Министерство транспорта и связи Украины
Государственный департамент по вопросам связи и информатизации
Одесская национальная академия связи им. А.С. Попова

Кафедра сетей связи

Д.А. Зайцев, А.В. Дорошук

Конспект лекций по курсу «Сетевые операционные системы»

Для подготовки бакалавров и магистров по направлению «Телекоммуникации»

Одобрено
на заседании кафедры
«Сети связи»
Протокол № 2 от
07.09.2007 г.

Одеса 2007

УДК 621.39, 004.7, 51.681.3

План НМВ 2007/2008

Рецензент – д.т.н., профессор А.И. Слепцов

Составители: д.т.н., доц. Д.А. Зайцев, к.т.н. А.В. Дорошук

Изложены основы теории операционных систем, а также специальные разделы, посвящённые особенностям функционирования операционных систем в сетевых средах, организации управления сетевыми ресурсами. Изучены структура и функции операционных систем семейства Unix, основы конфигурирования сетевых сервисов для семейства протоколов TCP/IP.

Утверждено
Советом факультета
Информационных сетей
Протокол № 2 от
27.09.2007 г.

Содержание

Вступление

Основы классической теории операционных систем (ОС) были созданы в 60-70 годы XX столетия при разработке таких известных операционных систем как MULTICS, OS 360. Дальнейшее развитие теории ОС было связано с интеграцией в состав операционных систем средств коммуникации посредством компьютерных сетей и графических интерфейсов пользователя. Операционные системы семейства Unix, которые появились в начале 70-х годов XX столетия практически одновременно с развитием сети ARPANET (трансформировавшейся затем во всемирную сеть Internet), к настоящему времени стали по существу стандартом де-факто сетевой ОС. Unix-подобными являются также специализированные операционные системы сетеобразующих устройств, например, ОС IOS маршрутизаторов компании Cisco.

Материал лекций скомпонован таким образом, что после изучения особенностей построения сетевых ОС рассматриваются классические разделы управления процессами, оперативной памятью, внешними устройствами, информацией, затем изучаются операционные системы семейства Unix. Полученные знания и навыки позволяют позиционировать успешных слушателей курса как квалифицированных администраторов сетевых операционных систем.

I. ВВЕДЕНИЕ В СЕТЕВЫЕ ОПЕРАЦИОННЫЕ СИСТЕМЫ

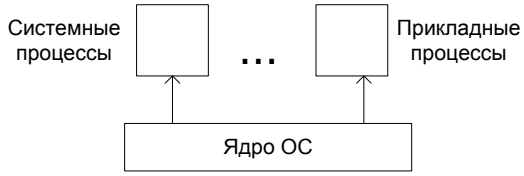
1. Определение операционной системы

Операционная система (ОС) – это комплекс программных средств для организации взаимодействия с пользователями, прикладными программами, аппаратными средствами и управления ресурсами компьютера.

Спецификой сетевой ОС является использование аппаратных средств подключения к сети (сетевые адаптеры, модемы) и обеспечение доступа к ресурсам сети (устройствам, информации, приложениям).

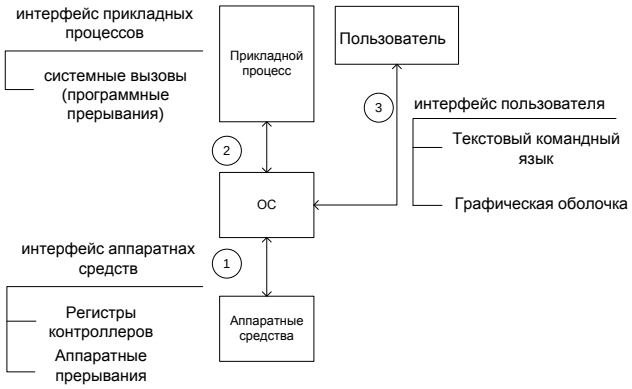
2. Структура ОС

ОС состоит из ядра и множества системных процессов. Ядро – это постоянно находящаяся в ОП часть операционной системы не являющаяся процессом. Ядро представляет собой множество обработчиков прерываний: аппаратных – для взаимодействия с устройствами, программных – для реализации системных вызовов. Системные процессы запускаются по мере необходимости ядром ОС.



3. Интерфейсы операционной системы

Организация взаимодействия предполагает создание множества интерфейсов: с пользователями, прикладными процессами, а также внутреннего интерфейса ОС с аппаратными средствами компьютера.



Из всех интерфейсов только командные языки доступны конечному пользователю. Наиболее популярными в настоящее время являются графические командные оболочки; текстовый командный язык является более лаконичным и удобным для удалённого доступа. Кроме того, на основе команд текстового языка создаются командные файлы.

Системные вызовы используются компиляторами алгоритмических языков для преобразования операторов ввода/вывода (и других) в конкретные системные вызовы ОС.

Интерфейсы внешних устройств представлены наборами их регистров (портов ввода/вывода) и аппаратными прерываниями.

ОС как средство взаимодействия:

С этой точки зрения функцией ОС является предоставление пользователю некоторой расширенной или виртуальной машины, которую легче программировать и с которой легче работать, чем непосредственно с аппаратурой, составляющей реальную машину.

4. Управление ресурсами

Основные ресурсы компьютера:

- процессорное время (процессы);
- оперативная память;
- внешние устройства:
  - диски,
  - ленты,
  - принтеры,
  - сетевые адаптеры,
  - модемы.
- информация – файловая система, предоставляемая ОС.

Основные операции над ресурсами:

- выделение ресурса;
- освобождение ресурса;
- отслеживание состояния ресурса;
- планирование ресурса.

Таким образом, ОС должна насчитывать, по крайней мере, 16 подсистем (4 типа ресурсов, 4 операции). Наиболее сложной является операция планирования ресурса, поскольку она обеспечивает оптимальный режим работы ОС.

ОС как система управления ресурсами:

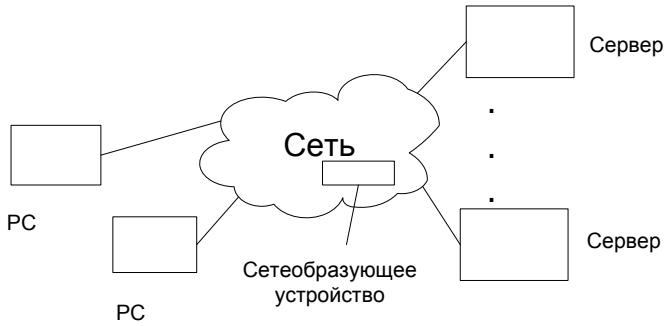
Функцией ОС является распределение процессоров, памяти, устройств и данных между процессами, конкурирующими за эти ресурсы. ОС должна управлять всеми ресурсами вычислительной машины таким образом, чтобы обеспечить максимальную эффективность ее функционирования. Критерием эффективности может быть, например,

пропускная способность или реактивность системы. Управление ресурсами включает решение двух общих, не зависящих от типа ресурса задач:

- планирование ресурса - то есть определение, кому, когда, а для делимых ресурсов и в каком количестве, необходимо выделить данный ресурс;
- отслеживание состояния ресурса - то есть поддержание оперативной информации о том, занят или не занят ресурс, а для делимых ресурсов - какое количество ресурса уже распределено, а какое свободно.

5. Общая организация компьютерных сетей

В обобщённом виде компьютерная сеть может быть представлена как множество терминальных устройств (компьютеров), подключенных к некоторой универсальной коммуникационной среде, именуемой «Сеть»:



Терминальными устройствами сети являются рабочие станции (PC) и серверы (C):

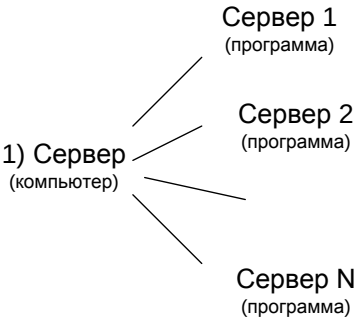
- PC предназначена для работы пользователя с приложениями; PC обеспечивает потребление сетевых ресурсов, соответствующие процессы называют клиентами;
- Сервер обеспечивает представление ресурсов в сети.

Сетеобразующие устройства (коммутаторы, маршрутизаторы) обеспечивают обмен информацией в сети и являются прозрачными с точки зрения доставки информации терминальным устройствам.

По специализации различают следующие PC:

- графические процессоры (CAD/CAM);
- текстовые процессоры (офис);
- администрирование;
- разработка ПО (CASE).

Сервер – представляет набор услуг в сети, причём термином «сервер» обозначают как конкретный компьютер, так и приложение (процесс), обеспечивающий некоторый вид сервиса:



2) иногда серверы специализируются по видам сервиса, например:

- устройства публичного доступа (сетевые принтеры);
- файловая система общего пользования;
- доступ к приложениям;

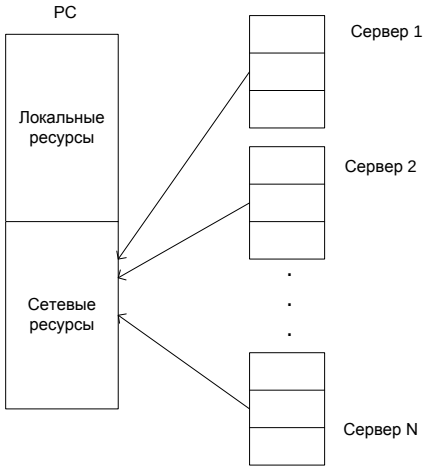
- каталоги пользователей;
- контроль доступа (авторизация, аутентификация);
- электронная почта;
- WEB-серверы;
- FTP-серверы.

