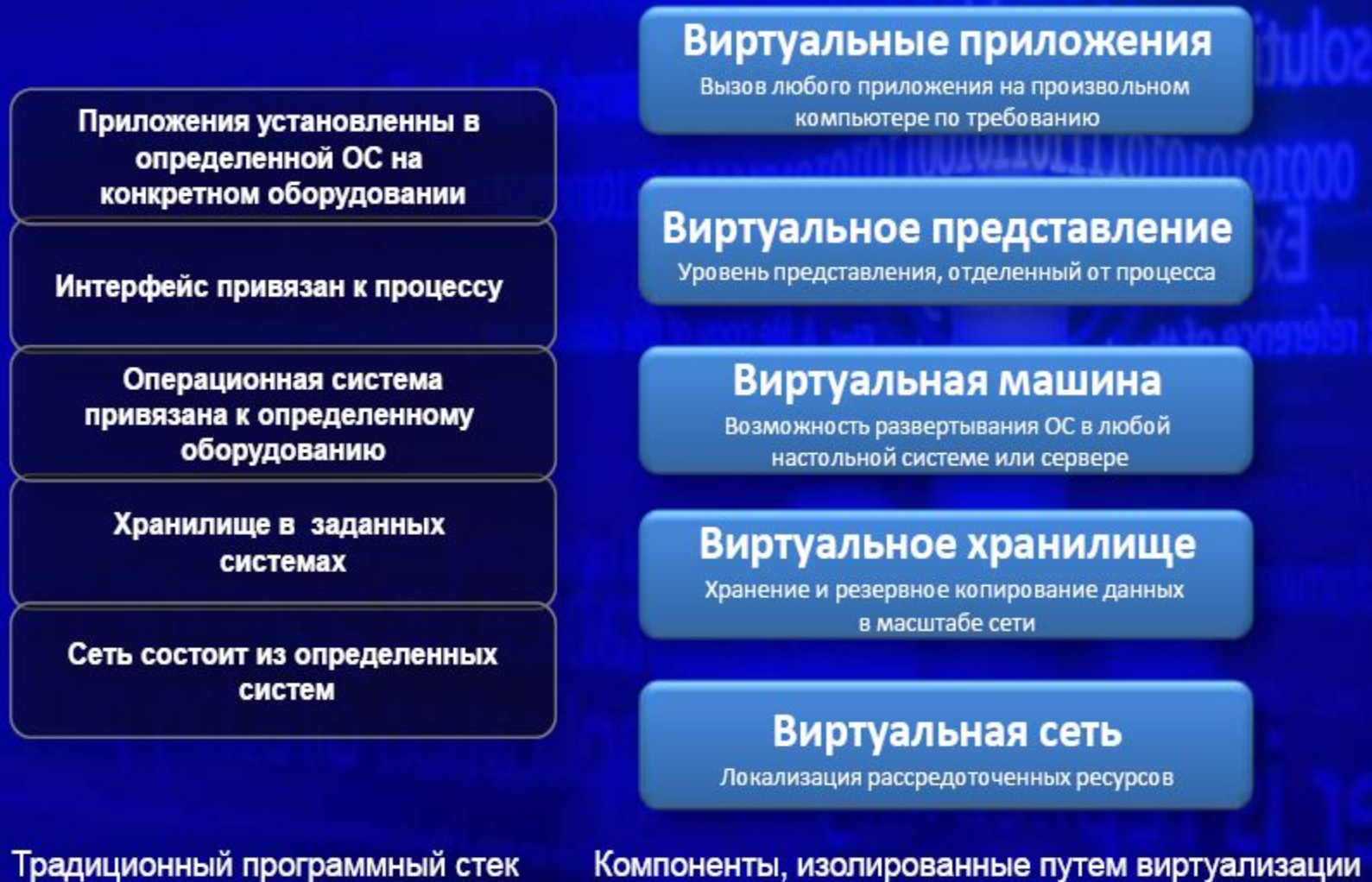


Решение Type 2 hypervisor отличается тем, что гипервизор работает поверх операционной среды, так называемого «хоста». Типичными представителями этого направления виртуализации являются VMware Server и Microsoft Virtual Server. К примеру, Microsoft Virtual Server 2005 устанавливается как приложение на операционную систему Windows 2003 Server, выполняющую функцию «хоста». Таким образом создается виртуализационный уровень, обеспечивающий доступ к физическим ресурсам. Virtual Server 2005 доступен в двух версиях: Standard Edition и Enterprise Edition. Хостом для сервера VMware GSX Server могут быть операционные системы Windows 2000, Windows 2003 или Linux.



# 1.9. Виртуализация от Microsoft

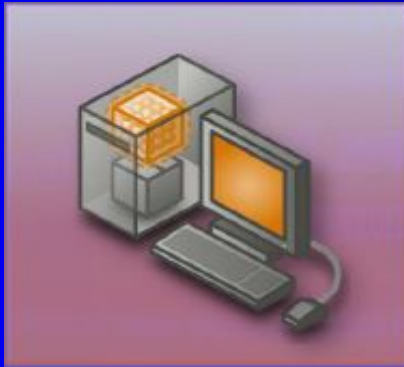
Виртуализация – это отделение одного вычислительного ресурса от других:



# Виртуализация приложений

- Абстрагирование приложения от базовой операционной системы
- Исполнение приложений в виртуальной среде
- Сокращение риска нежелательного взаимодействия между приложениями и операционной системой
- Устранение конфликтов между приложениями
- Существенные преимущества на всех этапах жизненного цикла приложений
  - Управление
  - Внедрение
  - Запуск

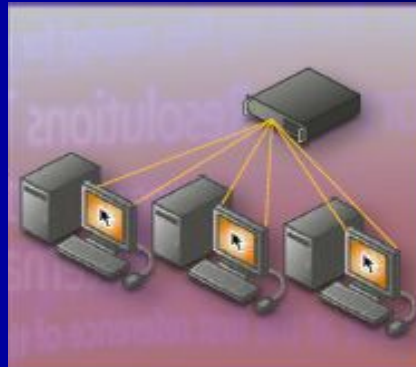




Виртуализация настольных систем

*Создание в стандартной настольной системе дополнительной, изолированной среды ОС*

- Поддержка устаревших приложений в современных ОС
- Минимизация конфликтов между приложениями и ОС
- Ускорение миграции на новые ОС



Виртуализация представления

*Централизованная обработка и хранение данных; локальное представление пользовательского интерфейса*

- Минимизация конфликтов между приложениями и ОС
- Упрощение процедур соответствия нормативным требованиям и обеспечения конфиденциальности данных
- Сокращение затрат на администрирование настольных систем



