Архитектура драйверов Windows спроектирована для поддержки четырех ключевых возможностей:

* **Модульная многоуровневая архитектура**. Устройство могут обслуживать несколько драйверов, организованных в стек.
* **Пакетный механизм ввода/вывода**. Все запросы обрабатываются в пакетах, которые передаются между системой и драйвером и между драйверами в стеке.
* **Асинхронный ввод/вывод**. Драйверы могут начать обрабатывать другие запросы, не ожидая завершения обработки текущего запроса.
* **Динамическая загрузка и выгрузка**. Время жизни драйвера привязано к времени жизни устройства. Драйвер выгружается только после того, как все связанные устройства были удалены.

Каждый драйвер имеет ассоциированный с ним объект устройства (NB: правильнее сказать, что каждый объект устройства имеет ассоциированный с ним драйвер). Объект устройства – это структура данных, содержащая указатели на рабочие процедуры, которые позволяют менеджеру ввода/вывода взаимодействовать с драйвером. Объекты устройств организованы в стек устройства, при этом для каждого устройства создается отдельный стек. Обычно под термином «стек устройства» понимается стек объектов устройства с их связанными драйверами. Сам по себе стек устройства ассоциируется только с одним устройством, в то время как набор драйверов может обслуживать несколько стеков. Набор драйверов иногда называют стеком драйверов.

Стек устройства состоит из следующих компонентов.

* **Драйвер шины и объект физического устройства**. Внизу стека находится объект физического устройства (в дальнейшем PDO – Physical Device Object), который ассоциирован с драйвером шины. Устройства обычно подключаются к стандартным аппаратным шинам, таким как, например, PCI или USB. Драйвер шины обычно управляет несколькими аппаратными устройствами, подключенными к физической шине.   
   Например, после установки драйвер шины перечисляет устройства, подключенные к шине, и запрашивает ресурсы для этих устройств. PnP-менеджер использует эту информацию для выделения ресурсов каждому устройству. Каждое устройство имеет собственный объект PDO. PnP-менеджер идентифицирует драйверы для каждого устройства и создает соответствующий стек устройства над PDO.
* **Драйвер функции и объект функционального устройства**. Основным компонентом стека устройств является объект функционального устройства (в дальнейшем FDO – Functional Device Object), который ассоциируется с драйвером функции. Драйвер функции преобразует абстракцию устройства, создаваемую Windows в конкретные команды, необходимые для обмена данными с физическим устройством. Он предоставляет так называемый «верх» (иными словами, интерфейс устройства) для взаимодействия с приложениями и устройствами, и обычно определяет, каким образом драйвер реагирует на изменения в состоянии Plug and Play или энергопотребления. Так называемый «низ» драйвера обрабатывает взаимодействие с устройством или другими драйверами, такими, как, например, нижестоящий драйвер фильтра или драйвер шины.
* **Драйверы фильтров и объекты фильтра устройства**. Стек устройства может иметь несколько объектов фильтров устройства (в дальнейшем FiDO – Filter Device Object), которые могут располагаться вокруг объекта FDO. С каждым объектом FiDO ассоциируется драйвер фильтра. Драйверы фильтров не являются обязательными, но часто присутствуют. Различные производители могут снабжать стек устройства драйверами фильтров с целью придания ему дополнительных возможностей. Обычно драйвер фильтра служит для модифицирования некоторых запросов ввода/вывода, проходящих по стеку, подобно тому, как аудиофильтр может изменять сырой аудиопоток.   
   Например, с помощью драйверов фильтра можно зашифровывать или расшифровывать при запросах на чтение и запись. Драйверы фильтров можно также применять для выполнения задач, не требующих модификации траффика, например, таких как отслеживание запросов и предоставление информации обратно отслеживающему приложению (NB: очень похоже на паттерн «декоратор»).

Эти три типа объектов отличаются в деталях, но работают по обеспечению системы возможностями обработки запросов ввода/вывода в основном одинаково.

**Дерево устройств.**

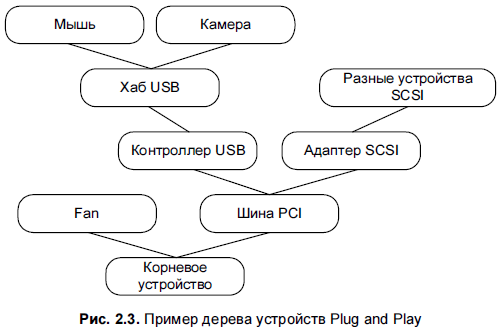
Шины вычислительных систем обычно организованы в несколько уровней. К шине самого низкого уровня могут быть подключены отдельные устройства, или одна или более других шин, к котором, в свою очередь, могут подключаться другие устройства и шины. Поэтому большинство драйверов шин должны играть две роли:

1. Драйвер функции шины.
2. Перечислитель и менеджер шины.

При загрузке системы PnP-менеджер начинает работу с шины самого низкого уровня и выполняет следующие действия:

1. Загружает драйвер шины, который перечисляет объекты PDO для всех устройств, подключенных к ней, и запрашивает ресурсы для этих устройств (NB: запрашивает ресурсы драйвер шины).
2. Выделяет ресурсы каждому устройству (NB: выделяет ресурсы PnP-менеджер).
3. Определение соответствия драйвера и устройства (ассоциирование) на основе своей базы данных.
4. Создает стек устройства поверх каждого PDO.
5. Запускает все устройства.
6. Опрашивает каждое устройство на возможное наличие у него объектов PDO для перечисления (NB: т.е. устройство может быть составным?).
7. При необходимости повторяет шаги 2-6.

