Кто Я?

Существуют две альтернативы:

- $Я_{и}$ – *Вечный закон Вечного творения;*
- $я_{л}$ – *временный результат Вечного творения.*

Есть свобода выбора:

Что Человек выбирает тем и становится!

Важнее исследовать базовые принципы организации Операционных Систем чем конкретную её реализация

Лек.№ 8. Диспетчер объектов.

№	Вопрос
1	Требования при разработке диспетчера
2	Объекты исполнительной системы
3	Структура объектов
4	Объекты типа
5	Методы объекта
6 *	Описатели объектов и таблица описателей в сост. процесса
7	Хранение объектов в памяти
8	Защита объектов
9	Учет ресурсов
10	Имена объектов
11	Объекты «каталоги объектов»
12	Символьные ссылки
13	Пространство имен Terminal Services

Объектная модель

- Первейшей задачей при разработке системы является обеспечение простого и дешевого способа ее модификации. ***Объектно-ориентированное программное обеспечение позволяет упростить процесс модификации за счет того, что реальное физическое представление данных скрыто внутри объектов.*** **Объект - это структура, физический формат которой скрыт под определением типа. Он состоит из ряда формальных свойств, называемых атрибутами, и управляется набором сервисов.**
- **В Windows NT объекты используются для представления системных ресурсов.** Любой ресурс системы, который одновременно может быть использован более чем одним процессом, включая файлы, совместно используемую память и физические устройства, реализован в виде объекта и управляется рядом сервисов. Такой подход сокращает число изменений, которые необходимо внести в операционную систему в процессе ее эксплуатации. Если, скажем,

изменилось что-то в технике, то все, что необходимо сделать - заменить соответствующий объект. Аналогично, если требуется поддержка новых ресурсов, то надо добавить только новый объект, не изменяя при этом остального кода операционной системы.

Дополнительно можно назвать следующие преимущества использования объектной модели:

- операционная система управляет ресурсами и осуществляет доступ к ним однотипно.
- упрощается реализация защиты в системе. Когда кто-либо пытается получить доступ к объекту, операционная система перехватывает обращение и выполняет проверку прав на доступ, независимо от того, чем в действительности является объект: процессом, областью памяти или портом связи;
- объекты предоставляют удобный и единый способ разделения ресурсов между двумя и более процессами.

Два вида внутренних объектов

В Windows 2000 существует два вида внутренних объектов: ***объекты исполнительной системы*** (executive objects) и ***объекты ядра*** (kernel objects). **Первые реализуются различными компонентами исполнительной системы** (диспетчером процессов, диспетчером памяти, подсистемой ввода-вывода и т. д.). **Вторые являются более примитивными объектами, которые реализуются ядром Windows 2000.** Эти объекты, невидимые коду пользовательского режима, создаются и используются только в исполнительной системе. Объекты ядра поддерживают фундаментальную функциональность (например, синхронизацию), на которую опираются объекты исполнительной системы. Так, многие объекты исполнительной системы содержат (инкапсулируют) один или несколько объектов ядра, как показано на рис. 3-11.

Объекты исполнительной системы, включающие объекты ядра

Рис.3-11

Каждая подсистема окружения Windows 2000 проецирует на свои приложения разные образы операционной системы. **Объекты исполнительной системы и сервисы объектов – именно те примитивы, из которых подсистемы окружения конструируют собственные версии объектов и других ресурсов.**

Как правило, объекты исполнительной системы создаются подсистемой окружения в интересах пользовательских приложений или компонентов операционной системы в процессе обычной работы. Так, для создания файла Win32-приложение вызывает Win32-функцию *CreateFile,* реализованную в DLL подсистемы Win32, Kerne132.dll. После проверки и инициализации *CreateFile* в свою очередь вызывает *NtCreateFile,* встроенный сервис Windows 2000, для создания объекта «файл» исполнительной системы.

Набор объектов, предоставляемый приложениям подсистемой окружения, может быть больше или меньше того набора, который предоставляется исполнительной системой. Подсистема Win32 использует объекты исполнительной системы для экспорта собственных объектов, многие из которых прямо соответствуют объектам исполнительной системы. Например, Win32-объекты «мьютекс» и «семафор» основаны непосредственно на объектах исполнительной системы (которые в свою очередь базируются на соответствующих объектах ядра). Кроме того, подсистема Win32 предоставляет именованные каналы и почтовые ящики – ресурсы, созданные на основе объектов «файл» исполнительной системы. Некоторые подсистемы вроде POSIX вообще не поддерживают объекты как таковые. Подсистема POSIX использует объекты и сервисы исполнительной системы просто как основу для POSIX-процессов, каналов и других ресурсов, которые она предоставляет своим приложениям.

Объекты исполнительной системы, доступные Win32

Тип обьектов	**Представляет:**
Символьная ссылка (symbolic link)	Механизм косвенной ссылки на имя объекта
Процесс (process)	Виртуальное адресное пространство и управляющую информацию, необходимую для выполнения набора объектов «поток»
Поток (thread)	Исполняемую часть процесса
Задание (job)	Совокупность процессов, управляемую как единая
Раздел (section)	Область разделяемой памяти (называемую в Win32 объектом «проекция файла»)
Файл (file)	Экземпляр открытого файла или устройства ввода
Маркер доступа (access token)	Профиль защиты (SID, права пользователя и т. д.) процесса или потока
Событие (event)	Объект, который может пребывать либо в занятом, либо в свободном состоянии; используется для синхронизации или уведомления

