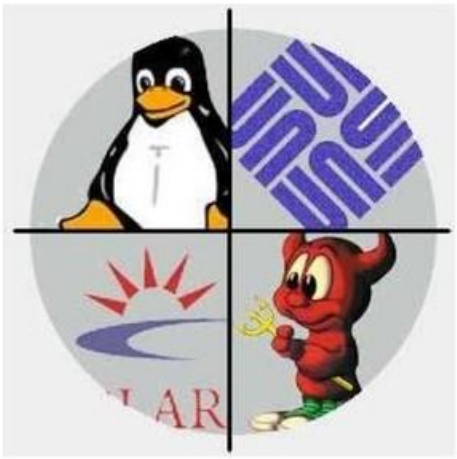
Частное профессиональное образовательное учреждение
«СЕВЕРО-КАВКАЗСКИЙ КОЛЛЕДЖ ИННОВАЦИОННЫХ ТЕХНОЛОГИЙ»
(ЧПОУ «СККИТ»)

ОПЕРАЦИОННЫЕ СИСТЕМЫ И СРЕДЫ

КУРС ЛЕКЦИЙ

09.02.06 Сетевое и системное администрирование



Пятигорск

Тема 1. История, назначение и функции операционных систем

ОПЕРАЦИОННЫЕ СИСТЕМЫ И СРЕДЫ.

Под *операционной системой* обычно понимают комплекс управляющих и обрабатывающих программ, который, с одной стороны, выступает как интерфейс между аппаратурой компьютера и пользователем, а другое предназначение для более эффективного пользования ресурсов вычислительной системы и организации надёжных вычислений.

Любой из компонентов прикладного программного обеспечения обязательно работает под управлением операционных систем. На схеме отображена обобщённая процедура программного обеспечения.

Видно, что не один из компонентов программного обеспечения, за исключением самой операционной системы, не имеет непосредственного доступа к аппаратуре компьютера. Даже пользователь взаимодействует со своими программами через интерфейс. Любые их команды, прежде чем попасть в прикладные программы, проходят через операционные системы, основными функциями, которые выполняются операционной системой, являются:

1) приём от пользователя заданий или команд, формулированных на соответствующем языке и их обработка;
2) приём и исполнение программы запроса на запуск/приостановку других программ;
3) загрузка в оперативную память, подлежащие исполнению, программы;
4) инициализация программ (передача ей управления), в результате чего процессор использует программу;
5) идентификация программ;
6) обеспечение работы системы управления файлами базы данных, что позволяет резко увеличить эффективность программного обеспечения;
7) обеспечение режима мультипрограммирования, т.е. выполнение двух или более программ на одном процессоре, создающая видимость их одновременного исполнения;
8) обеспечение функции по организации и управления всеми операциями ввода и вывода;
9) удовлетворение жёстким ограничениям на время в режиме реального времени;
10) распределение памяти:
    а) организация виртуальной памяти;
    б) в большинстве современных систем.
11) планирование и диспетчеризация в соответствии с заданием;
12) организация м-ма обмена сообщениями и данными между выполняющимися программами;
13) защита одной программы от влияния других программ, обеспечение сохранение данных;
14) предоставление услуг на случай частичного сбоя системы;
15) обеспечение работы системы программ, с помощью которых пользователи готовят свои программы.

Как правило, все современные операционные системы имеют систему управления памятью. Назначающаяся СУП-организация более удобного доступа к данным организациям как файл. Ряд операционных систем позволяют работать с несколькими файловыми системами одновременно. В этом случае говорят о вмонтированной файловой системе, т.е. дополнительную память можно установить.

Есть простейшие операционные системы, которые могут работать и без файловых систем или только с одной стороны из файловых систем. Любая система управления файлами разработана для работы конкретной операционной системы и конкретной файловой системы.

Например, известная файловая система *FAT.*

File
Allocation
Table

Имеет множество реализации как система управления файлами, например, *FAT 16* под систему *MS-DOS* или *Super FAT* для *OS/2* или *FAT* для *Windows.*

Для работы с файлами, организованные в соответствии с некоторыми файловыми системами для каждой операционной системы должна быть организована соответствующая система управления файлами. Она будет работать только в той операционной системе, для которой она разработана. Для удобства пользования с операционной системой может использоваться дополнительные интерфейсные оболочки. Их основное назначение, либо расширить возможность операционной системы, либо изменить встроенные в систему возможности. Классическим примером интерфейсных оболочек можно назвать:

1) *X Windows* в системах семейства *Unix;*
2) *KDE – K Desktop Environment*;
3) *PM Shell;*
4) *Object Desktop.*

Существуют различные варианты интерфейса для семейства операционной системы *Windows,* которые заменяют *Explover,* в файле *system.ini.*

В операционной системе заменяемой является только интерфейсная оболочка. Операционная среда определяется программными интерфейсами.

Application
Program
Interface

API – интерфейс прикладного программирования, включает в себя управление процессами, памятью и вводом/выводом.

Ряд операционных систем может выполнять ряд программ, созданных для выполнения в других операционных системах. Соответствующая среда организуется в рамках данной машины. Аналогично, в *Linux* можно создать условия для выполнения программ, написанных для *Windows 98.*

Под *утилитами* понимают специальные системы программирования, с помощью которых можно обслужить операционную систему, выполнять обработку данных, осуществляющих оптимизацию данных на носителе и производить работы по обслуживанию операционной системы.

К утилитам относится программа разбиения накопителя на магнитных дисках на разделы и программу форматирования, программу переноса основных системных файлов самой операционной системой. Утилиты могут работать только в соответствующей операционной системе.

ПОНЯТИЕ ОПЕРАЦИОННОЙ СРЕДЫ.

Операционная система выполняет функции управления вычислительными процессами в вычислительной системе, распределяет ресурсы вычислительной системы между различными вычислительными процессами и образует программную среду, в которой выполняются прикладные программы пользователя. Такая среда называется *операционной.*

Любая программа имеет дело с некоторыми исходными данными, которые она обрабатывает и порождает некоторые выходные данные, т.е. результаты вычислений. В абсолютном большинстве случаев исходные данные попадают в оперативную память внешних (периферийных) устройств.

Результаты вычислений также выводятся на внешние устройства. Программирование операций ввода/вывода является наиболее сложной задачей. Именно поэтому развитие операционной системы пошло по пути выделения наиболее часто встречающихся операций и создании для них соответствующих модулей, которые можно в дальнейшем использовать во вновь создаваемых

программах. //В конечном итоге возникла ситуация, когда при создании двоичных машинных программ …//

Программисты могут вообще не знать многих деталей управления ресурсами вычислительной системы, а должны обращаться к некоторой программной подсистеме с соответствующими выводами и получить необходимые функции сервиса. Эта программная подсистема и есть операционная система, а набор её функций сервиса и привело обращение к ней и образует базовое понятие, которое называется *операционной средой,* т.е. термин операционная среда означает необходимые интерфейсные программы пользователя для обращения к операционной системе с целью получить определённый сервис. Параллельное существование терминов “операционная система” и “операционная среда” вызвано тем, что операционная система может поддержать несколько операционных сред. Например, операционная система *OS/2 Warp* может выполнять следующие программы:

1) так называемые нативные (*Native*) программы, созданные с учётом 32-разрядного операционного интерфейса;
2) 16-битные программы, созданные для *OS/2* первого поколения;
3) 16-битные программы, разработанные для *MS-DOS PS* и *DOS.*
4) 16-битовые программы для операционной среды *Windows.*
5) Сама операционная оболочка *Windows 3.X* и уже в ней, созданные для неё, программы.

ПРЕДНАЗНАЧЕНИЕ И ФУНКЦИИ ОПЕРАЦИОННОЙСИСТЕМЫ.

Операционная система – это программа, контролирующая работу пользовательской программы и систем приложений и исполняемая роль интерфейса между приложениями и аппаратным обеспечением компьютера. Её предназначения можно разделить на три основные составляющие:

1) удобство: операционная система делает исполнение компьютера простым и удобным;
2) эффективность: операционная система позволяет эффективно использовать ресурсы компьютерной системы;
3) возможность развития: операционная система должна допускать разработку тестирования новых приложений и системных функций без нарушения нормального функционирования вычислительной системы.

ОПЕРАЦИОННАЯ СИСТЕМА КАК ИНТЕРФЕЙС МЕЖДУ ПОЛЬЗОВАТЕЛЕМ И КОМПЬЮТЕРОМ.

Пользователь, как правило, не интересуется деталями устройства аппаратного обеспечения компьютера, он видится ему как набор приложений. Приложение можно написать на каком-то из языков программирования. Чтобы упростить эту задачу имеется набор системных программ, некоторые из которых называют утилитами, с их помощью реализуется часто исполнение Функции, которые помогают при создании пользовательских программ в работе с файлами и управление устройствами ввода/вывода. Программист использует эти средства при разработке этих программ, а приложения во время выполнения обращаются к утилитам для выполнения определённых функций. Наиболее важной из системных программ являются операционные системы, которые скрывает от программиста детали аппаратного обеспечения и предоставляет удобный интерфейс для исполнения системы операционной среды. Может включать несколько интерфейсов:

1) пользовательский;
2) программный
   Например, система *Linux* им. для пользователя как интерфейсные команды (различные оболочки): *C-Shell, K-Shell, B-Shell, bash-shell.*

ИНТЕРФЕЙС ТИПА Midnight *Commander.*

Так и графические интерфейсы (*X Windows*). В нём могут быть различные менеджеры окон (*KDE Grome*).

Что касается программных интерфейсов, то операционная система *Windows* программы может обращаться как операционная система за соответствующими сервисами и функциями, так и к графической подсистеме. С точки зрения архитектуры процессора, вторая программа, созданная для работы в *Linux* использует те же команды и форматы данных, что и программа, созданная для работы в среде *Windows.* Однако в первом случае имеет место обращение к операционной среде, во втором – к другой. Таким образом, операционная среда – это то системное программное обеспечение, в котором могут выполняться программы, созданные по правилам работы этой среды.

Типичные операционные системы предоставляют следующие сервисы:

1) *разработка программ*. Операционная система предоставляет программисту разнообразные инструменты и сервисы, например, редакторы и отладчики. Эти сервисы, реализованные в виде программных утилит, которые поддерживают операционные системы, хотя и не входят в его ядро, такие программы называют *инструментами разработки приложений*;
2) *исполнение программ.* Для запуска программы требуется выполнить ряд действий. Следует загрузить в основную память команды и данные, инициализировать устройства. Операционная система выполняет рутинную работу;
3) *доступ к устройствам ввода/вывода*. Для управления работой каждым устройством ввода/вывода нужен свой набор команд или контролируемый сигнал. Операционная система предоставляет пользователю единообразный интерфейс, который вскрывает все эти детали и обеспечивает программисту доступ к устройствам ввода/вывода с помощью простых команд чтения и записи;
4) *контролируем доступ к файлам*. При работе с файлами, управление его стороны операционной системы предназначено не только понимание природы устройств ввода/вывода и знание структур данных записанные в файлах. Многопользовательские операционные системы, кроме того, обеспечивают работу механизмов защиты при обращении к файлам;
5) *системы доступа*. Операционная система управляет доступом к общедоступной вычислительной системе в целом, а также к отдельным системным ресурсам. Она должна обеспечить защиту ресурсов и данных от несанкционированного использования, также разрешать конфликтные ситуации;
6) *обнаружение ошибок и их обработка*. При работе компьютерной системы происходят различные сбои, к их числу относятся внутренние и внешние ошибки, возникшие в аппаратном обеспечении, например, ошибки памяти, отказ или сбой устройств, возможны и программные ошибки: арифметическое переполнение, попытка обратиться к ячейке памяти, доступ к которым запущен и невозможность выполнения запроса приложения. В каждом из этих случаев операционная система должна выполнить действие, минимизирующее влияние ошибки на работу приложения. Реакция операционной системы на ошибку может быть различной: от простого сообщения об ошибке, до аварийной остановки программы;
7) *учёт использования ресурсов*. Хорошая операционная система должна иметь средства учёта использования различных ресурсов и отображение параметров производителя. Эта информация крайне важна для дальнейшего улучшения и настройки система, для повышения производительности.

ОПЕРАЦИОННАЯ СИСТЕМА КАК ДИСПЕТЧЕР РЕСУРСОВ.

Компьютер представляет собой набор ресурсов, поддерживающих выполнение задач, накопление, хранение, перемещение и обработки данных, также контролирует работу этих и других функций. Именно операционная система управляет ресурсами компьютера и контролирует его основные функции. Однако это управление имеет следующие особенности:

1) функции операционной системе работают так же, как и всё остальное программное обеспечение, т.е. они реализуются в виде отдельных программ или набора программ, исполняющихся процессов;

2) операционная система должна передавать управление другими процессами и ожидать, когда процессор снова позволит ей выполнить свои обязанности.

Операционная система – это, по сущности, набор компьютерных программ, как и любая другая программа, она отдаёт процессору команды. Ключевым отличием является назначение этой программы.

Операционная система //способна//: как использовать другие системные ресурсы, и как распределять время при использовании других программ, но для этого процессор должен приостановить работу с ней и перейти к выполнению других программ.

Таким образом, операционная система уступает управление процессору, чтобы он смог выполнить некоторую полезную работу, а затем возобновляет контроль ровно на столько, чтобы подготовить процессор к следующей части работы.

Часть операционной системы находится в оперативной памяти (основная, базовая). В эту часть входят ядро (*Kernel*), содержащее основную часть наиболее часто используемых функций, там же находятся и некоторые другие компоненты операционной системы, использующиеся в данный момент времени.

Остальная часть содержит другие программы и данные пользователя. Размещение этих данных в оперативной памяти управляется совместно операционной системой и аппаратной частью процессора, предназначенной для управления памятью. Операционная система принимает решение, когда исполняющая программа может испортить нужные ей устройства ввода/вывода и управляет доступом к файлам.

Процессор также является ресурсом, которому операционная система должна определить, сколько времени он должен уделить исполнению той или иной пользовательской программы. Многопроцессорные системы: решение должно быть принято по каждому процессу.

ВОЗМОЖНОСТИ РАЗВИТИЯ ОПЕРАЦИОННОЙ СИСТЕМЫ.

Большинство операционных систем постоянно развиваются. Происходит это в силу следующих причин:

а*) обновление и возникновение новых видов аппаратного обеспечения*;

б) *новые сервисы*. В операционной системе могут быть добавлены новые инструменты для контроля и оценки производительности, чтобы поддержать высокую эффективность работы, с имеющимся инструментарием пользователя;

в) *исправление*. В каждой операционной системе есть ошибки. Время от времени они обнаруживаются и исправляются. Необходимость регулярного изменения операционных систем, накладываются определённые ограничения на устройство. Очевидно, что эти системы должны иметь модульную конструкцию, чётко определённую взаимодействием модулей. Для больших программ важную роль играет хорошее и полное документирование.

Принятые условные обозначения:

- I/O - ввод/вывод;
- АО - аппаратное обеспечение;
- БД - база данных;
- ОЗУ - оперативное запоминающее устройство;
- ОС - операционная система;
- ПЗУ - постоянное запоминающее устройство;
- ПК - персональный компьютер;
- ПО - программное обеспечение;
- РВ - реальное время;
- СУ - система управления;
- СУБД - система управления базами данных;
- УВВ - устройство ввода/вывода;

- ФС - файловая система;
- ЦП - процессор (центральный процессор).

