Архитектура операционной системы

1. Ядро и вспомогательные модули ОС

Наиболее общим подходом к структуризации операционной системы является разделение всех ее модулей на две группы:

- *ядро* – модули ОС, выполняющие основные функции;
- *модули,* выполняющие вспомогательные функции ОС.

Состав ядра

- Модули ядра выполняют такие базовые функции ОС, как управление процессами, памятью, устройствами ввода-вывода.
- В состав ядра входят функции, решающие внутрисистемные задачи организации вычислительного процесса, такие, как переключение контекстов, загрузка/выгрузка страниц, обработка прерываний. Эти функции недоступны для приложений.

- Другой класс функций ядра служит для поддержки приложений, создавая для них так называемую *прикладную программную среду.*
- Приложения могут обращаться к ядру с запросами – *системными вызовами* – для выполнения тех или иных действий, например, для открытия и чтения файла, вывода графической информации и т. д
- Функции ядра, которые могут вызываться приложениями, образуют интерфейс прикладного программирования – API.

Вспомогательные модули ОС обычно подразделяются на следующие группы:

- *утилиты* – программы, решающие отдельные задачи управления и сопровождения компьютерной системы, такие, как программы архивирования данных;
- *системные обрабатывающие программы* – текстовые или графические редакторы, компиляторы, отладчики;
- *программы предоставления пользователю дополнительных услуг* – специальный вариант пользовательского интерфейса, калькулятор;
- *библиотеки процедур* различного назначения, упрощающие разработку приложений, например библиотека математических функций и т. д.

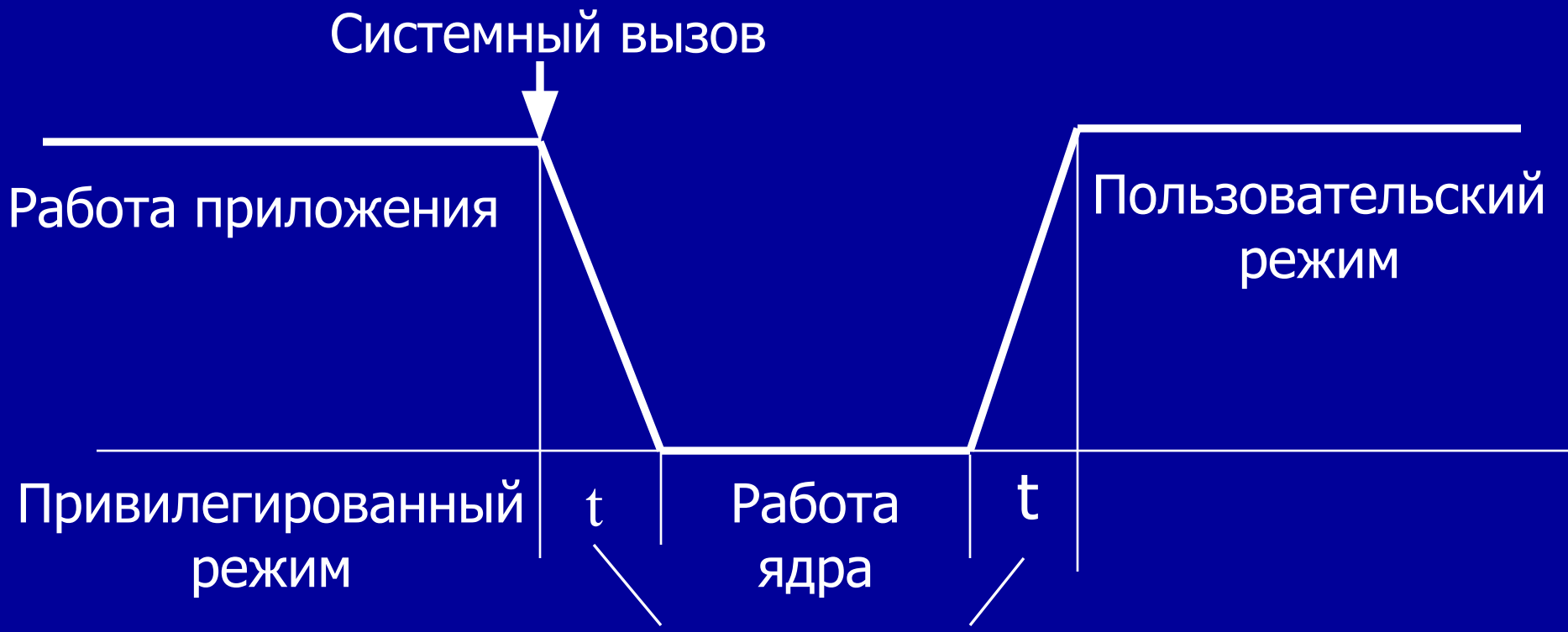
- Для обеспечения высокой скорости работы ОС все модули ядра или большая их часть постоянно находятся в оперативной памяти, то есть являются *резидентными.*
- Модули ОС, оформленные в виде утилит, системных обрабатывающих программ и библиотек, обычно загружаются в оперативную память только на время выполнения своих функций, то есть являются *транзитными.*

2 Ядро и привилегированный режим

- Для надежного управления ходом выполнения приложений операционная система должна иметь по отношению к приложениям определенные привилегии.
- Аппаратура компьютера должна поддерживать как минимум два режима работы – *пользовательский режим (user mode)* и *привилегированный режим,* который также называют *режимом ядра (kernel mode)* или *режимом супервизора (supervisor mode).*

- Архитектура ОС, основанная на привилегированном ядре и приложениях пользовательского режима, считается классической
- Приложения ставятся в подчиненное положение за счет запрета выполнения в пользовательском режиме некоторых критичных команд, связанных с переключением процессора с задачи на задачу, управлением устройствами ввода-вывода, доступом к механизмам распределения и защиты памяти.