Семафор (semaphore)	Счетчик, действующий как шлюз к ресурсам; позволяет указывать максимальное число потоков, которым разрешен доступ к защищенным этим объектом ресурсам
Мьютекс (mutex)	Механизм синхронизации, используемый для упорядочения доступа к ресурсам
Таймер (timer)	Механизм уведомления потока об истечении фик сированного периода времени
loCompletion	Метод постановки в очередь и извлечения из нее уведомлений о завершении операций ввода-вывода (в Win32 API называется портом завершения ввода-вывода)
Раздел реестра (key)	Механизм ссылки на данные в реестре; хотя разделы реестра появляются в пространстве имен диспетчера объектов, они управляются диспетчером конфигурации по аналогии с тем, как объекты. «файл» управляются драйверами файловой системы; с объектом «раздел реестра» может быть сопоставлено произвольное количество параметров (от 0 и более)
Рабочий стол (desktop)	Объект, содержащий буфер обмена, набор глобальных атомов и группу объектов «рабочий стол»
WindowStation	Объект, содержащийся в объекте типа WindowStation; он описывает логическую поверхность экрана и содержит окна, меню и ловушки

Стандартные атрибуты заголовка объекта

Атрибут	Описание
Имя объекта (object name)	**Позволяет совместно использовать объект, делая его видимым для других процессов**
Каталог объектов (object directory)	**Предоставляет иерархическую структуру для хранения имен объектов**
Дескриптор защиты (security descriptor)	**Определяет, кто и как может использовать длинный объект**
Квота (quota charges)	Устанавливает ограничения на объемы ресурсов при открытии процессом описателя данного объекта
Счетчик открытых описателей (open handle count)	**Подсчитывает, сколько раз был открыт описатель объекта**
Список открытых описателей (open handles list)	**Указывает на список процессов, открывших описатели данного объекта**
Тип объекта (object type)	**Указывает на объект типа, содержащий атрибуты, общие для объектов данного типа**
Счетчик ссылок (reference count)	**Подсчитывает, сколько раз компоненты режима ядра ссылались на адрес данного объекта**

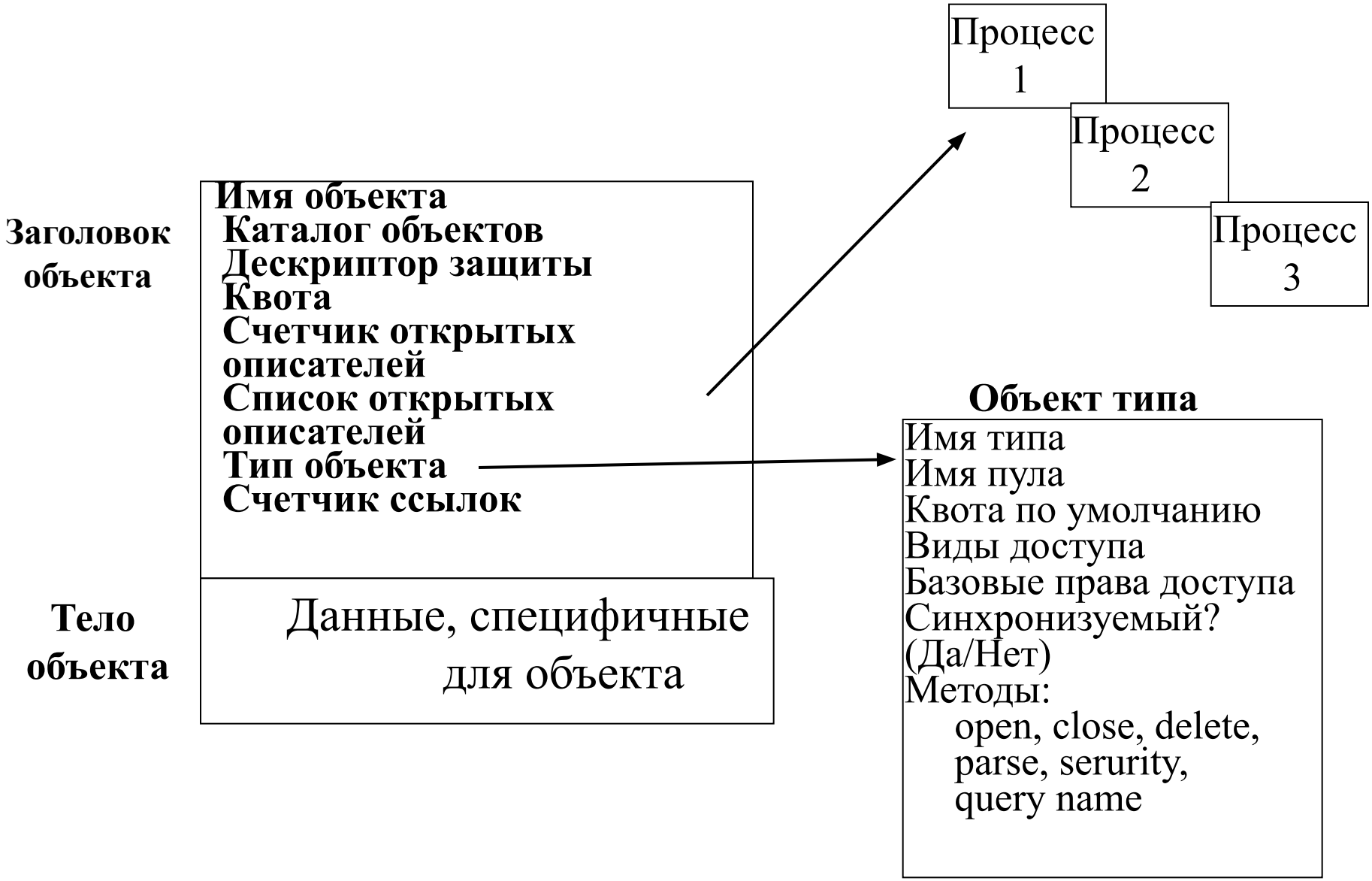
Структура объектов

Как показано на рис. 3-12, у каждого объекта есть заголовок и тело. Диспетчер объектов управляет заголовками объектов, а телами объектов управляют владеющие ими компоненты исполнительной системы. Кроме того, каждый заголовок объекта указывает на список процессов, которые открыли этот объект, и на специальный объект, называемый ***объектам типа*** (type object), – он содержит общую для всех экземпляров информацию.

