Задачи операционной системы по управлению файлами и устройствами. Многослойная модель подсистемы ввода-вывода. Менеджер ввода-вывода. Аппаратные и многоуровневые драйверы. Блок-ориентированные и байт-ориентированные драйверы. Подсистема ввода-вывода Windows. Обработка типичного запроса ввода-вывода в Windows. Типы драйверов Windows.

Одной из основных задач ОС является обеспечение обмена данными между приложениями и периферийными устройствами. В современной ОС функции обмена данными с периферийными устройствами выполняет подсистема ввода-вывода (Input-Output Subsystem).

Основными компонентами этой подсистемы ввода-вывода являются драйверы, управляющие внешними устройствами, и файловая система. К подсистеме ввода-вывода можно также с некоторой долей условности и диспетчер прерываний, рассмотренный в Лекции 6.

Подсистема ввода-вывода при обмене данными с внешними устройствами компьютера должна решать ряд задач, из которых наиболее важными являются:

- организация параллельной работы устройства ввода-вывода и процессора;
- согласование скоростей обмена и кэширование данных;
- разделение устройств и данных между процессами;
- обеспечение удобного логического интерфейса между устройствами и и остальной частью системы;
- поддержка широкого спектра драйверов с возможностью простого включения в систему нового драйвера;
- динамическая загрузка и выгрузка драйверов;
- поддержка нескольких файловых систем;
- поддержка синхронных и асинхронных операций ввода-вывода.

Рассмотрим эти задачи.

Организация параллельной работы устройств ввода-вывода и процессора.Каждое устройство ввода-вывода вычислительной системы снабжено специализированным блоком управления, называемым *контроллером*. Контроллер взаимодействует с *драйвером*– системным программным модулем, предназначенным для управления данным устройством. Контроллер периодически принимает от драйвера выводимую на это устройство информацию, а также команды управления, которые говорят, что с этой информацией нужно сделать.

Под управлением контроллера устройство может некоторое время выполнять свои операции автономно, не требуя внимания со стороны центрального процессора. Процессы, происходящие в контроллерах, протекают в периоды между выдачами команд независимо от ОС. От подсистемы ввода-вывода требуется спланировать в реальном масштабе времени (в котором работают внешние устройства) запуск и приостановку большого количества разнообразных драйверов, обеспечив приемлемое время реакции каждого драйвера на независимые события контроллера. С другой стороны, необходимо минимизировать загрузку процессора задачами ввода-вывода, оставив как можно больше процессорного времени на выполнение пользовательских потоков.

Данная задача является классической задачей планирования систем реального времени и обычно решается на основе многоуровневой приоритетной схемы обслуживания по прерываниям. Для реализации этой схемы обычно задействуется общий диспетчер прерываний.

Согласование скоростей обмена и кэширование данных. При обмене данными всегда возникает задача согласования скорости. В подсистеме ввода-вывода для согласования скоростей используется буферизация данных в оперативной памяти. Однако буферизация только на основе оперативной памяти в подсистеме ввода-вывода оказывается недостаточной – разница между скоростью обмена с оперативной памятью, куда процессы помещают данные для обработки, и скоростью работы внешнего устройства часто становится слишком значительной, чтобы в качестве временного буфера можно было использовать оперативную память – ее объема может просто не хватить. В таких случаев в качестве буфера используют дисковый файл, называемый также *спул-файлом* (otspool– шпулька). Типичный пример применения спулинга дает организация вывода данных на принтер.

Другим решением этой проблемы является использование большой буферной памяти в контроллерах внешних устройств. Такой подход особенно полезен в тех случаях, когда помещение данных на диск слишком замедляет обмен (или когда данные выводятся на сам диск).

Буферизация позволяет решать и другую задачу – сократить количество реальных операций ввода-вывода за счет кэширования данных.

Разделение устройств и данных между процессами. Устройства ввода-вывода могут предоставляться процессам как в монопольное, так и в разделяемое использование. В разные моменты времени одно и то же устройство может использоваться как в разделяемом, так и в монопольном режимах. При этом ОС должна обеспечить контроль доступа теми же способами, что и при доступе процессов к другим ресурсам вычислительной системы – путем проверки прав пользователей или группы пользователей, от

имени которых действует процесс, на выполнение той или иной операции над устройством.

Операционная система может контролировать доступ не только к устройству в целом, но и к отдельным порциям данных, хранимых или отображаемых этим устройством. Диск является типичным примером устройства, для которого важно контролировать доступ не к устройству в целом, а к отдельным каталогам и файлам. При этом каждой порции данных или части устройства могут быть заданы свои права доступа, не связанные прямо с правами доступа ко всему устройству в целом.

