Системное и прикладное программное обеспечение

Малышенко Владислав Викторович

Программное обеспечение

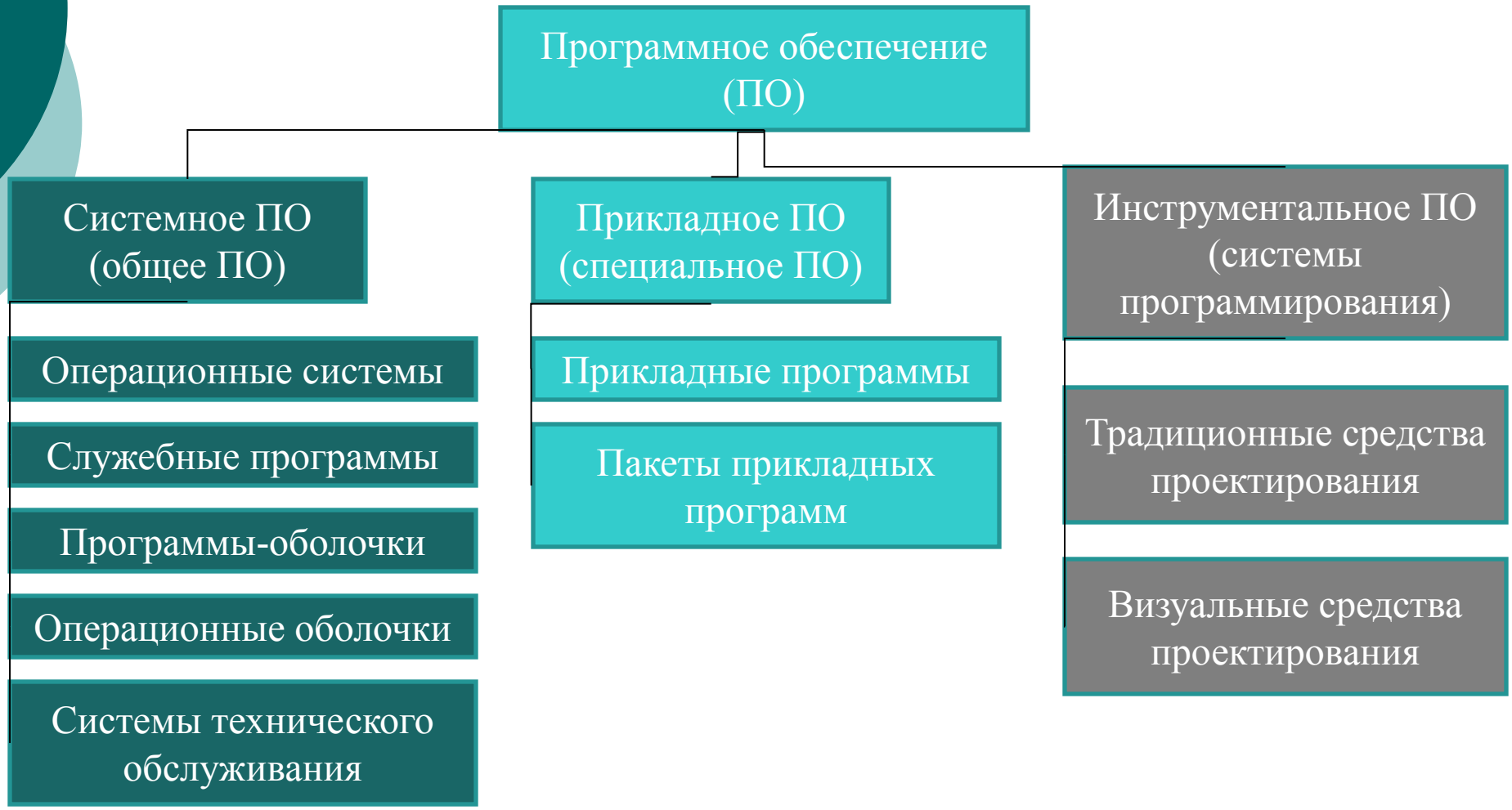
- Совокупность программ, предназначенная для решения задач на ПК, называется **программным обеспечением**.
- Состав программного обеспечения ПК называют **программной конфигурацией**.
- Программное обеспечение, можно условно разделить на три категории:
  - системное ПО;
  - прикладное ПО;
  - инструментальное ПО (системы программирования).

Программное обеспечение

Программное обеспечение, можно условно разделить на три категории:

- **системное ПО** (программы общего пользования), выполняющие различные вспомогательные функции, например создание копий используемой информации, выдачу справочной информации о компьютере, проверку работоспособности устройств компьютера и т.д.
- **прикладное ПО**, обеспечивающее выполнение необходимых работ на ПК: редактирование текстовых документов, создание рисунков или картинок, обработка информационных массивов и т. д.
- **инструментальное ПО** (системы программирования), обеспечивающее разработку новых программ для компьютера на языке программирования.

Программное обеспечение



Содержание курса

- Системное программное обеспечение
  - Введение
  - Обзор операционных систем
  - Процессы и потоки
  - Управление памятью
  - Ввод-Вывод
  - Файловые системы
  - Архитектуры операционных систем
- Прикладное программное обеспечение
- Инструментальное программное обеспечение
  - Технология программирования
  - Проектирование программного обеспечения

Литература

1. **Таненбаум Э.С.** Современные операционные системы. 3-е изд. СПб.:Питер, 2010. - 1120с.
2. **Олифер В.Г., Олифер Н.А.** Сетевые операционные системы. 2-е изд. СПб.:Питер, 2009. - 672с.
3. **Иванова Г.С.** Технология программирования: Учебник для ВУЗов. изд.:КноРус, 2011 – 336с.
4. **Соммервилл И.** Инженерия программного обеспечения, 6-е издание, 2002.
5. Таненбаум Э.С., Вудхалл А.С. Операционные системы: Разработка и реализации
6. Калянов Г.Н. CASE структурный и системный анализ (автоматизация и применение)/М.: Изд-во «ЛОРИ», 1996.
7. Роббинс Дж. Отладка приложений: Пер. с англ. – СПб,: БХВ-Петербург, 2001.

Введение.
Операционные системы.

Системное и прикладное программное обеспечение

Малышенко Владислав Викторович

Введение

Современная компьютерная система состоит из:

- одного или нескольких процессов,
- оперативной памяти,
- дисков,
- клавиатуры,
- монитора,
- принтеров,
- сетевого интерфейса

и других устройств, то есть является **сложной комплексной системой**.

Написание программ, которые следят за всеми компонентами, корректно используют их и при этом работают оптимально, представляет собой крайне трудную задачу.

По этой причине компьютеры оснащаются специальным уровнем программного обеспечения, называемым **операционной системой**.

Операционная система отвечает за управление всеми перечисленными устройствами и обеспечивает пользователя имеющими простой, доступный интерфейс программами для работы с аппаратурой.

Компьютерная система

Физические устройства

Самый нижний уровень содержит физические устройства, состоящие из

- интегральных микросхем,
- проводников,
- источников питания,
- электронно-лучевых трубок
- и т. п.

Микроархитектура

На **микроархитектурном уровне** физические устройства рассматриваются с точки зрения функциональных единиц.

На этом уровне находятся внутренние регистры центрального процессора (CPU) и арифметико-логическое устройство.

На каждом такте процессора из регистра выбирается один или два операнда, которые обрабатываются в арифметико-логическом устройстве. Результат сохраняется в одном или нескольких регистрах. В некоторых машинах операции над данными контролируются программными, которые называются **микропрограммами**.

Машинный язык

Определенная система команд передается по маршруту передачи данных, такие команды могут использовать регистры. Команды формируют уровень **ISA**, часто называемый **машинным языком**.

Обычно машинный язык содержит от 50 до 300 команд, служащих преимущественно для перемещения данных по компьютеру, выполнения арифметических операций и сравнения величин.

Управление устройствами на этом уровне осуществляется с помощью загрузки определенных величин в специальные **регистры устройств**. Например, диску можно дать команду чтения, записав в его регистры адрес места на диске, адрес в основной памяти, число байтов для чтения и направление действия.

Операционная система

Операционная система предназначена для того, чтобы скрыть от пользователя сложности, связанные с управление устройствами.

Она состоит из уровня программного обеспечения, который частично избавляет от необходимости общения с аппаратурой напрямую, вместо этого предоставляя программисту более удобную систему команд.

Например: чтения блока из файла.

Операционная система (2)

Над операционной системой находятся интерпретатор команд (оболочка), системы окон, компиляторы, редакторы и т. д. Данные программы не являются частью операционной системы.

Под операционной системой обычно понимается то программное обеспечение, которое запускается **в режиме ядра** или, как его еще называют, режиме **супервизора**. Она защищена от вмешательства пользователя с помощью аппаратных средств.

Компиляторы и редакторы запускаются в **пользовательском режиме**. Если пользователю не нравится какой-либо компилятор, он при желании может написать свой собственный, но он не может написать собственный обработчик прерываний системных часов.

Системные программы

Существуют системы, в которых понятие **ядра** размыто:

- встроенные системы;
- интерпретируемые системы (Java-машина).

В традиционных компьютерах операционная система представляет собой набор программ, запускающихся в режиме ядра.

Во многих системах есть программы, которые работают в пользовательском режиме, но выполняют специализированные функции.

Например:

- смена паролей.

Прикладные программы

Над системными программами расположены прикладные программы.

Обычно они покупаются или пишутся пользователем для решения собственных проблем:

- обработки текста;
- электронных таблиц;
- технических расчетов;
- сохранения информации в базе данных.

Функции операционной системой

- расширение возможностей машины;
- управление ее ресурсами.

Операционная система как расширенная машина

Архитектура (система команд, организация памяти, ввод-вывод данных и структура шин) большинства компьютеров на уровне машинного языка примитивна и неудобна для работы с программами, особенно в процессе ввода-вывода данных.

Например:

Ввод-вывод данных с гибкого диска через совместимые микросхемы контроллера NEC PD765. Контроллер PD765 имеет 16 команд: чтение и запись данных, перемещение головки диска и форматирование дорожек, а также для инициализация, распознавание, установка в исходное положение и калибровка контроллера и приводов.

Операционная система как расширенная машина (2)

Программа, скрывающая истину об аппаратном обеспечении, является операционной системой.

Операционная система не только устраняет необходимость работы непосредственно с дисками и предоставляет простой интерфейс, но и скрывает множество неприятной работы с прерываниями, счетчиками времени, организацией памяти и другими элементами низкого уровня.

В любом случае абстракция, предлагаемая операционной системой, намного проще и удобнее в обращении, чем то, что может предложить нам непосредственно основное оборудование.

Операционная система как расширенная машина (3)

С точки зрения пользователя операционная система выполняет функцию расширенной машины или виртуальной машины, в которой проще программировать и легче работать, чем непосредственно с аппаратным обеспечением, составляющим реальный компьютер.

