Современные ОС. ОС Windows

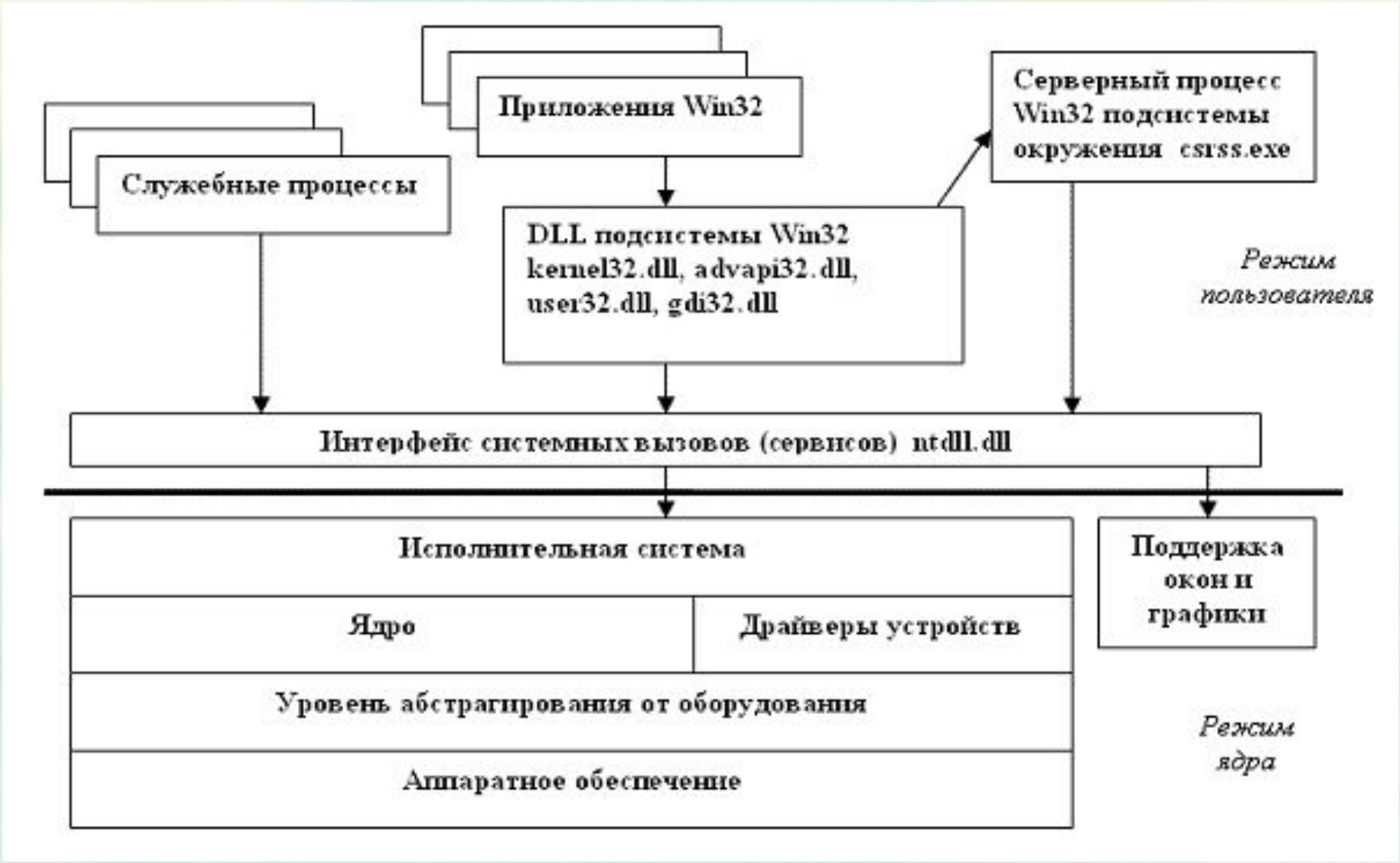
Структура ОС Windows

Первые версии системы **Windows** создавались с использованием микроядерного подхода, основанный на микроядре Mach, которое было разработано в университете Карнеги-Меллона. Архитектура более поздних версий системы микроядерной уже не является.

Большой объем системного кода (управление системными вызовами и экранная графика) был перемещен из адресного пространства пользователя в пространство ядра и работает в привилегированном режиме.

В результате в ядре ОС Windows переплетены **элементы микроядерной архитектуры и элементы монолитного ядра** (комбинированная система). Основные компоненты ядра Windows NT располагаются в вытесняемой памяти и взаимодействуют друг с другом путем передачи сообщений (микроядерная архитектура) В тоже время все компоненты ядра работают в одном адресном пространстве и активно используют общие структуры данных, что свойственно операционным системам с монолитным ядром.

Структура ОС Windows



Структура ОС Windows

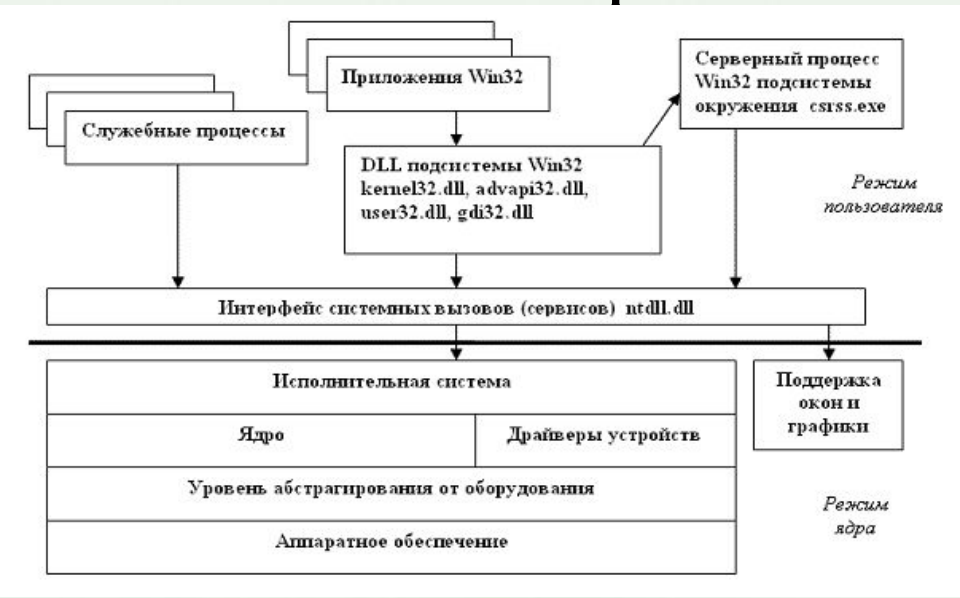
ОС Windows состоит из компонентов, работающих в режиме ядра, и компонентов, работающих в режиме пользователя.

Задача **уровня абстрагирования от оборудования** (hardware abstraction layer, **HAL**) - скрыть аппаратные различия аппаратных архитектур для потенциального переноса системы с одной платформы на другую. HAL предоставляет выше лежащим уровням аппаратные устройства в абстрактном виде, свободном от индивидуальных особенностей. Это позволяет изолировать ядро, драйверы и исполнительную систему ОС Windows от специфики оборудования (например, от различий между материнскими платами).

Корпорация Microsoft называет **ядром (kernel) компонент, находящийся в невыгружаемой памяти и содержащий низкоуровневые функции операционной системы**, такие, как диспетчеризация прерываний и исключений, планирование потоков и др. Оно также предоставляет набор процедур и базовых объектов, применяемых компонентами высших уровней.

Ядро и HAL являются аппаратно-зависимыми и написаны на языках Си и ассемблера. Верхние уровни написаны на языке Си и являются машинно-независимыми.

Структура ОС Windows



Исполнительная система (executive) обеспечивает управление памятью, процессами и потоками, защиту, ввод-вывод и взаимодействие между процессами.

Драйверы устройств содержат аппаратно-зависимый код и обеспечивают трансляцию пользовательских вызовов в запросы, специфичные для конкретных устройств.

Подсистема поддержки окон и графики реализует функции графического пользовательского интерфейса (GUI), более известные как Win-32-функции модулей USER и GDI

В пространстве пользователя работают разнообразные **сервисы**, управляемые **диспетчером сервисов** и решающие системные задачи. Некоторые системные процессы (например, обработка входа в систему) диспетчером сервисов не управляются и называются **фиксированными процессами поддержки системы**. Пользовательские приложения (user applications) могут быть пяти типов: Win32, Windows 3.1, MS-DOS, POSIX и OS/2 1.2. Среду для выполнения пользовательских процессов предоставляют три **подсистемы окружения**: Win32, POSIX и OS/2. Таким образом, пользовательские приложения не могут вызывать системные вызовы ОС Windows напрямую, а вынуждены обращаться к DLL (**Dynamic-link library — динамически подключаемая библиотека**) подсистем.

Структура ОС Windows

Подсистема Win32

Win32 API - 32-разрядный API для современных версий Windows.

Базовые функции этого API реализованы в DLL **kernel32.dll** и **advapi32.dll**; базовые модули GUI — в **user32.dll** и **gdi32.dll**.

Win32 появился вместе с Windows NTWin32 появился вместе с Windows NT и затем был перенесён (в несколько ограниченном виде) в системы серии Windows 9x.

В современных версиях Windows, происходящих от Windows NT, работу Win32 GUI обеспечивают два модуля: csrss.exe (*Client/Server Runtime Subsystem*), работающий в пользовательском режиме, и win32k.sys в режиме ядра.

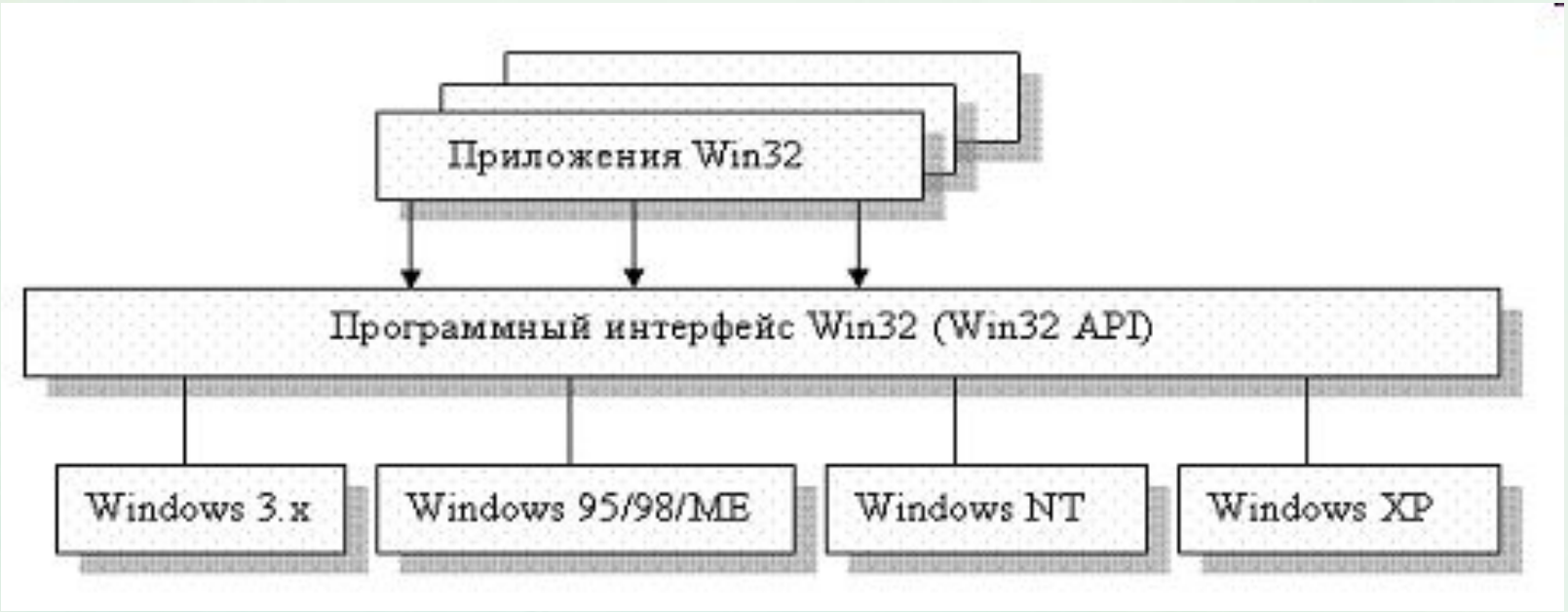
Работу же системных Win32 API обеспечивает ядро - ntoskrnl.exe

Основные компоненты Win32 реализованы в следующих системных файлах, находящихся в каталоге system32:

- **ntoskrnl.exe** - исполнительная система и ядро;
- **ntdll.dll** - внутренние функции поддержки и интерфейсы диспетчера системных сервисов с функциями исполнительной системы;
- **hal.dll** - уровень абстрагирования от оборудования;
- **win32k.sys** - часть подсистемы Win32, работающая в режиме ядра;
- kernel32.dll, advapi32.dll, user32.dll, gdi32.dll - основные dll подсистемы Win32.

