Операционная система как интерфейс между ПО и АО

Программное обеспечение – это все программы, установленные на компьютере, а аппаратное обеспечение – узлы и оборудование, которые находятся внутри системного блока или подключены снаружи.

Взаимосвязь между участниками компьютерной системы называют **интерфейсом**.

Взаимодействие между различными узлами – это **аппаратный интерфейс**, взаимодействие между программами – **программный интерфейс**, а взаимодействие между аппаратурой и программами – **аппаратно-программный интерфейс**.

Согласование между программным и аппаратным обеспечением выполняет операционная система.

Способ взаимодействия человека с программой и программы с человеком называют **интерфейсом пользователя**.

Если программа сделана так, что с ней работать удобно, говорят, что она имеет **удобный интерфейс пользователя.**

Виды интерфейсов

Оболо́чка операцио́нной систе́мы — интерпретатор команд операционной системы, обеспечивающий интерфейс для взаимодействия пользователя с функциями системы.

В общем случае различают оболочки с двумя типами интерфейса для взаимодействия с пользователем:

текстовый пользовательский интерфейс (TUI) и

графический пользовательский интерфейс (GUI).

В каждой операционной системе существует несколько видов интерфейсов:

- **командный (текстовый) интерфейс;**
- **текстовый или графический полноэкранный интерфейс;**
- **графический многооконный пиктографический интерфейс;**
- **интерфейс API.**

В большинстве ОС присутствует унифицированный формат командной строки.
Командная строка включает в себя:

· **Тип операции (имя команды или программы);**

· **Рабочий вход (входные файлы или устройства);**

· **Рабочий выход (выходные файлы или устройства);**

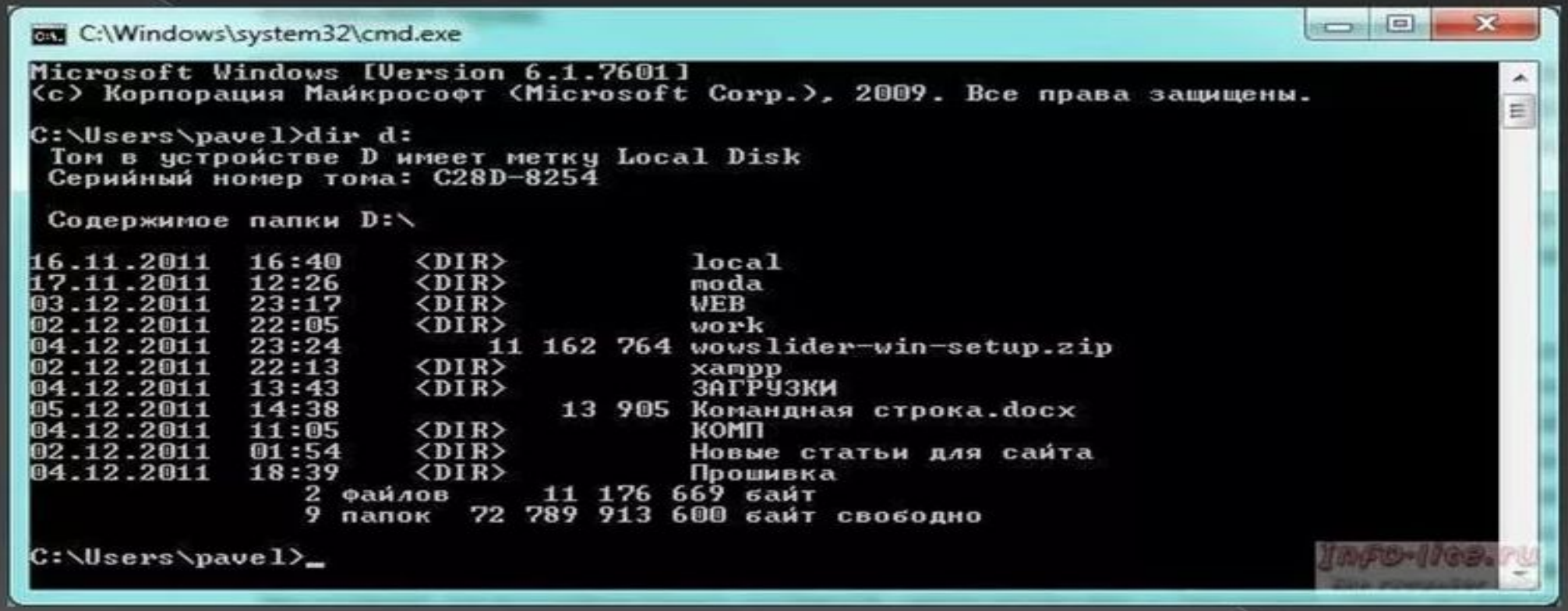
· **Управляющий вход (управляющие параметры или ключи команды);**

· **Управляющий выход (обычно – протокол, содержащий диагностику ошибок, код завершения или другую информацию).**

Командная строка — приглашение **оболочки**, обозначающее готовность системы принимать команду пользователя, — в наиболее явной форме демонстрирует идею диалога.