Кроме заголовка у каждого объекта имеется **тело, формат и содержимое которого уникальны для данного типа объектов**; ***все объекты одного типа имеют одинаковый формат тела.***

Создавая тип объектов и предоставляя для него сервисы, компонент исполнительной системы может контролировать манипуляции с данными в телах всех объектов этого типа.

Структура объекта



Диспетчер объектов предоставляет небольшой набор базовых сервисов, которые работают с атрибутами заголовка объекта и применимы к объектам любого типа (хотя некоторые базовые сервисы не имеют смысла для отдельных объектов). Эти сервисы, часть которых доступна Win32-приложениям через подсистему Win32, перечислены в таблице 3-5.

Базовые сервисы поддерживаются для всех типов объектов, **но у каждого объекта есть свои сервисы для создания, открытия и запроса.** Так, подсистема ввода-вывода реализует сервис создания файлов для объектов «файл», а диспетчер процессов – сервис создания процессов для объектов «процесс».

Конечно, можно было бы реализовать **единый сервис** создания объектов, но **подобная процедура оказалась бы весьма сложной**, так как набор параметров, необходимых для инициализации объекта «файл», значительно отличается, скажем, от параметров инициализации объекта «процесс». А при вызове потоком сервиса объекта для определения его типа и обращении к соответствующей версии сервиса диспетчер объектов каждый раз сталкивался бы с необходимостью обработки дополнительных данных. **В силу этих и иных причин сервисы, обеспечивающие создание, открытие и запросы, для каждого типа объектов реализованы отдельно.**

Базовые сервисы объектов

Сервис	Описание
Закрытие (close)	**Закрывает описатель объекта**
Дублирование (duplicate)	**Позволяет совместно использовать объект за счет дублирования его описателя и передачи его другому процессу**
Запрос объекта (query object)	**Сообщает информацию о стандартных атрибутах объекта**
Запрос защиты (query security)	**Сообщает дескриптор защиты объекта**
Установка защиты (set security)	**Изменяет защиту объекта**
Ожидание одного объекта (wait for a single object)	**Синхронизирует выполнение потока с одним объектом**
Ожидание нескольких объектов (wait for multiple objects)	**Синхронизирует выполнение потока с несколькими объектами**

Объекты типа

В заголовках объектов содержатся общие для всех объектов атрибуты, но их значения могут меняться у конкретных экземпляров данного объекта. Так, у каждого объекта есть уникальное имя и может быть уникальный дескриптор защиты. Однако некоторые атрибуты объектов остаются неизменными для всех объектов данного типа. Например, при открытии описателя объекта можно выбрать права доступа из набора, специфичного для объектов данного типа. Исполнительная система предоставляет в том числе атрибуты доступа «завершение» (terminate) и «приостановка» (suspend) для объектов «поток», а также «чтение» (read), «запись» (write), «дозапись» (append) и «удаление (delete) для объектов «файл». Другой пример атрибута, специфичного для типа объектов, – синхронизация, о которой мы кратко расскажем ниже.

Чтобы сэкономить память, диспетчер объектов сохраняет статические атрибуты, специфичные для конкретного типа объектов, только при создании нового типа объектов. Для записи этих данных он использует собственный объект типа. Как показано на рис. 3-13, если установлен отладочный флаг отслеживания объектов (описанный ранее), все объекты одного типа (в данном случае – «процесс») связываются между собой с помощью объекта типа, что позволяет диспетчеру объектов при необходимости находить их и перечислять.

Объектами типов нельзя управлять из пользовательского режима из-за отсутствия соответствующих сервисов диспетчера объектов. Но некоторые из определяемых ими атрибутов видимы через отдельные системные сервисы и функции Win32 API. Атрибуты, хранящиеся в объектах типа, описываются в таблице 3-6.