Структура ОС Windows

Приложение, ориентированное на использование Win32 API, может работать практически на всех версиях Windows, несмотря на то, что сами системные вызовы в различных системах различны. Таким образом Microsoft обеспечивает преемственность своих операционных систем.



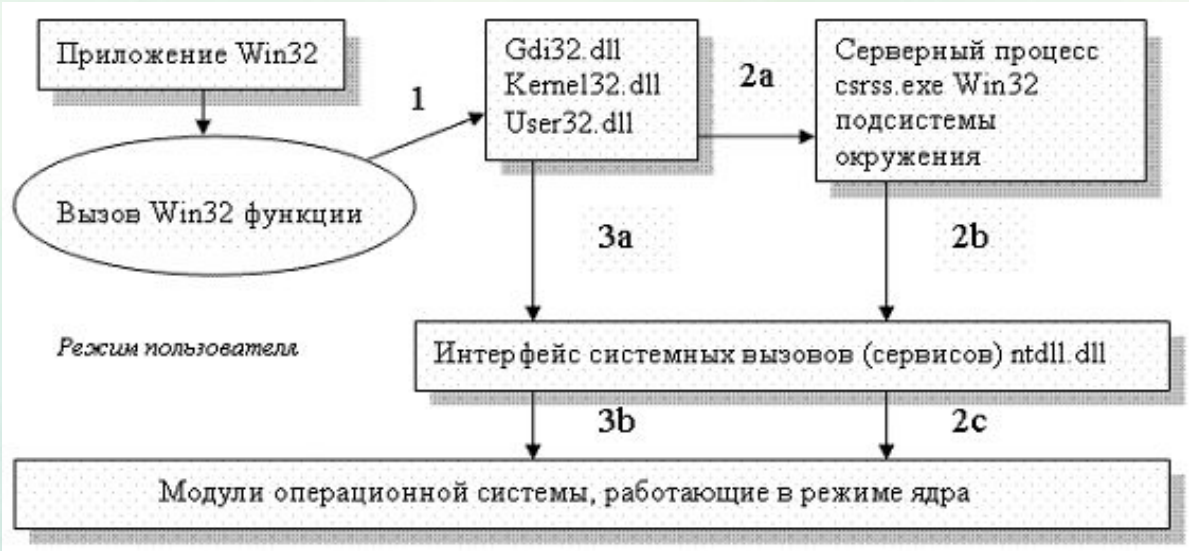
Структура ОС Windows

При запуске процесса все требуемые динамические библиотеки отображаются на его виртуальное адресное пространство, а для быстрого вызова библиотечной процедуры используется специальный вектор передачи.

При вызове приложением одной из Win32-функций dll-подсистем может возникнуть одна из трех ситуаций:

1. Функция полностью выполняется внутри одной dll.

2. Для выполнения функции привлекается сервер csrss, для чего ему посылается сообщение (2a, за которым обычно следуют 2b и 2c).



3. Данный вызов транслируется в системный сервис (системный вызов), который обычно обрабатывается в модуле ntdll.dll (шаги 3a и 3b). Например, Win32-функция ReadFile выполняется с помощью недокументированного сервиса NtReadFile.

В первых версиях ОС Windows практически все вызовы Win32 API выполнялись, следуя маршруту 2 (2a, 2b, 2c). После того, как существенная часть кода системы для увеличения производительности была перенесена в ядро (начиная с Windows NT 4.0), вызовы Win32 API, как правило, идут напрямую по 3-му (3a, 3b) пути, минуя подсистему окружения Win32.

Структура ОС Windows

Объекты ядра

Для работы с важными системными ресурсами ОС Windows создает объекты, управление которыми осуществляет менеджер объектов.

Понятие "**объект ядра**" имеет разный смысл в различных интерпретациях.
Дело в том, что совокупность объектов образует слоеную структуру. Ядро поддерживает базовые объекты двух видов: **объекты диспетчера** (события, мьютексы, семафоры, потоки ядра, таймеры и др.) и **управляющие** (DPC, APC, прерывания, процессы, профили и др.)

Объекты исполнительной системы предназначены для управления памятью, процессами и межпроцессным обменом. К ним относятся такие объекты, как: процесс, поток, открытый файл, семафор, мьютекс и ряд других. (Именно эти объекты и называются объектами ядра в руководствах по программированию).

Структура ОС Windows

Объекты ядра

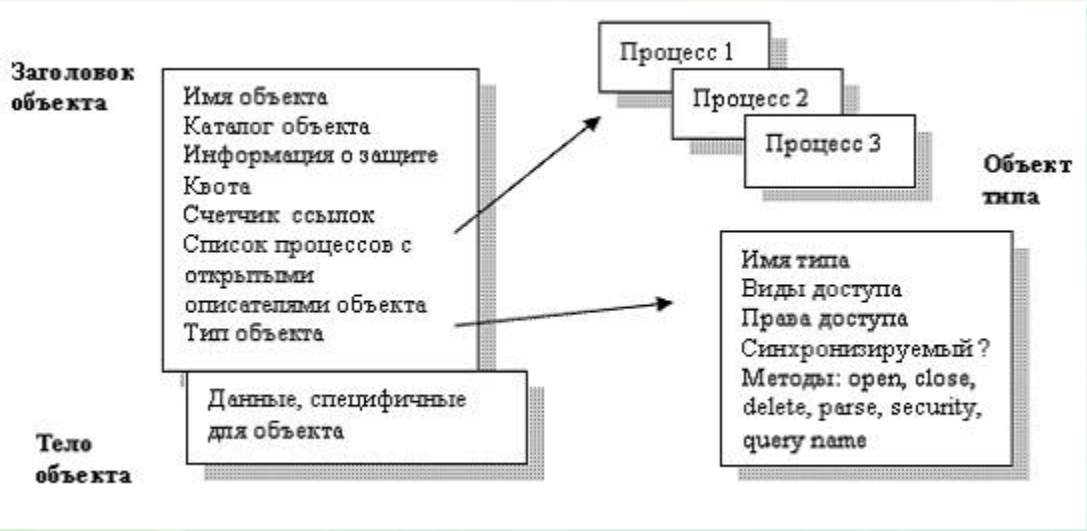
Внешнее отличие объектов ядра (объектов исполнительной системы) от объектов User и GDI состоит в наличии у первых атрибутов защиты. Далее эти объекты ядра (объекты исполнительной системы) будут называться просто объектами.

Объект представляет собой блок памяти в виртуальном адресном пространстве ядра. Этот блок содержит информацию об объекте в виде структуры данных. Структура содержит как общие, так и специфичные для каждого объекта элементы. Объекты создаются в процессе загрузки и функционирования ОС и теряются при перезагрузке и выключении питания.

Содержимое объектов доступно только ядру, приложение не может модифицировать его непосредственно. Доступ к объектам можно осуществить только через его функции-методы (инкапсуляция данных), которые инициируются вызовами некоторых библиотечных Win32-функций.

Структура ОС Windows

Объекты ядра



- Каждый объект имеет «заголовок» с информацией, общей для всех объектов, а также данные, специфичные для объекта. Например, в поле заголовка имеется список процессов, открывших данный объект, и информация о защите, определяющая, кто и как может использовать объект.
- Счетчик ссылок на объект увеличивается на 1 при открытии объекта и уменьшается на 1 при его закрытии.

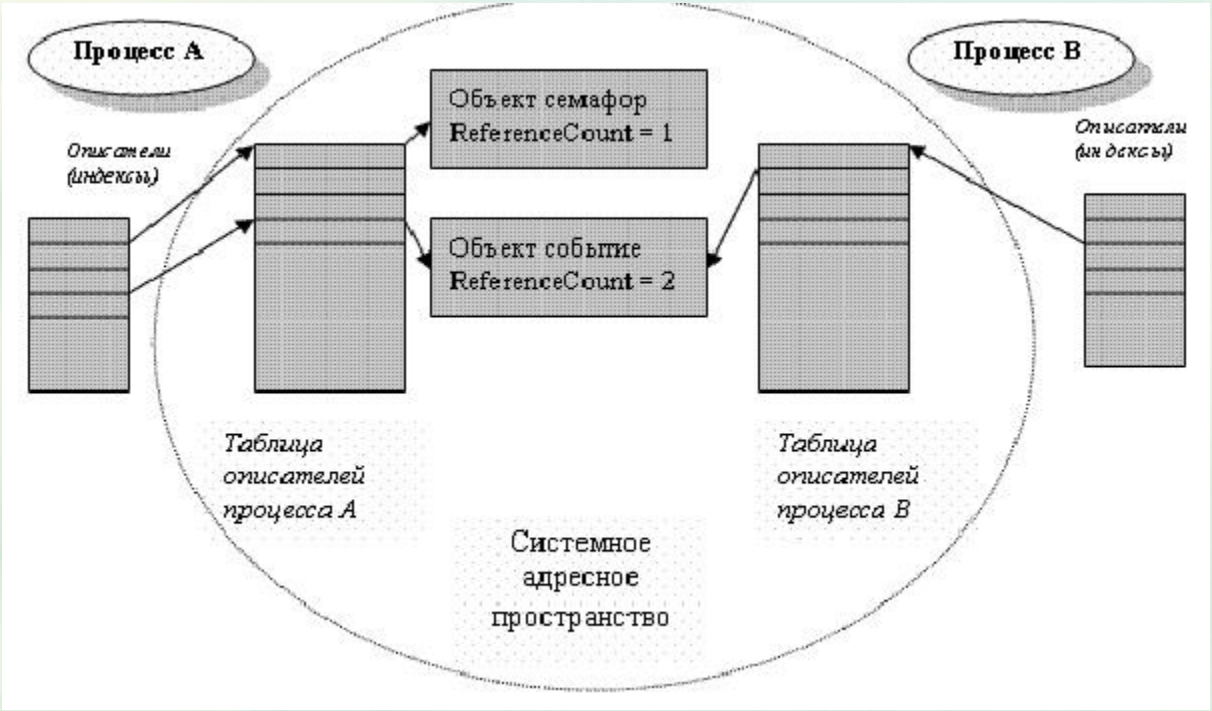
Квота устанавливает ограничения на объемы ресурсов. Например, по умолчанию лимит на открытые объекты для процесса - 230. Множество объектов делится на типы, а у каждого из объектов есть атрибуты, неизменные для объектов данного типа. Ссылка на тип объекта также входит в состав заголовка.