Операционная система предоставляет нам ряд возможностей, которые могут использовать программы с помощью специальных команд, называемых **системными вызовами.**

Операционная система как менеджер ресурсов

Операционная система как механизм, присутствующий в устройстве компьютера, для управления всеми частями этой сложнейшей машины.

Современные компьютеры состоят из процессоров, памяти, датчиков времени, дисков, мыши, сетевого интерфейса, принтеров и огромного количества других устройств.

В соответствии с этим подходом работа операционной системы заключается в обеспечении организованного и контролируемого распределения процессоров, памяти и устройств ввода-вывода между различными программами, состязающимися за право их использовать.

Операционная система как менеджер ресурсов. Пример

На одном компьютере оказались работающими три программы и все они одновременно попытались бы напечатать свои выходные данные на одном и том же принтере.

Возможно, первые несколько строк на листе появились бы от первой программы, следующие несколько - из второй программы и т. д. В результате получилась бы полная неразбериха.

Операционная система наводит порядок в подобных ситуациях, буферизируя на диске все данные, предназначенные для печати.

В процессе работы программы операционная система сохраняет ее выходные данные на диске во временном файле.

Операционная система как менеджер ресурсов (2)

Когда компьютером (или сетью) пользуются несколько пользователей, необходимость в управлении ресурсами и их защите сильно возрастает, поскольку пользователи могут обращаться к ним в абсолютно непредсказуемом порядке. К тому же часто приходится распределять между пользователями не только оборудование, но и информацию.

С этой точки зрения **основная задача** операционной системы заключается в **отслеживании** того, **кто** и **какой ресурс** использует, в обработке запросов на ресурсы, в подсчете коэффициента загрузки и разрешении проблем конфликтующих запросов от различных программ и пользователей.

Операционная система как менеджер ресурсов (3)

Управление ресурсами включает в себя их распределение двумя способами:

- во времени;
- в пространстве.

Когда ресурс распределяется во времени, различные пользователи и программы используют его по очереди. Сначала один из них получает доступ к использованию ресурса, потом другой и т. д.

Например: несколько программ хотят обратиться к центральному процессору.

Операционная система как менеджер ресурсов (4)

Определение того, как долго ресурс будет использоваться во времени, кто будет следующим и на какое время ему предоставляется ресурс - это **задача операционной системы**.

Пример: временного мультиплексирования - распределение заданий, посылаемых для печати на принтер. Когда задания выстраиваются в очередь для печати на одном принтере, операционной системе каждый раз нужно принимать решение о том, которое из них будет печататься следующим.

Операционная система как менеджер ресурсов (4)

Пространственное мультиплексирование. Вместо поочередной работы каждый клиент получает часть ресурса.

Например: Обычно оперативная память разделяется между несколькими работающими программами, так что все они одновременно могут постоянно находиться в памяти. Для справедливого распределения, защиты памяти и т. д., и для разрешения спорных вопросов существует операционная система.

Распределение дискового пространства и отслеживание того, кто какие блоки диска использует, является типичной задачей управления ресурсами, которую также выполняет операционная система.

История операционных систем

История развития операционных систем насчитывает уже много лет. Так как операционные системы появились и развивались в процессе конструирования компьютеров, то эти события исторически тесно связаны. Поэтому чтобы представить, как выглядели операционные системы, мы обсудим следующие друг за другом поколения компьютеров.

Первый настоящий цифровой компьютер был изобретен английским математиком Чарльзом Бэббиджем (Charles Babbage, 1792-1871).

История операционных систем

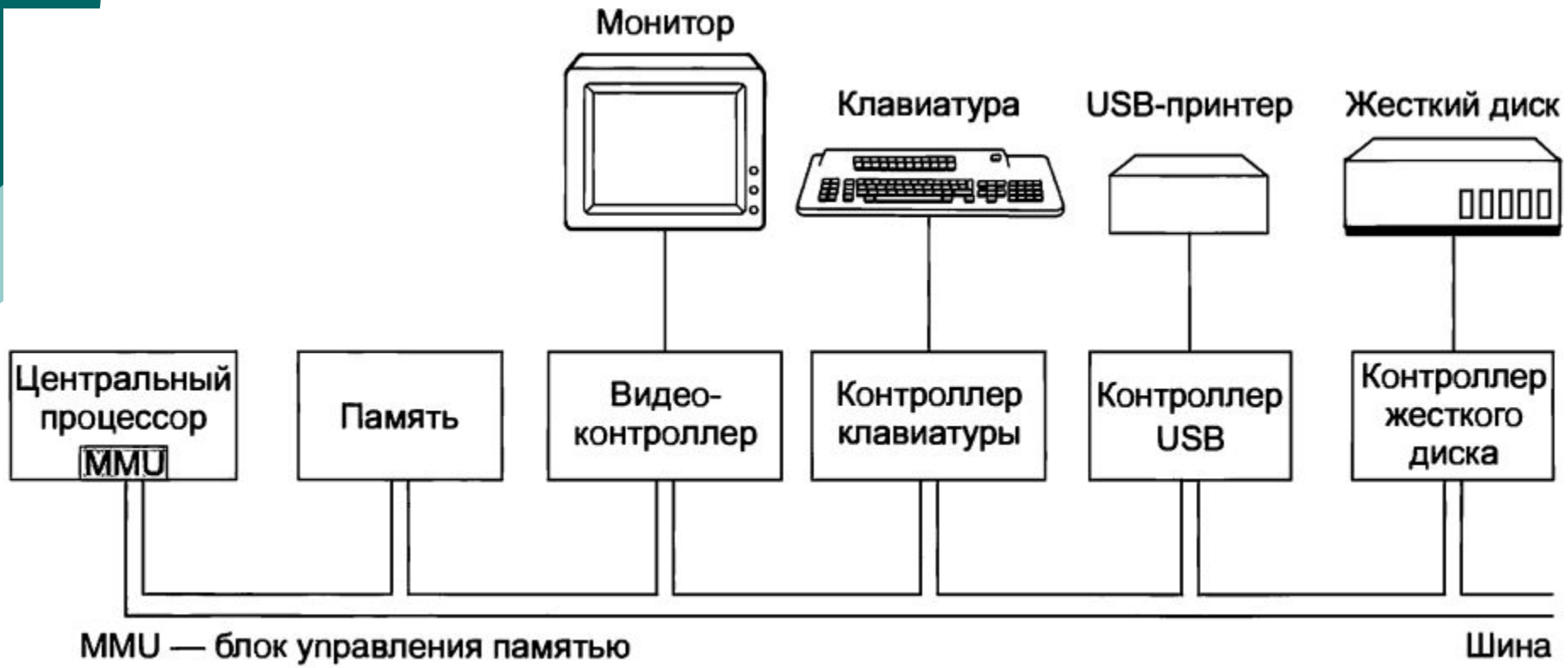
- Первое поколение (1945-55):
  - электронные лампы и коммутационные панели
- Второе поколение (1955-65):
  - транзисторы и системы пакетной обработки
- Третье поколение (1965-1980):
  - интегральные схемы и многозадачность
- Четвертое поколение (с 1980 года по наши дни):
  - персональные компьютеры
- Онтогенез повторяет филогенез

Обзор аппаратного обеспечения компьютера

Операционная система тесно связана с оборудованием компьютера, на котором она должна работать. Аппаратное обеспечение влияет на набор команд операционной системы и управление его ресурсами. Поэтому необходим определенный объем знаний о компьютере, в каком виде оборудование предстает перед программистом.

Концептуально простой персональный компьютер можно представить в виде абстрактной модели. Центральный процессор, память и устройства ввода-вывода соединены системной шиной, по которой они обмениваются друг с другом информацией.

Обзор аппаратного обеспечения компьютера



Процессоры

«Мозгом» компьютера является **центральный процессор** (CPU - Central Processing Unit).

Цикл работы центрального процессора выглядит так:

- он читает первую команду из памяти,
- декодирует ее для определения ее типа и операндов,
- выполняет команду,
- затем считывает, декодирует и выполняет последующие команды.

Для каждого центрального процессора существует набор команд, который он в состоянии выполнить.

Например: процессор Pentium не может обработать программы, написанные для SPARC, и наоборот.

Процессоры (1.5)

Доступ к памяти для получения команд или наборов данных занимает намного больше времени, чем выполнение этих команд, все центральные процессоры содержат внутренние регистры для хранения ключевых переменных и временных результатов.

Поэтому набор инструкций обычно содержит команды для загрузки слова из памяти в регистр и сохранения слова из регистра в памяти.

Другие команды объединяют два операнда из регистров, памяти или и того и другого и получают результат. Например, складывают два слова и сохраняют результат в регистре или памяти.

Процессоры (2)

Операционная система должна знать все обо всех регистрах. При временном мультиплексировании центрального процессора операционная система часто останавливает работающую программу для запуска (или перезапуска) другой. Каждый раз при таком прерывании операционная система должна сохранять все регистры процессора.

В целях улучшения характеристик центральных процессоров их разработчики давно отказались от простой модели, в которой за один такт может быть считана, декодирована и выполнена только одна команда.

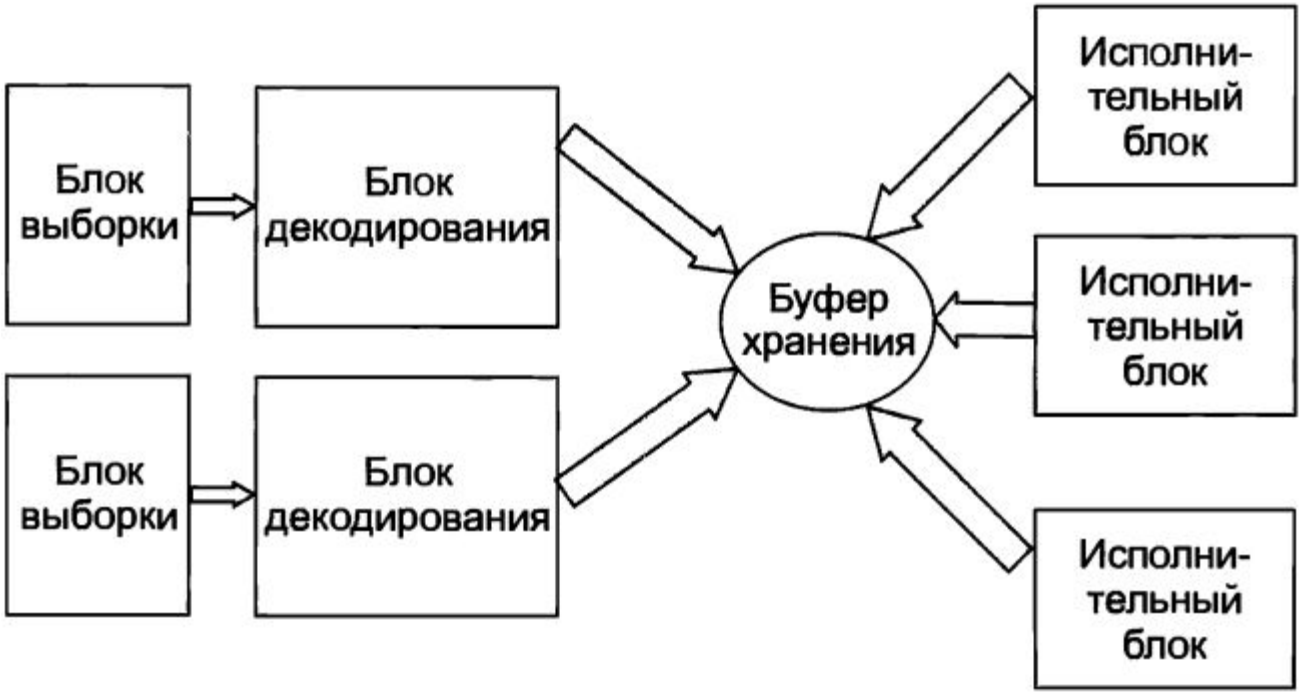
Многие современные CPU обладают возможностями выполнения нескольких команд одновременно.

