Оглавление

1 Компоненты подсистемы ввода вывода 1
2 Обработка запросов ввода-вывода 3
2.1 Диспетчер ввода-вывода 3
2.2 Драйверы устройств 5
2.2.1 Типы драйверов устройств 5
2.2.2 Структура драйвера. Основные процедуры. 7
2.3 Примеры файлового ввода-вывода 9
3 Структуры данных, связанные с вводом-выводом 10
3.1 Объекты «файл». 10
3.2 Объекты «драйвер» и «устройство» 12
3.3 Пакеты запросов ввода-вывода 14
4 Технология ввода-вывода 14
4.1 Типы ввода-вывода 15
4.2 Синхронный ввод-вывод 15
4.3 Асинхронный ввод-вывод 16
4.4 Проецируемый ввод вывод 17
4.5 Быстрый ввод-вывод 18

Подсистемы ввода-вывода Windows разрабатывалась с тем расчетом, чтобы драйверы можно было устанавливать и удалять **динамически**. При этом термин драйвер в Windows употребляется в более широком смысле, чем обычный термин драйвер устройства. Файловые системы - это тоже драйверы, но более сложные. Они принимают запросы ввода-вывода, анализируют их и, в свою очередь, обращаются к драйверам физических устройств.

Для работы системы ввода-вывода необходима поддержка параллельного выполнения, аппаратно-программной синхронизации и взаимного исключения при доступе к общим буферам.

1 Компоненты подсистемы ввода вывода

Диспетчер ввода-вывода подключает приложения и системные компоненты к виртуальным. логическим и физическим устройствам, а также определяет инфраструктуру, поддерживающую драйверы устройств.

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.

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

Диспетчер PnP и драйверы шин отвечают за загрузку соответствующего драйвера при обнаружении нового устройства. Если устройство добавляется в систему, в которой нет нужного драйвера устройства, компоненты исполнительной системы, отвечающие за поддержку PnP, вызывают сервисы установки устройств**,** поддерживаемые диспетчером PnP пользовательского режима.

Реестр служит в качестве базы данных, в которой хранится описание основных устройств**,** подключенных к системе, а также параметры инициализации драйверов и конфигурационные настройки.

INF-файлы используются для установки драйверов; 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.

CAT-файлы хранят цифровые подписи, которые удостоверяют файлы драйверов, прошедших испытания в лаборатории Microsoft Windows Hardware Qualiry 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.

2 Обработка запросов ввода-вывода

Как правило, запрос на ввод-вывод выдается приложением, выполняющим соответствующую операцию (например, чтение данных с устройства); такие операции обрабатываются диспетчером ввода-вывода, одним или несколькими драйверами устройств и HAL.

В Windows потоки выполняют операции ввода-вывода над **виртуальными файлами.** 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 - Win32, POSIX или OS/2) вызывают документированные функции, которые в свою очередь обращаются к внутренним функциям подсистемы ввода-вывода для чтения/записи файла и для выполнения других операций. Запросы, адресованные виртуальным файлам, диспетчер ввода-вывода динамически направляет соответствующим драйверам устройств.

2.1 Диспетчер ввода-вывода

Диспетчер ввода-вывода **(I/O manager)** определяет модель доставки запросов 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(быстрый ввод-вывод IRP не использует).

Подсистема ввода-вывода позволяет потоку приложения управлять сразу несколькими запросами на ввод-вывод.

IRP — это структура данных, которая содержит информацию, полностью описывающую запрос 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.

Диспетчер ввода-вывода не только создает и уничтожает IRP, но и содержит **общий для различных драйверов** код, который они используют при обработке ввода-вывода. Благодаря этому драйверы стали проще и компактнее. Так, одна из функций диспетчера ввода-вывода позволяет драйверу вызывать другие драйверы.

Диспетчер также управляет буферами запросов ввода-вывода, тайм-аутами драйверов и регистрирует, какие устанавливаемые файловые системы загружаются в операционную систему. Драйверы устройств могут вызывать около сотни функций, предоставляемых диспетчером ввода-вывода.

Диспетчер ввода-вывода также предоставляет гибкие сервисы ввода-вывода, на основе которых подсистемы окружения (например, Win32 и POSIX) реализуют свои функции. В их число входят сервисы **асинхронного ввода-вывода**, которые дают возможность разработчикам создавать высокопроизводительные масштабируемые серверные приложения.

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.

