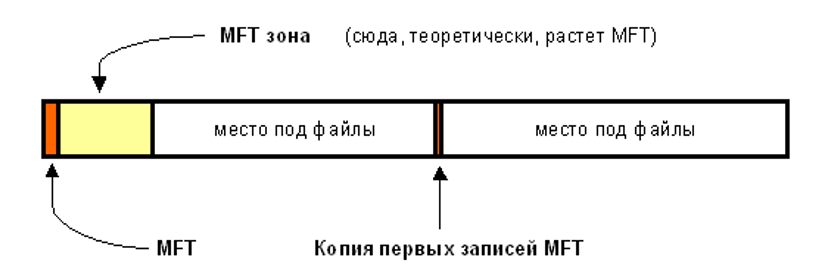
**43. Организация файловой системы NTFS(*new technology file system*).**



Диск NTFS условно делится на две части:

1. Первые 12% диска отводятся под так называемую MFT зону - пространство, в котором располагается и растет самый главный файл(метафайл), MFT(Master File Table - главная таблица файлов).
2. Остальные 88% диска представляют собой обычное пространство для хранения файлов.

Базовая единица распределения дискового пространства для NTFS: область кластеров, называемая отрезком.

B NTFS все данные, хранящиеся на томе, содержатся в файлах.

Главная таблица файлов MFT представляет собой файл (разделенный на записи (строки), обычно размером 1 Кб, 2 Кб, 4 Кб), в котором хранится информация обо всех файлах тома.

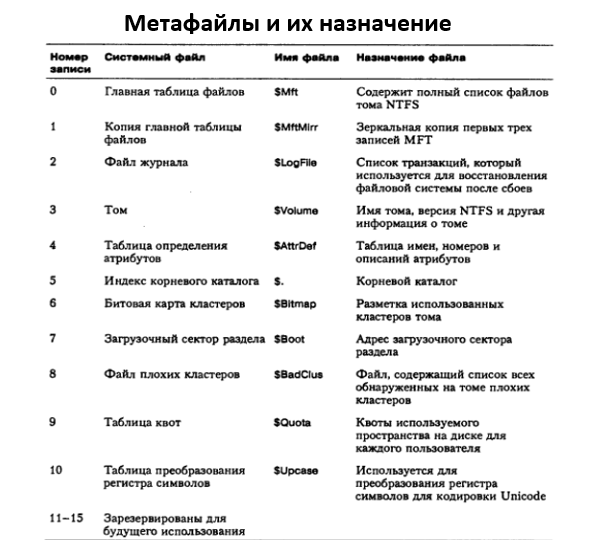
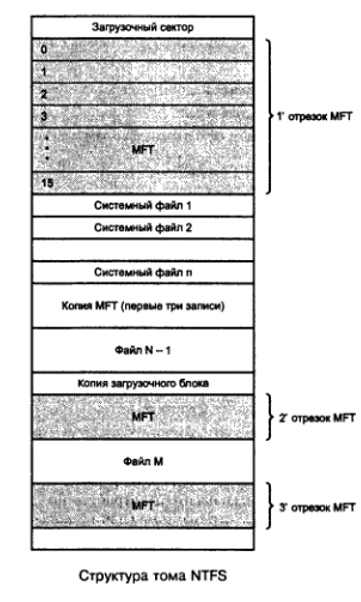
Первые 16 записей являются служебными(метафайлы).

Метафайлы находятся в корневом каталоге NTFS диска.

Самая первая запись хранит информацию о самом файле MTF.

Имена всех метафайлов NTFS начинаются со знака доллара ($), хотя эти знаки скрыты.

Остальные файлы NTFS-тома являются обычными файлами и каталогами.



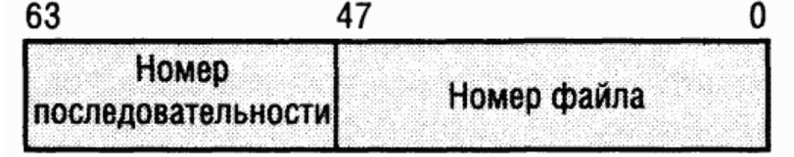
Каталог в NTFS представляет собой специальный файл, хранящий ссылки на другие файлы и каталоги.