В состав компонентов объекта типа входит атрибут методы - указатели на внутренние процедуры для выполнения стандартных операций. Методы вызываются диспетчером объектов при создании и уничтожении объекта, открытии и закрытии описателя объекта, изменении параметров защиты. Система позволяет динамически создавать новые типы объектов. В этом случае предполагается регистрация его методов у диспетчера объектов. Например, метод open вызывается всякий раз, когда создается или открывается объект и создается его новый описатель.

Структура ОС Windows

Объекты ядра

Описатели объектов



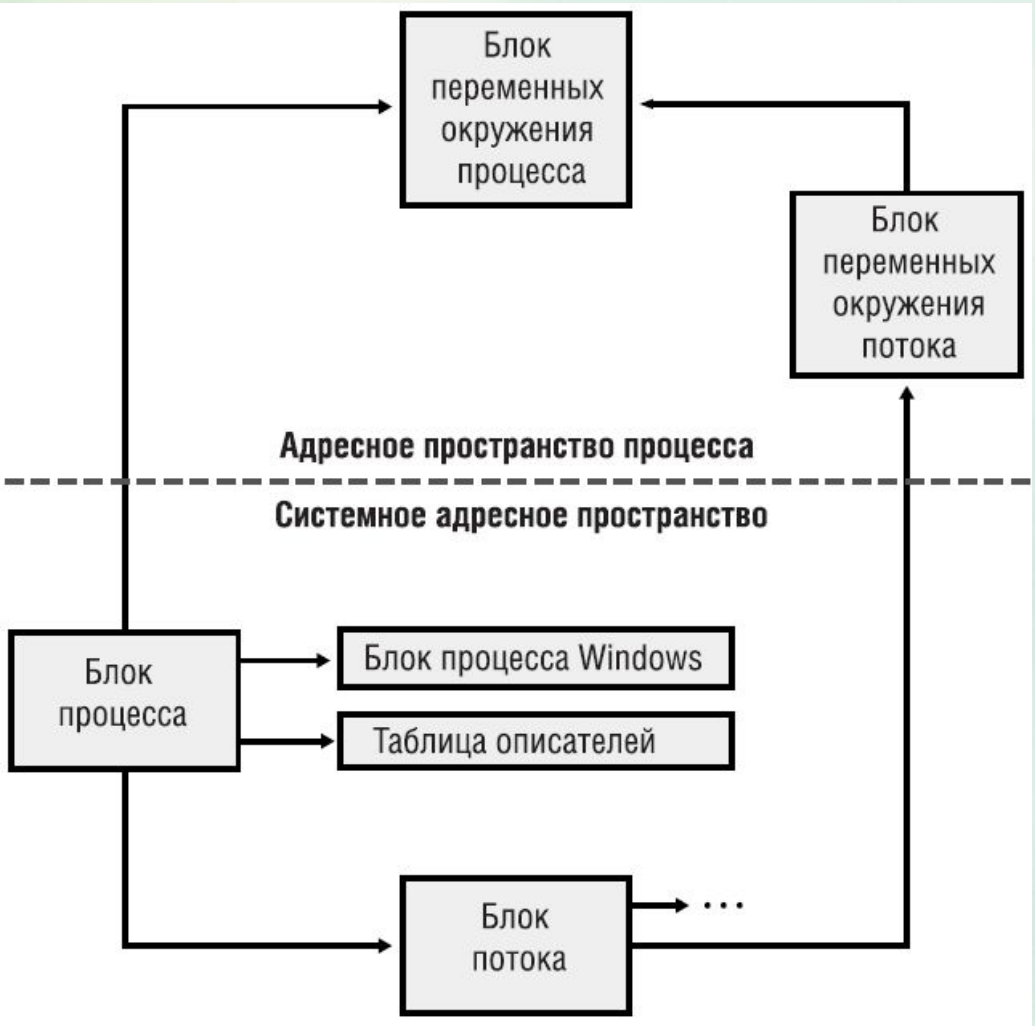
- Создание новых объектов, или открытие по имени уже существующих, приложение может осуществить при помощи Win32-функций, таких, как CreateFile, CreateSemaphore, OpenSemaphore и т.д.
- Это библиотечные процедуры, за которыми стоят сервисы Windows и методы объектов. В случае успешного выполнения создается 64-битный описатель в таблице описателей процесса в памяти ядра. На эту таблицу есть ссылка из блока управления процессом EPROCESS

Из 64-х разрядов описателя 29 разрядов используются для ссылки на блок памяти объекта ядра, 3 - для флагов, а оставшиеся 32 - в качестве маски прав доступа. Маска прав доступа формируется на этапе создания или открытия объекта, когда выполняется проверка разрешений. Таким образом, описатель объекта - принадлежность процесса, создавшего этот объект. По умолчанию он не может быть передан другому процессу. Тем не менее, система предоставляет возможность дублирования описателя и передачи его другому процессу специальным образом . Win32-функции, создающие объект, возвращают приложению не сам описатель, а индекс в таблице описателей, то есть малое число: типа 1,2 а не 64-разрядное. Впоследствии это значение передается одной из функций, которая принимает описатель объекта в качестве аргумента. Одной из таких функций является функция CloseHandle, задача которой - закрыть объект.

Структура ОС Windows

Процессы и потоки в Windows

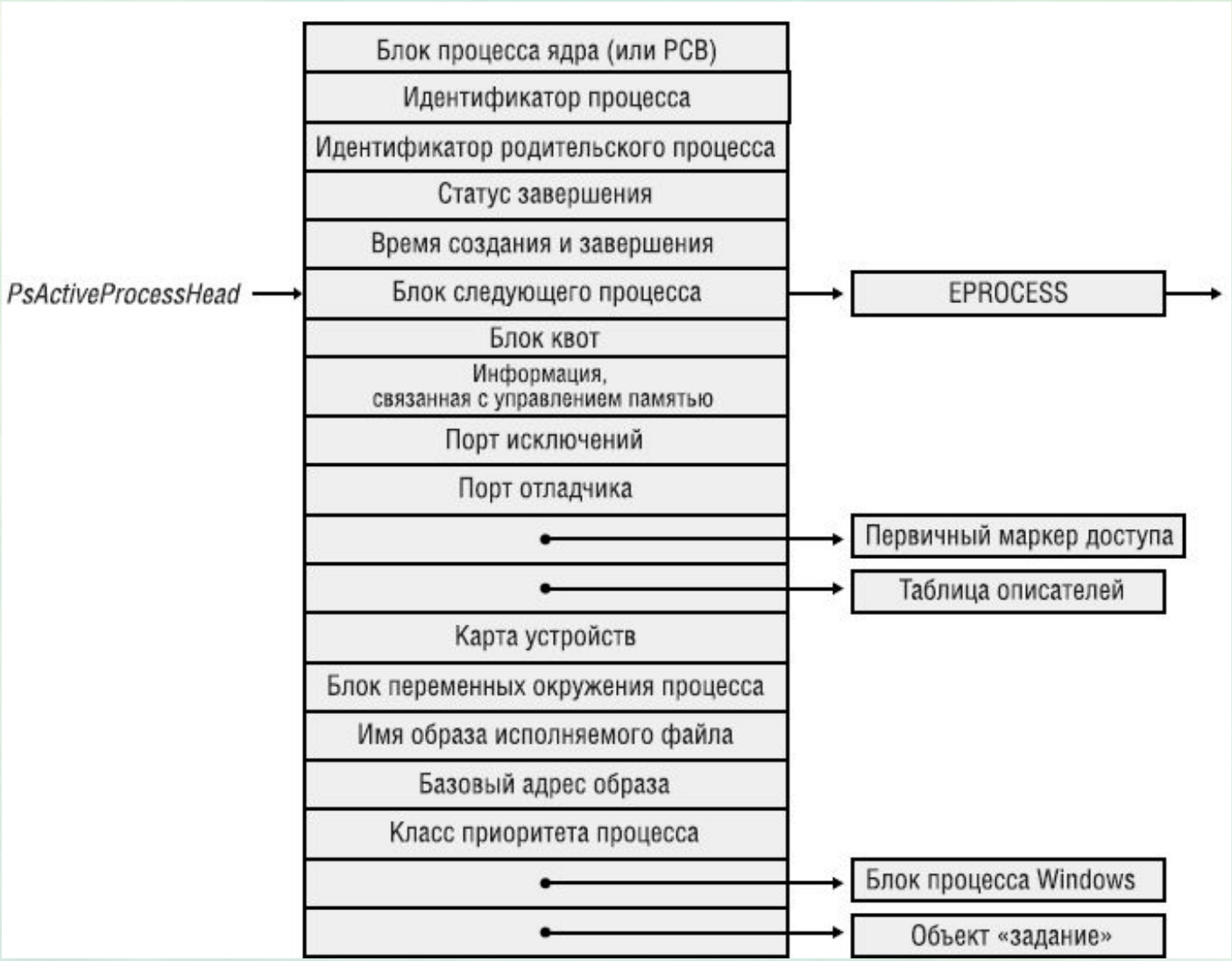
Каждый процесс в Windows представлен блоком процесса, создаваемым исполнительной системой (EPROCESS)



Для каждого процесса, выполняющего Windows-программу, процесс подсистемы Windows (Csrss) поддерживает в дополнение к блоку EPROCESS параллельную структуру данных. Кроме того, часть подсистемы Windows, работающая в режиме ядра (Win32k.sys), поддерживает структуру данных для каждого процесса, которая создается при первом вызове потоком любой функции USER или GDI, реализованной в режиме ядра.

Структура ОС Windows

Процессы и потоки в Windows



Блок процесса (EPROCESS), создаваемый исполнительной системой

Структура ОС Windows

Процессы и потоки в Windows

Блок KPROCESS, входящий в блок EPROCESS, и PEB(Блок переменных окружения), на который указывает EPROCESS, содержат дополнительные сведения об объекте «процесс».

Блок KPROCESS, иногда называют **блоком управления процессом** (process control block, PCB), Он содержит базовую информацию, нужную ядру Windows для планирования потоков.