Процессоры (3)

- Конвейер с тремя стадиями
- Суперскалярный процессор

Процессоры (3)

- Конвейер с тремя стадиями
- Суперскалярный процессор



Многопоточные и многоядерные микропроцессоры

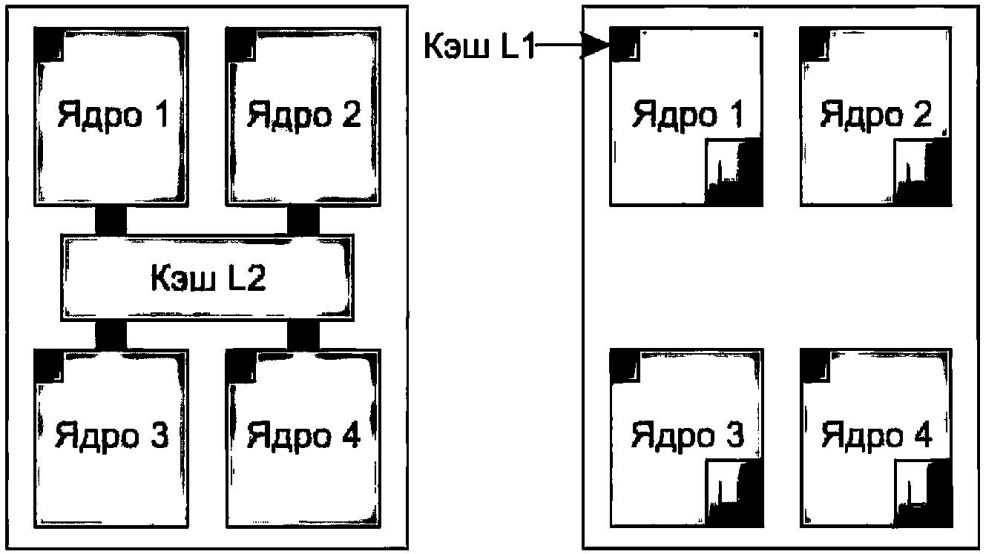
- Закон Мура гласит, что количество транзисторов на одном кристалле удваивается каждые 18 месяцев.
- Высокая плотность размещения транзисторов ведет к проблеме:
  - как распорядиться их возросшим количеством? Использование суперскалярной архитектуры, имеющей множество функциональных блоков. Но с ростом числа транзисторов открываются более широкие возможности.
  - Размещение на кристалле центрального процессора более объемной кэш-памяти. Однако это порог, за которым дальнейшее увеличение объема кэш-памяти только уменьшает отдачу от этого решения.

Многопоточные и многоядерные микропроцессоры

- Следующим очевидным шагом является дублирование не только функциональных блоков, но и части управляющей логики.
  - Реализация дублированных управляющих блоков в процессорах типа Pentium 4 называются **многопоточностью**, или **гиперпоточностью** (hyperthreading по версии Intel). В первом приближении эта технология позволяет процессору сохранять состояние двух различных потоков и осуществлять переключения между ними за наносекунды.
- Многопоточность оказывает влияние на операционную систему, поскольку каждый поток предстает перед ней как отдельный центральный процессор.
  - Представим себе систему с двумя реальными процессорами, у каждого из которых организовано по два потока. Операционной системе будут видны четыре процессора.

Многопоточные и многоядерные микропроцессоры

- Существуют процессоры, содержащие на одном кристалле два, четыре и более полноценных процессоров, или **ядер**.
  - Четырехъядерные процессоры, фактически имеют в своем составе четыре мини-чипа, каждый из которых представляет собой независимым процессор.
- Несомненно, для использования такого многоядерного процессора потребуется многопроцессорная операционная система.



Память

Второй основной составляющей любого компьютера является память. В идеале память должна быть максимально быстрой, достаточно большой и чрезвычайно дешевой. На данный момент не существует технологий, удовлетворяющих всем этим требованиям, поэтому используется другой подход.

Память (2)

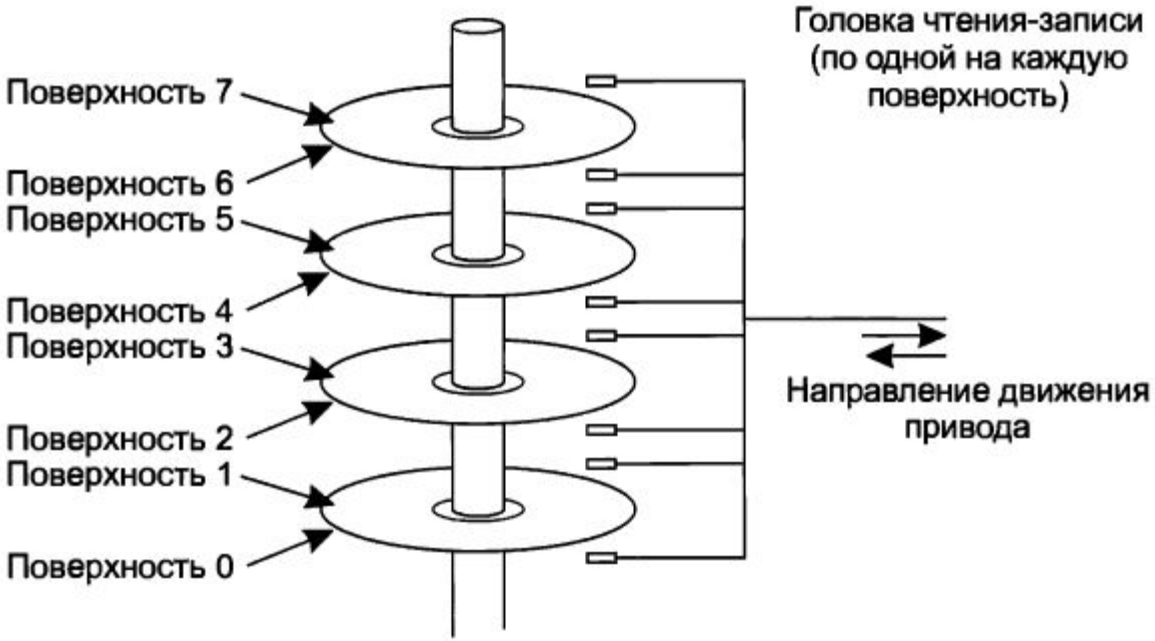
Верхний слой состоит из внутренних регистров центрального процессора. Они сделаны из того же материала, что и процессор, и так же быстры, как и сам процессор. Поэтому при доступе к ним обычно не возникает задержек.

В следующем слое находится кэш-память, в основном контролируемая оборудованием. Оперативная память разделена на кэш-строки, обычно по 64 байт.

Наиболее часто используемые строки кэша хранятся в высокоскоростной кэш-памяти, расположенной внутри центрального процессора или очень близко к нему. Кэш-память ограничена в размере, что обусловлено ее высокой стоимостью.

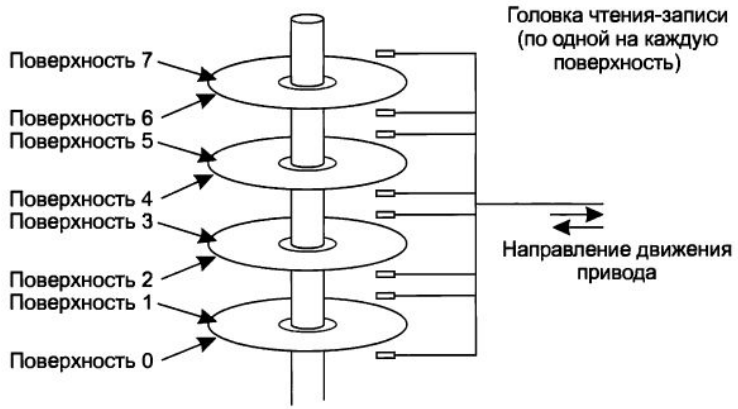
Жесткие диски

- Дисковый накопитель в пересчете на бит информации на два порядка дешевле, чем ОЗУ, а его емкость зачастую на два порядка выше.
- Единственная проблема состоит в том, что время произвольного доступа к данным примерно на три порядка медленнее. Причина низкой скорости доступа к данным заключается в том, что диск является механическим устройством.



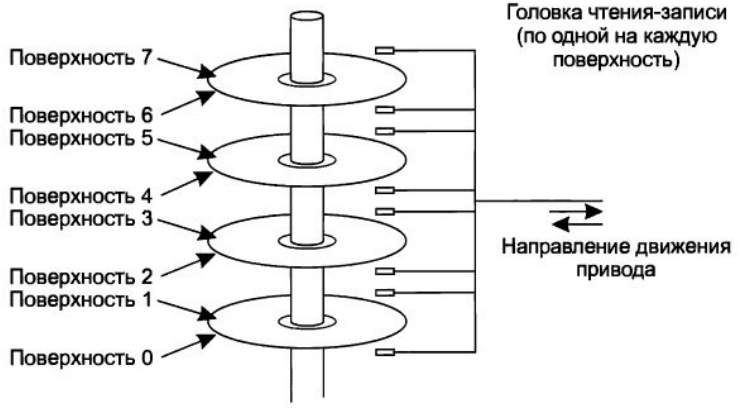
Жесткие диски

- Жесткий диск состоит из одной или нескольких металлических пластин, вращающихся со скоростью 5400, 7200 или 10 800 оборотов в минуту. Механический привод поворачивается на определенный угол над пластинами.
- Информация записывается на диск в виде последовательности концентрических окружностей. В каждой заданной позиции привода каждая из головок может считывать кольцеобразный участок, называемый дорожкой. Из совокупности всех дорожек в заданной позиции привода составляется цилиндр.



