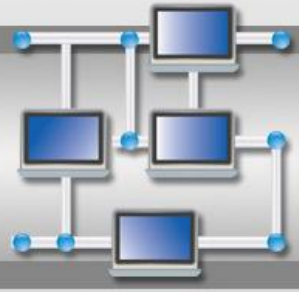
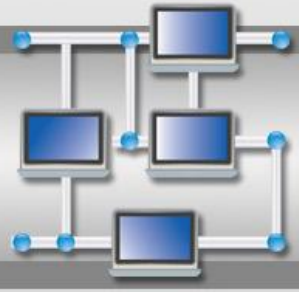
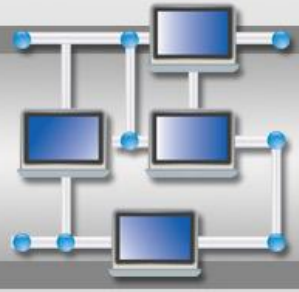


Тема 10. Операционные системы с сетевыми возможностями

Гончаров Сергей Леонидович,
старший преподаватель

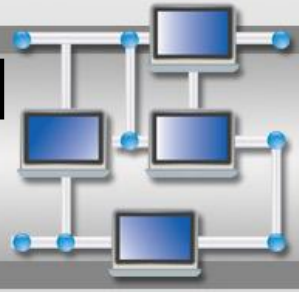


- Структура таких операционных систем.
- Сетевые оболочки и встроенные средства ОС с выделенными серверами.
- Одноранговые сети.
- Функции ОС по управлению локальными ресурсами.
- Определение локальной ОС.
- Управление процессами, памятью, файловой системой.
- Функции ОС по организации сетевой работы.
- Примитивы передачи сообщений.
- Вызов удаленных процедур.
- Кэширование файлов в распределенных системах
- Проблема согласования копий
- Репликация.
- Требования к современным ОС, передовые технологии проектирования ОС, критерии выбора сетевых ОС
- Обзор популярных семейств сетевых ОС.



СТРУКТУРА СЕТЕВЫХ ОПЕРАЦИОННЫХ СИСТЕМ

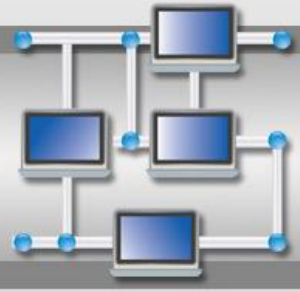
Сетевая операционная система



- Сетевая операционная система составляет основу любой вычислительной сети.
- Каждый компьютер в сети в значительной степени автономен, поэтому под сетевой операционной системой в широком смысле понимается совокупность операционных систем отдельных компьютеров, взаимодействующих с целью обмена сообщениями и разделения ресурсов по единым правилам - протоколам.
- В узком смысле сетевая ОС - это операционная система отдельного компьютера, обеспечивающая ему возможность работать в сети.

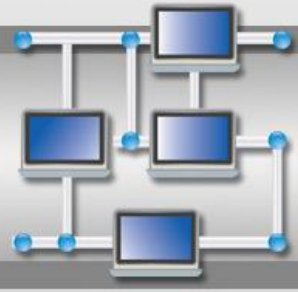

Средства управления локальными ресурсами
(Локальная ОС)
Серверная часть
Клиентская часть
Средства предоставления собственных ресурсов в общее пользование
Средства запроса доступа к удаленным ресурсам и услугам
Оболочка
Коммуникационные средства
в сеть

В сетевой операционной системе отдельной машины можно выделить несколько частей



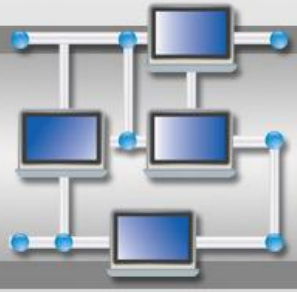
- Средства управления локальными ресурсами компьютера:
  - функции распределения оперативной памяти между процессами, планирования и диспетчеризации процессов, управления процессорами в мультипроцессорных машинах, управления периферийными устройствами и другие функции управления ресурсами локальных ОС.
- Средства предоставления собственных ресурсов и услуг в общее пользование - серверная часть ОС (сервер).
  - Эти средства обеспечивают блокировку файлов и записей, что необходимо для их совместного использования; ведение справочников имен сетевых ресурсов; обработку запросов удаленного доступа к собственной файловой системе и базе данных; управление очередями запросов удаленных пользователей к своим периферийным устройствам.

В сетевой операционной системе отдельной машины можно выделить несколько частей



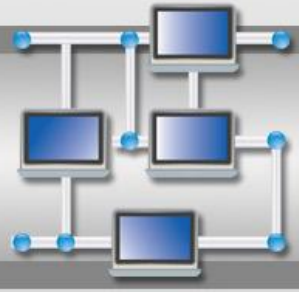
- Средства запроса доступа к удаленным ресурсам и услугам и их использования - клиентская часть ОС (редиректор).
  - Эта часть выполняет распознавание и перенаправление в сеть запросов к удаленным ресурсам от приложений и пользователей, при этом запрос поступает от приложения в локальной форме, а передается в сеть в другой форме, соответствующей требованиям сервера. Клиентская часть также осуществляет прием ответов от серверов и преобразование их в локальный формат, так что для приложения выполнение локальных и удаленных запросов неразличимо.
- Коммуникационные средства ОС, с помощью которых происходит обмен сообщениями в сети.
  - Эта часть обеспечивает адресацию и буферизацию сообщений, выбор маршрута передачи сообщения по сети, надежность передачи и т.п., то есть является средством транспортировки сообщений.

Сетевая ОС



предоставляет пользователю некую виртуальную вычислительную систему, работать с которой гораздо проще, чем с реальной сетевой аппаратурой.

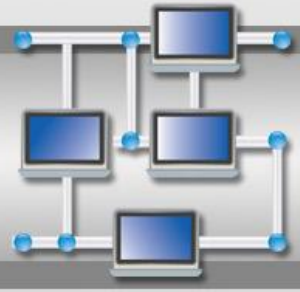
Распределенная ОС



динамически и автоматически распределяя работы по различным машинам системы для обработки, заставляет набор сетевых машин работать как виртуальный унипроцессор.

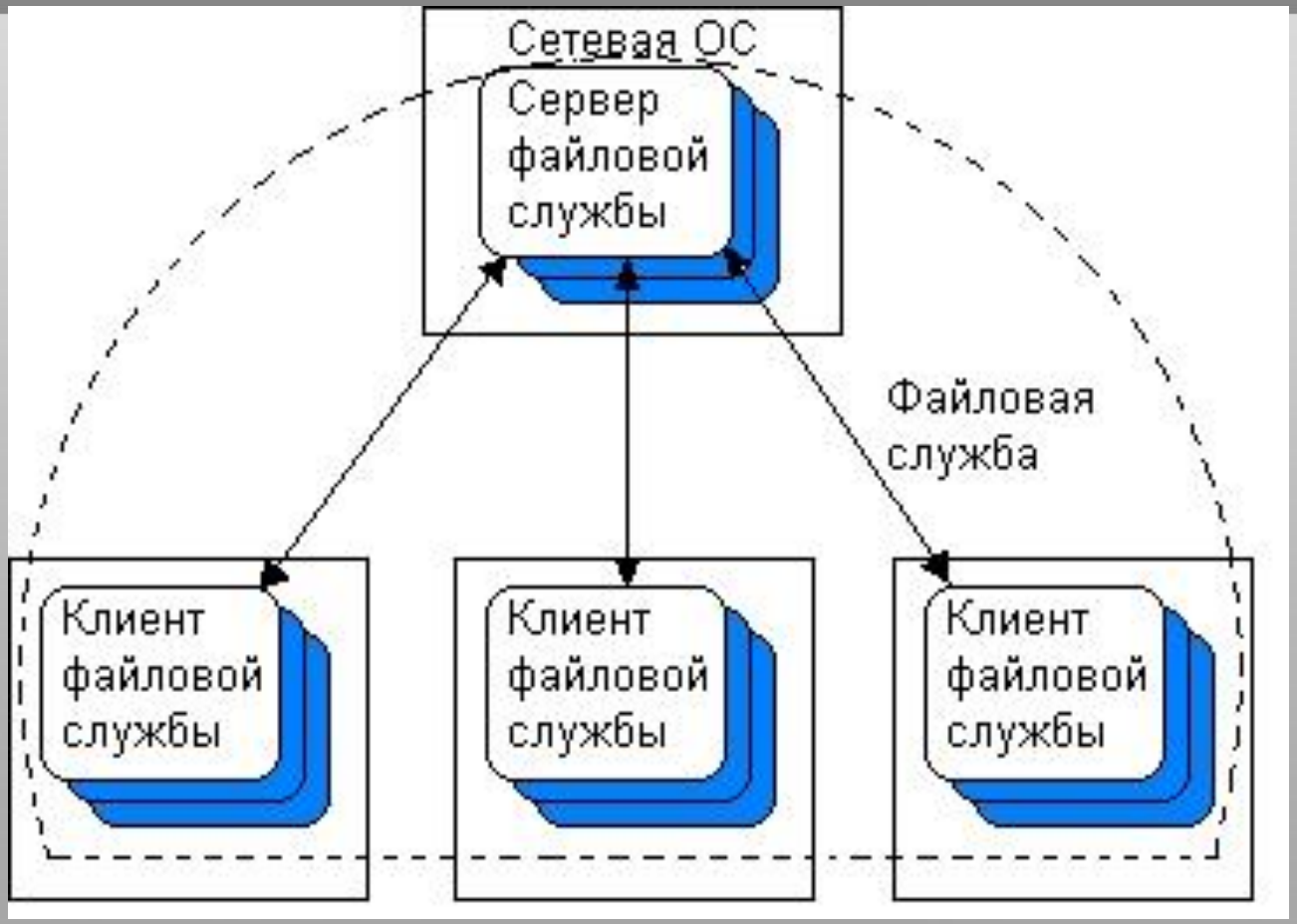
- Пользователь распределенной ОС, вообще говоря, не имеет сведений о том, на какой машине выполняется его работа.
- Существует как единая операционная система в масштабах вычислительной системы. Каждый компьютер сети, работающей под управлением распределенной ОС, выполняет часть функций этой глобальной ОС.

Сетевые службы и сетевые сервисы

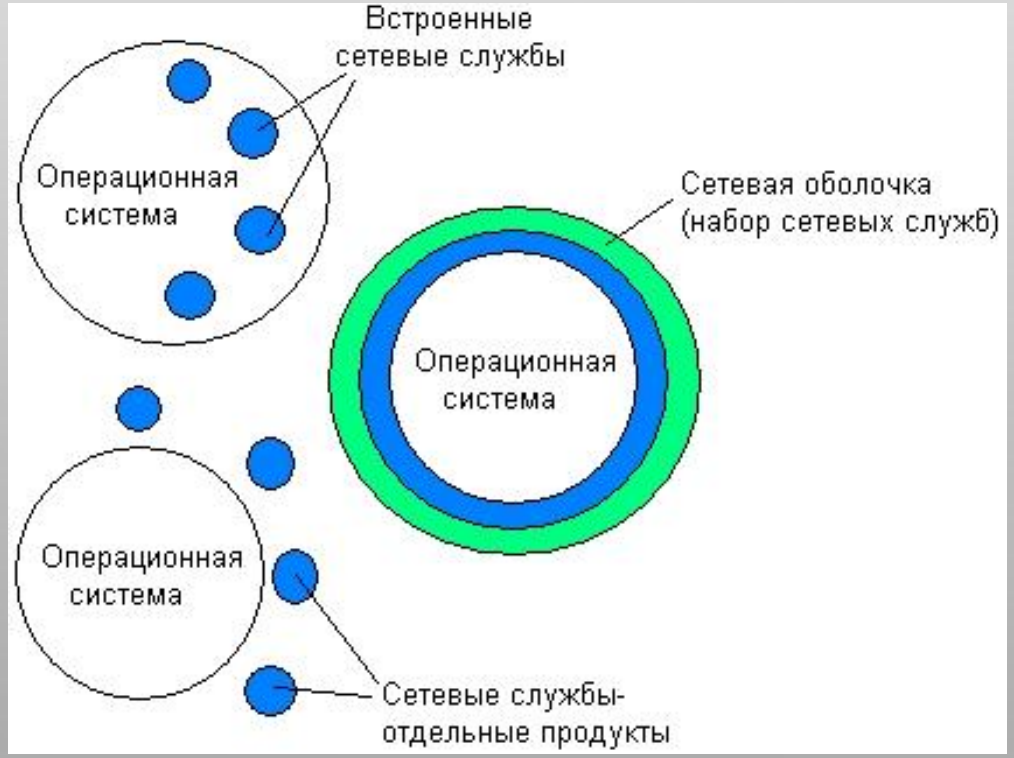


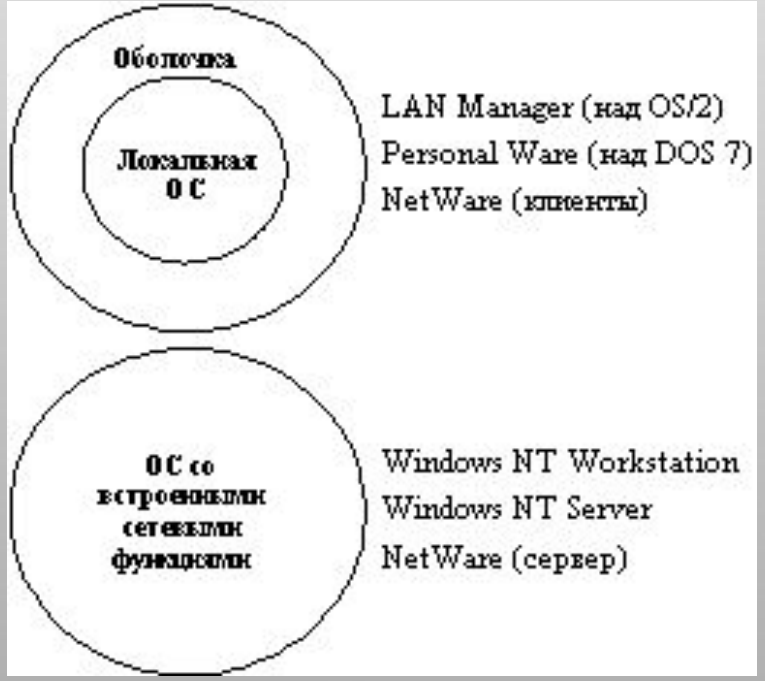
- Совокупность серверной и клиентской частей ОС, предоставляющих доступ к конкретному типу ресурса компьютера через сеть, называется *сетевой службой.*
- Сетевые службы по своей природе являются клиент-серверными системами. Поскольку при реализации любого сетевого сервиса естественно возникает источник запросов (клиент) и исполнитель запросов (сервер), то и любая сетевая служба содержит в своем составе две несимметричные части — клиентскую и серверную.

Сетевые службы и сетевые сервисы



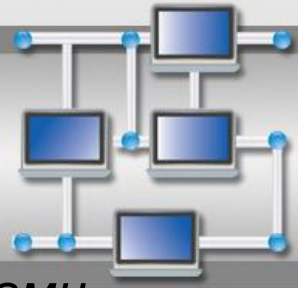
Встроенные сетевые службы и сетевые оболочки



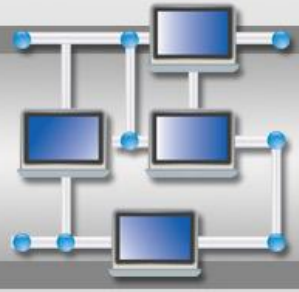


- сетевые службы глубоко *встроены* в ОС;
- сетевые службы объединены в виде некоторого набора — *оболочки;*
- сетевые службы производятся и поставляются в виде *отдельного продукта.*

Встроенные сетевые службы и сетевые оболочки

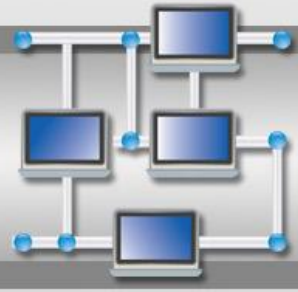


- Примерами сетевых ОС со *встроенными сетевыми службами* являются все современные версии UNIX, NetWare, OS/2, Windows (NT, XP …).
- Сетевые оболочки часто подразделяются на клиентские (MS-DOS с установленной над ней клиентской оболочкой NetWare) и серверные (LAN Server и LAN Manager, а также NetWare for UNIX, File and Print Services for NetWare, Windows 2003 Server).
- Пример отдельных продуктов - сервер удаленного управления WinFrame — продукт компании Citrix — предназначен для работы в среде Windows NT. Он дополняет возможности встроенного в Windows NT сервера удаленного доступа Remote Access Server. Аналогичную службу удаленного доступа для NetWare - программный продукт NetWare Connect.