Виртуализация серверов

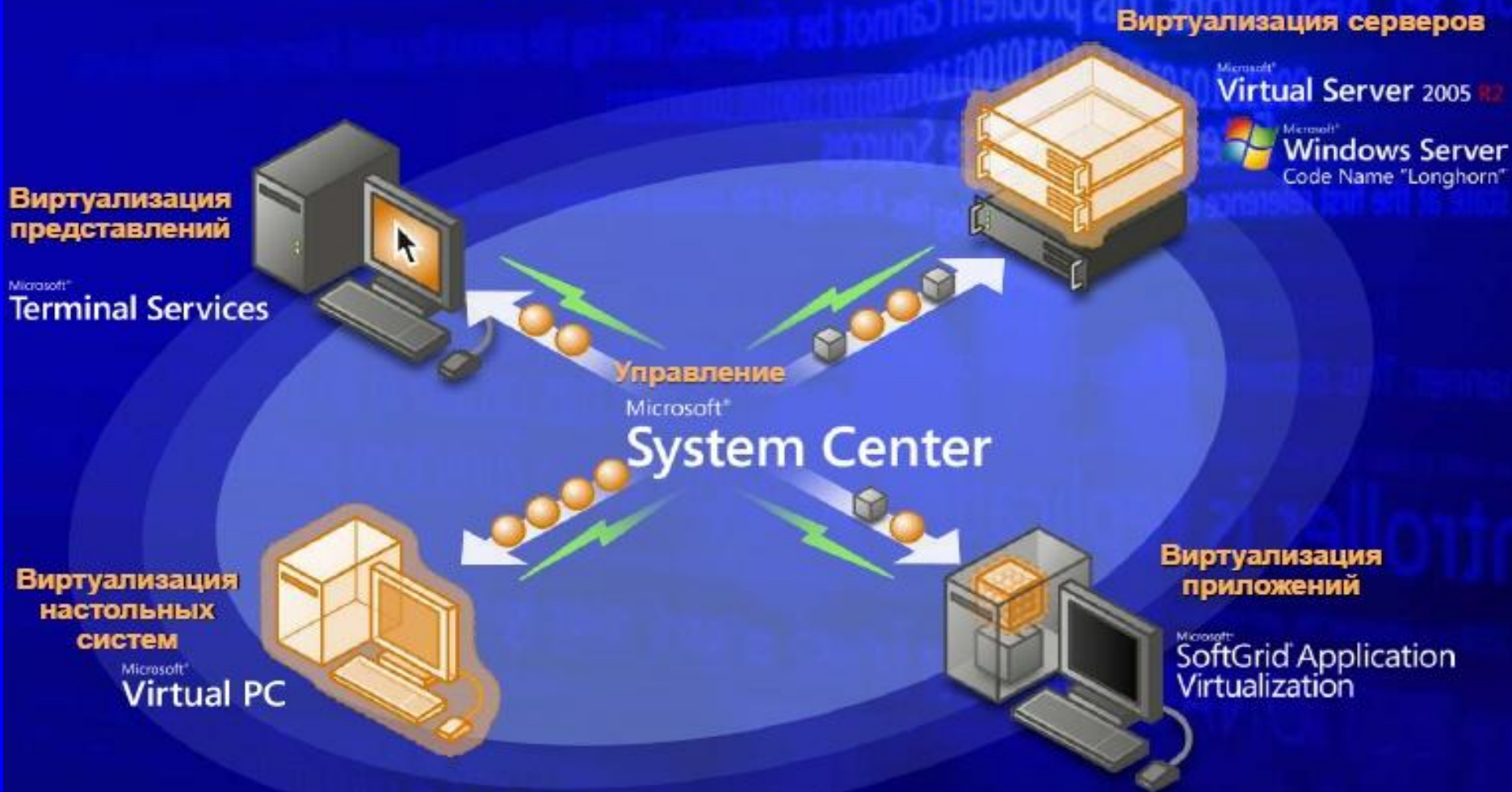
*Консолидация нагрузки для более эффективного использования ресурсов*

- Сокращение эксплуатационных затрат (на оборудование, энергоресурсы, площади)
- Увеличение времени работоспособного состояния и доступности
- Надежное аварийное восстановление
- Сокращение простоя из-за обслуживания
- Упрощение контроля потребления и масштабирования ресурсов



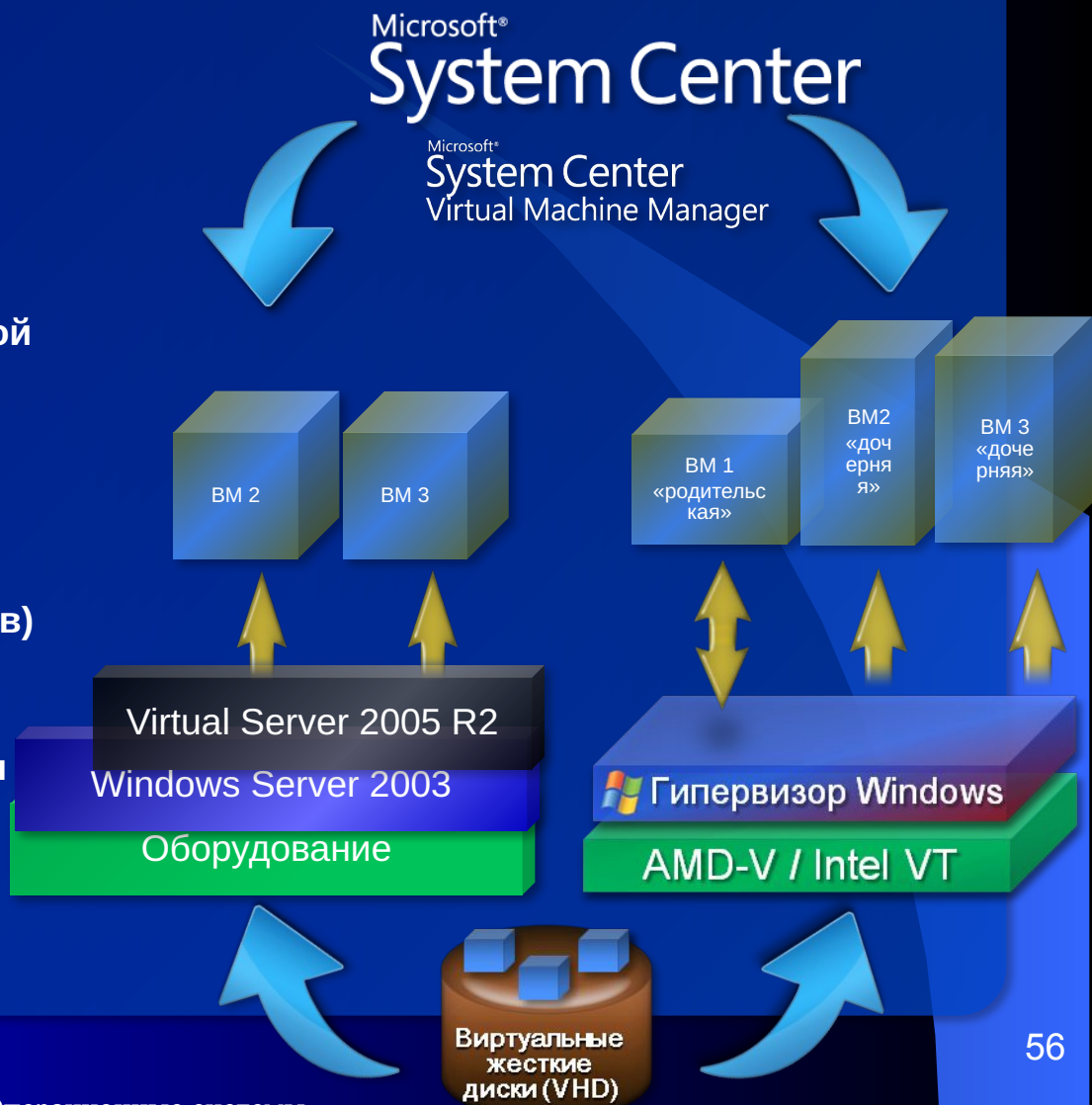
# Виртуализация от Microsoft

Комплексный набор средств виртуализации, начиная от центра обработки данных и заканчивая настольными системами – как виртуальными, так и физическими – управляется с одной платформы



