Оглавление

1 Архитектура системы 1
1.1 Компоненты пользовательского режима 1
1.1.1 Процессы пользовательского режима: 2
1.2 Компоненты, работающие в режиме ядра 3
1.2.1 Исполняющая система 3
1.2.2 *Ядро* Windows 4
1.2.3 Драйверы устройств 4
1.3 Уровень аппаратных абстракций (hardware abstraction layer, HAL) 5
1.4 Система организации многооконного интерфейса и графики 5
1.5 Различия между клиентскими и серверными версиями 5
2 Объектная модель Windows 6
2.1 Понятие объекта 6
2.2 Структура объекта 8
2.2.1 Типовые объекты 8
2.2.2 Полная структура объекта 8
2.2.3 Имена объектов и каталогов 9
2.2.4 Структура объекта-каталога. 9
2.2.5 Дескрипторы объектов 10
2.2.6 Методы объекта 10
2.2.7 Защита объектов 11

1 Архитектура системы

1.1 Компоненты пользовательского режима

Приложения взаимодействуют с исполнительной системой опосредованно, через **защищенные подсистемы среды**, которые реализуют интерфейсы прикладного программирования.

Защищенные подсистемы среды могут взаимодействовать с клиентскими приложениями

- либо по принципу **клиент-сервер**,
- либо функционировать как **совместно используемые библиотеки**, связываемые с клиентскими приложениями во время их

компоновки.

На практике часто используется сочетание этих двух механизмов.

Благодаря такой организации Windows соединяет в себе достоинства микроядерной и расширяемой библиотечной архитектур.

Потоки пользовательского режима выполняются в защищенном адресном пространстве процесса (когда они выполняются в режиме ядра, у них есть доступ к системному адресному пространству).

Таким образом, у вспомогательных системных процессов, у процессов служб, у пользовательских приложений и у подсистем среды окружения,- у всех есть свое собственное закрытое адресное пространство.

1.1.1 Процессы пользовательского режима:

- Фиксированные (или реализованные на аппаратном уровне) ***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).
- ***Служебные процессы***

  реализующие такие службы Windows, как Диспетчер задач (Task Scheduler) и спулер печати (Print Spooler). Как правило, от служб требуется, чтобы они работали независимо от входов пользователей в систему.

  Многие серверные приложения Windows, такие как Microsoft SQL Server и Microsoft Exchange Server, также включают компоненты, работающие как службы.
- ***Пользовательские приложения***
- ***Серверные процессы*** *подсистемы окружения*, которые реализуют часть поддержки среды операционной системы или специализированную часть, представляемую пользователю и программисту.
- «**Подсистемы DLL-библиотек**». При выполнении под управлением Windows пользовательские приложения не вызывают имеющиеся в операционной системе Windows службы напрямую, а проходят через одну или несколько

подсистем **динамически подключаемых библиотек** (dynamic-link libraries, DLL). Подсистемы DLL-библиотек предназначены для перевода документированной функции в соответствующий внутренний (и зачастую недокументированный) вызов системной службы. Этот перевод может включать в себя (или не включать) отправку сообщения процессу подсистемы среды, обслуживающему пользовательское приложение.

1.2 Компоненты, работающие в режиме ядра

1.2.1 Исполняющая система

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

- **Диспетчер объектов.**

Создает и удаляет объекты исполнительной системы. Когда пользовательскому процессу требуется создать объект определенного типа, он вызывает соответствующую подсистему, а та направляет вызов менеджеру объектов. Последний реализует унифицированный механизм управления объектами и хранения соответствующих данных и используется всеми компонентами исполнительной подсистемы.

Дескрипторы объектов исполнительной системы Windows универсальные: с их помощью можно идентифицировать **процессы, потоки и другие объекты.**

- **Монитор безопасности**

Обеспечивает защиту объектов во время работы системы, участвует в операциях с объектами и может обеспечивать аудит их использования.

- **Диспетчер процессов**.

Отвечает за управление процессами и потоками. При создании процесса в его состав сразу включается один поток (единицей диспетчеризации является поток). Во время существования процесса в нем могут быть созданы дополнительные потоки. Процесс является единицей владения ресурсами.

- **Система вызова локальных процедур.**

(Local Procedure Call, LPC). Поддерживает взаимодействие между клиентами и серверами, расположенными на одном узле.

- **Менеджер памяти.**

Обеспечивает страничную организацию памяти и защиту адресного пространства каждого процесса (совместно используемого всеми его потоками).

- **Диспетчер ввода-вывода.**

Поддерживает независимые от устройств функции ввода-вывода, управляет файлами и сетевыми буферами.

1.2.2 *Ядро* Windows

состоит из низкоуровневых функций операционной системы, таких как диспетчеризация потоков, диспетчеризация прерываний и исключений и мультипроцессорная синхронизация. Оно также предоставляет набор подпрограмм и базовых объектов, используемых остальной исполняющей системой для реализации высокоуровневых конструктивных элементов.