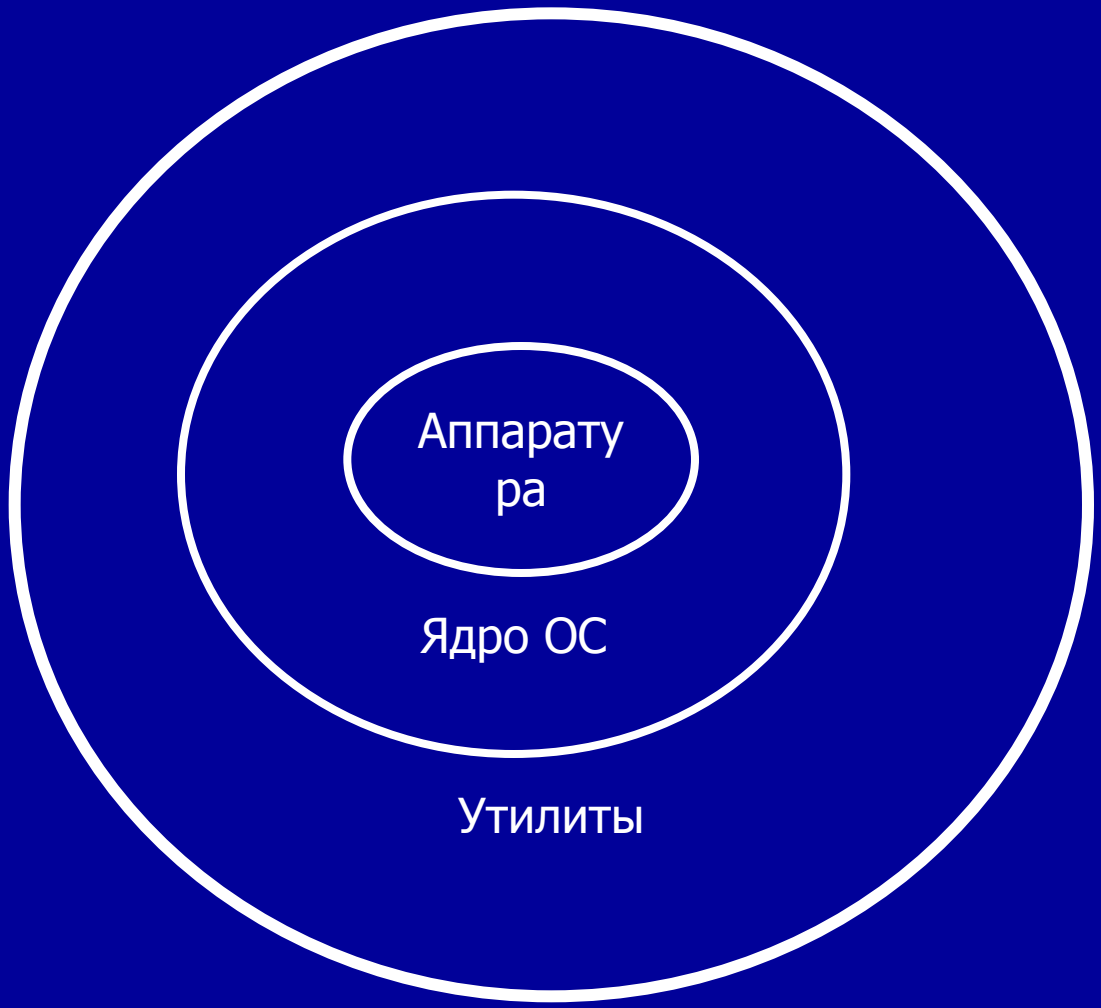
- Выполнение некоторых инструкций в пользовательском режиме запрещается безусловно (к таким инструкциям относится инструкция перехода в привилегированный режим), тогда как другие запрещается выполнять только при определенных условиях.
- Полный контроль ОС над доступом к памяти достигается за счет того, что инструкции конфигурирования механизмов защиты памяти разрешается выполнять только в привилегированном режиме.
- Системный вызов привилегированного ядра инициирует переключение процессора из пользовательского режима в привилегированный, а при возврате к приложению – переключение из привилегированного режима в пользовательский.



Из-за дополнительной двукратной задержки переключения переход на процедуру со сменой режима выполняется медленнее, чем вызов процедуры без смены режима.

2.3 Многослойная структура ОС

Трехслойная схема вычислительной системы



Ядро может состоять из следующих слоев:

1. Средства аппаратной поддержки
2. Машинно-зависимые компоненты ядра
3. Базовые механизмы ядра
4. Менеджеры ресурсов
5. Интерфейс системных вызовов

Средства аппаратной поддержки ОС

До сих пор об операционной системе говорилось как о комплексе программ, но часть функций ОС может выполняться и аппаратными средствами.

Типичный набор средств аппаратной поддержки ОС:

- средства поддержки привилегированного режима;
- средства трансляции адресов;
- средства переключения процессов;
- система прерываний;
- системный таймер;
- средства защиты областей памяти.

Машинно-зависимые компоненты ОС

- Этот слой образуют программные модули, в которых отражается специфика аппаратной платформы компьютера. В идеале этот слой полностью экранирует вышележащие слои ядра от особенностей аппаратуры. Это позволяет разрабатывать вышележащие слои на основе машинно-независимых модулей, существующих в единственном экземпляре для всех типов аппаратных платформ, поддерживаемых данной ОС.

Базовые механизмы ядра

- Этот слой выполняет наиболее примитивные операции ядра, такие, как программное переключение контекстов процессов, диспетчеризацию прерываний, перемещение страниц из памяти на диск и обратно и т. п. Модули данного слоя не принимают решений о распределении ресурсов – они только отрабатывают принятые «наверху» решения, что и дает повод называть их исполнительными механизмами для модулей верхних слоев.

Менеджеры ресурсов

- Этот слой состоит из мощных функциональных модулей, реализующих стратегические задачи по управлению основными ресурсами вычислительной системы. Обычно на данном слое работают менеджеры процессов, ввода-вывода, файловой системы и оперативной памяти.
- Каждый из менеджеров ведет учет свободных и используемых ресурсов определенного типа и планирует их распределение в соответствии с запросами приложений.

Интерфейс системных вызовов

- Этот слой является самым верхним слоем ядра и взаимодействует непосредственно с приложениями и системными утилитами, образуя прикладной программный интерфейс операционной системы.
- Функции API, обслуживающие системные вызовы, предоставляют доступ к ресурсам системы в удобной и компактной форме, без указания деталей их физического расположения.

Архитектура ОС семейства **Windows**

ОС Windows можно разделить на 2 части:

1. Основная часть ОС, работающая в режиме ядра (управление процессами, памятью, файловой системой , устройствами и т. д.).

2. Подсистемы окружения (среды), работающие в режиме пользователя (процессы, помогающие пользователям выполнять определенные системные функции).

Основная часть разделена на несколько уровней, каждый из которых пользуется службами лежащего ниже уровня. Основными уровнями являются:

- системные службы (сервисные процессы, являющиеся системными демонами);

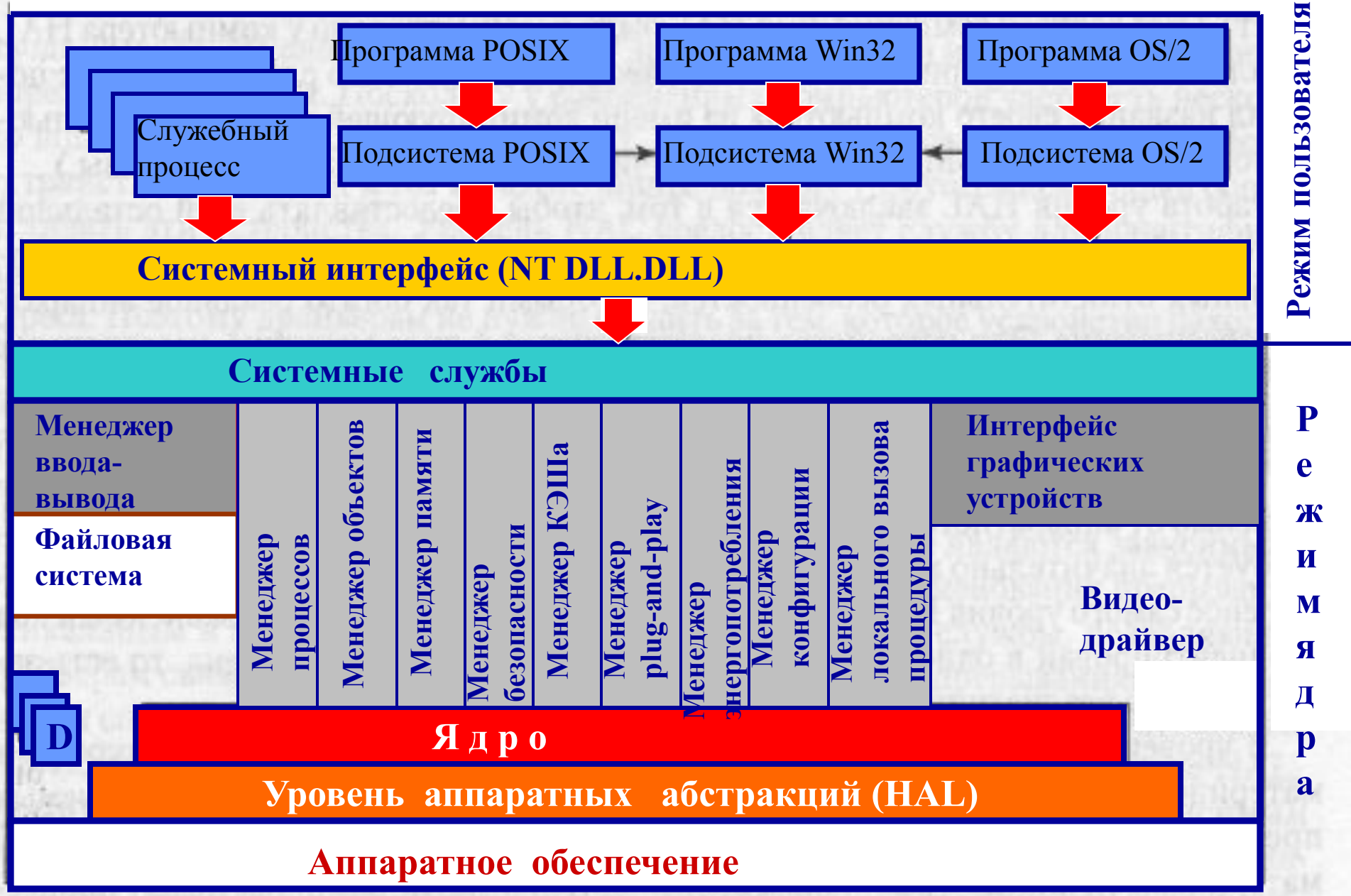
- исполняющая система (супервизор или диспетчер);