# Виртуализация Windows Server

- Масштабируемость и производительность
  - Поддержка 64-разрядных серверов и гостевых ОС
  - Поддержка SMP для гостевых систем
- Надежность и защита
  - Минимальный объем доверенной базы кода
  - Решение Windows
- Большая гибкость и управляемость
  - Динамическое добавление виртуальных ресурсов (памяти, процессоров, сетевых адаптеров)
  - Динамический перенос ОС и приложений
  - Новый интерфейс пользователя и интеграция с SCVMM





# Архитектура. Virtual Machine Monitor (VMM)

- ЦП вынужден переключаться между процессами базовой ОС и гостевой ОС
  - VMM переключает контекст между этими процессами
  - Компьютер работает в контексте хоста либо VMM
- На одном ЦП может работать только одна ОС
- Сжатие кода нулевого кольца (ring 0) гостевой ОС



# Виртуализация ЦП. Проблемы

При прямом доступе гостевая ОС будет работать быстро! (99%)

Когда требуется выполнить привилегированную операцию, срабатывает **ловушка**, и VMM обрабатывает эту операцию в режиме ядра.

**Проблема:** полная виртуализация платформы x86 таким способом невозможна, так как некоторые инструкции ЦП для режима ядра, выполняющие чтение, разрешены не только в нулевом кольце

## Возможные решения:

- a) Перекомпилировать ОС и приложения, избегая этих 20 инструкций, т.е. исключить 20 «проблемных» инструкций.
- b) Воспользоваться исполнением с **трансляцией двоичного кода** ( модификация кода «на лету» во время выполнения на хосте).
- c) Установить в гостевой системе **VM Additions**, что позволит модифицировать код в памяти VM.
- d) Использовать **аппаратную поддержку виртуализации** (перехват инструкций в особом “кольце -1”).



# Решения

## 1. Преобразование двоичного кода

Трансляция инструкций гостевой операционной системы в инструкции базовой ОС. Всегда возможна, но работает очень медленно.

## 2. VM Additions

Модифицирует dll-код в памяти VM (невозможно в 64-разрядных версиях Vista и Longhorn).

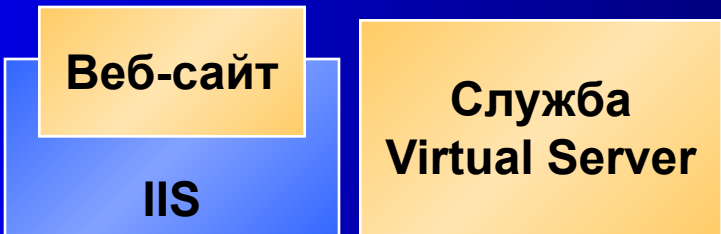
VM Additions поддерживают синхронизацию времени, «пульс», завершение работы, оптимизированный SCSI-диск, лучшие драйверы мыши и видео.

## 3. Аппаратная виртуализация

ЦП с поддержкой технологий Intel VT или AMD Virtualization. ЦП решает проблемы, отслеживая параметры каждой VM (фактически, это «кольцо 1»).



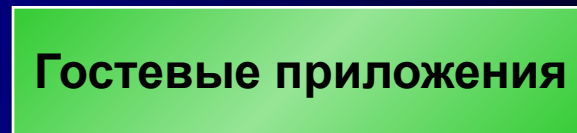
## Базовая система



Кольцо 3

Кольцо 1

## Гостевая система (VM)



Кольцо 3

Кольцо 1

VM Additions

Windows в VM

Виртуальное  
оборудование

Кольцо 0

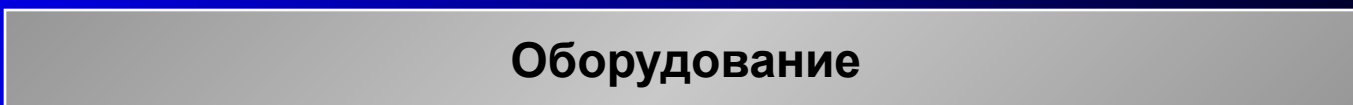


Win2003 или WinXP

Ядро

VMM.sys

Оборудование



# Версии VM Additions

| Сборка | Выпуск                         | Примечание  |
|--------|--------------------------------|---|
| 10.21  | В составе Virtual PC 5.2       | (дано название – Virtual PC Additions)  |
| 13.40  | В составе Virtual PC 2004      |   |
| 13.187 | (отдельная загрузка)           | Поддерживает Win XP SP2   |
| 13.206 | В составе VS2005               |   |
| 13.306 | В составе Virtual PC 2004 SP1  |   |
| 13.518 | В составе VS2005 SP1 beta      |   |
| 13.531 | (отдельная загрузка)           | Поддерживает Win2003 SP1  |
| 13.552 | В составе VS2005 R2            | Поддерживает Win2003 R2 и Vista (-build 5270)   |
| 13.705 | В составе VS2005 R2 SP1 beta1  |   |
| 13.706 | (отдельная загрузка)           | Поддерживает Vista B2 (-build 5384) и Longhorn  |
| 13.709 | (отдельная загрузка)           | Поддерживает Vista RC1  |
| 13.715 | В составе VS2005 R2 SP1 beta2  | Поддерживает Vista RTM  |
| 13.724 | В составе Virtual PC 2007 beta |   |
| 13.803 | В составе Virtual PC 2007      | Загрузка – по адресу <a href="http://www.microsoft.com/virtualpc">www.microsoft.com/virtualpc</a> |