На каждую введенную команду пользователь получает ответ от системы: либо очередное приглашение, обозначающее, что команда выполнена и можно вводить следующую, либо сообщение об ошибке, представляющее собой высказывание системы о произошедших в ней событиях, адресованное пользователю.

Командный интерфейс



- Взаимодействие человека с компьютером осуществляется путем подачи компьютеру команд, которые он выполняет и выдает результат пользователю

Текстовый или графический полноэкранный интерфейс

Он имеет, как правило, в верхней части экрана **систему меню с подсказками**. Меню часто бывает выпадающим (ниспадающим – **pull-down**).

Для управления компьютером курсор экрана или курсор мыши после поиска в древе каталогов устанавливается на командные файлы программ (*.exe, *.com, *.bat) и для запуска программы нажимается клавиша или правая кнопка мыши.

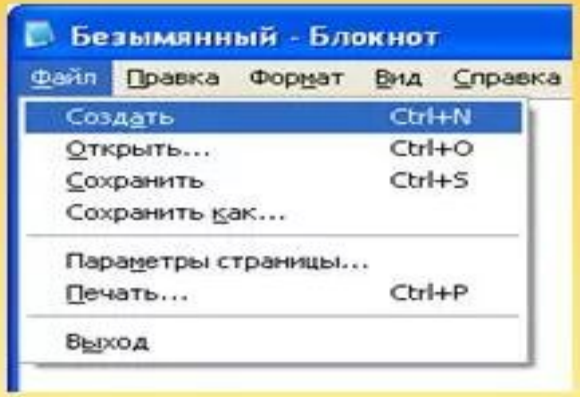
Различные файлы могут выделяться разным цветом или иметь разный рисунок.

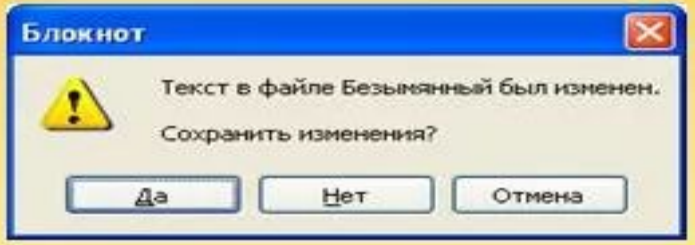
Каталоги (папки) отличаются от файлов размером или рисунком.

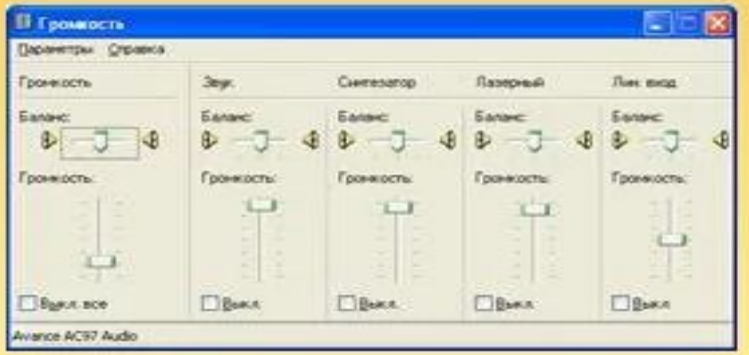
Данный интерфейс является основным для всех видов программных оболочек.