Файл каталога разделен на блоки, каждый из которых содержит:

* имя файла
* ссылку на элемент MFT, который предоставляет полную информацию об элементе каталога

Корневой каталог диска ничем не отличается от обычных каталогов, кроме специальной ссылки на него из начала метафайла MFT.

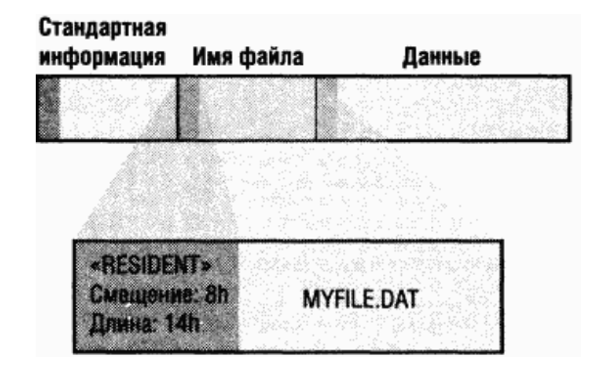
Когда NTFS впервые получает запрос на открытие файла, она начинает его поиск с записи каталога. Файловая ссылка имеет вид:



Номер файла равен позиции его записи в MFT.

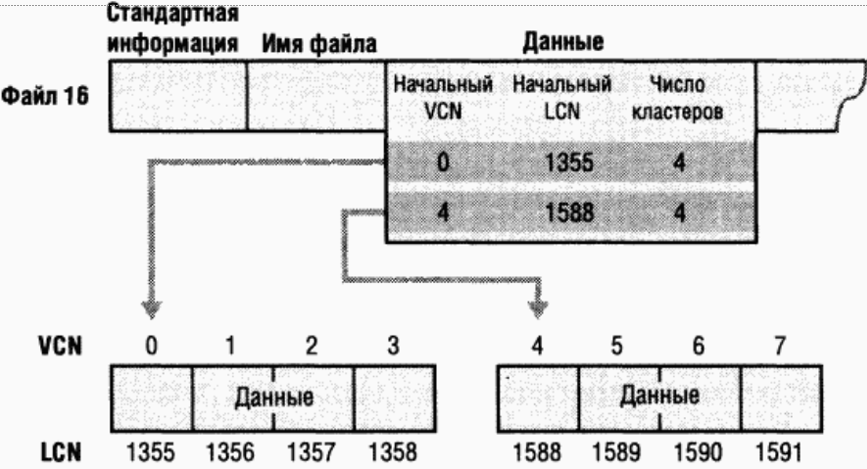
Номер последовательности позволяет проверять внутреннюю целостность файловой системы.

Хранение файла малого размера:



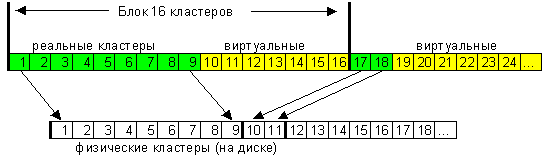
* Если файл невелик, атрибуты и данные умещаются в одной записи файла в MFT (1 Кбайт).
* Когда файл хранится непосредственно в записи MFT, он называется резидентным

Хранение файла большого размера:



Запись о файле большого размера хранит номера кластеров (отрезков) файла на диске.

Любой файл или каталог в индивидуальном порядке может хранится на диске в сжатом виде - этот процесс совершенно прозрачен для приложений.



Сжатие файлов имеет очень высокую скорость и только одно большое отрицательное свойство - огромная виртуальная фрагментация сжатых файлов.

Транзакция - действие, совершаемое целиком и корректно или не совершаемое вообще. У NTFS не бывает промежуточных (ошибочных или некорректных) состояний - квант изменения данных не может быть поделен на до и после сбоя, он либо совершен, либо отменен.

Перед совершением любой транзакции она записывается в журнал транзакции.

Журнал повышает надежность работы NTFS при сбоях.