- драйверы устройств;

- ядро операционной системы;

- уровень аппаратных абстракций (HAL).

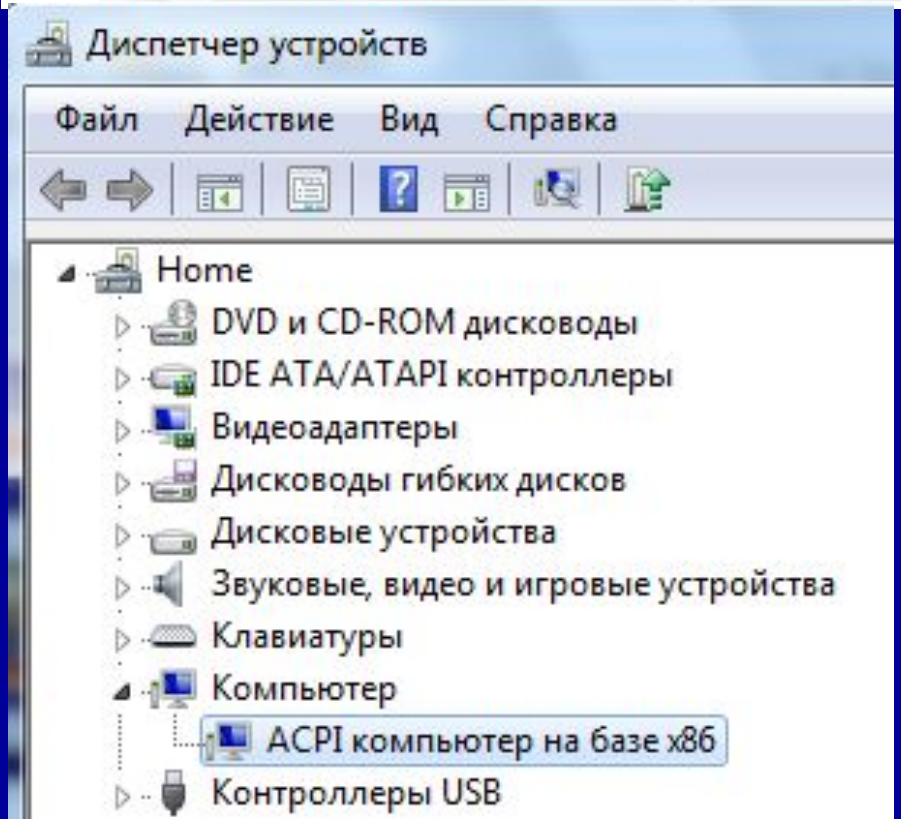
Два нижних уровня написаны на языке С и ассемблере и являются частично машинно-зависимыми. Верхние уровни написаны исключительно на языке С и почти полностью машинно-независимы. Драйверы написаны на С и в некоторых случаях на С++.


Программа POSIX
Программа Win32
Программа OS/2
Служебный процесс
Подсистема POSIX
Подсистема Win32
Подсистема OS/2
Системный интерфейс (NT DLL.DLL)
Режим пользователя
Системные службы
Менеджер ввода-вывода
Файловая система
Менеджер процессов
Менеджер объектов
Менеджер памяти
Менеджер безопасности
Менеджер КЭШа
Менеджер plug-and-play
Менеджер энергопотребления
Менеджер конфигурации
Менеджер локального вызова процедуры
Интерфейс графических устройств
Видео-драйвер
D
Ядро
Уровень аппаратных абстракций (HAL)
Аппаратное обеспечение
Режим ядра

В уровень HAL включены те службы, которые зависят от набора микросхем материнской платы и меняются от машины к машине :

- набор низкоуровневых функций (около 100), обеспечивающий стандартный интерфейс взаимодействия с аппаратнозависимыми элементами для функций , вызываемых компонентами ядра , драйверов и исполнительной системы , позволяющий абстрагироваться от того , на какой конкретно элементной базе (чипе контроллера прерывания, контроллера ПДП) реализовано выполнение доступа к шине, таймеру и т .д .
- В операционную систему включено несколько HAL-модулей , и Windows определяет во время загрузки, какой HAL-модуль нужно использовать.

Имя файла HAL	Поддерживаемые системы
Hal.dll	Стандартные персональные компьютеры (ПК)
Halacpi.dll	ПК с ACPI (Advanced Configuration and Power Interface)
Halapic.dll	ПК с APIC (Advanced Programmable Interrupt Controller)
Halaacpi.dll	ПК с APIC и ACPI
Halmps.dll	Многопроцессорные ПК
Halmacpi.dll	Многопроцессорные ПК с ACPI
Halborg.dll	Рабочие станции Silicon Graphics (только для Windows 2000; больше не продаются)
Halsp.dll	Compaq SystemPro (только для Windows XP)



- **Halacpi.dll** - Персональные компьютеры с усовершенствованным интерфейсом управления конфигурированием и энергопотреблением — Advanced configuration and Power Interface (ACPI).
- **Halmacpi.dll -** Персональные компьютеры с усовершенствованным программируемым контроллером прерываний — Advanced Programmable Interrupt Controller (APIC).
- На x64-машинах имеется только один HAL-образ по имени Hal.dll. Это обусловлено наличием у всех x64-машин материнских плат одинаковой конфигурации, поскольку процессы требуют поддержки ACPI и APIC.

Уровень ядра

Назначение ядра – сделать остальную часть ОС независимой от аппаратуры. Для этого ядро на основе низкоуровневых служб HAL формирует абстракции более высоких уровней. Например, у уровня HAL есть вызовы для связывания процедур обработки прерываний с прерываниями и установки их приоритетов. Больше ничего в этом отношении HAL не делает. Ядро же предоставляет полный механизм для переключения контекста и планирования потоков.

Ядро также предоставляет низкоуровневую поддержку двум классам объектов ядра – управляющим объектам и объектам диспетчеризации. Эти объекты используются системой и приложениями для управления ресурсами компьютерной системы: процессами, потоками , файлами и т. д.

Каждый объект ядра – это блок памяти, выделенный ядром, доступный только ему и представляющий собой структуру данных, в которой содержится информация об объекте.