Ядро осуществляет

- **управление процессами,**
- **управление памятью,**
- **диспетчеризацию прерываний и исключений,**
- **реализует базовые синхронизационные примитивы, необходимые исполнительной системе.**

1.2.3 Драйверы устройств

К ним относятся как аппаратные драйверы устройств, которые переводят вызовы функций ввода-вывода в запросы ввода-вывода конкретного аппаратного устройства, так и неаппаратные драйверы устройств, такие как драйверы файловой системы и сети.

1.3 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 (таких как различия между материнскими платами).
    - **Уровень аппаратных абстракций** располагается между ядром и аппаратной частью компьютера Он включает программы, которые предназначены для **конкретного аппаратного обеспечения,** и «изолирует» систему от особенностей последнего (В частности, эти программы осуществляют взаимодействие с контроллером прерываний или управление взаимодействием между центральными процессорами в мультипроцессорной системе).

1.4 Система организации многооконного интерфейса и графики

- реализующая функции графического пользовательского интерфейса (graphical user interface, GUI), более известные как имеющиеся в Windows USER- и GDI-функции, предназначенные для работы с окнами, элементами управления пользовательского интерфейса и графикой

Два нижних уровня программного обеспечения: уровень аппаратных абстракций (**HAL, Hardware Abstraction Layer**) и ядро написаны на языке С и ассемблере и являются частично машинно-зависимыми. Верхние уровни написаны **исключительно** на С и почти полностью машинно-независимы. Драйверы написаны на С или, в некоторых случаях, на **C++.**

1.5 Различия между клиентскими и серверными версиями

Windows поставляется как в **клиентских**, так и в **серверных** версиях.

Серверные системы по умолчанию оптимизированы под **системную пропускную способность**, позволяющую им выступать в роли высокопроизводительных **серверов приложений**,

а клиентская версия (при наличии серверных возможностей) **оптимизирована по времени отклика** для интерактивного использования в качестве рабочего стола.

Например, на основе типа продукта по-другому принимается ряд решений по распределению ресурсов в процессе загрузки системы. В частности, это касается размеров и количества областей памяти, выделяемых программе для динамически размещаемых структур данных (или пулов), количества внутренних рабочих потоков системы и размера кэш-памяти системных данных.

Также серверная и клиентская версии отличаются друг от друга решениями политики времени выполнения, способом учета диспетчером памяти потребностей в системной памяти и в памяти процессов. Отличия между двумя семействами прослеживаются даже в некоторых деталях диспетчеризации потоков, составляющих их поведение по умолчанию

2 Объектная модель Windows

2.1 Понятие объекта

Объекты – все системные ресурсы и структуры данных (процессы, потоки, файлы, семафоры и т.д.).

Объекты предоставляют **унифицированный интерфейс.**

Объекты подразделяются на **типы.** У каждого объекта есть свойства, общие для всех объектов **данного типа.**

Поддержка объектов позволяет реализовать унифицированный подход к **именованию** ресурсов, **защите от несанкционированного** доступа и **совместному использованию**.

В Windows используется следующая объектная терминология:

- **Объект** -экземпляр объектного типа
- **Атрибуты объекта** - элементы данных состояния объекта.
- **Объектные сервисы** операционной системы (наборы операций, связанных с объектами) используются для управления объектами.

Одна часть этих сервисов является **внутренней** и используется только компонентами исполнительной системы, а другая часть открыта для процессов пользовательского режима и составляет **интерфейс исполнительной системы**.

Термин **метод**, который в соответствии с традиционной объектной терминологией используется для обозначения интерфейсной операции объекта, в Windows имеет специфическое значение.

Диспетчер объектов создает и удаляет объекты, а также отслеживает их использование.

Каждый тип объектов поддерживается определенной подсистемой. Использование объектов

Использование объектов в операционной системе Windows осуществляется согласно общим принципам: **открытие, работа с объектом, закрытие.**

Объекты могут быть:

Постоянными (файлы),

Динамическими (потоки).

Постоянные объекты открываются с помощью операции **открыть ( )**, а динамические - с помощью операции **создать ()**, в составе которой выполняется и операция **открыть**().

Схема именования объектов — иерархическая.

Она поддерживается с помощью **объектов-каталогов**

Имя объекта применяется для его открытия, совместного использования и защиты.

Процесс, который знает имя требующегося ему объекта, может его открыть в одном из поддерживаемых режимов.

Когда он выполнит вызов **открыть ( )** операционная система проверит, имеет ли процесс разрешение на использование объекта в указанном режиме, и в случае подтверждения вернет **дескриптор объекта** (или описатель), С открытым объектом связан **дескриптор защиты**, используемый для проверки

прав процесса при последующих обращениях к объекту. Для работы с объектом процесс выполняет соответствующие вызовы

Дескрипторами открытых объектов процессов могут пользоваться все его потоки.