Классификация ОС

Развитие компьютеров привело к развитию ОС. Сейчас насчитывается более 100 ОС.

По назначению ОС принято делить на семь уровней.

1. Мэйнфреймы (mainframe)

У них отличаются от ПК возможности I/O. Обычно мэйнфреймы содержат тысячи дисков и терабайты ОЗУ. Они используются в виде мощных web-серверов, серверов для крупномасштабных коммерческих сайтов и серверов для транзакций в бизнесе. ОС для мэйнфреймов ориентированы на обработку множества одновременных заданий, большинству из которых требуется огромное количество операций I/O. Обычно они предполагают три вида обслуживания:

1) пакетную обработку. Система выполняет стандартные задания без присутствия пользователей. В пакетном режиме обрабатываются иски страховых компаний и составляются отчеты о продаже в магазине;
2) обработку транзакций (групповые операции: обработка и запись данных). Система обработки транзакций управляет очень большим количеством маленьких запросов (например, контролирует процесс работы в банке, бронирует авиабилеты). Каждый отдельный запрос невелик, но система должна отвечать на тысячи запросов в секунду;
3) разделение времени. Системы, работающие в режиме разделения времени, позволяют множеству удаленных пользователей выполнять свои задания на одной машине, например, работать с большой БД. Все эти функции тесно связаны между собой и часто ОС мэйнфрейма выполняет их все. Примером ОС для мэйнфрейма является OS/390 (от IBM).

2. Серверные (сетевые) ОС

Работают на серверах, которые представляют собой или очень большие ПК, или рабочие станции, или даже мэйнфреймы. Они одновременно обслуживают множество пользователей и позволяют им делить программные и аппаратные ресурсы. Серверы представляют возможность работать с печатающими устройствами, файлами и Internet. Internet-провайдеры обычно запускают в работу несколько серверов, чтобы поддерживать одновременный доступ к сети множества клиентов. На серверах хранятся страницы web-сайтов и обрабатываются входные запросы. Типичные серверные ОС: Windows 2000 и Unix. В этих целях в настоящее время стала использоваться и ОС Linux.

3. Многопроцессорные ОС (кластеры)

Наиболее часто применяемый способ увеличения мощности компьютера заключается в соединении ЦП в одну систему. В зависимости от вида соединения ЦП и разделения работы такие системы называются параллельными компьютерами, мультикомпьютерами или многопроцессорными системами. Для них требуются специальные ОС, но, как правило, такие ОС представляют собой варианты серверных ОС со специальными возможностями связи.

4. ОС для ПК

Работа этих ОС заключается в представлении удобного интерфейса для одного пользователя. Такие ОС широко используются для работы с текстом, электронными таблицами и доступа к Internet. Яркие примеры: Windows 98, 2000, MacOS, Linux.

5. ОС РВ

Главным параметром ОС РВ является время. Например, в СУ производством компьютеры, работающие в режиме РВ, собирают данные о промышленном процессе и используют их для управления машинами. Такие процессы должны удовлетворять жестким временным требованиям. Так, если автомобиль передвигается по конвейеру, то каждое действие должно быть осуществлено в строго

определенный момент времени. Если сварочный робот сварит шов слишком рано/поздно, то он нанесет непоправимый вред. Если некоторое действие должно произойти в какой-то момент времени или внутри заданного диапазона времени, то говорят о жесткой системе РВ. Существует гибкая система РВ, в которой допустимы случающиеся время от времени пропуски сроков выполнения операций. В эту категорию попадает цифровое аудио и multimedia-системы. Примеры ОС: VxWorks, QNX.

6. Встроенные ОС

Карманный компьютер, или PDA (Personal Digital Assistant), - маленький компьютер, помещающийся в кармане брюк и выполняющий некоторые функции (записная книжка, блокнот). Примеры ОС: PalmOS, Windows CE (Consumer Electronics - бытовая техника).

7. ОС для Smart-карт (smart-cards - разумные карты)

Самые маленькие ОС работают на Smart-картах, представляющих собой устройство с ЦП. На такие ОС накладываются крайне жесткие ограничения по мощности ЦП и памяти. Некоторые из них могут управлять только одной операцией, но другие ОС на тех же самых Smart-картах выполняют сложные функции. Некоторые ОС являются Java-ориентированными, т.е. ПЗУ содержит интерпретатор виртуальной машины Java (ROM - Read Only Memory). Апплеты Java загружаются на карту и выполняются интерпретатором JVM (Java Virtual Machine). Некоторые из этих карт могут одновременно управлять несколькими Java-апплетами, что приведет к многозадачности и необходимости планирования. Также возникает необходимость в защите. Эти задачи обычно выполняет крайне примитивная ОС.

Тема 2. Архитектура операционной системы

АО компьютера

ОС тесно связана с оборудованием компьютера, поскольку оно влияет на набор команд ОС и управление ресурсами. Поэтому необходим определенный объем знаний о компьютере, т.е. нужно знать, в каком виде оборудование предстает. Концептуально простой ПК можно представить в виде абстрактной модели.

ЦП, память, устройства I/O соединены системной шиной, по которой они обмениваются информацией. "Мозгом" компьютера является ЦП (CPU - Central Processing Unit). Он выбирает из памяти команды и выполняет их. Обычный цикл работы ЦП выглядит так: он читает первую команду из памяти, декодирует ее для определения ее типа и операндов, выполняет команду, затем считывает, декодирует последующие команды. Т.о. образом осуществляется выполнение программ.

Для каждого ЦП существует набор команд, который он в состоянии выполнить. Поскольку доступ к памяти для получения команд или набор данных занимает намного больше времени, чем выполнение этих команд, то все ЦП содержат внутренние регистры для хранения переменных и промежуточных результатов. Поэтому набор инструкций обычно содержит команды для загрузки слова из памяти в регистр и сохранение слова из регистра в память. Кроме основных регистров, используемых для хранения переменных, большинство ЦП имеет несколько специальных регистров, используемых для хранения переменных, большинство ЦП имеет несколько специальных регистров, видимых для программистов.

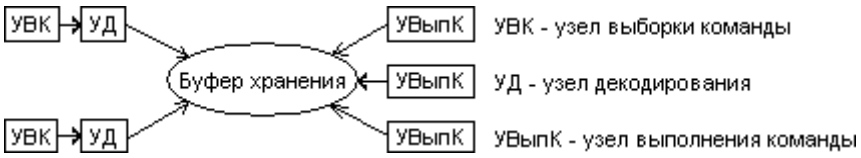
Один из них называется счетчиком команд (PC - Program Counter). В нем содержится адрес следующей стоящей в очереди на выполнение команды. После того как команда выбрана из памяти, регистр команд корректируется и указатель переходит к следующей команде.

Еще один регистр - указатель стека (SP - Stack Pointer). Он содержит адрес вершины стека в памяти. Стек содержит по одному фрейму (области данных) для каждой процедуры, которая уже начала выполняться, но еще не закончена. В стеке процедуры хранятся ее выходные параметры, а также локальные и промежуточные переменные, не хранящиеся в регистрах.

Следующий регистр называется "слово состояния ЦП" (PSW - Processing Status Word). Этот регистр содержит биты кода состояний, которые задаются командами сравнения, приоритетом ЦП, режимом (пользовательский или режим ядра) и другую служебную информацию.

Пользовательские программы могут читать весь регистр PSW, но писать могут только в некоторых из его полей. Регистр PSW играет важную роль в системных вызовах и операциях I/O. При временном мультиплексировании ЦП ОС останавливает работающую программу для запуска другой. Каждый раз при таком прерывании ОС должна сохранять все регистры ЦП, чтобы позже, когда прерванная программа продолжит свою работу, их можно было восстановить. Для повышения быстродействия ЦП их разработчики отказались от простой модели, когда за один такт может быть считана, декодирована, выполнена только одна команда. Современные ЦП обладают возможностью выполнения нескольких команд одновременно. Например, у ЦП могут быть отдельные модули, занимающиеся выборкой, декодированием и выполнением команд, и во время выполнения команды с номером `n` он может декодировать команду с номером `n+1` и считывать команду с номером `n+2`. Такая организация называется конвейером, и три его стадии можно проиллюстрировать схемой.

Встречаются и более длинные конвейеры. Более передовым по сравнению с конвейерной конструкцией является суперскалярный ЦП. В этой структуре присутствует множество выполняющих узлов: один - для целочисленных арифметических операций, второй - для операций с плавающей точкой, один - для логических операций. За один такт считывается две или более команд, которые декодируются и сбрасываются в буфер хранения, где ждут своей очереди на выполнение. Когда выполняющее устройство освобождается, оно просматривает буфер хранения, выбирает команду, если она там есть, и выполняет ее.



В большинстве случаев АО должна гарантировать, что результат совпадает с тем, который выдала бы последовательная конструкция.

Большинство ЦП имеет два режима работы: режим ядра и пользовательский режим. Режимы задаются битом слова состояния ЦП, т.е. регистра PSW. Если ЦП запущен в режиме ядра, он может выполнять все команды из набора инструкций и использовать все возможности АО. ОС работает в режиме ядра, предоставляя доступ ко всему АО. В противоположность этому прерыванию пользователи работают в пользовательском режиме, разрешающем выполнение подмножества программ и делающем доступным лишь часть аппаратных средств. Как правило, все команды, включая I/O данных и защиту памяти, запрещены в пользовательском режиме. Установка бита режима ядра в регистре PSW программным способом недоступна. Для связи с ОС пользовательская программа должна сформировать системный вызов, который обеспечивает переход в режим ядра и активирует функции ОС.

Команда TRAP, называемая эмулированным прерыванием, переключает режимы работы ЦП из пользовательского в режим ядра, и передается управление ОС. После завершения работы системного вызова управление передается пользовательской программе, следующей за системным вызовом.

В компьютере, помимо инструкций или команд для выполнения системных вызовов есть и другие прерывания. Большинство этих прерываний вызывается аппаратно для предупреждения в исключительных случаях, таких как операции попытки деления на нуль ли переполнение при операции с плавающей точкой. В таких случаях управление переходит ОС, которая должна решать, что делать дальше.

Память

Второй основной составляющей любого компьютера является память. В идеале память должна быть максимально быстрой (быстрее, чем обработка одной инструкции, чтобы работа ЦП не замедляло обращение к памяти, достаточно большой и чрезвычайно дешевой. На сегодня не существует технологий, удовлетворяющих всем этим требованиям. Поэтому имеется другой подход.

Система памяти конструируется в виде иерархии слоев, которые иллюстрируются на следующем рисунке.

Верхний слой состоит из внутренних регистров ЦП, поэтому при доступе к ним не возникает задержек. Внутренние регистры хранят 32 нс 32 бита на 32-разрядном ЦП или 64x64 бита на 64-разрядном, это составляет менее 1 Кб в обоих случаях. Программы могут управлять регистрами без вмешательства.

В следующем слое находится кэш-память, в основном контролируемая аппаратурой. Она разделена на кэш-строки по 64 байта с адресацией от 0 до 63 в нулевой строке, от 64 до 127 в первой строке и т.д. Наиболее часто используемые строки кэша хранятся в высокоскоростной кэш-памяти, расположенной внутри ЦП. Когда программа должна прочитать слово из памяти, кэш-микросхема определяет, есть ли нужная строка в кэше; если это так, то происходит результативное обращение к кэш-памяти. Запрос удовлетворяется целиком из кэша, и запрос к памяти на шину не выставляется. Удачное обращение к кэшу по времени занимает около двух тактов, а неудачных приводит к обращению к ОЗУ и существенной потере времени. Кэш-память ограничена в размере, что обусловлено ее высокой стоимостью. В некоторых машинах есть два или три уровня кэша, причем каждый последующий медленнее и больше предыдущего.

Далее следует ОЗУ (RAM - Random Acces Memory - память с произвольным доступом) - главная рабочая область запоминающего устройства машины. Все запросы ЦП, которые не могут быть выполнены кэш-памятью, поступают для обработки в ОЗУ.

Далее - магнитный диск. Дисковая память на два порядка дешевле ОЗУ в пересчете на бит и на два порядка больше по величине. У диска есть одна проблема - случайный доступ к данным на нем занимает примерно на три порядка больше времени. Причиной низкой скорости HDD - диск представляет собой механическую конструкцию, он состоит из одной или нескольких металлических пластин, вращающихся со скоростями 5400, 7200 или 10700 об/мин. Механическая вилка с магнитными головками поворачивается над дисками подобно звукоснимателю. Информация записывается на пластины в виде концентрических окружностей. Магнитная головка каждой данной позиции вилки может прочитать кольцо на пластине, называемое дорожкой. Все вместе дорожки для заданной позиции вилки формируют цилиндр. Каждая дорожка разделена на некоторое количество секторов, обычно по 512 байт на сектор. Внешние цилиндры содержат большее количество секторов, чем внутренние. Перемещение головки от одного цилиндра к другому занимает около 1 мс, а перемещение к произвольному цилиндру требует от 5 до 10 мс в зависимости от диска. Когда сектор уже находится под головкой, процесс чтения или записи происходит со скоростью 5 Мб/с для низкоскоростных дисков до 160 Мб/с для высокоскоростных.

Магнитная лента часто используется для создания резервных копий HDD или для хранения очень больших наборов данных. Для доступа к информации на ленте ее нужно поместить в устройство для чтения магнитных лент (стример). Это делает человек или робот. Затем лента перематывается до запрашиваемого блока с информацией. Процесс может длиться минуту. Большое достоинство заключается в том, что они крайне дешевы и мобильны. Это важно для резервных копий, которые нужно содержать отдельно, чтобы они сохранялись после стихийных бедствий (пожара, наводнений). Магнитная лента хранится в другой комнате от компьютера. Срок хранения - 5 лет.

Кроме описанных видов, в компьютерах есть небольшое количество постоянной памяти с произвольным доступом. В отличие от RAM, она не теряет свое содержимое при выключении питания. Она называется ПЗУ или ROM. ПЗУ программируется в процессе производства и после этого его содержимое нельзя изменить. Эта память достаточно быстра и дешева. Программы начальной загрузки

компьютера, используемые при запуске, находятся в ПЗУ. Кроме этого, некоторые карты I/O содержат ПЗУ для управления низкоуровневыми устройствами.

Электрические стираемая ПЗУ EEPROM и flash-RAM также энергонезависимы, но их содержимое можно стереть и переписать. Поэтому они используются точно так же, как и ПЗУ. Однако запись данных в них требует намного больше времени, чем запись в ОЗУ. Дополнительное преимущество электрически стираемого ПЗУ и flash-памяти состоит в том, что с их помощью можно исправить ошибки, содержащиеся в программах.

Вид памяти, называемый CMOS (читается "смос"), является энергозависимым. CMOS используется для хранения текущей даты и времени и конфигурационных параметров, например, указания, с какого HDD производить загрузку. Эта память потребляет энергию от установленного аккумулятора.

Виртуальная память

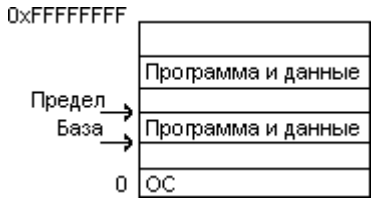
Вычислительной системе нужны средства для долгосрочного хранения информации после выключения компьютера. ОС решает эту задачу с помощью средств виртуальной памяти и ФС. ФС обеспечивает долгосрочное хранение информации, помещая ее в именованные объекты - файлы. Файл - удобная для использования структура данных, доступ к которой и ее защита осуществляется ОС.