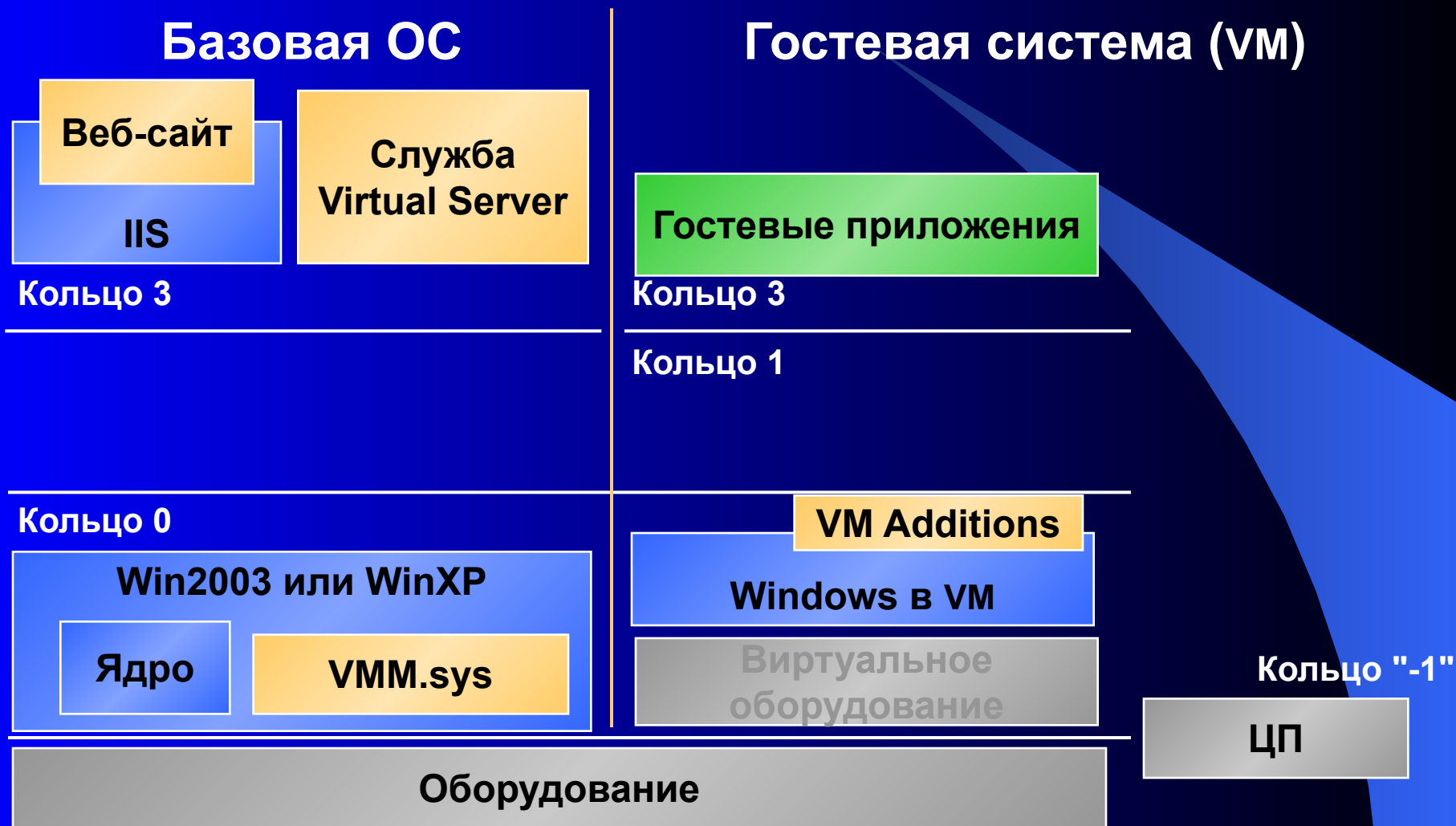
# Linux VM Additions

- Добавляется поддержка:
  - Синхронизации времени
  - «Пульса»
  - Завершения работы
  - SCSI-дисков
  - Драйвер мыши и видео
  - Поддержки **прямого исполнения кода** нет!
- Дистрибутивы (9x):
  - Red Hat 7.3/9.0, Enterprise 2.1/3/4
  - SuSE Linux 9.2/9.3/10.0, Enterprise Server 9

В выпуске VS 2005 R2 SP1 поддерживаются гостевые ОС : Red Hat Enterprise Linux 2.1 (update 7), Red Hat Enterprise Linux 3.0 (update 8), Red Hat Enterprise Linux 4.0 (update 4), Red Hat Enterprise Linux 5.0, SuSE Linux Enterprise Server 9.0, SuSE Linux Enterprise Server 10.0, Red Hat Linux 9.0, SuSE Linux 9.3, SuSE Linux 10.0, SuSE Linux 10.1, SuSE Linux 10.2.



# Архитектура виртуализации с аппаратной поддержкой

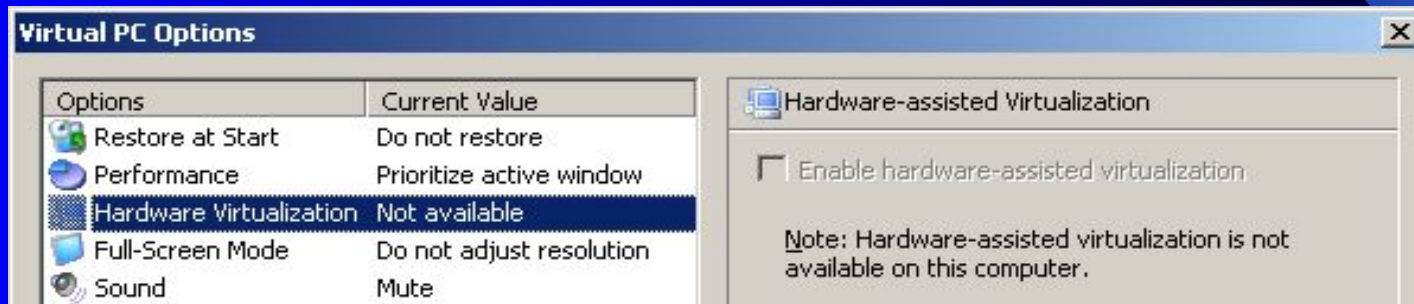


# Виртуализация с аппаратной поддержкой (Intel VT или AMD Virtualization)

Поддерживается в:

- Virtual PC 2007
- Virtual Server 2005 R2 SP1
- Windows Virtualization (обязательно)

Необходимо включить в BIOS и в параметрах Virtual PC 2007



Скорость работы гостевых ОС windows не повышается

- Последние версии **VM Additions** уже поддерживают прямой доступ к ЦП
- Установка Windows выполняется в 2-3 раза быстрее
- Гостевые ОС типа Linux и Netware работают быстрее





# Спецификации Virtual Server 2005 R2

## Базовая система:

VS2005 Standard Edition: до 4 ЦП (1- или 2-ядерные),  
VS2005 Enterprise Edition: до 32 ЦП (1- или 2-ядерные),  
ОЗУ: до 64 Гб

## Гостевая система:

ЦП: до 1, ОЗУ: до 3,6 Гб, Сетевые адаптеры: до 4, (неограниченная пропускная способность). USB: нет, поддерживаются USB-клавиатура и USB-мышь, можно также подключить USB-устройство для чтения смарт-карт.

## Дополнительные возможности Server 2005 R2 SP1:

Поддержка Intel VT и AMD Virtualization,  
Поддержка 64-х разрядных базовых систем: Win2003 и WinXP.  
Поддержка теневого копирования томов (Volume Shadow Copy, VSS),  
Интеграция с Active Directory средствами Service Connection Points,  
Поддержка Vista как гостевой ОС,  
Утилита для монтирования VHD,  
Емкость по умолчанию VHD - 127 Гб (ранее – 16 Гб),  
Исправление Virtual SCSI для гостевых ОС Linux 2.6.x,  
Кластеризация VM,  
Передача VM при ее сбое в пределах того же хоста,  
Общий SCSI- (iSCSI-) диск для гостевых систем.