Объекты типа

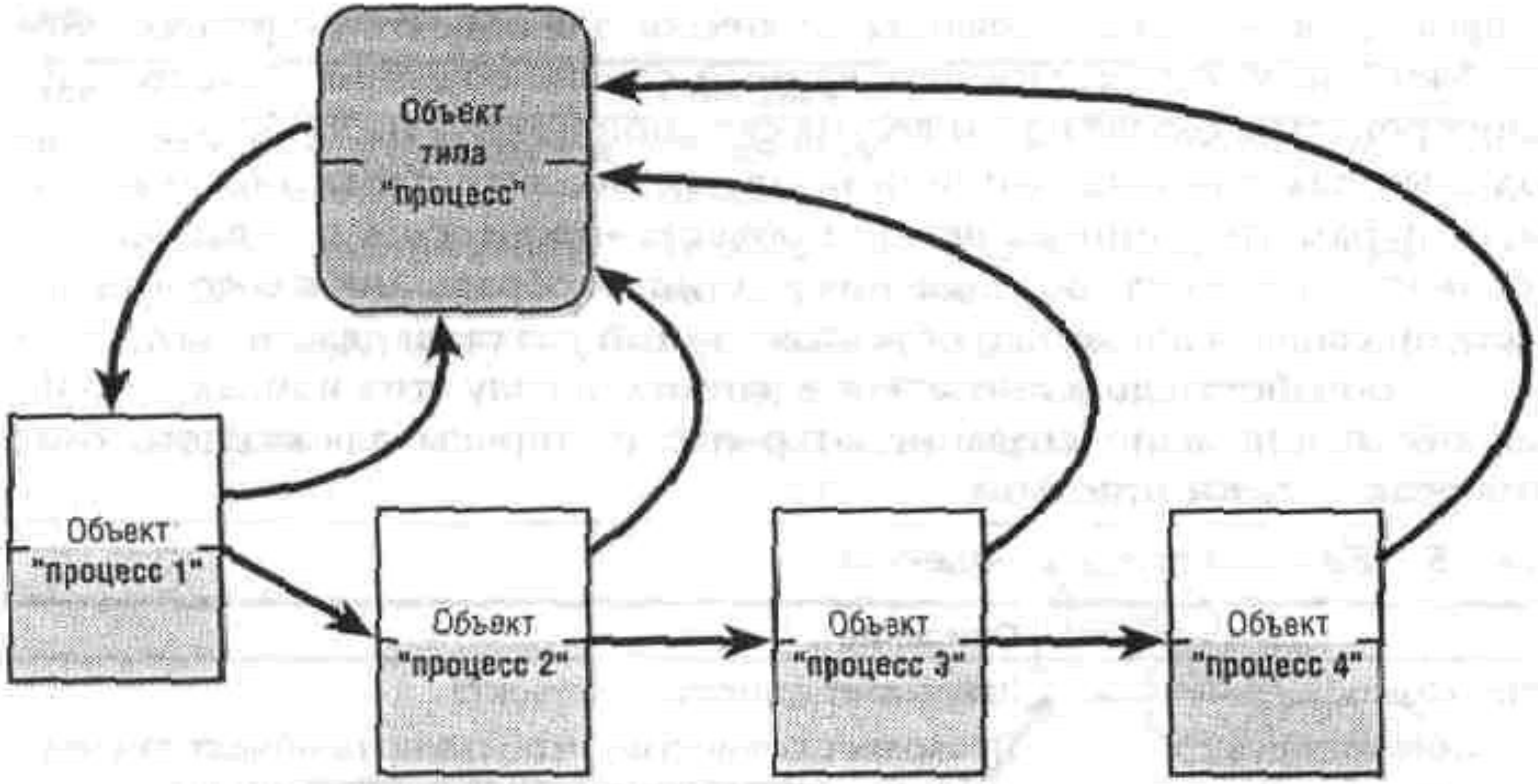


Рис. 3-13. *Объекты «процесс» и объект типа «процесс»*

Атрибуты объекта типа

Атрибут	Описание
Имя типа (type name)	Название объектов этого типа («процесс», «событие», «порт» и т. д.)
Тип пула (pool type)	Определяет тип памяти, выделяемый для объектов этого типа (подкачиваемая или неподкачиваемая)
Квота по умолчанию (default quota charges)	Ограничение по умолчанию на объемы подкачивае мой и неподкачиваемой памяти при открытии про цессом описателя данного объекта

Методы объекта

Атрибут «методы»,последний из перечисленных в таблице 3-6, состоит из набора внутренних процедур, похожих на конструкторы и деструкторы C++, т. е. процедуры, автоматически вызываемые при создании или уничтожении объекта. В диспетчере объектов эта идея получила дальнейшее развитие: он вызывает методы объекта и в других ситуациях, например при открытии или закрытии описателя объекта или при попытке изменения параметров защиты объекта. В некоторых типах объектов методы определяются в зависимости от того, как должен использоваться данный тип объектов.

При создании нового типа объектов компонент исполнительной системы может зарегистрировать у диспетчера объектов один или несколько методов. После этого диспетчер объектов вызывает методы на определенных этапах жизненного цикла объектов данного типа – обычно при их создании, удалении или модификации. Поддерживаемые диспетчером объектов методы перечислены в таблице 3-7.

Виды доступа (access types)	Виды доступа, который поток может запрашивать при открытии описателя объекта этого типа («чтение», «запись, «завершение, «приостлновка» и т. д.)
Базовые права доступа (generic access rights mapping)	Сопоставление четырех базовых прав доступа («чтение», «запись», «выполнение» и «все») с правами доступа, специфичными для данного типа
Синхронизация (synchroniza tion)	Определяет, может ли поток ждать на объектах данного типа
Методы (methods)	Одна или несколько процедур, автоматически вызываемых диспетчером объектов на определенных этапах жизненного цикла объекта

Методы объекта

Метод	**Когда происходит вызов метода**
Open	При **открытии** описателя объекта
Close	При **закрытии** описателя объекта
Delete	Перед **удалением** объекта диспетчером объектов
Query name	При **запросе** потоком имени объекта (например, файла), существующего во вторичном домене объектов
Parse	При **поиске** диспетчером объектов имени, существующего во вторичном домене объектов
Security	При считывании или изменении процессом параметров **защиты** объекта, например файла, существующего во вторичном домене объектов