Виртуальная память - устройство, позволяющее программистам рассматривать ОЗУ как логический объект, не интересуясь его физическим объемом. Принципы работы с виртуальной памятью были разработаны, чтобы задания нескольких пользователей, выполняясь параллельно, могли одновременно присутствовать в ОЗУ. Виртуальная память решает две задачи:

1) защищает программы друг от друга, а ядро ОС - от программ;
2) управляет перемещением программ в памяти.

Все возможные способы решения этих задач основаны на снабжении ЦП специальным оборудованием. Одно из простых решений - снабжение ЦП двумя специальными регистрами: базовым и предельным (граничным). При начале работы программы в базовый регистр помещается адрес ее начала, а в предельный - размер программы вместе с данными. При выборке команды из памяти аппаратура проверяет счетчик команд (PC), и если он меньше, чем предельный регистр, то добавляет к нему значение базового регистра и сумму передает в адресную шину.

Базовый регистр позволяет программе ссылаться на любую часть памяти, следующую за хранящейся в ней адресом. Предельный регистр запрещает программе обращение к памяти за границы программы. С помощью этой схемы решаются обе задачи защиты и перемещения программ. В результате проверки и преобразования адрес, сформированный программой и называемый виртуальным, переводится в адрес, используемый памятью и называемый физическим. Устройство, которое выполняет проверку и преобразование, называется диспетчером памяти - MMU (Memory Management Unit). Оно расположено в ЦП.



Более сложный диспетчер памяти содержит две пары базовых и предельных регистров. Одна пара - для текста программы, другая - для данных. Появляется возможность делить одну и ту же программу между несколькими пользователями и при этом хранить в памяти только одну копию программы.

Из-за различий в количестве памяти, требующейся для разных программ, их трудно компактно разместить в ОЗУ. Поэтому разработаны системы со страничной организацией памяти, когда программа разбивается на блоки фиксированного размера - страницы (1 страница = 4 Кб). В этом случае обращение программы к ячейке памяти происходит по виртуальной памяти, адрес которой состоит из номера страницы и смещения относительно ее начала. Страницы одной и той же программы могут быть разбросаны по всему ОЗУ. Система разбивки на страницы обеспечивает динамическое соответствие между виртуальным адресом, использующимся программой, и реальным (физическим) адресом ОЗУ. Если программа обращается к странице, отсутствующей в ОЗУ, то диспетчер памяти обнаруживает это и загружает недостающую страницу.

На характеристики памяти в основном влияют два аспекта:

1) кэш скрывает низкую скорость ОЗУ. Когда ОС переключается от одной программы к другой, кэш

остается заполненным данными первой программы, а необходимые строки новой программы должны загружаться из физической памяти. Эта операция может стать главной причиной снижения производительности, если происходит слишком часто;

2) при переключении программ регистры управления памятью должны меняться. Вне зависимости от количества этих регистров эта операция занимает некоторое время. Переключение от одной программы к другой - переключение контекста.

Тема 3. Общие сведения о процессах и потоках

УВВ

ОС взаимодействует с УВВ как с ресурсами. УВВ тоже тесно взаимодействуют с ОС. УВВ обычно состоят из контроллера и самого устройства.

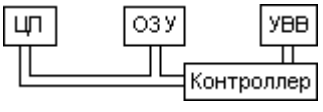
Контроллер - набор микросхем на вставляемой в разъем плате, физически управляющее устройство. Он принимает команды ОС (например, указания прочитать данные с устройства) и выполняет их. Фактическое управление устройством очень сложно и требует высокого уровня детализации. Поэтому в функции контроллера входит представление простого ин7терфейса для ОС.

Следующей частью является само устройство. Устройства имеют достаточно простые интерфейсы, потому что их возможности невелики и их нужно привести к единому стандарту, который необходим, чтобы каждый IDE контроллер диска (Integrated Drive Electronics - встроенный интерфейс накопителя) мог управлять любым IDE диском. IDE интерфейс является стандартным для дисков на компьютерах с ЦП Pentium, а также на других компьютерах. Т.к. настоящий интерфейс устройства скрыт с помощью контроллера, ОС видит только интерфейс контроллера, который может сильно отличаться от интерфейса самого устройства.

Поскольку все виды контроллеров отличаются, то для них требуется разное ПО. Программа, которая общается с контроллером, - драйвер устройства. Каждый производитель контроллеров должен поставлять драйверы для поддерживаемых ОС. Для использования драйвера его нужно установить в ОС так, чтобы он мог работать в режиме ядра. Есть три способа установки драйвера в ядро:

1) заново скомпоновать ядро вместе с новым драйвером и затем перезагрузить ОС (так работает множество ОС Unix);
2) создать запись во входящем в ОС файле, говорящую о том, что требуется драйвер и затем перезагрузить ОС; во время начальной загрузки ОС сама находит нужные драйверы и загружает их (так работает Windows);
3) ОС может принимать новые драйверы, не прерывая работы, и оперативно устанавливать их, не нуждаясь в перезагрузке. Этот способ становится все более и более распространенным. Такие устройства как шины USB, IEEE 1394 всегда нуждаются в динамически загружаемых драйверах.

Для связи с каждым контроллером существует небольшое количество регистров. Например, минимальный контроллер диска может иметь регистр для определения адреса на диске, адреса в памяти, номера сектора и направления операции (чтение или запись). Чтобы активизировать контроллер, драйвер получает команду от ОС, затем транслирует ее в величины, подходящие для записи в регистры устройства. На некоторых компьютерах отображаются в адресное пространство ОС, поэтому их можно читать или записывать как обычные слова в памяти, т.е. на таких машинах не нужны специальные команды I/O. На других компьютерах регистры устройств располагаются в специальных портах I/O, и каждый регистр имеет свой адрес порта. На этих машинах в режиме ядра доступны команды `IN` и `OUT`. Они позволяют драйверам считывать и записывать регистры. Первая схема устраняет необходимость специальных команд I/O, но использует некоторое количество адресного пространства. Вторая схема не затрагивает адресного пространства, но требует наличия специальных команд. Обе схемы широко используются.



I/O данных можно осуществлять тремя различными способами.

1. Простейший способ: пользовательская программа выдает системный запрос, который ядро транслирует в вызов процедуры, соответствующей драйверу, затем драйвер начинает процесс I/O. В этом время он выполняет короткий программный цикл, постоянно опрашивая устройство, с которым он работает (есть бит, указывающий занятость устройства). При завершении

операций I/O драйвер помещает данные туда, куда требуется, и возвращается в исходное состояние. Затем ОС возвращает управление программе, осуществлявшей вызов. Этот метод - ожидание готовности (активное ожидание). Он имеет один недостаток: ЦП должен опрашивать устройство, пока оно не завершит работу.

2. Драйвер запускает устройство и просит его выдать прерывания по окончании I/O; после этого драйвер возвращает управление ОС, и она начинает выполнять другие задания. Когда контроллер обнаруживает окончание передачи данных, он генерирует прерывание о завершении операции, Процесс I/O, использующий прерывания, состоит из четырех шагов (ступеней). На первом шаге драйвер передает команду контроллеру, записывая информацию в регистры устройств. Затем контроллер запускает устройство. Когда контроллер заканчивает чтение или запись того количества байтов, которое ему было указано передать, он посылает сигнал микросхеме контроллера прерываний, используя определенные провода шины. Это шаг второй. На третьем шаге если контроллер прерываний готов к обработке прерываний, то он подает сигнал на определенный контакт ЦП, информируя его таким образом. На четвертом шаге контроллер прерываний вставляет номер устройства на шину, чтобы ЦП мог узнать, какое устройство завершило работу.

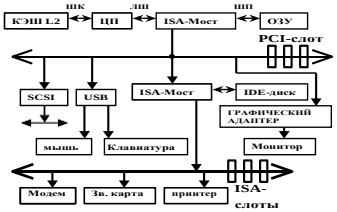
Когда Центральный процессор (ЦП) принимает прерывание, содержимое счетчика команд и слов состояние процессора помещается в текущий стек, а процессор переключается в режим работы ядра. Номер устройства используется как адрес части памяти, хранящей адрес обработчика прерываний данного устройства. Эта часть памяти называется вектором прерывания. Когда обработка прерывания начинает свою работу он удаляет расположенные в стеке: счетчик команд слово состояние процессора, сохраняет, и запрашивает устройство, чтобы получить информацию об его состоянии. После того как обработка прерываний целиком завершена, управление возвращается к рабочей до этого программе пользователя.

Третий метод вв-выв информации заключается в использовании специального контролера прямого доступа к памяти (Direct Memory Access). Который управляет потоком битов между Оперативной памятью (ОП) и некоторыми контролерами без вмешательства ЦП. ЦП обращается к микросхеме DMA, сообщает ей число байтов для передачи, а также адрес устройства и памяти, а также направление передачи данных. По завершении работы DMA инициирует прерывание, которое обрабатывается обычным порядком.

ЦП может запрещать прерывание и разрешать их позже. Пока прерывания запрещены все устройства завершившие работу продолжают посылать свои сигналы , но работа ЦП не прерывается до тех пор, пока прерывания не будут разрешены. Если работу заканчивают сразу несколько устройств, в то время когда прерывания запрещены, контролер прерывания решает какое из них должно быть обработано первым, основываясь на статических приоритетах назначенных для каждого устройства.

Шины.

Из-за роста и быстродействия ЦП и памяти, в систему добавились дополнительные шины как для ускорения общения устройств вв-выв, так и для пересылки данных между ЦП и памятью. Вследствие этой эволюции вычислительная система выглядит так:



ШК- Шина Кэш
ЛШ- Локальная Шина
ШП- Шина Памяти

В этой системе 8 шин, каждая со своей скоростью передачи данных и своими функциями. В ОС для управления компьютером должны находится сведения об всех этих шинах.(ISA -- Industry Standard Architecture ; PCI – Peripheral Component Interconnect).

Шина ISA работает на частоте 8,33 МГц и может передавать 2 байта за такт с максимальной скоростью16,67 Мб/с.

Шина PCI работает на частоте 66 МГц и передает по 8 байт за такт с максимальной скоростью 528Мб/с.

Большинство высокочастотных устройств вв-выв используют шину PCI. ЦП по ЛК передает данные микросхеме PCI-моста, – который в свою очередь обращается к памяти по выделенной шине, часто работающей на частоте 100МГц.

Система Pentium имеет КЭШ первого уровня L1 встроенный в процессор и намного больше КЭШ второго уровня L2, подключенный к процессору отдельной ШК. В систему входят 3 специальных шины IDE, USB и SCSI. IDE служит для присоединения периферийных устройств к системе (CD-ROM). USB (Universal Serial Bus) служит для присоединения к компьютеру медленных устройств вв-выв, таких как клавиатура, мышь, принтер и т.д. USB – это централизованная шина по которой главное устройство каждую миллисекунду опрашивает устройство вв-выв. Она может управлять загрузкой данных со скоростью 1,5Мб/с. Все USB используют один драйвер, поэтому нет необходимости устанавливать драйвер для нового USB, т.е. они присоединяются к системе без ее перезагрузки. SCSI –(Small Computer System Interface) это высокопроизводительная шина, применяемая для быстрых дисков, сканеров и др. устройств, нуждающиеся в значительной пропускной способности, ее производительность 160 Мб/с. Шина SCSI используется в системах Макинтош, популярна в UNIX-системах и некоторых системах на базе Intel.

IEEE 1394 (Fire Wire). Эта шина является последней шиной, скорость ее 50Мб/с. Это свойство позволяет подключать к компьютеру портативные цифровые видеокамеры и мультимедийные устройства. Не имеет центрального контролера. В настоящее время шина SCSI и IEEE 1394 конкурируют с разработкой более быстрой версии шины USB.Стандарт под названием Plug and Play позволяет системе автоматически собирать информацию об устройствах вв-выв. Назначать уровни прерывания и адреса вв-выв, а затем сообщать каждой плате эту информацию. На материнской плате находится программа называемая системой BIOS (Basic Input Output System) – она содержит программы вв-выв низкого уровня, включая процедуры для чтения с клавиатуры, вывода информации на экран, вв-выв данных с диска. В настоящее время эти функции хранятся во Flash-ОЗУ, которая в обычных условиях является неизменяемой, но ее можно изменить с помощью ОС.

При изучении ОС в них принято выделять следующие части:

1. Процессор
2. Управление памятью
3. Защита информации и безопасность
4. Планирование и управление ресурсами
5. Структура системы

В основном развитие современных ОС также происходит по этим направлениям. Каждое из этих направлений можно охарактеризовать набором абстрактных принципов, разработанных для решения сложных прикладных программ.

Процессы.

Понятие процесса относятся к одному из основополагающих в ОС. Существует много определений термина процесс в том числе:

1. Выполняющаяся прикладная программа пользователя
2. Экземпляр программы, выполняющийся на компьютере
3. Объект, который можно индетифицировать и выполнять на процессоре
4. Единица активности, которую можно охарактеризовать единой цепочкой последовательных действий, текущим состоянием и связанных с ней набором системных ресурсов.

Как понятие процесс является определенным типом абстракции, и обычно следует придерживаться следующего неформального определения.

Последовательный процесс (задача) – выполнение отдельной программы и ее данные на последовательном процессоре.

В качестве примера можно назвать следующие процессы:

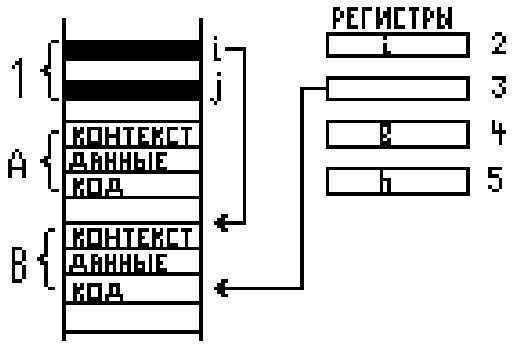
1. Выполняющаяся прикладная программа пользователя
2. Утилит
3. Трансляция программ
4. Компоновка, выполнение

Определение понятия процесс ставит цель выработать механизм распределения и управления ресурсами. Понятие ресурс как и понятие процесса является основным при рассмотрении ОС. Термин ресурс применяется к повторно используемым, относительно стабильным и часто недостающем объектом, которые запрашиваются, используются и освобождаются процессами в период их активности, т.е. ресурсом называется всякий объект, который может распределяться внутри системы.

Мысленно процесс можно разделить на 3 компонента:

1. Выполняющаяся программа
2. Данные нужные для работы

Контекст выполняющейся программы (execution context) или состояние процесса (process state) Включат в себя всю информацию нужную ОС для управления процессами и процессору для его выполнения. Данные, характеризующие это состояние, включают в себя содержимое различных регистров процессора, таких как программный счетчик и регистры данных, приоритет процесса и сведения о том, находится ли данный процесс в состоянии ожидания вв-выв.



1-список процессов
A-процесс A
B-процесс B
2-индекс процесса
3-PC счётчик команд
4-базовый регистр
5-граничный регистр

На рис. Показан пример реализации процесса. Два процесса A и B находятся в различных областях основной памяти. Каждому процессу отведён блок памяти, в к-ом создаётся код программы, данные и информация о состоянии процесса. Каждый процесс заносится в список процессов, к-ый создаётся и поддерживается ОС. Часть этого списка, соответствующая определённому процессу, содержит указатель размещения этого процесса в памяти. В регистре индекса процесса содержится индекс выполняющегося в текущий момент времени процесса, идентифицирующий его в списке процессов.