К управляющим объектам ядра относятся: объекты заданий, процессов, потоков, прерываний, DPC (Deferred Procedure Call – отложенный вызов процедуры), APC (Asynchronous Procedure Call – асинхронный вызов процедуры)

К объектам диспетчеризации ядра относятся объекты, изменение состояния которых могут ждать потоки. Это –семафоры, мьютексы, события, таймеры, очереди, файлы, порты, маркеры доступа и др.

Исполняющая система

Написана на языке С, не зависит от архитектуры машины и относительно просто может быть перенесена на новые машины. Исполняющая система состоит из 10 компонентов, между которыми нет жестких границ. Компоненты одного уровня могут вызывать друг друга. Большинство компонентов представляют собой процедуры, которые выполняются потоками системы в режиме ядра.

Менеджер объектов управляет всеми объектами, создаваемыми или известными операционной системе (процессами, потоками, семафорами, файлами и т. д.). Он управляет пространством имен, в которое помещается созданный объект, чтобы к нему можно было обратиться по имени. Остальные компоненты ОС пользуются объектами во время своей работы.

Менеджер ввода-вывода предоставляет остальной части ОС независимый от устройств ввод-вывод, вызывая для выполнения физического ввода-вывода соответствующий драйвер.

Менеджер процессов управляет процессами и потоками, включая их создание и завершение. Он основывается на объектах **потоков** и процессов ядра и добавляет к ним дополнительные функции. Это ключевой элемент многозадачности.

Менеджер памяти реализует архитектуру виртуальной памяти со страничной подкачкой по требованию ОС. Он управляет преобразованием виртуальных страниц в физические и реализует правила защиты, ограничивающие доступ каждому процессу только теми страницами, которые принадлежат его адресному пространству.

Менеджер безопасности приводит в исполнение сложный механизм безопасности Windows, удовлетворяющий требованиям класса C2 Оранжевой книги Министерства обороны США.

Менеджер кэша хранит в памяти блоки диска, которые использовались последнее время, чтобы ускорить доступ к ним и обслуживает все файловые системы. Взаимодействует с менеджером виртуальной памяти, чтобы обеспечить требуемую непротиворечивость.

Менеджер plug-and-play управляет установкой новых устройств, контролирует их динамическое подключение и осуществляет при необходимости загрузку драйверов.

Менеджер энергопотребления управляет потреблением энергии, следит за состоянием батарей, взаимодействует с ИБП.

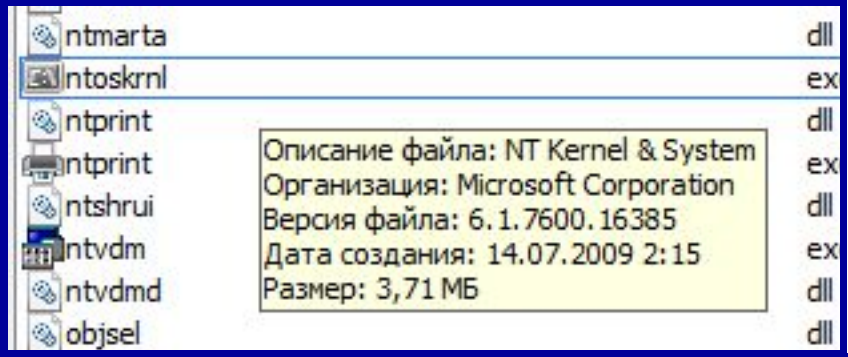
Менеджер конфигурации отвечает за сохранение реестра.

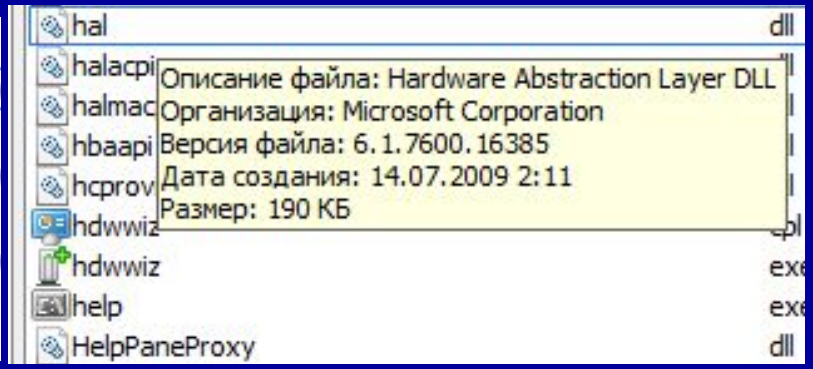
Менеджер вызова локальной процедуры обеспечивает высокоэффективное взаимодействие между процессами и их подсистемами. Поскольку этот путь нужен для выполнения некоторых системных вызовов, эффективность оказывается критичной, поэтому здесь не используются стандартные механизмы межпроцессного взаимодействия.

Интерфейс графических устройств GDI (Graphic Device Interface) управляет графическими изображениями для монитора и принтеров, предоставляя системные вызовы для пользовательских программ. (Интерфейс Win32 и модуль GDI превосходят по объему всю остальную исполняющую систему.)

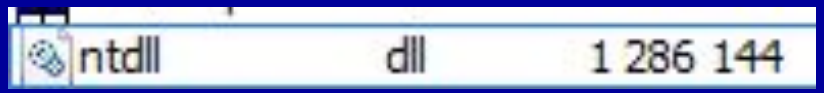
Системные службы предоставляют интерфейс к исполняющей системе. Они принимают системные вызовы Windows и вызывают необходимые части исполняющей системы для их выполнения.

Драйверы устройств устанавливаются в систему, добавляются в реестр и затем динамически загружаются при каждой загрузке системы. Драйверы не являются частью файла **ntoskrnl.exe.**





Основная часть ОС, состоящая из ядра и исполняющей системы, хранится в файле **ntoskrnl.exe .** Уровень **HAL**, представляющий собой библиотеку общего доступа, находится в файле **hal.dll.** Интерфейс Win32 и графических устройств хранятся в файле **win32.sys**. Системный интерфейс для связи пользовательского режима с режимом ядра – файл **NTDLL.DLL**



В каждой системе Windows выполняются перечисленные ниже процессы. (Два из них, Idle и System, не являются процессами в строгом смысле этого слова, поскольку они не выполняют какой-либо код пользовательского режима.)

- Процесс Idle (включает по одному потоку на процессор для учета времени простоя процессора).
- Процесс System (содержит большинство системных потоков режима ядра).
- Диспетчер сеансов (Smss.exe).
- Подсистема Windows (Csrss.exe).
- Процесс входа в систему (Winlogon.exe).
- Диспетчер управления сервисами (Services.exe) и создаваемые им дочерние процессы сервисов (например, универсальный процесс для хостинга сервисов, Svchost.exe).
- Серверный процесс локальной аутентификации (Lsass.exe).

Подсистемы окружения

В Windows существует три типа компонентов, работающих в режиме пользователя: динамические библиотеки DLL, подсистемы окружения и служебные процессы. Эти компоненты работают вместе, предоставляя пользовательским процессам три различных документированных интерфейса прикладного программирования API (Win32, POSIX, OS/2).

У каждого интерфейса есть список библиотечных вызовов, которые используются программистами. Работа библиотек DLL и подсистем окружения заключается в том, чтобы реализовать функциональные возможности опубликованного интерфейса, скрывая истинный интерфейс системных вызовов от прикладных программ.

Программы, пользующиеся интерфейсом Win32, содержит, как правило, большое количество обращений к функциям Win32 API. Поэтому, если работает одновременно несколько программ, то в памяти будут находиться многочисленные копии одних и тех же библиотечных процедур. Чтобы избежать подобной проблемы Windows поддерживает динамически присоединяемые библиотеки DLL. При этом при запуске нескольких приложений, использующих одну и ту же DLL, в памяти будет находиться только одна копия текста DLL.