Организация доступа к сетевым ресурсам

Следует выделить три основных способа доступа к сетевым ресурсам:

- Специальные приложения, которые обеспечивают доступ к определённым видам сервиса:
  WEB – браузер (IE, Netscape);
  FTP – CutFTP.

- Отображение сетевых ресурсов на локальные ресурсы компьютера:



В этом случае способ обращения к сетевым ресурсам такой же, как и к локальным ресурсам. Отображение задаёт либо администратор системы, либо пользователь (например, подключение сетевых дисков в MS Windows).

- использование сетевых имён и адресов:

Локальные ресурсы	Сетевой ресурс
C:/Docs/resume.doc	//MKMOST/C:/Docs/resume.doc

Сетевое имя компьютера является префиксом, предваряющем имя ресурса в компьютере. В этом случае используются те же операции, что и для локальных ресурсов, (например, копирование – команда cp) только с указанием сетевых имён ресурсов.

6. Обзор современных операционных систем

Было предложено множество различных классификаций операционных систем; пример детализированная классификация, приведенная в Приложении 1. Однако современные ОС, как правило, аккумулируют возможности, указываемые в качестве ключевых признаков классификации. Например, ОС Unix обеспечивает как пакетный, так и диалоговый режим работы, поэтому её нельзя классифицировать как только диалоговую либо только пакетную ОС.

Наиболее прагматичной представляется приведенная ниже классификация и последующее изучение возможностей современных ОС, которые могут быть фундаментальными, как мультипрограммирование и квантование времени, либо специфичными, как создание виртуальных машин. Набор конкретных возможностей в

настоящее время, как правило, выбирается при инсталляции (генерации) операционной системы.

1. Операционная система общего назначения

SOHO: для дома и офиса	Рабочих станций	Серверов
MacOS MS WINDOWS 95/98, XP	MS Win NT Workstation UNIX: Linux,Solaris,Irix,AIX	IBM OS 390, ZOS Unix Server MS Win NT Server

2. Специализированные операционные системы:
   - реального времени: QNX, RTOS;
   - суперкомпьютеров: UNICOS, Cray OS;
   - сетеобразующих устройств: CISCO/IOS;
   - сетевых компьютеров: Java OS;
   - интеллектуальных устройств: Windows CE, PalmOS.

Обзор современных аппаратных платформ

Практический опыт работы только с IBM совместимыми персональными компьютерами подчас создаёт ложное представление о современном рынке, насчитывающем более десятка основных аппаратных платформ. Далее приведен обзор современных аппаратных платформ и их ОС:

Фирма	Процессор	Серия компьютеров	Операционная система
Intel	Intel Pentium Intel Itanium Intel Xeon		Windows Unix (Linux)
Apple	Power PC	Xserie Power Mac iMac	Mac OS
Sun	Ultra Spare Intel	Sun Fire Sun Enterprise	Solaris (Unix) Linux
IBM	Intel RS6000 Power 4 Z Processor Super Comp	Intellistation, xserie RS6000. Intellistarion. Pserie Zserie	Linux, Windows AIX, Linux AIX, Linux, Windows ZOS, Linux
SGI	Intel Itanium MIPS	SGI Altix SGI Origin SGI Onyx SGI Graphics	IRIX Linux
HP	PA-8700+ Intel Itanium Alpha	HP Super done HP RX Alpha Server	HP-UX, Linux Windows True 64 Unix

В настоящее время существует два доминирующих семейства операционных систем: MS Windows и Unix. Переносимость ОС Unix обеспечивает их применение практически на каждой аппаратной платформе. Среди ОС семейства Unix различают:

- коммерческие: Solaris, AIX, HP-UX, Irix;
- свободно распространяемые: Linux, Free BSD.

Обзор основных возможностей современных ОС

Возможности современных ОС:

- пакетный и диалоговый режим работы,
- мультипрограммирование,
- квантование времени,
- использование приоритетов,
- средства взаимодействия процесса,
- виртуальная страничная (сегментно-страничная) память,
- средства защиты – пользователей, процессов, устройств информации.
- графический оконный интерфейс,
- виртуальная машина.

7. Организация функционирования сетевых операционных систем

- Средства доступа к сети:
  - Локальная сеть (Ethernet, Tokin Ring, Apple Talk) – сетевой адаптер (карта);
  - Глобальная сеть (удалённый доступ) – модем.
- Сетевые протоколы:

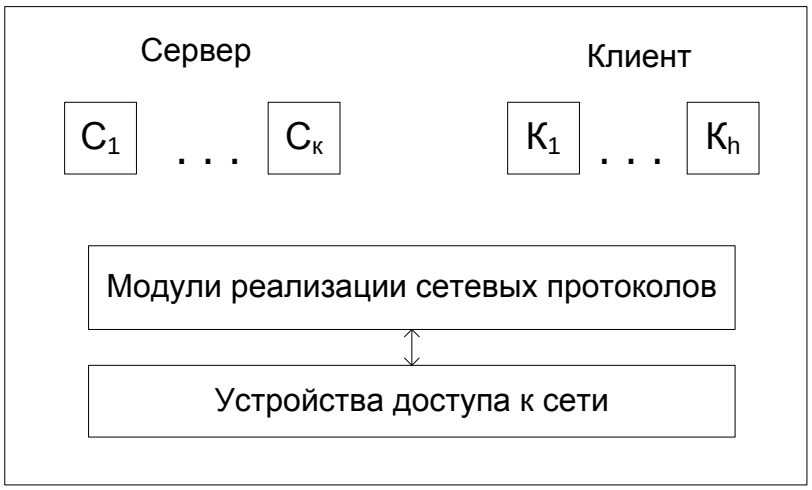
Протокол – строго регламентированный порядок взаимодействия систем в сети. Основным стандартом является OSI/ISO – семиуровневая эталонная модель взаимодействия открытых систем.

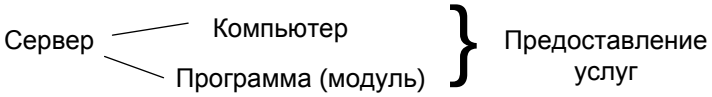
Доминирующие семейства протоколов:

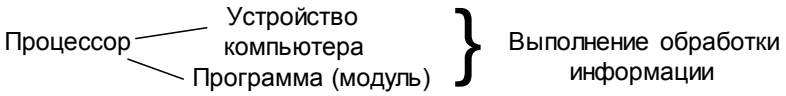
TCP/IP – и глобальная и локальная сеть

IPX/SPX, NetBEUI – локальные сети

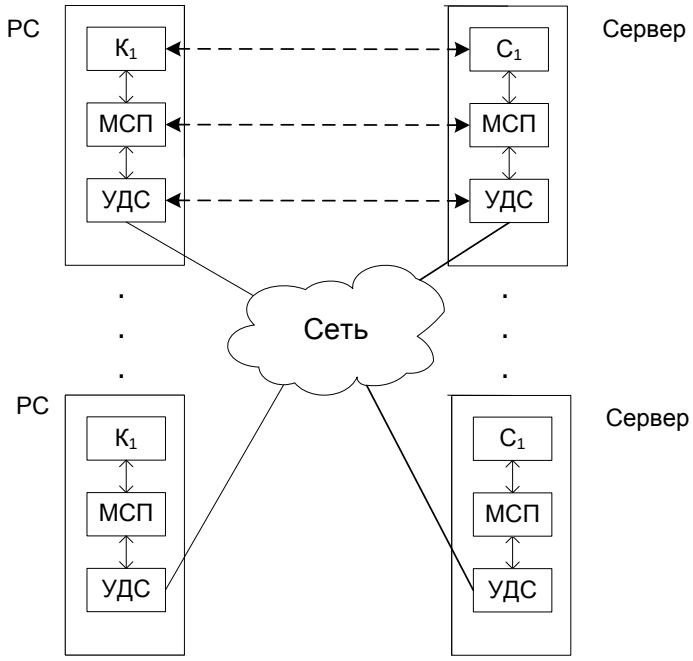
- Организация функционирования сетевой операционной системы:
  - Типовая структура сетевой ОС:







- Взаимодействие сетевых ОС:

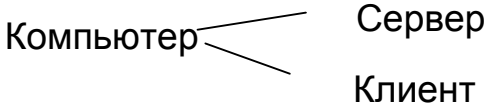


Каждое клиентское приложение (клиент) взаимодействует со своим сервером.

PC – множество клиентов.

Сервер – множество сервисов (серверных процессов).

- Отсутствует жёсткая специализация:



Если отсутствует жёсткая специализация, то сети называют одноранговыми.

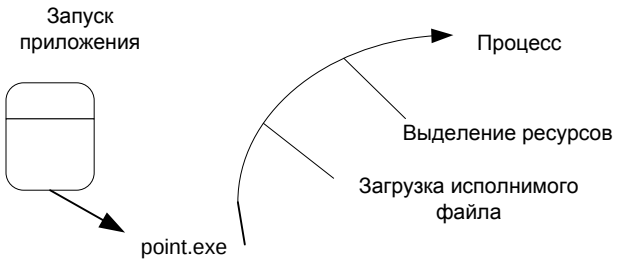
II. УПРАВЛЕНИЕ ПРОЦЕССАМИ

Ключом к пониманию функционирования средств управления процессорами (процессами) современных ОС является диаграмма состояний процесса. Модули ОС обеспечивают изменение состояний процессов в соответствии с диаграммой, переключение контекста процессора для запуска текущего процесса, вызов подсистемы ввода-вывода для инициирования операций на внешних устройствах.