Содержимое команд (РС) указывает на очередную инструкцию, к-ую нужно выполнить. Базовый и граничный регистры задают область памяти, занимаемую процессом. В базовый регистр заносится адрес начальной ячейки памяти, а в граничный- её размер в байтах.

Содержимое програмного счётчика и всех ссылок на данные отсчитывается от значения базового регистра. По своей величине эти ссылки не могут превосходить значения граничного регистра. Регистр индекса процесса показывает, что выполняется процесс В, до этого выполняется процесс А, но он временно прерван. Содержимое всех регистров в момент прекращения этого процесса записано в виде данных о состоянии процесса. Впоследствии ОС сможет к выполнению процесса А. При этом будет сохранён контекст выполнения процесса В и восстановлен контекст процесса А. Когда в програмный счётчик загрузится значение, указывающее на область кода, программа процесса Аавтоматически возобновиться выполнение этого процесса. Таким образом процесс реализируется в виде структуры данных. Он может выполнятьсяили находиться в состоянии ожидания. Состояние процесса в каждый момент времени заносится в специально отведённую область данных. Использование структуры позволяет развить мощные методы координации и взаимодействия процесса. Расширяя и добавляя данные о состоянии процесса доп. информации можно разработать новые возможности ОС. В обычных ОС общего назначения процес может находиться в активном и пассивном состоянии.

В активном состоянии процесс участвует в конкуренции за исп-ие ресурса выч системы, а в пассивном- он только известен системе, но в конкуренции не участвует. В свою очередь активный процесс м/б в одном из следующих состояний:

1. *Выполнение*- все затребованные процессом ресурсы выделены. В этом состоянии в каждый момент времени может находиться только один процесс (В однопроцессорной выч системе)
2. *Готовность к выполнению*- ресурсы м/б предоставлены, тогда процесс перейдёт в состояние выполнения
3. *Блокирование или ожидание*- затребованные ресурсы не м/б предоставлены или не завершена операция вв/выв.

В большенстве ОС последнее состояние подразделяется на мн-во состояний ожиданий, соответствующих определённому виду ресурсов из-за отсутствия которого процесс переходит в заблокированное состояние.

В обычных ОС процесс появляется при запуске какой-нибудь программы ОС пораждает для нового процесса соответствующий дескриптор процесса и процесс начинает выполняться. Поэтому пассивного состоянияне существует. В ОС реального времени уже заранее бывает известен состав программ которые должны выполняться. Известны и многие их параметры которые необходимо учитывать при распределении ресурса. Например: объём памяти, приоритет, средняя длительность выполнения, открываемые файлы, исп-ые устройства и т. д. Поэтому для них заранее заводят дискрипторы процесса, чтобы в последствии не тратить время на организацию дискрипторов таким образом в ОСРВ многие процессы находятся в состоянии бездействия.

Процесс состояния бездействия может перейти в состояние готовности в следующих случаях:

1. По команде оператора.
2. При выборе из очереди планировщиком.
3. По вызову из другой задачи. Один процесс может создать, инециировать, приостановить, уничтожить другой процесс.
4. По прерыванию от внешнего инециативного устройства. (Сигнал о завершении некоторого события может запускать соответствующий процесс).
5. При наступлении запланированного времени запуска программ.

Последние два способа запуска характерны для ОСРВ (реал. времени).

Из состояния выполнения процесс может выйти по одной из следующих причин:

1)Процесс завершается, при этом он передаёт управление ОС и сообщает о своём завершении.

2)Процесс переводится ОС в состояние готовности к выполнению в связи с появлением более приоритетной задачи и в связи с окончанием выделенного ему кванта времени.

3)процесс блокируется (переводится в состояние ожидания) либо в силу невозможности предоставить ему ресурс, запрошенный в настоящий момент. При наступлении соответствующего события (завершилась операция вв/выв, освободился затребованный ресурс в опер. памяти, загружена необходимая страница виртуальной памяти и т. д.) процесс деблокируется и переводится в состояние

готовности к исполнению. Таким образом движущей силой, меняющей состояние процесса является событие. Один из ост. видов- прерывание.

Для того чтобы ОС могла управлять процессами, она должна распологать необходимой для этого информацией. С этой целью на каждый процесс заводится специальная специальная информационная структур, называемая диструктором процесса. (описатель задачи, блок управления задачей) Деструктор содержит следующую информацию:

1. Идентификатор процесса (Process Identificator (ID))
2. Тип или класс процесса, который определяет для ОС некоторые правила предоставления ресурсов.
3. Приоритет процесса. В соответствии с которым ОС предоставляет ресурсы В рамках одного класса процессов в первую очередь обслуживается более приоритетный процесс.
4. Переменную состояния, которая определяет в каком состоянии находится процесс (готов кработе, состояние выполнения, ожидание устройства ввода/вывода и т. д.)
5. Защищённую область памяти в которой хранится текущее значение регистров процессора, если прерывается не закончив работу. Эта информация называется контекстом процесса(задачи).
6. Информацию о ресурсах, которыми процесс владеет и имеет право пользоваться (указатели на открытые файлы, инфа о независимых операциях вв/выв и т. т.)
7. Место памяти или адрес этого места для организации общения с другими процессами.
8. Параметры времени запуска. (момент времени, когда процессор должен активизироваться и периодичность этой операции).

Дескрипторы процесса постоянно находятся в опер. памяти, чтобы ускорить работу ОС, которая организует их в списки (очереди) и отображает изменение состояния процесса, перемещением соответствующего дескриптора из одного списка в другой. В некоторых ОС количество дескрипторов определяется жёстко и заранее на этапе генерации варианта ОС или в конфигурационном файле, который исполняется при загрузке ОС. В других ОС по мере необходимости система выделяет участки памяти под новые дескрипторы.

Для аппаратной поддержки работы ОС с дескрипторами задач процессора реализованы соответствующие механизмы. Например процессора intel (i80*86) имеется специальный регистр Task Register (TR) указывающий местонахождение сигмента состояния задачи Task State Segment (TSS) в котором при переключении с процесса на процесс автоматически сохраняется содержимое регистров процессора.

ПОТОКИ

Концепцию процесса можно охарактеризовать двумя параметрами:

1. Владение ресурсами.

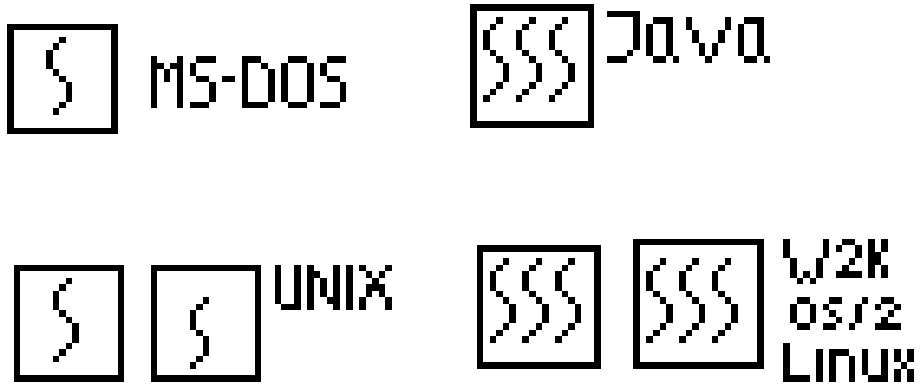
Процесс включает виртуальное адресное пространство в котором содержится образ процесса и время от времени может владеть такими ресурсами, как основная память, устройство вв/выв, файлы. ОС выполняет защитные функции предотвращая нежелательные взаимодействия процессов.

2. Планирование и выполнение.

Выполнение процесса осуществляется путём выполнения кода программы при этом выполнение процесса может чередоваться с выполнением других процессов. Таким образом процесс имеет такие параметры как состояние и текущий приоритет, в соответствии с которым ОС осуществляет его планирование и диспетчеризацию.

В большинстве ОС эти две характеристики являются сущностью процесса. Однако они являются независимыми и ОС может рассматривать их отдельно друг от друга. Чтобы различать эти две характеристики единицу диспетчеризации называют потоком (термин нить, волокно), а единицу владения ресурсами – процессом или задачей. Многопоточностью называется способность ОС поддерживать в рамках одного процесса выполнение нескольких потоков.

Традиционный подход, при котором каждый процесс представляет собой единый поток выполнения называется однопоточным. Например MS-DOS.



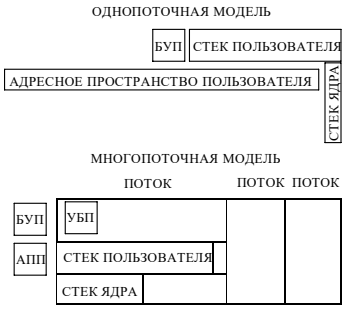
В однопоточной модели процесса, в его представление входит управляющий блок этого процесса и пользовательского адресного пространства, а также стеки ядра и пользователя, с помощью которого осуществляются вызовы процедур и возвраты из них при выполнении процесса, когда выполнение процесса прерывается, содержимое регистра процессора сохраняе-тся в памяти, в много-пользовательской сре-де с каждым процесс-сом также связан уп-равляющий блок и адресное прост-ранство, но теперь для каждого потока соз-даются свои отдельные стеки, а также свой уп-равляющий блок, в котором содержатся значения регистра, приоритет и другая информация о сос-тоянии потока.

Таким образом все потоки процесса разделены между собой состоянием и ресурсами этого процесса, они нахо-дятся в одном и том же адресном пространстве и имеют доступ к од-ним и тем же данным. Если один поток изме-няет в памяти какие-то данные, то другой поток, во время этого доступа к этим данным , может отследить эти изменения. Если один поток открывает файл с правом чтения, то другие потоки данного процесса могут читать данные из этого файла. Основные преимущества использования потоков:1)создание нового потока в уже существующем процессе занимает меньше времени, чем создание нового процесса2) поток можно завершить быстрее чем процесс3)переключение потоков в рамках одного и того же процесса происходит быстрее4)при использовании потока повышается эффективность обмена информацией между двумя выполняющимися программами. В большинстве о.с. обмен между независимыми процессами происходит с участием ядра, в функции которого входит обеспечение защиты и механизма необходимого для осуществления обмена. Однако благодаря тому, что различные потоки одного и того же процесса используют одну и ту же область памяти и одни и те же файлы, они могут обмениваться информаций без участия ядра.

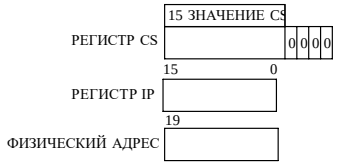
Таким образом если приложение или функцию надо реализовать в виде набора взаимосвязанных модулей, намного эффективнее реализовать ее в виде потока, чем в виде набора отдельных процессов. Пример программы, где удачно применяются потоки является файловый сервер. При получении каждого нового файлового запроса программа, управляющая файлами, порождает новый поток, из-за того, что серверам приходится обрабатывать очень большое количество запросов за короткий промежуток, будут создаваться и удалятся много потоков. Если такая серверная программа работает на многопроцессорной машине, то на разных процессорах в рамках одного процесса могут одновременно выполнятся несколько потоков, кроме того из-за того , что процессы или потоки файлового сервера должны использовать совместно данные из файлов , рациональнее использовать потоки и общую область памяти, а не процессы и обмен сообщениями. Потоковая конструкция процессов полезна и на однопроцессорных машинах, так как она помогает упростить структуру программы, выполняющую несколько логически различных функций.

Реальные и защищенные режимы работы процессора.

Первые микропроцессоры фирмы INTEL предназначались для работы в однозадачном режиме, то есть



специальных аппаратных средств для поддержки надежности и эффективных мультипрограммных о.с. в них не было. Поэтому для совместимости с однопроцессорными компьютерами в последних версиях микропроцессорах была реализована возможность использовать их в двухпроцессорном режимах: реальном и защищенном.Защищенный режим означает , что параллельное вычисление защищено аппаратно-програмными средствами. При работе микропроцессора в реальном режиме обращение к памяти с возможным адресным пространством в одном мегабайт, осуществляется посредством механизма сегментной адресации. Этот механизм был использован для увеличения количества разрядов, участвующих в образовании ячеек памяти с 16 до 20, тем самым увеличения объема памяти. Для определения физического адреса команды, содержимое сегментного регистра умножают на 16, за счет добавления СП-рава 4 нулей, после чего к полученному значению прибавляют содержимое указателя команды, получим 20-и битное значение, которое и позволяет указать любой байт из количества, равного 2^{20}.



В защищенном режиме определение физии-ческого адреса осущ-ествляется испол-ьзованием сегментных механизмов для орган-изации виртуальной памяти. При этом адр-есат задается 32-битн-ым значением, кроме того возможна стран-ичная трансляция адресов с 32-битного значения. При работе в защищенном режиме, который по умолчанию предполагает 32-битный код, возможно использование двоичных команд, созданных для работы в 16-битном режиме. Для этого введем режим виртуальной машины и 20-битные адреса реального режима, которые транслируются с помощью страничного механизма в 32-битное значение защищенного режима.

Защита адресногопространства задач.

Для возможности создания надежных мультипрограммных о.с. в процессорах семейства i80x86 имеются несколько механизмов защиты: это и разделение адресных пространств задач и введение уровней привилегий для сегментов кода и сегмента данных, все это позволяет обеспечить как защиту задач друг то друга, так и защиту самой о.с. от прикладных задач, защиту одной части о.с. от других ее компонентов, защиту самих задач от своих ошибок. Защита адресного пространства задач осуществляется за счет того, что каждая задача имеет свое собственное локальное адресное пространство. О.с. должна корректно манипулировать дескрипторами таблиц.

Сами таблицы дескрипторов относятся к адресному пространству о.с. и имеют соответствующие привилегии доступа, исправлять их, задачи не могут. Для организации взаимодействия задач имеющих разные виртуальные пространства надо иметь общее адресное пространство, и здесь для обеспечения защиты самой о.с. используется механизм защиты сегментов с помощью уровней привилегий.

Уровни привилегий для защиты адресного пространства задач.

Для того, чтобы запретить пользователю задач модифицировать область памяти, принадлежащую о.с. надо иметь специальные средства, одного разграничения адресного пространства через механизм сегментов мало, так как можно указывать различное значение адреса начала сегмента и тем самым получать доступ к чужим сегментам, то есть надо в явном виде разграничивать системные сегменты от сегментов пользователя. этому были введены два режима процессора: пользовательский и ядра. В режиме ядра программа может выполнять все действия и иметь доступ по любым адресам в пользовательском режиме существует ограничения, чтобы обнаруживать и пресекать запрещенные

действия, перехватывать их и передавая их управление ядру о.с., в пользовательском режиме запрещается выполнение команды ввода-вывода и некоторые других, чтобы гарантировать , что только о.с. выполняет эти операции, то есть эти два режима имеют разные уровни привилегий. Микропроцессоры i80x86 имеют 4 уровня привилегий, часто уровни привилегий называются кольцами защиты, так как это помогает объяснить принцип действия самого механизма. Для указания уровня привилегий используются два вида, поэтому код 00 обозначает самый высший уровень, а 11 низший. Самый высший уровень привилегий предназначен для ядра о.с. , низкий-прикладных задач.Промежуточные уровни привилегий введены для большей свободы системных программистов в организации надежных вычислений при создании о.с. Предполагается, что уровень с кодом 1 может быть использован для систем сервиса-программ, обслуживающих аппаратуру, работу с портами вводавывода. Уровень привилегий с кодом 2 может быть использован для создания пользовательских интерфейсов, систем управления базами данных, то есть для реализации специальных системных функций, которые по отношению к ядру о.с. ведут себя как обычные приложения. На практике используются только два уровня:0 и 3.