При разделении устройства между процессами может возникнуть необходимость в разграничении порции данных двух процессов друг от друга. Обычно такая потребность возникает при совместном использовании так называемых последовательных устройств, данные в которых в отличие от устройств прямого доступа не адресуются. Типичным представителем такого рода устройств является принтер, который не выделяется в монопольное пользование процессам, и в то же время каждый документ должен быть напечатан в виде последовательного набора страниц. Для таких устройств организуется очередь заданий на вывод, при этом каждое задание представляет собой порцию данных, которые нельзя разрывать.

Обеспечение удобного логического интерфейса между устройствами и остальной частью системы. Разнообразие устройств ввода-вывода делают особенно актуальной функцию ОС по созданию экранирующего логического интерфейса между периферийными устройствами и приложениями. Практически все современные операционные системы поддерживают в качестве основы такого интерфейса файловую модель периферийных устройств, когда любое устройство выглядит для прикладного программиста последовательным набором байт, с которым можно работать с помощью унифицированных системных вызовов (например, read и write), задавая имя файла-устройства и смещение от начала последовательности байт. Для поддержания такого интерфейса подсистема ввода-вывода должна учитывать разницу в организации операций обмена данными, например, с жестким диском и графическим терминалом.

Поддержка широкого спектра драйверов и простота включения нового драйвера в систему. Достоинством подсистемы ввода-вывода любой универсальной ОС является наличие разнообразного набора драйверов для наиболее популярных периферийных устройств. Чтобы операционная система не испытывала недостатка в драйверах, необходимо наличие четкого, удобного и открытого интерфейса между драйверами и другими компонентами операционной системы. Такой интерфейс нужен для того, чтобы драйверы писали не только разработчики операционной системы, но

программисты предприятий, которые выпускают внешние устройства для компьютеров.

Драйвер взаимодействует, с одной стороны, с модулями ядра ОС (модулями подсистемы ввода-вывода, системных вызовов, подсистем управления процессами и памятью и т.д.), а с другой стороны – с контроллерами внешних устройств. Поэтому существуют два типа интерфейсов: интерфейс *«драйвер-ядро» (Driver Kernel Interface, DKI)* и интерфейс *«драйвер-устройство» (Driver Device Interface, DDI)*. DKI должен быть стандартизован в любом случае, а DDI имеет смысл стандартизировать тогда, когда подсистема ввода-вывода не разрешает драйверу непосредственно взаимодействовать с аппаратурой контроллера, а выполняет эти операции самостоятельно. Экранирование драйвера полезно, так как драйвер в этом случае становится независимым от аппаратной платформы. Подсистема ввода-вывода может поддерживать несколько различных типов интерфейсов DKI/DDI, предоставляя специфический интерфейс для устройств определенного класса.

Для поддержки процесса разработки драйверов операционной системы обычно выпускается так называемый пакет *DDK (Driver Development Kit)*, представляющий собой набор соответствующих инструментальных средств – библиотек, компиляторов и отладчиков.

Динамическая загрузка и выгрузка драйверов. Кроме проблемы разработки новых драйверов существует также проблема включения драйвера в состав модулей работающей ОС, то есть динамической загрузки-выгрузки драйвера. Так как выбор потенциально поддерживаемых данной ОС периферийных устройств всегда существенно шире набора устройств, которыми ОС должна управлять при установке на конкретной машине, ценным свойством ОС является возможность динамически загружать в память требуемый драйвер без останова ОС и выгружать его после того, как потребность в поддержке устройства отпадает, что может существенно сэкономить системную область памяти.

Альтернативой динамической загрузке драйверов является повторная компиляция кода ядра с требуемым набором драйверов, что создает между всеми компонентами ядра статические связи вместо динамических.

Поддержка нескольких файловых систем. Диски представляют особый род периферийных устройств, так как именно на них хранится большая часть как пользовательских, так и системных данных. Данные на дисках организуются в файловые системы, и свойства файловой системы во многом определяют свойства самой ОС – ее быстродействие, отказоустойчивость, максимальный объем хранимых данных. Популярность файловой системы часто приводит к ее миграции из «родной» ОС в другие операционные системы. Ввиду этого

поддержка нескольких популярных файловых систем для подсистемы ввода-вывода очень важна. Важно также, чтобы архитектура подсистемы ввода-вывода позволяла достаточно просто включать в ее состав новые типы файловых систем.