Жесткие диски

- Каждая дорожка поделена на определенное количество секторов, обычно по 512 байт на сектор. На современных дисках внешние цилиндры содержат больше секторов, чем внутренние.
- Перемещение привода с одного цилиндра на следующий цилиндр занимает около 1 мс.
- Перемещение к произвольно выбранному цилиндру обычно занимает от 5 до 10 мс, в зависимости от конкретного накопителя.
- После попадания требуемого сектора под головку производится операция чтения или записи со скоростью от 50 Мбайт/с (для низкоскоростных дисков) до 160 Мбайт/с (для высокоскоростных).



Ленты

- На последнем уровне иерархии памяти представлена магнитная лента.
- Этот носитель информации часто используется для создания резервных копий дискового накопителя и для хранения очень больших наборов данных.
- Большим плюсом магнитной ленты является ее чрезвычайная дешевизна в пересчете на бит хранимой информации, а также возможность смены носителей, что приобретает особую важность для лент, хранящих резервные копии, которые могут храниться в каком-нибудь другом месте во избежание повреждений при пожарах, наводнениях, землетрясениях и других стихийных бедствиях.
- Оптические диски, флеш-память и др.

Устройство ввода/вывода

Устройства ввода-вывода обычно состоят из двух частей: **контроллера** и **самого устройства**.

Контроллер - это микросхема физически управляющая устройством. Он принимает команды операционной системы, например указание прочитать данные с устройства, и выполняет их.

Устройства имеют достаточно простые интерфейсы, во-первых, потому что их возможности весьма невелики и, во-вторых, потому что нужно привести их к единому стандарту. Единый стандарт необходим, чтобы любой IDE-контроллер диска мог управлять любым IDE-диском.

Устройство ввода/вывода (2)

Так как все типы контроллеров отличаются друг от друга, для управления ими требуется различное программное обеспечение.

Программа, которая общается с контроллером, отдает ему команды и получает ответы, называется **драйвером устройства**. Каждый производитель контроллеров должен поставлять драйверы для поддерживаемых им операционных систем.

Теоретически драйверы могут работать вне ядра, но такую возможность поддерживают всего несколько существующих систем, гак как для этого требуется, чтобы драйвер в пространстве пользователя имел доступ к устройству неким контролируемым способом - очень редко поддерживаемое свойство.

Устройство ввода/вывода (3)

Есть три способа установки драйвера в ядро.

Первый: заключается в том, чтобы заново скомпоновать ядро вместе с новым драйвером и затем перезагрузить систему.

Второй: создать запись во входящем в операционную систему файле, говорящую о том, что требуется драйвер, и затем перезагрузить систему. Во время начальной загрузки операционная система сама находит нужные драйверы и загружает их.

Третий: операционная система может принимать новые драйверы, не прерывая работы, и оперативно устанавливать их, не нуждаясь при этом в перезагрузке. Этот способ используется, для таких съемных устройств, как шины USB и IEEE 1394.

Устройство ввода/вывода (3)

Ввод и вывод данных можно осуществлять тремя различными способами:

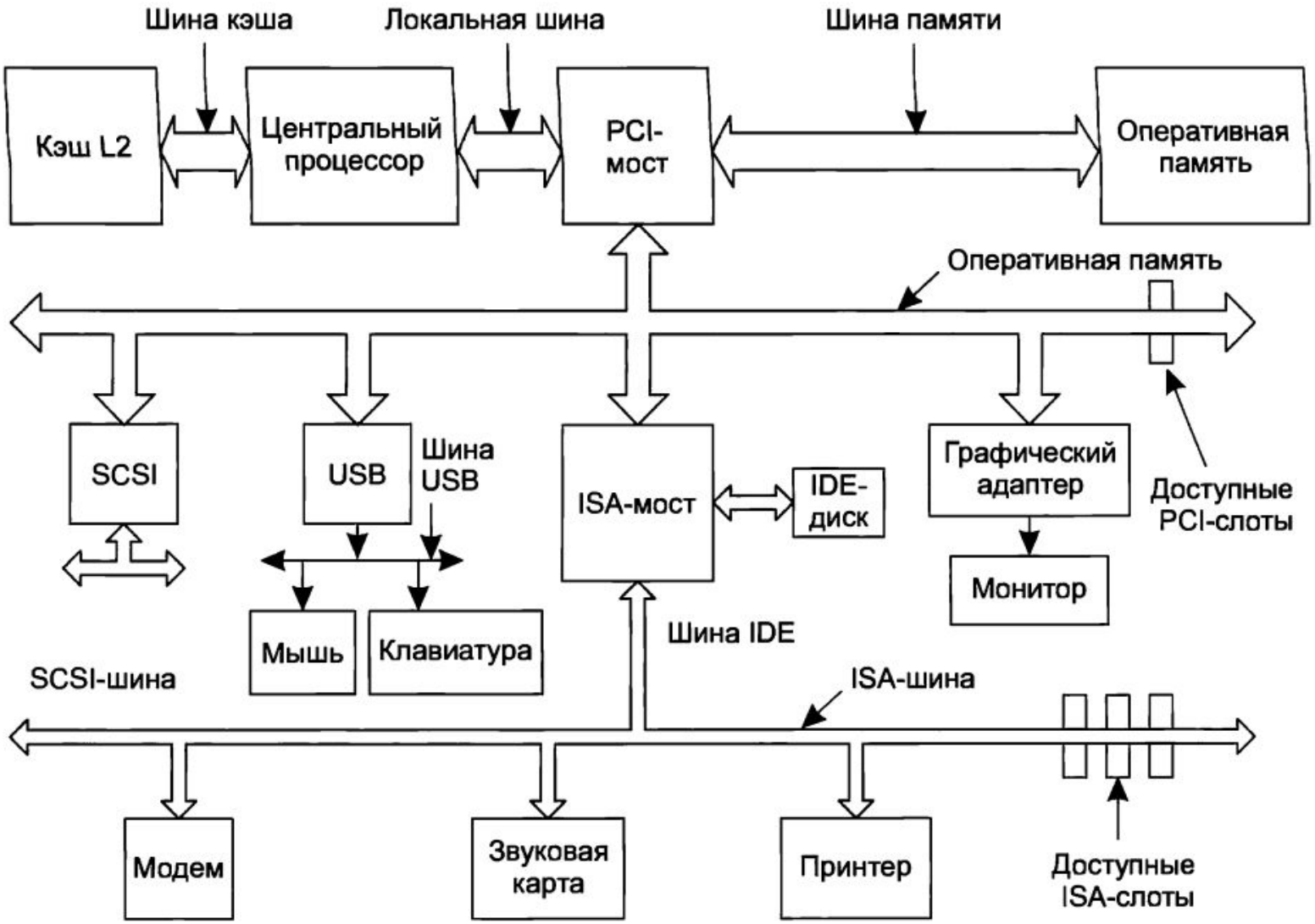
- **системный вызов**, который транслируется ядром в процедуру вызова соответствующего драйвера. После этого драйвер приступает к процессу ввода-вывода. Этот способ называется **активным ожиданием** или **ожиданием готовности.**
- драйвер запускает устройство и просит его выдать прерывание по окончании выполнения команды. Когда контроллер обнаруживает окончание передачи данных, он генерирует **прерывание**, чтобы просигнализировать о завершении операции.
- использование специального контроллера **прямого доступа к памяти** (DMA, Direct Memory Access), который может управлять потоком битов между оперативной памятью и некоторыми контроллерами без постоянного вмешательства центрального процессора.

Шины

Структура, показанная на слайде 41, в течение многих лет использовалась на мини-компьютерах. Далее в систему добавились дополнительные шины как для ускорения общения с устройствами ввода-вывода, так и для пересылки данных между процессором и памятью.

У этой системы восемь шин (шина кэша, локальная шина, шина памяти, PCI, SCSI, USB, IDE и ISA), каждая со своей скоростью передачи данных и своими функциями.

Шины (2)


Шина кэша
Локальная шина
Шина памяти
Кэш L2
Центральный процессор
PCI-мост
Оперативная память
Оперативная память
SCSI
USB
Шина USB
ISA-мост
IDE-диск
Графический адаптер
Доступные PCI-слоты
Монитор
Мышь
Клавиатура
Шина IDE
SCSI-шина
ISA-шина
Модем
Звуковая карта
Принтер
Доступные ISA-слоты

Шины (5)

В операционной системе для управления компьютером и его конфигурации должны находиться сведения обо всех этих шинах. Две основные шины - это **ISA** и **PCI**.

При такой конфигурации центральный процессор по локальной шине передает данные микросхеме PCI-моста, который, в свою очередь, обращается к памяти по выделенной шине памяти, часто работающей на частоте 100 МГц. Системы Pentium имеют кэш первого уровня (кэш L1), встроенный в процессор, и намного больший внешний кэш второго уровня (кэш L2), подключенный к процессору отдельной шиной кэша.

Кроме того, в систему входят три специализированные шины: IDE, USB и SCSI.

Шины (6)

Шина IDE служит для присоединения периферийных устройств к системе - дисков и устройств для чтения компакт-дисков (CD-ROM).

Шина USB служит для того, чтобы присоединить к компьютеру устройства ввода-вывода, такие как клавиатура и мышь.

Шина SCSI - это высокопроизводительная шина, применяемая для быстрых дисков, сканеров и других устройств, нуждающихся в значительной пропускной способности.

Шина IEEE 1394 (FireWire) - Как и USB, IEEE 1394 является бит-последовательной шиной.

Зоопарк операционных систем

1. Операционные системы мэйнфреймов
2. Серверные операционные системы
3. Многопроцессорные операционные системы
4. Операционные системы для персональных компьютеров
5. Операционные системы сенсорных узлов
6. Операционные системы реального времени
7. Встроенные операционные системы
8. Операционные системы для смарт-карт

Операционные системы мэйнфреймов

На самом верхнем уровне находятся операционные системы для мэйнфреймов. Эти компьютеры размером с комнату все еще можно встретить в центрах данных больших корпораций.

Мэйнфреймы отличаются от персональных компьютеров по своим возможностям ввода-вывода. Довольно часто встречаются мэйнфреймы с тысячью дисков и терабайтами данных, а персональный компьютер с такими параметрами показался бы действительно необычным.

Мэйнфреймы как бы возвращаются в виде мощных web-серверов, серверов для крупномасштабных электронно-коммерческих сайтов и серверов для транзакций в бизнесе.

Операционные системы мэйнфреймов (2)

Операционные системы для мэйнфреймов в основном ориентированы на обработку множества одновременных заданий, большинству из которых требуется огромное количество операций ввода-вывода.

Обычно они предлагают три вида обслуживания:

- пакетную обработку;
- обработку транзакций (групповые операции);
- разделение времени.

Операционные системы мэйнфреймов (3)

- **Пакетная обработка** представляет собой систему, выполняющую стандартные задания без присутствия пользователей, работающих в интерактивном режиме.
- **Системы обработки транзакций** управляют очень большим количеством маленьких запросов. Каждый отдельный запрос невелик, но система должна отвечать на сотни или тысячи запросов в секунду.
- **Системы**, работающие в режиме **разделения времени**, позволяют множеству удаленных пользователей одновременно выполнять свои задания на одной машине.