Таким образом, режим ядра для микропроцессора i80*86 соответствует выполнению кода с уровнем привилегии 0 (PL0). Именно уровень привилегий задач определяет, какие команды в них можно использовать, какое подмножество сегментов или страниц в их адресном пространстве они могут обрабатывать.

Основными системными объектами, которыми манипулирует процессор в защищенном режиме, являются дескрипторы. Дескрипторы сегментов содержат информацию об уровне привилегий соответствующего сегмента кода или данных.

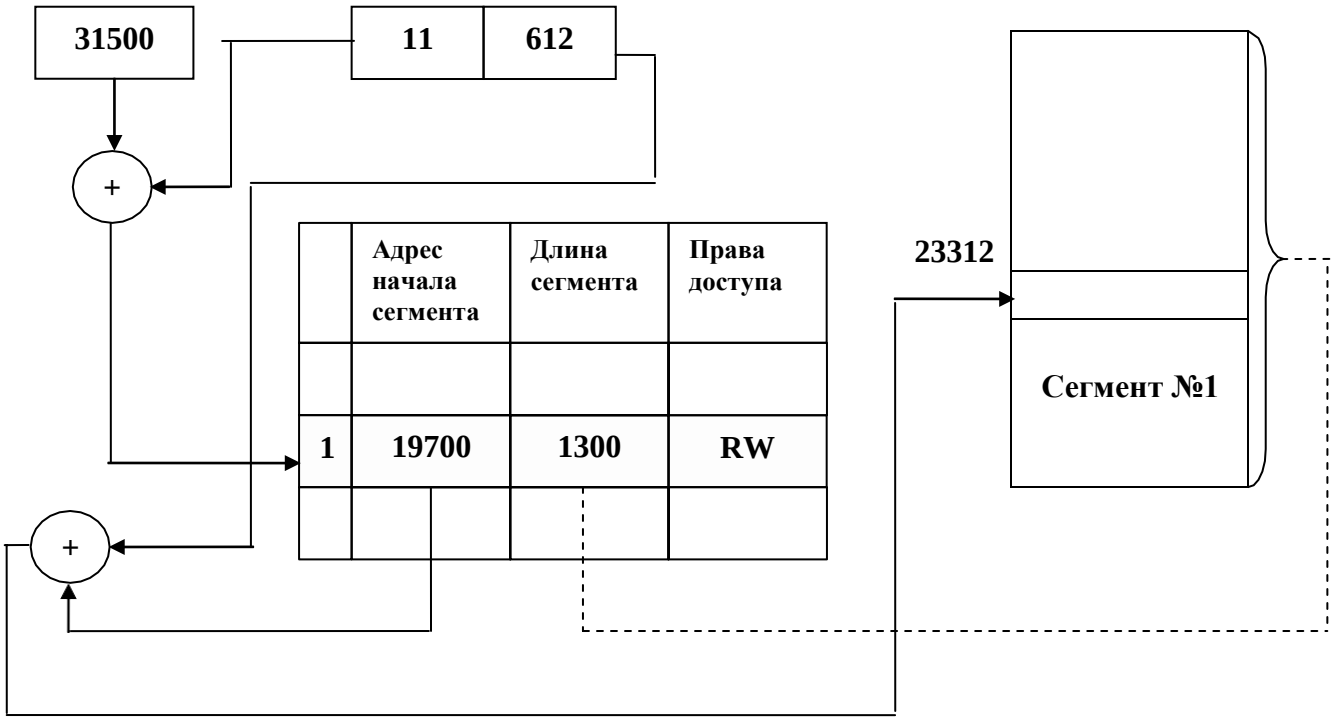
Тема 5. Управление памятью

Способы организации виртуальной памяти.

Методы распределения памяти, при которых задаче не может предоставляться смежная область, называют разрывными. Выделение оперативной памяти фрагментами требует для своей реализации аппаратной поддержки – нужно иметь относительную адресацию. Если указывать адрес начала текущего фрагмента программы и величину смещения относительно этого начального адреса, то можно указать нужную команду, т.е. виртуальный адрес, состоящий из двух полей. Программист может либо сам разбивать программу на фрагменты либо сделать это с помощью системы программирования.

Сегментный способ организации виртуальной памяти.

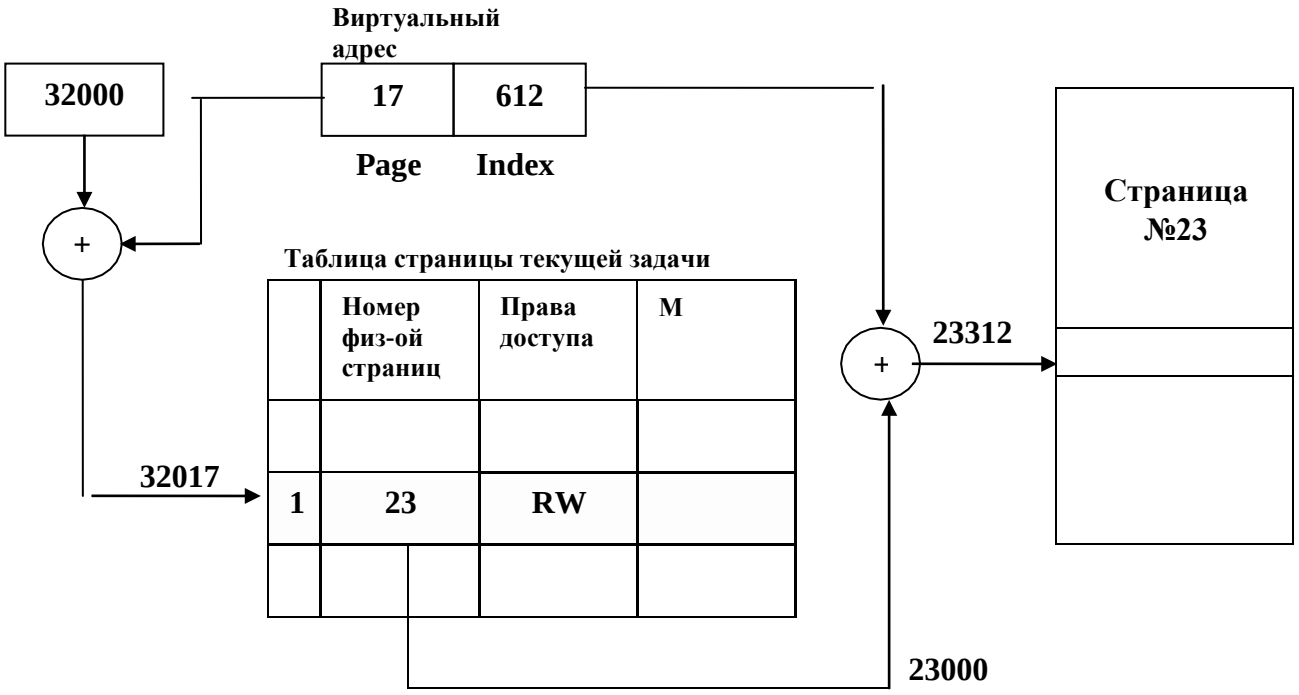
Естественным способом разбиения программы на части является её разбиение на логические фрагменты - сегменты. Обычно сегментом является модуль или совокупность программных модулей. Логическое обращение к элементам программы будет означать указание имени сегмента, и смещения относительно его начала. Каждый сегмент, размещенный в памяти, имеет информационную структуру – дескриптор сегмента. Операционная система строит для каждого выполняемого процесса таблицу дескрипторов сегмента и при размещении каждого из сегментов в ОЗУ или внешней памяти в дескрипторе отмечается его текущее местоположение. Для этого используется бит присутствия. В поле «Адрес» диспетчер памяти записывает адрес физической памяти, с которого начинается сегмент, а в поле «Длина сегмента» его размер в байтах. Это поле используется для того, чтобы избежать наложения сегментов друг на друга и для того, чтобы контролировать, не обращается ли подзадача за пределы текущего сегмента (если это так то генерируется прерывание). Если сегмент находится во внешней памяти, то поля адреса и длины используются для указания адреса сегмента в координатах внешней памяти. В дескрипторе сегмента также содержатся данные о его типе (код или данные), права доступа, отметка об обращениях к сегменту. При передаче управления следующей задаче операционная система заносит в соответствующий регистр микропроцессора адрес таблицы дескрипторов сегментов этой задачи. Сама таблица дескрипторов сегмента также является сегментом данных, которые обрабатываются диспетчером памяти.



Страничный способ организации виртуальной памяти.

Способ разрывного размещения задач в памяти при котором все фрагменты задачи одинакового размера кратного степени двойки называется страничным, а фрагменты страницами. В этом случае память разбивается на физические страницы (кадры, фреймы). А программа разбивается на виртуальные страницы. Часть виртуальных страниц размещается в ОЗУ, а часть во внешней памяти. Место на жестком диске, где размещаются виртуальные страницы называют файлом подкачки или страничным файлом (SWAP-файл). Чтобы подчеркнуть, что записи этого файла-страницы замещают друг друга в ОЗУ в некоторых операционных системах виртуальные страницы располагаются не в файле, а в специальном разделе диска.

Физический адрес ячейки памяти определяется парой (P_p, i), а виртуальный (P_v, i). P_v – номер виртуальной страницы, P_p – номер физической страницы, а I – номер ячейки (индекс) внутри страницы. Для отображения виртуального адресного пространства на физическую память для каждой задачи необходимо иметь таблицы страниц для трансляции адресных пространств. Для описания каждой страницы диспетчер памяти операционной системы заводит соответствующий дескриптор. По номеру виртуальной страницы в таблице дескрипторов текущей задачи находится соответствующий элемент (дескриптор). Если бит присутствия равен единице, то данная страница находится в ОЗУ и в дескрипторе находится номер физической страницы, отведенной под данную виртуальную страницу.



Защита страничной памяти основана на контроле уровня доступа каждой страницы. Существующие уровни доступа: только чтения, чтение и запись, только выполнение. Каждая страница снабжается соответствующим кодом. При обращении к виртуальной странице отсутствующей в физической памяти возникает прерывание и управление передается диспетчеру памяти, который должен найти свободное место. Если свободной страницы нет, то диспетчер памяти, по одной из дисциплин замещения определит страницу подлежащую расформированию или сохранению во внешней памяти. На её место он разместит виртуальную страницу к которой было обращение из задачи. Диспетчер памяти выбирает для замещения ту страницу, на которую не было ссылки на протяжении наиболее длительного периода времени.

Дисциплина замещения – Least Recently Used связывает с каждой страницей время последнего её использования.

В Windows 2000 она называется FIFO. Страничный механизм приводит к тому, что без специальных аппаратных средств он существенно замедляет работу вычислительной системы. Поэтому обычно используют кэширование страничных дескрипторов. В микропроцессоре i80*86 используется кэш на 32 страничных дескриптора. Поскольку размер страницы в этих микропроцессорах 4Кб, то возможно быстрое обращение к 128Кб памяти.

Основным достоинством страничной организации является минимально возможная фрагментация, поскольку на каждую задачу может приходится по одной незаполненной странице.

Недостатки:

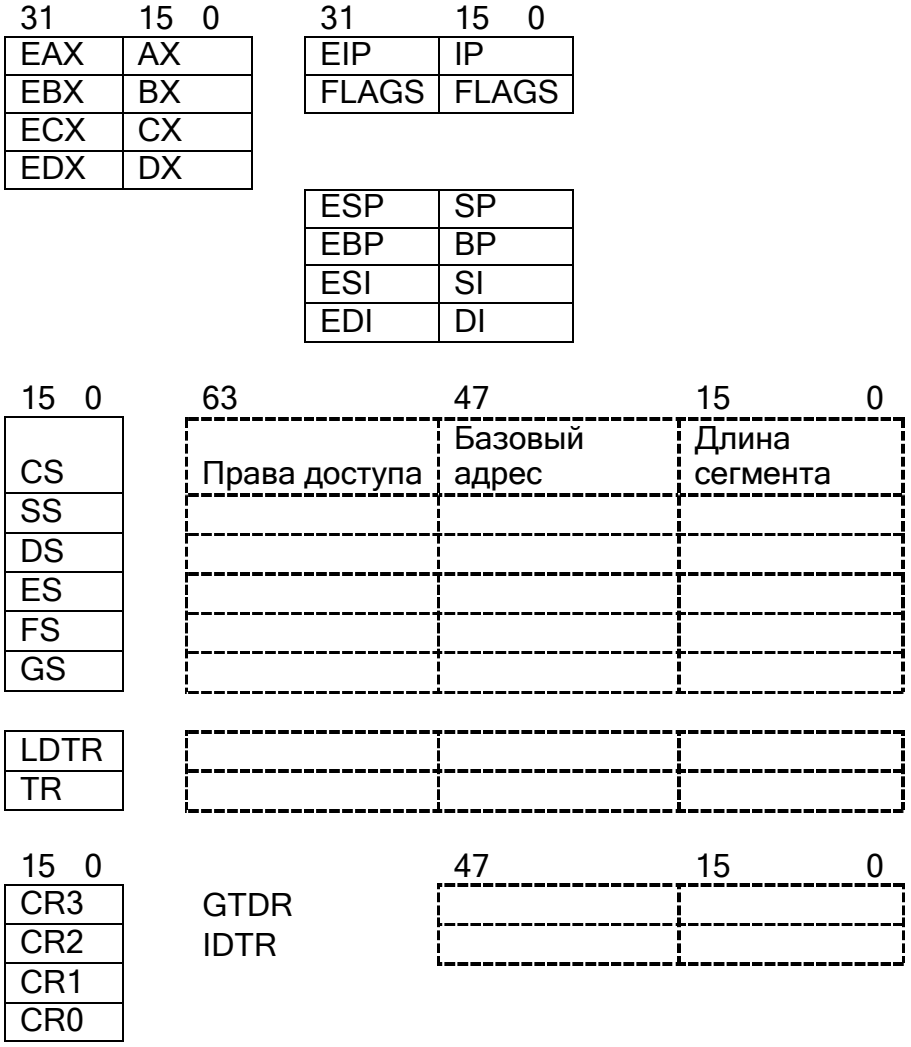
1) Накладные расходы, т.е. таблицы страниц нужно размещать в памяти и их нужно обрабатывать.

2) Программы разбиваются на страницы случайно без учета логических взаимосвязей имеющихся в коде программы. Поэтому межстраничные переходы осуществляются чаще нежели межсегментные и трудно организовать разделение программных модулей между выполняющимися программами.

Чтобы избежать второго недостатка сохранив достоинства страничного способа распределения памяти был предложен сегментно-страничный способ организации виртуальной памяти.

Сегментно-страничный способ организации виртуальной памяти.

Для понимания защищенного режима работы необходимо знать основные регистры процессора i80*86.