Поддержка синхронных и асинхронных операций ввода-вывода. Операция ввода-вывода может выполняться по отношению к программному модулю, запросившему операцию, в синхронном или асинхронном режимах. Подсистема ввода-вывода должна предоставлять своим клиентам (пользовательским процессам и кодам ядра) возможность выполнять как синхронные, так и асинхронные операции ввода-вывода, в зависимости от потребностей вызывающей стороны. Системные вызовы ввода-вывода чаще оформляются как синхронные процедуры в связи с тем, что такие операции длятся долго и пользовательскому процессу или потоку все равно придется ждать получения результатов операции для того, чтобы продолжить свою работу. Внутренние же вызовы операций ввода-вывода из модулей ядра обычно выполняются в виде асинхронных процедур, так как кодам ядра нужна свобода в выборе дальнейшего поведения после запроса операции ввода-вывода. Использование асинхронных процедур приводит к более гибким решениям, так как на основе асинхронного вызова всегда можно построить синхронный, создав промежуточную процедуру, блокирующую выполнение вызвавшей процедуры до момента завершения ввода-вывода.

При построении систем ввода-вывода используется принцип многослойного построения программного обеспечения. При большом разнообразии устройств ввода-вывода, обладающими существенно различными характеристиками, иерархическая структура программного обеспечения позволяет соблюсти баланс между двумя противоречивыми требованиями: с одной стороны, необходимо учесть все особенности каждого устройства, а с другой стороны, обеспечить единое логическое представление и унифицированный интерфейс для устройств всех типов. При этом нижние слои подсистемы ввода-вывода должны включать индивидуальные драйверы, написанные для конкретных физических устройств, а верхние должны обобщать процедуры управления этими устройствами.

Обобщенная структура подсистемы ввода-вывода представлена на рис. 10.1. Видно, что программное обеспечение ввода-вывода делится как на горизонтальные слои, так и на вертикальные. Это объясняется тем, что для всего разнообразия внешних устройств трудно обеспечить единообразие в разбиении функций управления на слои. Поэтому общий принцип многослойности остается справедливым, однако для конкретного типа устройств он реализуется по-разному. В представленной структуре в качестве примера приведены три вертикальные подсистемы, управляющие дисками, графическими устройствами и сетевыми адаптерами.

В подсистеме ввода-вывода наряду с модулями, отражающими специфику внешних устройств и образующими вертикальные подсистемы, существуют модули универсального назначения. Эти модули организуют согласованную работу всех остальных компонентов подсистемы ввода-вывода и взаимодействие с пользовательскими процессами и другими подсистемами ОС. Эти организующие модули распределены по всем уровням, образуя *оболочку*, которую иногда называют *менеджером ввода-вывода*.

Верхний слой менеджера составляют системные вызовы ввода-вывода, которые принимают от пользовательских процессов запросы на ввод-вывод и переадресуют их отвечающим за определенный класс устройств модулям и драйверам, а также возвращают процессам результаты операций ввода-вывода. Таким образом этот слой поддерживает пользовательский интерфейс ввода-вывода.

Нижний слой менеджера реализует непосредственное взаимодействие с контроллерами внешних устройств, экранируя драйверы от особенностей аппаратной платформы компьютера – шины ввода-вывода, системы прерываний и т.п. Этот слой принимает от драйверов запросы на обмен данными с регистрами контроллеров в некоторой обобщенной форме с использованием независимых от шины адресации и формата, а затем преобразует эти запросы в зависящий от аппаратной платформы формат. Диспетчер прерываний, рассмотренный в Лекции 6, может входить в состав менеджера ввода-вывода или же представлять отдельный модуль ядра.

Важной функцией менеджера ввода-вывода является создание некоторой среды для остальных компонентов подсистемы, которая бы облегчала их взаимодействие друг с другом. Эта задача может быть решена за счет создания некоторого внутреннего интерфейса взаимодействия модулей ввода-вывода между собой. Наличие такого интерфейса существенно облегчает включение новых драйверов и файловых систем в состав ОС. Кроме того, разработчики драйверов и других программных компонентов освобождаются от написания общих процедур, таких как буферизация данных и синхронизация нескольких модулей между собой при обмене данными. Все эти функции берет на себя менеджер ввода-вывода.