Описатели объектов и таблица описателей, принадлежащая процессу

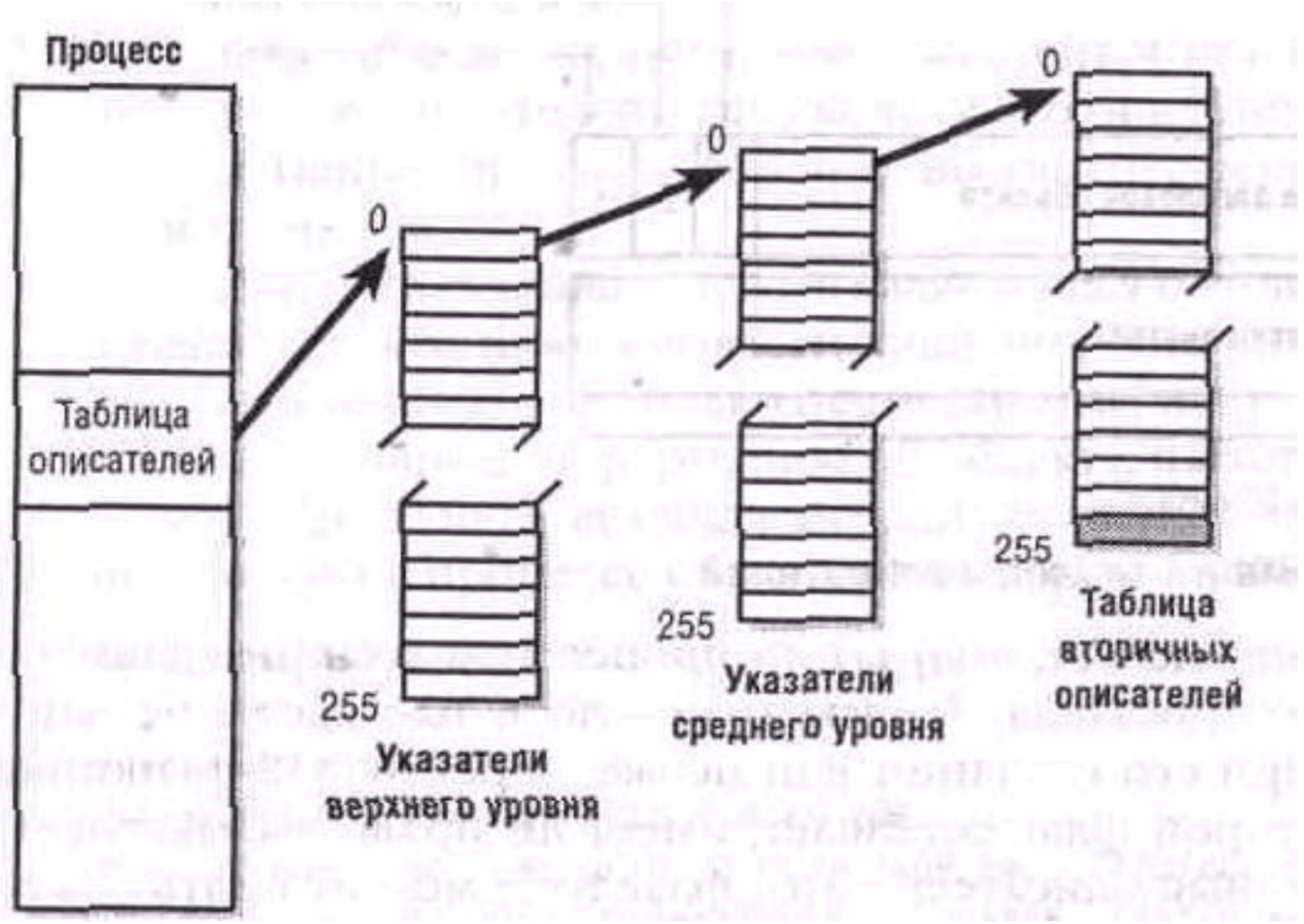
Когда процесс создает или открывает объект по имени, он получает *описатель* (handle), который дает ему доступ к объекту. Ссылка на объект по описателю работает быстрее, чем по имени, так как при этом диспетчер объектов может сразу найти объект, не просматривая список имен.

Чтобы потоки процесса пользовательского режима могли оперировать объектом, им нужен описатель этого объекта.

Описатели служат косвенными указателями на системные ресурсы, что позволяет прикладным программам избегать прямого взаимодействия с системными структурами данных.

Описатели дают и другие преимущества. Во-первых, описатели файлов, событий или процессов совершенно одинаковы – просто ссылаются на разные объекты. Это дает возможность создать унифицированный интерфейс для ссылок на объекты любого типа. Во-вторых, только диспетчер объектов имеет право физически создавать описатели и искать их объекты, а значит, он может проанализировать любое действие с объектом в пользовательском режиме и решить, позволяет ли профиль защиты вызывающей программы выполнить над объектом запрошенную операцию.