При каждом из сегментных регистров CS-GS изображены пунктиром скрытые от программистов, доступные только микропроцессору, 64 битовые регистры, в которые загружаются дескрипторы соответствующих сегментов. Регистр LDTR – регистр указатель на локальную таблицу сегментов текущей задачи, также имеет «теневой» 64 битовый регистр, в который микропроцессор заносит дескриптор указывающий на таблицу дескрипторов-сегментов задачи, описывающих её локальное виртуальное адресное пространство. Регистр - указатель задачи – TR (Task Register) указывает на дескриптор в глобальной таблице дескрипторов, чтобы получить доступ к дескриптору задачи Task State Segment – в информационной структуре для управления процессами (задачами).

Регистр Global Descriptor Table Register указывает на начальный адрес глобальной таблицы GDT регистров, содержащей дескрипторы общих сегментов.

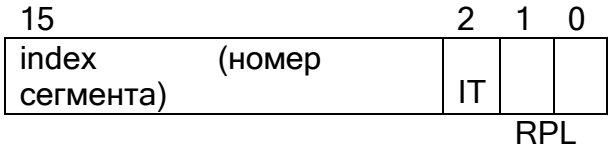
Регистр IDTR содержит информация для доступа к таблице прерываний IDT. Управляющий регистр CR0 содержит целый ряд блоков которые управляют режимом работы микропроцессора. Самый младший бит этого регистра Protect Enable определяет режим работы. Если PE равен 0 – это реальный режим, если 1 – защищенный. Старший бит этого разряда Paging, определяет включен (PG=1) или нет (PG=0) режим страничного преобразования адресов. Регистр CR2 содержит адрес подпрограммы, которая вызывается, если происходит обращение к отсутствующей странице. Для выполнения эффективной и надежной работы вычислительной системы в !!! режиме необходимо выполнение двух требований:

1) наличие у каждого процесса собственного (локального) адресного пространства непересекающегося с адресными пространствами других задач

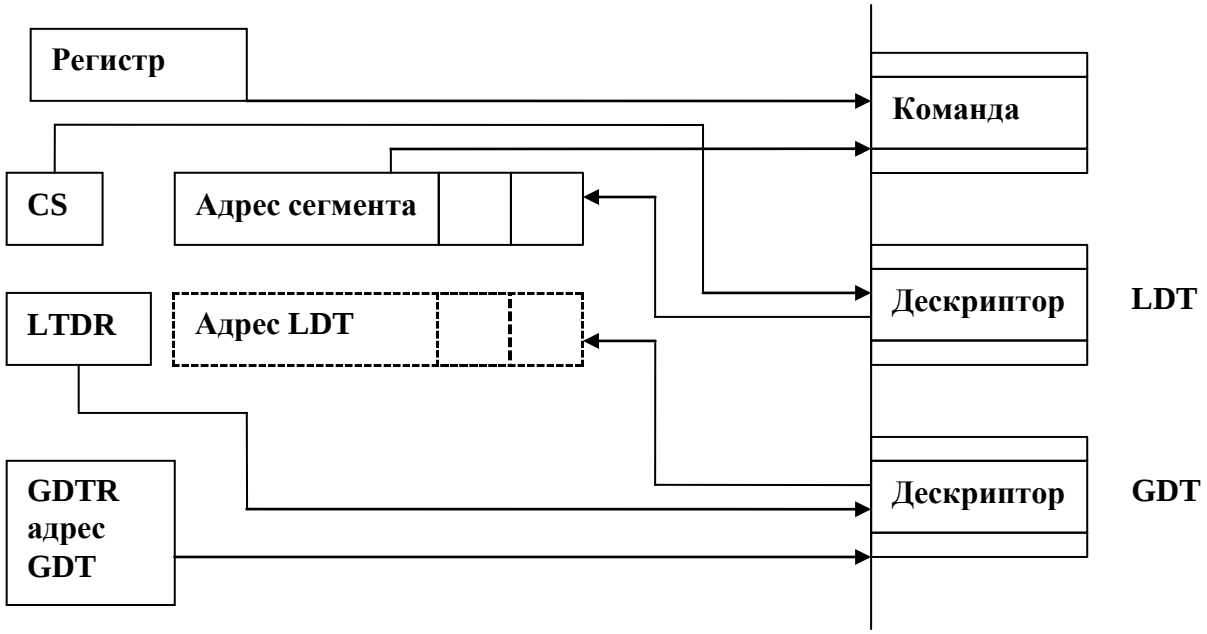
2) Наличие общего разделяемого адресного пространства, поэтому в микропроцессоре i80*86 реализован сегментный способ распределения памяти. Кроме этого, в этих микропроцессорах может быть и страничная организация памяти. Каждый сегмент описывается дескриптором сегмента в котором указывется базовый адрес сегмента, размер сегмента, права доступа и некоторая другая информация. Локальный адрес пространства процесса определяется через таблицу Local Descriptor Table.

Общее или глобальное адресное пространство определяется через другую таблицу Global Descriptor Table.

Заполнение этих таблиц и их модификацию осуществляет операционная система. При переключении микропроцессора в защищенный режим он совершенно иначе, чем в реальном режиме вычисляет физические адреса команд и операндов. Прежде всего содержимое сегментных регистров интерпретируется не как адрес начала сегмента, а как номер соответствующего сегмента. Поэтому сегментные регистры даже называются селекторами сегмента. При этом каждый сегментный регистр разбивается на три поля.



Поле индекса определяет номер сегмента в таблице дескрипторов. Поле индикатора в таблице сегментов (Table Index)(это бит с номером два) определяет общий это сегмент (0) или локальный (1). Это пространство описывает локальную таблицу дескрипторов IDT. Поле уровня привилегии – биты 0 и 1 указывают запрашиваемый уровень привилегии (Request Privilege Level). При своем запуске операционная система инициализирует GDTR. Этот регистр содержит начальный адрес глобальной таблицы дескрипторов и её размер. Для управления процессами операционная система имеет информационную структуру – дескриптор процесса. Часть дескриптора процесса, с которой работает микропроцессор называется сегментом состояния задачи (Task State Segment). В основном этот сегмент содержит контекст задачи. Процессор получает доступ к TSS с помощью регистра задачи TR. Регистр TR содержит индекс т.е. номер элемента глобальной таблицы дескрипторов сегмента TSS. Дескриптор заносится в теневую часть регистра. В одном из полей TSS находится указатель, т.е. селектор на локальную таблицу дескрипторов данной задачи. При переходе процессора с одной задачи на другую содержимое поля заносится микропроцессом в одноименный регистр. Линейный адрес команды определяется следующим образом – микропроцессор анализирует бит TI в селекторном регистре поля и в зависимости от его значения извлекает из глобальной либо локальной таблицы дескриптор сегментного кода с номером который равен содержимому поля индекс. Этот дескриптор заносится в теневую часть регистра CS. Микропроцессор сравнивает значения регистра EIP с полем размера сегмента, содержащегося в извлеченном сегменте и если смещение относительно начала сегмента не превышает размера сегмента, то значения EIP прибавляется к значению поля начала сегмента и получается искомый линейный адрес. Линейный адрес будет либо равен физическому (если страничное преобразование отключено), либо с помощью страничной трансляции преобразуется физический адрес. Если смещение из регистра EIP превышает размер сегментного кода, то это аварийная ситуация вызывает прерывание, и управление предается ядру операционной системы.



31 12	11 9	8 7	6	5	4 3	2	1
Адрес таблицы страниц	Для операционной системы	OO	Dirty	Acces	OO	User/Ядро	Read/Write

Дескриптор для описания страницы имеет следующий формат.

Прежде всего микропроцессор анализирует 0-ой бит. Если поле Prescut равно 0, то данная страница отсутствует в ОЗУ. Бит Dirty отмечает что данная страница модифицирована и её необходимо сохранить на винчестере при её замещении. Бит управления Access устанавливается в 1, если к данной странице установлен доступ. Этот бит используется для определения страницы которая будет замещаться.

Первый и второй биты используются для защиты памяти.

Основные понятия и концепции организации ввода/вывода.

По функциональному назначению внешние устройства подключаемы к компьютеру делятся на следующие категории:

1) Устройства внешней памяти:

a) Устройства памяти с произвольным доступом (дискеты, магнитные, оптические и магнитооптические диски) Direct Access Storage Device.

b) Устройства памяти с последовательным доступом (стримеры)

2) Сетевые и коммуникационные устройства (модем, сетевые адаптеры)

3) Устройства алфавитно-цифрового ввода/вывода (телетайпы, текстовые терминалы)

4) Устройства звукового ввода/вывода

5) Устройства графического ввода/вывода (сканер, плоттеры, видео кодеры)

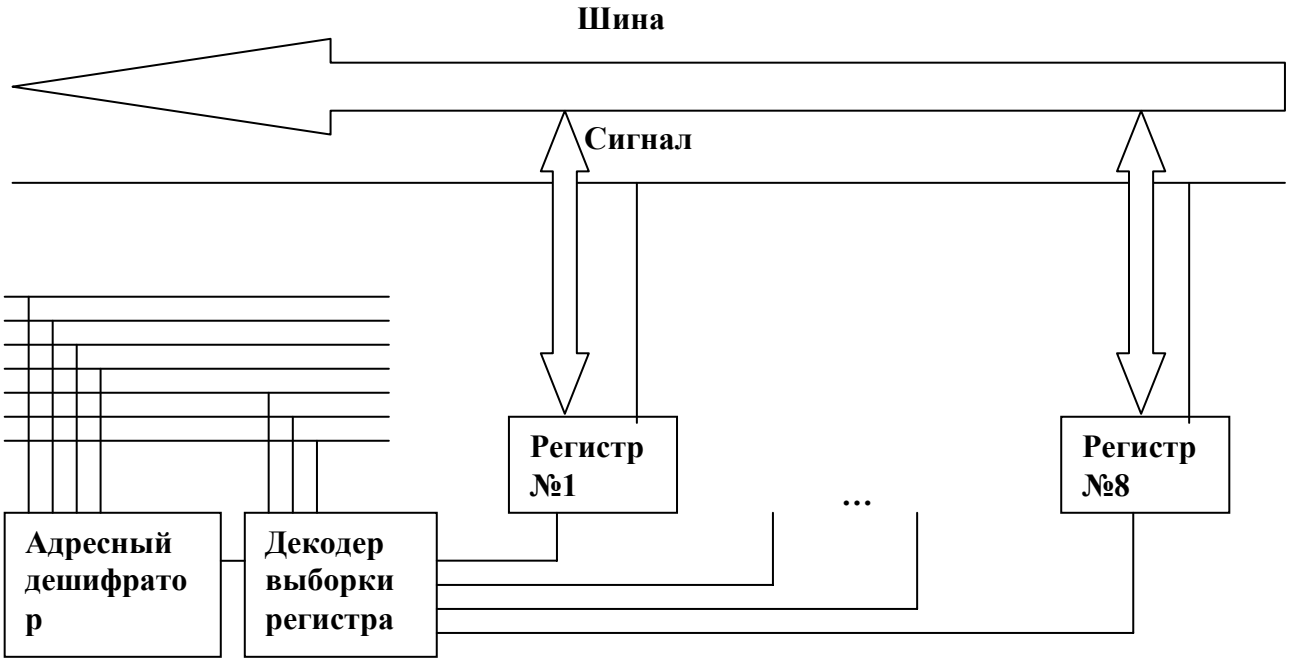
6) Позиционные устройства ввода/вывода (дигитайзеры, световые перья, мыши)

7) Датчики и исполнительные устройства управляющих систем

Нередко в эту классификацию вводят ещё один уровень. Устройства ввода делят на пассивные (выдающие данные только по команде центрального процессора) и активные (могут порождать данные

по своей инициативе – клавиатуры, мыши, сетевые адаптеры, таймеры и датчики управляющих устройств).

По отношению центрального процессора к выполняющейся на нем программе внешние устройства представляют собой набор специализированных ячеек памяти или регистров. Регистры устройств подключаются к шинам адреса и данных вычислительной системы. Внешнее устройство имеет адресный дешифратор. Если выставленный по шине адрес соответствует адресу одного из регистров устройства дешифратор подключает соответствующий регистр к шине данных. Таким образом регистры устройства получают адреса в физическом адресном пространстве микропроцессора



.

Существует два основных подхода к адресации этих режимов:

1) отдельные адресные пространства ввода/вывода
2) отображаемый в память ввод/вывод, когда память и регистры внешнего устройства размещаются в одном адресном пространстве

В первом случае для обращения к регистрам используется команды IN и OUT. Во втором случае могут использоваться любые команды работающие с операндами в памяти.

Тема 6. Файловая система и ввод и вывод информации

Процессор использует шины адреса и данных, но имеет дополнительный сигнал адресной шины, указывающей какое из адресных пространств используется в данном случае.

Используют 2 основных подхода к выделению адресов

- Фиксированная адресация, когда одно и тоже устройство всегда имеет один и тот же адрес реестра.
- Географическая, когда каждому разъёму периферической шины соответствует свой диапазон адресов.

При географической адресации перемещение платы устройства в другой разъём приводит к переконфигурации операционной системы. Однако, этот способ исключает возможность конфликта адресов между устройствами разных производителей или между двумя однотипными устройствами.

Большинство периферийных шин современных компьютеров (PSI) организуют географическую адресацию. Многие операционные системы требуют, чтобы устройства имели конфигурационные регистры через обращение к которым операционная система может выдать информацию об устройстве:

1. фирму изготовителя
2. модель
3. версию
4. количество регистров

Наличие регистров позволяет операционной системе автоматически определять установленное оборудование и подгружать соответствующие управляющие модули.

Режимы управления вводом/выводом.

Управление вводом/выводом осуществляется операционной системой, точнее компонентом, который называют подсистема ввода/вывода - диспетчером или супервизором ввода/вывода. Этот компонент выполняет следующие задачи:

1. Получает запросы на ввод/вывод прикладных задач и программных модулей самой системы. Проверяет их корректность и выдает соответствующее диагностическое сообщение.
2. Определяет очередность предоставления устройств ввода/вывода задачам затребовавшим их.
3. Инициирует ввод/вывод (передает управление соответствующим драйверам) и в случае выполнения ввода/вывода с использованием прерывания передает управление диспетчеру задач. Чтобы он передал его первой задаче, стоящей в очереди на выполнение.
4. Идентифицирует сигналы прерывания от устройств ввода/вывода и передает управление соответствующей программе обработки прерывания.
5. Передает сообщения об ошибках, случившихся в процессе ввода/вывода.
6. Посылает сообщение о завершении операции ввода /вывода, запросившему эту операцию процессу и снимает его с состояния ожидания ввода/вывода, Если процесс ждал завершение операции.

Существует 2 основных режима ввода/вывода

1. режим обмена опросом готовности устройства ввода/вывода
2. режим обмена с прерыванием

Для организации ввода/вывода по 1 варианту процессор посылает устройству управления команду для устройства ввода/вывода выполнить некоторое действие. Устройство управления выполнит команду преобразования, её сигналы управления, которое оно передает устройству ввода/вывода. Поскольку быстродействие устройства ввода/вывода меньше на несколько порядков устройства быстродействия процессора, то драйвер управляющий обилием данных с внешних устройств вынужден в цикле опрашивать готовность устройств. При этом нерационально используется время процессора. Выгоднее после команды ввода/вывода перейти на выполнение другой команды, а появление сигнала готовности трактовать как запрос на прерывание. Драйверы работающие в режиме прерывания представляют собой сложный комплекс программных модулей и имеют несколько секций:

- секция запуска
- секция продолжения
- секция завершения

Секция запуска запускается для включения устройств ввода/вывода либо для инициализации очередной операции ввода/вывода .