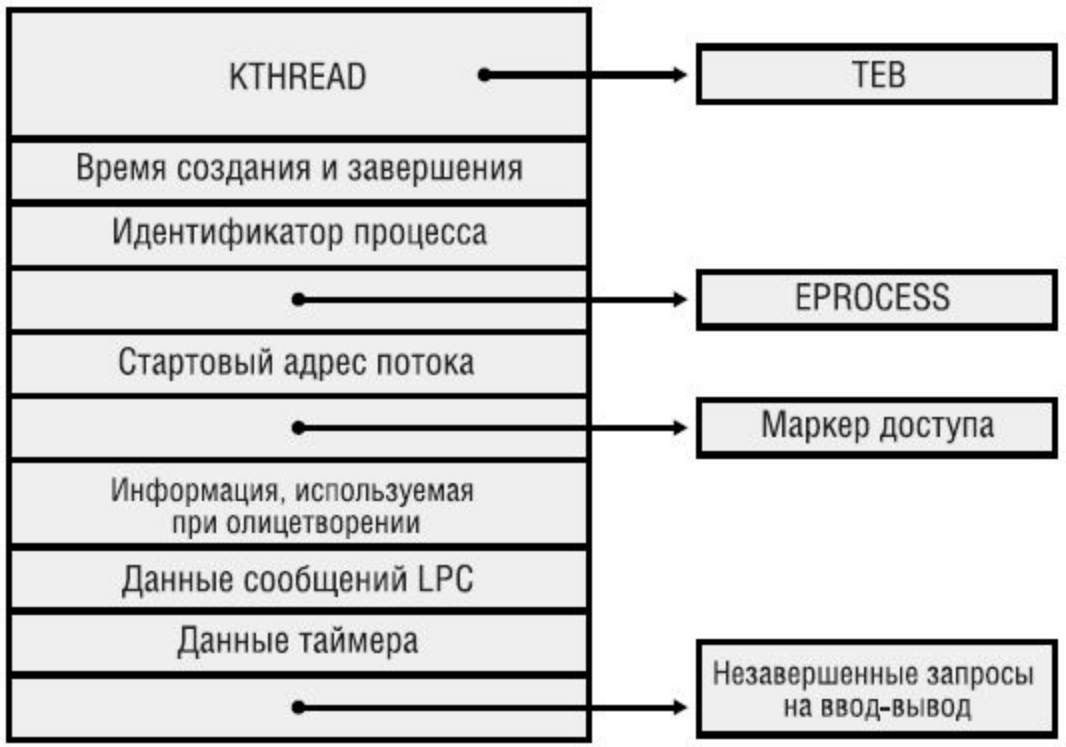
Блок процесса исполнительной системы

Структура ОС Windows

Процессы и потоки в Windows

На уровне операционной системы поток представляется блоком потока, принадлежащим исполнительной системе (ETHREAD).

Блок ETHREAD и все структуры данных, на которые он ссылается, существуют в системном адресном пространстве, кроме блока переменных окружения потока (thread environment block, TEB) — он размещается в адресном пространстве процесса.



Помимо этого, процесс подсистемы Windows (Csrss) поддерживает параллельную структуру для каждого потока, созданного в Windows-процессе. Часть подсистемы Windows, работающая в режиме ядра (Win32k.sys), также поддерживает для каждого потока, вызывавшего USER- или GDI-функцию, структуру W32THREAD, на которую указывает блок ETHREAD.

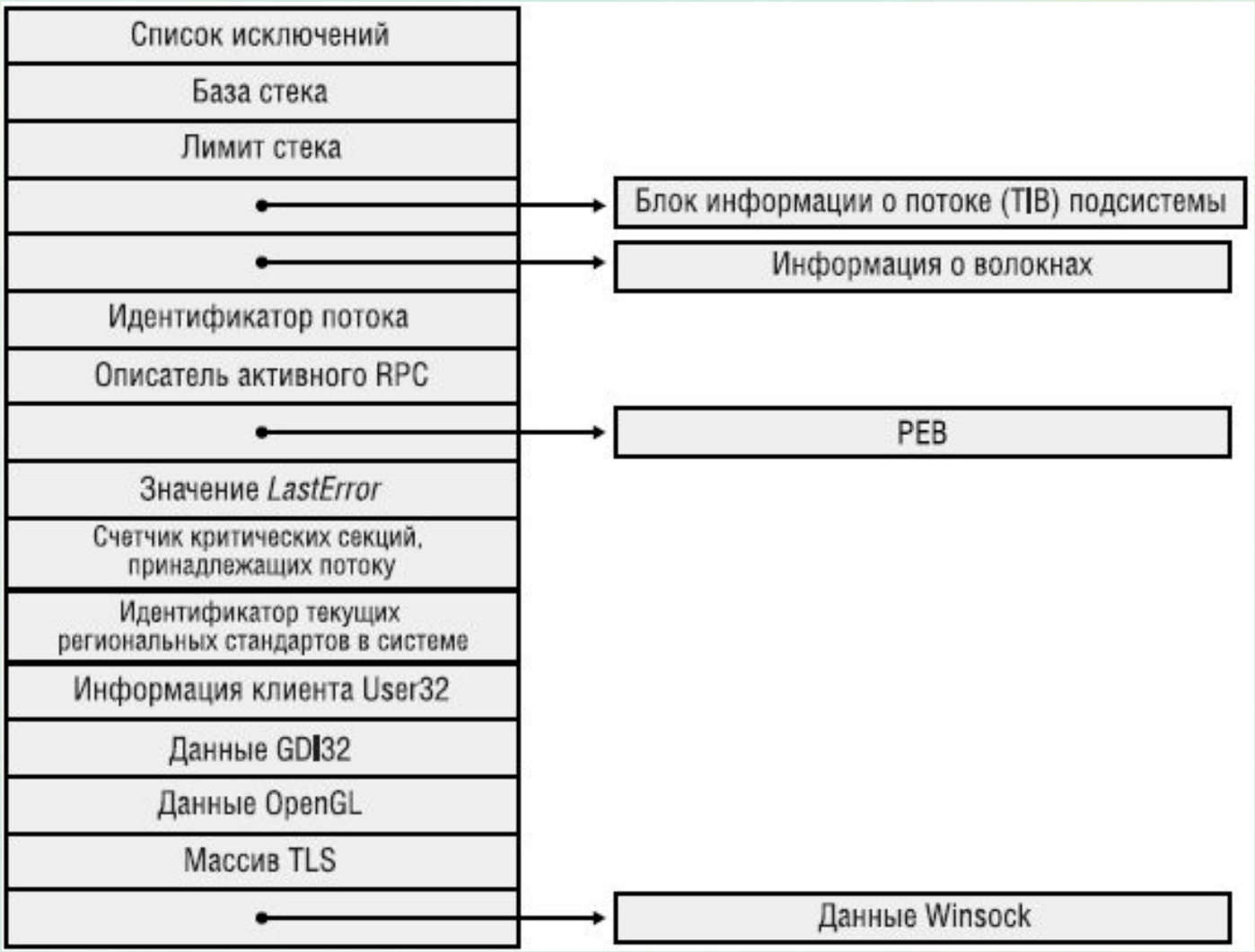
Структура ОС Windows

Процессы и потоки в Windows

Схема блока потока ядра

Структура ОС Windows

Процессы и потоки в Windows



Поля блока переменных окружения потока

Планирование потоков в Windows

В Windows реализована **подсистема вытесняющего планирования на основе уровней приоритета**, в которой всегда выполняется поток с наибольшим приоритетом, готовый к выполнению. Однако выбор потока для выполнения может быть ограничен набором процессоров, на которых он может работать. Это явление называется *привязкой к процессорам (processor affinity).*

Выбранный для выполнения поток работает в течение некоего периода, называемого *квантом.* По умолчанию в Windows 2000 Professional и Windows XP потоки выполняются в течение 2 интервалов таймера (clock intervals), а в системах Windows Server — 12. Длительность интервала таймера зависит от аппаратной платформы и определяется HAL, а не ядром. Например, этот интервал на большинстве однопроцессорных x86-систем составляет 10 мс, а на большинстве многопроцессорных x86-систем — около 15 мс.

Длительность квантов зависит от трех факторов: конфигурационных параметров системы (длинные или короткие кванты), статуса процесса (активный или фоновый) и использования объекта «задание» для изменения длительности квантов.

Код Windows, отвечающий за планирование, реализован в ядре. Поскольку этот код рассредоточен по ядру, единого модуля или процедуры с названием «планировщик» нет. Совокупность процедур, выполняющих эти обязанности, называется *диспетчером ядра (kernel's dispatcher)*

Структура ОС Windows

Планирование потоков в Windows

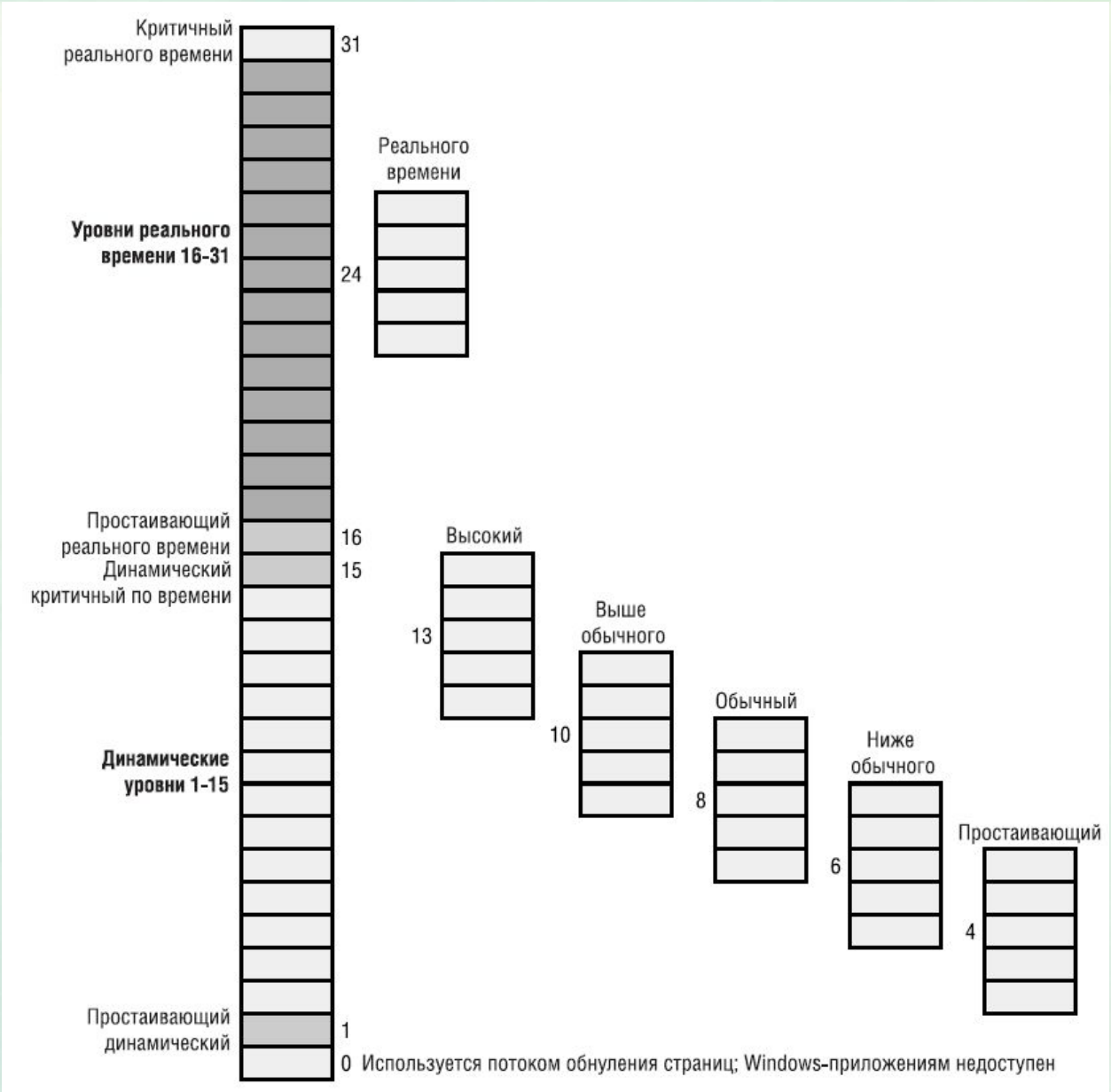
В Windows предусмотрено **32 уровня приоритета** — от 0 до 31. Эти значения группируются так:

- шестнадцать уровней реального времени (16–31);
- пятнадцать варьируемых (динамических) уровней (1–15);
- один системный уровень (0), зарезервированный для потока обнуления страниц (zero page thread).