Структура таблицы описателвй, принадлежащая процессу в Windows 2000



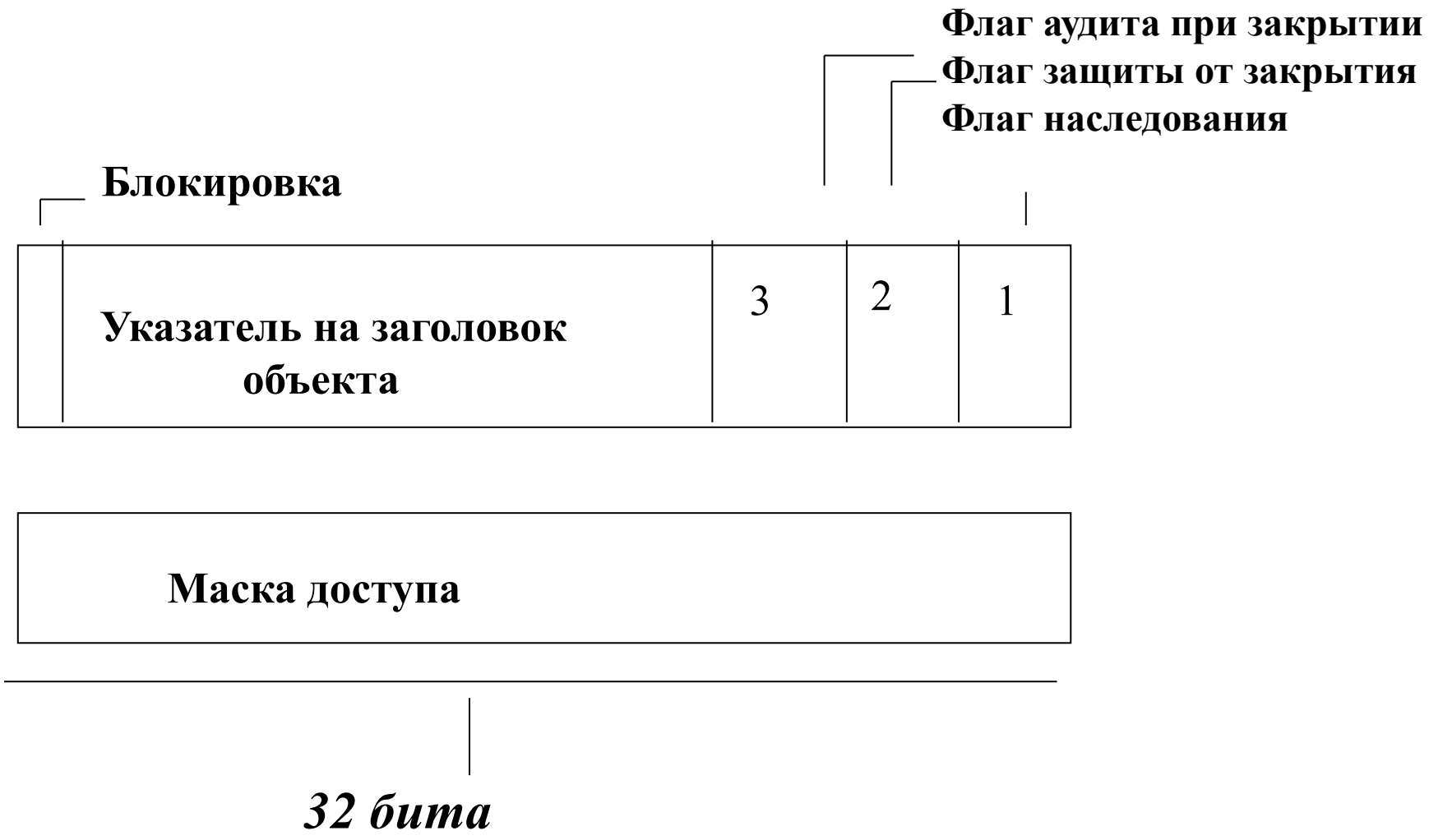
Описатель объекта представляет собой индекс в *таблице описаттей* (handle table), принадлежащей процессу, На нее указывает блок процесса исполнительной системы (EPROCESS), рассматриваемый в главе 6. Индекс первого описателя равен 4, второго – 8 и т.д. **Таблица содержит указатели на все объекты, описатели которых открыты данным процессом.** Эти таблицы реализованы по трехуровневой схеме аналогично тому, как блок управления памятью в системах типа x86 реализует трансляцию виртуальных адресов в физические (см. главу 7).

При создании процесса диспетчер объектов формирует верхний уровень таблицы описателей, содержащий указатели на таблицы среднего уровня; средний уровень содержит первый массив указателей на таблицы вторичных описателей; нижний уровень содержит первую таблицу вторичных описателей. Массивы каждого уровня состоят из 256 элементов, а начальная таблица процесса может содержать до 255 описателей.

Как показано на рис. 3-15, каждый элемент таблицы описателей состоит из структуры с двумя 32-битными элементами. Первый из них содержит указатель на заголовок объекта и четыре флага. В качестве флагов используются младшие 3 бита и один старший. Старший бит является флагом блокировки. Когда диспетчер объектов транслирует описатель в указатель на объект, он блокирует соответствующую запись на время трансляции. Так как все объекты находятся в системном адресном пространстве, старший бит указателя на объект устанавливается в 1. (Гарантируется, что адреса всегда будут превышать Ox80000000 даже на системах, запускаемых с ключом загрузки /3GB.) Так что, пока элемент таблицы описателей не заблокирован, диспетчер объектов может сбросить старший бит в 0. При блокировке элемента таблицы диспетчер объектов устанавливает этот бит и получает корректное значение указателя. Диспетчер блокирует всю таблицу описателей (используя специальную блокировку, сопоставляемую с каждым процессом), только если процесс создает новый

описатель или закрывает существующий. Второй элемент структуры является маской доступа к объекту).

Структура элемента таблицы описателей



- ✔Флаг наследования определяет, получат ли процессы, созданные данным процессом, копию этого описателя. Как уж отмечалось, наследование описателя можно указать при его создании или позже, через Win32-функцию *SetHcindleInformation.*
- ✔Второй флаг сообщает, имеет ли право вызывающая программа закрывать данный описатель. (Этот флаг тоже можно задать вызов*ом 5etHandleInfonnation.)*
- ✔Третий флаг указывает, будет ли генерироваться сообщение аудита при закрытии объекта. (Этот флаг не экспортируется в Win32 и предназначен для внутреннего использования диспетчером объектов.)

Защита объектов

Открыв файл, нужно указать, для чего это делается – для чтения или записи. Если Вы попытаетесь записать что- нибудь в файл, открытый для чтения, Вы получите ошибку. Аналогичным образом действует и исполнительная система: когда процесс создает объект или открывает описатель существующего объекта, он должен указывать набор *желательных прав доступа* (desired access rights), сообщая тем самым, что именно он собирается делать с объектом. Процесс может запросить либо набор стандартных прав доступа (чтение, запись, выполнение), применимых ко всем объектам, либо специфические права доступа, различные для объектов разного типа. Так, в случае объекта «файл» процесс может запросить права на удаление файла или дозапись, а в случае объекта поток» – права на приостановку потока или его завершение.

Когда .процесс открывает описатель объекта, диспетчер объектов вызывает так называемый *справочный монитор безопасности'* (security reference monitor), часть системы защиты, работающую в режиме ядра, и посылает ему уведомление о наборе желательных для процесса прав доступа. Справочный монитор безопасности проверяет, разрешает ли дескриптор защиты объекта запрашиваемый тип доступа. Если да, справочный монитор безопасности возвращает процссу набор *предоставленных прав доступа* (granted access rights), информацию о которых диспетчер объектов сохраняет в создаваемом им описателе объекта.