Кроме обычных функций для открытия, закрытия, чтения и записи подсистема ввода-вывода Windows предоставляет ряд дополнительных функций, например, для асинхронного, прямого и буферизованного ввода-вывода, а также для ввода-вывода по механизму «scatter/gather».

2.2 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.

2.2.1 Типы драйверов устройств

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

Драйверы Windows. Драйверы устройств, при необходимости интегрируемые с диспетчерами электропитания и PnP. В их число входят драйверы для устройств внешней памяти, стеков протоколов и сетевых адаптеров.

WDM-драйверы. Для взаимодействия драйверов устройств с ядром и исполнительной системой во всех операционных системах Windows используется так называемая модель драйверов Windows (Windows Driver Model, WDM).

Существует три типа WDM-драйверов.

■ ***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

РпР и управление параметрами электропитания шины.

■ ***Функциональные драйверы***. Управляют конкретным типом устройств. Функциональным считается драйвер, экспортирующий рабочий интерфейс устройства операционной системе. Как правило, это драйвер, больше других знающий о функционировании определенного устройства.

■ ***Драйверы фильтров***. Занимающие более высокий логический уровень, чем функциональные драйверы, они дополняют функциональность или изменяют поведение устройства либо другого драйвера. Так, утилиту для перехвата ввода с клавиатуры можно реализовать в виде драйвера фильтра клавиатуры, расположенного на более высоком уровне, чем функциональный драйвер клавиатуры.

В WDM ни один драйвер не отвечает за все аспекты управления конкретным устройством.

Драйвер шины определяет изменения в составе устройств на шине (при подключении и отключении устройств), помогает диспетчеру РпР 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.

- **Драйверы классов (class drivers).** Реализуют обработку ввода-вывода для конкретного класса устройств, например, дисковых устройств, ленточных накопителей или приводов CD-ROM.
- **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.

2.2.2 Структура драйвера. Основные процедуры.

Выполнением драйверов устройств управляет подсистема ввода-вывода. Драйвер устройства состоит из набора процедур, вызываемых на различных этапах обработки запроса ввода-вывода.

Инициализирующая процедура. Диспетчер ввода-вывода выполняет инициализирующую процедуру драйвера (которая обычно называется *Driver-Entry)* при загрузке этого драйвера в операционную систему. Данная процедура регистрирует остальные процедуры драйвера в диспетчере ввода-вывода, заполняя соответствующей информацией системные структуры данных, и выполняет необходимую глобальную инициализацию драйвера.

Процедура добавления устройства. Такие процедуры реализуются в драйверах, поддерживающих Plug and Play. Через эту процедуру диспетчер PnP посылает драйверу уведомление при обнаружении устройства, за которое отвечает данный драйвер. Выполняя эту процедуру, драйвер обычно создает объект «устройство», представляющий аппаратное устройство.

Процедуры диспетчеризации. Это основные функции, предоставляемые драйвером устройства, например для открытия, закрытия, чтения, записи и реализации других возможностей устройства, файловой системы или сети. Диспетчер ввода-вывода, вызванный для выполнения операции ввода-вывода, генерирует IRP и обращается к драйверу через одну из его процедур диспетчеризации.

Процедура инициации ввода-вывода. С помощью этой процедуры драйвер может инициировать передачу данных как на устройство, так и с него. Эта процедура определяется лишь в драйверах, использующих поддержку диспетчера ввода-вывода для сериализации IRR Диспетчер ввода-вывода сериализует IRP для драйвера, гарантируя одновременную обработку им только одного IRR. Большинство драйверов обрабатывают сразу несколько IRP, но сериализация имеет смысл для некоторых драйверов, в частности для драйвера клавиатуры.

Процедура обслуживания прерываний (ISR). Когда устройство

генерирует прерывание, диспетчер прерываний ядра передает управление этой процедуре. В модели ввода-вывода Windows процедуры ISR работают на уровне DIRQL (Device IRQL), поэтому они выполняют минимум действий во избежание слишком продолжительной блокировки прерываний более низкого уровня. Для выполнения остальной части обработки прерывания ISR ставит в очередь DPC (deferred procedure call), выполняемый при более низком IRQL (уровня «DPC/dispatch»). ISR имеются лишь в драйверах устройств, управляемых прерываниями, — например, в драйвере файловой системы ISR нет.

DPC-процедура обработки прерываний. DPC-процедура выполняет основную часть обработки прерывания, оставшуюся после выполнения ISR. Она работает при более низком IRQL (уровня «DPC/dispatch»), чем ISR, чтобы не блокировать без необходимости другие прерывания. DPC-процедура инициирует завершение **текущей** операции ввода-вывода и выполнение следующей операции ввода-вывода из очереди на данном устройстве.