Уровни приоритета потока назначаются с учетом двух разных точек зрения — Windows API и ядра Windows. Windows API сначала упорядочивает процессы по **классам приоритета**, назначенным при их создании [Real-time (реального времени), High (высокий), Above Normal (выше обычного), Normal (обычный), Below Normal (ниже обычного) и Idle (простаивающий)], а затем — по **относительному приоритету индивидуальных потоков** в рамках этих процессов [Time-critical (критичный по времени), Highest (наивысший), Above-normal (выше обычного), Normal (обычный), Below-normal (ниже обычного), Lowest (наименьший) и Idle (простаивающий)].

Структура ОС Windows

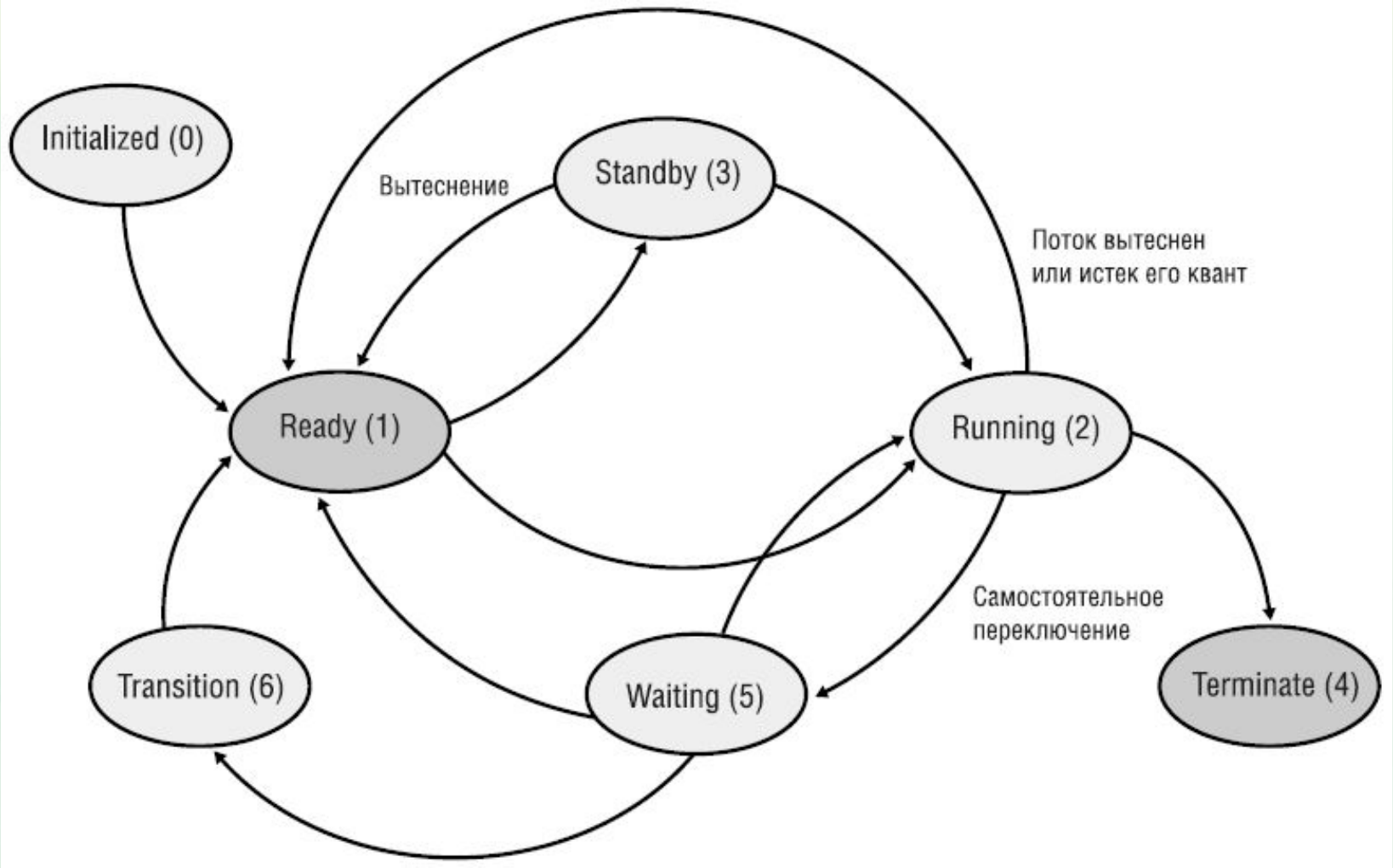
Планирование потоков в Windows



Структура ОС Windows

Планирование потоков в Windows

Состояния потоков



Управление памятью в Windows

По умолчанию виртуальный размер процесса в 32-разрядной Windows — 2 Гб.

Если образ помечен как поддерживающий большое адресное пространство и система загружается со специальным ключом, 32-разрядный процесс может занимать до 3 Гб в 32-разрядной Windows и до 4 Гб в 64-разрядной. Размер виртуального адресного пространства процесса в 64-разрядной Windows составляет 7152 Гб на платформе IA64 и 8192 Гб на платформе x64.

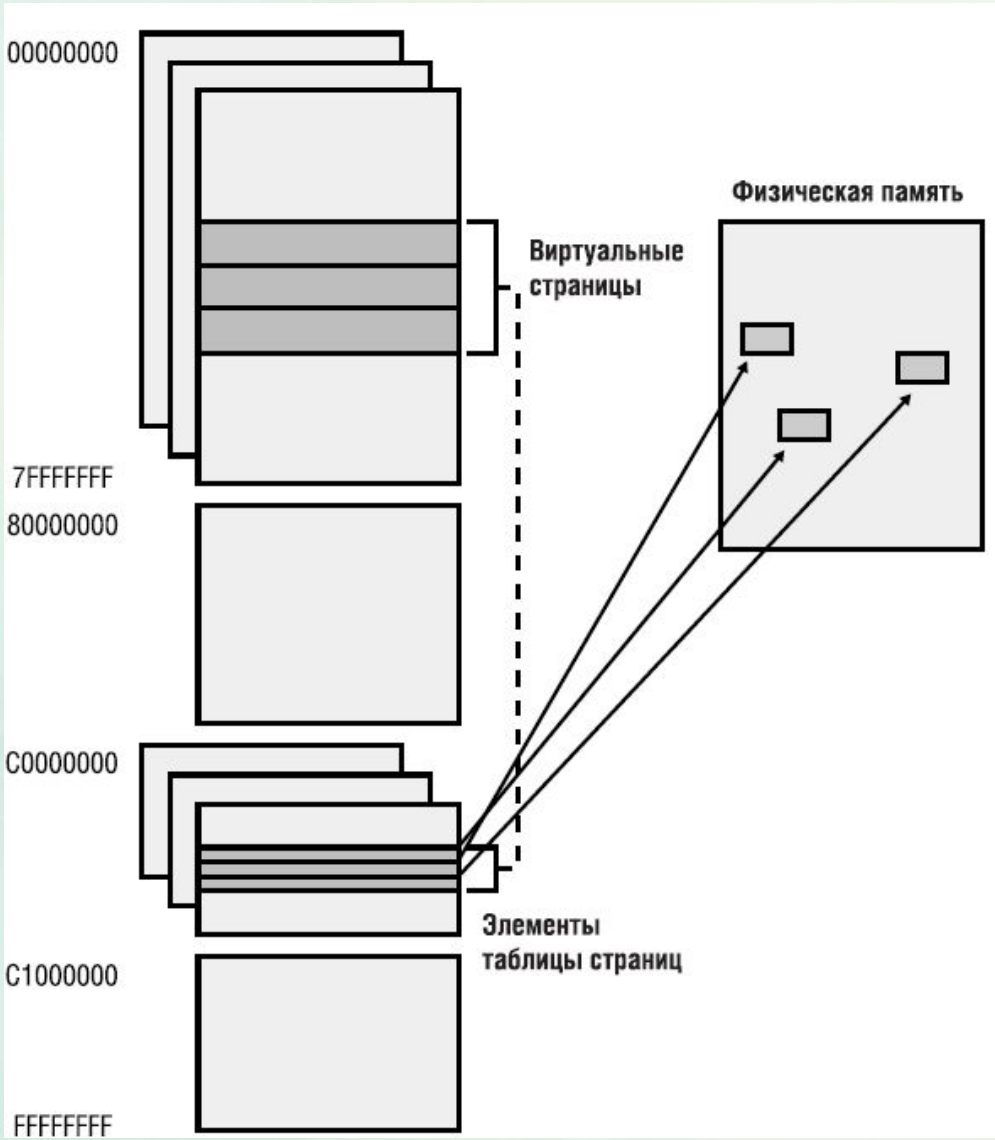
Сведения о виртуальном адресном пространстве процесса хранятся в *таблицах страниц (page tables).*

Таблицы страниц размещаются на страницах памяти, доступных только в режиме ядра, поэтому пользовательские потоки в процессе не могут модифицировать структуру адресного пространства своего процесса.

Структура ОС Windows

Управление памятью в Windows

С помощью структур данных (таблиц страниц), создаваемых и поддерживаемых диспетчером памяти, процессор транслирует виртуальные адреса в физические. Каждый виртуальный адрес сопоставлен со структурой системного пространства, которая называется *элементом таблицы страниц (page* table entry, PTE) и содержит физический адрес страницы, соответствующий виртуальному.



Структура ОС Windows

Управление памятью в Windows

По умолчанию в x86-системе Windows для трансляции виртуальных адресов в физические использует двухуровневую таблицу страниц

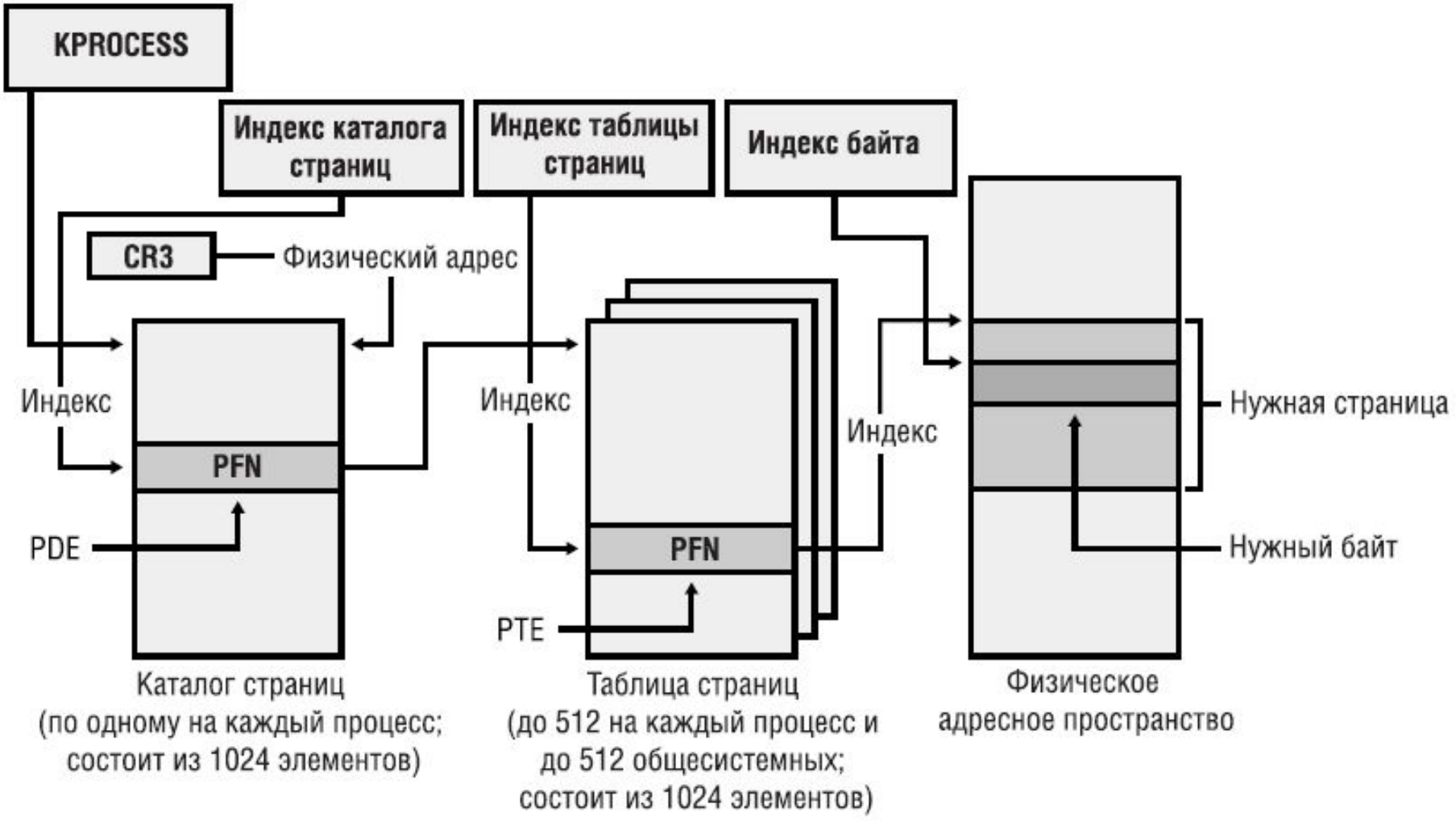
Индекс каталога страниц (page directory index) применяется для поиска таблицы страниц, содержащей PTE для данного виртуального адреса.