Характеристики NTFS

//Считаю, что это можно не учить

* NTFS поддерживает размеры 512, 1024, 2048, 4096, 8192, 16КБ, 32КБ, 64K.
* Максимальный размер диска - 264 байт (16 экзабайт или 18 446 744 073 709 551 616 байт).
* Максимальный размер файла теоретически — 264 байт -16экзабайт)

**Сравнительный анализ FAT32 и NTFS(дополнительный пункт)**

Преимущества FAT32 перед NTFS:

* Значительная скорость доступа к файлам средних и малых размеров;
* Низкая требовательность к оперативному запоминающему устройству;
* Меньший износ жесткого диска.

К недостаткам файловой системы FAT32 можно отнести:

* Уязвимость и возможности сбоя системы;
* Медленные запросы при работе с большими каталогами файлов;
* Отсутствие поддержки малых кластеров;
* Необходимость фрагментации пространства на диске.

Достоинства NTFS являются:

* Рациональное использование места на носителе;
* Высокая производительность при работе с большими файлами;
* Значительная надежность;
* Поддержка сжатия;
* Восстановление системы при сбоях.

Недостатки:

* Высокая требовательность к объему оперативной памяти;
* Отсутствие доступа NTFS-томов в MS-DOS;
* Снижение производительности при работе с малыми объемами томов.

**44. Статическая и динамическая компоновка. Понятие DLL.**

Объектный модуль – модуль, полученный после компиляции.

Каждый скомпилированный объектный модуль создается компилятором со ссылками относительно начала своего объектного модуля.

Компоновщик - это программа, которая получает на входе набор объектных модулей, полученных после компиляции программы и формирует на их основе загружаемый модуль для последующей передачи его загрузчику для загрузки в память.

Компоновщик, объединяя модули, пересчитывает относительные адреса относительно начала единого модуля.

Компоновщик, который создает перемещаемый загрузочный модуль, называется редактором связей.

Компоновщик обычно входит в состав интегрированной среды разработки

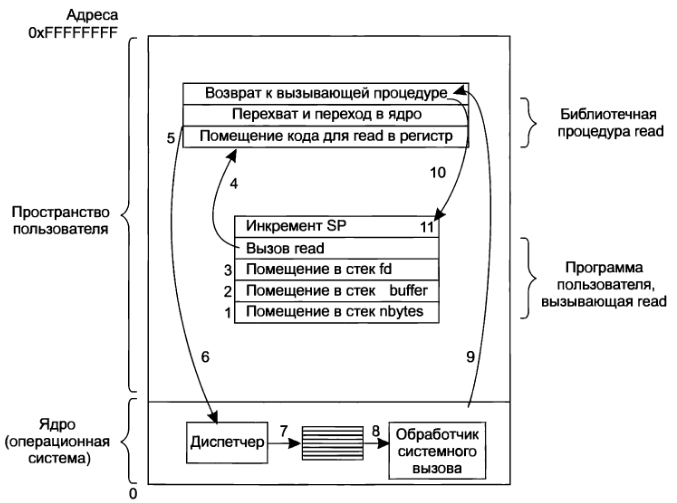
Компоновка зависит от типа создаваемого загрузочного модуля.

**Компоновка бывает статическая и динамическая.**

* Статическая компоновка - модуль подключается к программе на этапе загрузки процесса в память, помещается в адресное пространство процесса и находится в памяти на время выполнения процесса. При статической компоновке (подключении) всех модулей (библиотек) к каждой исполняемой программе значительно возрастает объём её загружаемого модуля. При этом разные программы могут содержать одинаковые модули, которые дублируются в памяти
* Динамически-загружаемая библиотека (DLL – dynamic link libraries, DLL) – двоичный модуль операционной системы. Это отдельно откомпилированный исполняемый код, который подключается к приложению динамически на этапе его выполнения. DLL загружается в память всякий раз, когда из процесса идет обращение к ней в момент и может быть выгружена после обращения.

**45. Понятие системного вызова**

Системный вызов: read(fd, buffer, nbytes)



1-3 Помещение передаваемых параметров системному вызову, а также адреса возврата (*адреса команды, следующей за системным вызовом*) в стек;

4 Помещение номера системного вызова read в один из регистров процессора (*это делает небольшая программа на языке ассемблера*);

5 Переход в ядро и вызов диспетчера процессов ОС.