Одна или несколько процедур завершения ввода-вывода. У драйвера могут быть процедуры завершения ввода-вывода, уведомляющие его об окончании обработки IRP драйвером более низкого уровня. Например, диспетчер ввода-вывода вызывает процедуру завершения ввода-вывода драйвера файловой системы, когда драйвер устройства заканчивает передачу данных в файл или из него. Эта процедура уведомляет драйвер файловой системы об удачном или неудачном завершении операции или о ее отмене, а также позволяет драйверу файловой системы освободить ресурсы.

Процедура отмены ввода-вывода. Если операция ввода-вывода может быть отменена, драйвер определяет одну или более процедур отмены ввода-вывода. Получив IRP для запроса ввода-вывода, который может быть отменен, драйвер связывает с IRP процедуру отмены. Если поток, выдавший запрос на ввод-вывод, завершается до окончания обработки запроса или отменяет операцию, диспетчер ввода-вывода выполняет процедуру отмены, связанную с IRP (если таковая есть). Процедура отмены отвечает за выполнение любых действий, необходимых для освобождения всех ресурсов, выделенных при обработке IRP, а также за завершение

IRP со статусом отмены.

Процедура выгрузки. Эта процедура освобождает все системные ресурсы, задействованные драйвером, после чего диспетчер ввода-вывода может удалить их из памяти. При выполнении процедуры выгрузки обычно освобождаются ресурсы, выделенные процедурой инициализации. Драйвер может загружаться и выгружаться во время работы системы.

Процедура уведомления о завершении работы системы. Эта процедура позволяет драйверу проводить очистку при завершении работы системы.

Процедуры регистрации ошибок. При возникновении неожиданных ошибок (например, когда на диске появляется поврежденный блок), процедуры регистрации ошибок, принадлежащие драйверу, уведомляют о них диспетчер ввода-вывода. Последний записывает эту информацию в файл журнала ошибок.

2.3 Примеры файлового ввода-вывода

Драйвер файловой системы принимает запрос на запись данных в определенное место конкретного файла. Он преобразует его в запрос на запись определенного числа байт по определенному «логическому» адресу на диске. После этого он передает этот запрос(через диспетчер ввода-вывода) простому драйверу диска. Последний в свою очередь преобразует запрос в физический адрес на диске (цилиндр/дорожка/сектор) и позиционирует головки дискового устройства для записи данных.

Диспетчер ввода-вывода получает запрос на запись, в котором адрес записи относителен началу конкретного файла. Далее диспетчер ввода-вывода передает запрос драйверу файловой системы, который преобразует информацию запроса в адрес начала записи на диске и число байт, которые нужно записать на диск. Драйвер файловой системы передает через диспетчер ввода-вывода запрос драйверу диска, который транслирует его в физический адрес на диске и переносит нужные данные.

Поскольку все драйверы - и устройств, и файловой системы - предоставляют операционной системе одинаковую инфраструктуру, в их иерархию легко добавить еще один драйвер, не изменяя существующие драйверы или подсистему

ввода-вывода. Например, введя соответствующий драйвер, можно логически представить несколько дисков как один большой диск. Такой драйвер, кстати, имеется в Windows - он обеспечивает поддержку отказоустойчивых дисков. Драйвер диспетчера томов вполне логично размещается между драйверами файловой системы и дисков.

3 Структуры данных, связанные с вводом-выводом

С запросами ввода-вывода связаны четыре основные структуры данных:

- объекты «**файл**»,
- объекты «**драйвер**»,
- объекты «**устройство**»
- **пакеты запросов ввода-вывода (IRP)**.

3.1 Объекты «файл».

Объекты «файл» являются структурами режима ядра, которые точно соответствуют определению объектов в Windows:

Это системные ресурсы, доступные для совместного использования двум или нескольким процессам, у них могут быть имена, их защита обеспечивается моделью защиты объектов, и они поддерживают синхронизацию.

Хотя большинство разделяемых ресурсов в Windows базируется в памяти, основная часть ресурсов, с которыми имеет дело подсистема ввода-вывода, размещается на физических устройствах или представляет их. Несмотря на эту разницу, операции над совместно используемыми ресурсами подсистемы ввода-вывода осуществляются как над объектами. Объекты «файл» - представление ресурсов в памяти, которое обеспечивает чтение и запись данных в эти ресурсы.