Примером операционной системы для мэйнфрейма является OS/390.

Серверные операционные системы

Уровнем ниже находятся серверные операционные системы. Они работают на серверах, которые представляют собой или очень большие персональные компьютеры, или рабочие станции, или даже мэйнфреймы.

Они одновременно обслуживают множество пользователей и позволяют им делить между собой программные и аппаратные ресурсы. Серверы предоставляют возможность работы с печатающими устройствами, файлами или Интернетом.

Серверные операционные системы (2)

UNIX системы, Windows 2000 и Windows Server 2003 являются типичными серверными операционными системами.

Пример использования:

Интернет-провайдеры запускают в работу несколько серверов для того, чтобы поддерживать одновременный доступ к сети множества клиентов. На серверах хранятся страницы web-сайтов и обрабатываются входящие запросы.

Многопроцессорные операционные системы

Все более часто применяемый способ увеличения мощности компьютеров заключается в соединении нескольких центральных процессоров в одной системе.

В зависимости от вида соединения процессоров и разделения работы такие системы называются:

- параллельными компьютерами;
- мультикомпьютерами;
- многопроцессорными системами.

Для них требуются специальные операционные системы, но зачастую такие операционные системы представляют собой варианты серверных операционных систем со специальными возможностями связи.

Операционные системы для персональных компьютеров

Следующую категорию составляют операционные системы для персональных компьютеров. Их работа заключается в предоставлении удобного интерфейса для одного пользователя. Такие системы широко используются для работы с текстом, электронными таблицами и доступа к Интернету.

Наиболее яркие примеры - это Windows 98, Windows 2000, Windows XP, Windows Vista, Mac OS и Linux.

Множество людей даже не имеет понятия о существовании других видов операционных систем, кроме той, которой они пользуются.

Операционные системы сенсорных узлов

- Сети, составленные из миниатюрных сенсорных узлов, связанных друг с другом и с базовой станцией по беспроводным каналам, развертываются для различных целей.
  - Применение: защита периметров зданий, охрана государственной границы, обнаружение возгораний в лесу, измерение температуры и уровня осадков, сбор информации о перемещениях противника на поле боя и многого другого.
- Узлы представляют собой миниатюрные компьютеры, питающиеся от батареи и имеющие встроенную радиосистему.
  - ограничены по мощности;
  - должны работать длительный период времени в необслуживаемом режиме в тяжелых климатических условиях.
  - сеть должна быть достаточно надежной и допускать отказы отдельных узлов.

Операционные системы сенсорных узлов

- Каждый сенсорный узел является компьютером, оснащенным процессором, оперативной памятью и постоянным запоминающим устройством, а также одним или несколькими датчиками.
- На нем работает небольшая операционная система, обычно управляемая событиями — откликающаяся на внешние события или периодически производящая измерения по сигналам встроенных часов.
  - Операционная система небольшая по объему и несложная, поскольку основной проблемой этих узлов является малая емкость оперативной памяти и ограниченное время работы батарей;
  - Все программы являются предварительно загруженными;

Примером широко известной операционной системы для сенсорных узлов может послужить TinyOS.

Операционные системы реального времени

Еще один вид операционной системы - это системы реального времени.

Главным параметром таких систем является **время**.

Например, в системах управления производством компьютеры, работающие в режиме реального времени, собирают данные о промышленном процессе и используют их для управления машинами на фабрике. Часто такие процессы должны удовлетворять жестким временным требованиям.

Если некоторое действие должно произойти в конкретный момент времени (или внутри заданного диапазона времени), мы имеем дело с жесткой системой реального времени.

Операционные системы реального времени (2)

Существует и другой вид: гибкая система реального времени, в которой допустимы случающиеся время от времени пропуски сроков выполнения операции.

В эту категорию попадают цифровое аудио и мультимедийные системы. Системы Vx Works и QNX являются операционными системами реального времени.

Встроенные операционные системы

Карманный компьютер или PDA.

Встроенные системы, управляющие действиями устройств, работают на машинах, обычно не считающихся компьютерами, например в телевизорах, микроволновых печах и мобильных телефонах. Они часто обладают теми же самыми характеристиками, что и системы реального времени, но при этом имеют особый размер, память и ограничения мощности, что выделяет их в отдельный класс.

Примерами таких операционных систем являются PalmOS, Windows CE, Windows Embedded.

Операционные системы для смарт-карт

Самые маленькие операционные системы работают на смарт-картах, представляющих собой устройство размером с кредитную карту, содержащее центральный процессор.

На такие операционные системы накладываются крайне жесткие ограничения по мощности процессора и памяти.

Некоторые из них могут управлять только одной операцией, например электронным платежом, но другие операционные системы на тех же самых смарт-картах выполняют сложные функции. Зачастую они являются патентованными системами.

Операционные системы для смарт-карт (2)

Некоторые смарт-карты являются Java-ориентированными. ПЗУ смарт-карт содержит интерпретатор виртуальной машины Java.

Апплеты Java загружаются на карту и выполняются JVM-интерпретатором. Некоторые из таких карт могут одновременно управлять несколькими апплетами Java, что приводит к многозадачности и необходимости планирования.

Из-за одновременной работы двух и более программ возникает необходимость в управлении ресурсами и защитой. Соответственно, все эти задачи выполняет обычно крайне примитивная операционная система, находящаяся на смарт-карте.

Понятия операционной системы

1. Процессы
2. Взаимоблокировка
3. Управление памятью
4. Ввод-вывод данных
5. Файлы
6. Безопасность
7. Оболочка

Процессы

Ключевое понятие операционной системы - **процесс**.

Процессом, по существу, называют *программу в момент выполнения*.

С каждым процессом связывается его адресное пространство - список адресов в памяти, которые процесс может прочесть и в которые он может писать.

Процессы: состав процесса

- Адресное пространство
  - программу,
  - данные;
  - стек;
- набор регистров,
- счетчик команд,
- указатель стека,
- другие аппаратные регистры,
- вся остальная информация, необходимая для запуска программы.

Процессы (2)

Рассмотрим системы, работающие в режиме разделения времени.

Предположим, периодически операционная система решает остановить работу одного процесса и запустить другой, потому что первый израсходовал отведенную для него часть рабочего времени центрального процессора в прошедшую секунду.

Если процесс был приостановлен подобным образом, позже он должен быть запущен заново из того же состояния, в каком его остановили. Следовательно, всю информацию о процессе нужно где-либо явно сохранить на время его приостановки. Вся информация о каждом процессе, хранится в **таблице процессов**.

Процессы (3)

Приостановленный процесс состоит из собственного адресного пространства - **образа памяти** и **компонентов таблицы процесса**, содержащей его регистры. Главными системными вызовами, управляющими процессами, являются вызовы, связанные с созданием и окончанием процессов.

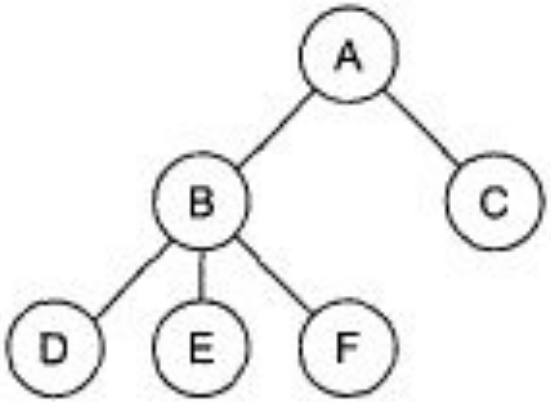
Пример:

Процесс, называемый **интерпретатором команд**, читает команды с терминала. Пользователь напечатал команду, содержащую запрос на компиляцию программы. Теперь оболочка должна создать новый процесс, который запустит компилятор. Когда процесс закончит компиляцию, он выполнит системный вызов, завершающий его собственную работу.

Процессы (4)

Если процесс может создавать несколько других процессов (называющихся **дочерними процессами**), а эти процессы, в свою очередь, тоже могут создать дочерние процессы, перед нами предстает дерево процессов.

Связанные процессы - это те, которые объединены для выполнения некоторой задачи, и им нужно часто передавать данные от одного к другому и синхронизировать свою деятельность.



Процессы (5)

Другие системные вызовы предназначаются для запросов о предоставлении дополнительной памяти, ожидании завершения дочерних процессов и наложении одной программы на другую.

Время от времени необходимо передавать информацию работающему процессу так, чтобы он не простаивал в ожидании получения этой информации.

Например:

процесс, связанный с другим процессом на удаленном компьютере, делает это, посылая сообщения по сети.

Процессы (6)

Если по истечении определенного количества секунд нет ответа от процесса операционная система посылает процессу **сигнал тревоги**.

Сигнал вызывает временную остановку работы процесса независимо от того, что процесс делает в данный момент; сохраняет его регистры в стеке и запускает специальную процедуру обработки сигнала.

После завершения обработки сигнала работающий процесс запускается заново в том состоянии, в котором он находился до сигнала.

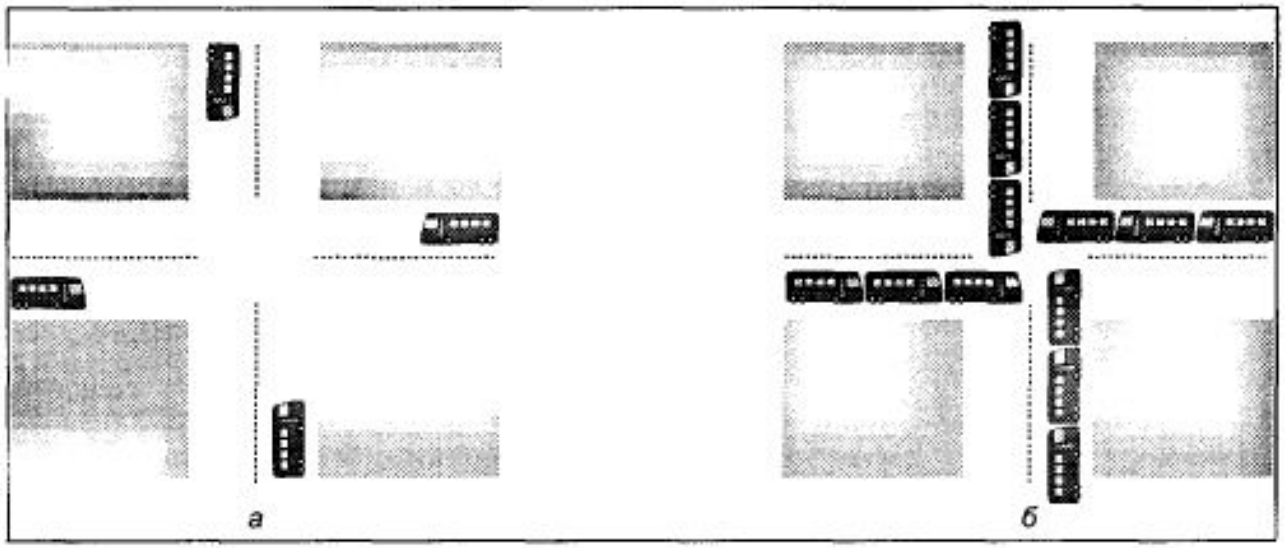
Процессы (7)

Каждому пользователю, которому разрешено пользоваться системой, системный администратор присваивает **UID** (User IDentification – **идентификатор пользователя**). У каждого работающего процесса есть идентификатор пользователя, запустившего его. Дочерний процесс получает тот же самый UID, что и его родитель.