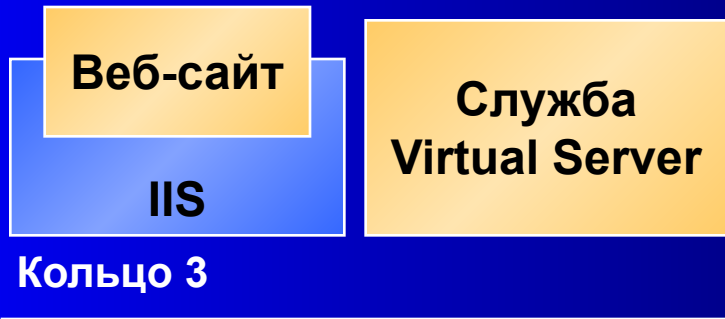


# Virtual PC / Virtual Server 2005 R2

Поставщик

- Windows
- Virtual Server
- Другие компоненты

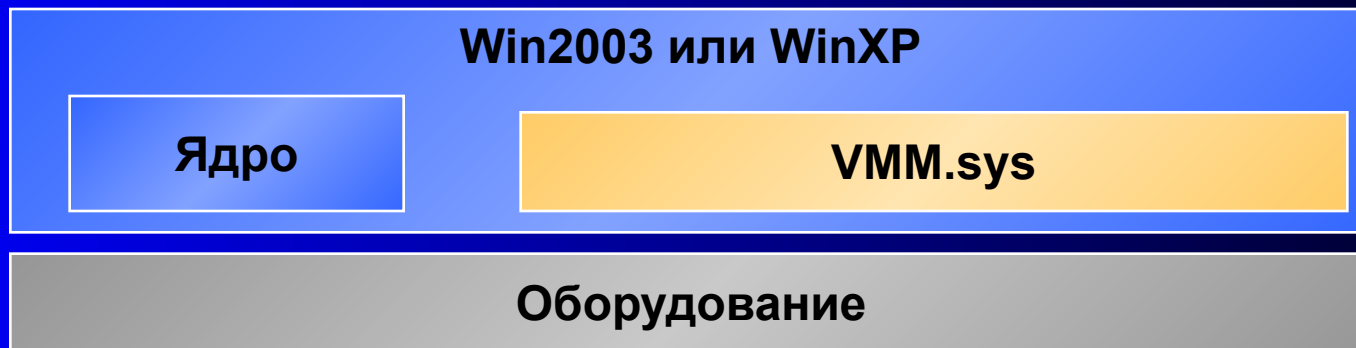
## Базовая система



## Гостевая система (VM)



## Кольцо 0



# Windows Virtualization

## Поддержка виртуализации для Windows Server

**Windows Hypervisor (Гипервизор), кодовое имя - "Viridian":**  
**«Тонкий» (~160 Кб) программный уровень, «внутренняя базовая ОС»,**  
Родительский раздел – управляет дочерними разделами,  
Дочерний раздел включает любое число ОС, управляемых родительским разделом.

### Стек виртуализации:

Работает в корневом (= родительском) разделе,  
Обеспечивает виртуализацию устройств,  
WMI-интерфейс для управления

## Провайдеры служб виртуализации (Virtualization Service Providers, VSPs)

Архитектура совместного использования оборудования,  
гостевой ОС устанавливаются драйверы "viridian«.

В

**Windows Virtualization Server** требует x64-совместимого оборудования, ЦП с поддержкой Intel VT или AMD-V

**Поддерживает:** 32- и 64-разрядные гостевые ОС; до 8 ЦП на VM; горячее добавление» ЦП, ОЗУ, сетевых адаптеров, дисков; > 32 Гб ОЗУ на VM; возможность переноса VM без отключения; традиционную модель драйверов; использование существующих драйверов Windows; прежний же набор эмулируемого оборудования; Server Core в качестве родительской ОС



# Windows Virtualization

## Схемы VMM

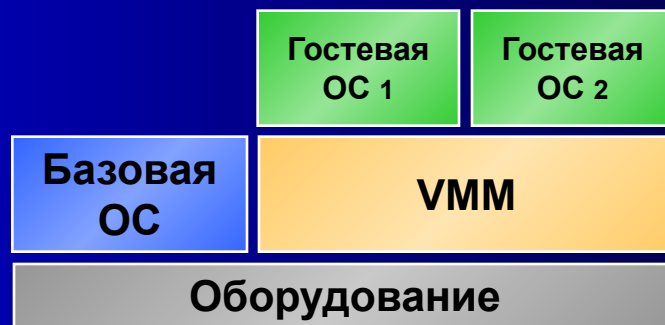
VMM типа 2



Примеры:

- JVM
- .NET CLR

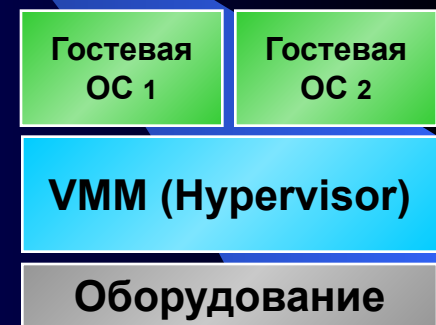
Гибридный VMM



Примеры:

- Virtual PC
- Virtual Server

VMM типа 1  
Hypervisor



Примеры:

- Виртуализация Windows ("Viridian")



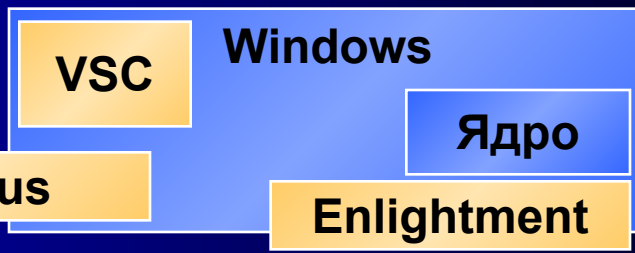
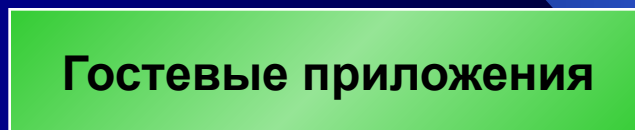
# Windows Virtualization

Поставщик

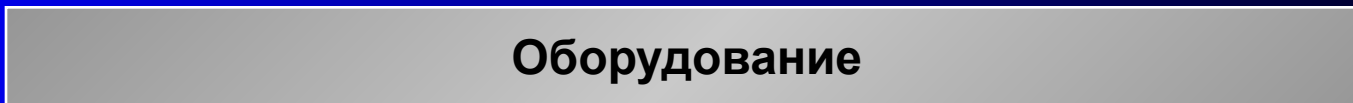
- Window s
- Win Virtualizaton
- Другие компоненты

Родительский раздел

Дочерний раздел



VMBus



Кольцо 3

Кольцо 0

Кольцо "-1"





- Установлены только набор исполняемых файлов и библиотеки DLL
- Не установлен графический интерфейс пользователя
- Доступно для части серверных ролей
- Можно управлять с помощью удаленных средств