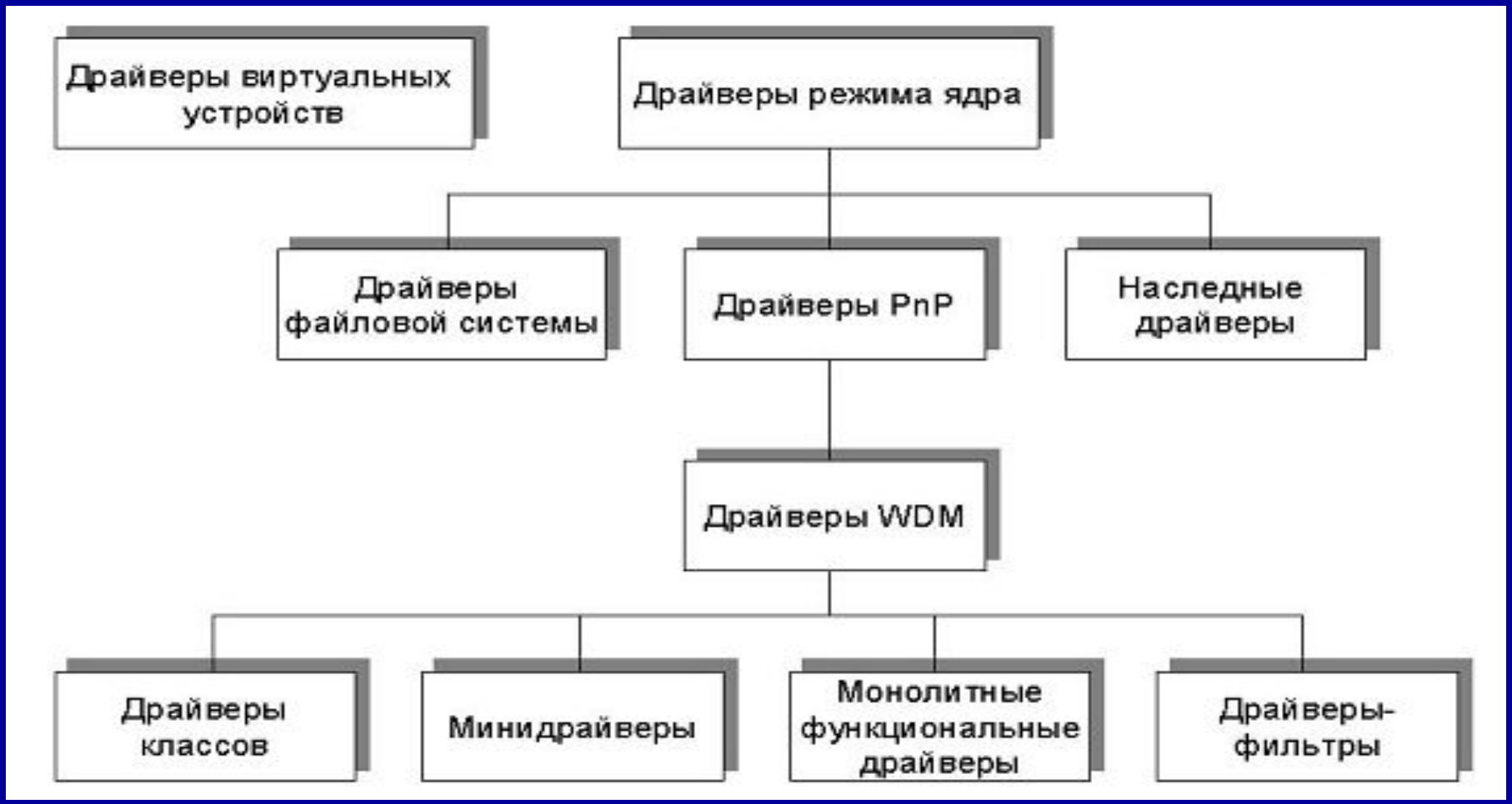
В каталоге \winnt\system32 есть более 2000 отдельных файлов DLL

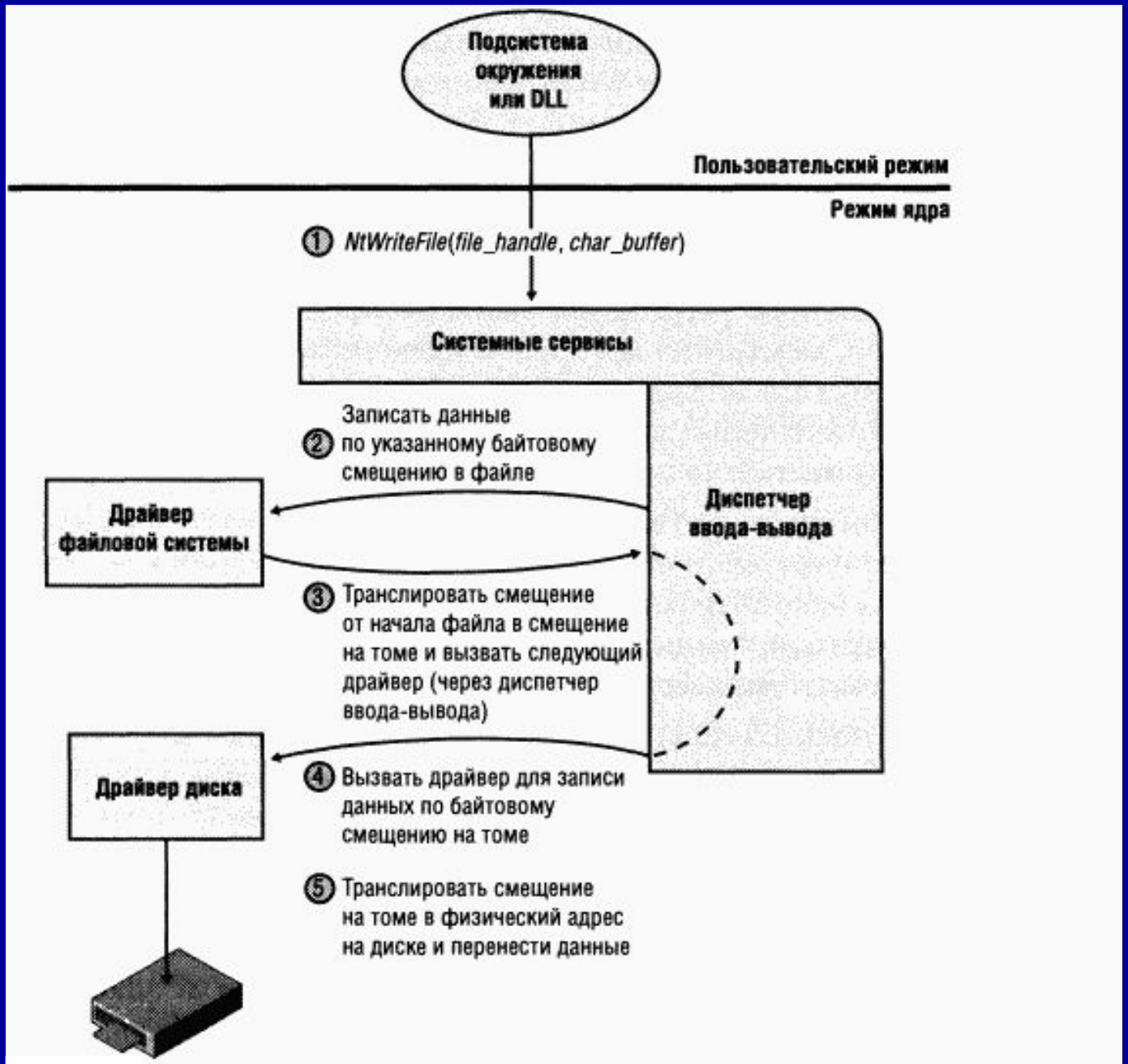
- Изначально Windows NT поставляется тремя подсистемами среды: Windows, POSIX и OS/2. Но подсистемы POSIX и OS/2 последний раз поставлялись с Windows 2000. Выпуски клиентской версии Windows Ultimate и Enterprise, а также все серверные версии включают поддержку для усовершенствованной подсистемы POSIX, которая называется подсистемой для приложений на основе Unix (Unix-based Applications, SUA).

