**Виртуальные файловые системы**

Ключевая идея состоит в том, чтобы выделить какую-то часть файловой системы, яв- ляющуюся общей для всех файловых систем, и поместить ее код на отдельный уровень, из которого вызываются расположенные ниже конкретные файловые системы с целью фактического управления данными.

Все относящиеся к файлам системные вызовы направляются для первичной обработки в адрес виртуальной файловой системы. Эти вызовы, поступающие от пользователь ских процессов, являются стандартными POSIX-вызовами, такими как *open*, *read*, *write*, *lseek* и т. д. Таким образом, VFS обладает «верхним» интерфейсом к пользовательским процессам, и это хорошо известный интерфейс POSIX.

У VFS есть также «нижний» интерфейс к конкретной файловой системе, который на рис. 4.15 обозначен как VFS-интерфейс. Этот интерфейс состоит из нескольких десят- ков вызовов функций, которые VFS способна направлять к каждой файловой системе для достижения конечного результата. Таким образом, чтобы создать новую файловую систему, работающую с VFS, ее разработчики должны предоставить вызовы функций, необходимых VFS. Вполне очевидным примером такой функции является функция, считывающая с диска конкретный блок, помещающая его в буферный кэш файловой системы и возвращающая указатель на него. Таким образом, у VFS имеются два ин- терфейса: «верхний» — к пользовательским процессам и «нижний» — к конкретным файловым системам.

Хотя большинство файловых систем, находящихся под VFS, представляют разделы ло- кального диска, так бывает не всегда. На самом деле исходной мотивацией для компании Sun при создании VFS служила поддержка удаленных файловых систем, использующих протокол **сетевой файловой системы** (Network File System (**NFS**)). Конструктивная особенность VFS состоит в том, что пока конкретная файловая система предоставляет требуемые VFS функции, VFS не знает или не заботится о том, где данные хранятся или что собой представляет находящаяся под ней файловая система.

аким образом, добавление файловых систем становится относительно простой зада- чей. Чтобы добавить какую-нибудь новую систему, разработчики берут перечень вы- зовов функций, ожидаемых VFS, а затем пишут свою файловую систему таким образом, чтобы она предоставляла все эти функции. В качестве альтернативы, если файловая система уже существует, им нужно предоставить функции-оболочки, которые делают то, что требуется VFS, зачастую за счет осуществления одного или нескольких вызовов, присущих конкретной файловой системе.