# Версии продуктов

| Продукт                    | Выпуск                | Базовые системы   | Гостевые системы **   |
|----------------------------|-----------------------|---|---|
| Virtual PC 2004            | Октябрь 2003          | <ul style="list-style-type: none"> <li>• Win2000 Pro SP4</li> <li>• Win XP Pro (Tablet, SP1)</li> </ul>   | <ul style="list-style-type: none"> <li>• MS-DOS 6.22 * / OS/2</li> <li>• Win 95, 98, 98SE, ME *</li> <li>• Win NT4 SP6a (workstation) *</li> <li>• Win2000 Pro SP4</li> <li>• Win XP (Tablet, SP1)</li> </ul> |
| Virtual Server 2005        | Июль 2004             | <ul style="list-style-type: none"> <li>• Win XP Pro</li> <li>• Win2003 SBS</li> <li>• Win2003 (SE, EE, Data)</li> </ul>                         | <ul style="list-style-type: none"> <li>• Win NT4 SP6a (server) *</li> <li>• Win2000 Server</li> <li>• Win2003 (SE, EE, Web)</li> </ul>  |
| Virtual PC 2004 SP1        | Октябрь 2004          | То же, что и для Virtual PC 2004 + Win2003 SE   | То же, что и для Virtual PC 2004 + Win XP SP2   |
| Virtual Server 2005 R2     | Ноябрь 2005           | То же, что и для Virtual Server 2005 + Win XP Pro SP2 (non prod) + Win2003 (SP1, R2) + Win XP / Win2003 x64                                     | То же, что и для Virtual Server 2005 + Win XP Pro SP2 + Win2003 (SP1, R2) + Linux (9x) - Apr 2006   |
| Virtual PC 2004 Express    | Март 2006             | То же, что и для Virtual PC 2004 SP1 + Поддерживает не более одной VM + в Vista Enterprise / только для участников программы Software Assurance |   |
| Virtual PC 2007            | 19 февраля 2007       | + Поддержка ЦП с технологиями Intel VT и AMD Virtualization + Поддержка Vista (гостевые и хост-системы)   |   |
| Virtual Server 2005 R2 SP1 | Март 2007             | +Поддержка виртуализации процессоров Intel VT и AMD Virtualization +Поддержка Volume Shadow Copy Service (для резервного копирования)           |   |
| Windows Virtualization     | Longhorn + < 180 дней | Реализация Windows Hypervisor Новая модель виртуализации, требует аппаратной поддержки VT/Virtualization Кодовое имя "Viridian"                 |   |

\* Жизненный цикл этих продуктов близок к завершению

\*\* На <http://vpc.visualwin.com> находится список из > 1200 (!) ОС, совместимых с Virtual PC и Virtual Server

В статье KB 867572 см. список ОС, поддерживаемых Virtual Server 2005 R2



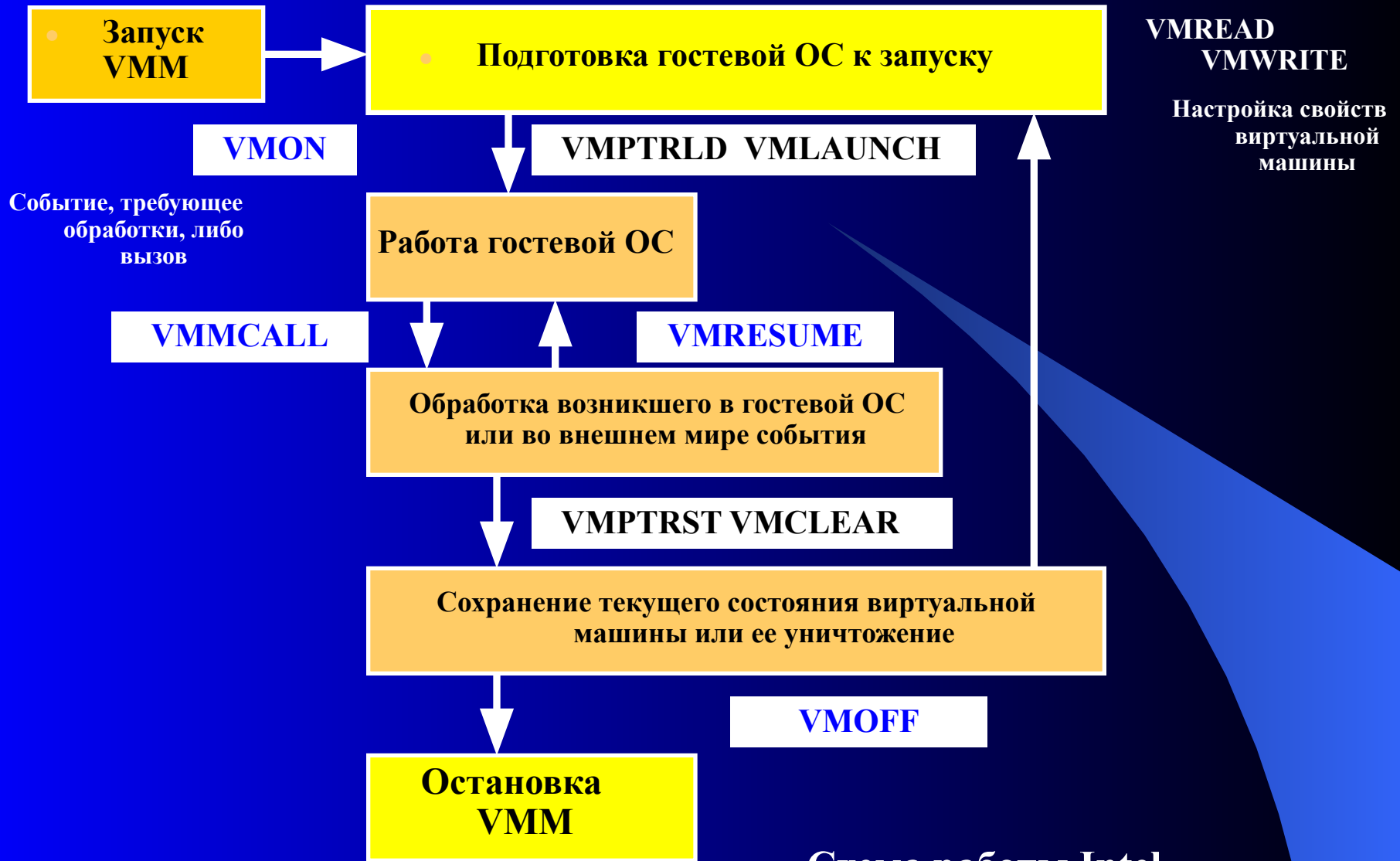


Схема работы Intel Virtualization Technology







## 1.10. Технология Virtuozzo

**SWsoft - это мировой лидер в области программного обеспечения для виртуализации серверов и автоматизации, которое помогает потребителям, бизнесменам и провайдерам услуг оптимизировать процесс использования технологии. Программное обеспечение компании поддерживает работу более 130 000 серверов и 600 000 рабочих станций по всему миру. Линейка продуктов компании SWsoft включает Virtuozzo - передовое решение для виртуализации операционных систем, Parallels - передовой продукт виртуализации рабочих станций и Plesk - ведущую панель управления серверами. Компания основана в 1999 году, офисы расположены по всей территории Северной Америки, Европы и Азии.**

**В 2006 году объемы продаж компании увеличились в 10 раз по сравнению с 2004 годом. Подразделение Parallels, входящее в SWSoft, разработало платформу для виртуального исполнения Windows ОС на платформе Mac, которая входит в десятку лучших продуктов 2006 года и является наиболее продаваемым на Amazon. В России сегодня работает свыше 750 инженеров компании.**



## Аппаратная модель виртуализации (гипервизор)

В модели гипервизора имеется базовый слой (обычно это тонкий слой ядра Linux, представленный здесь гипервизором или стандартной ОС), который загружается непосредственно на чистый сервер. Для выделения оборудования и ресурсов виртуальным машинам требуется виртуализация всего аппаратного обеспечения на сервере. В следующем слое показаны все чипы, платы и другие устройства, которые необходимо виртуализировать, чтобы их можно было предоставлять виртуальным машинам. В самой виртуальной машине содержится полная копия операционной системы и, наконец, приложение или рабочая нагрузка.