Хранение объектов в памяти

Поскольку для доступа к объекту все процессы пользовательского режима должны сначала открыть его описатель, диспетчер объектов может легко отслеживать, сколько процессов и даже какие именно из них используют объект. Учет описателей является одним из механизмов, реализующих *хранение объектов в памяти* (object retention) – сохранение временных объектов лишь до тех пор, пока они используются, с их последующим удалением.

Диспетчер объектов реализует этот механизм в двух фазах. Первая фаза называется *хранением имен* (name retention) и контролируется числом открытых описателей объекта. Каждый раз, когда процесс открывает описатель объекта, диспетчер увеличивает значение счетчика открытых описателей в заголовке объекта. По мере того как процессы завершают использование объекта и закрывают его описатели, диспетчер уменьшает значение этого счетчика. Когда счетчик обнуляется, диспетчер удаляет имя объекта из своего глобального пространства имен. После этого новые процессы уже не смогут открывать описатели данного объекта.

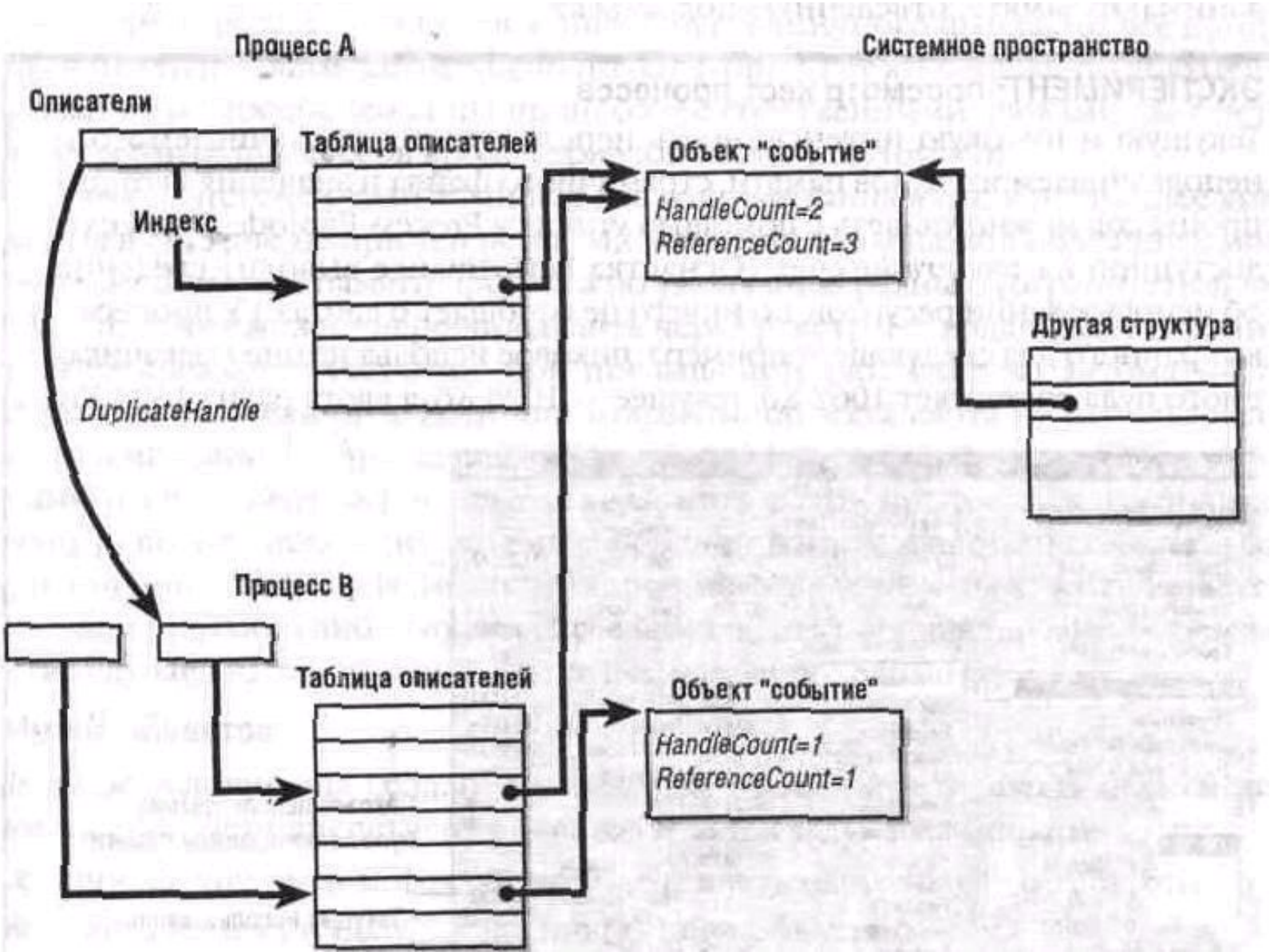
Вторая фаза заключается в том, что прекращается хранение тех объектов, которые больше не используются (т. е. они удаляются). Так как код операционной системы обычно обращается к объектам по указателям, а не описателям, диспетчер объектов должен регистрировать и число указателей объектов, переданных процессам операционной системы. При каждой выдаче указателя на объект он увеличивает значение *счетчика ссылок* на объект. Компоненты режима ядра, прекратив использовать указатель, вызывают диспетчер объектов для уменьшения счетчика ссылок. Система также увеличивает счетчик ссылок при увеличении счетчика описателей, а при уменьшении счетчика описателей соответственно уменьшает счетчик ссылок, поскольку описатель тоже является подлежащей учету ссылкой на объект.