С помощью *индекса таблицы страниц (page table index)* осуществляется поиск *элемента таблицы страниц* PTE, который содержит физический адрес, на который проецируется виртуальная страница.

*Индекс байта (byte index) по*зволяет найти конкретный адрес на

Структура ОС Windows

Управление памятью в Windows



Структура ОС Windows

Управление памятью в Windows

При **трансляции виртуального адреса** выполняются следующие операции:

. **Аппаратные средства** управления памятью находят **каталог страниц текущего процесса**. При каждом переключении контекста процесса эти средства получают адрес каталога страниц нового процесса. Обычно операционная система записывает этот адрес в специальный регистр процессора.

.Индекс каталога страниц используется как указатель для поиска **элемента каталога страниц** (page directory entry, PDE), который определяет местонахождение **таблицы страниц**, нужной для трансляции виртуального адреса. PDE содержит **номер фрейма страницы** (page frame number, PFN) таблицы страниц (если она находится в памяти; однако такие таблицы могут выгружаться в страничный файл).

Структура ОС Windows

Управление памятью в Windows

3. Индекс таблицы страниц используется как указатель для поиска в ней PTE, который определяет местонахождение требуемой виртуальной страницы.

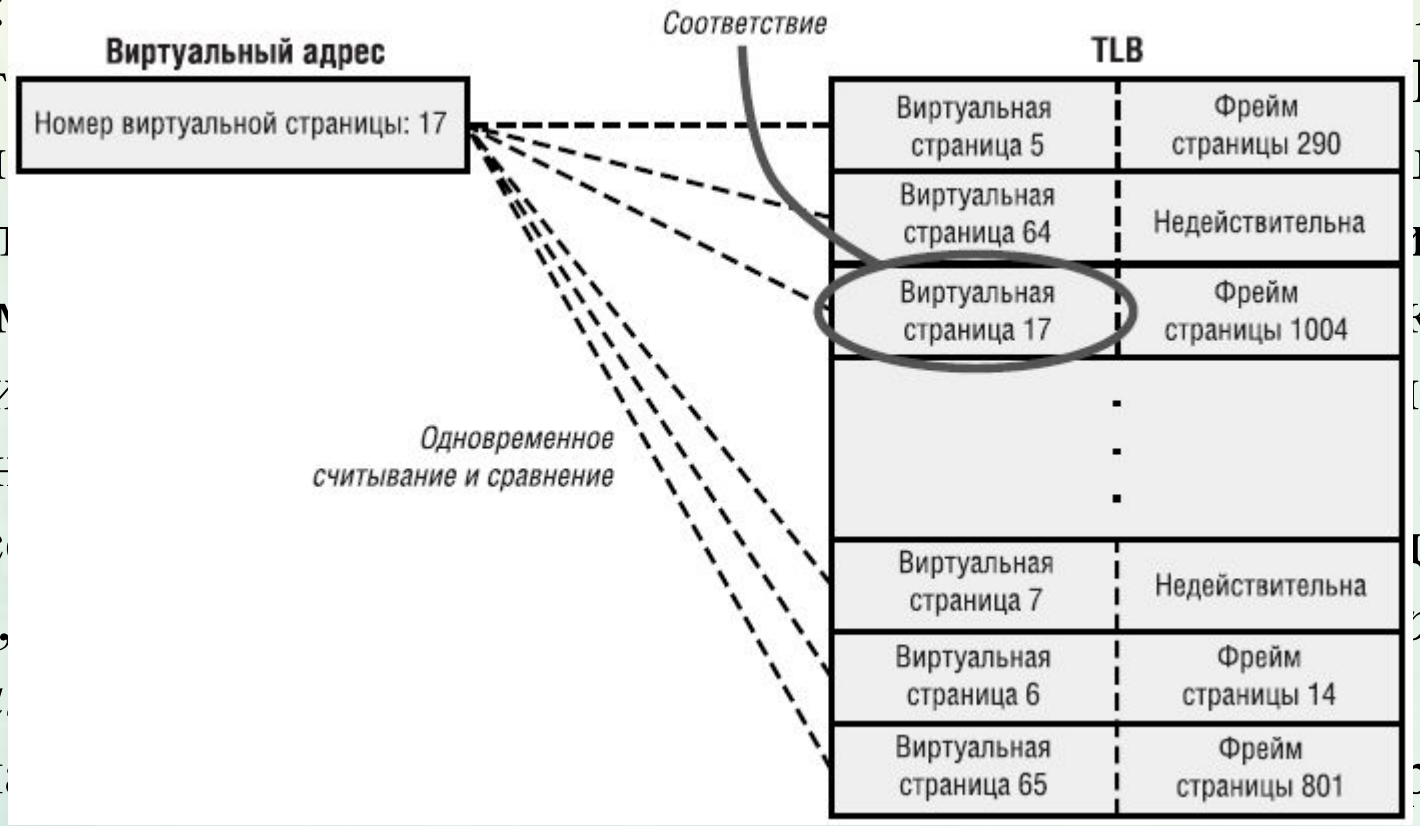
4. На основе PTE отыскивается страница. Если она действительна, то содержит PFN соответствующей страницы физической памяти. Если PTE сообщает, что страница недействительна, обработчик ошибок подсистемы управления памятью пытается найти страницу и сделать ее действительной. Если сделать страницу действительной не удалось (например, из-за ошибки защиты), обработчик ошибок генерирует нарушение доступа или вызывает переход в состояние отладки.

5. Если PTE указывает на действительную страницу, для поиска адреса нужных данных на физической странице используется индекс байта.

Структура ОС Windows

Управление памятью в Windows

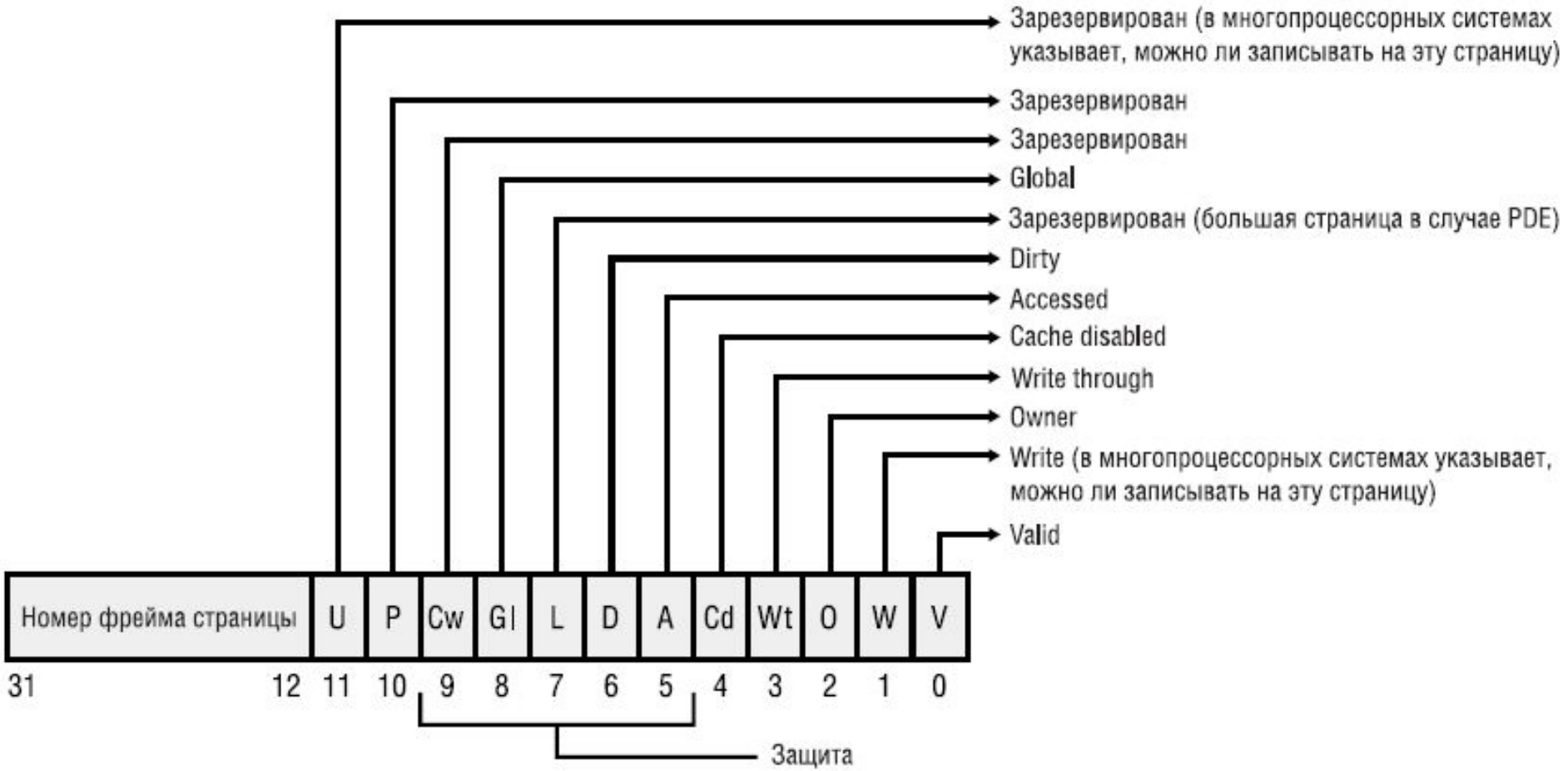
Таким образом, трансляция каждого адреса требует двух операций поиска: сначала нужно найти подходящие таблицы страниц в каталог Поскольку выполн цении по виртуал истемы до неприем кэшируют трансли мость в повторн отпадает. Процес циативной памяти, рансляции *(transla* Ассоци , ячейки которого можно считывать и сразу сравнивать с целевым значением.