Windows поддерживает два базовых типа драйверов:

1. Драйверы пользовательского режима (User-Mode Drivers):
   - Драйверы виртуальных устройств (Virtual Device Drivers, VDD) - используются для поддержки программ MS-DOS;
   - Драйверы принтеров (Printer Drivers).
2. Драйверы режима ядра (Kernel-Mode Drivers):
   - Драйверы файловой системы (File System Drivers) - реализуют ввод-вывод на локальные и сетевые диски;
   - Унаследованные драйверы (Legacy Drivers) - написаны для предыдущих версий Windows NT;
   - Драйверы видеоадаптеров (Video Drivers) - реализуют графические операции;
   - Драйверы потоковых устройств (Streaming Drivers) - реализуют ввод-вывод видео и звука;
   - WDM-драйверы (Windows Driver Model, WDM) - поддерживают технологию Plug and Play и управления электропитанием.

Типы драйверов

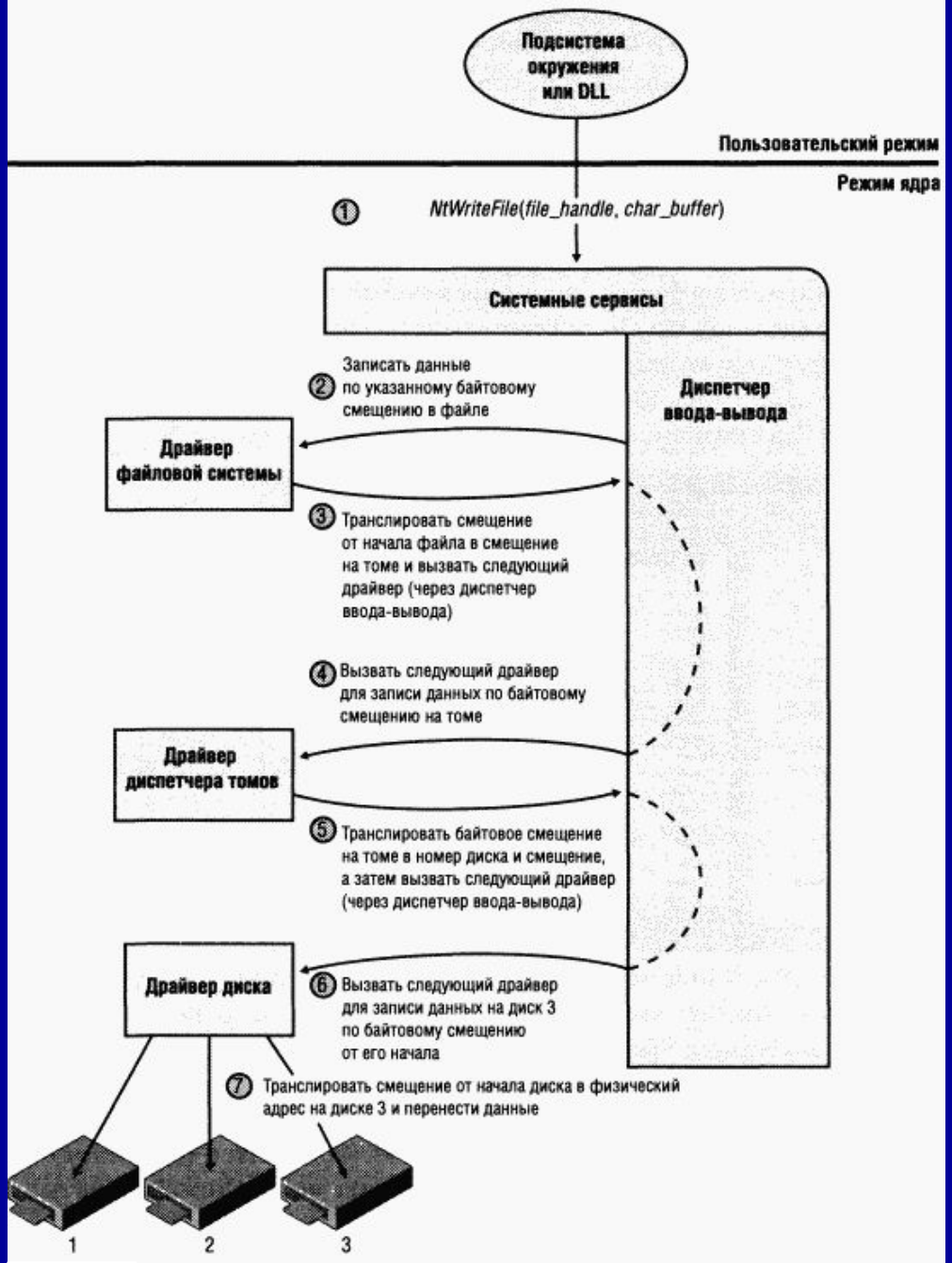




Взаимодействие драйвера файловой системы и драйвера диска

Одно- и многоуровневые драйверы

- **одноуровневые** (monolithic drivers)
- Но большинство драйверов, управляющих физическими устройствами являются **многоуровневыми** (layered drivers). Обработка запроса ввода-вывода разделяется между несколькими драйверами. Каждый выполняет свою часть работы. Например, запрос на чтение файла передается драйверу файловой системы, который, выполнив некоторые операции (например, разбиение запроса на несколько частей), передает его "ниже" - драйверу диска, а тот, в свою очередь, отправляет запрос драйверу шины. Кроме того между этими драйверами можно добавить любое количество драйверов-фильтров (например, шифрующих данные). Выполнив запрос нижестоящий драйвер (lower-level driver) передает его результаты "наверх" - вышестоящему (higher-level driver).