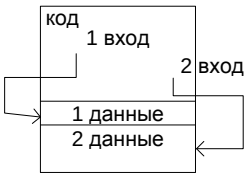
1. Организация выполнения приложений в ОС

Приложение – исполнимый файл.

Процесс – это приложение на стадии выполнения, либо это действие производимое процессором по выполнению приложения (программы) на заданных исходных данных. Процесс создаётся путём загрузки соответствующего приложения в ОП, выделения ему требуемых ресурсов и запуска на исполнение:



Реентерабельность (повторная входимость) – организация нескольких процессов по одному и тому же находящемуся в оперативной памяти программному коду:



2. Диаграмма состояний процесса

Состояния процесса:

Представление - образование процесса из исполняемого файла.

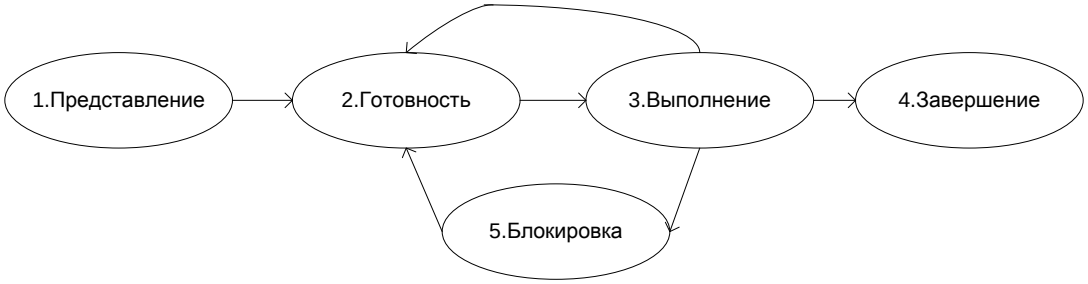
Готовность – пассивное состояние процесса, процесс имеет все требуемые для него ресурсы, кроме процессора; он готов выполняться, однако процессор занят выполнением другого процесса.

Выполнение – активное состояние процесса, во время которого процесс обладает всеми необходимыми ресурсами и непосредственно выполняется на процессоре.

Блокировка – пассивное состояние процесса, процесс заблокирован, он не может выполняться по своим внутренним причинам, он ждет наступления некоторого события, например, завершения операции ввода-вывода, получения сообщения от другого процесса, освобождения какого-либо необходимого ему ресурса.

Завершение – процесс освобождает ресурсы и покидает вычислительную систему.

Диаграмма состояний процесса является ключом к пониманию организации функционирования ОС. Типовая диаграмма состояний процесса имеет вид:



1 → 2: поиск исполняемого файла, выделение оперативной памяти, загрузка исполняемого файла в ОЗУ, постановка в очередь готовности процесса.

2 → 3: Копирование контекста процессора из блока управления процессом на физические секции.

3 → 5: Типовой запуск операции ввода/вывода на внешние устройства. Контекст процессора сохраняется в блоке управление процесса.

3 → 2: Завершение кванта времени (прерывание таймера).

3 → 4: Процесс освобождения выделенных ресурсов, ОС уничтожает все внутренние данные описывающие процесс.

Детализированная диаграмма состояния процесса является основным средством проектирования ОС и организации взаимосвязи модулей ОС.

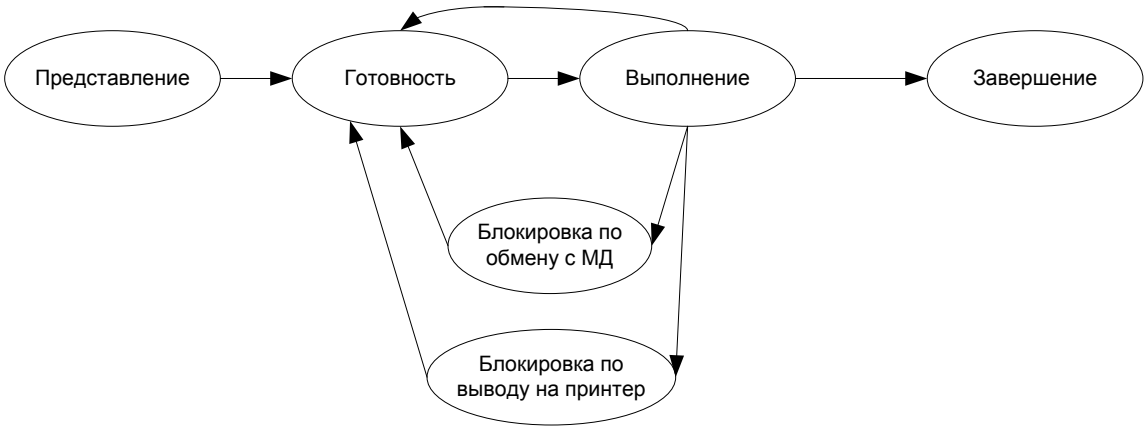
Основное направление детализации – выделение различных типов блокировок, и уровней готовности (например, по приоритетам).

Запуск перехода в диаграмме инициируется некоторым событием (прерыванием).

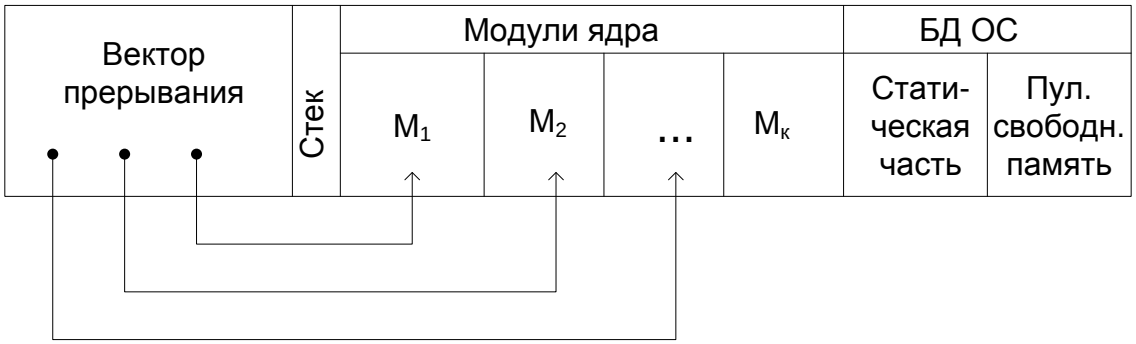
Реализация перехода – вызов соответствующего модуля ядра ОС.

Пример: пусть существует следующие типы блокировок:

- ожидание обмена с шиной данных (ШД);
- ожидание завершение операции на принтере;



3. Структура ядра и базы данных ОС



Ядро – набор обработчиков прерываний (не имеет собственной активности).

Аппаратные средства и процессы однотипно взаимодействуют с ядром – системные вызовы реализуют в форме программных прерываний.

Функции ядра операционной системы:

- создание/уничтожение процесса,
- планирование процесса,
- организация взаимодействия процесса,
- управление оперативной памятью,
- управление внешними устройствами,
- реализация основных механизмов защиты.

База данных ОС

Управление ресурсами организуется на основе описателей (блоков управления) для каждого из типов ресурсов:

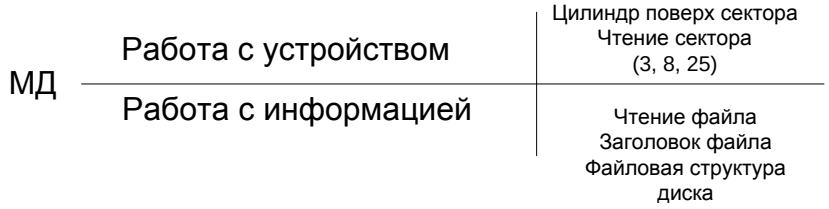
БУП – блок управления процессом.

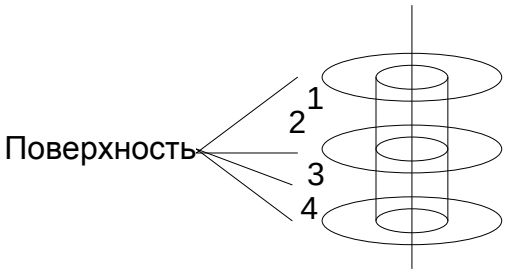
БУМД – магнитный диск.

БУОП – блок управления оперативной памятью;

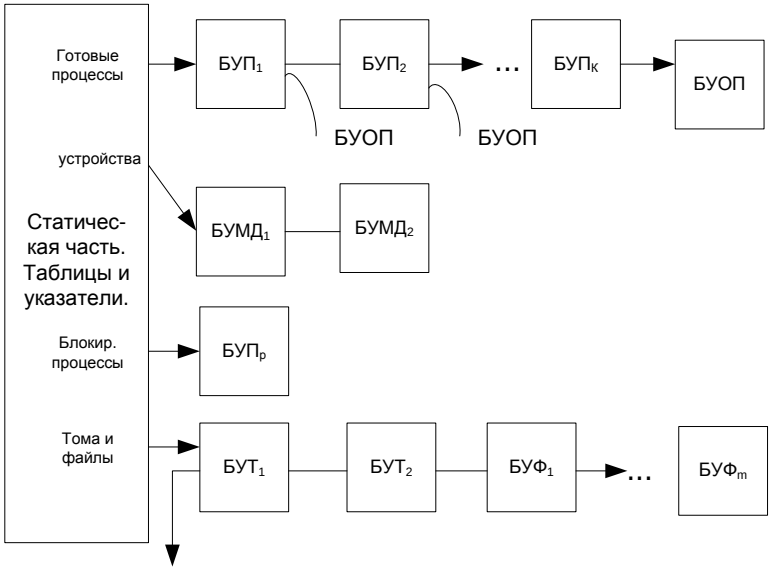
БУТ – блок управления томом (том - описатель файловой системы устройства).