Структура ОС Windows

Управление памятью в Windows

Элемент таблицы страниц



Структура ОС Windows

Управление памятью в Windows

Элемент таблицы страниц

Accessed - Была операция чтения с данной страницы
Cache disabled - Кэширование данной страницы отключено
Dirty - Страница модифицирована
Global - Трансляция относится ко всем процессам (например, сброс буфера трансляции не повлияет на этот PTE)
Large page - Указывает, что PDE относится к 4-мегабайтной странице
Owner - Указывает, доступна ли страница из кода пользовательского режима
Valid - Указывает, соответствует ли PTE странице в физической памяти
Write through - Отключает кэширование записи на данную страницу, в результате чего все измененные данные сразу же “сбрасываются” на диск
Write - В однопроцессорных системах указывает тип доступа (для чтения и записи или только для чтения), а в много процессорных системах определяет, доступна ли эта страница для записи

Структура ОС Windows

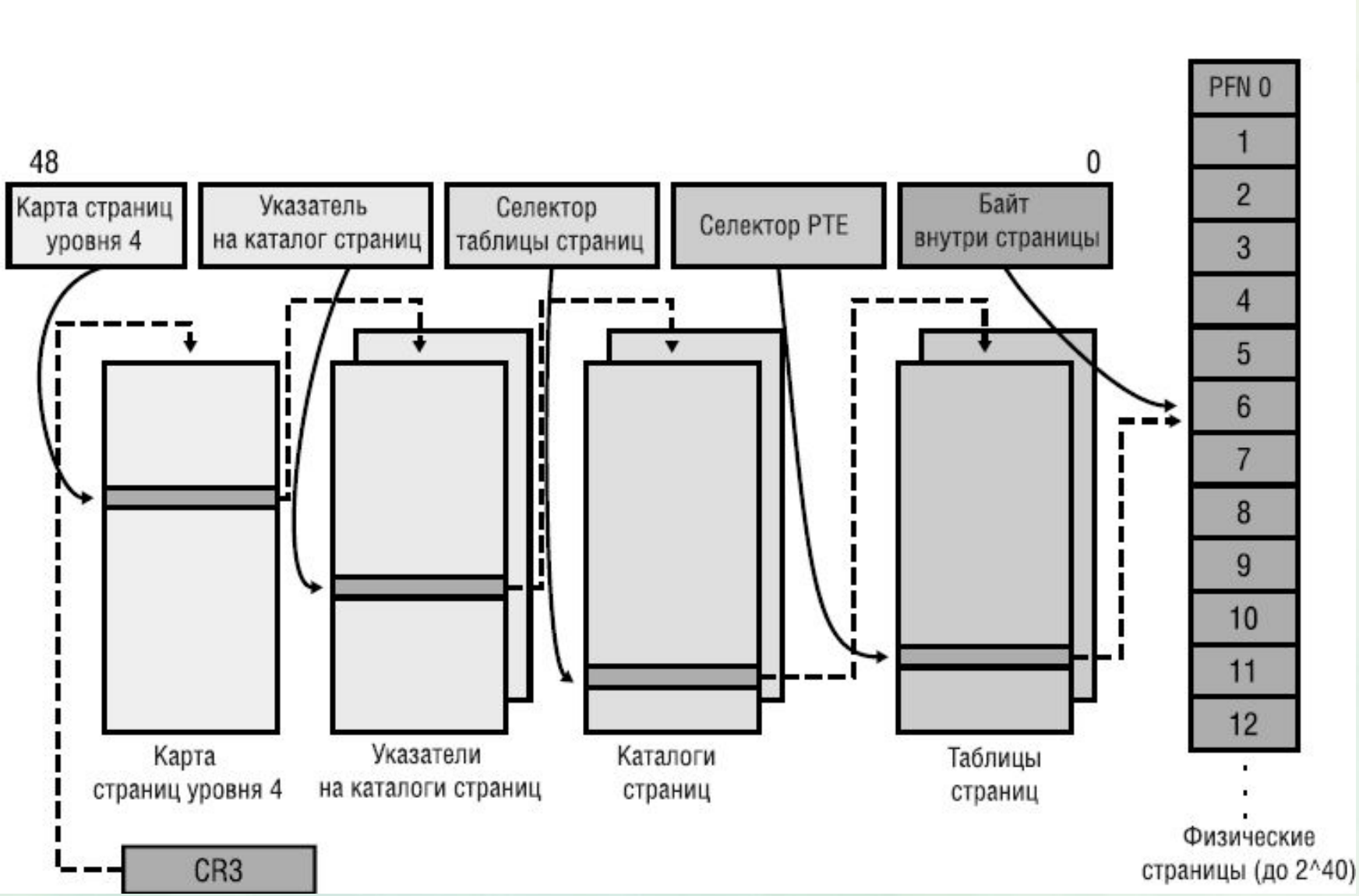
Управление памятью в Windows

Трансляция виртуальных адресов на платформе x64

64-разрядная Windows на платформе x64 применяет **четырехуровневую** схему таблиц страниц. У каждого процесса имеется расширенный каталог страниц верхнего уровня (называемый **картой страниц уровня 4**), содержащий 512 указателей на структуру третьего уровня — **родительский каталог страниц**. Каждый родительский каталог страниц хранит 512 указателей на **каталоги страниц второго уровня**, а те содержат по 512 указателей на **индивидуальные таблицы страниц**. Наконец, таблицы страниц (в каждой из которых 512 PTE) указывают на страницы в памяти. В реализациях архитектуры x64 размер виртуальных адресов ограничен 48 битами.

Структура ОС Windows

Управление памятью в Windows



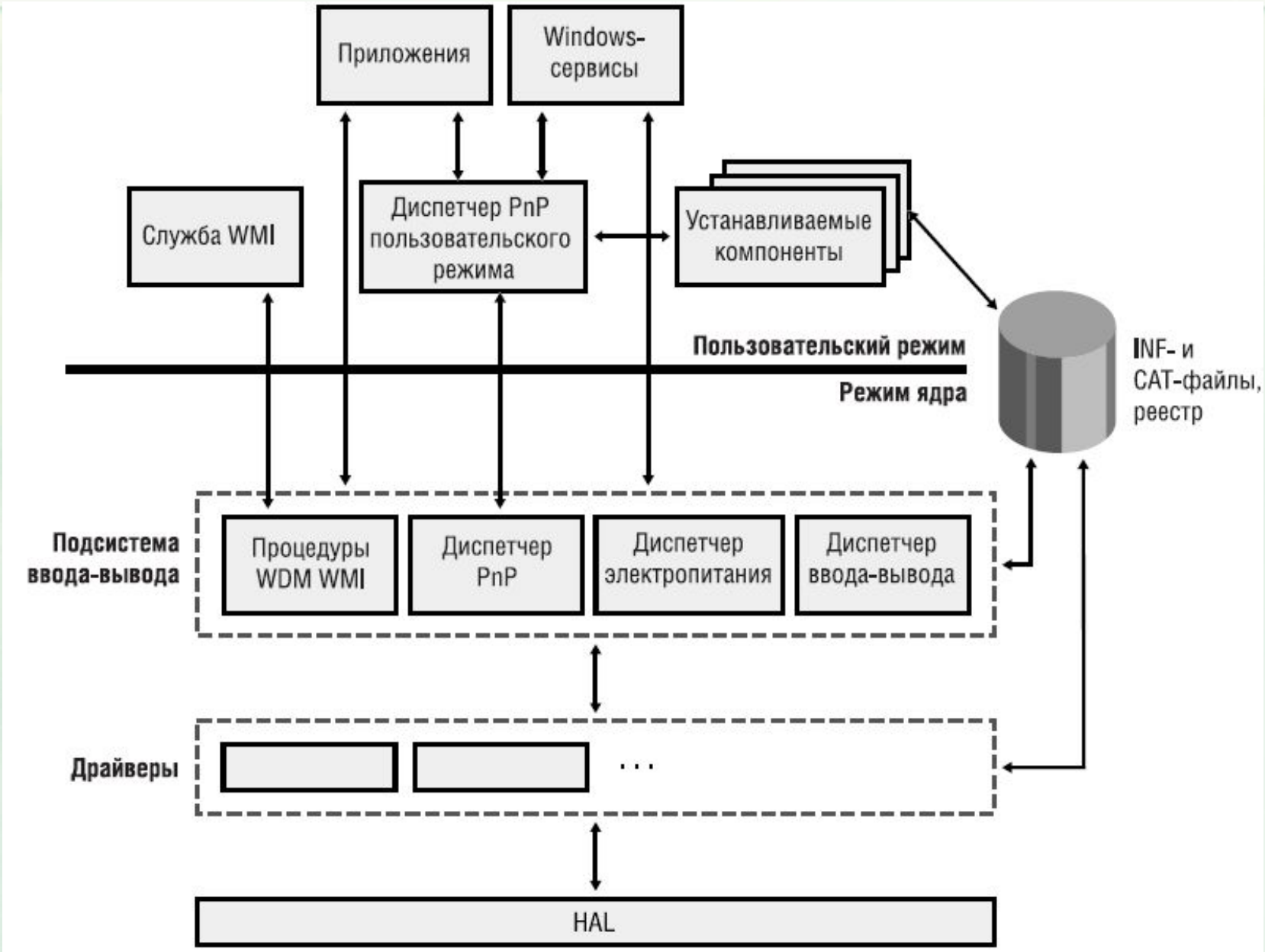
Подсистема ввода-вывода

Подсистема ввода-вывода в Windows состоит из нескольких компонентов исполнительной системы и драйверов устройств. Центральное место в этой подсистеме занимает **диспетчер ввода-вывода**.

Драйвер устройства, как правило, предоставляет интерфейс ввода-вывода для устройств конкретного типа. Такие драйверы принимают от диспетчера ввода-вывода команды, предназначенные управляемым ими устройствам, и уведомляют диспетчер ввода-вывода о выполнении этих команд. Драйверы часто используют этот диспетчер для пересылки команд ввода-вывода другим драйверам, задействованным в реализации интерфейса того же устройства и участвующим в управлении им.

Структура ОС Windows

Подсистема ввода-вывода



Структура ОС Windows

Подсистема ввода-вывода

Диспетчер PnP работает в тесном взаимодействии с диспетчером ввода вывода и *драйвером шины (bus driver) — одной из разновидностей* драйверов устройств. Он управляет выделением аппаратных ресурсов, а также распознает устройства и реагирует на их подключение или отключение.

Диспетчер электропитания, также в тесном взаимодействии с диспетчером ввода-вывода, управляет системой и драйверами устройств при их переходе в различные состояния энергопотребления.

Процедуры поддержки **Windows Management Instrumentation** (WMI) (Инструментарий управления Windows), образующие провайдер WDM (Windows Driver Model) WMI, позволяют драйверам устройств выступать в роли провайдеров, взаимодействуя со службой WMI пользовательского режима через провайдер WDM WMI.

Реестр Windows служит в качестве базы данных, в которой хранится описание основных устройств.

Структура ОС Windows

Подсистема ввода-вывода

Для установки драйверов используются **INF-файлы**; они связывают конкретное аппаратное устройство с драйвером, который берет на себя ведущую роль в управлении этим устройством. Содержимое INF-файла состоит из инструкций, описывающих соответствующее устройство, исходное и целевое местонахождение файлов драйвера, изменения, которые нужно внести в реестр при установке драйвера, и информацию о зависимостях драйвера. В CATфайлах хранятся цифровые подписи, которые удостоверяют файлы драйверов, прошедших испытания в лаборатории Microsoft Windows Hardware Quality Lab (WHQL).

Уровень абстрагирования от оборудования (HAL) изолирует драйверы от специфических особенностей конкретных процессоров и контроллеров прерываний, поддерживая API, скрывающие межплатформенные различия.
В сущности HAL является драйвером шины для тех устройств на материнской плате компьютера, которые не контролируются другими драйверами.