Пользователи могут становиться членами групп, каждая из которых имеет идентификатор группы (GID, Group IDentification).

Пользователь с особым идентификатором UID, называемый в UNIX «суперпользователем» (superuser), имеет особые полномочия и может игнорировать множество правил защиты.

Взаимоблокировка



Когда взаимодействуют два или более процессов, они могут попадать в патовые ситуации, из которых невозможно выйти без посторонней помощи. Такая ситуация называется **тупиком**, тупиковой ситуацией или **взаимоблокировкой**.

Управление памятью

Оперативная память, используется для хранения выполняющихся программ.

В очень простых операционных системах в конкретный момент времени в памяти может находиться только одна программа. Для запуска второй программы сначала нужно удалить из памяти первую и загрузить на ее место вторую.

Более изощренные системы позволяют одновременно находиться в памяти нескольким программам. Для того чтобы они не мешали друг другу, необходим некий защитный механизм. Хотя этот механизм располагается в аппаратуре, он управляется операционной системой.

Ввод-вывод данных

Во всех компьютерах есть физическое устройство для получения входных данных и вывода информации. Существует много видов устройств ввода-вывода, например клавиатуры, мониторы, принтеры и т. д. Всеми ими должна управлять операционная система.

Каждая операционная система имеет свою подсистему ввода-вывода для управления устройствами ввода-вывода. Некоторые из программ ввода-вывода являются независимыми от устройств. Другая часть программного обеспечения ввода-вывода, в которую входят драйверы устройств, предназначена для определенных устройств ввода-вывода.

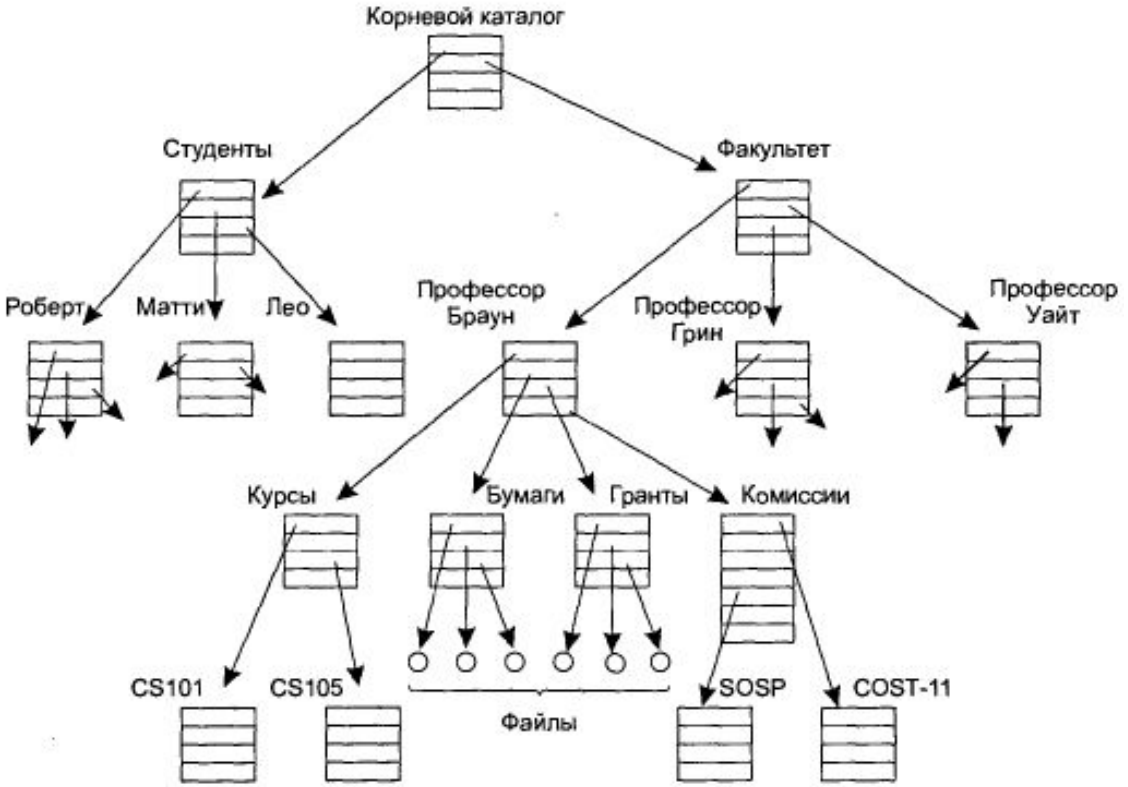
Файл

Файловая система - это еще одно ключевое понятие, поддерживаемое виртуально всеми операционными системами.

Предоставляя место для хранения файлов, операционные системы используют понятие каталога как способ объединения файлов в группы. Для создания и удаления каталогов также необходимы системные вызовы. Они же обеспечивают перемещение существующего файла в каталог и удаление файла из каталога. Содержимое каталогов могут составлять файлы или другие каталоги.

Файл (2)

Каждый файл в иерархии каталогов можно определить, задав его имя пути, называемое также полным именем файла. Путь начинается из вершины структуры каталогов, называемой **корневым каталогом**.



Файл (3)

Важное понятие в UNIX - это **специальный файл**.

Специальные файлы служат для того, чтобы устройства ввода-вывода выглядели как файлы. При этом можно прочесть информацию из специальных файлов или записать ее туда с помощью тех же самых системных вызовов, что используются для чтения и записи файлов.

Существует два вида специальных файлов:

- блочные специальные файлы;
- символьные специальные файлы.

Файл (4)

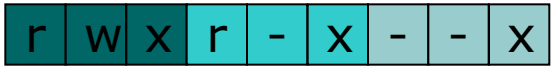
Каналы (pipe), имеющие отношение и к процессам и к файлам. Канал (труба) представляет собой псевдофайл, который можно использовать для связи двух процессов.

Таким образом, соединение между процессами в UNIX выглядит очень похожим на обычное чтение и запись файлов. Более того, только сделав специальный системный вызов, процесс может обнаружить, что выходной файл, в который он пишет данные, не реальный файл, а канал.

Безопасность

Компьютеры содержат большое количество информации, конфиденциальность которой пользователи зачастую хотят сохранить: электронную почту, бизнес-планы и многое другое. В задачу операционной системы входит управление системой защиты подобных файлов.

В качестве простейшего примера, дающего представление о том, как работает система безопасности, рассмотрим систему UNIX. В UNIX для защиты файлов им присваивается 9-битовый двоичный код.



Кроме защиты файлов, существует еще множество других вопросов безопасности: защита системы от нежелательных гостей, людей, и не только (вирусов).

Оболочка

Операционная система представляет собой программу, выполняющую системные вызовы.

Редакторы, компиляторы, ассемблеры, компоновщики и командные интерпретаторы не являются частью операционной системы, несмотря на их большую важность и полезность.

Часть 2.

Системные вызовы

Интерфейс между операционной системой и программами пользователя определяется набором системных вызовов, предоставляемых операционной системой.

Системные вызовы, доступные в интерфейсе, меняются от одной операционной системы к другой:

1. неопределенные обобщения («операционные системы имеют системные вызовы для чтения файлов»);
2. какая-либо конкретная система («в UNIX существует системный вызов для чтения с тремя параметрами: один для задания файла, второй - для того, чтобы указать, куда нужно поместить прочитанные данные, третий задает количество байтов, которое нужно прочитать»).

Системные вызовы (2)

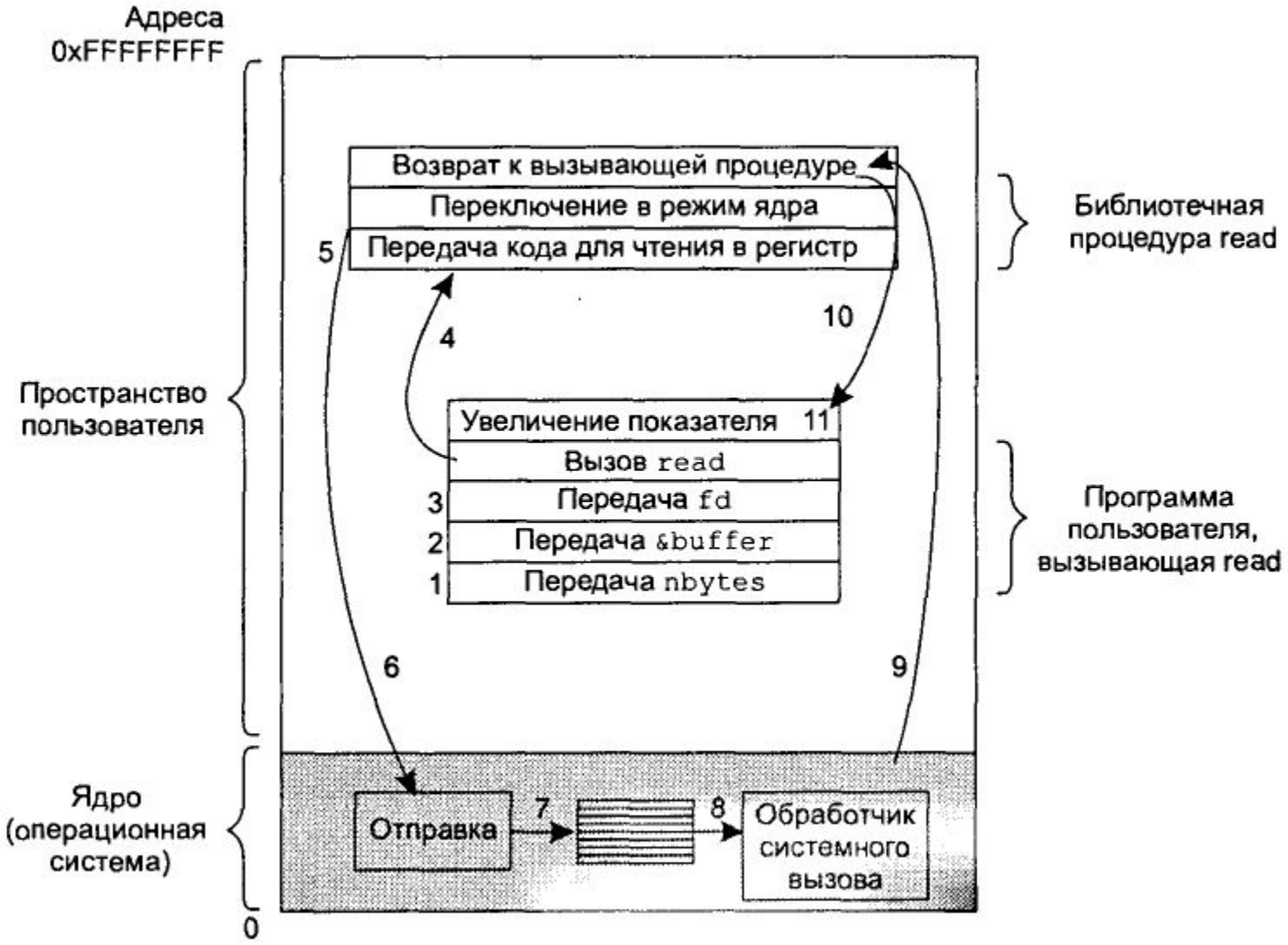
Рассмотрим системный вызов *read*.