БУФ – блок управления файла.



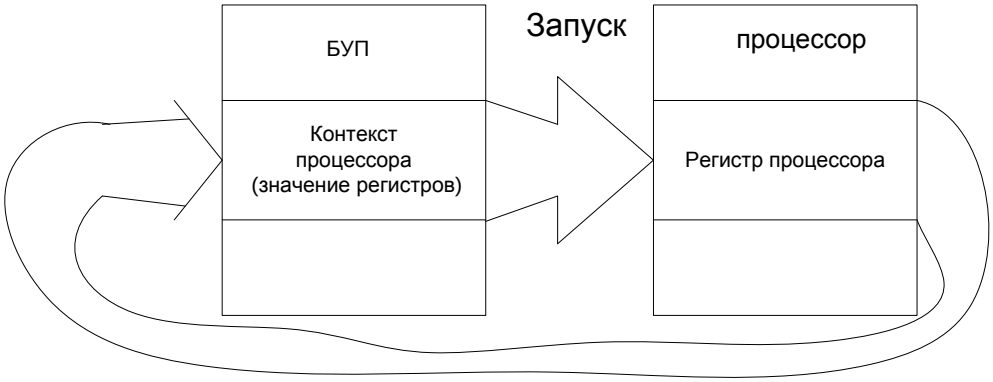


Управляющие блоки объединяются в многоуровневые очереди либо списки:



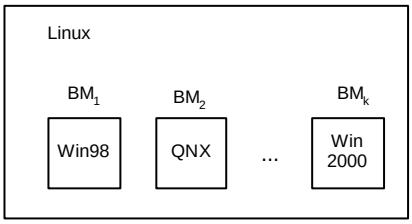
Блоки создаются в динамической памяти (пуле) ядра ОС. Указатели объединяются в многоуровневые связные списки.

Поясним использование БУП для переключения процессов. Фактический запуск процесса на исполнение осуществляется путём копирования его контекста на регистры процессора; во время прерывания или переключения, контекст процессора запоминается в БУП. Таким образом, процесс «не замечает» переключений:



Виртуальная машина

Виртуальная машина – это программная модель аппаратных средств компьютера. Виртуальных машин может быть несколько; в каждой виртуальной машине может быть запущена своя собственная ОС.



Как правило, запуск виртуальной машины требует статического выделения ей ресурсов, в особенности ОП.

4. Пакетный и диалоговый режимы

Текстовый командный язык ОС вопреки ожиданиям не был вытеснен графическими командными оболочками. Он представляет собой лаконичный интерфейс пользователя/администратора. Кроме того, современные ОС обеспечивают формирование командных (пакетных) файлов (скриптов) на его основе. Пакетные файлы могут выполняться как в интерактивном, так и в отложенном (пакетном) режиме, обеспечивая загрузку ОС заданиями во время отсутствия пользователя.

текстовый язык
- Запуск/завершение процесса
- Создание/уничтожение элементов файловой структуры
- Информация о состоянии программы/устройства/ффайл

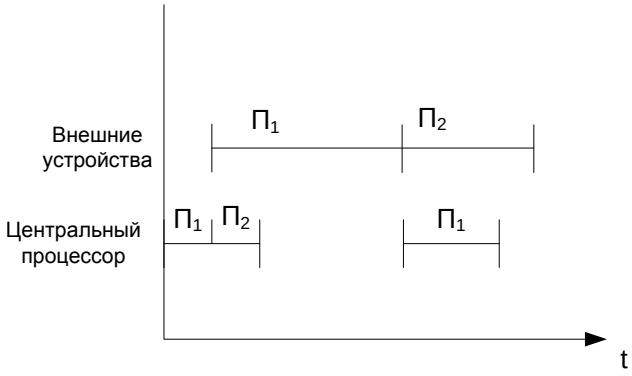
Язык пакетных файлов:
- последовательность команд,
- управляющие конструкции: ветвление, цикл.

5. Мультипрограммирование

Мультипрограммирование – одновременное выполнение нескольких процессов.

Обеспечивается за счёт параллельной работы центрального процессора и внешних устройств.

Пример временной диаграммы:

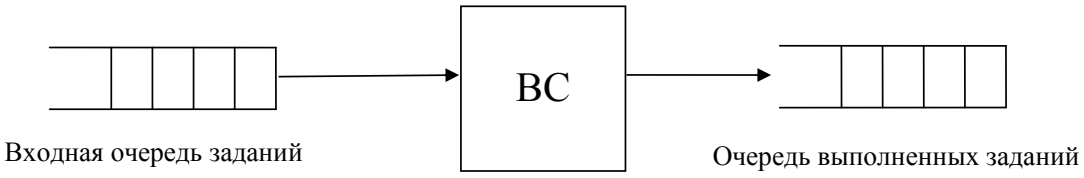


Мультипрограммирование обеспечивает эффективное использование устройств вычислительной системы, повышает их загрузку.

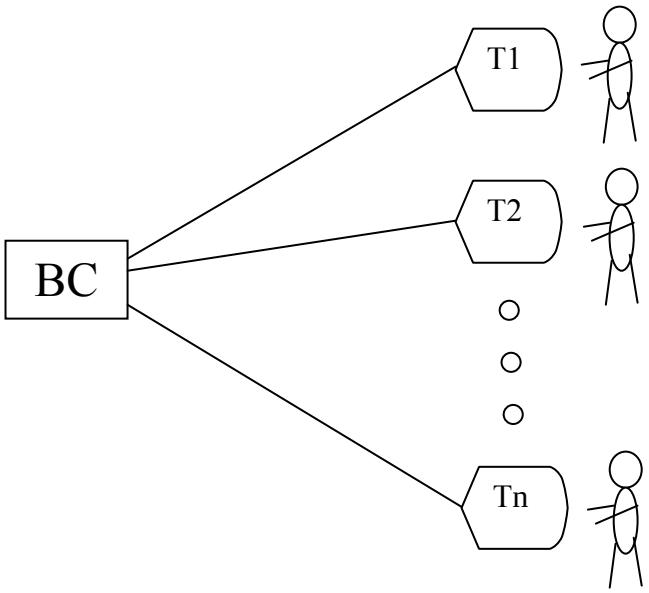
Процессы загружаются в ОП и образуют очереди готовых процессов. В то время, как один из процессов выполняет операцию ввода/вывода, другой процесс занимает процессор. Заметим, что мультипрограммирование возможно только при наличии аппаратных средств, обеспечивающих параллельный с работой процессора ввод/вывод: контроллеров ВУ, каналов. Сигнализация о завершении ввода/вывода выполняется с помощью аппаратных прерываний; обработчиком прерывания является соответствующий модуль ядра ОС, который и выполняет переключение процессов.

6. Квантование времени

Первые мультипрограммные ОС функционировали в *пакетном режиме*, в котором пользователь отделялся от процесса выполнения его заданий. Предварительно подготовленные задания организовывались в очереди на перфокартах либо магнитном диске. Затем смесь заданий подавалась на вход вычислительной системы:

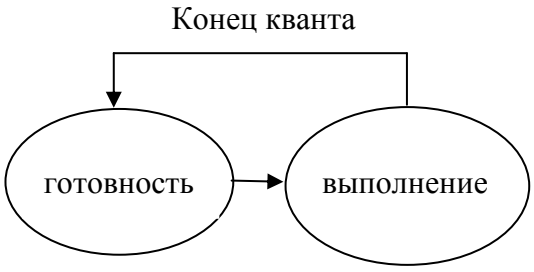


Необходимость взаимодействия пользователя с выполняющимися программами, автоматизация процессов подготовки заданий с помощью текстовых редакторов, а также появление дисплеев с электронно-лучевой трубкой обусловили появление ВС с *диалоговым режимом* работы. В этом режиме несколько пользователей одновременно взаимодействуют с ВС с помощью терминалов:

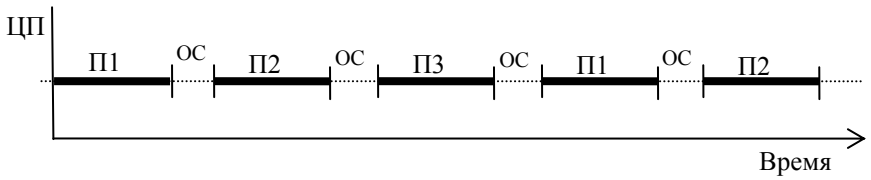


Для обеспечения работы нескольких пользователей в диалоговом режиме особенно важным становится *время реакции ОС* на команды пользователя. При использовании классического мультипрограммирования один из процессов может захватить процессор на неопределённое время и, таким образом, блокировать работу остальных пользователей. Эффективный диалоговый режим возможен при гарантированном времени реакции ОС не превышающем несколько секунд.

Для обеспечения диалоговой работы был предложен *режим разделения времени*, обеспечивающий совместную работу нескольких пользователей. Основой для обеспечения режима разделения времени является *циклическое квантование времени* центрального процессора. Каждому процессу при получении процессора выделяется определённый интервал времени – *квант*, в течение которого он выполняется на процессоре; затем процесс возвращается в конец очереди готовых процессов. Таким образом, процессор циклически переключается между готовыми процессами и каждый из пользователей считает, что ОС работает только с ним одним; далее представлен фрагмент диаграммы состояний процесса, обеспечивающий циклическое квантование:



Следует отметить, что организация квантования влечёт за собой некоторое снижение производительности ВС. Переключение между процессами требует определённого времени, в течение которого работают программы ОС, и представляет собой *накладные расходы* ресурсов ОС при организации режима разделения времени. При реализации циклического квантования встаёт вопрос о выборе *оптимального размера кванта*. Большой размер кванта снижает время реакции ОС, малый – увеличивает накладные расходы из-за частого переключения. Реальные ОС применяют адаптивные механизмы выбора размера кванта.