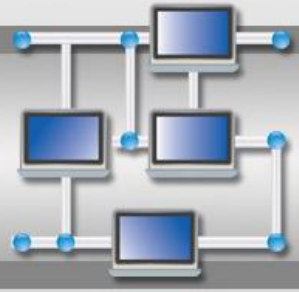
ОДНОРАНГОВЫЕ И СЕРВЕРНЫЕ СЕТЕВЫЕ ОПЕРАЦИОННЫЕ СИСТЕМЫ

Одноранговые и серверные сетевые операционные системы



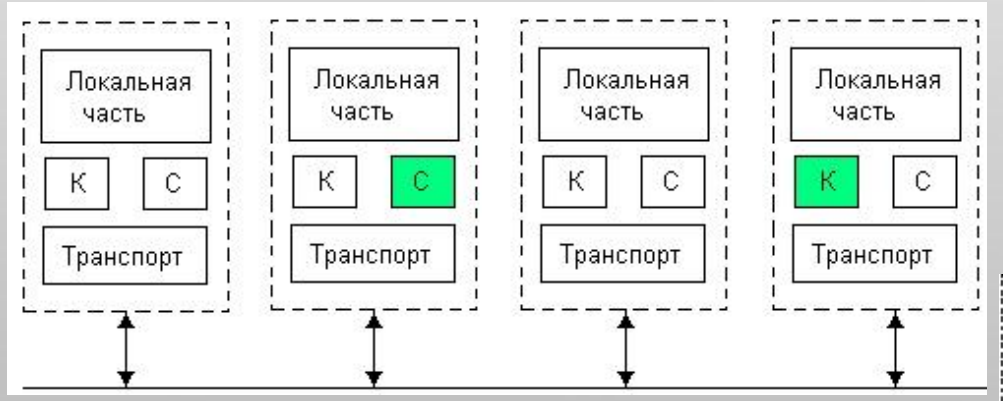
- В зависимости от того, как распределены функции между компьютерами сети, они могут выступать в трех разных ролях:
  - компьютер, занимающийся исключительно обслуживанием запросов других компьютеров, играет роль *выделенного сервера* сети;
  - компьютер, обращающийся с запросами к ресурсам другой машины, исполняет роль *клиентского узла;*
  - компьютер, совмещающий функции клиента и сервера, является *одноранговым узлом.*

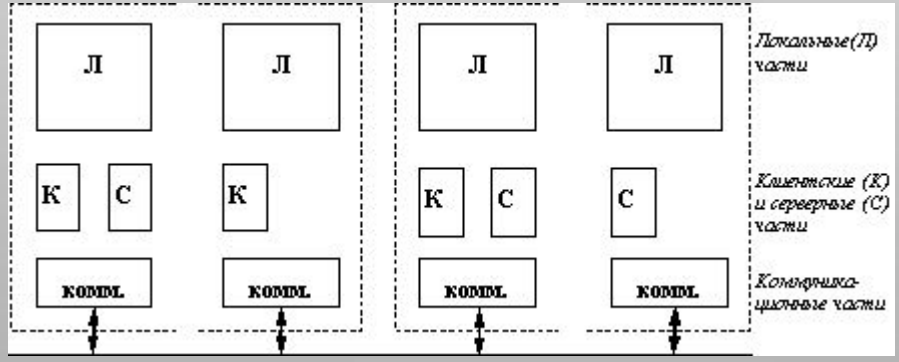
Одноранговые и серверные сетевые операционные системы



- Сеть не может состоять только из клиентских или только из серверных узлов.
- Сеть, оправдывающая свое назначение и обеспечивающая взаимодействие компьютеров, может быть построена по одной из трех следующих схем:
  - сеть на основе одноранговых узлов — *одноранговая сеть;*
  - сеть на основе клиентов и серверов — *сеть с выделенными серверами;*
  - сеть, включающая узлы всех типов, — *гибридная сеть.*

Одноранговые ОС



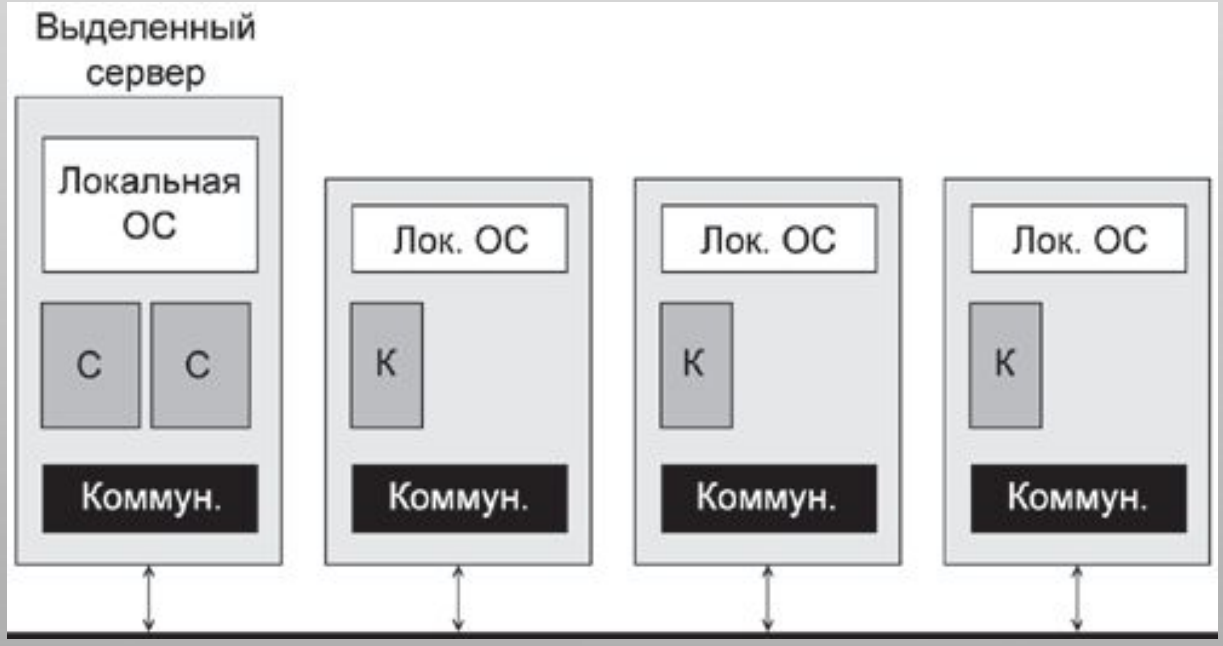


В одноранговых сетях все компьютеры равны в возможностях доступа к ресурсам друг друга.

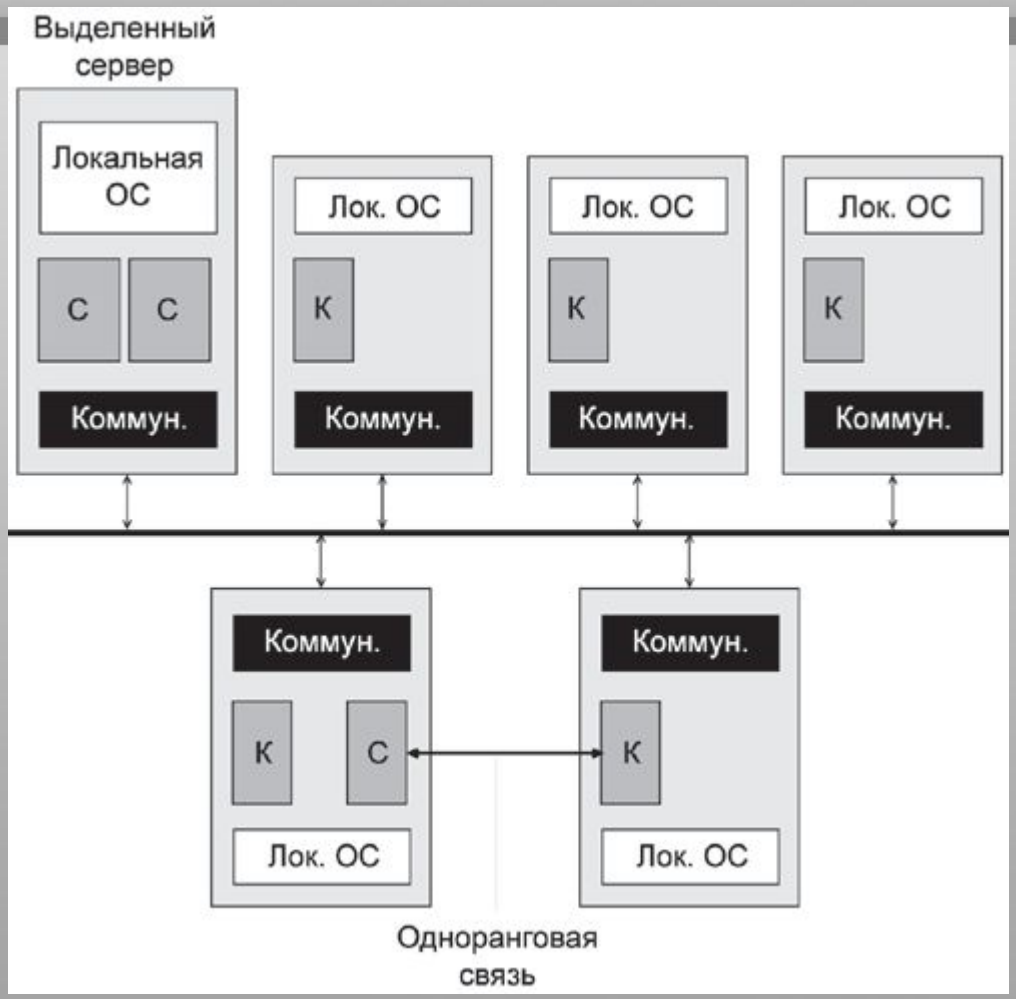
Примеры - LANtastic, Personal Ware, Windows for Workgroups, Windows NT Workstation, Windows 95/98

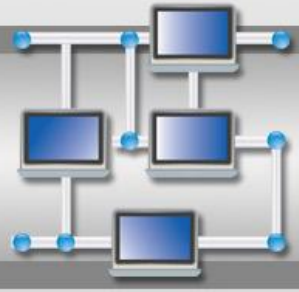
ОС в сетях с выделенными серверами

- В сетях с выделенными серверами используются специальные варианты сетевых ОС, которые называются *серверными ОС.*
- Пользовательские компьютеры в этих сетях работают под управлением *клиентских ОС.*

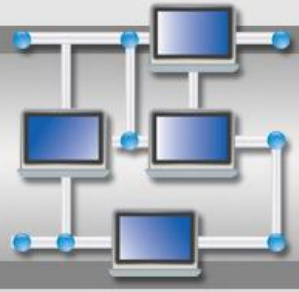


Гибридная сеть



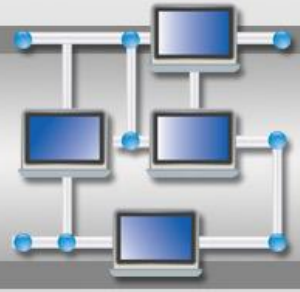


УПРАВЛЕНИЕ ПРОЦЕССАМИ, ПАМЯТЬЮ, ФАЙЛОВОЙ СИСТЕМОЙ



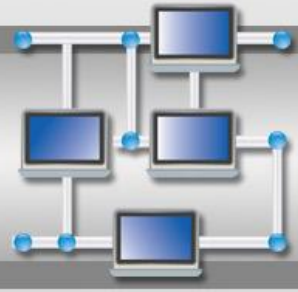
УПРАВЛЕНИЕ ПРОЦЕССАМИ

Процесс



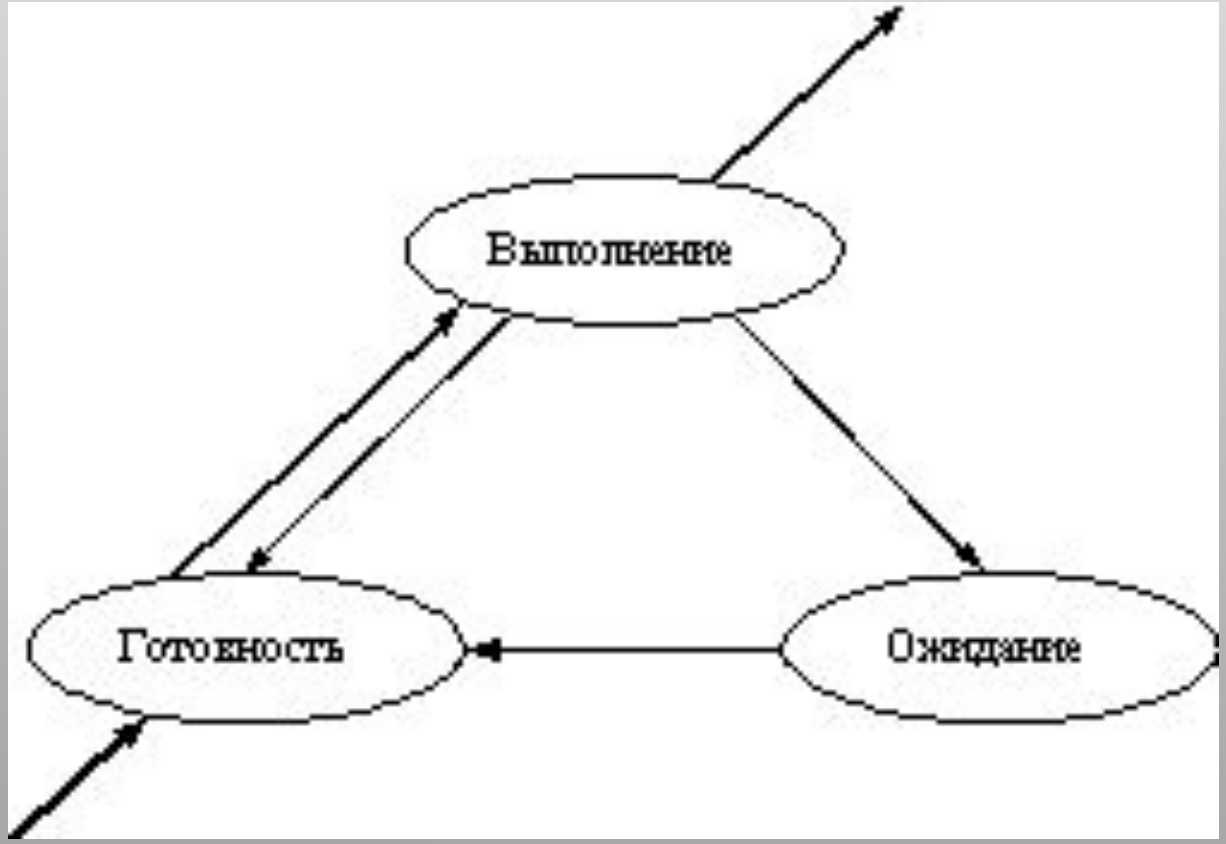
- *Процесс* (или по-другому, задача) - абстракция, описывающая выполняющуюся программу. Для операционной системы процесс представляет собой единицу работы, заявку на потребление системных ресурсов.
- Подсистема управления процессами планирует выполнение процессов, то есть распределяет процессорное время между несколькими одновременно существующими в системе процессами, а также занимается созданием и уничтожением процессов, обеспечивает процессы необходимыми системными ресурсами, поддерживает взаимодействие между процессами.

Состояние процессов

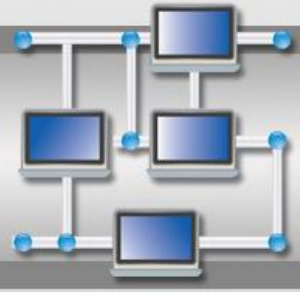


- В многозадачной (многопроцессной) системе процесс может находиться в одном из трех основных состояний:
  - ВЫПОЛНЕНИЕ - активное состояние процесса, во время которого процесс обладает всеми необходимыми ресурсами и непосредственно выполняется процессором;
  - ОЖИДАНИЕ - пассивное состояние процесса, процесс заблокирован, он не может выполняться по своим внутренним причинам, он ждет осуществления некоторого события, например, завершения операции ввода-вывода, получения сообщения от другого процесса, освобождения какого-либо необходимого ему ресурса;
  - ГОТОВНОСТЬ - также пассивное состояние процесса, но в этом случае процесс заблокирован в связи с внешними по отношению к нему обстоятельствами: процесс имеет все требуемые для него ресурсы, он готов выполняться, однако процессор занят выполнением другого процесса.

Граф состояний процесса в многозадачной среде

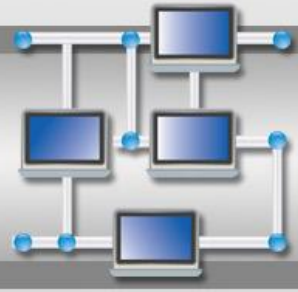


Контекст процесса



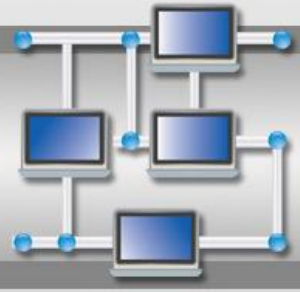
- На протяжении существования процесса его выполнение может быть многократно прервано и продолжено.
- Для того, чтобы возобновить выполнение процесса, необходимо восстановить состояние его операционной среды.
- Состояние операционной среды отображается состоянием регистров и программного счетчика, режимом работы процессора, указателями на открытые файлы, информацией о незавершенных операциях ввода-вывода, кодами ошибок выполняемых данным процессом системных вызовов и т.д.

Дескриптор процесса



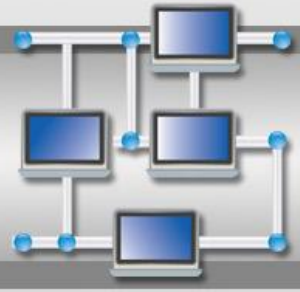
- Кроме этого, операционной системе для реализации планирования процессов требуется дополнительная информация:
  - идентификатор процесса,
  - состояние процесса,
  - данные о степени привилегированности процесса,
  - место нахождения кодового сегмента и другая информация.
- Дескриптор процесса по сравнению с контекстом содержит более оперативную информацию, которая должна быть легко доступна подсистеме планирования процессов.
- Контекст процесса содержит менее актуальную информацию и используется операционной системой только после того, как принято решение о возобновлении прерванного процесса.

Алгоритмы планирования процессов



- Планирование процессов включает в себя решение следующих задач:
  - определение момента времени для смены выполняемого процесса;
  - выбор процесса на выполнение из очереди готовых процессов;
  - переключение контекстов "старого" и "нового" процессов.