Как практически все системные вызовы, он запускается из программы на С с помощью вызова библиотечной процедуры с тем же именем, что и системный вызов: *read*.

Вызов из программы на С может выглядеть так:

count = read(fd, buffer, nbytes)

Системные вызовы (3)



Вызов	Описание
Управление процессами	
Pid=fork()[1]	Создает дочерний процесс, идентичный родительскому
Pid=waitpid(pid, &statloc, options)	Ожидает завершения дочернего процесса
s=execve(name, argv, environp)	Перемещает образ памяти процесса
Exit(status)	Завершает выполнение процесса и возвращает статус
Управление файлами	
fd=open(file, how, ...)	Открывает файл для чтения, записи или того и другого
s=close(fd)	Закрывает открытый файл
n=read(fd, buffer, nbytes)	Читает данные из файла в буфер
n=write(fd, buffer, nbytes)	Пишет данные из буфера в файл
Position=lseek(fd, offset, whence)	Передвигает указатель файла
s=stat(name, &buf)	Получает информацию о состоянии файла
Управление каталогами и файловой системой	
s=mkdir(name, mode)	Создает новый каталог
s=rmdir(name)	Удаляет пустой каталог
s=link(name1, name2)	Создает новый элемент с именем name2, указывающий на name1
s=unlink(name)	Удаляет элемент каталога
s=mount(special, name, flag)	Монтирует файловую систему
s=umount(special)	Демонтирует файловую систему
Разные	
s=chdir(dimame)	Изменяет рабочий каталог
s=chmod(name, mode)	Изменяет биты защиты файла
s=kill(pid, signal)	Посылает сигнал процессу
Seconds=time(&seconds)	Получает время, прошедшее с 1 января 1970 года

Системные вызовы

1. Системные вызовы для управления процессами
2. Системные вызовы для управления файлами
3. Системные вызовы для управления каталогами
4. Разные системные вызовы
5. Windows Win32 API

Системные вызовы для управления процессами

Системный вызов **fork** – создание нового процесса в UNIX.

Он создает точную копию исходного процесса, включая дескрипторы файла, регистры и т. п.

После вызова **fork** исходный процесс и его копия (родительский и дочерний) развиваются по отдельности друг от друга.

Вызов **fork** возвращает величину, равную нулю в дочернем процессе и равную идентификатору дочернего процесса или PID в родительском.

Системные вызовы для управления процессами (2)

В большинстве случаев после вызова fork дочернему процессу необходимо выполнить программный код, отличный от предназначенного для родительского процесса.

Рассмотрим пример оболочки:

Она читает команды с терминала, запускает дочерний процесс, ждет, пока дочерний процесс выполнит команду, и читает следующую команду после завершения работы дочернего процесса.

Системные вызовы для управления файлами

Открытие файла осуществляется при помощи вызова **open**. Для этого вызова указывается имя открываемого файла и код.

Файл закрывается с помощью вызова **close**, который делает дескриптор файла доступным при следующем открытии (**open**).

Системные вызовы для управления файлами

Вызов **Lseek** возвращает абсолютную позицию в файле. У вызова три параметра:

- это идентификатор файла;
- позиция в файле;
- третий говорит, является ли второй параметр позицией в файле относительно начала файла (абсолютная позиция), относительно текущей позиции или относительно конца файла.

Для каждого файла UNIX хранит следующие данные: тип файла (обычный, специальный, каталог и т. д.), размер, время последнего изменения и другую информацию.

Системные вызовы для управления каталогами

- **mkdir** – создает пустые каталоги;
- **rmdir** – удаляет пустые каталоги;
- **link** – Он разрешает одному файлу появляться под двумя или более именами, часто в разных каталогах.
- Системный вызов **mount** позволяет объединять в одну две файловые системы.

Windows Win32 API

Фактические системные вызовы и запускающиеся для их выполнения библиотечные вызовы полностью разделены. Корпорацией Microsoft определен набор процедур, называемый Win32 API (Application Program Interface - интерфейс прикладных программ).

Отделяя интерфейс от фактических системных вызовов, Microsoft поддерживает возможность изменения со временем действительных системных вызовов (даже от одной версии к другой), не делая при этом недействительными существующие программы.

В UNIX графический интерфейс пользователя (например, X Windows или Motif) запускается целиком в пользовательском пространстве.

Windows Win32 API (2)

В Win32 API имеет огромное количество вызовов для управления окнами, геометрическими фигурами, текстом, шрифтами, полосами прокрутки, диалоговыми окнами, пунктами меню и другими элементами графического интерфейса.

В том случае, когда графическая подсистема запускается в режиме ядра (это верно для большинства версий Windows, но не для всех), вызовы являются системными; в противном случае вызовы являются только библиотечными.

UNIX	Win32	Описание
Fork	CreateProcess	Создать новый процесс
Waitpid	WaitForSingleObject	Ждать завершения процесса
Execve	(нет)	CreateProcess=fork+execve
Exit	ExitProcess	Завершить выполнение
Open	CreateFile	Создать файл или открыть существующий файл
Close	CloseHandle	Закрыть файл
Read	ReadFile	Читать данные из файла
Write	WriteFile	Записать данные в файл
Lseek	SetFilePointer	Передвинуть указатель файла
Stat	GetFileAttributesEx	Получить различные атрибуты файла
Mkdir	CreateDirectory	Создать новый каталог
Rmdir	RemoveDirectory	Удалить пустой каталог
Link	(нет)	Win32 не поддерживает связи
Unlink	DeleteFile	Удалить существующий файл
Mount	(нет)	Win32 не поддерживает монтирование
Urnount	(нет)	Win32 не поддерживает монтирование
chdir	SetCurrentDirectory	Изменить текущую рабочую папку
chmod	(нет)	Win32 не поддерживает защиту файла (хотя NT поддерживает)
kill	(нет)	Win32 не поддерживает сигналы
time	GetLocalTime	Получить текущее время

Разные системные вызовы

- Вызов **chdir** изменяет текущий рабочий каталог.

В UNIX для каждого файла определен режимный код файла (mode), используемый для его защиты.

- вызов **chmod** предоставляет возможность изменения режимного кода файла.

Например, *chmod(“file”, 0644);*

- вызов **kill** позволяет пользователям и пользовательским процессам посылать сигналы.
- Стандартом POSIX определено несколько процедур, имеющих отношение ко времени.

Например, **time** возвращает текущее время в секундах.

Структура операционной системы

1. Монолитные системы
2. Многоуровневые системы
3. Виртуальные машины
4. Экзоядро
5. Модель клиент-сервер

Монолитные системы

В общем случае организация монолитной системы представляет собой «большой беспорядок».

Структура отсутствует как таковая.

Операционная система написана в виде набора процедур, каждая из которых может вызывать другие, когда ей это нужно. При использовании такой техники каждая процедура системы имеет строго определенный интерфейс.

Для построения монолитной системы необходимо скомпилировать все отдельные процедуры, а затем связать их в единый объектный файл с помощью компоновщика. Полностью отсутствует сокрытие деталей реализации - каждая процедура видит любую другую процедуру.

Монолитные системы

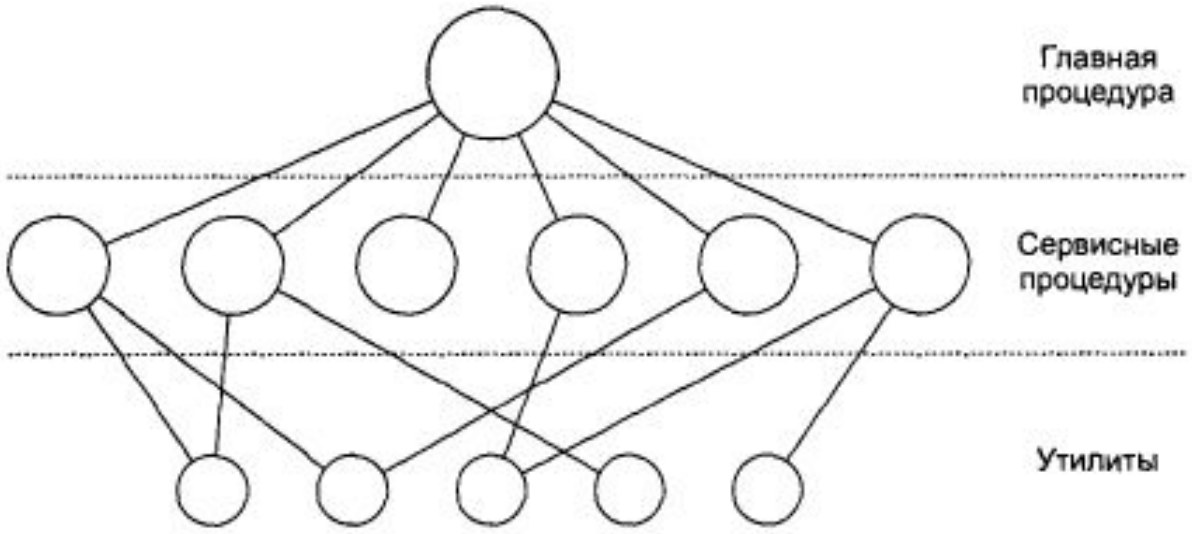
Монолитные системы могут иметь некоторую структуру.

При обращении к системным вызовам, поддерживаемым операционной системой, параметры помещаются в строго определенные места - регистры или стек, после чего выполняется специальная команда прерывания, известная как **вызов ядра** или **вызов супервизора**.

Такая организация операционной системы предполагает следующую структуру:

1. Главная программа.
2. Набор служебных процедур.
3. Набор утилит.

Монолитные системы



В этой модели для каждого системного вызова имеется одна служебная процедура. Утилиты выполняют функции, которые нужны нескольким служебным процедурам.

Многоуровневые системы

Обобщением предыдущего подхода, является организация операционной системы в виде иерархии уровней. Первой системой, построенной таким образом, была система THE, созданная в Э. Дейкстрой и его студентами в 1968 году. Она была простой пакетной системой для голландского компьютера Electrologica X8.