Квантование трёх процессов

Для реализации квантования времени используется аппаратный таймер. Таймер устанавливается на продолжительность кванта при выделении процессора процессу; прерывание таймера по истечении времени кванта инициирует переключение.

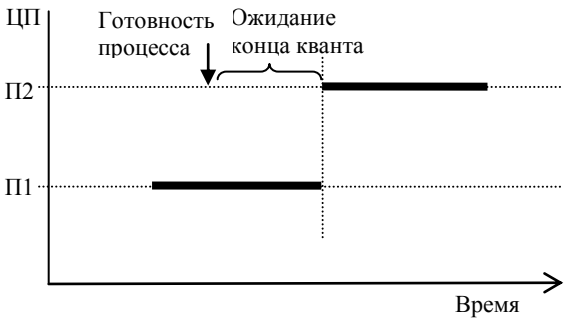
7. Приоритетные дисциплины планирования процессов

Множество процессов, исполняемых в среде ОС, можно разделить на *системные* и *прикладные*. Системные процессы являются частью операционной системы и обеспечивают работу ВС в целом. Таким образом, во многих случаях требуется обеспечить преимущественное выделение им ресурсов, в особенности, центрального процессора.

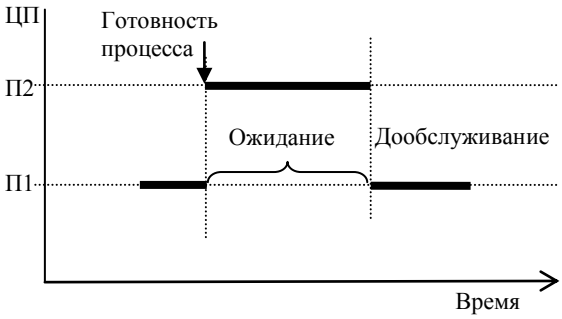
Кроме того, прикладные процессы также следует различать на основе предпочтительного выделения ресурсов, особенно при использовании ВС в технологическом управлении. Ввиду жёстких ограничений на время реакции и связанных с этим особенностей построения, операционные системы, предназначенные для технологического управления, выделяют в специальный класс, именуемый *операционные системы реального времени* (ОСРВ).

Приоритетные дисциплины планирования процессов были предложены как для оптимизации выполнения смеси заданий, так и для обеспечения оперативности исполнения системных процессов и процессов реального времени.

Уровень приоритета во многих случаях может быть представлен целым числом. Различают два типа приоритетов процессов: *относительные и абсолютные*. Относительные приоритеты используются только при постановке процессов в очередь, таким образом, появление процесса с более высоким относительным приоритетом не прерывает исполнение на процессоре текущего процесса. Абсолютный приоритет сравнивается с приоритетом текущего выполняемого процесса и, в случае превышения, прерывает исполнение текущего процесса до завершения его кванта:



Относительный приоритет



Абсолютный приоритет

Таким образом, абсолютные приоритеты используются для мгновенного переключения на новый процесс, и применяется для процессов реального времени и наиболее важных системных процессов, например, самой подсистемы управления процессами.

Различаю *статические и динамические приоритеты*. Статические приоритеты назначаются вручную администратором ОС, либо пользователем в определённых отведенных ему администратором ОС границах и не изменяются во время исполнения процессов. Динамические приоритеты назначаются и изменяются ОС во время выполнения процессов в целях оптимизации функционирования ВС. Например, ОС может повысить приоритет процесса, часто выполняющего ввод/вывод и понизить приоритет процесса, продолжительно занимающего процессор, для повышения загрузки устройств ВС.

Статические абсолютные приоритеты обеспечивают минимальное время реакции ОС РВ в технологическом управлении. Следует отметить, что применение приоритетных дисциплин не всегда оптимально с точки зрения критерия обеспечения максимальной загрузки устройств ВС. Для ОС РВ на первое место выдвигаются критерии минимального времени реакции, что оправдано, так как потери из-за неоперативного технологического управления могут значительно превышать потери из-за простоя устройств ВС.

8. Средства взаимодействия процессов

Средства взаимодействия процессов – служат для обмена информации между выполняющимися процессами.

Основные средства взаимодействия:

- флаги событий,
- семафоры,
- сообщения,
- общие области памяти.

Средства взаимодействия процессов – применяются для построения самой ОС.

Часть ОС, не являющаяся ядром, представляет собой набор взаимодействующих процессов.

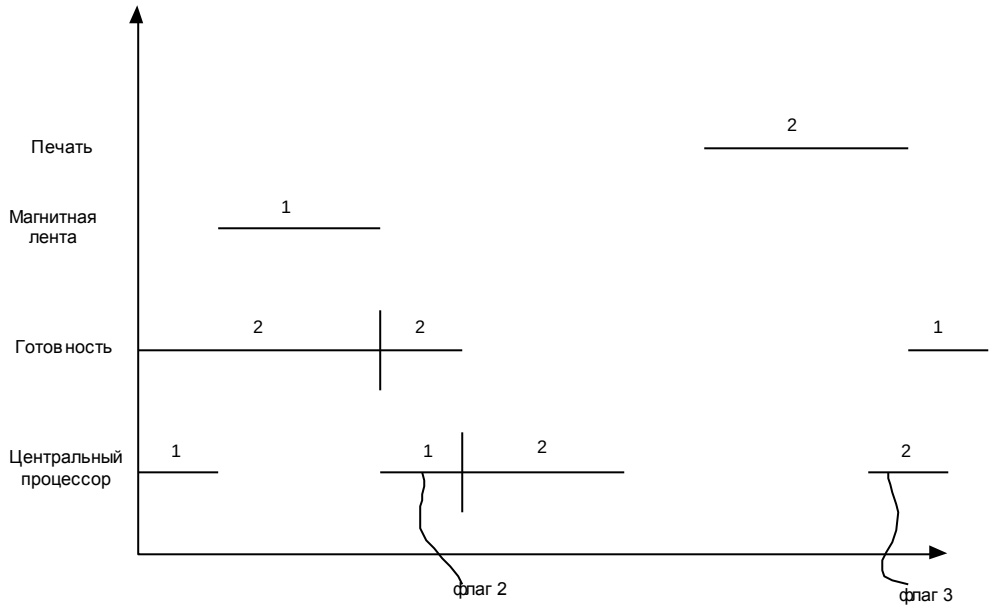
Пример:

Флаги событий – битовые переменные, доступ к которым имеют все процессы с помощью специальных операций.

Операции: Установить флаг (N)
Сбросить флаг (N)
Ожидать флаг (N)

Процессы, которые используют эти флаги:

Процесс 1	Процесс 2
Сбросить флаг (2)	Ожидать флаг (2)
Читать МЛ	Сбросить флаг (3)
Установить флаг (2)	Вывести на печать
Передать флаг (3)	Установить флаг (3)
Перейти к (1)	Перейти к (3)



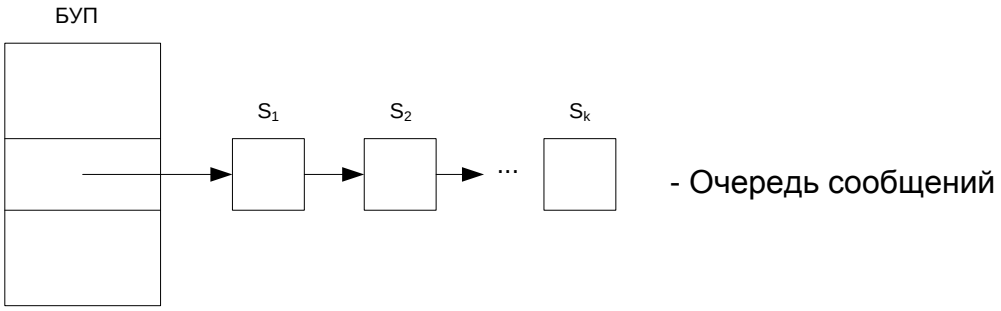
Семафоры – целые числа.

- P (S) +1
- V (S) -1, если 0, то ожидание.

Сообщение

Послать (S,P), где S – сообщение, P – процесс

Ожидать сообщение (S)



9. Средства защиты ОС

Объекты, подлежащие защите:

- пользователи,
- ресурсы (процессы, ОП, устройства, информация),
- сама ОС.

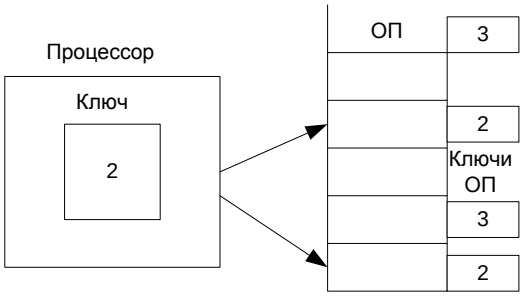
Следует отметить, что ОС должна защитить, прежде всего, саму себя – ядро, системные процессы, БД ОС для обеспечения защиты других объектов.

Полноценная защита возможна только при использовании специальных аппаратных средств защиты, позволяющих отследить несанкционированное действие и передать управление соответствующему модулю ОС с помощью прерывания.