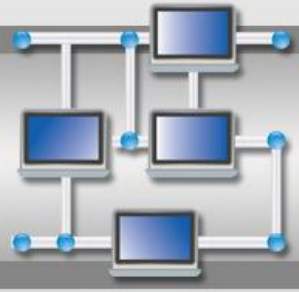
Вытесняющие алгоритмы планирования



- *вытесняющая многозадачность* - это такой способ, при котором решение о переключении процессора с выполнения одного процесса на выполнение другого процесса принимается планировщиком операционной системы, а не самой активной задачей.

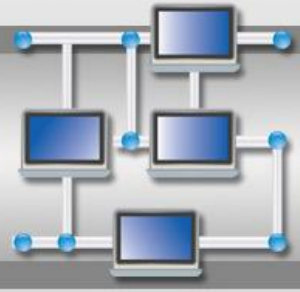
Невытесняющие алгоритмы планирования

- *невытесняющая многозадачность* - это способ планирования процессов, при котором активный процесс выполняется до тех пор, пока он сам, по собственной инициативе, не отдаст управление планировщику операционной системы для того, чтобы тот выбрал из очереди другой, готовый к выполнению процесс.



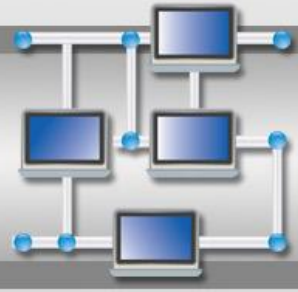
УПРАВЛЕНИЕ ПАМЯТЬЮ

Управление памятью



- Под *памятью (memory)* подразумевается оперативная память компьютера.
- В отличие от памяти жесткого диска, которую называют *внешней памятью (storage),* оперативной памяти для сохранения информации требуется постоянное электропитание.

Управление памятью



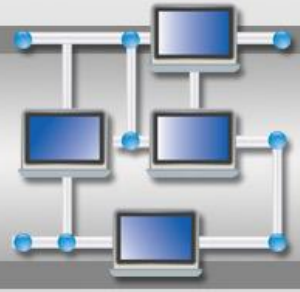
- Память является важнейшим ресурсом, требующим тщательного управления со стороны мультипрограммной операционной системы.
- Распределению подлежит вся оперативная память, не занятая операционной системой.
- Функциями ОС по управлению памятью являются:
  - отслеживание свободной и занятой памяти,
  - выделение памяти процессам и освобождение памяти при завершении процессов,
  - вытеснение процессов из оперативной памяти на диск, когда размеры основной памяти не достаточны для размещения в ней всех процессов, и возвращение их в оперативную память, когда в ней освобождается место,
  - а также настройка адресов программы на конкретную область физической памяти.

Типы адресов

Методы распределения памяти

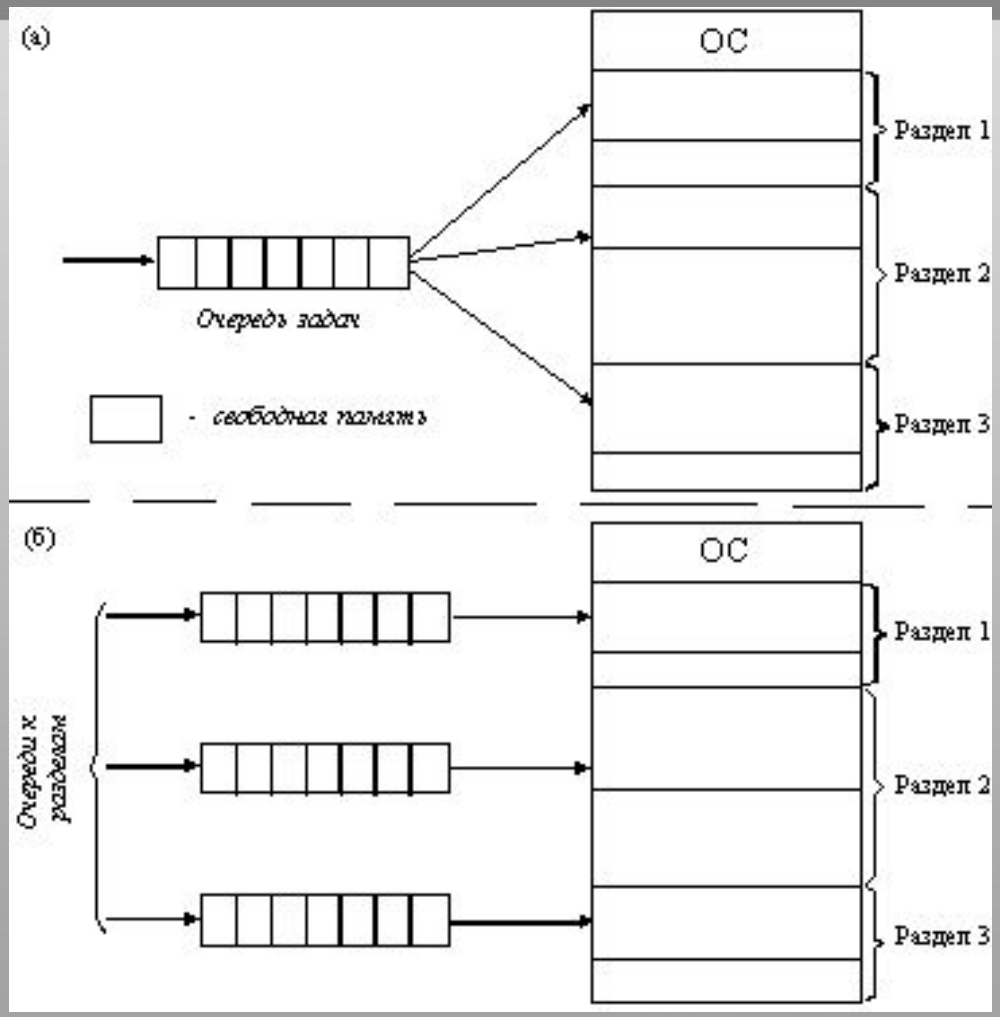
МЕТОДЫ РАСПРЕДЕЛЕНИЯ ПАМЯТИ БЕЗ ИСПОЛЬЗОВАНИЯ ДИСКОВОГО ПРОСТРАНСТВА

Распределение памяти фиксированными разделами

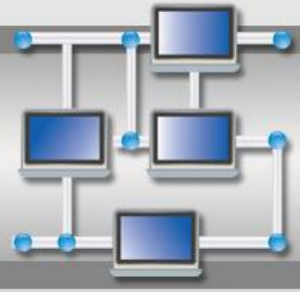


- Самым простым способом управления оперативной памятью является разделение ее на несколько разделов фиксированной величины. Э
- то может быть выполнено вручную оператором во время старта системы или во время ее генерации.
- Очередная задача, поступившая на выполнение, помещается либо в общую очередь, либо в очередь к некоторому разделу.

Распределение памяти фиксированными разделами: а - с общей очередью; б - с отдельными очередями

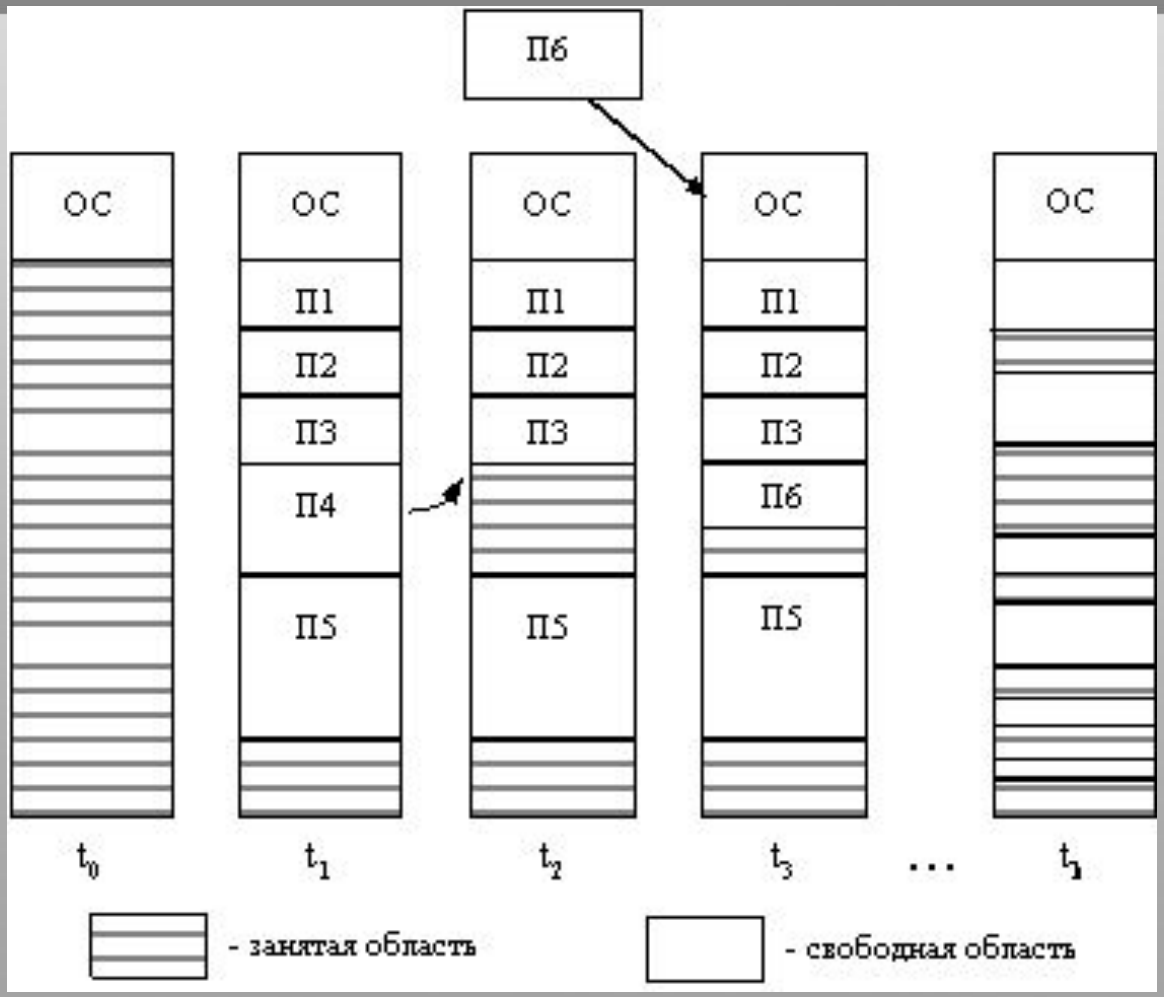


Распределение памяти разделами переменной величины

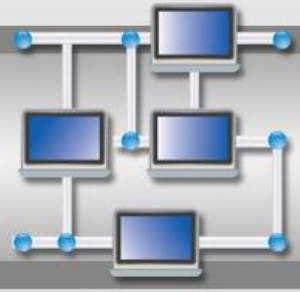


- В этом случае память машины не делится заранее на разделы.
- Сначала вся память свободна.
- Каждой вновь поступающей задаче выделяется необходимая ей память.
- Если достаточный объем памяти отсутствует, то задача не принимается на выполнение и стоит в очереди.
- После завершения задачи память освобождается, и на это место может быть загружена другая задача.
- Таким образом, в произвольный момент времени оперативная память представляет собой случайную последовательность занятых и свободных участков (разделов) произвольного размера.

Распределение памяти динамическими разделами

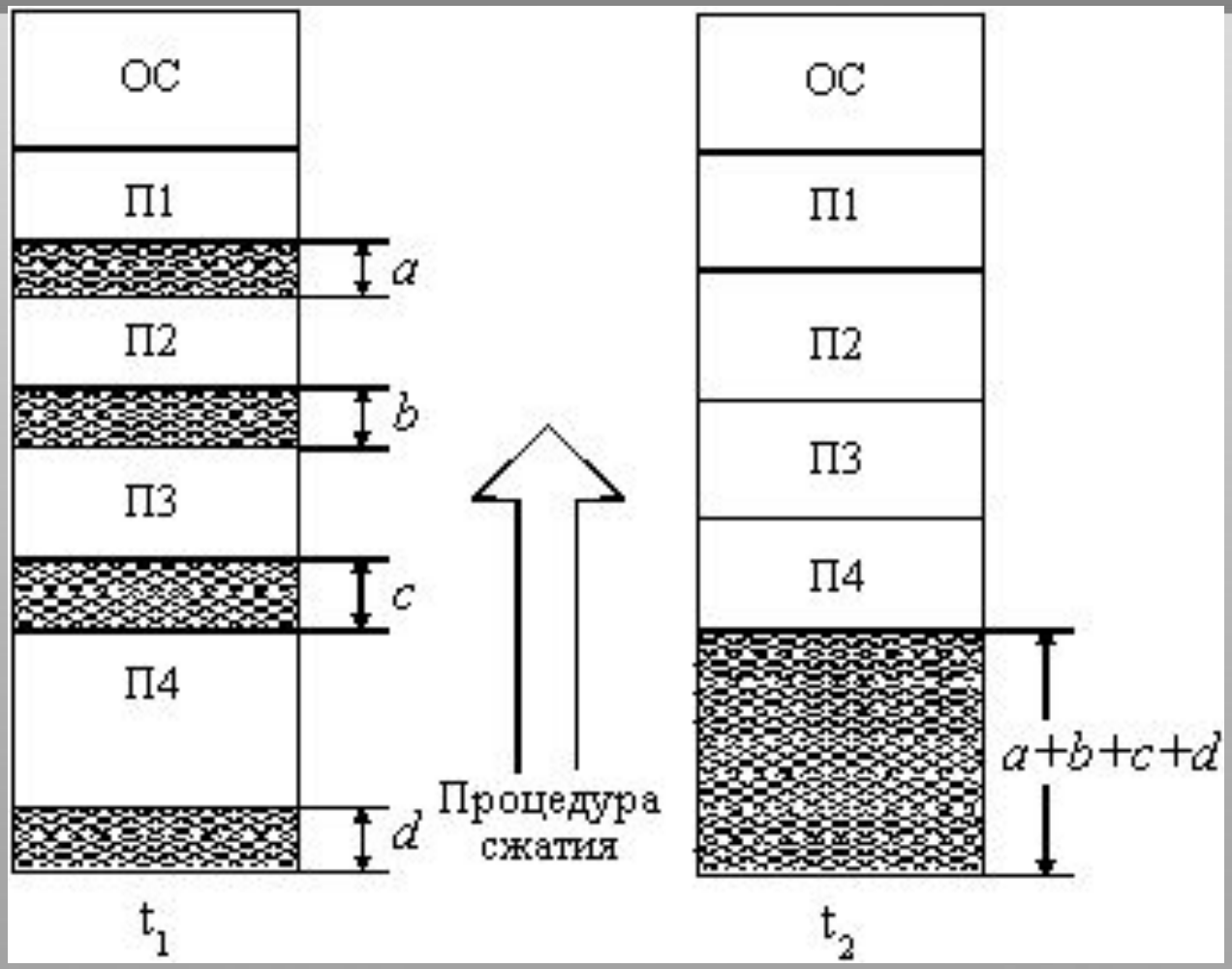


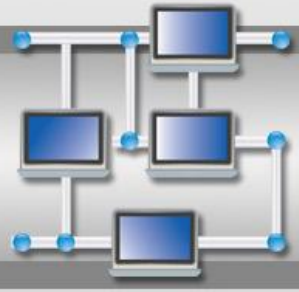
Перемещаемые разделы



- Одним из методов борьбы с фрагментацией является перемещение всех занятых участков в сторону старших либо в сторону младших адресов, так, чтобы вся свободная память образовывала единую свободную область.
- В дополнение к функциям, которые выполняет ОС при распределении памяти переменными разделами, в данном случае она должна еще время от времени копировать содержимое разделов из одного места памяти в другое, корректируя таблицы свободных и занятых областей.
- Эта процедура называется "сжатием". Сжатие может выполняться либо при каждом завершении задачи, либо только тогда, когда для вновь поступившей задачи нет свободного раздела достаточного размера.

Распределение памяти перемещаемыми разделами



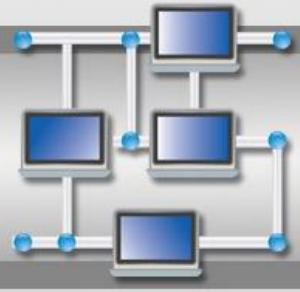


МЕТОДЫ РАСПРЕДЕЛЕНИЯ ПАМЯТИ С ИСПОЛЬЗОВАНИЕМ ДИСКОВОГО ПРОСТРАНСТВА

МИОЭС

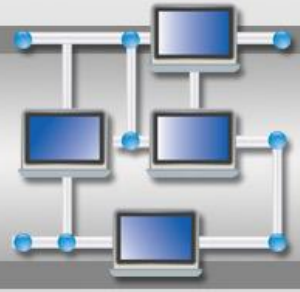
Компьютерные сети

Понятие виртуальной памяти



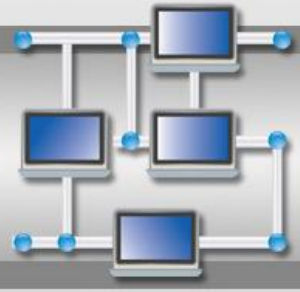
- Уже достаточно давно пользователи столкнулись с проблемой размещения в памяти программ, размер которых превышал имеющуюся в наличии свободную память.
- Решением было разбиение программы на части, называемые *оверлеями*.
- 0-ой оверлей начинал выполняться первым. Когда он заканчивал свое выполнение, он вызывал другой оверлей.
- Все оверлеи хранились на диске и перемещались между памятью и диском средствами операционной системы.
- Однако разбиение программы на части и планирование их загрузки в оперативную память должен был осуществлять программист.