На рис. 3-16 показаны два задействованных объекта-события. Процесс 1 открыл первый объект, а процесс В – оба объекта. Кроме того, на первый объект ссылается какая-то структура режима ядра, и его счётчик ссылок равен 3. Так что, даже если оба процесса закроют свои описатели первого объекта, он по-прежнему будет существовать, поскольку его счетчик ссылок ещё не обнулился. Но, когда процесс В закроет свой описатель второго объекта, этот объект будет удален.

Таким образом, даже если счетчик открытых описателей объекта обнулится, счетчик ссылок может превышать нулевое значение, указывая, что операционная система еще использует объект. В конце концов счетчик ссылок тоже обнулится, и тогда диспетчер удалит соответствующий объект из памяти.

Такой механизм позволяет хранить объект и его имя в памяти, просто не закрывая его описатель. Программистам, создающим приложения с двумя и более взаимодействующими процессами, не приходится беспокоиться о том, что один из процессов удалит объект в то время, когда он еще используется другим процессом.

Счетчики описателей и ссылок



Учет ресурсов

Учет ресурсов, как и хранение объектов, тесно связан с использованием описателей объектов. Положительное значение счетчика открытых описателей указывает на то, что данный ресурс задействован какими-то процессами. Когда счетчик описателей обнуляется, процессы, использовавшие этот объект, больше не занимают память, отведенную под объект.

Во многих операционных системах для ограничения доступа процессов к системным ресурсам применяется система квот. Однако типы устанавливаемых для процессов квот иногда весьма разнообразны, а отслеживающий квоты код распределен по всей операционной системе. Так, в некоторых операционных системах компонент ввода-вывода может регистрировать и ограничивать число , которые может открыть процесс, а компонент управления памятью может накладывать ограничения на объем памяти, выделяемой потокам процесса. Компонент, отвечающий за управление процессами, способен ограничивать максимальное число новых процессов или новых потоков процесса. Каждое из этих ограничений отслеживается и реализуется в различных частях операционной системы.

Диспетчер объектов Windows 2000, напротив, представляет собой компонент централизованного учета ресурсов. В заголовке каждого объекта содержится атрибут квоты, определяющий, насколько диспетчер объектов уменьшает квоту подкачиваемой или неподкачиваемой памяти процесса при открытии его потоком описателя этого объекта.

У каждого процесса в Windows 2000 имеется структура квот, регистрирующая лимиты и текущее количество используемой памяти из подкачиваемого и неподкачиваемого пулов, а также из страничного файла. Но все процессы в интерактивном сеансе используют один и тот же блок квот (документированного способа создания процессов с собственными блоками квот нет), а у системных процессов вроде сервисов квоты отсутствуют.

Имена объектов

Важное условие для создания множества объектов — эффективная система учета. Для учета диспетчеру объектов нужна следующая информация:

- способ, которым можно было бы отличать один объект от другого;
- метод поиска и получения конкретного объекта.

Первое требование реализуется за счет присвоения имен объектам. Это расширение обычной для большинства операционных систем функциональности, в которых отдельным системным ресурсам, например файлам, каналам или блокам разделяемой памяти, можно присваивать имена. Исполнительная система, напротив, позволяет именовать любой объект, представляющий ресурс. Второе требование (поиск и получение объектов) также реализуется через именование объектов. Если диспетчер хранит объекты в соответствии с их именами, он может быстро найти объект по его имени.

Имена объектов отвечают и третьему требованию, не упомянутому в предыдущем списке: процессам должна быть предоставлена возможность совместного использования объектов. Пространство имен объектов исполнительной системы является глобальным, видимым любому процессу в системе. Если один процесс создает объект и помещает его имя в глобальное пространство имен, то другой процесс может открыть описатель этого объекта, указав нужное имя. Если объект не предназначен для совместного использования, процесс-создатель просто не присваивает ему имя.

Где именно хранятся имена объектов, зависит от типа объектов. В таблице 3-8 перечислены стандартные каталоги объектов, имеющиеся на всех системах под управлением Windows 2000. (Пользовательским программам видны только каталоги \BaseNamedObjects и \??.)

Каталог	**Типы объектов, имена которых хранятся в каталоге**
\??	Имена устройств MS-DOS (\DosDevices является символьной ссылкой на этот каталог)
\BaseNmed Objects	Мьютексы, события, семафоры, ожидаемые таймеры и разделы
\Callback	Объекты обратного вызова
\Device	Объекты устройств
\Driver	Объекты драйверов
\FileSystem	Объекты драйверов файловых систем и объекты распознавания файловых систем

\KnownDlls	Имена разделов и пути поиска известных DLL (DLL, проецируемых системой при запуске)
\Nls	Имена разделов спроецированных таблиц поддержки национальных языков
\ObjectTypes	Имена типов объектов
\RPC Control	Объекты портов, используемые для вызова удаленных процедур (RPC)
\Security	Имена объектов, специфичных для подсистемы защиты
\Windows	Порты подсистемы Win32 и объекты WindowStation

Манифест гуманной педагогики Ш.А. Амонашвили
http://www.enlightening.ee/articles/rus_manifest.htm

- Гуманная педагогика в том виде, как мы ее предлагаем, есть детище классической педагогики. Она приемлет классическую основу с той оговоркой, что оставляет возможность, чтобы в понятие духовности, кроме других составляющих, можно было закладывать суть того или иного классического мирового религиозного учения. Потому в качестве смысла духовности мы принимаем допущения в трех аксиоматических постулатах:
- Реальность Высшего Мира, Высшего Сознания, Бога.
- Реальность бессмертия человеческого духа и его устремленность к вечному совершенствованию.
- Понимание земной жизни как отрезка пути духовного совершенствования и восхождения.
- Из этих допущений делаем выводы о философском восприятии Ребенка:
- он есть явление (веление духа) в нашей земной жизни,
- он есть носитель своего предназначения, своей миссии,