Основой аппаратных средств защиты является наличие нескольких режимов процессора с различными привилегиями. Должно быть, по крайней мере, два режима:

- Пользователь
- Система

Отличия между ними заключаются в наборе доступных команд. Для обеспечения защиты памяти могут использоваться специальные ключи:

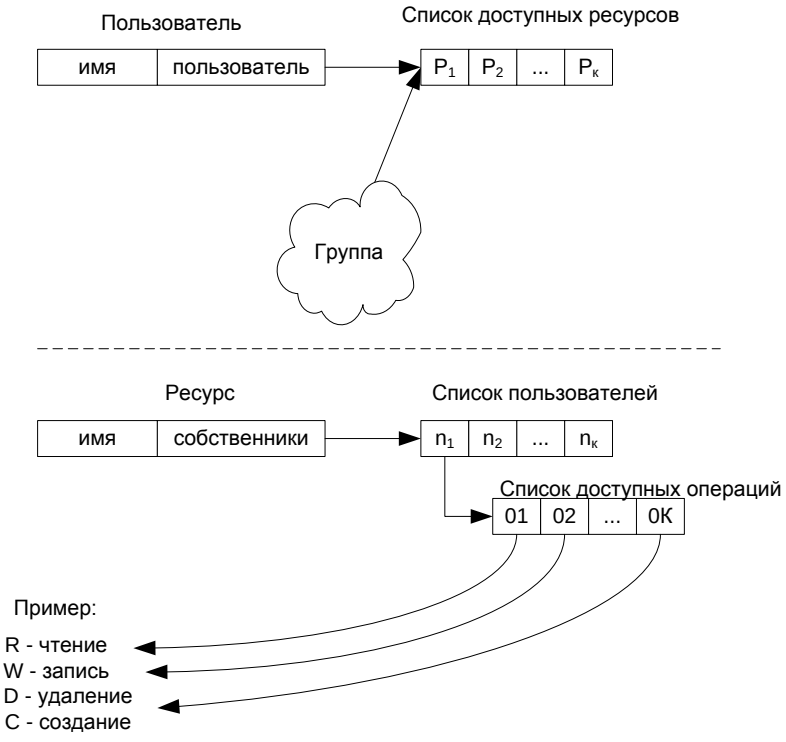


Все команды непосредственного изменения регистров распределения ОП, команды ввода/вывода (записи в регистры ВУ), как и сама команда переключения режима работы процессора, должны быть привилегированными. Попытка исполнения такой команды прикладным процессом, выполняющегося в режиме Пользователь, приводит в аппаратной генерации соответствующего прерывания «недопустимая команда».

Далее, после обеспечения защиты ОП, которая также должна быть аппаратной и генерировать соответствующие прерывания при несанкционированной попытке доступа возможно построение полноценной защиты всех остальных ресурсов.

Как правило, описатель каждого ресурса дополняется полями прав доступа, указывающими его владельца, а также остальных пользователей, которым разрешён доступ к ресурсу; кроме того, указывают санкционированный вид доступа: чтение, запись, выполнение.

Общая схема организации защиты в ОС



Во многих случаях громоздких списков доступа можно избежать за счёт разумного объединения пользователей в группы.

III. УПРАВЛЕНИЕ ПАМЯТЬЮ

Оперативная память (ОП) является вторым после ЦП ресурсом по своей значимости для исполнения процесса. Множество способов управления памятью конкретных ОС можно классифицировать по следующим основным признакам:

- полное либо частичное размещение процесса в ОП;
- связное либо несвязное выделение оперативной памяти;
- выделение участков памяти фиксированной либо переменной длины.

Многие сложные методы управления ОП, обусловленные исторически высокой стоимостью и ограничениями объёма физической памяти, например, оверлейные структуры, практически не используются в настоящее время в связи с развитием концепции виртуальной памяти. Стандартом де-факто для современных ОС является виртуальная сегментно-страничная память.

Стратегия управления памятью

1. Выделение - (что)
2. Размещение - (куда)
3. Вытеснение – (когда)

Выделение:

- по запросам
- упреждающее

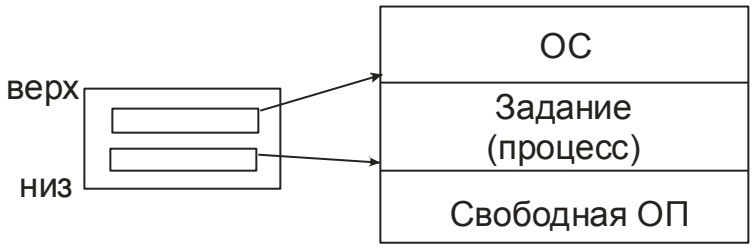
Способ выделения памяти:

- связное
- несвязное

Объем использования ресурса:

- физическая память (ОЗУ)
- виртуальная память (ОЗУ+ВП)

1. Управление памятью в однопрограммном режиме



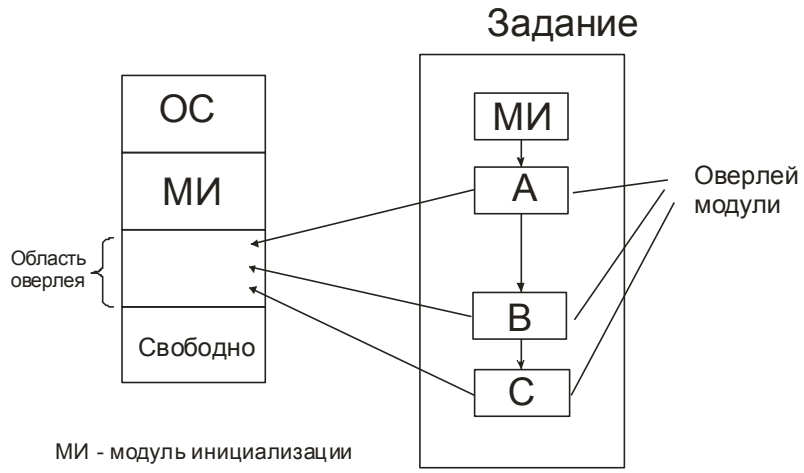
1. Заданию (процессу) выделялась вся доступная ОП
2. Для защиты ОС использовались граничные регистры
3. Для обеспечения возможности изменения размера ОС применялась базовая адресация

   Адрес – относительно начала программы

   GOTO A

   Исп.адрес – (Версия гран. регистр) + A
4. Для исполнения программ, размер которых превосходил размер ОП, использовались оверлеи

Оверлеи

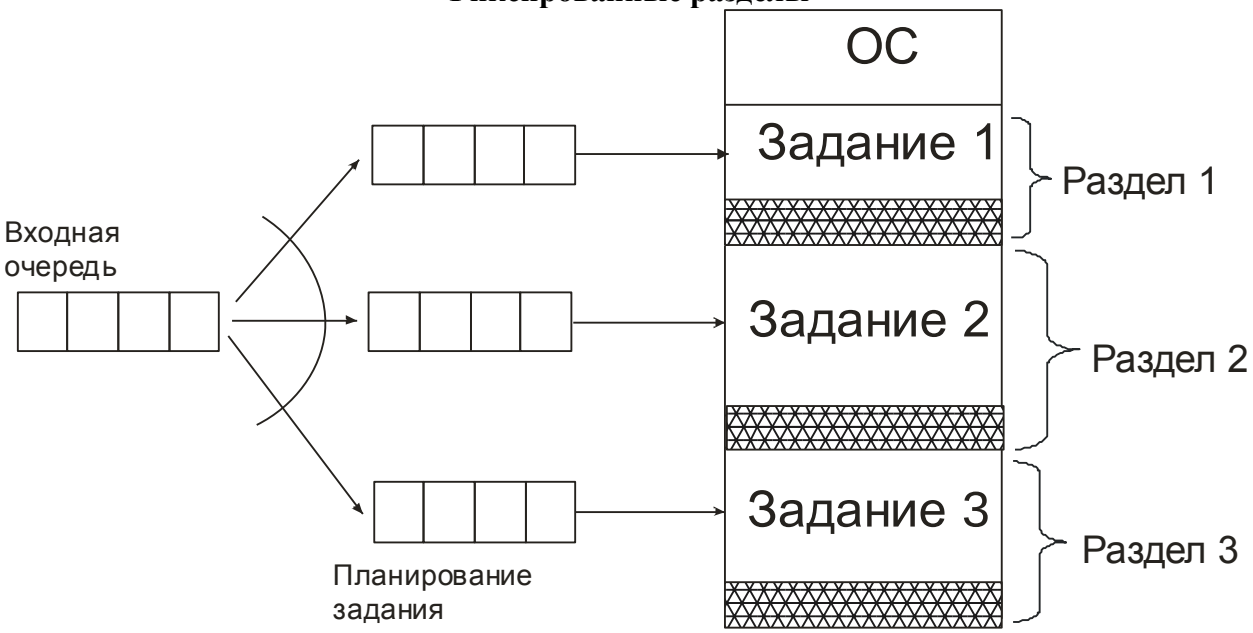


Оверлейная область выделялась по размеру максимального модуля, затем модули загружались последовательно в оверлейную область.

2. Управление разделами

Исторически управление оперативной памятью разделами было первым способом управления ОП в мультипрограммном режиме, когда потребовалось одновременное размещение нескольких процессов в оперативной памяти, для обеспечения быстрого переключения между ними.