Понятие виртуальной памяти



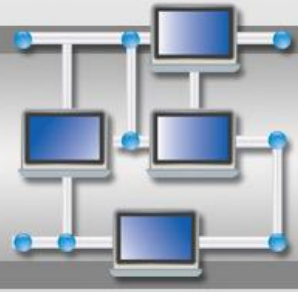
- Развитие методов организации вычислительного процесса в этом направлении привело к появлению метода, известного под названием *виртуальная память*.
- Виртуальным называется ресурс, который пользователю или пользовательской программе представляется обладающим свойствами, которыми он в действительности не обладает.
- Так, например, пользователю может быть предоставлена виртуальная оперативная память, размер которой превосходит всю имеющуюся в системе реальную оперативную память.
- Пользователь пишет программы так, как будто в его распоряжении имеется однородная оперативная память большого объема, но в действительности все данные, используемые программой, хранятся на одном или нескольких разнородных запоминающих устройствах, обычно на дисках, и при необходимости частями отображаются в реальную память.

Виртуальная память



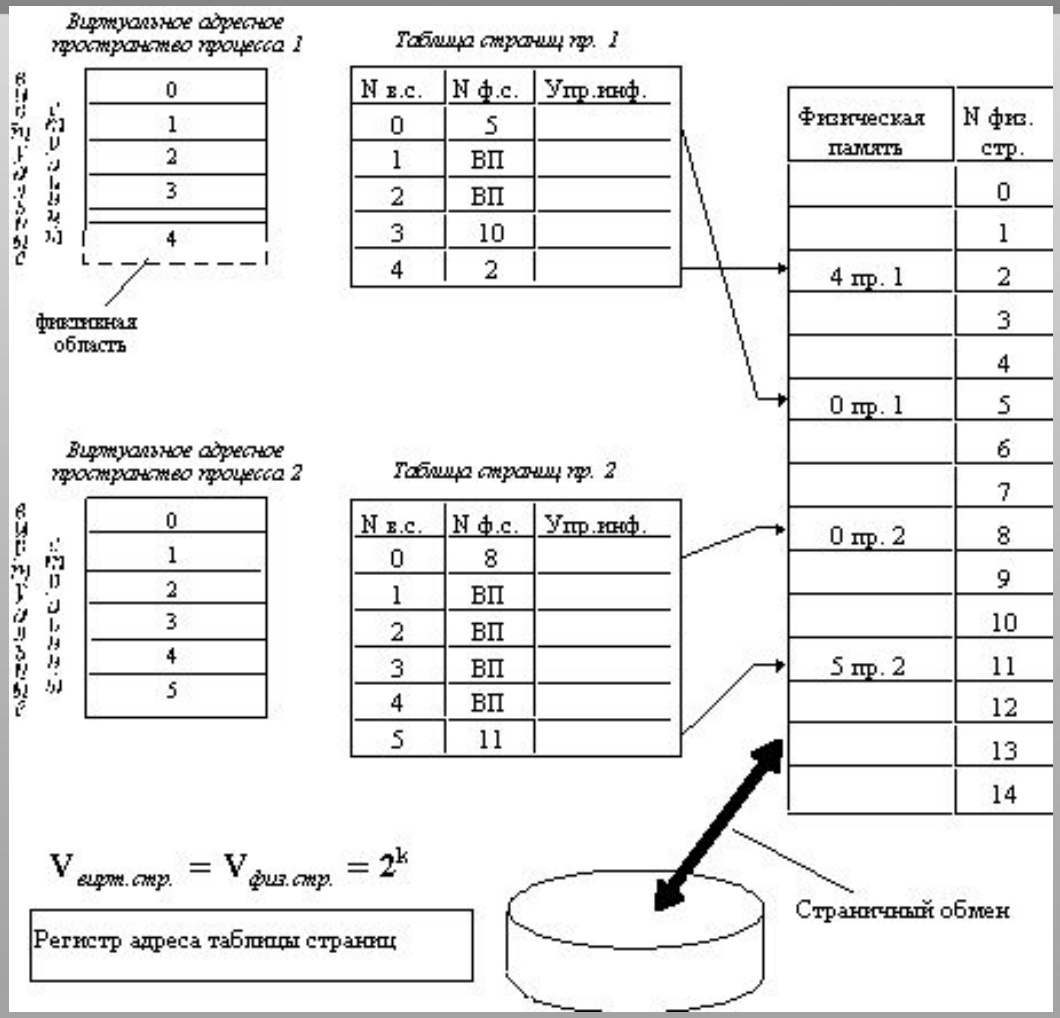
- Таким образом, виртуальная память - это совокупность программно-аппаратных средств, позволяющих пользователям писать программы, размер которых превосходит имеющуюся оперативную память; для этого виртуальная память решает следующие задачи:
  - размещает данные в запоминающих устройствах разного типа, например, часть программы в оперативной памяти, а часть на диске;
  - перемещает по мере необходимости данные между запоминающими устройствами разного типа, например, подгружает нужную часть программы с диска в оперативную память;
  - преобразует виртуальные адреса в физические.
- Все эти действия выполняются *автоматически*, без участия программиста, то есть механизм виртуальной памяти является прозрачным по отношению к пользователю.

Страничное распределение

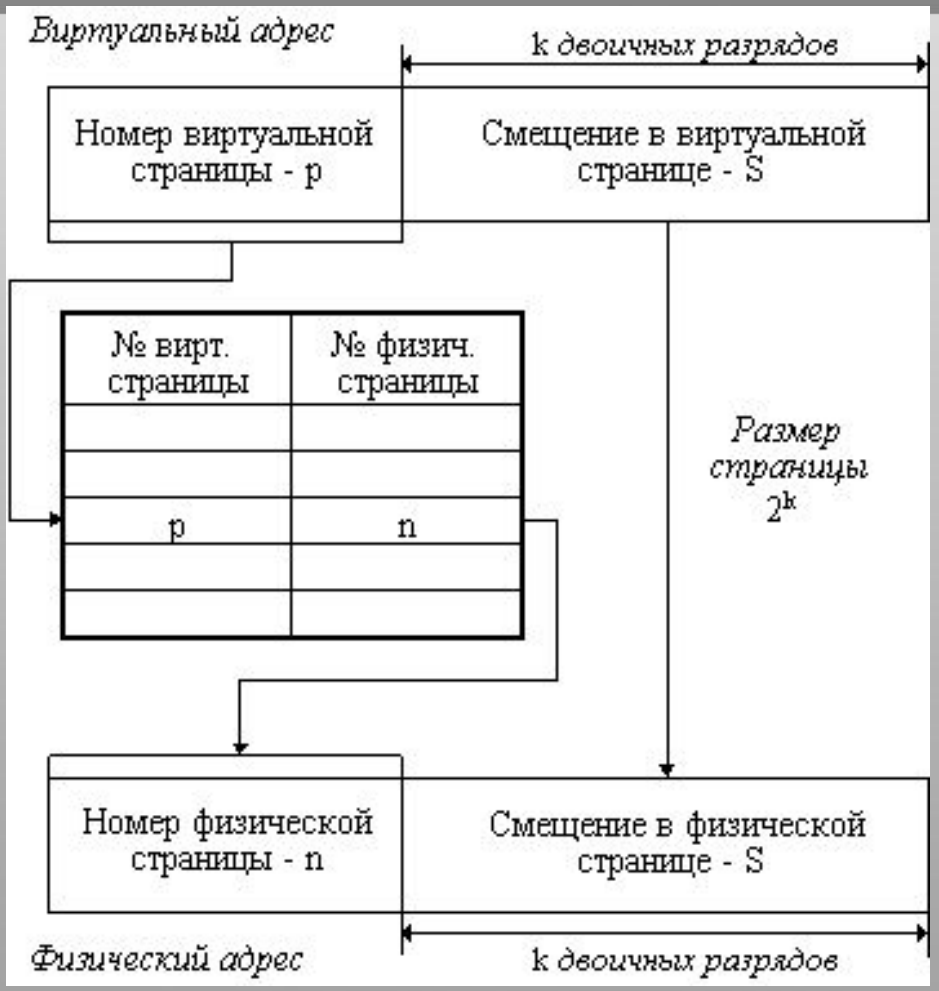


- Виртуальное адресное пространство каждого процесса делится на части одинакового, фиксированного для данной системы размера, называемые виртуальными страницами.
- В общем случае размер виртуального адресного пространства не является кратным размеру страницы, поэтому последняя страница каждого процесса дополняется фиктивной областью.
- Вся оперативная память машины также делится на части такого же размера, называемые физическими страницами (или блоками).
- Размер страницы обычно выбирается равным степени двойки: 512, 1024 и т.д., это позволяет упростить механизм преобразования адресов.

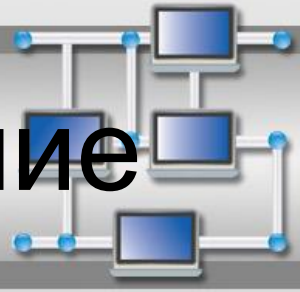
Страничное распределение памяти



Механизм преобразования виртуального адреса в физический при страничной организации

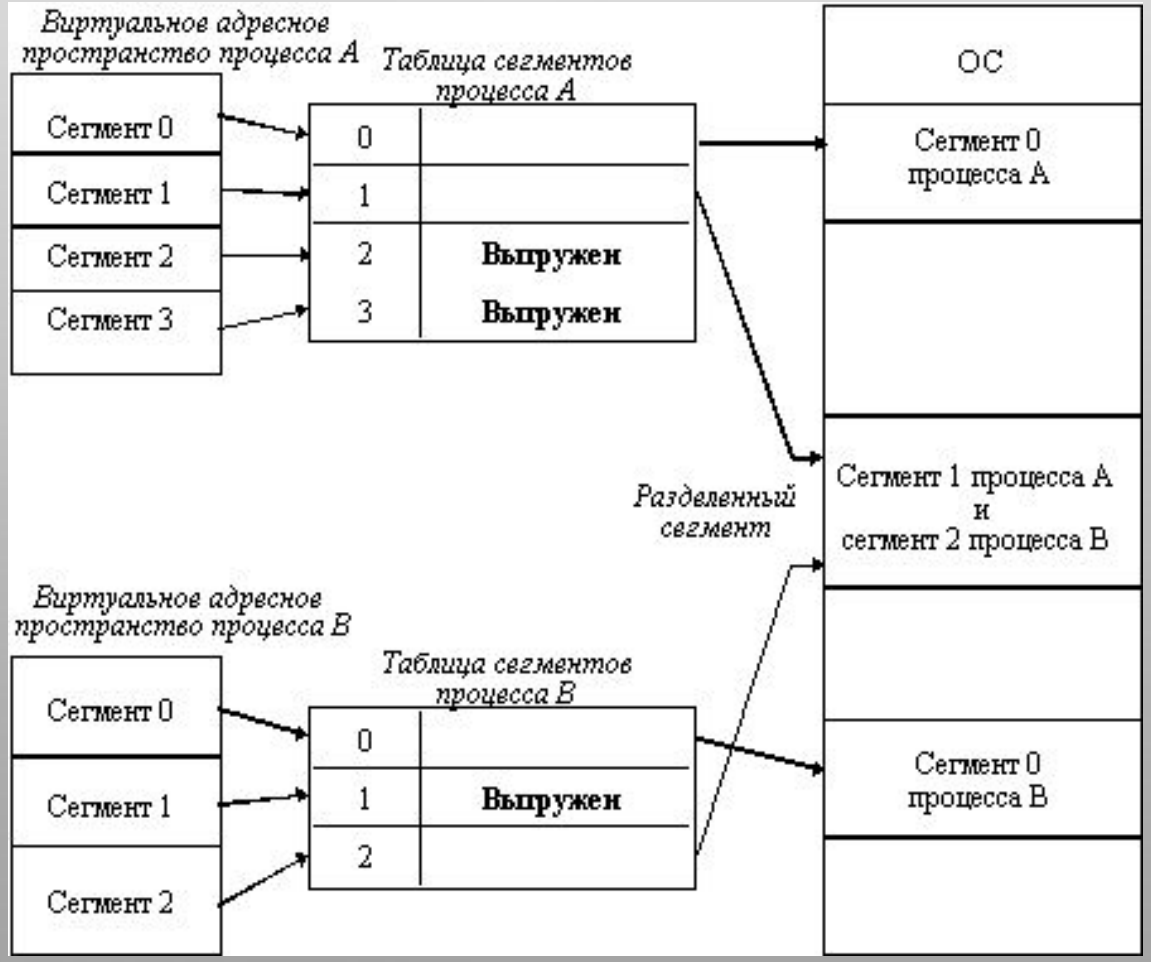


Сегментное распределение

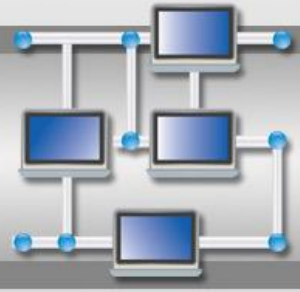


- При страничной организации виртуальное адресное пространство процесса делится механически на равные части.
- Это не позволяет дифференцировать способы доступа к разным частям программы (сегментам), а это свойство часто бывает очень полезным.
- Например, можно запретить обращаться с операциями записи и чтения в кодовый сегмент программы, а для сегмента данных разрешить только чтение.
- Кроме того, разбиение программы на "осмысленные" части делает принципиально возможным разделение одного сегмента несколькими процессами.
- Например, если два процесса используют одну и ту же математическую подпрограмму, то в оперативную память может быть загружена только одна копия этой подпрограммы.