6 Диспетчер процессов по номеру процесса из регистра процессора вызывает системную программу для выполнения функции read.

7-8 Модуль ОС используя, переданные через стек параметры, выполняет системный вызов read;

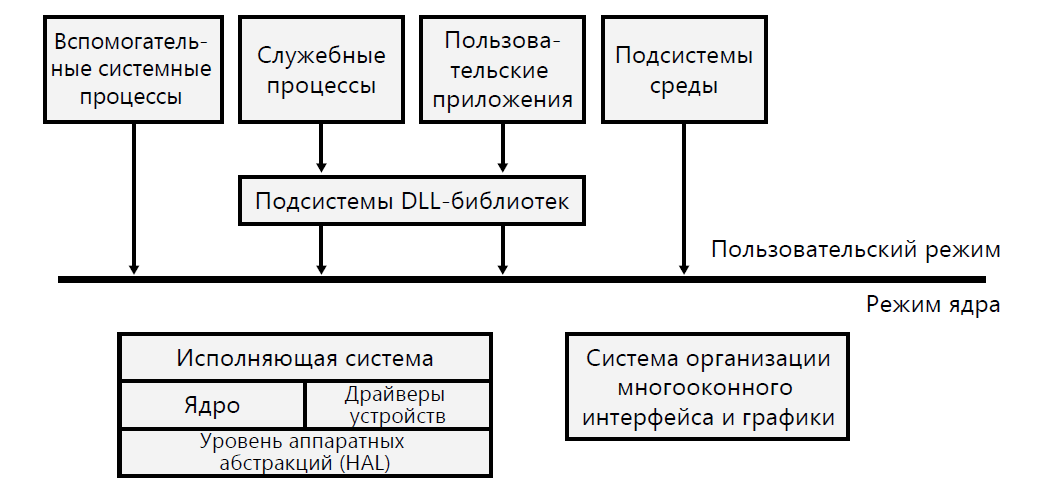
9 После окончания работы системного вызова, управление передается в пользовательский процесс, команде следующей за системным вызовом. Адрес команды был сохранен в стек преред началом системного вызова.

**46. Архитектура ОС Windows в режиме пользователя. Процессы пользовательского режима: вспомогательные системные, служебные процессы, приложения пользователей, подсистемы окружения). Выполнение вызовов Win32 API.**

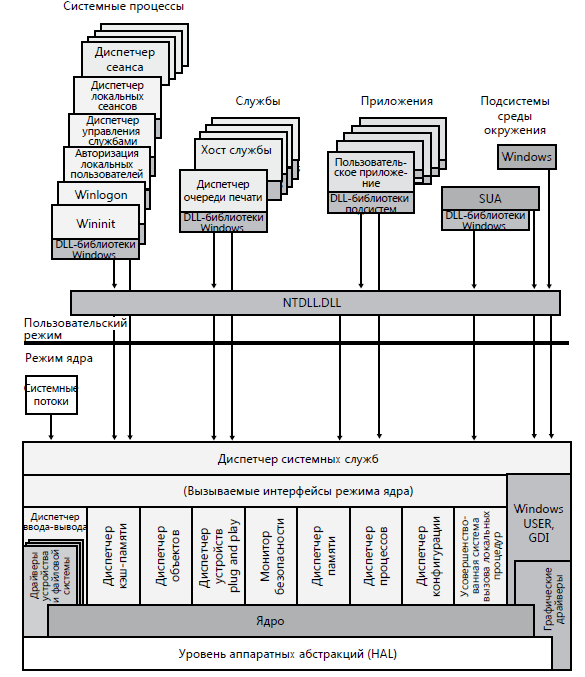
***Операционная система Windows:***

* является истинно 32 и 64‐разрядной, поддерживает вытесняющую многозадачность;
* работает на разных аппаратных архитектурах и обладает способностью к сравнительно легкому переносу на новые аппаратные архитектуры;
* поддерживает работу с виртуальной памятью;
* является распределенной вычислительной платформой, способной выступать в роли как клиента сети, так и сервера;
* Защищена от внутренних сбоев (у приложений нет возможности нарушить работу операционной системы или других приложений);
* Пользовательский интерфейс и API совместимы с предыдущими версиями Windows и MS‐DOS.\
* Под управлением Windows NT можно запускать приложения, написанные для операционной систем UNIX,
* Обладает высокой производительностью независимо от аппаратной платформы;
* Поддержка Unicode;
* Поддерживает многопоточность.