В результате этого процесса формируется иерархическая структура, которая называется деревом устройств. Каждый стек устройства представлен в дереве устройств узлом, который называется devnode. Узел devnode устройства используется PnP-менеджером для хранения информации о конфигурации и отслеживания устройства. Фундамент дерева устройств – это абстрактное устройства, называемое корневым устройством. Пример дерева устройств показан на рисунке ниже:



**Модель ввода/вывода Windows.**

Модель ввода/вывода Windows определяет, каким образом система и связанные драйверы обрабатывают запросы ввода/вывода. Модель охватывает не только обмен данными с устройствами, она также работает как общий пакетный механизм взаимодействия между клиентами, выдающими запросы к стеку устройства. Клиентами являются приложения, подсистемы ядра, такие как PnP-менеджер, и сами драйверы.

Все запросы ввода/вывода Windows исполняются посредством пакетов IRP. Кроме исполнения запросов, связанных с вводом/выводом, которые приложения посылают драйверам (например, запросы на чтение, запись, создание и т.д.), пакеты IRP также применяются для исполнения запросов Plug and Play, управления энергопотреблением и интерфейса WMI (Windows Management Instrumentation – инструментарий управления Windows), а также для передачи извещений об изменениях состояния устройств и обращений за информацией о ресурсах устройств и драйверов.

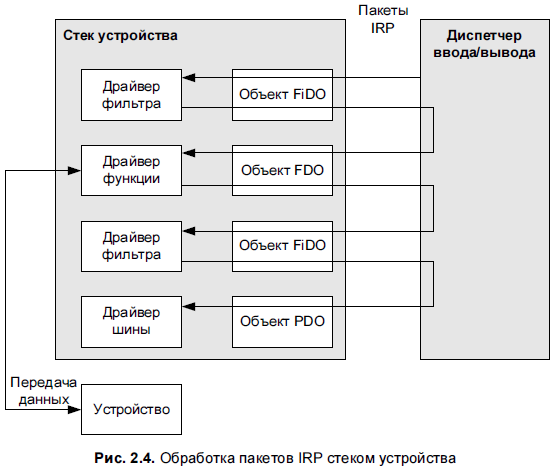
Наиболее распространенными типами запросов ввода/вывода являются следующие.

* **Запросы на запись**. Эти запросы передают драйверу данные для записи в устройство.
* **Запросы на чтение**. Эти запросы передают драйверу буфер для заполнения данными из устройства.
* **Запросы управления вводом/выводом устройства (IOCTL)**. Эти запросы применяются для взаимодействия с драйверами иного характера, нежели чтение или запись данных. С некоторыми устройствами запросы управления вводом/выводом являются наиболее применяемым типом запросов.

Приложения выдают эти запросы ввода/вывода посредством дескрипторов устройств и вызывая соответствующую функцию Windows API: WriteFile, ReadFile и DeviceIoControl.

**Обработка запросов стеком устройства.**

Когда клиент посылает устройству запрос ввода/вывода, менеджер ввода/вывода упаковывает запрос и связанную информацию, например буферы данных, в пакет IRP. Менеджер ввода/вывода находит указатель на соответствующую рабочую процедуру в верхнем объекте стека устройства и вызывает данную процедуру, после чего передает пакет IRP соответствующему драйверу для обработки. На рисунке 2.4 показана схема обработки пакетов IRP стеком устройства.



Если верхний драйвер может удовлетворить данный IRP, он дает инструкцию менеджеру ввода/вывода выполнить его. Это действие возвращает пакет IRP менеджеру ввода/вывода, который, в свою очередь, передает запрошенные данные обратно клиенту. Но драйвер не всегда может удовлетворить запрос собственными силами. В таком случае драйвер обрабатывает пакет IRP должным образом и потом указывает менеджеру ввода/вывода передать пакет IRP следующему драйверу вниз по стеку и т.д. Например, драйвер фильтра может модифицировать пакет IRP на запись, после чего передать его вниз по стеку, чтобы драйвер функции мог передать данные устройству и завершить запрос. Иногда пакеты IRP приходится передавать до самого драйвера шины. В конце концов, запрос достигает того драйвера, который может удовлетворить и завершить его. Для удовлетворения запроса драйверам часто необходимо взаимодействовать с устройством или напрямую через его регистры, или косвенно, посредством драйвера протокола нижнего уровня. Иногда это взаимодействие осуществляется быстро, и драйвер может немедленно возвратить клиенту запрошенные данные. Но чаще при взаимодействии с устройством проходит много времени (исчисляемого в циклах центрального процессора), прежде чем устройство ответит на запрос. В таких случаях драйвер может возвратить управление клиенту, оставив запрос в процессе ожидания завершения исполнения, и завершает запрос по окончании процесса взаимодействия.

После завершения отработки запроса может оказаться необходимым выполнить какую-либо дополнительную обработку прежде, чем возвращать результаты клиенту. В таком случае перед тем, как вызвать менеджер ввода/вывода для передачи пакета IRP следующему драйверу, драйвер может установить процедуру завершения ввода/вывода. Когда запрос будет отработан, менеджер ввода вывода вызывает процедуры завершения в порядке, обратном их установке, т.е. «разматывает» стек обратно вверх.

**Типы передачи ввода/вывода.**