Распределение памяти сегментами

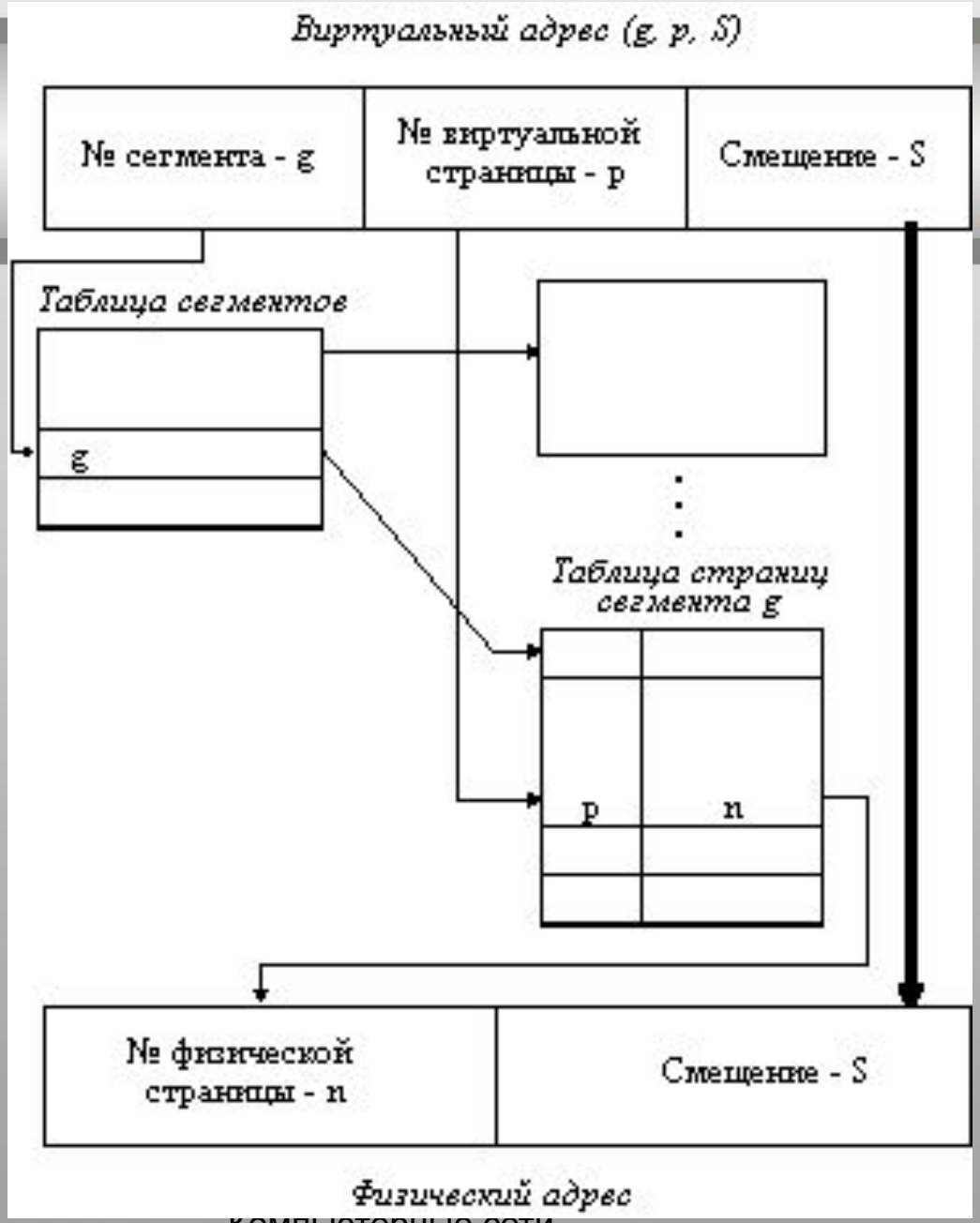


Странично-сегментное распределение

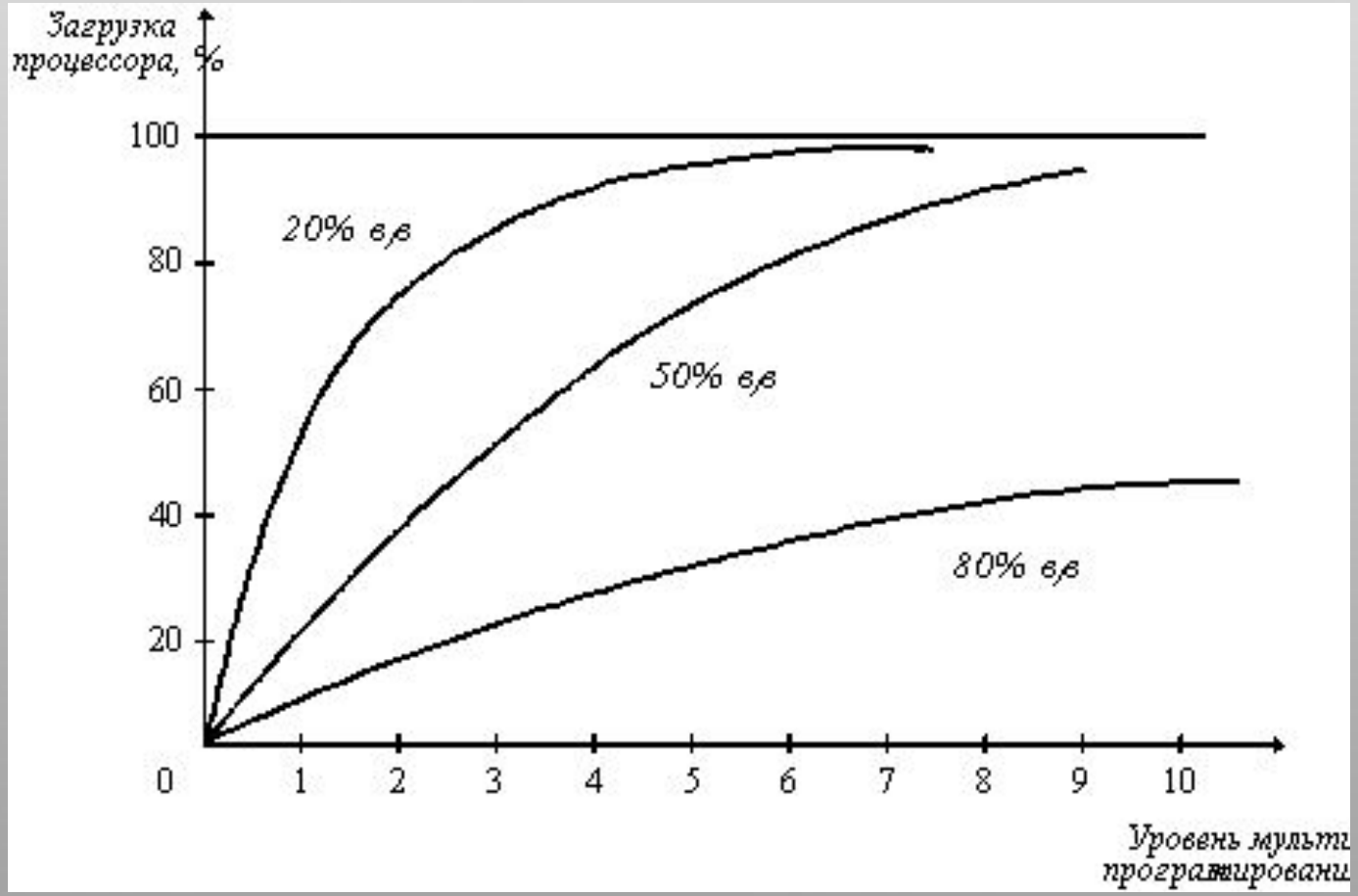


- Как видно из названия, данный метод представляет собой комбинацию страничного и сегментного распределения памяти и, вследствие этого, сочетает в себе достоинства обоих подходов.
- Виртуальное пространство процесса делится на сегменты, а каждый сегмент в свою очередь делится на виртуальные страницы, которые нумеруются в пределах сегмента.
- Оперативная память делится на физические страницы.
- Загрузка процесса выполняется операционной системой постранично, при этом часть страниц размещается в оперативной памяти, а часть на диске.
- Для каждого сегмента создается своя таблица страниц, структура которой полностью совпадает со структурой таблицы страниц, используемой при страничном распределении.
- Для каждого процесса создается таблица сегментов, в которой указываются адреса таблиц страниц для всех сегментов данного процесса.
- Адрес таблицы сегментов загружается в специальный регистр процессора, когда активизируется соответствующий процесс.

Схема преобразования виртуального адреса в физический для сегментно-страничной организации памяти

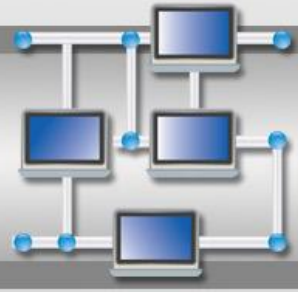


Свопинг



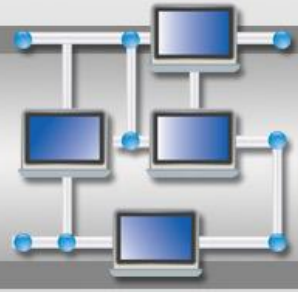
Зависимость загрузки процессора от числа задач и интенсивности ввода-вывода

Свопинг



- Из рисунка видно, что для загрузки процессора на 90% достаточно всего трех счетных задач.
- Однако для того, чтобы обеспечить такую же загрузку интерактивными задачами, выполняющими интенсивный ввод-вывод, потребуются десятки таких задач. Необходимым условием для выполнения задачи является загрузка ее в оперативную память, объем которой ограничен.
- В этих условиях был предложен метод организации вычислительного процесса, называемый свопингом.
- В соответствии с этим методом некоторые процессы (обычно находящиеся в состоянии ожидания) временно выгружаются на диск.

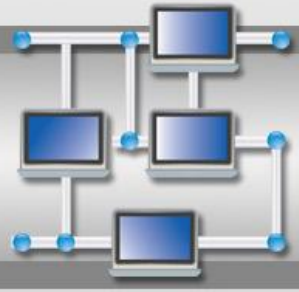
Свопинг

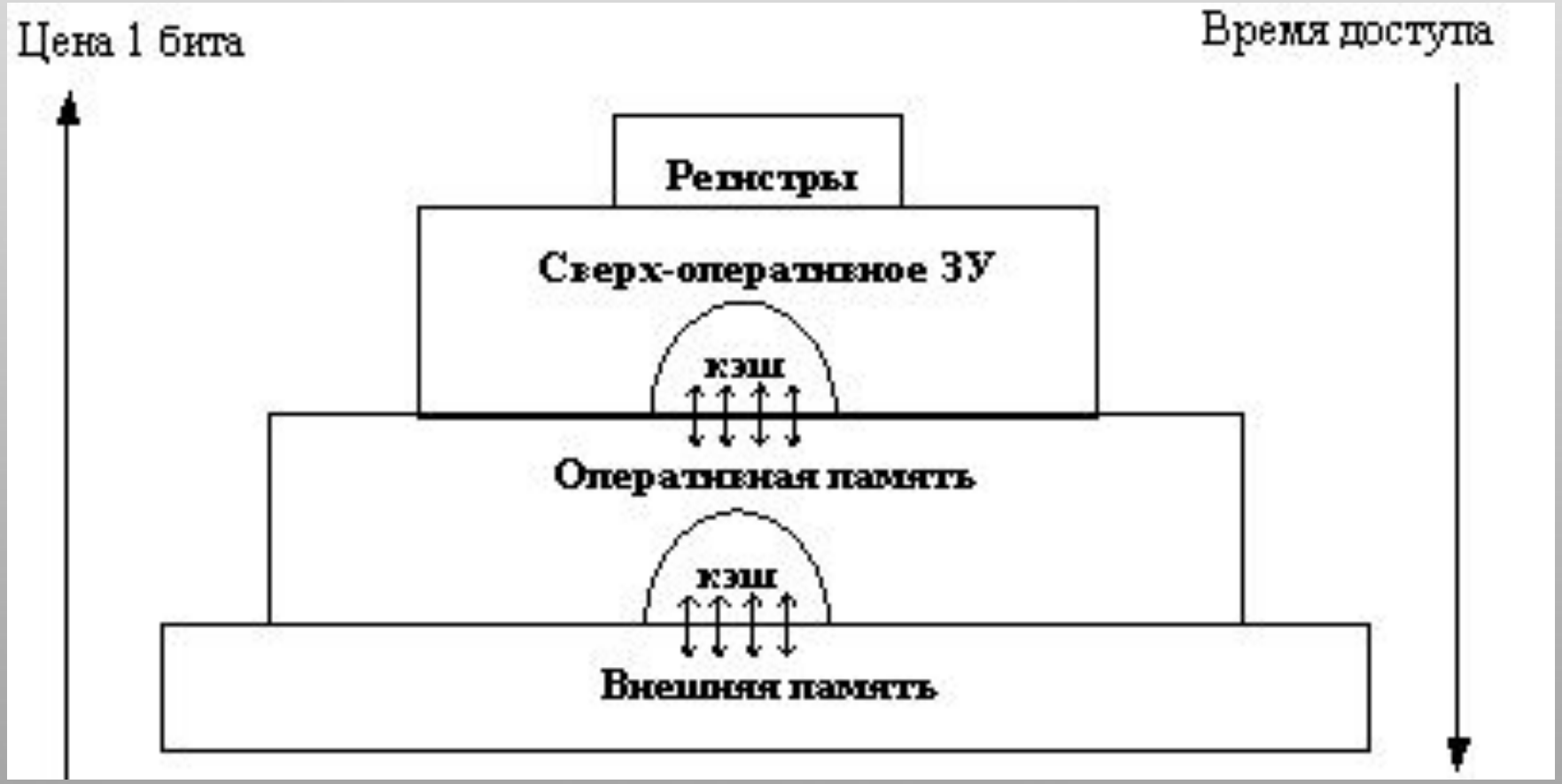


- Планировщик операционной системы не исключает их из своего рассмотрения, и при наступлении условий активизации некоторого процесса, находящегося в области свопинга на диске, этот процесс перемещается в оперативную память.
- Если свободного места в оперативной памяти не хватает, то выгружается другой процесс.
- При свопинге, в отличие от рассмотренных ранее методов реализации виртуальной памяти, процесс перемещается между памятью и диском целиком, то есть в течение некоторого времени процесс может полностью отсутствовать в оперативной памяти.
- Существуют различные алгоритмы выбора процессов на загрузку и выгрузку, а также различные способы выделения оперативной и дисковой памяти загружаемому процессу.

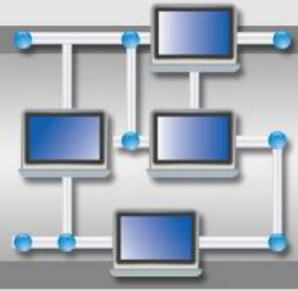
ИЕРАРХИЯ ЗАПОМИНАЮЩИХ УСТРОЙСТВ. ПРИНЦИП КЭШИРОВАНИЯ ДАННЫХ

Иерархия ЗУ



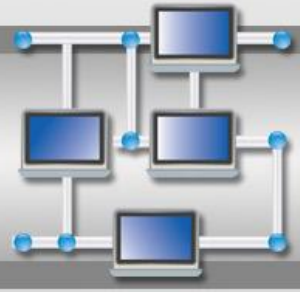


Кэш-память

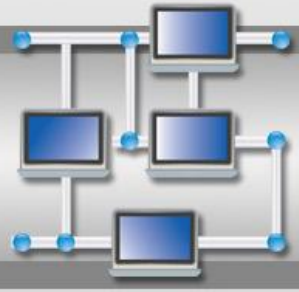


- это способ организации совместного функционирования двух типов запоминающих устройств, отличающихся временем доступа и стоимостью хранения данных, который позволяет уменьшить среднее время доступа к данным за счет динамического копирования в "быстрое" ЗУ наиболее часто используемой информации из "медленного" ЗУ.

Кэш-память

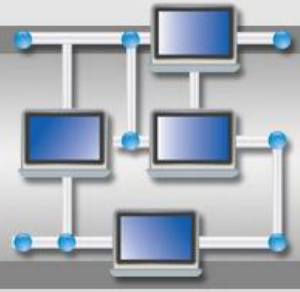


- Кэш-памятью часто называют не только способ организации работы двух типов запоминающих устройств, но и одно из устройств - "быстрое" ЗУ.
- Оно стоит дороже и, как правило, имеет сравнительно небольшой объем.
- Важно, что механизм кэш-памяти является прозрачным для пользователя, который не должен сообщать никакой информации об интенсивности использования данных и не должен никак участвовать в перемещении данных из ЗУ одного типа в ЗУ другого типа, все это делается автоматически системными средствами.



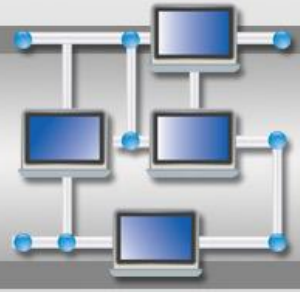
УПРАВЛЕНИЕ ФАЙЛОВОЙ СИСТЕМОЙ

Файловая система



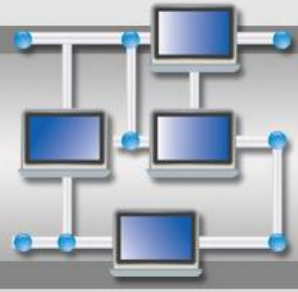
- это часть операционной системы, назначение которой состоит в том, чтобы обеспечить пользователю удобный интерфейс при работе с данными, хранящимися на диске, и обеспечить совместное использование файлов несколькими пользователями и процессами.

Файловая система



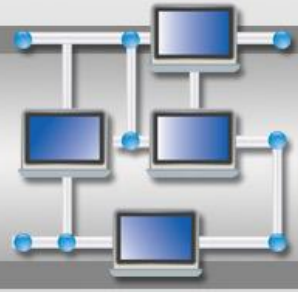
- В широком смысле понятие "файловая система" включает:
  - совокупность всех файлов на диске,
  - наборы структур данных, используемых для управления файлами, такие, например, как каталоги файлов, дескрипторы файлов, таблицы распределения свободного и занятого пространства на диске,
  - комплекс системных программных средств, реализующих управление файлами, в частности: создание, уничтожение, чтение, запись, именование, поиск и другие операции над файлами.

Имена файлов



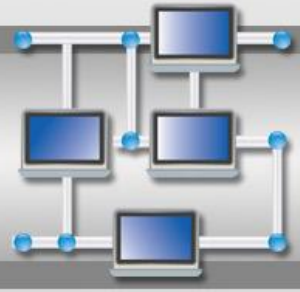
- Файлы идентифицируются именами.
- Пользователи дают файлам символьные имена, при этом учитываются ограничения ОС как на используемые символы, так и на длину имени.
- В файловой системе FAT длина имен ограничивается известной схемой 8.3 (8 символов - собственно имя, 3 символа - расширение имени).
- В ОС UNIX System V имя не может содержать более 14 символов.
- Однако пользователю гораздо удобнее работать с длинными именами, поскольку они позволяют дать файлу действительно мнемоническое название, по которому даже через достаточно большой промежуток времени можно будет вспомнить, что содержит этот файл.
- Поэтому современные файловые системы, как правило, поддерживают длинные символьные имена файлов.
- Например, Windows NT в своей новой файловой системе NTFS устанавливает, что имя файла может содержать до 255 символов, не считая завершающего нулевого символа.

Типы файлов



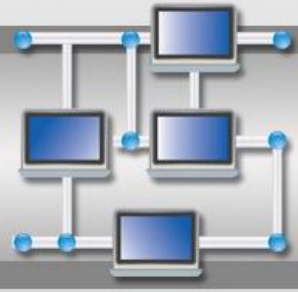
- Обычные файлы в свою очередь подразделяются на текстовые и двоичные.
  - Текстовые файлы состоят из строк символов, представленных в ASCII-коде.
  - Двоичные файлы не используют ASCII-коды, они часто имеют сложную внутреннюю структуру, например, объектный код программы или архивный файл. Все операционные системы должны уметь распознавать хотя бы один тип файлов - их собственные исполняемые файлы.
- *Специальные файлы* - это файлы, ассоциированные с устройствами ввода-вывода, которые позволяют пользователю выполнять операции ввода-вывода, используя обычные команды записи в файл или чтения из файла.
- *Каталог* - это, с одной стороны, группа файлов, объединенных пользователем исходя из некоторых соображений (например, файлы, содержащие программы игр, или файлы, составляющие один программный пакет), а с другой стороны - это файл, содержащий системную информацию о группе файлов, его составляющих.