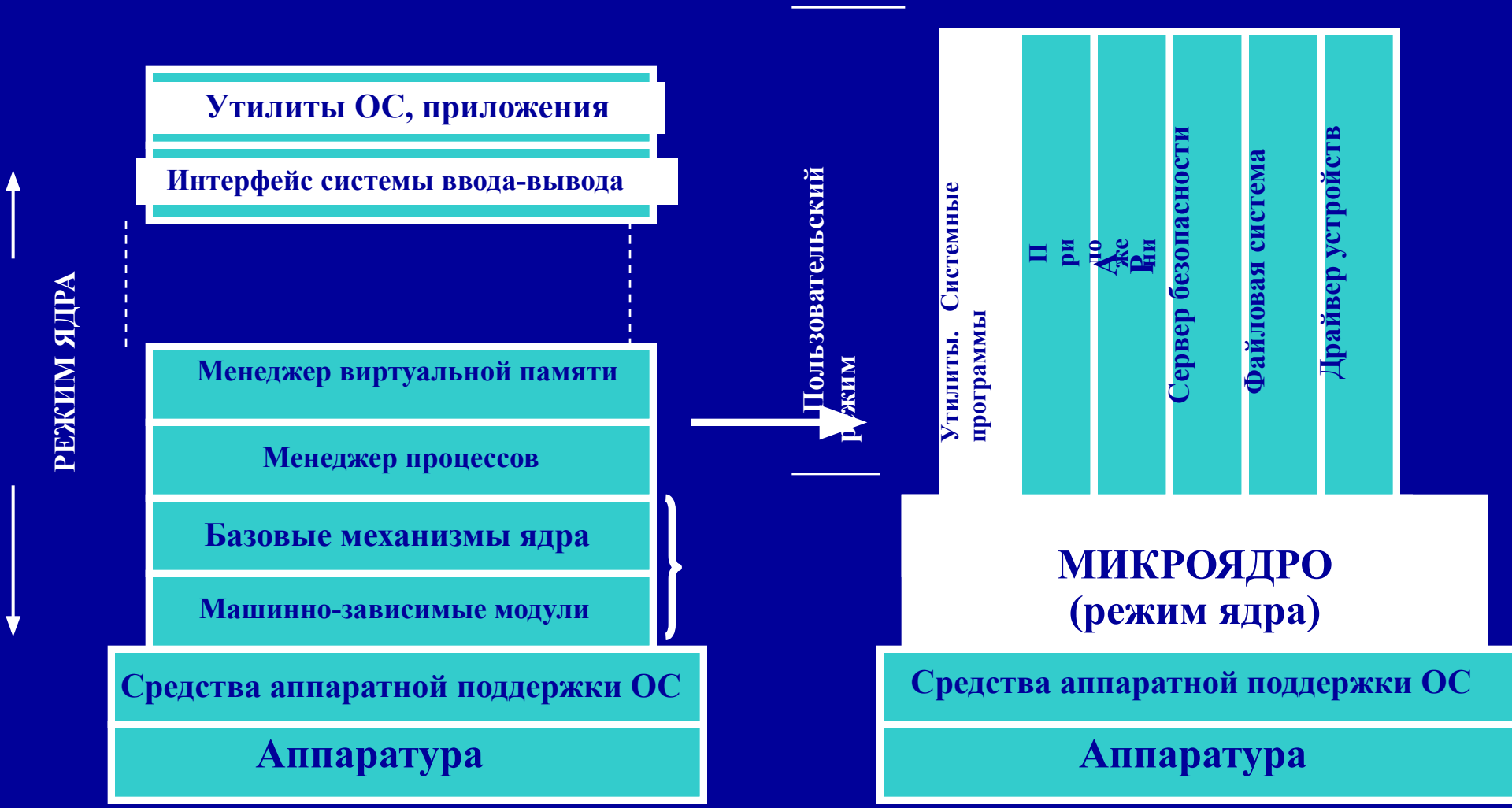
Добавление промежуточного драйвера

2.4 Концепция микроядерной архитектуры

- В привилегированном режиме остается работать только очень небольшая часть ОС, называемая *микроядром*
- Микроядро защищено от остальных частей ОС и приложений.
- В состав микроядра обычно входят машинно-зависимые модули, а также модули, выполняющие базовые функции ядра по управлению процессами, обработке прерываний, управлению виртуальной памятью, пересылке сообщений и управлению устройствами ввода-вывода, связанные с загрузкой или чтением регистров устройств.

- Все остальные более высокоуровневые функции ядра оформляются в виде приложений, работающих в пользовательском режиме
- Менеджеры ресурсов, являющиеся неотъемлемыми частями обычного ядра – файловая система, подсистемы управления виртуальной памятью и процессами, менеджер безопасности, – становятся «периферийными» модулями, работающими в пользовательском режиме и называются ***серверами* ОС**

Микроядерная архитектура ОС



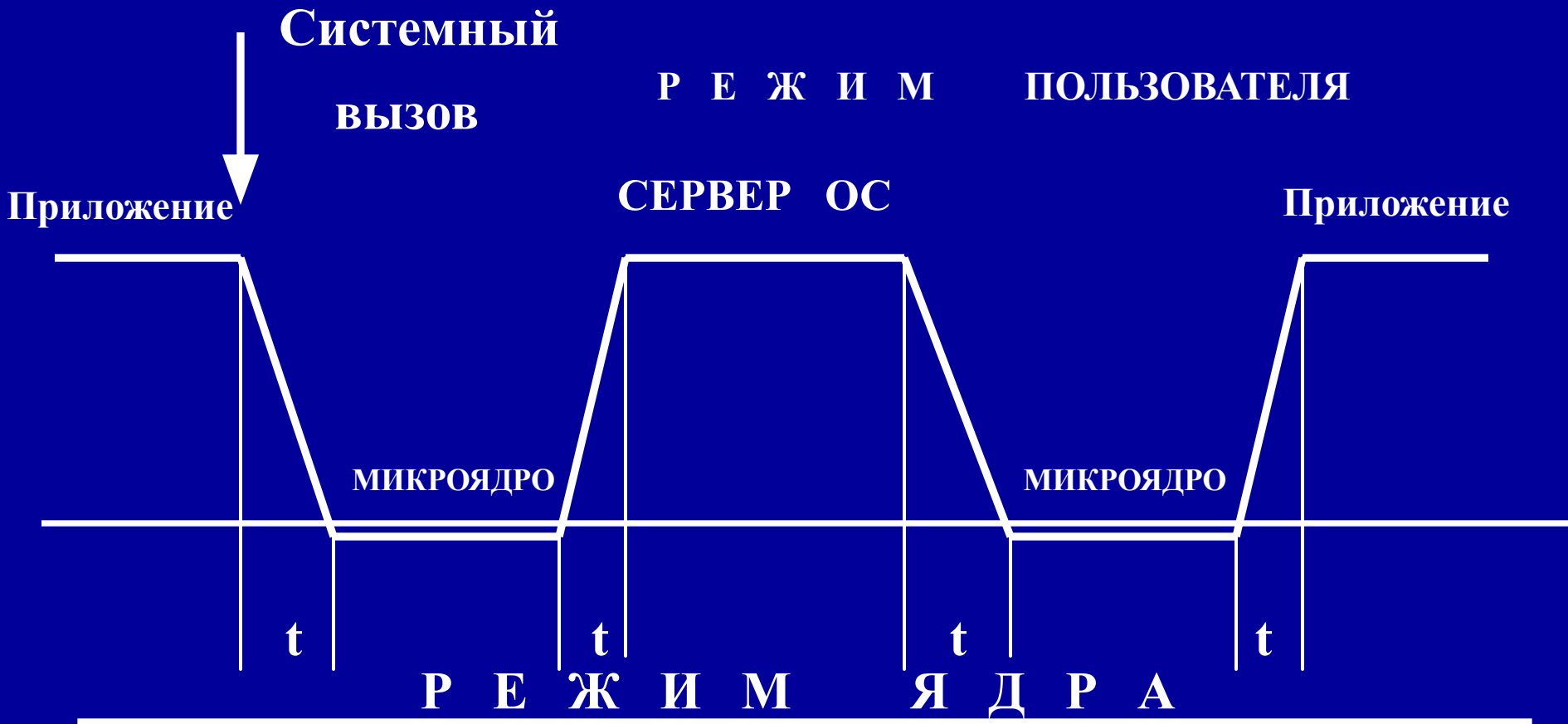

Приложение
Сервер памяти
Файл-сервер
Принт-сервер
Сервер процессов
Запрос
Ответ
Запрос
Ответ
РЕЖИМ ПОЛЬЗОВАТЕЛЯ
РЕЖИМ ЯДРА
МИКРОЯДРО
А П П А Р А Т У Р А

Механизм обращения к функциям ОС, оформленным в виде серверов

- Каждое приложение пользовательского режима работает в собственном адресном пространстве и защищено тем самым от вмешательства других приложений. Непосредственная передача сообщений между приложениями невозможна.
- Клиент, которым может быть либо прикладная программа, либо другой компонент ОС, запрашивает выполнение некоторой функции у соответствующего сервера, посылая ему сообщение.