Фиксированные разделы



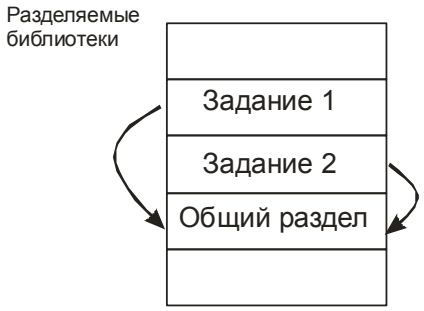
Структура раздела формировалась при инсталляции ОС.

Раздел – процесс

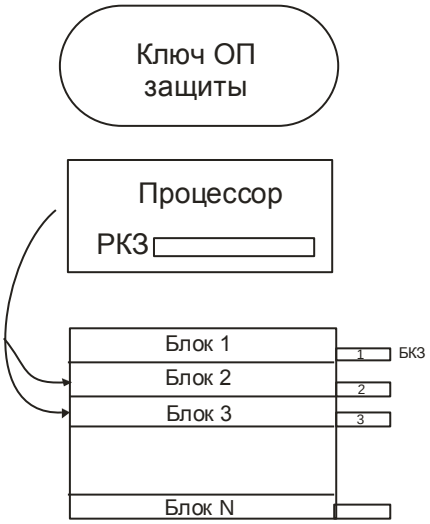
Стратегии:
- первый подходящий (свободный)
- наиболее подходящий
- наименее подходящий

Недостаток:
- большой резерв ОП
- невозможность предоставить одному заданию всю свободную ОП

Общие разделы памяти



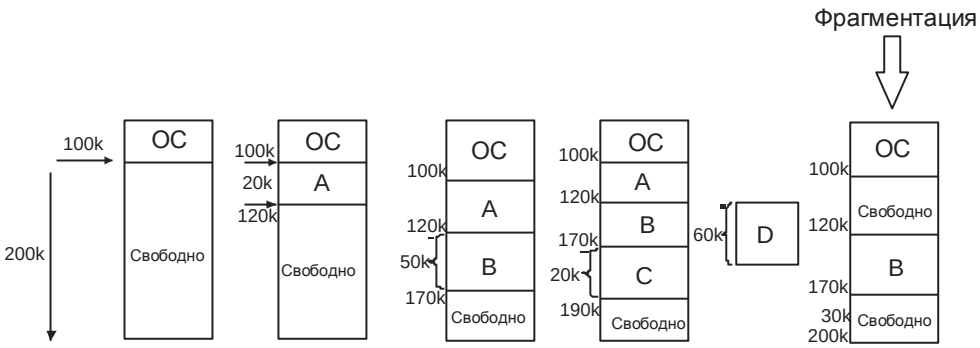
Общий раздел – раздел, к которому может обращаться несколько заданий
Проблемы – защита



Разделы переменной длинны (динамические разделы)

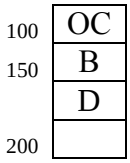
Задание выполняется в произвольном подходящем по размеру участке ОП. Оставшаяся память может использоваться другими заданиями.

Проблемы:
- учет свободной ОП
- дефрагментация



Суммарный объем позволяет выполнить операцию, но память находиться в смежных областях.

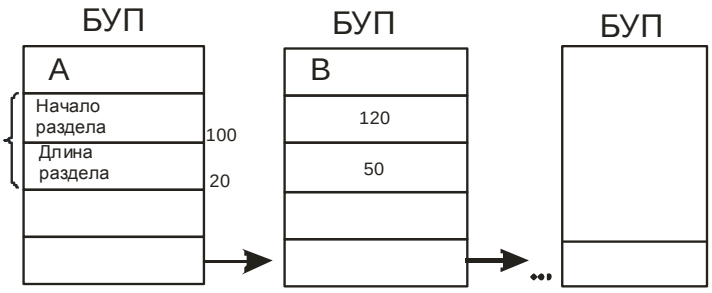
Для ликвидации:



Перемещение – накладные расходы.

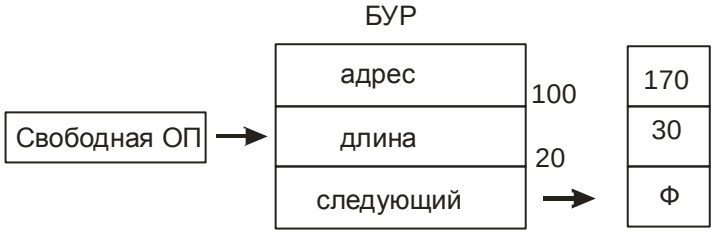
Отслеживание свободных (занятых) участков ОП

Отслеживание занятых участков:



Отслеживание свободных участков:

1. Создание блоков управления разделом в БД ОС
2. Заголовок свободного фрагмента



Свопинг – временное вытеснение заблокированного процесса во внешнюю память.

3. Виртуальная страничная память

Страничная организация:

1. Выделение памяти несмежными областями:
- равного размера – страничная организация памяти
- произвольного размера – сегментная

2. Загружать для выполнения только ту часть процесса, которая необходима в настоящее время.

Описание:

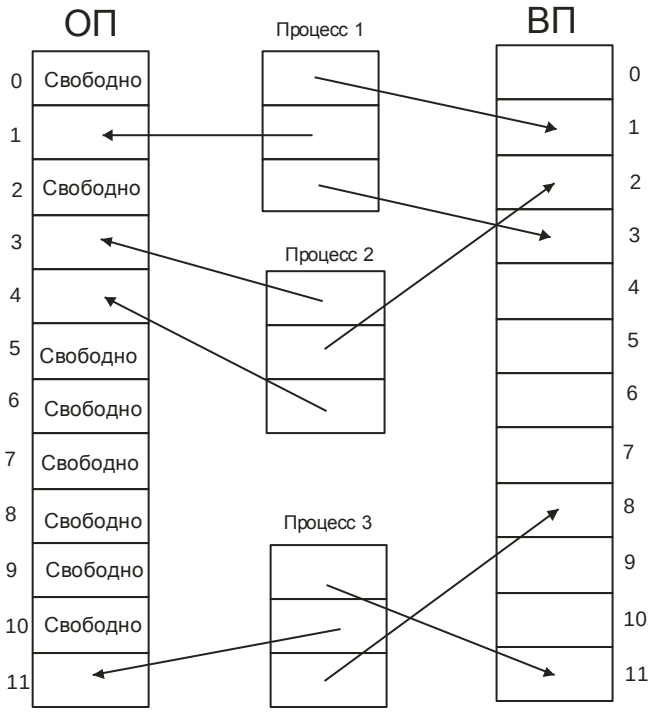
При адресации процесса используется виртуальное адресное пространство, состоящее из смежных виртуальных адресов.

Физ. адресное пространство – ФАП:
ОП – ФА
ВП – ВФА

ВА:	Номер виртуальной страницы	Смещение
ФА:	Номер физической страницы	Смещение

Проблема отображения адресов

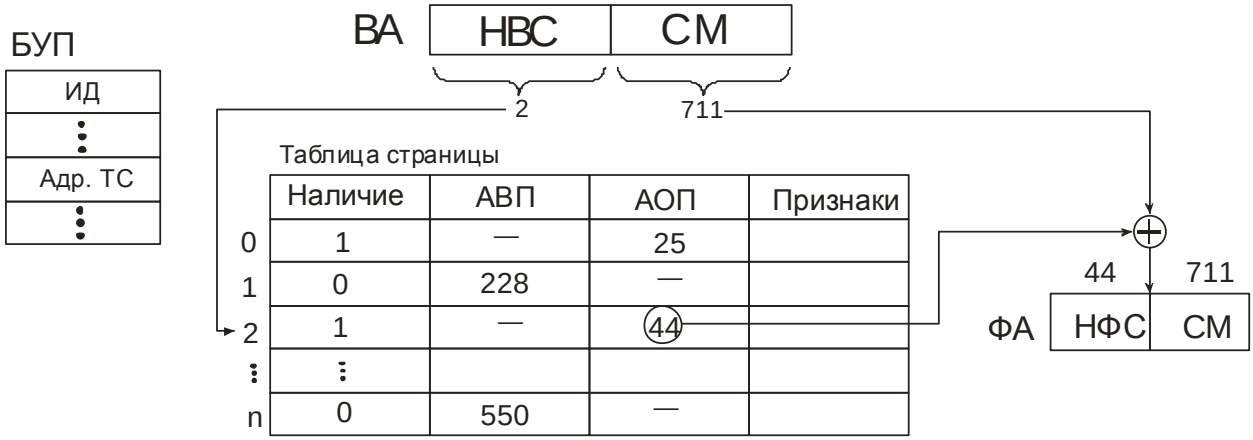
Предполагают, что произвольная ВС может находиться в ВФА



Отображение адресов

Для этого используется таблица страниц процесса

ИД – идентификатор
АТС – адрес таблицы страниц
АВП – адрес внешней памяти



Комментарии:

1. если страница отсутствует в ОП, то возникает прерывание процесса по отсутствию страницы, запускается системный процесс, загружающий соответствующую страницу в ОП (подкачка страниц).
2. в обычном случае однократное обращение к виртуальной памяти вызывает двукратное обращение к физической памяти:
   - первое обращение – к таблице страниц
   - второе обращение – к физическому адресу (ФА)

Кроме таблицы страниц процессов ОС использует собственные таблицы свободных/занятых страниц ОП и ВП (внешней)

Пример таблицы свободной/занятой страниц ОП

	Признак занятости	Дополнительный признак
0	1	
1	1	
2	0	
	…	
M	0	

Таблица свободных страниц ВП аналогична (строки 0 и 1- заняты; 2 и M - свободны)