**SWsoft Virtuozzo - это запатентованное решение по виртуализации ОС. Virtuozzo позволяет создавать изолированные виртуальные среды (VE) или контейнеры на одном физическом сервере и экземпляре ОС. По сравнению с другими технологиями виртуализации Virtuozzo обеспечивает наиболее высокий уровень плотности, производительности и управляемости.**

**Интеллектуальное разбиение на разделы - разделение сервера на сотни виртуальных сред, функционирующих как самостоятельные серверы.**

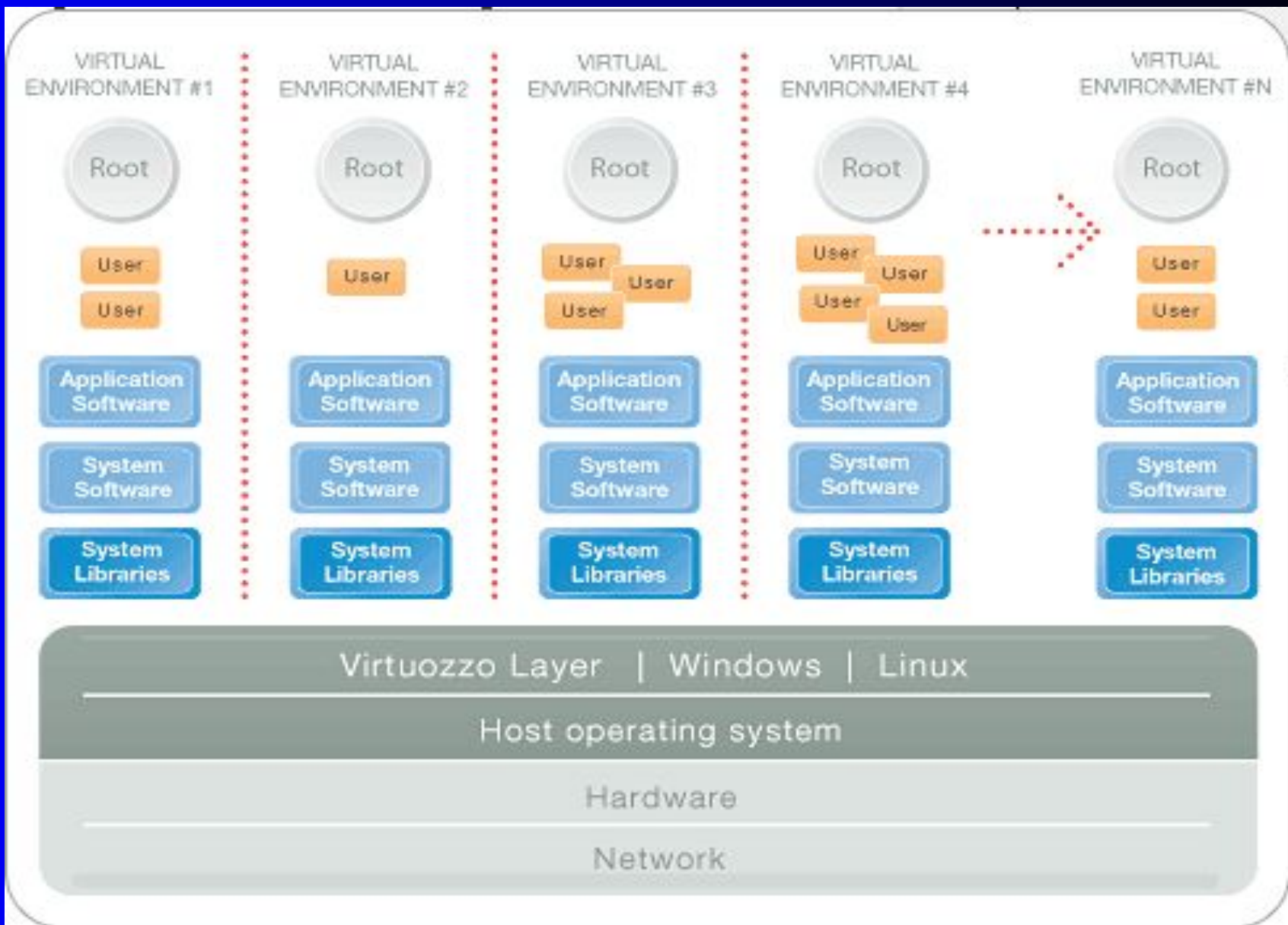
**Абсолютная изоляция - гарантируется безопасность, полная изоляция функций, ошибок и производительности виртуальных сред.**

**Динамическое выделение ресурсов - можно изменять ресурсы процессора, объем памяти, сетевых ресурсов, дискового пространства и подсистемы ввода-вывода без перезагрузки.**

**Миграция в реальном времени - функции обеспечения непрерывности бизнес-процесса, включая миграцию в реальном времени, гарантируют доступность и восстановимость данных.**

**Групповое управление - комплекс инструментов и шаблонов для автоматизированного администрирования множеством виртуальных сред и серверов.**





Виртуализация ОС заключается в создании виртуальных серверов на уровне операционной системы (ядра). Такой метод виртуализации предполагает создание изолированных разделов, или виртуальных окружений, на одном физическом сервере и одной копии ОС, чтобы добиться максимально эффективного использования ресурсов оборудования, программ, центров обработки данных и возможностей управленческого персонала.

Модель виртуализации ОС подверглась модернизации с целью достижения более высокой производительности, управляемости и эффективности. В основе находится стандартная главная операционная система, в случае с Virtuozzo это может быть Windows и Linux. Далее идет слой виртуализации (Virtuozzo Layer) с внутренней файловой системой и слой абстрагирования служб, которые обеспечивают изоляцию и безопасность ресурсов, выделенных для различных виртуальных окружений. Слой виртуализации служит для того, чтобы виртуальное окружение появилось как автономный сервер. Наконец, в самом виртуальном окружении размещается приложение или рабочая нагрузка.

Поддерживаемые архитектуры микропроцессоров: Virtuozzo для Linux: x86, ia64, AMD64, EM64T, Itanium; Virtuozzo для Windows: 32 и 64 бит

Минимальные требования к серверу: не менее 1 ГБ памяти и 4 ГБ свободного дискового пространства. Чем больше производительность процессора и объем памяти сервера, тем больше виртуальных частных серверов и приложений он может поддерживать.



# 1.11. Открытая платформа виртуализации VirtualBox

В 2006 году немецкая компания InnoTek представила продукт VirtualBox для виртуализации десктопов с открытым исходным кодом, в разработке которого (за исключением некоторых компонентов) может принять участие любой желающий.



**Платформа VirtualBox представляет собой настольную систему виртуализации для Windows, Linux и Mac OS хостов, поддерживающую операционные системы Windows, Linux, OS/2 Warp, OpenBSD и FreeBSD в качестве гостевых.**

- На данный момент VirtualBox включает в себя следующие возможности:
- нативная x86-виртуализация, не требующая наличия поддержки аппаратных техник Intel VT или AMD-V (которая, однако, может быть включена в настройках);
- дружественный пользовательский интерфейс (построенный с помощью Qt3);
- поддержка Windows, Linux и Mac OS хостовых систем;
- наличие Guest VM Additions для упрощения взаимодействия с хостовыми ОС и оптимизации их быстродействия;
- поддержка многопроцессорных и многоядерных систем;
- стабильность (в сравнении с другими Open Source решениями);
- поддержка виртуализации аудиоустройств;
- высокая производительность (выше, чем у продуктов VMware);
- поддержка различных видов сетевого взаимодействия;
- поддержка дерева сохраненных состояний виртуальной машины (snapshots), к которым может быть произведен откат из любого состояния гостевой системы;
- описание настроек виртуальной машины в XML-формате;
- поддержка Shared Folders для простого обмена файлами между хостовой и гостевой системами.