- Микроядро, имеет доступ к адресным пространствам каждого из этих приложений и поэтому может работать в качестве посредника.
- Микроядро сначала передает сообщение, содержащее имя и параметры вызываемой процедуры нужному серверу, затем сервер выполняет запрошенную операцию, после чего микроядро возвращает результаты клиенту с помощью другого сообщения.
- Таким образом, работа микроядерной операционной системы соответствует известной модели **клиент-сервер**, в которой роль транспортных средств выполняет микроядро.

Смена режимов при выполнении вызова функции микроядра



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

Недостатки микроядерной архитектуры

- При классической организации ОС выполнение системного вызова сопровождается двумя переключениями режимов, а при микроядерной организации – четырьмя. Таким образом, операционная система на основе микроядра при прочих равных условиях всегда будет менее производительной, чем ОС с классическим ядром.

Классификация ядер ОС

1. Наноядро (НЯ) – крайне упрощённое 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. Бóльшая 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 боль-шую 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 производительно, чем микроядро, поскольку работает как один большой процесс. МЯ является большинство 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.

UEFI

- Аббревиатура UEFI расшифровывается как **Unified Extensible Firmware Interface** (унифицированный расширяемый интерфейс прошивки). Эта технология предназначена для преобразования традиционной системы загрузки компьютеров и должна прийти на смену устаревшей системе **BIOS**. Однако это не просто модернизация старой технологии, а принципиально новый подход к технологии загрузки компьютера и запуска ОС. По сути, UEFI практически не имеет ничего общего с системой PC BIOS.
- Если BIOS – это код (жесткий и фактически неизменный), прошитый в специальном BIOS -чипе на системной плате, то UEFI – гибкий программируемый интерфейс, расположенный поверх всего аппаратных компонентов компьютера с их собственными прошивками. Код UEFI (намного больший по размеру, чем загрузочный код BIOS) находится в специальном каталоге /EFI/, который может храниться в самых различных местах: от отдельной микросхемы на системной плате, до раздела на жестком диске или сетевом хранилище. По сути – UEFI – это самостоятельная легкая операционная система, представляющая собой интерфейс между основной ОС и микропрограммами, управляющих аппаратным низкоуровневыми функциями оборудования, которая должна корректно инициализировать оборудование и передать управление загрузчику основной («большой») ОС, установленной на компьютере.


ASUS UEFI BIOS Utility - EZ Mode
Exit/Advanced Mode
10:18
Monday [7/23/2012]
P9X79 DELUXE
BIOS Version : 1203
CPU Type : Intel(R) Core(TM) i7-3820 CPU @ 3.60GHz
Total Memory : 8192 MB (DDR3 1862MHz)
English
Build Date : 05/24/2012
Speed : 4200 MHz
Temperature
CPU +95.0°F/+35.0°C
MB +86.0°F/+30.0°C
Voltage
CPU 1.482V
5V 5.120V
3.3V 3.280V
12V 12.288V
Fan Speed
CPU_FAN 1034RPM
CPU_OPT_FAN N/A
CHA_FAN1 896RPM
CHA_FAN2 N/A
System Performance
Quiet
Performance
Energy Saving
The advanced options or the hardware setup have been changed
Boot Priority
UEFI
Use the mouse to drag or keyboard to navigate to decide the boot priority.
Shortcut(F3)
Boot Menu(F8)
Default(F5)

- В состав UEFI входят сервисы тестирования железа, загрузочные и тестовые сервисы, а также реализации стандартных протоколы взаимодействия (в том числе сетевых), драйверы устройств, функциональные расширения и даже собственная EFI-оболочку, в которой можно запускать собственные EFI-приложения. Т.е. уже на уровне UEFI можно выйти в интернет, или организовать бэкап жесткого диска с помощью привычного пользователям графического GUI.
- Одними из наиболее востребованными особенностями UEFI, которые можно реализовать на работающем под ней компьютере являются: «безопасная загрузка» (**secure boot** в Windows 8), низкоуровневая криптография, сетевая аутентификация, универсальные графические драйверы и еще многое другое. UEFI поддерживает 32-х и 64-х битные процессоры и может быть использована на системах с процессорами Itanium, x86, x64 и ARM

- Для возможности загрузки старых ОС, поддерживающих только BIOS, в UEFI существует режим эмуляции BIOS, который называется Compatibility Support Module (CSM).
- Благодаря UEFI Windows 8 можно устанавливать на диски объёмом 3 ТБ и больше, и, соответственно, загружаться с этих дисков. Это связано с переходом от таблицы разделов MBR в (BIOS) к GPT (UEFI).
- Использование UEFI вместо BIOS, – это один из ключевых компонентов, обеспечивающих быструю загрузку Windows 8 (код UEFI работает быстрее за счет того, что целиком писался с нуля, без необходимостей тянуть за собой шлейф древних правил и совместимостей). Кроме того, в UEFI при чтении используется особый размер блока EFI I/O, позволяющий читать по 1 мб данных за раз (в BIOS – 64кб). Кроме того уменьшение времени запуска достигается за счет того, что нет необходимости искать загрузчик на всех устройствах: загрузочный диск назначается в UEFI на этапе установки ОС.
- Итак, мы отметили, что Windows 8 поддерживает загрузку UEFI, однако есть ряд особенностей:
- Компьютер должен совместим с UEFI v2.3.1
- UEFI поддерживается только в 64 разрядной версии Windows 8. 32-битные версии Windows не поддерживают функции UEFI (на новых компьютерах этой ОС придется работать в режиме эмуляции CSM).
- Windows 8 для ARM (Windows RT) не будет работать на оборудовании, не поддерживающем UEFI, или позволяющим отключить Secure Boot

- В последующих версиях Windows (и ближайшем Windows 8 SP1) разработчики планируют внедрение множества других функций UEFI, таких как: Rootkit prevention (обнаружение руткитов в процессе загрузки), Network authentication (аутентификация при загрузке, особенно актуальная в сценариях удаленного разворачивания ОС) и т.д.
- Доступ к настройкам UEFI из Windows 8
- Стоит отметить, что на новых компьютерах с предустановленной Windows 8, который использует UEFI, чтобы попасть в меню настройки UEFI (замены старичка BIOS), привычный способ нажатия на клавишу Delete или F2 (или другой клавиши заданной вендором) не сработает. Т.к. Windows 8 (особенно на SSD) грузится очень быстро, сложно успеть за это время нажать клавишу для входа в режим настройки UEFI. Где-то писалось, что Windows 8 на SSD с UEFI ждет нажатия клавиши всего 200мс. Поэтому существует процедура вызова программы настроек параметров UEFI из загрузочного меню Windows 8.
- Попасть в загрузочное меню Windows 8 можно одним из трех способов:
- В настройках ПК (**PC settings**) выберите раздел Общие (**Generel**) и в подразделе Особые варианты загрузки (**Advanced Startup**) нажмите кнопку Перезагрузить сейчас (**Restart now**)
- Того же самого эффекта можно достигнуть, зажав кнопку Shift при нажатии кнопки Restart в меню завершения работы.
- Другой вариант попадания в загрузочное меню Windows – выполнение в командной строке команды shutdown.exe /r /o /f /t 00
- После перезагрузки автоматически откроется меню загрузки Windows 8, в котором необходимо выбрать пункты **Troubleshoot**->**Advanced options**. В окне расширенных опций есть отдельная кнопка **UEFI Firmware Settings**, позволяющая после перезагрузки ПК попасть прямиком в BIOS компьютера (на самом деле это UEFI, настройки в котором эквивалентны традиционному