Когда поток открывает файл или простое устройство, диспетчер ввода-вывода возвращает описатель объекта **«файл».**

Пример того, что происходит при открытии файла.

Например, С-программа вызывает из стандартной библиотеки функцию ***fopen***, которая в свою очередь вызывает Win32-функцию ***CreateFile.***

Далее DLL подсистемы Win32 (в данном случае — Kernel32.dll) вызывает

функцию ***NtCreateFile*** **из** **Ntdll.dll**. Эта функция в Ntdll.dll содержит соответствующие команды, вызывающие переход в режим ядра в диспетчер системных сервисов, который и обращается к настоящей функции ***NtCreateFile*** из **Ntoskrnl.exe**

Как и другие объекты исполнительной системы, объекты «файл» защищаются дескрипторами защиты, которые содержат список управления доступом (ACL). Чтобы выяснить, позволяет ли ACL файла получить процессу тип доступа, запрошенный его потоком, диспетчер ввода-вывода обращается к системе защиты. Если да, диспетчер объектов разрешает доступ и сопоставляет предоставленные права доступа с описателем файла, возвращаемым потоку. Если этому или другому потоку того же процесса понадобятся дополнительные операции с файлом, не указанные в исходном запросе, ему придется открыть новый описатель, по которому тоже будет проведена проверка прав доступа

Поскольку объект «файл» — представление разделяемого ресурса в памяти, а не сам ресурс, он отличается от остальных объектов исполнительной системы.

Объект «файл» содержит лишь данные, уникальные для описателя объекта, тогда как собственно файл — совместно используемые данные или текст.

Всякий раз, когда поток открывает описатель файла, создается новый объект «файл» с новым набором атрибутов, специфичным для этого описателя.

Например, атрибут «текущее смещение в байтах» определяет место в файле, где будут записаны или считаны следующие данные по текущему описателю. У каждого описателя одного и того же файла свое значение атрибута «текущее смещение в байтах».

Кроме того, объект «файл» уникален для процесса; исключение составляют случаи, когда процесс дублирует описатель файла для передачи другому процессу, или когда дочерний процесс наследует описатель файла от родительского. Только в этих двух случаях процессы располагают отдельными описателями, которые ссылаются на один и тот же объект «файл».

Описатель файла уникален для процесса, но определяемый им физический ресурс - нет. Поэтому, как и при доступе к любому другому общему ресурсу, потоки

должны синхронизировать свое обращение к совместно используемым файлам, каталогам и устройствам. Если, например, поток что-то записывает в файл, то при открытии описателя файла он должен указать монопольный доступ для записи, чтобы другие потоки не могли записывать в этот файл одновременно с первым потоком. В качестве альтернативы поток может заблокировать на время записи часть файла с помощью Win32-функции *LockFile*.

3.2 Объекты «драйвер» и «устройство»

Когда поток открывает описатель объекта «файл», диспетчер ввода-вывода, исходя из имени этого объекта, должен определить, к какому драйверу (или драйверам) нужно обратиться для обработки запроса. Более того, диспетчер ввода-вывода должен знать, где найти эту информацию, когда в следующий раз поток вновь воспользуется тем же описателем файла. Для этого предназначены следующие объекты.

- Объект **«драйвер»**, представляющий отдельный драйвер в системе. Именно от этого объекта диспетчер ввода-вывода получает адрес процедуры диспетчеризации (точки входа) драйвера.
- Объект **«устройство»**, представляющий физическое или логическое устройство в системе и описывающий его характеристики, например, границы выравнивания буферов и адреса очередей для приема IRP, поступающих на это устройство.

Диспетчер ввода-вывода создает объект «драйвер» при загрузке в систему соответствующего драйвера и вызывает его инициализирующую процедуру (например, ***DriverEntry)***, которая записывает в атрибуты объекта точки входа этого драйвера.

После загрузки драйвер может создавать объекты «устройство» для представления устройств или даже для формирования интерфейса драйвера (вызовом *IoCreateDevice).* Однако большинство WDM-драйверов создают объекты «устройство» с помощью своих процедур добавления устройств, когда диспетчер PnP информирует их о присутствии управляемого ими устройства. С другой стороны, унаследованные драйверы создают объекты «устройство» при вызове

диспетчером ввода-вывода их инициализирующих процедур. Диспетчер ввода-вывода выгружает драйвер после удаления его последнего объекта «устройство», когда ссылок на устройство больше нет.