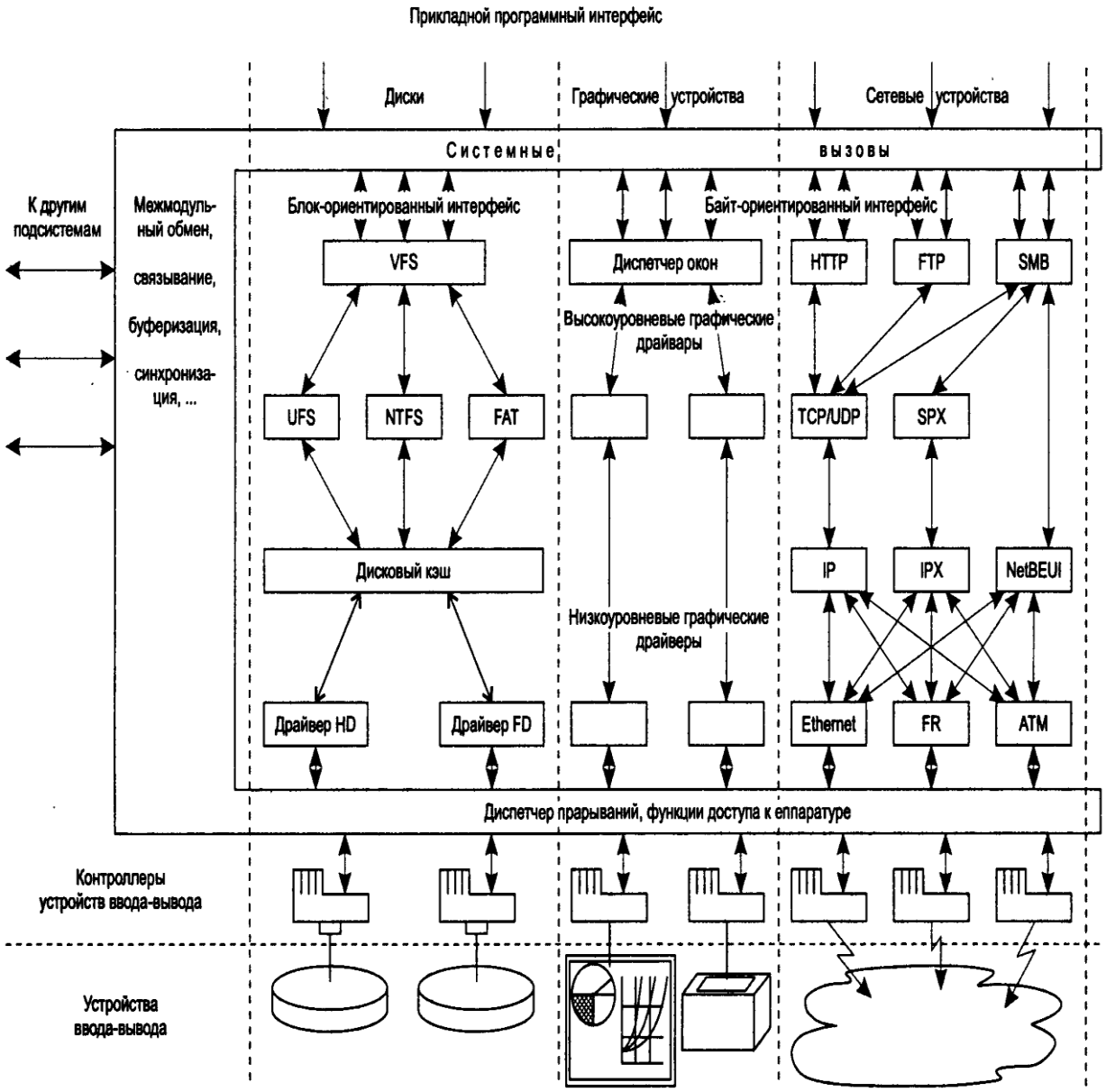

Рис. 10.1. Структура подсистемы ввода-вывода

Еще одной функцией менеджера ввода-вывода является организация взаимодействия модулей ввода-вывода с модулями других подсистем ОС, таких, как подсистема управления процессами, виртуальной памятью и другими.

В состав любой подсистемы ввода-вывода входят драйверы В традиционном смысле слова *драйвер*– это программный модуль, который:

- входит в состав ядра операционной системы, работая в привилегированном режиме;
- непосредственно управляет внешним устройством, взаимодействуя с его контроллером с помощью команд ввода-вывода компьютера;
- обрабатывает прерывания от контроллера устройства;

- предоставляет прикладному программисту удобный логический интерфейс работы с устройством, экранируя от него низкоуровневые детали управления устройством и организации его данных;
- взаимодействует с другими модулями ядра ОС с помощью строго оговоренного интерфейса, описывающего формат передаваемых данных, структуру буферов, способы включения драйвера в состав ОС, способы вызова драйвера, набор общих процедур подсистемы ввода-вывода, которыми драйвер может пользоваться, и т.п.