Физическая организация и адрес файла

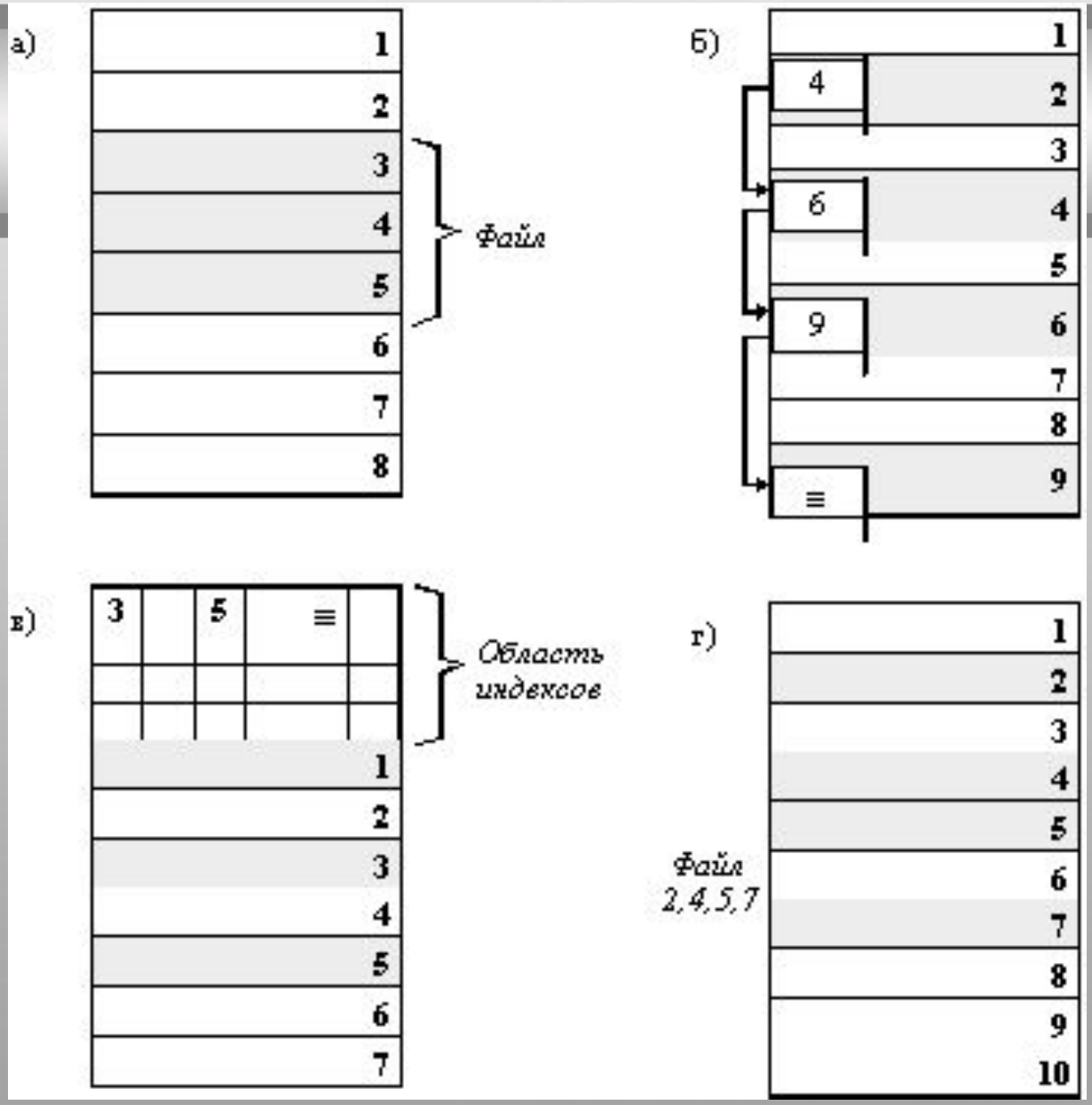


- Физическая организация файла описывает правила расположения файла на устройстве внешней памяти, в частности на диске.
- Файл состоит из физических записей - блоков.
- Блок - наименьшая единица данных, которой внешнее устройство обменивается с оперативной памятью.

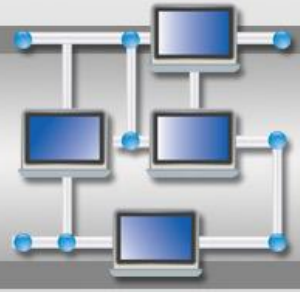
Физическая организация и адрес файла



- Непрерывное размещение - простейший вариант физической организации, при котором файлу предоставляется последовательность блоков диска, образующих единый сплошной участок дисковой памяти.
- Следующий способ физической организации - размещение в виде связанного списка блоков дисковой памяти. При таком способе в начале каждого блока содержится указатель на следующий блок.
- Популярным способом, используемым, например, в файловой системе FAT операционной системы MS-DOS, является использование связанного списка индексов. С каждым блоком связывается некоторый элемент - индекс. Индексы располагаются в отдельной области диска (в MS-DOS это таблица FAT). Если некоторый блок распределен некоторому файлу, то индекс этого блока содержит номер следующего блока данного файла.
- Задание физического расположения файла путем простого перечисления номеров блоков, занимаемых этим файлом. ОС UNIX использует вариант данного способа, позволяющий обеспечить фиксированную длину адреса, независимо от размера файла.

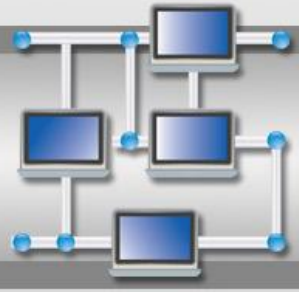

а)
1
2
3
4
5
6
7
8
Файл
б)
1
4
2
3
6
4
5
9
6
7
8
≡
9
в)
3
5
≡
Область индексов
1
2
3
4
5
6
7
г)
Файл 2,4,5,7
1
2
3
4
5
6
7
8
9
10

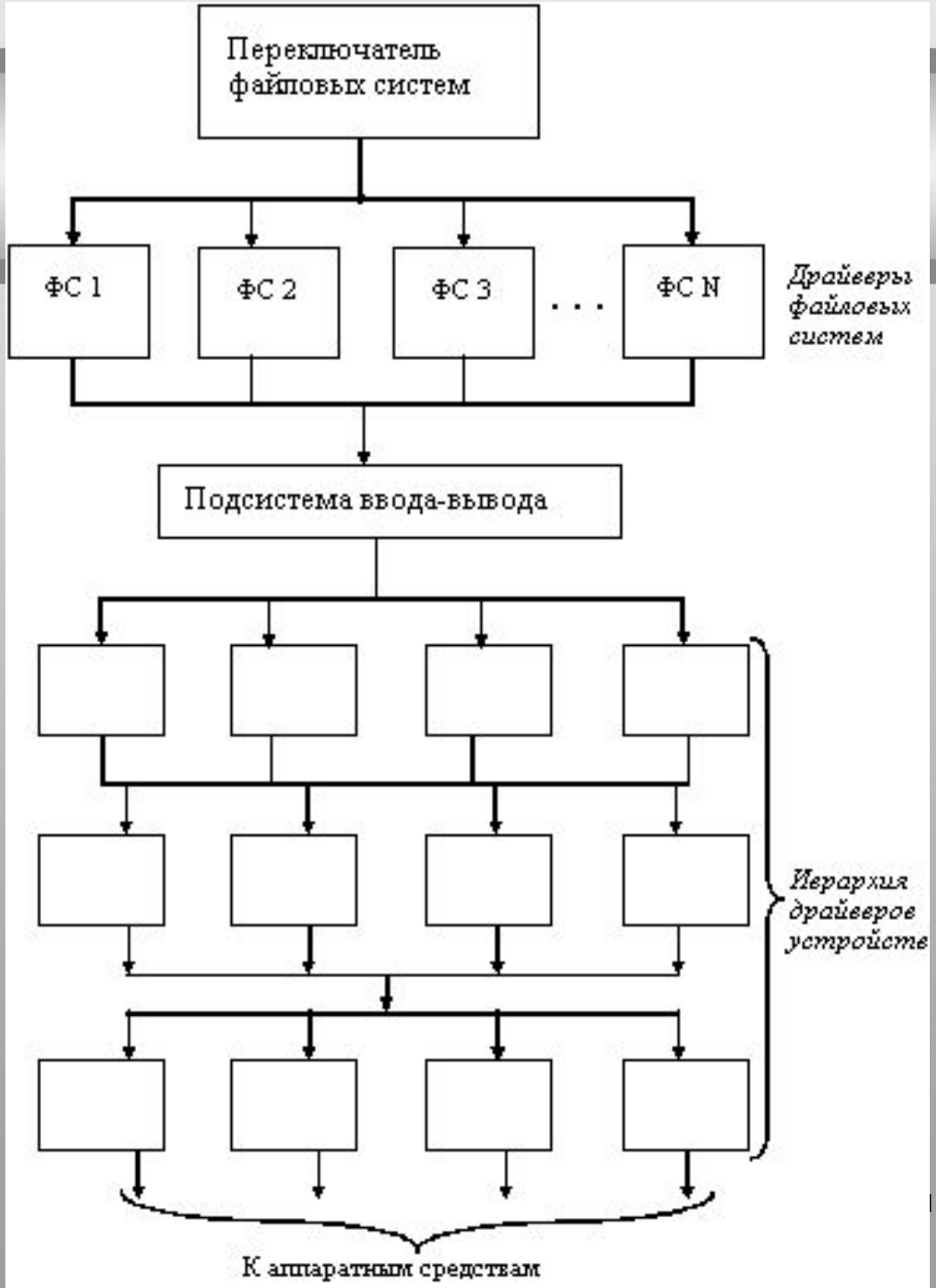
Права доступа к файлу



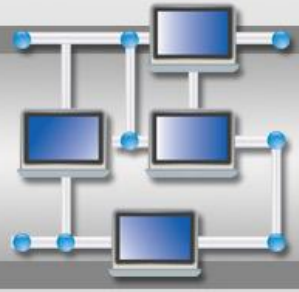
- Определить права доступа к файлу - значит определить для каждого пользователя набор операций, которые он может применить к данному файлу.
- В разных файловых системах может быть определен свой список дифференцируемых операций доступа.
- Различают два основных подхода к определению прав доступа:
  - избирательный доступ, когда для каждого файла и каждого пользователя сам владелец может определить допустимые операции;
  - мандатный подход, когда система наделяет пользователя определенными правами по отношению к каждому разделяемому ресурсу (в данном случае файлу) в зависимости от того, к какой группе пользователь отнесен.

Общая модель файловой системы





Архитектура современной файловой системы

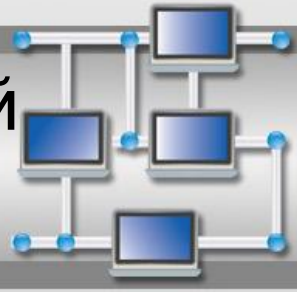


УПРАВЛЕНИЕ РАСПРЕДЕЛЕННЫМИ РЕСУРСАМИ

МИОЭС

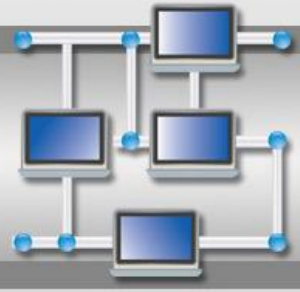
Компьютерные сети

Базовые примитивы передачи сообщений в распределенных системах



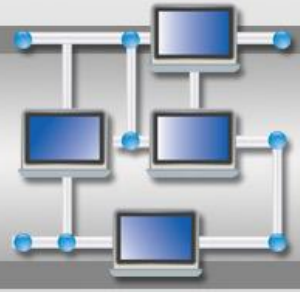
- Единственным по-настоящему важным отличием распределенных систем от централизованных является межпроцессная взаимосвязь.
- В централизованных системах связь между процессами, как правило, предполагает наличие разделяемой памяти. Типичный пример - проблема "поставщик-потребитель", в этом случае один процесс пишет в разделяемый буфер, а другой - читает из него.
- Даже наиболее простая форма синхронизации - семафор - требует, чтобы хотя бы одно слово (переменная самого семафора) было разделяемым.
- В распределенных системах нет какой бы то ни было разделяемой памяти, таким образом вся природа межпроцессных коммуникаций должна быть продумана заново.

Блокирующие примитивы



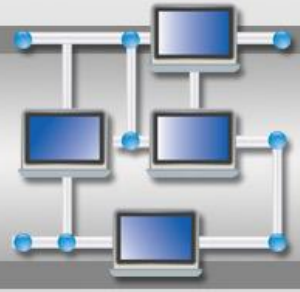
- При использовании блокирующего примитива, процесс, выдавший запрос на его выполнение, приостанавливается до полного завершения примитива.
- Например, вызов примитива ПОЛУЧИТЬ приостанавливает вызывающий процесс до получения сообщения.

Неблокирующие примитивы



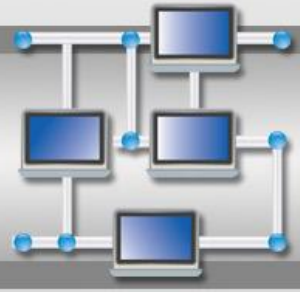
- При использовании неблокирующего примитива управление возвращается вызывающему процессу немедленно, еще до того, как требуемая работа будет выполнена.
- Преимуществом этой схемы является параллельное выполнение вызывающего процесса и процесса передачи сообщения.
- Обычно в ОС имеется один из двух видов примитивов и очень редко - оба

Буферизуемые примитивы



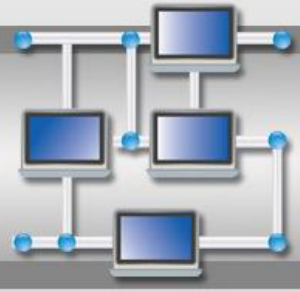
- Процесс, который заинтересован в получении сообщений, обращается к ядру с запросом о создании для него почтового ящика и сообщает адрес, по которому ему могут поступать сетевые пакеты, после чего все сообщения с данным адресом будут помещены в его почтовый ящик.
- Такой способ часто называют буферизуемым примитивом.

Небуферизуемые примитивы



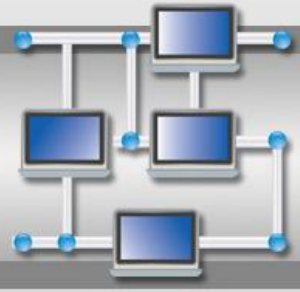
- Вызов ПОЛУЧИТЬ сообщает ядру машины, на которой он выполняется, адрес буфера, в который следует поместить пребывающее для него сообщение.
- Эта схема работает прекрасно при условии, что получатель выполняет вызов ПОЛУЧИТЬ
- раньше, чем отправитель выполняет вызов ПОСЛАТЬ. Вызов ПОЛУЧИТЬ сообщает ядру машины, на которой выполняется, по какому адресу должно поступить ожидаемое сообщение, и в какую область памяти необходимо его поместить.
- Проблема возникает тогда, когда вызов ПОСЛАТЬ сделан раньше вызова ПОЛУЧИТЬ.

Надежные примитивы

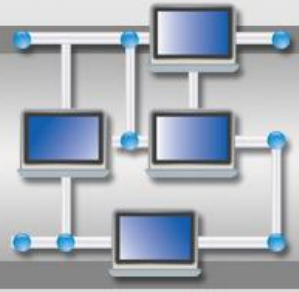


- Ядро принимающей машины посылает квитанцию-подтверждение ядру отправляющей машины на каждое сообщение. Посылающее ядро разблокирует пользовательский процесс только после получения этого подтверждения. Подтверждение передается от ядра к ядру. Ни отправитель, ни получатель его не видят.
- Либо используется ответав качестве подтверждения в тех системах, в которых запрос всегда сопровождается ответом. Отправитель остается заблокированным до получения ответа. Если ответа нет слишком долго, то посылающее ядро может переслать запрос специальной службе предотвращения потери сообщений.

Ненадежные примитивы

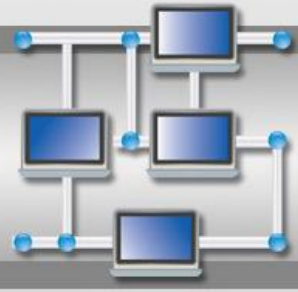


- Система не берет на себя никаких обязательств по поводу доставки сообщений.
- Реализация надежного взаимодействия становится целиком заботой пользователя.



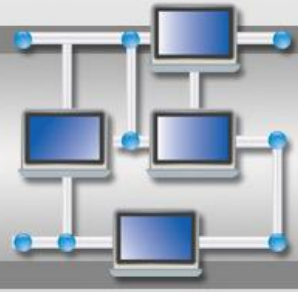
ВЫЗОВ УДАЛЕННЫХ ПРОЦЕДУР (RPC)

Концепция удаленного вызова процедур



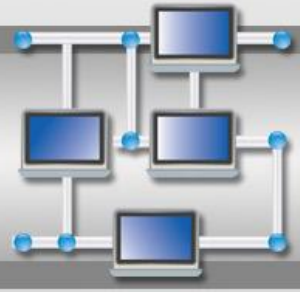
- Идея вызова удаленных процедур *(Remote Procedure Call - RPC)* состоит в расширении хорошо известного и понятного механизма передачи управления и данных внутри программы, выполняющейся на одной машине, на передачу управления и данных через сеть.
- Средства удаленного вызова процедур предназначены для облегчения организации распределенных вычислений.
- Наибольшая эффективность использования RPC достигается в тех приложениях, в которых существует интерактивная связь между удаленными компонентами с небольшим временем ответов и относительно малым количеством передаваемых данных.
- Такие приложения называются RPC-ориентированными.