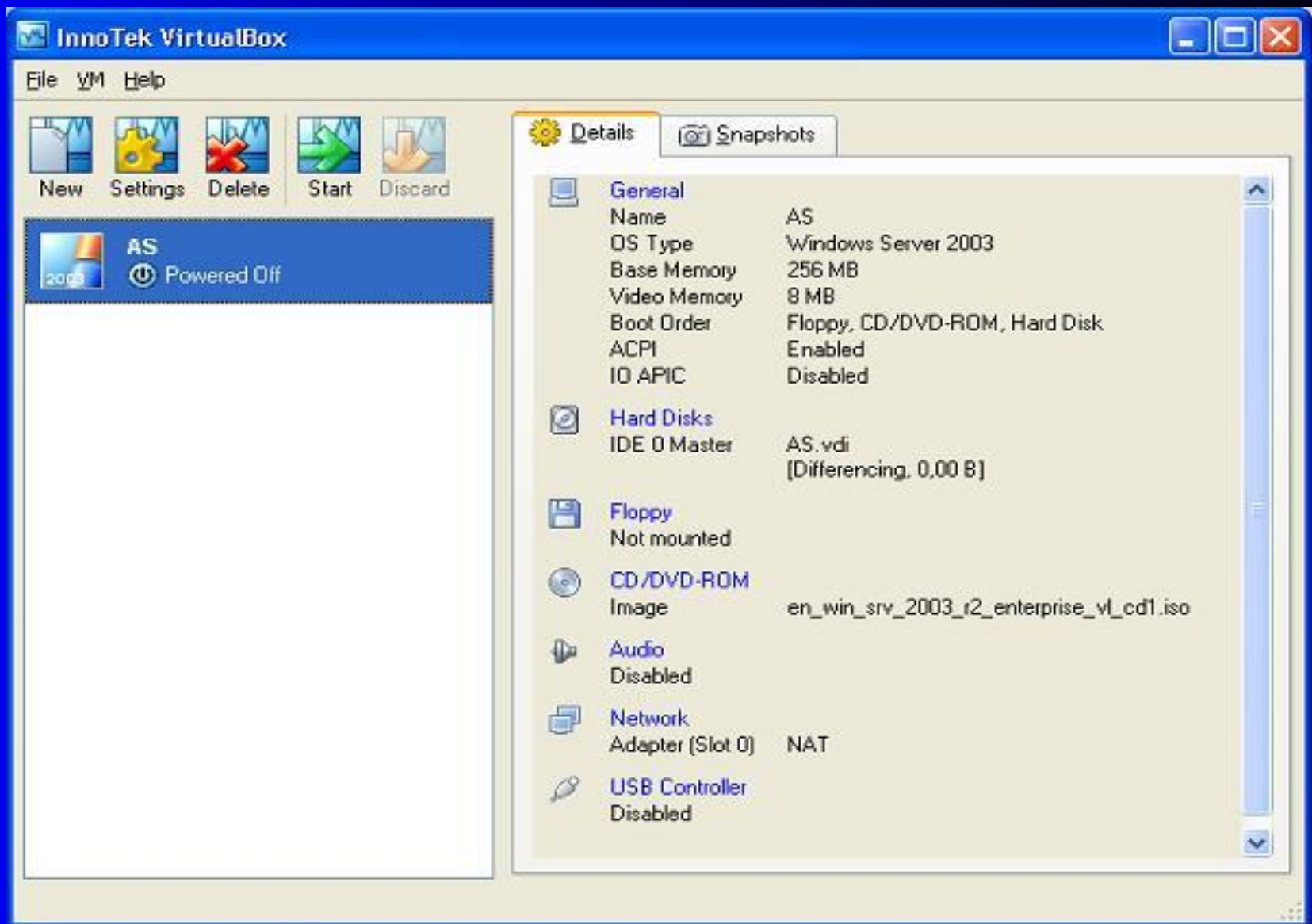




На данный момент продуктом поддерживаются следующие хостовые ОС:

- Операционные системы семейства Windows (2000/XP/2003/Vista)
- Linux-платформы, включая:
  - Ubuntu 7.04 («Feisty Fawn»)
  - Ubuntu 6.10 («Edgy Eft»)
  - Ubuntu 6.06 LTS («Dapper Drake»)
  - Debian 3.1 («Sarge»)
  - Debian 4.0 («Etch»)
  - openSUSE 10.2
  - Mandriva Linux 2007.1
  - Red Hat Enterprise Linux 4
  - Univention Corporate Server 1.3-2
- Mac OS X (в данный момент в стадии бета-тестирования)
- OS/2 Warp (экспериментально)







При старте виртуальной машины VirtualBox обычно запускается три процесса, которые можно наблюдать в диспетчере задач в Windows-системах или системном мониторе Linux:

1. Графический интерфейс окна управления.
2. Еще один похожий процесс, запущенный с параметром `startvm`, который означает, что GUI будет работать в качестве оболочки для виртуальной машины.
3. Автоматически создаваемый сервисный процесс `VBoxSVC`, необходимый для того, чтобы отслеживать количество и статусы запущенных виртуальных машин (поскольку они могут быть запущены различными способами).

Виртуальная машина с запущенной в ней гостевой системой инкапсулирует в себе необходимые детали реализации гостевой ОС и ведет себя по отношению к хостовой системе как обычное приложение.

## Преимущества и недостатки VirtualBox

Эксперты считают, что у этой платформы виртуализации определенно есть будущее, поскольку она готова занять пустующую нишу в сфере настольных систем виртуализации как мощная, производительная, удобная и, главное, бесплатная платформа. Безусловным плюсом системы является ее кроссплатформенность и поддержка со стороны сообщества Open Source. Большой список поддерживаемых гостевых и хостовых операционных систем открывает широкие возможности по применению VirtualBox в контексте различных вариантов использования.





- 2012 Windows 8
- 2008 Windows Server 2008
- 2007 Windows Vista, Windows 7
- 2005 Windows 2003, 64-разрядная
- 2003 Windows 2003 .NET Framework, MAC OS X
- 2000 Windows 2000
- Windows 4.0 – 1996
- 1995 Windows 95
- Корпоративные информационные системы
- NetWare 4.0 – 93, Windows NT 3.1 – 93
- Linux 0.01 - 1993
- 1990 MINIX – 87 (11800 стр. C + 800 стр. Asm.)
- OS/2 - 87
- 1985 OS-Net (Novell) - 83, MS-Net - 84, Windows 1.0 – 85
- Интернет (1983), Персональные компьютеры (1981)
- MS DOS 1.0 – (1981)
- 1980 Сети ЭВМ, UNIX, TCP/IP
- Локальные сети
- 1975 SNA (System Network Architecture), MULTICS
- Протокол X.25, телеобработка, базы данных
- 1965 Виртуальная ЭВМ, виртуальная память