Секция продолжения осуществляет основную работу по передаче данных

Секция завершения выключает устройства ввода/вывода либо просто завершает операцию.

Управление операциями ввода/вывода в режиме прерывания требует более сложных программ чем те, что работают в режиме опроса готовности.

Так, в операционных системах Windows 95,98 и Windows NT драйвер печати через параллельный код работает не в режиме прерывания, а в режиме опроса готовности, что приводит к 100% загрузке процессора на все время печати. Для организации и выполнения многие параллельно выполняющиеся задачи устройств ввода/вывода вводится понятие виртуального устройства, повышающего

эффективность вычислительных систем. Примером служит spooling, то есть имитация работы с устройством в режиме on-line.Главная задача spoolinga – создать единицу параллельно выполняемого устройства ввода/вывода с последовательным доступом.

Например, каждому вычислительному процессу предоставляется не реальный , а виртуальный прибор и поток выводимых символов, сначала направляемых в специальный файл на магнитном диске. Он называется spool- файл, затем по окончании виртуальной печати содержимое spool – файла выводится на принтер. Системный процесс который управляет spool – файлом называется spooler, spool-reader, spool-writer.

Основные системные таблицы ввода/вывода

Каждая операционная система имеет свои таблицы ввода/вывода для того чтобы управлять вводом/выводом через операционную систему (ядро) и выполнять при этом механизм прерывания операционной системы должна иметь по крайней мере 3 системные таблицы.

Первая таблица оборудования содержит информацию обо всех устройствах ввода/вывода подключенных к системе.(Unit Control Block). UCB содержит следующую информацию об устройстве:

1. тип устройства, его модель
2. подключение устройства (через какой интерфейс, к какому разъёму, какие порты и линии прерывания используются)
3. указание на драйвер (адрес секции запуска)
4. информацию о буфере памяти
5. состояние устройств
6. указатель на дескриптор задачи использующий устройство в данный момент

2 таблица описания виртуальных (логических) устройств.

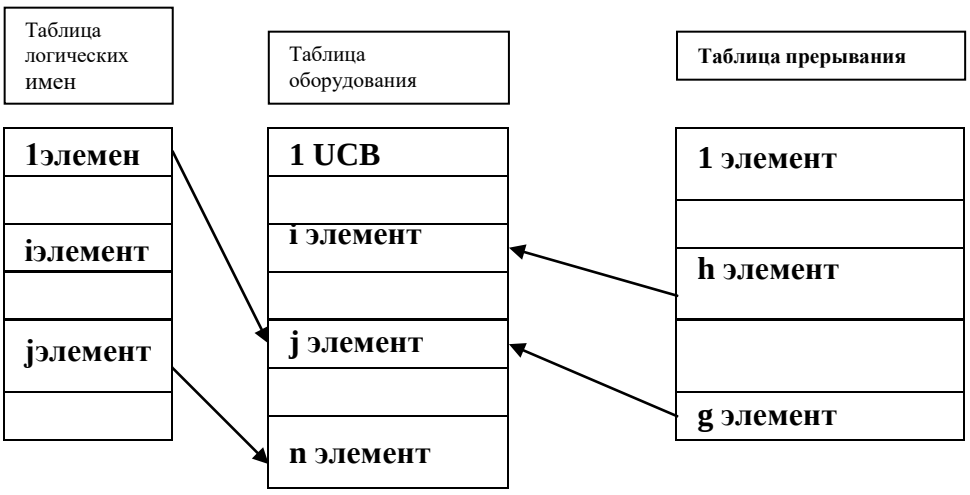
Её назначение – устранение связи между виртуальными устройствами, описанными в таблице 1. 2 таблица позволяет ядру операционной системы перенаправлять запрос на ввод/вывод из приложения на те программные модули и страницы данных, которые (или адреса которых) хранятся в соответствующем элементе 1 таблицы.

В многопользовательских системах таких таблиц нет : одна общая и по одной на каждого пользователя.

3 таблица прерывания, которая для всех сигналов прерывания указывает тот или иной элемент 1 таблицы, который описывает устройство выполняющее эту линию прерывания. Эта таблица может в явном виде не присутствовать поскольку может из основной таблицы прерываний попасть на драйвер, именующий связи с элементом UCB.

Наличие связи между таблицами 1 и 3 .

Взаимосвязь изображают так:

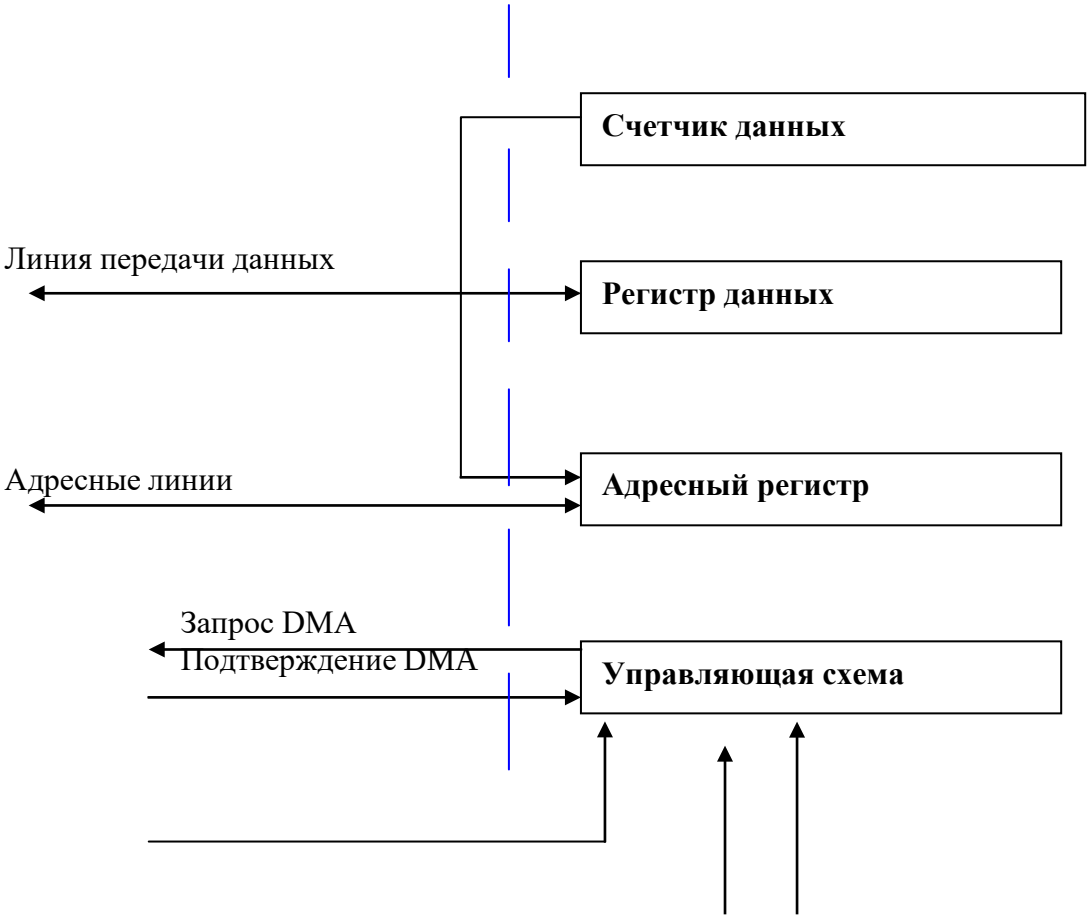


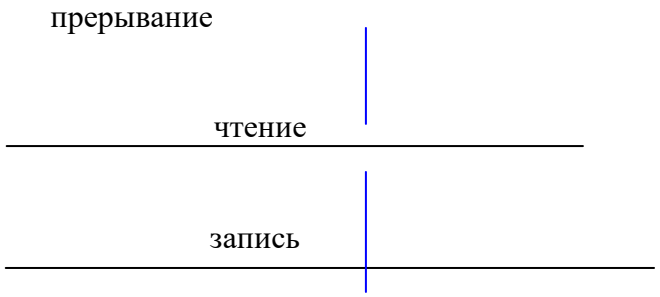
Управление вводом/выводом состоит в выполнении следующих действий :
Запрос на операции ввода/вывода от выполняющейся программы поступает в ядро операционной системы . Оно проверяет вызов на правильность и при отсутствии ошибок пересылает его в подсистему ввода/вывода.
Процесс ввода/вывода по логическому имени с помощью таблицы логических имен находит соответствующий элемент UCBтаблицы оборудования. Если устройство занято ,то дескриптор задачи от которой поступил запрос на ввод/вывод помещается в очередь ожидающих устройств. Если устройство свободно, то подсистема определяет по UCB тип устройства и передает управление соответствующему драйверу на секцию запуска. Драйвер инициализирует оптимизацию управления и возвращает управление диспетчеру задач , чтобы он поставил на процессор готовую к исполнению задачу , Когда устройство ввода/вывода обрабатывает команду оно поставляет запрос на прерывание , по которому через таблицу прерывания управление передается на секцию продолжения , получив новую команду устройство начинает её обрабатывать, а управление процессом передается диспетчеру задач и процессор продолжает полезную работу.

Таким образом осуществляется параллельное выполнение задач на фоне которого происходит управление операциями ввода/вывода.

Прямой доступ к памяти.

Третьей формой осуществления доступа к памяти является прямой доступ к памяти. В большинстве вычислительных систем он является основным способом передачи данных.
Он позволяет перемещать блоки данных из памяти или в неё без использования процессора. Модуль прямого доступа к памяти DMA способен дублировать функции процессора в частности отдавать управление системой для передачи данных по системной шине. Если процессор тоже нуждается в системной шине, то модуль DMA вынуждает процессор приостановить свою работу. Эта операция именуется захватом цикла, так как модуль DMA выполняет захват цикла шины. Работа модуля DMA представляется так:





Тема 7. Работа в операционных системах и средах

В момент, когда процессору необходимо произвести чтение или запись модуля данных он выполняет запрос модуля DMA передавая ему следующую информацию:

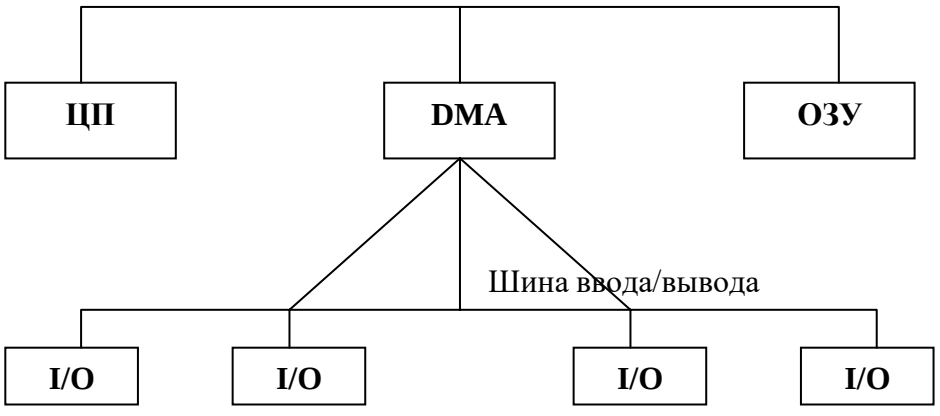
1) Какая операция чтения или записи запрашивается.
2) Адрес используемого УВВ (устройство ввода/вывода)
3) Начальный адрес считываемый или записываемый в области памяти хранящийся в адресном регистре DMA.
4) Число слов, которое необходимо прочесть или записать. Эта величина хранится в регистре счетчика данных в модуле DMA.

После этого процессор продолжает работу с другой программой, а модуль DMA минуя процессор передает весь блок данных непосредственно в память или считывает данные из нее. После передачи данных модуль DMA посылает процессору сигнал прерываний.

Таким образом, процессор включен в этот процесс лишь в начале и в конце передачи данных.

Конфигурирование прямого доступа к памяти может быть выполнено несколькими способами. Наиболее эффективны следующие:

СИСТЕМНАЯ ШИНА



Обмен данными между DMA и УВВ происходит вне системной шины.

ФАЙЛОВЫЕ СИСТЕМЫ

Под файлом понимается набор данных, организованных в виде совокупности записей одинаковой структуры. Для управления этими данными создаются системы управления файлами.

Возможность иметь дело с логическим уровнем структурных данных и операциями по их обработке предоставляет файловая система (ФС).

ФС - это набор спецификаций и соответствующее им программное обеспечение, которое отвечает за создание, уничтожение, организацию, чтение, запись, модификацию и перемещение файловой информации, а также за управление к доступам файлов. ФС определяет способ организации данных на диске или другом носителе информации. Пример: ФС FAT: реализации которой присутствуют в абсолютном большинстве операционных систем (ОП) для ПК. Все современные ОС имеют соответствующие системы управления файлами (СУФ). СУФ является основной подсистемой в ОС. Через СУФ осуществляется централизованное распределение дискового пространства и управления данными. Через СУФ пользователю предоставляются следующие возможности:

1) Создание, удаление, переименование и др. операции с именованными наборами данных из своих программ или посредством специальных управляющих программ реализующих функции интерфейса пользователя с его данными.
2) Работа с недисковыми периферийными устройствами как с файлами.
3) Обмен данными между файлами, между устройствами, между файлом и устройством и наоборот.
4) Работа с файлами с помощью обращений программных модулей СУФ (Application Program Interface). Часть функций API ориентирована на работу с файлами.
5) Защита файлов от несанкционированного доступа.

В ОС может быть несколько СУФ, чтобы иметь возможность работать с несколькими ФС. Основное назначение ФС и соответствующей ей СУФ - организация удобного доступа к данным организована как файл, т.е. вместо низкоуровневого доступа к данным с указанием конкретных физических адресов нужной записи используется логический доступ с указанием имени файла и записи в нем.

Термин ФС определяет принципы доступа к данным организованных в файле. А термин СУФ относится к конкретной реализации ФС, т.е. это комплекс программных модулей обеспечивающих работу с файлами конкретной ОС. В качестве примера можно привести ФС FAT (Fill Allocation Table), которая имеет множество реализаций как СУФ. Название FAT используется и по отношению к СУФ MS-DOS. В реализацию СУФ для OS/2, использующей основные принципы системы FAT наз. Super-FAT. Ее основные отличия - поддержка расширенных атрибутов для каждого файла.
Windows95/98 - VFAT.

СТРУКТУРА МАГНИТНОГО ДИСКА

Для того, чтобы с магнитного диска (МД) загрузить ОС и затем с ее помощью организовать работу СУФ были приняты специальные системные соглашения о структуре диска.