Графический интерфейс Windows

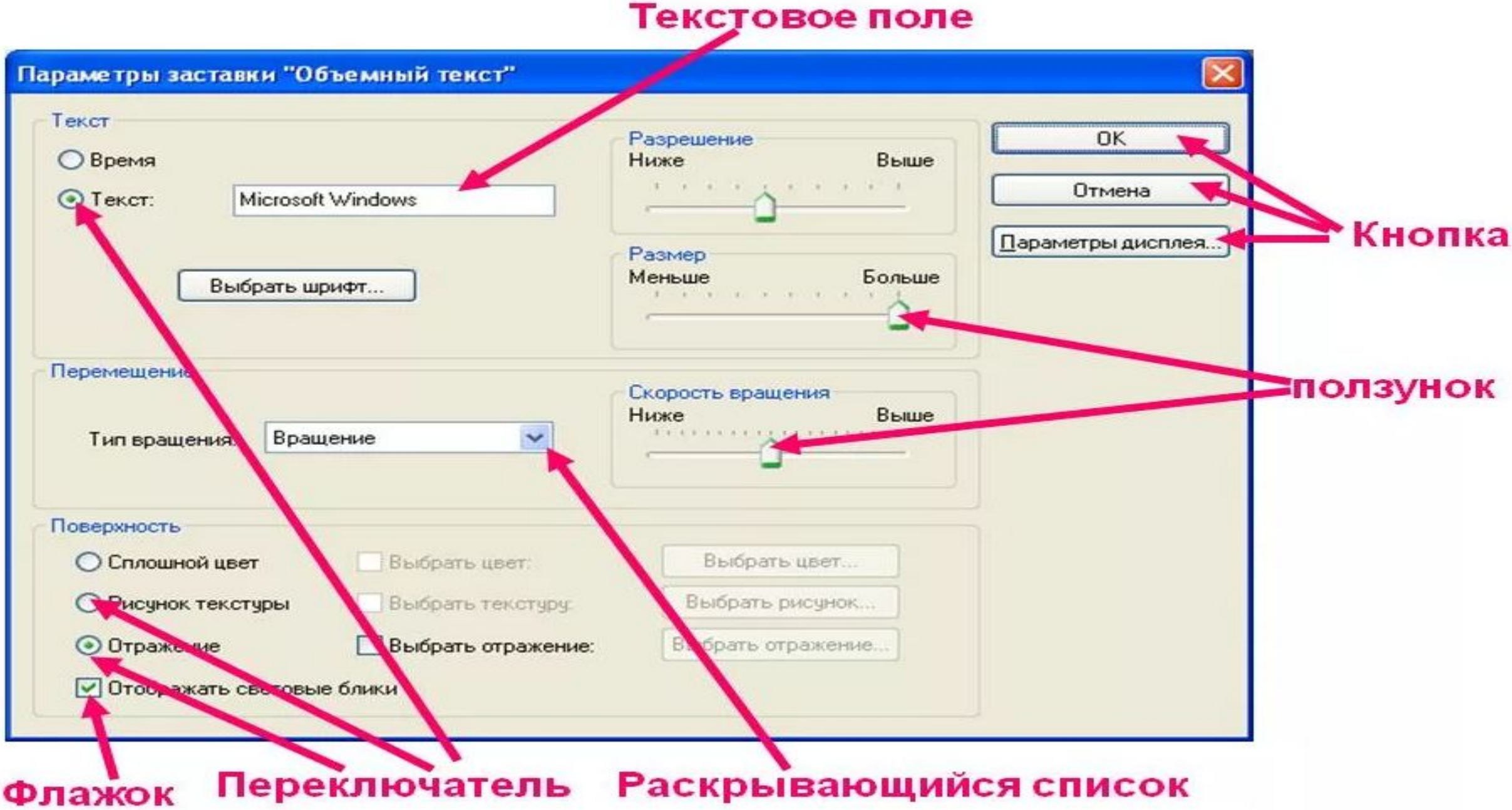
Графический интерфейс позволяет осуществлять взаимодействие человека с компьютером в форме диалога с использованием окон, меню и других элементов управления