Наряду с традиционными драйверами в операционных системах имеются так называемые *высокоуровневые драйверы*, которые располагаются в общей модели ввода-вывода над традиционными драйверами, которые еще называются *аппаратными, низкоуровневыми драйверами* или *драйверами устройств.* Применение высокоуровневых драйверов повышает гибкость и расширяемость функций по управлению устройством – вместо жесткого набора функций, сосредоточенных в единственном драйвере, администратор системы может выбрать требуемый набор функций, установив нужный высокоуровневый драйвер.

Высокоуровневые драйверы оформляются по тем же правилам и придерживаются тех же внутренних интерфейсов, что и низкоуровневые драйверы. Единственным отличием является то, что высокоуровневые драйверы, как правило, не вызываются по прерываниям, так как взаимодействуют с управляемым устройством через посредничество аппаратных драйверов. Менеджер ввода-вывода управляет всеми драйверами однотипно, независимо от их уровня.

Например, в подсистеме управления дисками аппаратные драйверы поддерживают для верхних уровней представление диска как последовательного набора блоков одинакового размера, преобразуя вместе с контроллером номер блока в более сложный адрес, состоящий из номеров цилиндра, головки и сектора. Однако понятие «файл» и «файловая система» аппаратные драйверы не поддерживают. Эти логические абстракции создаются на более высоком уровне программным обеспечением файловых систем, которое в современных ОС оформляется как высокоуровневый драйвер.

В унификацию драйверов большой вклад внесла операционная система UNIX. В ней все драйверы были разделены на два больших класса: *блок-ориентированные (block-oriented) драйверы* и *байт-ориентированные (character-oriented) драйверы*. Блок-ориентированные драйверы управляют устройствами прямого доступа, которые хранят информацию в блоках фиксированного размера, каждый из которых имеет свой собственный адрес. Самое распространенное внешнее устройство прямого доступа – диск.

Устройства, с которыми работают байт-ориентированные драйверы, не адресуемы и не позволяют производить операцию поиска данных, они генерируют или потребляют последовательности байт. Примерами таких устройств служат терминалы, сетевые адаптеры.

Блок- и байт-ориентированность является характеристикой как самого устройства, так и драйвера. Для байт-ориентированного устройства невозможно разработать блок-ориентированный драйвер. Однако блок-ориентированным устройством можно управлять с помощью байт-ориентированного драйвера.

Деление драйверов на блок- и байт-ориентированные оказывается полезным для структурирования подсистемы управления вводом-выводом. Тем не менее необходимо учитывать, что эта схема является упрощенной – имеются внешние устройства, драйверы которых не относятся ни к одному классу. В качестве примера можно привести таймер, который, с одной стороны, не содержит адресуемой информации, а с другой стороны, не порождает потока байт.

Подсистема ввода-вывода в Windows состоит из нескольких компонентов исполнительной системы и драйверов устройств (рис. 10.2).

- Центральное место в подсистеме ввода-вывода занимает *диспетчер ввода-вывода*; от 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 устройства конкретного типа. Такие драйверы принимают от диспетчера ввода-вывода команды, предназначенные управляемым ими устройствам, уведомляет 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 драйвером шины (busdriver) – одной из разновидностей драйверов устройств. Он управляет выделением аппаратных ресурсов, а также распознает устройства и реагирует на их подключение. Диспетчер PnP и драйверы шины 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.

- *Диспетчер электропитания*, также в тесном взаимодействии с диспетчером ввода-вывода, управляет системой и драйверами устройств при их переходе в различные состояния энергопотребления.
- *Процедуры поддержки Windows Management Instrumentation* (WMI, Инструментарий управления Windows), образующие провайдер WDM (WindowsDriverModel) WMI, позволяют драйверам устройств выступать в роли провайдеров, взаимодействуя со службой WMI пользовательского режима через провайдер WDM WMI.
- *Реестр* служит в качестве базы данных, в которой хранится описание основных устройств, подключенных к системе, а также параметры инициализации драйверов и конфигурационные настройки.
- Для установки драйверов используются *INF-файлы*; они связывают конкретное аппаратное устройство с драйвером, который берет на себя ведущую роль в управлении этими устройством. Содержимое INF-файла состоит из инструкций, описывающих соответствующее устройство, исходное и целевое местонахождение файлов драйвера, изменения, которые нужно внести в реестр при установке драйвера, и информацию о зависимостях драйвера. В CAT-файлах хранятся цифровые подписи, которые удостоверяют файлы драйверов, прошедших испытания в лаборатории Microsoft Windows Hardware Quality Lab.
- *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.