Создавая объект «устройство», драйвер может присвоить ему имя. Тогда объект «устройство» помещается в пространство имен диспетчера объектов. Драйвер может определить имя этого объекта явно или позволить диспетчеру ввода-вывода сгенерировать его автоматически По соглашению, объекты «устройство» помещаются в каталог пространства имен \Device, недоступный приложениям через Win32 API.

ПРИМЕЧАНИЕ Некоторые драйверы размещают объекты «устройство» в каталогах, отличных от \Device. Так, диспетчер томов Logical Disk Manager создает объекты «устройство», представляющие разделы жесткого диска, в каталоге \Device\HarddiskDmVolumes.

Объект **«устройство»** ссылается на свой объект «драйвер», благодаря чему диспетчер ввода-вывода знает, из какого драйвера нужно вызвать процедуру при получении запроса ввода-вывода. С помощью объекта «устройство» он находит объект «драйвер», который представляет драйвер, обслуживающий устройство. После этого он обращается к объекту «драйвер», используя номер функции из исходного запроса; каждый номер функции соответствует точке входа драйвера.

С объектом «драйвер» нередко сопоставляется несколько объектов **«устройство».**

Список объектов «устройство» представляет физические и логические устройства, управляемые драйвером. Так, для каждого раздела жесткого диска имеется отдельный объект «устройство* с информацией, специфичной для данного раздела. Но для обращения ко всем разделам используется один и тот же драйвер жесткого диска.

При выгрузке драйвера из системы диспетчер ввода-вывода с помощью очереди объектов «устройство» определяет устройства, на которые повлияет удаление драйвера.

Использование объектов для регистрации информации о драйверах означает,

что диспетчеру ввода-вывода не нужно знать никаких деталей реализации драйверов. Он просто находит драйвер по указателю, тем самым позволяя легко загружать новые драйверы и обеспечивая их переносимость. Кроме того, представление устройств и драйверов разными объектами упрощает подсистеме ввода-вывода закрепление драйверов за дополнительными устройствами, которые появляются при изменении конфигурации системы.

3.3 Пакеты запросов ввода-вывода

Пакет запроса ввода-вывода (I/O request packet, IRP) хранит информацию, нужную для обработки запроса на ввод-вывод.

Когда поток вызывает сервис ввода-вывода, диспетчер ввода-вывода создает IRP для представления операции в процессе ее выполнения подсистемой ввода-вывода.

После создания и инициализации IRP диспетчер ввода-вывода сохраняет в IRP указатель на объект «файл» вызывающего потока.

4 Технология ввода-вывода

Одной из важных особенностей системы ввода-вывода Windows является ее асинхронность.

Асинхронные сервисы позволяют приложению выдать запрос ввода-вывода и продолжать работу, пока устройство передает данные.

Система ввода-вывода поддерживает **два режима** выполнения операций (синхронные и асинхронные).

Приблизительно одна треть базовых сервисов асинхронны по умолчанию, к их числу относятся такие длительные операции, как чтение и запись файла, а также просмотр содержимого каталога.

Для того, чтобы использовать режим, противоположный заданному по умолчанию, поток должен явно указать его при вызове операции (в терминологии Win32 асинхронный ввод-вывод называется совмещенным). Когда придет время синхронизации с завершением операции ввода-вывода, поток может выполнить вызов **ждать <дескриптор>**, указав в качестве

параметра дескриптор файла.

Однако независимо от того, какой режим ввода-вывода выберет приложение, сама система ввода-вывода работает полностью асинхронно. Для выполнения асинхронных запросов в операционной системе предусмотрен специальный процесс с несколькими рабочими потоками.

4.1 Типы ввода-вывода

Синхронный

Асинхронный

Проецируемый

Быстрый

4.2 Синхронный ввод-вывод

Большинство операций ввода-вывода приложений является *синхронным,* т. е. приложение ждет, когда устройство выполнит передачу данных и вернет код статуса по завершении операции ввода-вывода. После этого программа продолжает работу и немедленно использует полученные данные. В такой простейшем варианте Win32-функции ***ReadFile*** и ***WriteFile*** выполняются синхронно. Перед возвратом управления они должны завершить операцию ввода-вывода.