Система включала 6 уровней.

Уровень	Функция
5	Оператор
4	Программы пользователя
3	Управление вводом-выводом
2	Связь оператор-процесс
1	Управление памятью и барабаном
0	Распределение процессора и многозадачность

Многоуровневые системы

1. *Уровень 0* занимался распределением времени процессора, переключая процессы при возникновении прерывания или при срабатывании таймера.
2. *Уровень 1* управлял памятью. Он выделял процессам пространство в оперативной памяти и на магнитном барабане.
3. *Уровень 2* управлял связью между консолью оператора и процессами.
4. *Уровень 3* управлял устройствами ввода-вывода и буферизовал потоки информации к ним и от них.
5. *На уровне 4* работали пользовательские программы.
6. Процесс системного оператора размещался *на уровне 5*.

Виртуальные машины

Принцип построения виртуальной машины:

- система с разделением времени обеспечивает многозадачность;
- расширенная машина с более удобным интерфейсом.

VM/370 основана на полном разделении этих двух функций.

Сердце системы, называемое **монитором виртуальной машины**, работает с оборудованием и обеспечивает многозадачность, предоставляя верхнему слою не одну, а несколько виртуальных машин.

В отличие от всех других операционных систем, эти виртуальные машины не являются расширенными.

Виртуальные машины (2)

На разных виртуальных машинах могут функционировать различные операционные системы. На некоторых из них для обработки пакетов и транзакций работают потомки OS/360, а на других для интерактивного разделения времени пользователей работает однопользовательская интерактивная система CMS (Conversational Monitor System - система диалоговой обработки).

Виртуальные машины (3)

Идея виртуальной машины очень часто используется в наши дни, но в несколько другом контексте: для работы старых программ, написанных для системы MS-DOS на Pentium.

При разработке компьютера Pentium и его программного обеспечения обе компании, Intel и Microsoft, понимали, что возникнет острая потребность в работе старых программ на новом оборудовании. Поэтому корпорация Intel создала на процессоре Pentium режим виртуального процессора 8086.

Виртуальные машины (4)

Возможны два варианта устройства.

1. сама система MS-DOS загружена в адресное пространство виртуальной машины 8086, так что монитор виртуальной машины только отсылает прерывания назад к MS-DOS, как это происходит на реальной 8086. Когда затем MS-DOS пытается самостоятельно осуществить ввод-вывод, операция перехватывается и выполняется монитором виртуальной машины.
2. монитор виртуальной машины перехватывает первое прерывание и сам выполняет ввод-вывод, так как он знает все системные вызовы MS-DOS и имеет представление о том, что должно делать каждое прерывание.

Виртуальные машины (5)

Для работы программ Java виртуальные машины используются несколько другим способом. Когда корпорация Sun Microsystems придумала язык программирования Java, она также разработала **виртуальную машину** (то есть архитектуру компьютера), называемую **JVM** (**Java Virtual Machine** - виртуальная машина Java).

Компилятор Java выдает код для JVM, который затем обычно выполняется программным интерпретатором JVM. Преимущество этого подхода заключается в том, что код JVM можно передавать через Internet на любой компьютер, имеющий интерпретатор JVM, и запускать там.

Экзоядро

В системе VM/370 каждый пользователь получает точную копию настоящей машины.

На Pentium, в режиме виртуальной машины 8086, каждый пользователь получает точную копию другой машины.

Развив эту идею дальше, исследователи из Массачусетсского технологического института изобрели систему, которая обеспечивает каждого пользователя абсолютной копией реального компьютера, но с подмножеством ресурсов.

Экзоядро

На нижнем уровне в режиме ядра работает программа, которая называется **экзоядро (exokernel)**.

В ее задачу входит распределение ресурсов для виртуальных машин, а после этого проверка их использования.

Каждая виртуальная машина на уровне пользователя может работать с собственной операционной системой, как на VM/370 или виртуальных 8086-х для Pentium, с той разницей, что каждая машина ограничена набором ресурсов, которые она запросила и которые ей были предоставлены.

Экзоядро (2)

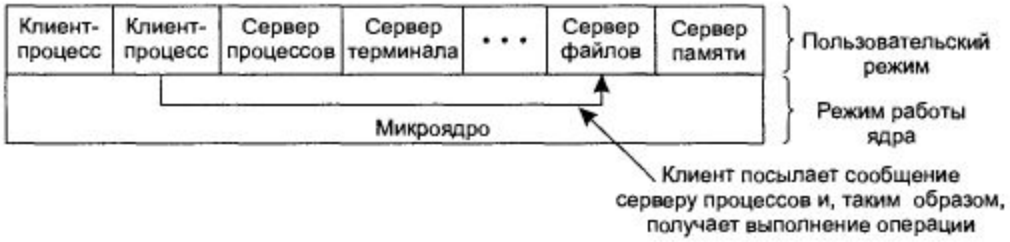
Преимущество схемы экзоядра заключается в том, что она позволяет обойтись без уровня отображения.

При других методах работы каждая виртуальная машина считает, что она использует свой собственный диск с нумерацией блоков от 0 до некоторого максимума. Поэтому монитор виртуальной машины должен поддерживать таблицы преобразования адресов на диске (и всех других ресурсов).

Необходимость преобразования отпадает при наличии экзоядра, которому нужно только хранить запись о том, какой виртуальной машине выделен данный ресурс.

Такой подход имеет еще одно преимущество: он отделяет многозадачность (в экзоядре) от операционной системы пользователя (в пространстве пользователя) с меньшими затратами, так как для этого ему необходимо всего лишь не допускать вмешательства одной виртуальной машины в работу другой.

Клиент-сервер

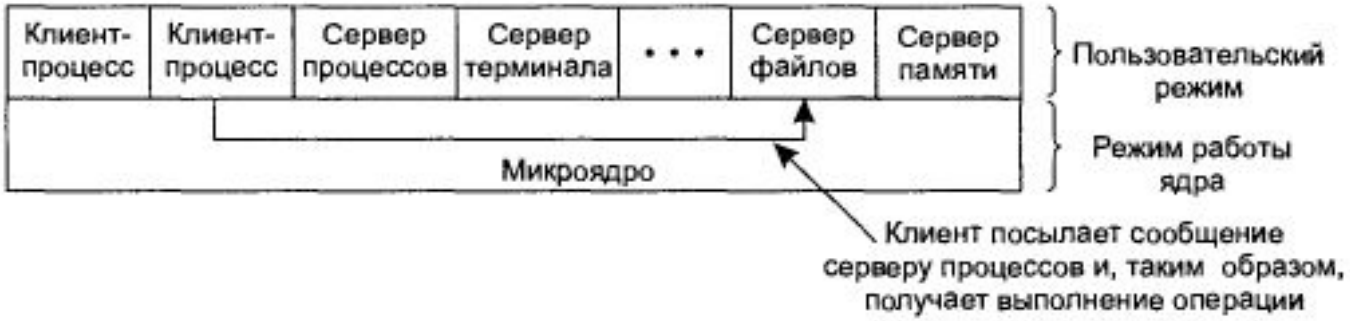


В развитии современных операционных систем наблюдается тенденция в сторону дальнейшего переноса кода в верхние уровни и удалении при этом всего, что только возможно, из режима ядра, оставляя **минимальное микроядро**.

Обычно это осуществляется перекладыванием выполнения большинства задач операционной системы на средства пользовательских процессов.

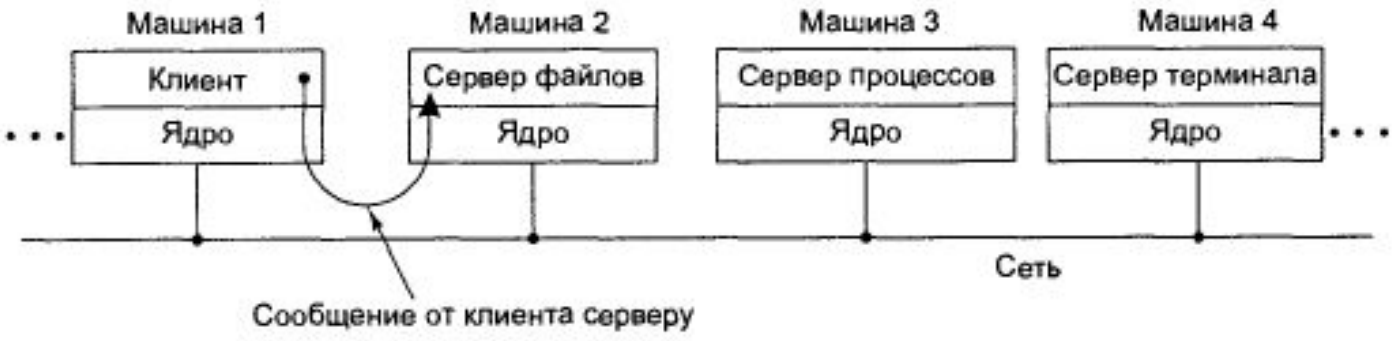
Получая запрос на какую-либо операцию, например чтение блока файла, пользовательский процесс (теперь называемый **обслуживаемым процессом** или **клиентским процессом**) посылает запрос **серверному процессу**, который его обрабатывает и высылает назад ответ.

Клиент-сервер (2)



В данной модели, в задачу ядра входит только управление связью между клиентами и серверами. Благодаря разделению операционной системы на части, каждая из которых управляет всего одним элементом системы, все части становятся маленькими и управляемыми. К тому же, поскольку все серверы работают как процессы в режиме пользователя, а не в режиме ядра, они не имеют прямого доступа к оборудованию.

Клиент-сервер (3)



Другое преимущество модели клиент-сервер заключается в ее простой адаптации к использованию в распределенных системах. Если клиент общается с сервером, посылая ему сообщения, клиенту не нужно знать, обрабатывается ли его сообщение локально на его собственной машине или оно было послано по сети серверу на удаленной машине. С точки зрения клиента происходит одно и то же в обоих случаях: запрос был послан, и на него получен ответ.

Итог

- Операционную систему можно рассматривать с двух точек зрения:
  - как менеджер ресурсов
  - как расширенную машину.
- Операционные системы имеют достаточно долгую историю развития, которая начинается с тех дней, когда операционные системы заменили оператора, и продолжается до современных многозадачных систем.
- Поскольку операционные системы тесно взаимодействуют с оборудованием, некоторые знания об аппаратуре могут оказаться очень полезны для понимания работы операционной системы.
- Основными понятиями, на которых построена операционная система, являются процессы, управление памятью, управление вводом-выводом, файловая система и безопасность.
- Сердцем любой операционной системы является набор системных вызовов, которые она может обработать. Они говорят о том, что реально делает операционная система.
- Операционная система может быть структурирована несколькими способами. Наиболее общими выделяемыми при структурировании понятиями являются: монолитные системы, иерархия слоев, система виртуальных машин, экзоядро или использование модели клиент-сервер.