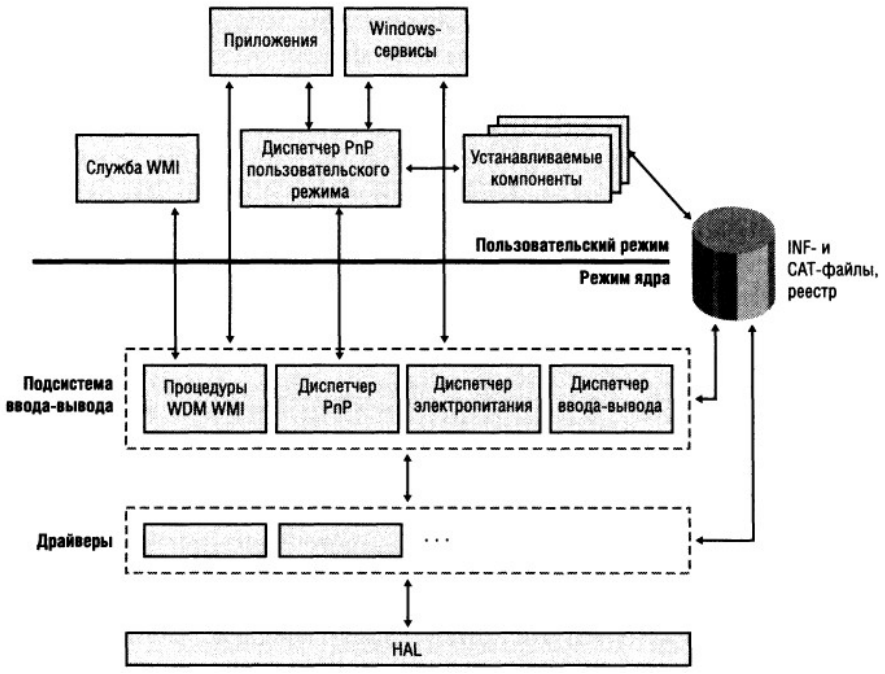

Рис. 10.2. Компоненты подсистемы ввода-вывода.

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. 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 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. Базовая схема обработки запроса на ввод-вывод показана на рис. 10.3.

В Windows драйверы могут работать в двух режимах: в пользовательском и в режиме ядра. Windows поддерживает несколько типов драйверов пользовательского режима:

- *Драйверы виртуальных устройств (virtual device drivers, VDD).* Используются для эмуляции 16-разрядных программ MS-DOS. Они перехватывают обращения таких программ к портам ввода-вывода, передаваемые реальным драйверам устройств. Поскольку Windows является полностью защищенной операционной системой, программы MS-DOS пользовательского режима не могут напрямую обращаться к аппаратным средствам – они должны это делать через драйверы устройств режима ядра.
- *Драйверы принтеров.* Драйверы подсистемы Windows, которые транслируют аппаратно-независимые запросы на графические операции в команды, специфичные для принтера. Далее эти команды обычно направляются драйверу режима ядра, например драйверу параллельного порта или драйверу порта принтера на USB-шине.

Рис. 10.3. Схема обработки типичного запроса на ввод-вывод в Windows

Драйверы режима ядра можно разбить не несколько основных категорий:

- *Драйверы файловой системы.* Принимают запросы на ввод-вывод и выполняют их, выдавая более специфические запросы драйверам устройств массовой памяти или сетевым драйверам.
- *PnP драйверы* .Драйверы, работающие с оборудованием и интегрируемые с диспетчерами электропитания и PnP. В их число входят драйверы для устройств массовой памяти, видеоадаптеров, устройств ввода и сетевых адаптеров.
- *Драйверы, не отвечающие спецификации Plug and Play.* Также называются расширениями ядра. Расширяют функциональность системы, предоставляя доступ из пользовательского режима к сервисам и драйверам режима ядра. Они не интегрируются с диспетчерами PnP и электропитания. К ним, в частности, относятся драйверы протоколов и сетевого API.