Концепция удаленного вызова процедур

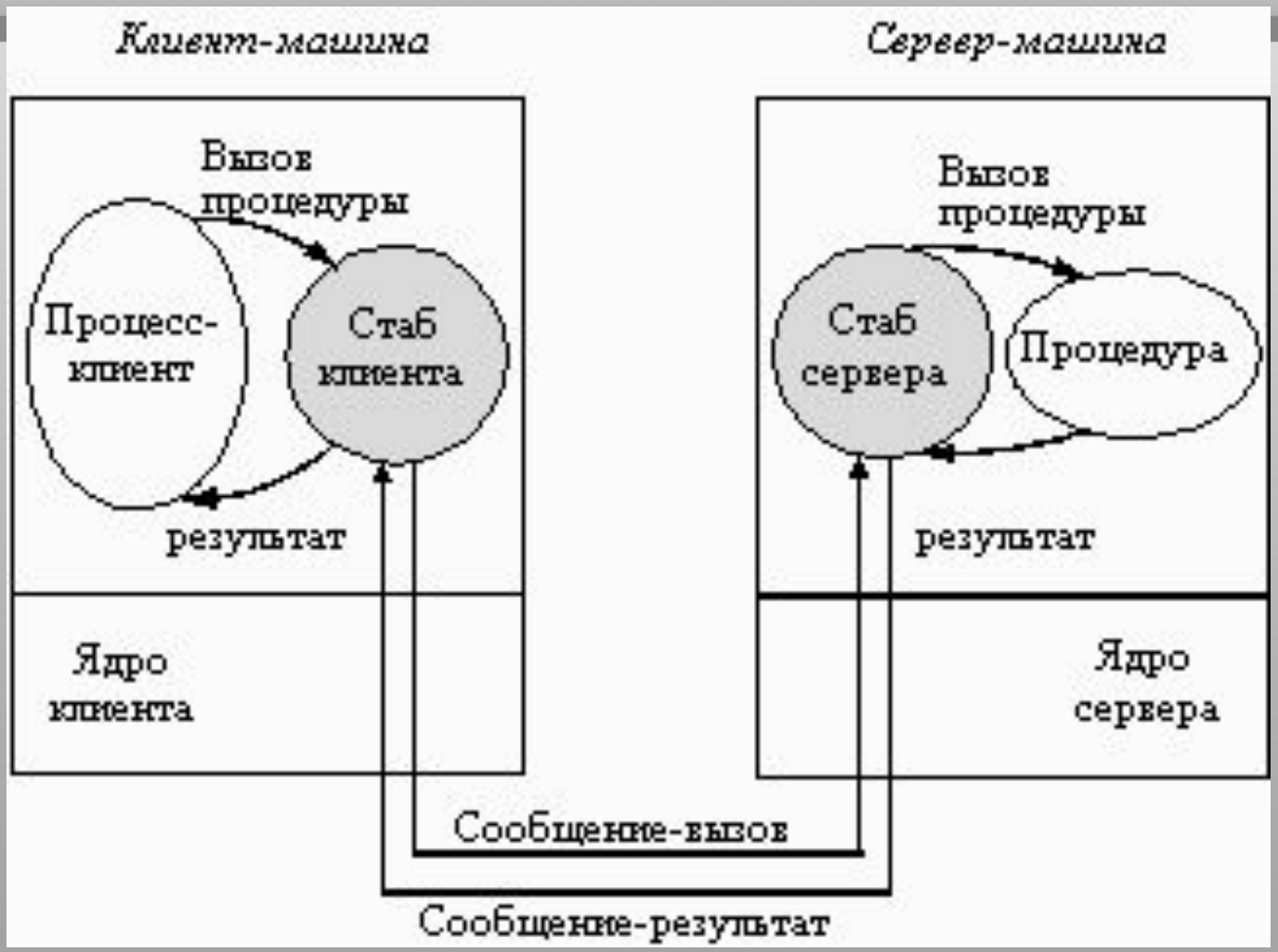


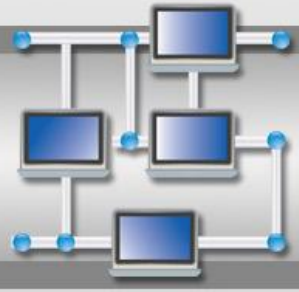
- Характерными чертами вызова локальных процедур являются:
  - Асимметричность, то есть одна из взаимодействующих сторон является инициатором;
  - Синхронность, то есть выполнение вызывающей процедуры при останавливается с момента выдачи запроса и возобновляется только после возврата из вызываемой процедуры.

Прозрачность в RPC



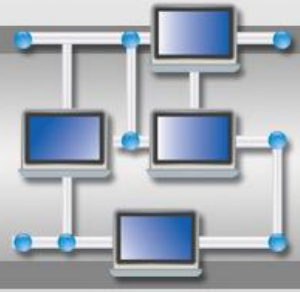
- RPC достигает прозрачности следующим путем.
- Когда вызываемая процедура действительно является удаленной, в библиотеку помещается вместо локальной процедуры другая версия процедуры, называемая клиентским стабом (stub - заглушка).
- Подобно оригинальной процедуре, стаб вызывается с использованием вызывающей последовательности, так же происходит прерывание при обращении к ядру.
- Только в отличие от оригинальной процедуры он не помещает параметры в регистры и не запрашивает у ядра данные, вместо этого он формирует сообщение для отправки ядру удаленной машины.


Клиент-машина
Сервер-машина
Вызов процедуры
Процесс-клиент
Стаб клиента
результат
Ядро клиента
Вызов процедуры
Стаб сервера
Процедура
результат
Ядро сервера
Сообщение-вызов
Сообщение-результат



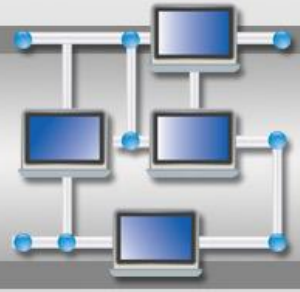
КЭШИРОВАНИЕ ФАЙЛОВ В РАСПРЕДЕЛЕННЫХ СИСТЕМАХ

Кэширование



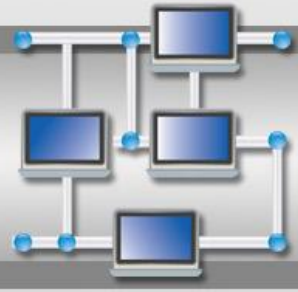
- В системах, состоящих из клиентов и серверов, потенциально имеется четыре различных места для хранения файлов и их частей:
  - диск сервера,
  - память сервера,
  - диск клиента (если имеется) и
  - память клиента.
- Наиболее подходящим местом для хранения всех файлов является диск сервера. Он обычно имеет большую емкость, и файлы становятся доступными всем клиентам.
- Проблемой при использовании диска сервера является производительность. Перед тем, как клиент сможет прочитать файл, файл должен быть переписан с диска сервера в его оперативную память, а затем передан по сети в память клиента.
- Значительное увеличение производительности может быть достигнуто за счет кэширования файлов в памяти сервера. Требуются алгоритмы для определения, какие файлы или их части следует хранить в кэш-памяти.

Кэширование

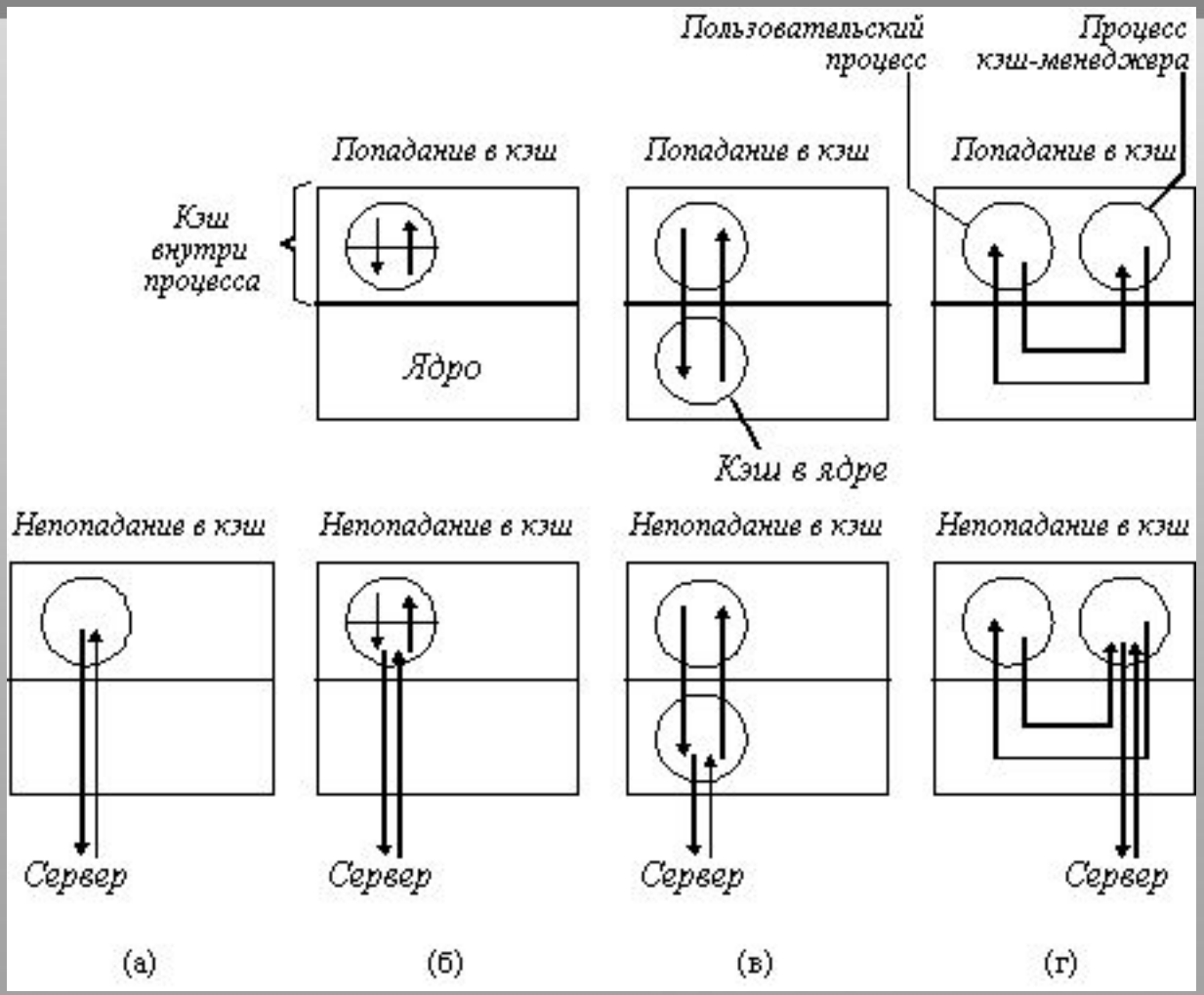


- При выборе алгоритма должны решаться две задачи.
  - Во-первых, какими единицами оперирует кэш. Этими единицами могут быть или дисковые блоки, или целые файлы. Если это целые файлы, то они могут храниться на диске непрерывными областями (по крайней мере в виде больших участков), при этом уменьшается число обменов между памятью и диском а, следовательно, обеспечивается высокая производительность. Кэширование блоков диска позволяет более эффективно использовать память кэша и дисковое пространство.
  - Во-вторых, необходимо определить правило замены данных при заполнении кэш-памяти. Здесь можно использовать любой стандартный алгоритм кэширования, например, алгоритм LRU (least recently used), соответствии с которым вытесняется блок, к которому дольше всего не было обращения.

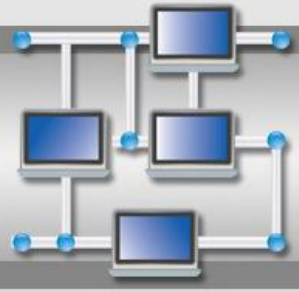
Кэширование



- Кэш-память на сервере легко реализуется и совершенно прозрачна для клиента. Так как сервер может синхронизировать работу памяти и диска, с точки зрения клиентов существует только одна копия каждого файла, так что проблема согласования не возникает.
- Хотя кэширование на сервере исключает обмен с диском при каждом доступе, все еще остается обмен по сети.
- Существует только один путь избавиться от обмена по сети - это кэширование на стороне клиента, которое, однако, порождает много сложностей.

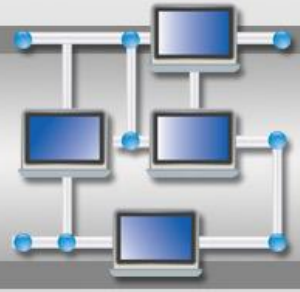


Различные способы выполнения кэша в клиентской памяти
а - без кэширования;
б - кэширование внутри каждого процесса;
в - кэширование в ядре;
г - кэш-менеджер как пользовательский процесс

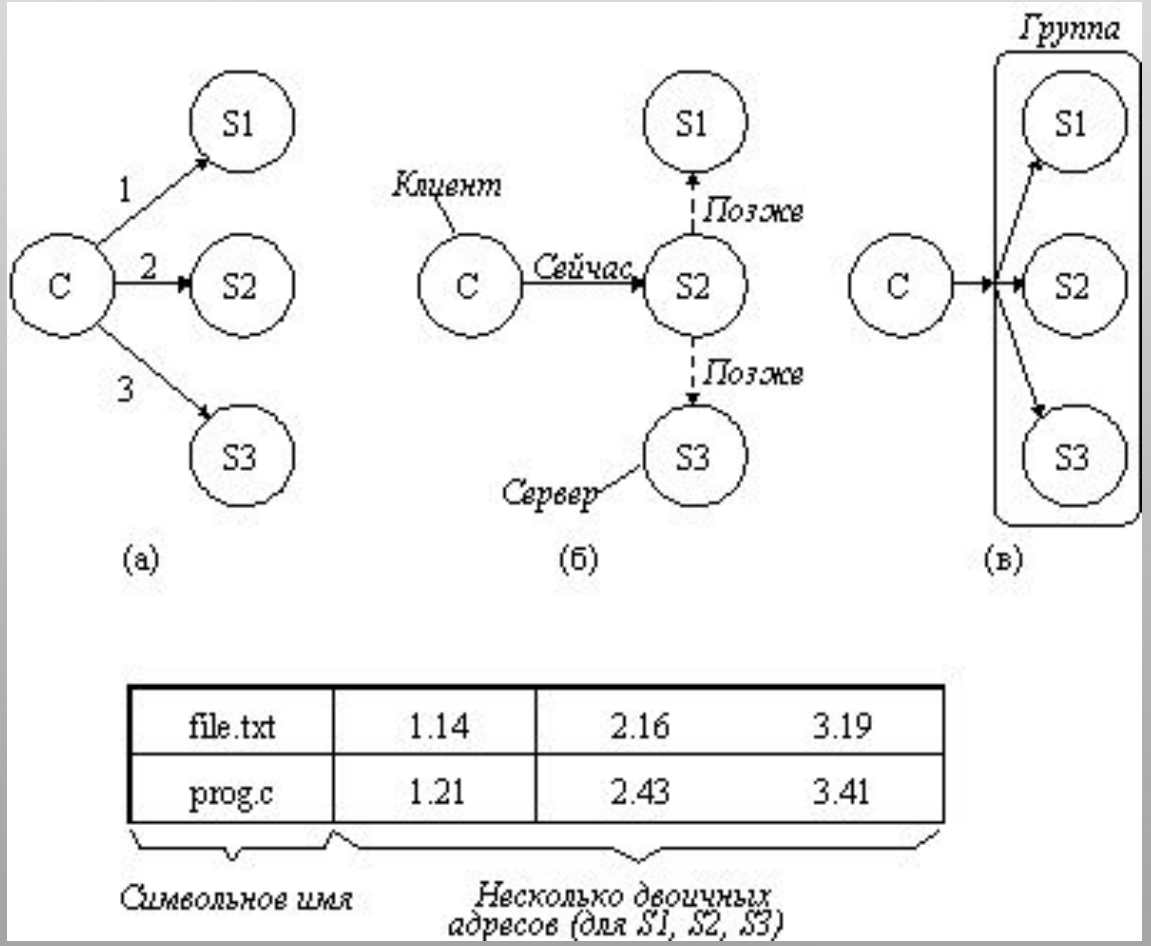


РЕПЛИКАЦИЯ

Репликация

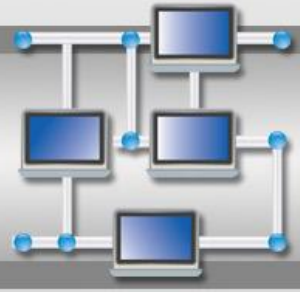


- - это асинхронный перенос изменений данных исходной файловой системы в файловые системы, принадлежащие различным узлам распределенной файловой системы.
- Другими словами, система оперирует несколькими копиями файлов, причем каждая копия находится на отдельном файловом сервере.
- Имеется несколько причин для предоставления этого сервиса, главными из которых являются:
  - Увеличение надежности за счет наличия независимых копий каждого файла на разных файл-серверах.
  - Распределение нагрузки между несколькими серверами.

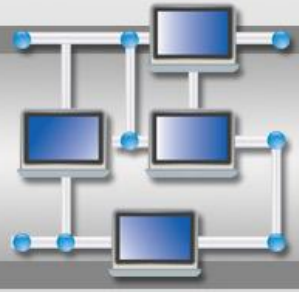


а) Точная репликация файла;
б) Ленивая репликация файла;
в) Репликация файла, использующая группу

Виды репликаций

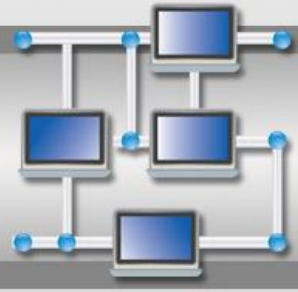


- При использовании первого способа программист сам управляет всем процессом репликации. Когда процесс создает файл, он делает это на одном определенном сервере. Затем, если пожелает, он может сделать дополнительные копии на других серверах.
- На рисунке показан альтернативный подход - ленивая репликация. Здесь создается только одна копия каждого файла на некотором сервере. Позже сервер сам автоматически выполнит репликации на другие серверы без участия программиста.
- Метод, использующий групповые связи. В этом методе все системные вызовы ЗАПИСАТЬ передаются одновременно на все серверы, таким образом копии создаются одновременно с созданием оригинала.