Основные элементы графического интерфейса

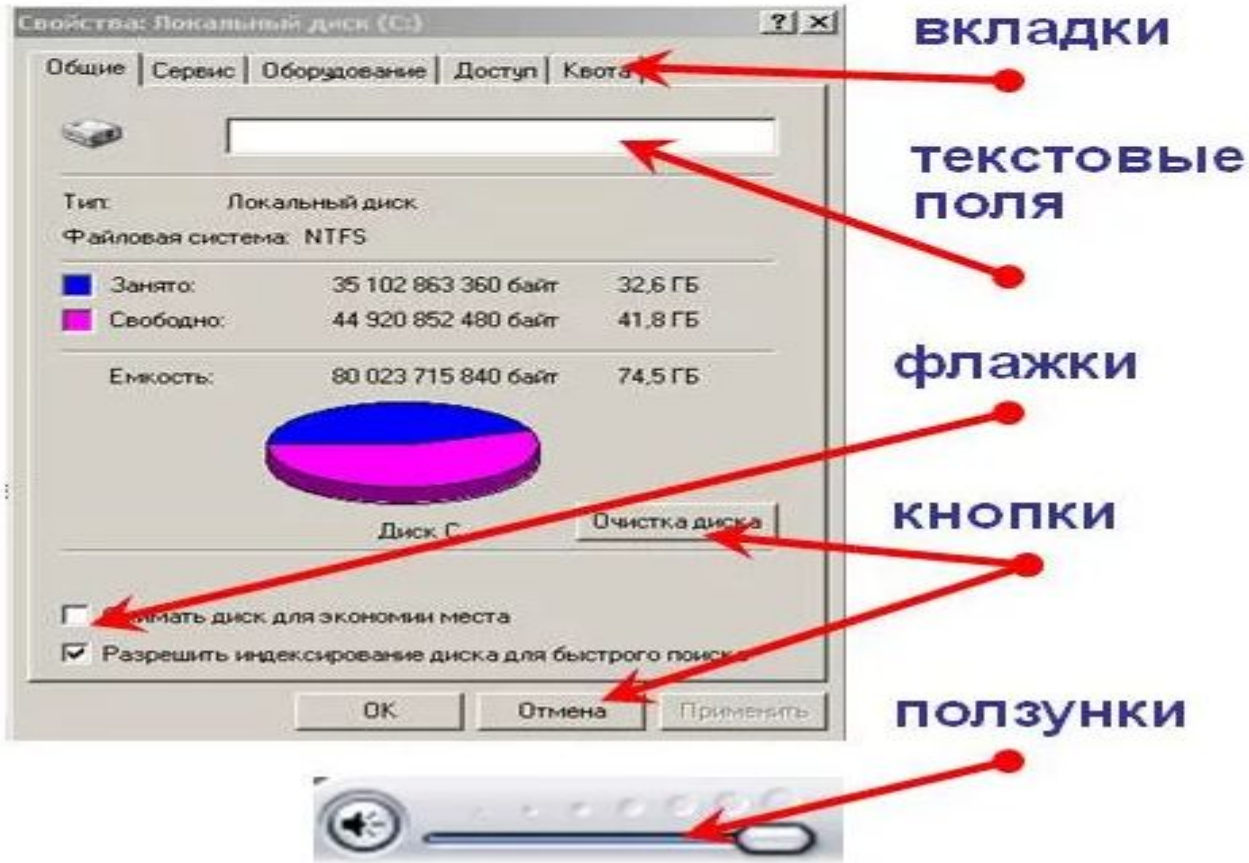


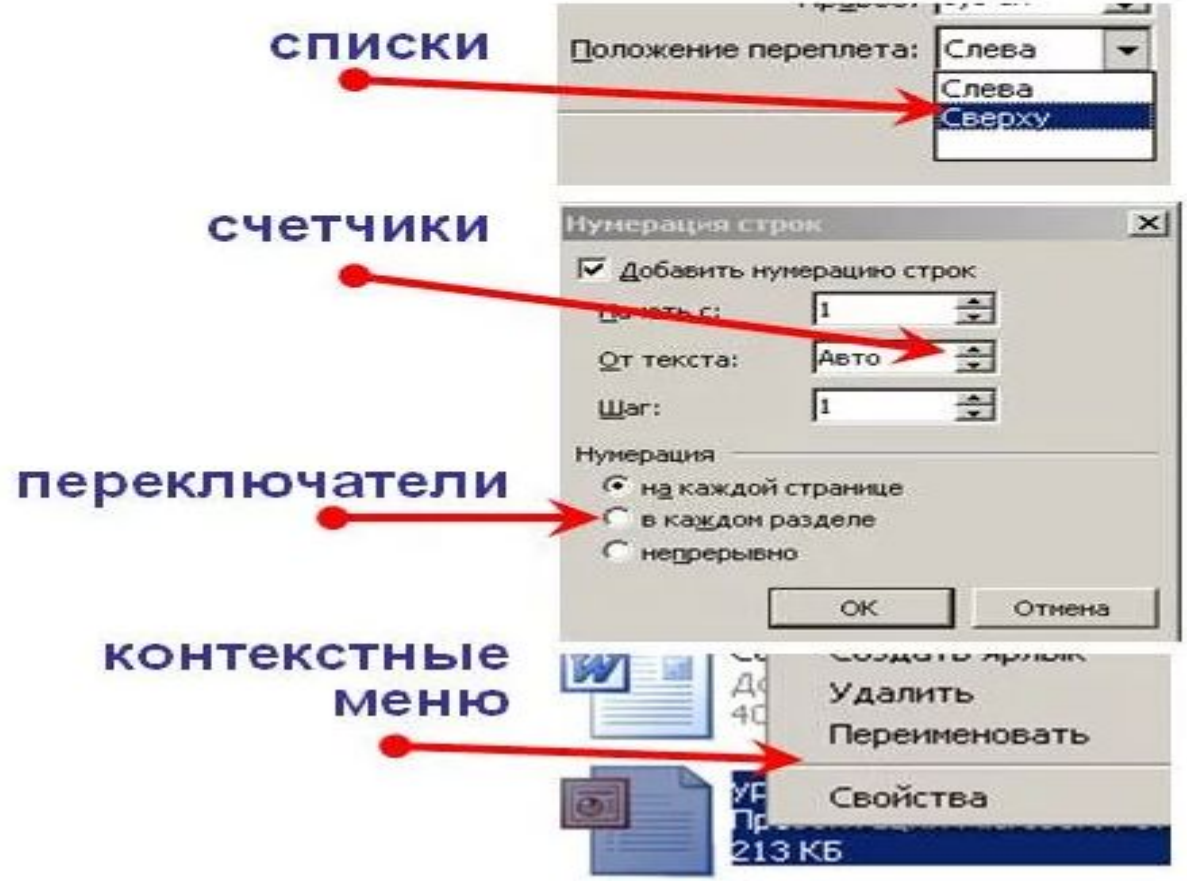
Графический интерфейс

Графический пользовательский интерфейс создают программные модули.

В операционных системах с графическим интерфейсом пользователь может вводить команды с помощью **диалоговых окон**.

Диалоговые окна могут включать в себя разнообразные **элементы управления:**





Графический многооконный пиктографический интерфейс

Представляет собой **рабочий стол (DeskTop)**, на котором располагаются пиктограммы (значки или иконки программ).

Все операции производятся, как правило, мышью.

Для управления компьютером курсор мыши подводят к пиктограмме и запуск программы осуществляют щелчком левой кнопки мыши по пиктограмме.

Графический интерфейс пользователя (GUI – Graphics User Interface).

Появление ОС и оболочек с развитыми диалоговыми графическими средствами (OS Macintosh, Windows 3.1, а особенно Windows 95/98/ME, а также NT/2000) и средств программирования, позволяющих создавать графические интерфейсы (FoxPro for Windows и пр.), а особенно – объектно – ориентированных систем программирования – привело к внедрению и широкому распространению элементов экранного интерфейса.

Графический интерфейс пользователя (GUI)

Главное окно Windows

Рабочий стол

папки

Ярлыки

Область уведомлений

Панель задач

Кнопка Пуск

значки

Интерфейс API

API — это специальный интерфейс программы или приложения (библиотеки классов и процедур), с помощью которого одна программа/приложение может взаимодействовать с другой.

С помощью API различные программы и приложения могут использовать функции и ресурсы друг друга. Это своего рода "язык программ", на котором они разговаривают и обмениваются данными и информацией.

Примеры использования API:

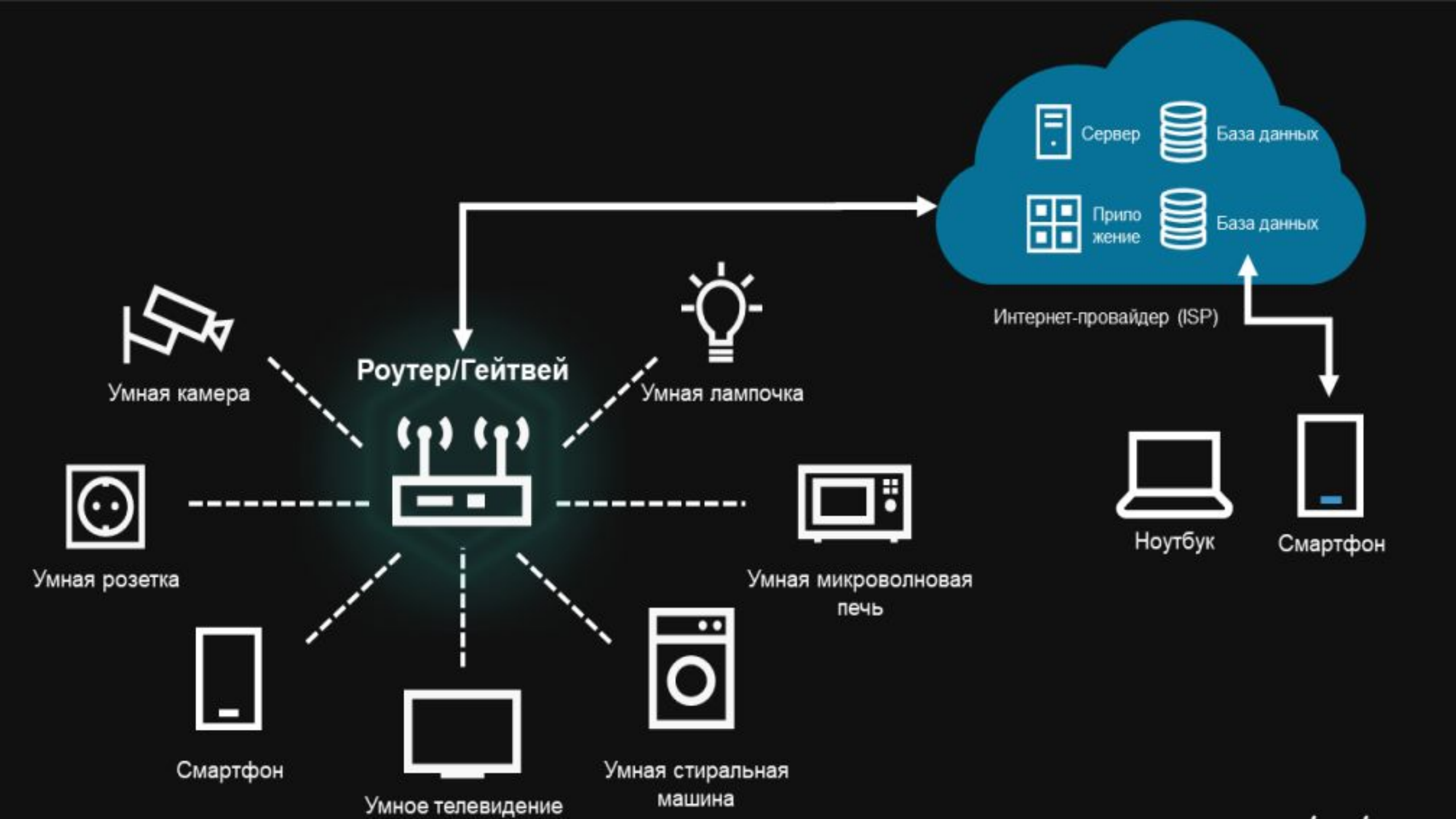
- Яндекс: API позволяет взаимодействовать с многими сервисами Яндекс, например, Погода, Директ.
- Вконтакте: с помощью API реализована функция отложенного постинга Вконтакте через сторонние приложения и сайты.

API определяет функциональность, которую предоставляет программа (модуль, библиотека), при этом API позволяет абстрагироваться от того, как именно эта функциональность реализована.

Программные компоненты взаимодействуют друг с другом посредством API. При этом обычно компоненты образуют иерархию — высокоуровневые компоненты используют API низкоуровневых компонентов, а те, в свою очередь, используют API ещё более низкоуровневых компонентов.

По такому принципу построены протоколы передачи данных по Internet. Стандартный протокол Internet (сетевая модель OSI) содержит 7 уровней (от физического уровня передачи пакетов бит до уровня протоколов приложений, подобных протоколам HTTP и IMAP).

Каждый уровень пользуется функциональностью предыдущего уровня передачи данных и, в свою очередь, предоставляет нужную функциональность следующему уровню.


Сервер
База данных
Прило
жение
База данных
Интернет-провайдер (ISP)
Роутер/Гейтвей
Умная камера
Умная лампочка
Умная розетка
Умная микроволновая
печь
Ноутбук
Смартфон
Смартфон
Умное телевидение
Умная стиральная
машина

Загрузчик ОС. Инициализация аппаратных средств. Процесс загрузки ОС.

1. **Нажатие кнопки включения питания.** При включении кнопки Power на элементы материнской платы поступают питающие напряжения; по сигналу Power Good запускается тактовый генератор; на процессор подается сигнал сброса, который устанавливает его в исходное состояние. Начинают работать программы системного BIOS.
2. **Проверка BIOS.** Контрольная сумма системных программ, находящихся в ПЗУ, находится в одной из ячеек. После запуска контрольная сумма пересчитывается и сравнивается с эталонным значением.
3. **Идентификация процессора.** Материнская плата предусматривает возможность установки различных моделей процессора. БИОС подает запрос на идентификацию процессора и по полученному ответу определяет тип процессора, частоту, напряжения и проч.
4. **Настройка базовых элементов.** Инициализируются и тестируются базовые компоненты системной платы: блок прямого доступа к памяти, таймер, блок аппаратных прерываний.
5. **Тестирование ОЗУ.** Определяется тип модулей памяти, их объем, организация; тестируются первые 64 Кб оперативной памяти.