***Упрощенная архитектура***



***Уточненная архитектура Windows***



В пользовательском режиме работают ***четыре типа процессов***:

1. Вспомогательные системные процессы, такие как процесс входа в систему и администратор сеансов — Session Manager, которые не входят в службы Windows (они не запускаются диспетчером управления службами).
2. Служебные процессы, реализующие такие службы Windows, как Планировщик задач (Task Scheduler) и спулер печати (Print Spooler). Как правило, от служб требуется, чтобы они работали независимо от входов пользователей в систему.
3. Пользовательские приложения, которые могут относиться к одному из следующих типов: для 32- или 64-разрядной версии Windows, для 16-разрядной версии MS-DOS (включая Windows7 - 32), для 32- или 64-разрядной версии POSIX.

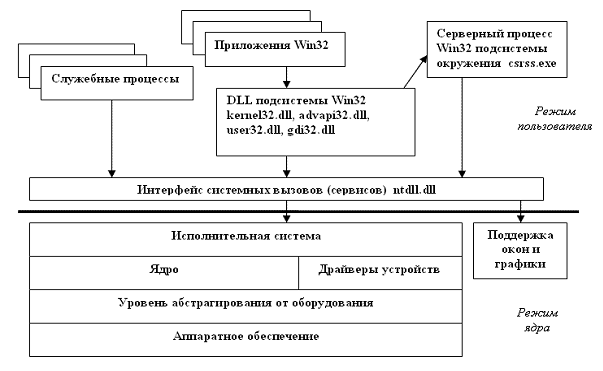
POSIX (portable operating system interface — переносимый интерфейс операционных систем) — набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой (системный API), библиотеку языка C и набор приложений. POSIX – для стандартизации. Приложения написанные по стандарту POSIX могут работать на различных операционных системах.

1. Процессы подсистемы окружения **–** позволяют под управлением Windows NT запускать приложения, написанные для различных операционных систем, с использованием соответствующих API (Win32, POSIX)

//без этого пункта будет непонятна суть вопроса

При выполнении под управлением Windows пользовательские приложения не вызывают имеющиеся в операционной системе Windows системные службы напрямую, а проходят через одну или несколько подсистем динамически подключаемых библиотек.

***Подсистемы DLL-библиотек*** предназначены для перевода документированной функции в соответствующий внутренний (и зачастую недокументированный) вызов системной службы ядра.

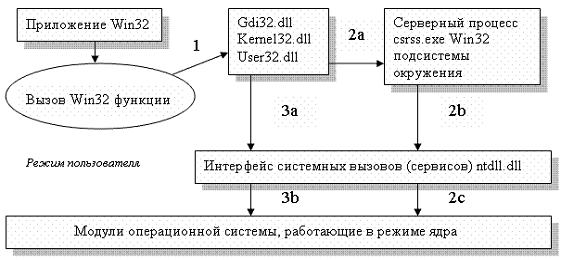


***Win32-API***

Взаимодействие между приложением и операционной системой осуществляется при помощи системных вызовов. Однако приложение не может вызвать системный вызов. Вместо этого приложение должно воспользоваться программным интерфейсом ОС - Win32 API. ***Win32 API****(Application Programming Interface)*- основной интерфейс программирования в семействе операционных систем Microsoft Windows. Функции Win32 API , например,*CreateProcess* или *CreateFile*, - документированные, вызываемые подпрограммы, реализуемые Win32 подсистемой.

В состав Win32 подсистемы входят: cерверный процесс подсистемы окружения csrss.exe, драйвер режима ядра Win32k.sys, dll - модули подсистем (kernel32.dll, advapi32.dll, user32.dll и gdi32.dll), экспортирующие Win32-функции и драйверы графических устройств.