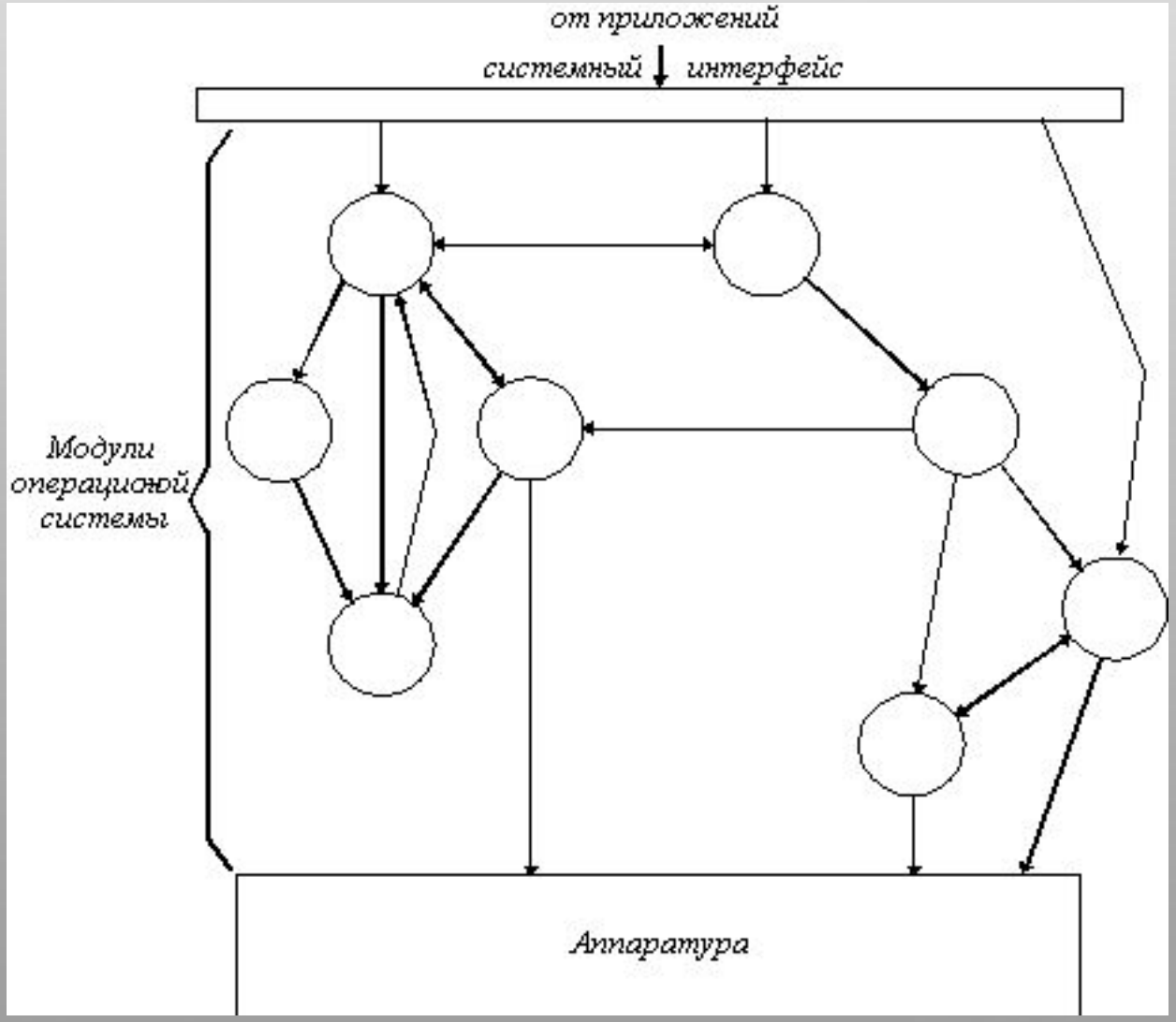
ТРЕБОВАНИЯ К СОВРЕМЕННЫМ ОС, ПЕРЕДОВЫЕ ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ ОС, КРИТЕРИИ ВЫБОРА СЕТЕВЫХ ОС

Требования, предъявляемые к ОС



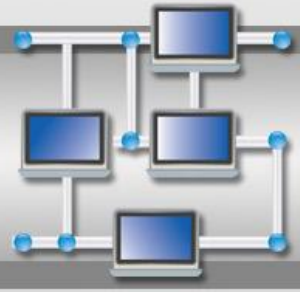
- *Расширяемость*. Код должен быть написан таким образом, чтобы можно было легко внести дополнения и изменения, если это потребуется, и не нарушить целостность системы.
- *Переносимость*. Код должен легко переноситься с процессора одного типа на процессор другого типа и с аппаратной платформы (которая включает наряду с типом процессора и способ организации всей аппаратуры компьютера) одного типа на аппаратную платформу другого типа.
- *Надежность и отказоустойчивость*. Система должна быть защищена как от внутренних, так и от внешних ошибок, сбоев и отказов. Ее действия должны быть всегда предсказуемыми, а приложения не должны быть в состоянии наносить вред ОС.
- *Совместимость*. ОС должна иметь средства для выполнения прикладных программ, написанных для других операционных систем. Кроме того, пользовательский интерфейс должен быть совместим с существующими системами и стандартами.
- *Безопасность*. ОС должна обладать средствами защиты ресурсов одних пользователей от других.
- *Производительность*. Система должна обладать настолько хорошим быстродействием и временем реакции, насколько это позволяет аппаратная платформа.

Монолитная ОС

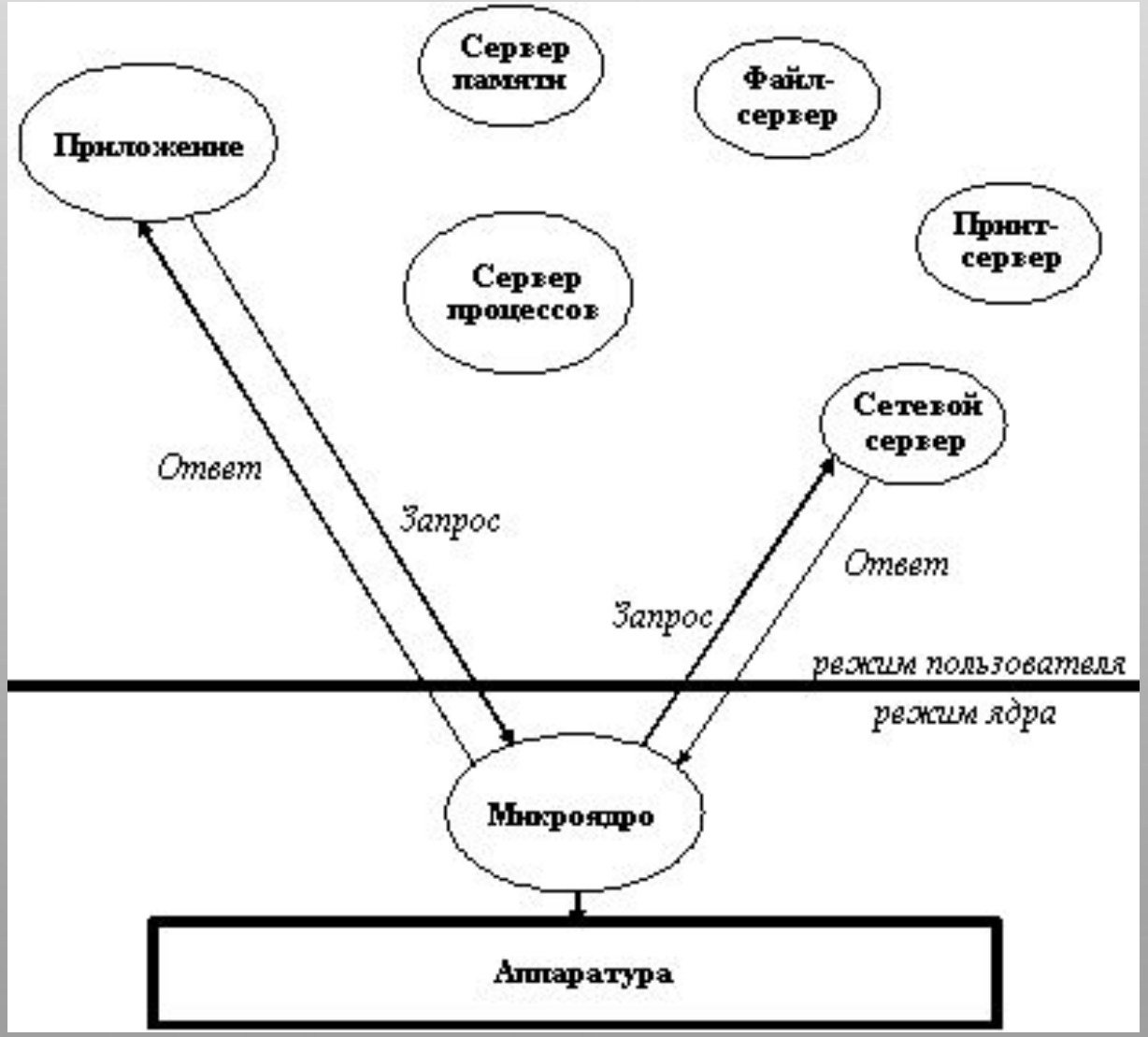


ОС написана как набор процедур, каждая из которых может вызывать другие, когда ей это нужно

Многоуровневые системы

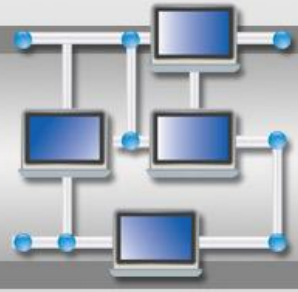


- Обобщением предыдущего подхода является организация ОС как иерархии уровней.
- Уровни образуются группами функций операционной системы - файловая система, управление процессами и устройствами и т.п.
- Каждый уровень может взаимодействовать только со своим непосредственным соседом - выше- или нижележащим уровнем.
- Прикладные программы или модули самой операционной системы передают запросы вверх и вниз по этим уровням.



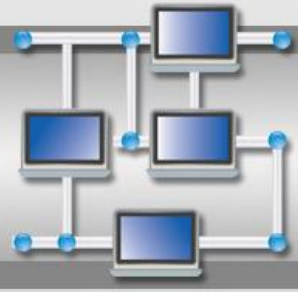
Модель клиент-сервер и микроядра

Объектно-ориентированный подход



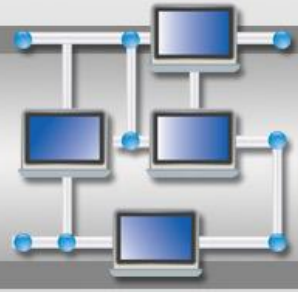
- Хотя технология микроядер и заложила основы модульных систем, способных развиваться регулярным образом, она не смогла в полной мере обеспечить возможности расширения систем.
- В настоящее время этой цели в наибольшей степени соответствует объектно-ориентированный подход, при котором каждый программный компонент является функционально изолированным от других.
- *Объект* - это единица программ и данных, взаимодействующая с другими объектам посредством приема и передачи сообщений. Объект может быть представлением как некоторых конкретных вещей - прикладной программы или документа, так и некоторых абстракций - процесса, события.
- Программы (функции) объекта определяют перечень действий, которые могут быть выполнены над данными этого объекта.
- Объект-клиент может обратиться к другому объекту, послав сообщение с запросом на выполнение какой-либо функции объекта-сервера.

Распределенная служба каталогов

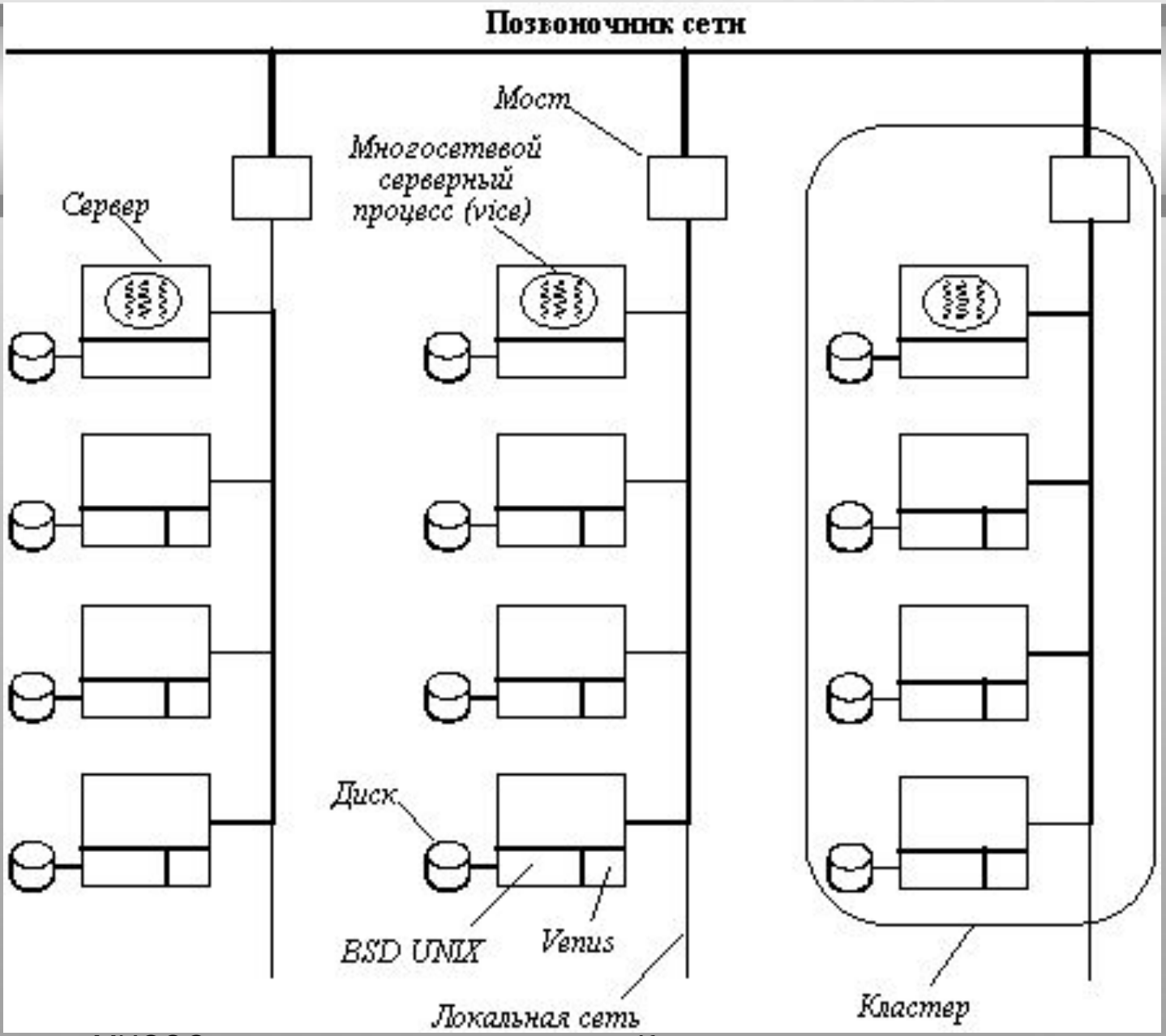


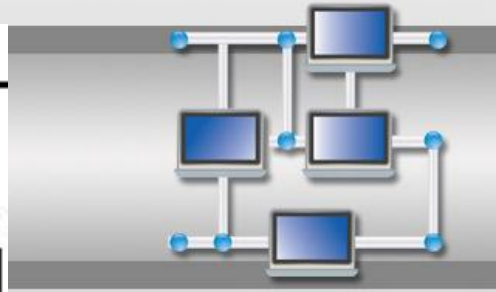
- Задачей службы каталогов в распределенной сети является поиск сетевых объектов, то есть пользователей, ресурсов, данных или приложений.
- Служба каталогов (или иначе, имен) должна отобразить большое количество системных объектов (пользователей, организаций, групп, компьютеров, принтеров, файлов, процессов, сервисов) на множество понятных пользователю имен.
- Более того, при появлении в сети распределенных приложений служба каталогов должна начать отслеживание всех таких объектов и всех их компонентов.

Распределенная служба безопасности



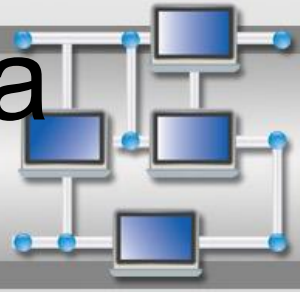
- Имеется две больших группы функций службы безопасности: идентификация и авторизация.
  - Идентификация проверяет идентичность объекта (например, пользователя или сервиса).
  - Авторизация (или управление доступом) назначает привилегии объекту, такие как доступ к файлу.
- Авторизация - это только часть решения. В
- распределенной сетевой среде обязательно должна работать глобальная служба идентификации, так как рабочей станции нельзя доверить функции идентификации себя или своего пользователя.
- Служба идентификации представляет собой механизм передачи третьей стороне функций проверки идентичности пользователя.





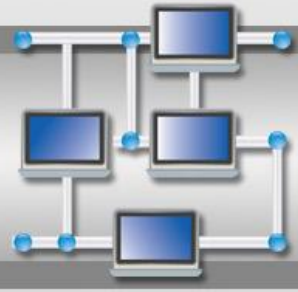
Эта система, созданная для студентов университета, не является прозрачной системой, в которой все ресурсы динамически назначаются всем пользователям при возникновении потребностей.

Распределенная служба времени

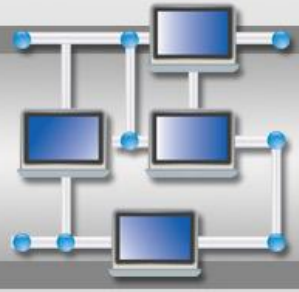


- В распределенных сетевых системах необходимо иметь службу согласования времени.
- Многие распределенные службы, такие как распределенная файловая система и служба идентификации, используют сравнение дат, сгенерированных на различных компьютерах.
- Чтобы сравнение имело смысл, пакет DCE должен обеспечивать согласованные временные отметки.

Популярные сетевые ОС



- Novell NetWare
- LANtastic
- Microsoft Windows (NT, XP, Vista, 7, 8)
- Различные UNIX системы, такие как Solaris, FreeBSD
- Различные GNU/Linux системы
- IOS
- ZyNOS компании ZyXEL



Спасибо за внимание!