6. Организация рабочих структур ОЗУ. Выделяется область под БИОС, настраиваются прерывания.

7. Проверка CMOS-памяти и батарейки. При неисправной батарейке CMOS все данные настройки БИОС, находящиеся в памяти теряются. Загрузка последней конфигурации становится невозможной, о чем сообщается на экране монитора. Есть возможность осуществить загрузку стандартных заводских значений БИОС.

8. Инициализация устройств материнской платы. Производится поиск и настройка загрузочных устройств (жесткий диск, привод CD, FDD), средств управления процессом загрузки (клавиатура, мышь), устройств ввода-вывода (COM, LPT). Устройствам выделяются соответствующие линии прерывания.
9. PnP. Идентифицируются устройства, подключенные через системные разъемы. Устройствам выделяются ресурсы и прерывания.
10. Включение видеосистемы. Запускается Video BIOS, который настраивает видеоконтроллер на режим VGA или EGA, которые поддерживают все видеоконтроллеры. После этого видеоконтроллер готов к работе.
11. Выдача сообщения на экран монитора. На экране монитора появляется первое сообщение: фирма-производитель BIOS, тип и частота процессора, тип и объем ОЗУ.
12. Тестирование ОЗУ. Производится выборочная проверка незадействованной оперативной памяти.
13. Инициализация контроллера дисководов.
14. Инициализация контроллера жестких дисков.
15. Инициализация клавиатуры. Включается контроллер клавиатуры, производится тест матрицы контактов, устанавливаются временные параметры опроса клавиш и режим NumLock. Клавиатура готова к работе. На экран выводится сообщение о возможности использования программы BIOS Setup (обычно для этого используется клавиша Del).

16. Поиск устройств с собственным BIOS. Если таковые устройства найдены, то управление передается BIOS-программам этих устройств и происходит их инициализация.
17. Передача управления загрузчику ОС. По программному прерыванию Int 19h на дисковых накопителях ищется загрузчик ОС (Boot Record). Он должен находиться на одном из устройств (HDD, CD, FDD, SCSI). Местоположение загрузчика везде одинаково. После того, как загрузчик ОС найден, управление передается ему.

Переносимость ОС. Поддержка операций ввода-вывода. Машино-зависимые модули ОС. Драйверы.

Операционная система называется **переносимой ОС** (portable), или *мобильной*, если ее код может быть сравнительно легко перенесен с процессора одного типа на процессор другого типа и с аппаратной платформы одного типа на аппаратную платформу другого типа.

Для того чтобы обеспечить свойство мобильности ОС, разработчик должен следовать установленным правилам.

1) **Подавляющая часть кода должна быть написана на языке, трансляторы которого имеются на всех машинах, куда предполагается переносить систему.**
   Такими языками, например, являются стандартизированные языки высокого уровня. Большинство переносимых ОС написано на языке С, который имеет много особенностей, полезных для разработки кодов ОС, и компиляторы которого широко доступны. Языки низкого уровня не подходят для решения подобных задач. Так, программа, написанная на ассемблере, является переносимой только в том случае, когда перенос ОС предполагается на компьютер, обладающий той же системой команд.
   В остальных случаях ассемблер используется только для тех непереносимых частей ОС, которые должны непосредственно взаимодействовать с аппаратурой (например, для обработчика прерываний), или частей, требующих максимальной производительности (например, для целочисленной арифметики повышенной точности).

2) Объем машинно-зависимых частей кода, непосредственно взаимодействующих с аппаратными средствами, должен быть по возможности минимизирован.

Всегда следует избегать прямого манипулирования регистрами и другими аппаратными средствами процессора. Для уменьшения аппаратной зависимости разработчики ОС должны также исключить возможность использования по умолчанию стандартных конфигураций аппаратуры или их характеристик.

Аппаратно-зависимые параметры можно “спрятать” в программно-задаваемые пользователем данные абстрактного типа. Для осуществления всех необходимых действий по управлению аппаратурой, представленной этими параметрами, должен быть написан набор аппаратно-зависимых функций.

Каждый раз, когда какому-либо модулю ОС требуется выполнить некоторое действие, связанное с аппаратурой, он манипулирует абстрактными данными, используя соответствующую функцию из имеющегося набора. Когда ОС переносится, то соответственно изменяются только эти данные и функции, которые ими манипулируют.

3) Аппаратно-зависимый код должен быть надежно изолирован в нескольких модулях, а не быть распределен по системе.

Изоляции подлежат все части ОС, которые отражают специфику как процессора в отдельности, так и используемой аппаратной платформы в целом.