Обработка синхронного запроса ввода-вывода

1. Запрос на ввод-вывод передается через DLL подсистемы.
2. DLL подсистемы вызывает сервис ***NtWriteFile*** диспетчера ввода-вывода.
3. Диспетчер ввода-вывода создает IRP, описывающий запрос, и посылает его драйверу (в данном случае - драйверу устройства), вызывая свою функцию ***loCallDriver***.
4. Драйвер передает данные из IRP на устройство и инициирует операцию ввода-вывода.
5. Драйвер уведомляет о завершении ввода-вывода, генерируя прерывание.
6. Когда устройство завершает операцию и вызывает прерывание, драйвер устройства обслуживает прерывание.
7. Драйвер вызывает функцию ***IoCompleteRequest*** диспетчера ввода-вывода, чтобы

уведомить его о завершении обработки IRE, и диспетчер ввода-вывода завершает данный запрос на ввод-вывод.

4.3 Асинхронный ввод-вывод

Асинхронный ввод-вывод позволяет приложению выдать запрос на ввод-вывод и продолжить выполнение, не дожидаясь передачи данных устройством. Этот тип ввода-вывода увеличивает эффективность работы приложения, позволяя заниматься другими задачами, пока выполняется операция ввода-вывода.

Для использования асинхронного ввода-вывода, необходимо указать при вызове ***CreateFile*** флаг **FILE__FLAG_OVERLAPPED**.

Инициировав операцию асинхронного ввода-вывода, поток должен соблюдать осторожность и не обращаться к запрошенным данным до их получения от устройства. Следовательно, поток должен синхронизировать свое выполнение с завершением обработки запроса на ввод-вывод, отслеживая описатель синхронизирующего объекта (которым может быть событие, порт завершения ввода-вывода или сам объект «файл»), который по окончании ввода-вывода перейдет в свободное состояние.

Независимо от типа запроса все операции ввода-вывода, представляемые IRP, самой системой выполняются асинхронно: после инициации запроса драйвер устройства возвращает управление подсистеме ввода-вывода. А когда она вернет управление приложению, зависит от типа запроса.

Пример прохождения запроса на асинхронный ввод-вывод через многоуровневые драйверы (пример относится к диску, управляемому файловой системой).

1. Диспетчер ввода-вывода получает запрос, создает IRP для его представления, но на этот раз передает пакет драйверу файловой системы. С этого момента драйвер файловой системы в основном и управляет операцией ввода-вывода. В зависимости от типа запроса файловая система посылает драйверу диска тот же IRP или генерирует дополнительные IRP и передает их этому драйверу по отдельности.
2. Файловая система скорее всего будет повторно использовать IRP, если

полученный запрос можно преобразовать в единый запрос к устройству. Например, если приложение выдаст запрос на чтение первых 512 байт из файла на дискете, файловая система FAT просто вызовет драйвер диска, попросив его считать один сектор с того места на дискете, где начинается нужный файл.

Для поддержки использования несколькими драйверами IRP содержит набор блоков стека (не путать со стеком потока). Эти блоки данных - по одному на каждый вызываемый драйвер - хранят информацию, необходимую каждому драйверу для обработки своей части запроса (например, номер функции, параметры, сведения о контексте драйвера).

По мере передачи IRP от одного драйвера другому заполняются дополнительные блоки стека. IRP можно считать аналогом стека в отношении добавления и удаления данных. Но IRP не сопоставляется ни с каким процессом, и его размер фиксирован.

После того как драйвер диска завершает передачу данных, диск генерирует прерывание, и ввод-вывод завершается.

4.4 Проецируемый ввод вывод

Файловый ввод-вывод с отображением в память осуществляется совместно системой ввода-вывода и менеджером виртуальной памяти. Расположенный на диске файл может рассматриваться как часть виртуальной памяти процесса, поэтому, например, для записи данных в файл процессу достаточно будет записать их по соответствующему адресу в памяти. Эти изменения будут записаны диспетчером виртуальной памяти на диск в ходе обычной операции откачки страниц.

При осуществлении файлового ввода-вывода с отображением в память производительность приложений, которые выполняют значительный объем ввода-вывода или работают с большим числом файлов, может быть повышена. Этого нетрудно достичь благодаря двум факторам: во-первых, запись в память выполняется гораздо быстрее записи на диск, а во-вторых, диспетчер виртуальной памяти оптимизирует свой доступ к диску.

4.5 Быстрый ввод-вывод

Быстрый ввод-вывод (fast I/O) — специальный механизм, который позволяет подсистеме ввода-вывода напрямую, не генерируя IRP, обращаться к драйверу файловой системы или диспетчеру кэша (быстрый ввод-вывод описывается в главах 11 и 12). Драйвер регистрирует свои точки входа для быстрого ввода-вывода, записывая их адреса в структуру, на которую ссылается указатель PFAST_IO_DISPATCH его объекта «драйвер».