Структура ОС Windows

Подсистема ввода-вывода

Диспетчер ввода-вывода

*Диспетчер ввода-вывода (I/O manager) определяет модель доставки запро*сов на ввод-вывод драйверам устройств. Подсистема ввода-вывода управляется пакетами. Большинство запросов ввода-вывода представляется *пакетами запросов ввода-вывода (I/O request packets, IRP),* передаваемых от одного компонента подсистемы ввода-вывода другому. (исключением является *быстрый* ввод-вывод, при котором IRP не используются.) Подсистема ввода-вывода позволяет индивидуальному потоку приложения управлять сразу несколькими запросами на ввод-вывод. IRP — это структура данных, которая содержит информацию, полностью описывающую запрос ввода-вывода.

Структура ОС Windows

Подсистема ввода-вывода

Диспетчер ввода-вывода

Диспетчер ввода-вывода создает IRP (представляющий операцию ввода вывода), передает указатель на IRP соответствующему драйверу и удаляет пакет по завершении операции ввода-вывода. Драйвер, получивший IRP, выполняет указанную в пакете операцию и возвращает IRP диспетчеру ввода-вывода, чтобы тот либо завершил эту операцию, либо передал пакет другому драйверу для дальнейшей обработки.

Унифицированный модульный интерфейс драйверов позволяет диспетчеру ввода-вывода вызывать любой драйвер, ничего не зная о его структуре и внутреннем устройстве. Операционная система обрабатывает запросы на ввод-вывод так, будто они адресованы файлам; драйвер преобразует запросы к виртуальному файлу в запросы, специфичные для устройства. Драйверы также могут вызывать друг друга (через диспетчер ввода-вывода), обеспечивая многоуровневую независимую обработку запросов на ввод-вывод.

Структура ОС Windows

Подсистема ввода-вывода

Типичная обработка ввода-вывода

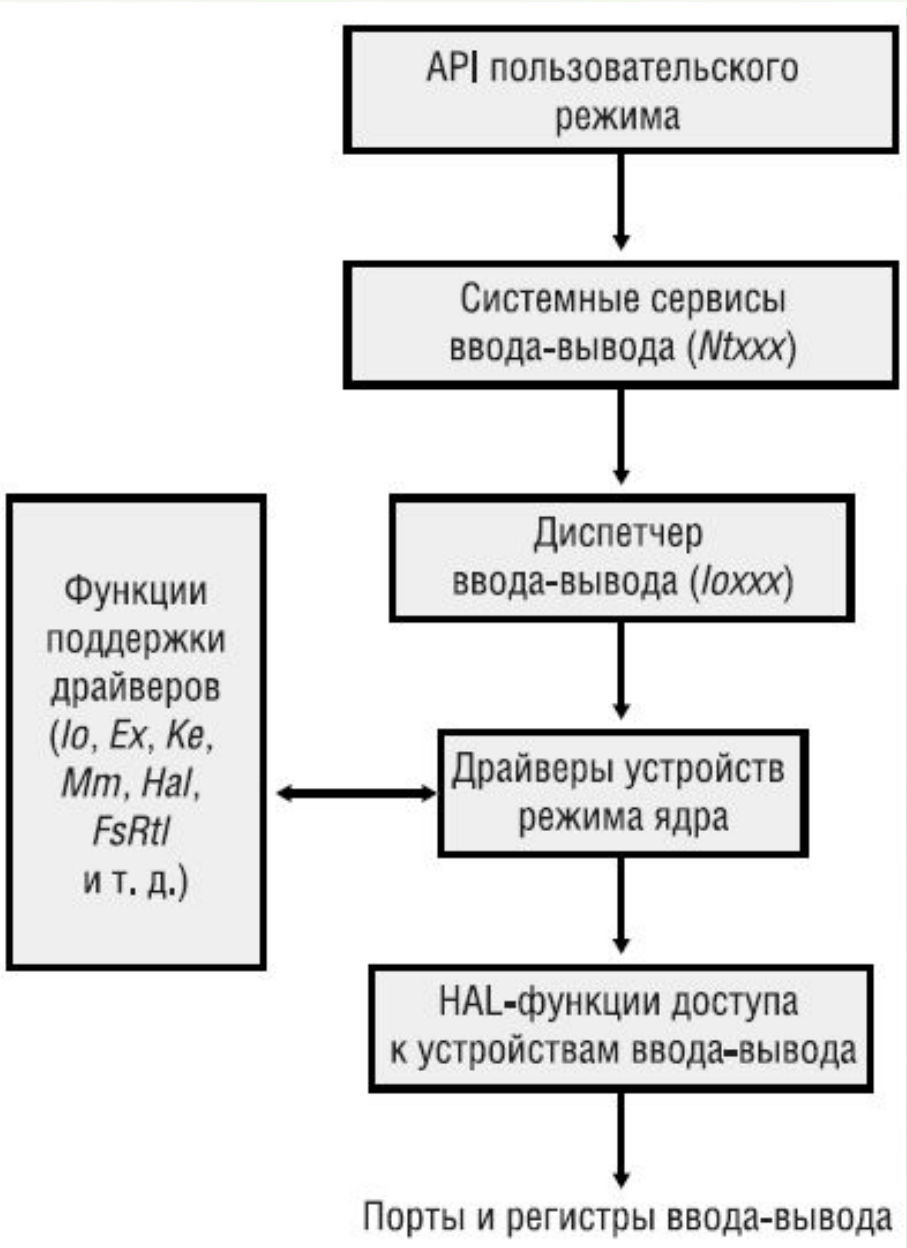
Операционная система абстрагирует все запросы на ввод-вывод, скрывая тот факт, что конечное устройство ввода-вывода может и не быть устройством с файловой структурой. Это позволяет обобщить интерфейс между приложениями и устройствами. Таким образом, виртуальный файл относится к любому источнику или приемнику ввода-вывода (файлу, каталогу, именованному каналу и почтовому ящику), который рассматривается как файл.

Все считываемые или записываемые данные представляются простыми потоками байтов, направляемыми в виртуальные файлы. Приложения пользовательского режима (к какой бы подсистеме они ни относились — Windows или POSIX) вызывают документированные функции, которые в свою очередь обращаются к внутренним функциям подсистемы ввода-вывода для чтения/записи файла и для выполнения других операций.

Структура ОС Windows

Подсистема ввода-вывода

Типичная обработка ввода-вывода



Структура ОС Windows

Подсистема ввода-вывода

Драйверы устройств

Для интеграции с диспетчером ввода-вывода и другими компонентами подсистемы ввода-вывода драйвер устройства должен быть написан в соответствии с правилами, специфичными для управляемого им типа устройств и для его роли в управлении такими устройствами.

Windows поддерживает множество типов драйверов устройств и сред их программирования. Среды программирования могут различаться даже для драйверов одного типа — в зависимости от типа устройства, для которого предназначен драйвер. Драйверы могут работать в двух режимах: в пользовательском или в режиме ядра.

Структура ОС Windows

Подсистема ввода-вывода

Драйверы устройств

Windows поддерживает несколько типов драйверов пользовательского режима:

- **Драйверы виртуальных устройств (virtual device drivers, VDD)** Используются для эмуляции 16разрядных программ MSDOS. Они перехватывают обращения таких программ к портам ввода-вывода и транслируют их в вызовы Windows-функций ввода-вывода, передаваемые реальным драйверам устройств.

- **Драйверы принтеров** Драйверы подсистемы Windows, которые транслируют аппаратно-независимые запросы на графические операции в команды, специфичные для принтера. Далее эти команды обычно направляются драйверу режима ядра, например драйверу параллельного порта(Parport.sys) или драйверу порта принтера на USBшине (Usbprint.sys).

Структура ОС Windows

Подсистема ввода-вывода

Драйверы устройств

Драйверы устройств, работающие в режиме ядра можно разбить на несколько основных категорий:

- **Драйверы файловой системы**

Принимают запросы на ввод-вывод и выполняют их, выдавая более специфические запросы драйверам устройств массовой памяти или сетевым драйверам.

- **PnP-драйверы**

Драйверы, работающие с оборудованием и интегрируемые с диспетчерами электропитания и PnP. В их число входят драйверы для устройств массовой памяти, видеоадаптеров, устройств ввода и сетевых адаптеров.

- **Драйверы, не отвечающие спецификации Plug and Play**

Также называются расширениями ядра. Расширяют функциональность системы, предоставляя доступ из пользовательского режима к сервисам и драйверам режима ядра. Они не интегрируются с диспетчерами PnP и электропитания.

Структура ОС Windows

Подсистема ввода-вывода

Драйверы устройств

Категория драйверов режима ядра подразделяется на группы в зависимости от модели, на которой они основаны, и их роли в обслуживании запросов к устройствам:

- **WDM-драйверы**

Это драйверы устройств, отвечающие спецификации Windows Driver Model (WDM). WDM требует от драйверов поддержки управления электропитанием, Plug and Play и WMI. Большинство драйверов Plug and Play построены как раз на модели WDM. Существует три типа WDMдрайверов:

1. Драйверы шин Управляют логическими или физическими шинами.

Примеры шин — PCMCIA, PCI, USB, IEEE 1394, ISA. Драйвер шины отвечает за распознавание устройств, подключенных к управляемой им шине, оповещение о них диспетчера PnP и управление параметрами электропитания шины.

Структура ОС Windows

Подсистема ввода-вывода

Драйверы устройств

2. **Функциональные драйверы** Управляют конкретным типом устройств. Драйверы шин представляют устройства функциональным драйверам через диспетчер PnP. Функциональным считается драйвер, экспортирующий рабочий интерфейс устройства операционной системе. Как правило, это драйвер, больше других знающий о функционировании определенного устройства.

3. **Драйверы фильтров** Занимают более высокий логический уровень, чем функциональные драйверы, они дополняют функциональность или изменяют поведение устройства, либо другого драйвера.

В WDM ни один драйвер не отвечает за все аспекты управления конкретным устройством. Драйвер шины определяет изменения в составе устройств на шине – и помогает диспетчеру PnP в перечислении устройств на шине, обращается к специфичным для шины регистрам и в некоторых случаях управляет электропитанием подключенных к шине устройств. К аппаратной части устройства обычно обращается только функциональный драйвер.

Структура ОС Windows

Подсистема ввода-вывода

Драйверы устройств

Многоуровневые драйверы

Поддержка индивидуального устройства часто распределяется между несколькими драйверами, каждый из которых обеспечивает часть функциональности, необходимой для нормальной работы устройства. Кроме WDM драйверов шин, функциональных драйверов и драйверов фильтров, оборудование могут поддерживать и следующие компоненты:

Драйверы классов устройств (class drivers)

Реализуют обработку ввода-вывода для конкретного класса устройств, например дисковых устройств, ленточных накопителей или приводов CDROM, где аппаратные интерфейсы стандартизированы и один драйвер может обслуживать аналогичные устройства от множества производителей.

Структура ОС Windows

Подсистема ввода-вывода

Драйверы устройств

Порт-драйверы (port drivers)

Обрабатывают запросы на ввод-вывод, специфичные для определенного типа порта ввода-вывода, например SCSI. Порт-драйверы реализуются как библиотеки функций режима ядра, а не как драйверы устройств.

Минипорт-драйверы (miniport drivers)

Преобразуют универсальные запросы ввода-вывода к порту конкретного типа в запросы, специфичные для адаптера конкретного типа, например для SCSI-адаптера.

Минипорт-драйверы являются истинными драйверами устройств, которые импортируют функции, предоставляемые порт-драйвером.

Структура ОС Windows

Подсистема ввода-вывода

Драйверы устройств