Низкоуровневые компоненты ОС, имеющие доступ к процессорно-зависимым структурам данных и регистрам, должны быть в обязательном порядке оформлены в виде компактных логически обособленных модулей, которые могут быть заменены аналогичными по выполняемым функциям модулями для других процессоров.

Для снятия платформенной зависимости, возникающей из-за различий между компьютерами разных производителей, построенными на одном и том же процессоре, должен быть введен хорошо локализованный программный слой машинно-зависимых функций с четко оговоренным межслойным интерфейсом.

Виды файловых систем. Физическая и логическая организация ФС

Логическая организация файловой системы

Одной из основных задач операционной системы является предоставление удобств пользователю при работе с данными, хранящимися на дисках.

Для этого ОС подменяет физическую структуру хранящихся данных некоторой удобной для пользователя логической моделью.

Логическая модель файловой системы материализуется в виде дерева каталогов, выводимого на экран такими утилитами, как Norton Commander или Windows Explorer, в символьных составных именах файлов, в командах работы с файлами.

Базовым элементом этой модели является файл, который так же, как и файловая система в целом, может характеризоваться как логической, так и физической структурой.

Типы файлов

Файловые системы поддерживают несколько функционально различных типов файлов, в число которых, как правило, входят:

-обычные файлы,
-файлы-каталоги,
-специальные файлы,
-именованные конвейеры,
-отображаемые в память файлы и другие.

Обычные файлы, или просто файлы, содержат информацию произвольного характера, которую заносит в них пользователь или которая образуется в результате работы системных и пользовательских программ.

Содержание обычного файла определяется приложением, которое с ним работает.

Например, текстовый редактор создает текстовые файлы, состоящие из строк символов, представленных в каком-либо коде. Это могут быть документы, исходные тексты программ и т. п.
Текстовые файлы можно прочитать на экране и распечатать на принтере.
Двоичные файлы не используют коды символов, они часто имеют сложную внутреннюю структуру, например исполняемый код программы или архивный файл.

Все операционные системы должны уметь распознавать хотя бы один тип файлов — их собственные исполняемые файлы.

Каталоги — это особый тип файлов, которые содержат системную справочную информацию о наборе файлов, сгруппированных пользователями по какому-либо неформальному признаку (например, в одну группу объединяются файлы, содержащие документы одного договора, или файлы, составляющие один программный пакет).

Во многих операционных системах в каталог могут входить файлы любых типов, в том числе другие каталоги, за счет чего образуется древовидная структура, удобная для поиска.

Каталоги устанавливают соответствие между именами файлов и их характеристиками, используемыми файловой системой для управления файлами.
В число таких характеристик входит, в частности, информация (или указатель на другую структуру, содержащую эти данные) о типе файла и расположении его на диске, правах доступа к файлу и датах его создания и модификации.

Во всех остальных отношениях каталоги рассматриваются файловой системой как обычные файлы.

Специальные файлы — это фиктивные файлы, ассоциированные с устройствами ввода-вывода, которые используются для унификации механизма доступа к файлам и внешним устройствам.

Специальные файлы позволяют пользователю выполнять операции ввода-вывода посредством обычных команд записи в файл или чтения из файла.

Эти команды обрабатываются сначала программами файловой системы, а затем на некотором этапе выполнения запроса преобразуются операционной системой в команды управления соответствующим устройством.

Имена файлов

Все типы файлов имеют символьные имена. В иерархически организованных файловых системах обычно используются три типа имен файлов:

-простые,

-полные (составные),

-относительные.

Простое, или ***короткое, символьное имя*** идентифицирует файл в пределах одного каталога. Простые имена присваивают файлам пользователи и программисты, при этом они должны учитывать ограничения ОС как на номенклатуру символов, так и на длину имени. До сравнительно недавнего времени эти границы были весьма узкими.

Полное имя представляет собой цепочку простых символьных имен всех каталогов, через которые проходит путь от корня до данного файла. Таким образом, полное Имя является *составным,* в котором простые имена отделены друг от друга принятым в ОС разделителем. Часто в качестве разделителя используется прямой или обратный слеш, при этом принято не указывать имя корневого каталога.

Относительное имя файла определяется через понятие «текущий каталог». Для каждого пользователя в каждый момент времени один из каталогов файловой системы является текущим, причем этот каталог выбирается самим пользователем по команде ОС. Файловая система фиксирует имя текущего каталога, чтобы затем использовать его как дополнение к относительным именам для образования полного имени файла. При использовании относительных имен пользователь идентифицирует файл цепочкой имен каталогов, через которые проходит маршрут от текущего каталога до данного файла.