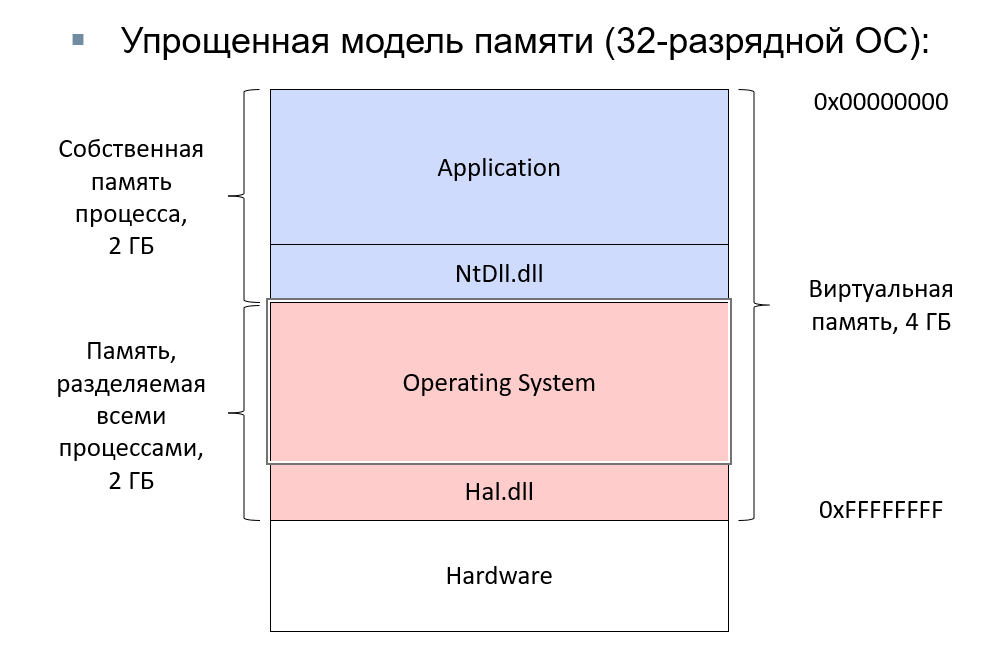
При ***вызове приложением Win32-функции*** может возникнуть одна из трех ситуаций:

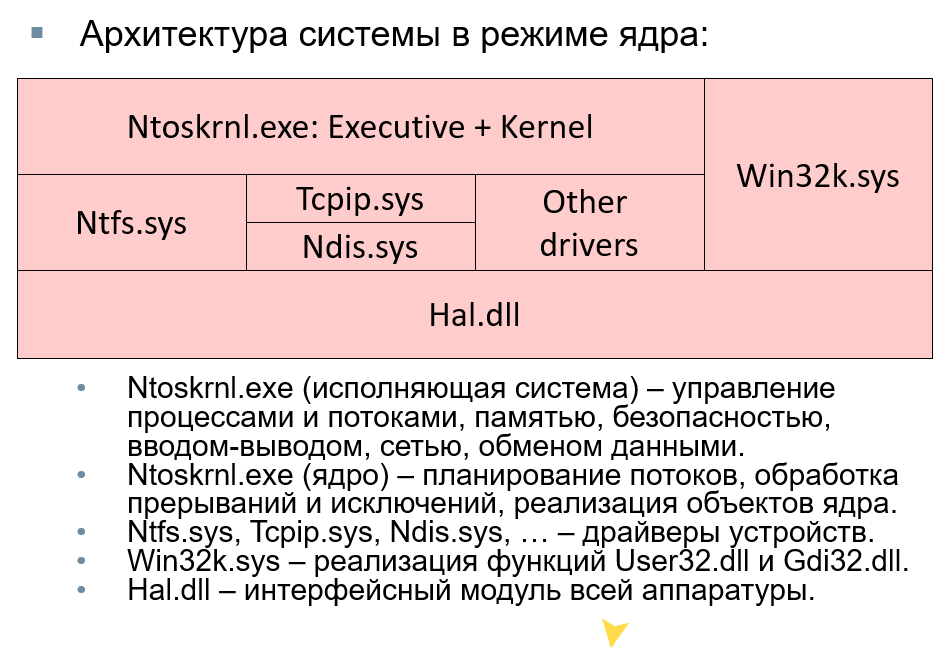


1. Функция полностью выполняется внутри одной из dll подсистем (шаг 1).
2. Для выполнения функции привлекается сервер csrss, для чего ему посылается сообщение (шаг 2a, за которым обычно следуют шаги 2b и 2c).
3. Данный вызов направляется в модуль ntdll.dll, который делает вызов системного сервиса (шаги 3a и 3b). Например, для выполнеия Win32-функции ReadFile делается вызов из NtDll.dll функции NtReadFile. Последняя в свою очередь обращается к исполнительной системе.