В самом первом секторе находятся данные о логической организации диска и программа с помощью которой находятся и загружаются программы загрузки ОС. Информация на МД размещается и передается блоками. Каждый блок называется сектором. Сектора расположены на концентрической дорожке поверхности диска. Каждая дорожка называется треком и образуется превращением МД под зафиксированным некоторым предопределяющем положением головки чтения/записи. Группы дорожек одного радиуса расположенные на поверхностях МД образуют цилиндр. Жесткие диски имеют по несколько десятков тысяч цилиндров, а на поверхности дискеты их 80. Каждый сектор состоит из поля данных и поля служебной информации, ограничивающей и идентифицирующей его. Размер сектора устанавливается контроллером или драйвером. В большинстве ОС размер сектора 512 байт. Физический адрес сектора на диске определяется с помощью трех координат [c-h-s], где c - номер цилиндра, h - номер рабочей поверхности диска, а s -номер сектора на дорожке. Номера цилиндра и поверхности диска номеруются с 0, а номер сектора с 1. Обмен информацией с дисками физически осуществляется только с секторами. Жесткий диск может быть разбит на несколько разделов (partition), которые могут использоваться либо одной ОС, либо различными. Главное, что в каждом разделе может быть организована своя ФС. Разделы дисков могут быть двух типов (primary, extended - первичный и расширенный). Максимальное число первичных разделов 4, минимальное - 1. Если их несколько, то только один из них может быть активным. Именно загрузчику расположенному в активном разделе

передается управление при включенном компьютере и загрузке ОС. Остальные первичные разделы в этом случае считаются скрытыми (hidden). Согласно спецификациям на диске может быть только один расширенный раздел, который в свою очередь может быть поделен на большое количество подразделов - логических дисков. С активного первичного раздела загружается программа загрузки ОС, называемая менеджером загрузки. Ее назначение - загрузить программу загрузки ОС из какого-нибудь другого раздела и с ее помощью загрузить саму ОС. Поскольку до загрузки ОС СУФ работать не может, то для указания загрузчика используются абсолютные адреса в формате [c-h-s]. По физическому адресу [0-0-1] на винчестере располагается главная загрузочная запись (Master Boot Record), содержащая внесистемный загрузчик (Non System Bootstrap), а также таблицу разделов. Эта запись равна ровно первому сектору, она размещается в памяти с адресом 0:7C00H, после чего управление передается коду программы, содержащемуся в первом секторе МД.

МД является основным средством загрузки диска, которое поддерживается BIOS. В MBR находится три важных элемента:

1) программа начальной загрузки
2) таблица описания разделов диска, располагается по смещению 0:1BE и занимает 64 байта
3) сигнатура MBR. Последние 2 байта MBR должны содержать число AA55h. По наличию этой сигнатуры BIOS проверяет, что первый блок был загружен успешно. Этот код выбран неслучайно. Его успешная проверка говорит о том, что все линии передачи данных могут передавать и нули и единицы.

Таблица разделов описывает размещение и характеристики разделов на жестком диске. Если она повреждена, то не только не будет загружена ОС, но и перестанут быть доступными все данные на диске.

СМЕЩЕНИЕ	РАЗМЕР	СОДЕРЖАНИЕ
0	446	
IBEh	16	Partition 1 Entry
1 CEh	16	Partition 2 Entry
1 DEh	16	Partition 3 Entry

Первым байтом в предельном разделе является Boot Indicator, 0- не активен, 128 – активен. Он определяет раздел, который является системным загрузочным. Далее следует байт № головки рабочей поверхности, с которой начинается раздел. Далее идут два байта, означающие № цилиндра, № загрузочного сектора, где располагается загрузчик ОС. Затем идет кодовый идентификатор длиной 1 байт (System Id), указывающий на установленную файловую систему.

Т.е. 00-пустой раздел
01-FAT12
06-FAT16
ОС-FAT32
83-LINUX NATIVE
85-LINUX EXTENTED

За байтом кода ОС располагается байт № рабочей поверхности конца раздела, за которым идут 2 байта: № сектора и № цилиндра последнего сектора данного раздела.

Формат записи, содержащей № сектора и цилиндра:

15 65	0
Биты № цилиндра	Биты № сектора
10 бит	6 бит

Вслед за MBR размечают сами разделы.

В DOS в первичном разделе может быть сформирован один логический диск, а в расширенном любое количество.

Расширенный раздел DOS содержит расширенную запись MBR, равную Secondary MBR, в состав которой входит таблица логического диска (Logical Disk Table). Эта таблица описывает размещение характеристики раздела, содержащей единственный логический диск, а также может определять следующую запись SMBR.

Утилиты, позволяющие разбить диск на разделы называются FDISK (Form disk), Disk editor, Partition Magic.

Файловая система FAT.

В FAT логическое пространство любого диска делится на 2 области: системную и область данных.

загр. сектор	резерв. сектор			корн. кат.	
BR	RecSecs	FAT1	FAT2	RDir	Каталоги и файлы

FAT является информационной структурой, в которой описывается состояние каждого участка области данных. Область даны разбивается на кластеры. Кластеры представляют один или несколько смежных секторов в области данных.

В таблице FAT кластеры, принадлежащие одному файлу связываются в цепочки. Для указания № кластера используется 16- битовой слово (65536).

Кластер – минимальная адресная единица дисковой памяти, выделенная файлу. Разбиение области данных на кластеры вместо секторов имеет смысл по причине:

Уменьшается размер самой FAT;

Уменьшается возможная фрагментация файлов;

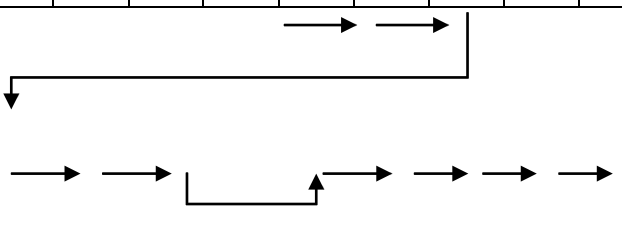
Ускоряется доступ к файлу, т.к. в несколько раз сокращается длина цепочек – фрагментов дискового пространства, выделенного файлу.

Большой размер файла ведет к неэффективному использованию дискового пространства, особенно при большом количестве малых файлов. Поэтому в файловых системах HDFS, NTFS, FAT32 размер кластера ограничивается (512 байт- 4 Кбайт). В FAT 32 проблема решается за счет того, что сама FAT может содержать 2^{28} кластеров.

Пример:

Имя файла		Время последнего обращения к файлу	Данные	№ начального кластера	Размер
MyFile.txt	Атрибуты	Time	Data	08	Size

	0	1	2	3	4	5	6	7	8	9	0A	0B	0C	0D	0E	0F
00	IP	FF	03	04	05	FF	00	00	09	0A	0B	15	00	00	00	00



01	00	00	00	00	00	16	17	19	F7	1A	1B	1C	1D	FF	00	00

- содержание FAT, по отношению к файлу MyFile.txt

Т.к. FAT интенсивно используется, то она помещается в ОЗУ, а на диске хранится в двух экземплярах. Используется только первый, если он оказывается поврежден, то происходит обращение ко второму экземпляру.

Утилиты ScanDisk при обнаружении несоответствия между копиями FAT предлагает восстановить главную по данным второй копии.

Файловая система VFAT, FAT32.

VFAT (32 разряда) предназначена для ОС Windows для выполнения файлового ввода- вывода в защищенном режиме. В эту систему добавлена поддержка длинных имен фалов наряду с формой (?????? . ???).

Эта файловая система поддерживается Win9x b WinNT 4.0.

Основными недостатками FAT и VFAT являются большие потери на кластеризацию при больших размерах логического диска и ограничениях на сам размер логического диска.

Т.е. необходимо разработать файловую систему с использованием идеи применения таблицы FAT. Файловая система типа FAT32 является самостоятельной 32- разрядной файловой системой. Главное ее отличие в более эффективном использовании дискового пространства.

Прежде всего FAT32 использует кластеры меньшего размера по сравнению с предыдущими версиями, который ограничивались 65535 кластерами на диске. Даже для дисков до 8 Гбайт FAT32 может использовать 4 Кбайт – кластеры, в результате экономится 10-15 % дискового пространства. FAT32 может перемещаться в корневой каталог и использовать резервную копию вместо стандартной. Корневой каталог FAT32 представлен в виде обычной цепочки кластеров, т.е. корневой каталог находится в любом месте диска, что снимает ограничения на его размер (512 бит).

Кроме повышения емкости FAT до 4 Тбайт. FAT32 изменяет структуры корневого каталога. Для хранения имен фалов используются дополнительные элементы каталога, в т.ч. и для корневых. Длинное имя может занимать до 256 символов и для его хранения используется 25- элементов каталога. Длина полной файловой спецификации , включая путь и имя файла ограничена 260-символами, т.е. рекомендуется ограничивать длину имени файла 75-80 символами для того, чтобы осталось место для пути.

Файловая система NTFS (New Technology File System).

Эта файловая система содержит ряд усовершенствований и изменений по сравнению с другими файловыми системами. Файлы хранятся в папках или каталогах. В NTFS повышена эффективность работы с дисками большого объема. Имеются средства для ограничения доступа, введены механизмы для повышения надежности файловой системы.

Основные особенности файловой системы NTFS следующие:

- способность восстановления данных. Файловая система восстанавливает при отказе системы и сбоев дисков. Это достигнуто по средствам использования механической транзакции, при котором осуществляется журналирование файловой операции
- безопасность. Файловая система поддерживает объектную модель безопасности и рассматривает все тома, каталоги, файлы как самостоятельные объекты. NTFS обеспечивает безопасность на уровне файлов, это означает, что право доступа к файлам зависит от учетной записи пользователя, и тех групп к которым он принадлежит.
- расширенная функциональность. NTFS проектировалась с учетом возможного расширения. В ней реализованы такие возможности, как эмуляция других операционных систем, параллельная обработка потоков данных и создание файловых атрибутов определенных пользователем.

- поддержка POSIX (Portable Operating System for computing environments). Международный стандарт машинно-независимого интерфейса вычислительной среды. В нем основное внимание уделяется взаимодействию прикладных программ с операционной системой. Написанная прикладная программа позволяет создавать программы легко переносимые из одной операционной системы в другую.
- эффективная поддержка больших дисков и файлов. Максимальный размер тома NTFS составляет 2^{64} байт = 1 Экзобайт = 16000 млр. Гб. Максимальный размер файла составляет 2^{32} кластера = 2^{64} байт. Размер кластера может меняться от 512 байт до 64 Кбайт. NTFS поддерживает длинные имена файлов, набор символов Unicode и имена 8.3. Количество файлов в корневом и не корневом каталоге не ограниченно.

Суть процедуры восстановления данных в NTFS заключается в регистрации. Каждая операция изменения файловой системы, т.е. операция по распределению дискового пространства обрабатывается как транзакция. Транзакция рассматривается как неделимая (атомарная) операция, во время которой не какие другие операции на изменения данными не разрешены.

Каждая транзакция осуществляется в 3 этапа:

- Система записывает в специальный журнальный файл то, что она собирается делать. Если запись в журнал была успешной, то система выполняет транзакцию.
- Если транзакция завершена нормально, то система отмечает в журнале этот факт.
- Если произошел сбой системы, то после загрузки запускается программа восстановления. Эта программа просматривает конец журнального файла. Если обнаружена импортирующая запись, то она игнорируется – сбой произошел во время записи в журнал. Если все записи помечены как успешно выполненные транзакции, то сбой произошел между транзакциями – ничего исправлять не надо. Если найдена запись, которая отмечает начинутую, но не выполненную транзакцию, то сбой произошел во время этой транзакции – но журнал содержит достаточно информации, чтобы восстановить состояние файловой системы до начала транзакции или же доделать ее до конца.

Структура тома с файловой системой NTFS.

Файловая система NTFS использует следующие единицы дискового хранения: сектор, кластер (размер кластера в секторах является степень 2), том.

Том – логический раздел диска, состоящий некоторого количества кластеров и используемый файловой системой для распределения дискового пространства. Том может занимать как весь диск, так и его часть или охватывать несколько дисков (RAID’s).

Размер кластера от 512 байт до 64 Кбайт. Стандартными считаются кластера 2 Кбайта или 4 Кбайта. Все дисковое пространство NTFS делится на две неравные части.

Загрузка сектора диска	Главная файловая таблица MFT	Системные файлы	Область файлов

Первые 12% диска – загрузочный сектор диска (размер до 16 физических секторов).

MFT (Master Fail Table) – специальный файл главной системы. Структура данных которого, позволяет определить место нахождения всех остальных файлов. Запись, каких либо данных в эту область невозможна. Далее идет область длиной в 1 Мбайт для системных файлов. MFT представляет собой централизованный каталог всех остальных файлов диска, в том числе и самого себя. MFT поделен га записи фиксированного размера в 1 Кбайт и каждая запись соответствует какому либо файлу.

Первые 16 файлов носят служебный характер и называются метафайлами, причем самый первый это MFT. Они хранятся в области системных файлов за MFT-зоной, так же как и копии первых 16 записей MFT. Метафайлы находятся в корневом каталоге тома, их имена начинаются с символа $.

Файл в томе с файловой системой NTFS идентифицируется файловой ссылкой (File Reference) которая представляется как 64 разрядное число. Файловая ссылка состоит из номера файла, который соответствует позиции его файловой записи и MFT и номера последовательности. Последний увеличивается всякий раз, когда данная позиция MFT используется повторно. Что позволяет файловой системе выполнить внутренние проверки целостности.

Каждый файл NTFS представлен с помощью атрибутов (поток) т.е. у него нет как таковых просто данных, а есть атрибуты. Один из атрибутов и есть данные файла. Таким образом, базовая сущность у файла только одна – номер в MFT, а все остальное атрибуты. Такой подход можно эффективно использовать. Например, файлу можно «прилепить» еще один атрибут или поток записав в него любые данные. В W2K, таким образом, записана информация об авторе содержание файлы. Атрибуты или потоки не видны стандартными средствами работы с файлами: наблюдаемый размер, размер основного потока, который содержит традиционные данные. Можно иметь файл нулевой длины при стирании которого, освободиться 1 Гбайт памяти. (На идей подмен потока основано написание вируса)

Таким образом, файл состоит из набора атрибутов (потока, данные, хранящиеся в файлах рассматриваются как атрибуты).

Атрибуты файлов в NTFS.

1. стандартная информация о файле. Традиционные атрибуты: только для чтения, скрытый, архивный, системный, время создания, число каталогов ссылающихся на файл.
2. список атрибутов, из которых состоит файл.
3. имя файла в символах Unicode. Файл может иметь несколько атрибутов имен, как в Unix системах.
4. дескриптор защиты. Структура данных защиты предохраняющая файл от несанкционированного доступа. Этот атрибут определяет, кто владелец файла и кто имеет доступ к нему.
5. данные. Соответственно данные файла – его содержимое. По умолчанию у файла есть один безымянный атрибут данных, и он может иметь дополнительно именованные данные.

Пункты 1-5 обязательны для каждого файла.

Имя файла в NTFS может содержать любые символы, включая национальный алфавит, т.к. они представлены в Unicode (16 битное представление, максимальная длина 255 символов).

Каталог NTFS представляет собой специальный файл, хранящий ссылки на другие файлы и каталоги, создавая иерархическое строение данных на диске.

Файл каталога поделен на записи, каждая содержит имя файла, базовый атрибут, ссылку на элемент MFT который представляет полную информацию об элементе каталога.

Корневой каталог не чем не отличается от обычного, кроме специальной ссылки на него из начала MFT. Внутренняя структура каталога представляет собой бинарное дерево, т.е. в каталоге имена файлов располагаются таким образом, что бы можно было понять, к какой группе относительно данного элемента находится искомое имя, выше или ниже. Если поиск начинается со среднего элемента, то каждое обращение сужает зону поиска в 2 раза. Если файлы отсортированы по алфавиту, то при поиске производится сравнение начальных букв.