Возможно и совместное использование объекта процессами, каждый из которых открыл его и получил описатель. Система отслеживает число дескрипторов открытого объекта, чтобы закрыть его, когда он больше не нужен.

2.2 Структура объекта

Каждый объект относится к некоторому типу, определяющему его данные и применимые к нему операции базовых системных сервисов.

Атрибуты объекта делится на две части: **заголовок и тело.**

С телом объекта работает создавший его компонент исполнительной системы, а с заголовком связан дескриптор объекта.

Поскольку один открытый объект могут использовать несколько процессов, то говорят, что процессы запрашивают разрешение на открытие и закрытие дескрипторов, а не самого объекта.

2.2.1 Типовые объекты

Тело объекта содержит специфические для его типа данные, за которые отвечает один из компонентов исполнительной системы.

Если значения некоторых атрибутов неизменны для всех объектов определенного типа, эти атрибуты входят в состав отдельного **типового объекта**, на который указывает последний атрибут заголовка объекта.

Типовой объект содержит структуру данных, которая связывает между собой все объекты соответствующего типа.

2.2.2 Полная структура объекта

Стандартные операции над объектом и атрибутами его заголовка составляют **интерфейс менеджера объектов**.

Значения этих атрибутов хранятся во внутренних структурах данных менеджера.

Тело объекта принадлежит тому компоненту исполнительной системы, в котором определен соответствующий объектный тип. Специфические для этого типа операции, связанные с его индивидуальными атрибутами, входят в состав интерфейса указанного компонента исполнительной системы.

2.2.3 Имена объектов и каталогов

Для того чтобы компоненты и приложения системы могли идентифицировать, находить и совместно использовать ее объекты, этим объектам должны быть присвоены имена.

В Windows 2000 эти имена уникальны в масштабах всей системы, а пространство имен видимо всем процессам. Win32 и другие подсистемы пользовательского режима, а также компоненты исполнительной системы могут создавать иерархические структуры, состоящие из **объектов-каталогов**.

2.2.4 Структура объекта-каталога.

Объект-каталог содержит список имен объектов и список сервисных операций. Первые две операции используются для **создания объекта-каталога и открытия его дескриптора**, а третья — для получения **списка имен объектов**, хранящихся в каталоге

После того как поток открыл дескриптор каталога (с доступом для записи), он может создавать другие объекты и помещать их в этот каталог.

Домены имен объектов.

Все пространство имен объектов разделено на **домены.**

Например, в нем имеется домен объектов, представляющий файлы и каталоги, которыми управляет файловая система. Одной из ее функций является поиск файлов по их именам.

Для **динамических объектов**, которые существуют только во время выполнения программы, поиск как таковой не требуется: обращения к ним осуществляется с использованием указателей на их местоположение в памяти.

Windows поддерживает символические ссылки, причем не только на файлы, но и на другие объекты. Они реализованы в виде специального объектного типа, атрибутами которого являются строка подстановки и время создания.

Структура пространства имен такова, что его можно расширять и на внешние объекты - таким образом, осуществляется поддержка распределенных систем.

2.2.5 Дескрипторы объектов

С каждым процессом связана **таблица объектов**, содержащая указатели на все открытые этим процессом объекты, а дескриптор объекта является индексом в этой таблице.

Таблица объектов имеет трехуровневую структуру, подобно многоуровневой таблице страниц, благодаря чему обеспечивается доступ к большому количеству дескрипторов без необходимости выделять для таблицы объектов одну непрерывную область памяти.

2.2.6 Методы объекта

Менеджер объектов выполняет над объектами стандартные действия с помощью набора интерфейсных операций**.**

Менеджер объектов позволяет компонентам исполнительной системы, отвечающим за управление конкретными объектами, самим выполнять связанные с ними стандартные операции. Такие операции называются **методами** объектов.

Когда компонент исполнительной системы создает новый объектный тип, он может зарегистрировать в менеджере объектов один или несколько методов. После этого менеджер вызывает указанные методы в строго определенные моменты существования объектов, в частности при их открытии, закрытии и удалении, а также в других случаях.

2.2.7 Защита объектов

Для того чтобы соответствовать требованиям класса защиты С2, система должна поддерживать защищенную процедуру входа, управление доступом, аудит и защиту памяти.

Когда пользователь входит в систему, подсистема защиты производит его аутентификацию, а затем связывает с его процессом **объект**, называемый **маркером доступа**.

Этот объект служит официальным «удостоверением личности» процесса, когда тот запрашивает какой-либо системный ресурс.

Он содержит **список контроля доступа (Access Control List, ACL)** - список прав доступа, присоединяемый к каждому создаваемому пользователем объекту в том случае, если пользователь не задал права доступа к объекту явно, и они не наследуются от другого объекта.