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*. Некоторые функции (например,CreateProcess) требуют выполнения обоих последних пунктов. 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, вызовы Win32 API, как правило, идут напрямую по 3-му (3a, 3b) пути, минуя подсистему окружения Win32. В настоящее время лишь небольшое число вызовов выполняется по длинному 2-му маршруту. Помимо перечисленных, наиболее важных dll-библиотек, в системном каталоге system32 имеется большое количество других dll-файлов. В настоящее время количество вызовов API составляет несколько десятков тысяч.

**47. Архитектура ОС Windows в режиме ядра (процессы уровня ядра: исполнительная система, система организации оконного интерфейса, ядро, HAL).**

****

****

Исполняющая система Windows содержит основные службы операционной системы, такие как управление памятью, управление процессами и потоками, безопасность, ввод-вывод, сеть и связь между процессами.

• Диспетчер процессов создает процессы и потоки и завершает их работу. Исходная поддержка процессов и потоков реализована в ядре Windows;

• Монитор безопасности обеспечивает соблюдение политики безопасности на локальном компьютере;

• Диспетчер ввода-вывода реализует аппаратно-независимый ввод-вывод и отвечает за направление запросов ввода – вывода на соответствующие драйверы устройств для дальнейшей обработки;

• Диспетчер устройств определяет, какие драйверы требуются для поддержки конкретного устройства, и загружает эти драйверы;

• Диспетчер электропитания согласовывает события электропитания и генерирует уведомления ввода-вывода, касающиеся управления электропитания, посылаемые драйверам устройств. При простое системы диспетчер электропитания может быть настроен на снижение расхода электроэнергии путем перевода центрального процессора в спящий режим.

Ядро Windows

Корпорация Microsoft называет ядром компонент, находящийся в невыгружаемой памяти и содержащий низкоуровневые функции операционной системы, такие, как:

* Поддержку уровня HAL (диспетчеризация прерываний и

исключений и т.д.);

* Управление процессором(и) – диспетчеризация и планирование

потоков;

* Для хранения специфических для процессора данных ядром

используется структура данных, называемая областью, относящейся к

управлению процессором, или KPCR. KPCR содержит основную

информацию, такую как процессорная таблица диспетчеризации

прерываний, сегмент состояния задачи и таблица глобальных

дескрипторов.

Уровень аппаратных абстракций (hardware abstraction layer, HAL), являющийся уровнем кода, который изолирует ядро, драйверы устройств и остальную исполняющую систему Windows от аппаратных различий конкретных платформ (Intel, AMD). HAL позволяет скрыть от всех программных компонентов особенности аппаратной платформы (например, различия между материнскими платами), на которой установлена операционная система.

Система организации оконного интерфейса

Оконный интерфейс — способ организации полноэкранного интерфейса программы, в котором каждая интегральная часть располагается в окне — собственном суб-экранном пространстве, находящемся в произвольном месте «над» основным экраном. Несколько окон, одновременно располагающихся на экране, могут перекрываться, находясь «выше» или «ниже» друг относительно друга.

Несмотря на то, что наиболее естественным для оконного интерфейса является графический режим, основные его элементы применимы и в текстовом режиме, где он применяется в равной степени. Окно обычно имеет прямоугольную форму, с обрамлением и/или цветом фона, отличным от цвета основного экрана. Модальные окна «монополизируют» фокус пользовательского внимания: работу с программой можно продолжить лишь после закрытия «модального» окна.

Оконный интерфейс очень быстро завоевал популярность и в настоящее время является самым популярным видом программного интерфейса.