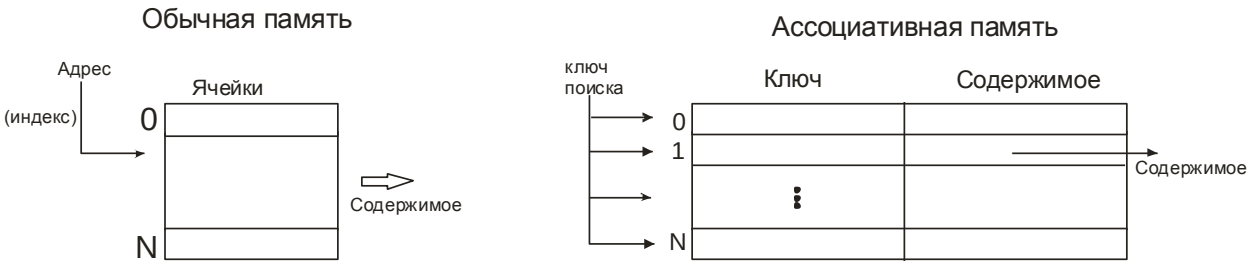
Использование таблиц свободных/занятых страниц:

1. При загрузке страниц – страница загружается в 1-ю свободную страницу ОП
2. При вытеснении страниц – если все заняты
3. При освобождении страницы – обнуление признака

Отображение страниц с помощью ассоциативных регистров

Чтобы избежать повторного обращения к ФП наиболее часто используемые страницы (информацию о них) размещают на быстрых регистрах процессов, имеющих ассоциативную организацию.

Ассоциативная память

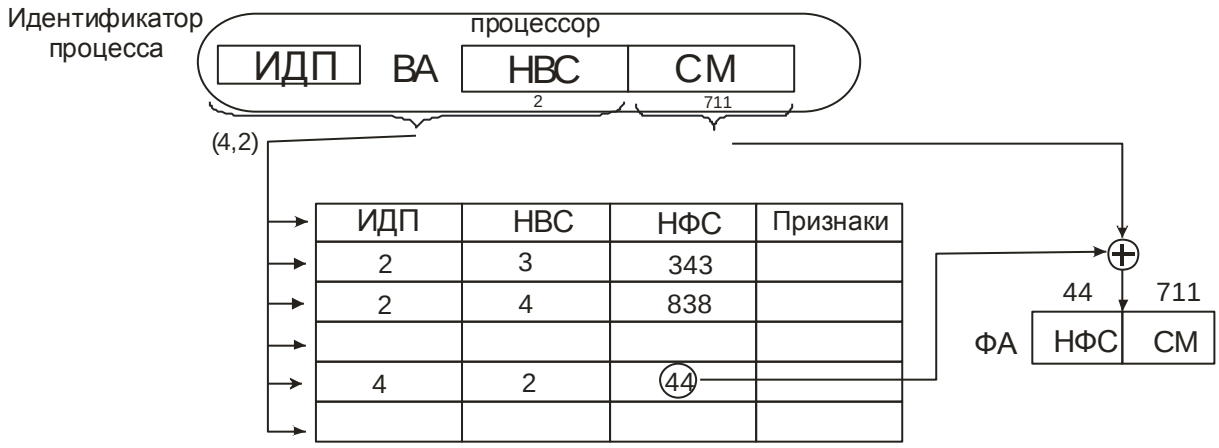


Если индекс неизвестен, то поиск – последовательная проверка содержимого. Это занимает N/2 тактов

Ключ 2 – информация для поиска

Обращение ко всем параллельное. Где ключи совпадают – выдается содержимое. Поиск – 1 такт.

Ассоциативные страничные регистры



Примечания:

1. На ассоциативных регистрах храниться информация о страницах, присутствующих в ОП.
2. При отсутствии страницы в ассоциативной памяти выполняется обращение к таблице страниц процесса.
3. Обращение на шаге 2 к таблице страниц выполняется, если размер ассоциативной памяти меньше количества страниц физической памяти.

Стратегии управления страничной памятью

Выделение:

- по запросу
- упреждающее

Размещение – первая свободная страница

Вытеснение

Упреждающее выделение – выделение страницы, если ее еще не запрашивали

1. При начальной загрузке загружают К-первых страниц
2. Вместо запуска холостого процесса (следующие страницы подкачиваются)

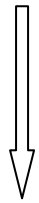
Стратегии вытеснения:

- **FIFO** – вытесняет, какая первая пришла
- **Случайную**
- **LRU** – вытесняет страницы, к которой было самое позднее обращение

Точная реализация LRU требует, чтобы «признаки» содержали время последнего обращения («временной штамп»).

Недостаток – большой объем «временных штампов» (больше 8 бит). Поэтому применяется упрощенный LRU, использующий 2 бита признаков (бит обращения и бит модификации).

Бит обращения	Бит модификации
0	0
0	1
1	0
1	1

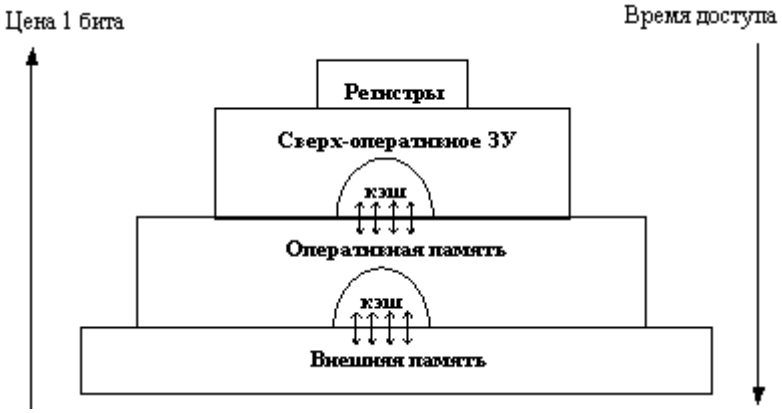


Приоритет вытеснения страниц

Биты обращения периодически сбрасываются ОС с интервалом Δt (бит обращения = 0, бит мод. = 1)

4. Иерархия запоминающих устройств. Принцип кэширования данных

Память вычислительной машины представляет собой иерархию запоминающих устройств (внутренние регистры процессора, различные типы сверхоперативной и оперативной памяти, диски, ленты), отличающихся средним временем доступа и стоимостью хранения данных в расчете на один бит. Пользователю хотелось бы иметь и недорогую и быструю память. Кэш-память представляет некоторое компромиссное решение этой проблемы.



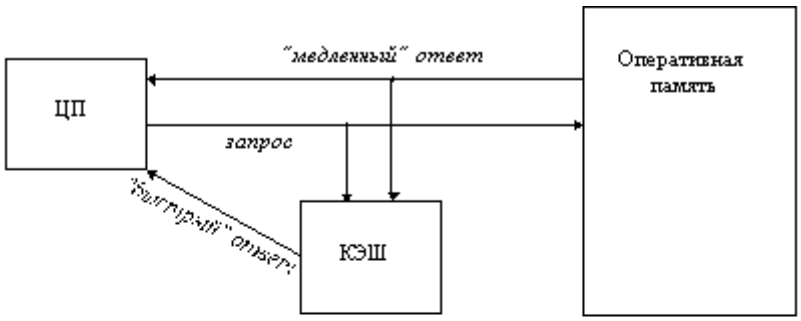
Иерархия ЗУ

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-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.

Рассмотрим частный случай использования кэш-памяти для уменьшения среднего времени доступа к данным, хранящимся в оперативной памяти. Для этого между процессором и оперативной памятью помещается быстрое ЗУ, называемое просто кэш-памятью. В качестве такового может быть использована, например, ассоциативная память. Содержимое кэш-памяти представляет собой совокупность записей обо всех загруженных в нее элементах данных. Каждая запись об элементе данных включает в себя адрес, который этот элемент данных имеет в оперативной памяти, и управляющую информацию: признак модификации и признак обращения к данным за некоторый последний период времени.



Структура кэш-памяти

Адрес данных в ОП	Данные	Управл. информация	
		бит модиф.	бит обращ.

В системах, оснащенных кэш-памятью, каждый запрос к оперативной памяти выполняется в соответствии со следующим алгоритмом:

1. Просматривается содержимое кэш-памяти с целью определения, не находятся ли нужные данные в кэш-памяти; кэш-память не является адресуемой, поэтому поиск нужных данных осуществляется по содержимому - значению поля "адрес в оперативной памяти", взятому из запроса.
2. Если данные обнаруживаются в кэш-памяти, то они считываются из нее, и результат передается в процессор.
3. Если нужных данных нет, то они вместе со своим адресом копируются из оперативной памяти в кэш-память, и результат выполнения запроса передается в процессор. При копировании данных может оказаться, что в кэш-памяти нет свободного места, тогда выбираются данные, к которым в последний период было меньше всего обращений, для вытеснения из кэш-памяти. Если вытесняемые данные были модифицированы за время нахождения в кэш-памяти, то они переписываются в оперативную память. Если же эти данные не были модифицированы, то их место в кэш-памяти объявляется свободным.

На практике в кэш-память считывается не один элемент данных, к которому произошло обращение, а целый блок данных, это увеличивает вероятность так называемого "попадания в кэш", то есть нахождения нужных данных в кэш-памяти.

Покажем, как среднее время доступа к данным зависит от вероятности попадания в кэш. Пусть имеется основное запоминающие устройство со средним временем доступа к данным t1 и кэш-память, имеющая время доступа t2, очевидно, что t2<t1. Обозначим через t среднее время доступа к данным в системе с кэш-памятью, а через p - вероятность попадания в кэш. По формуле полной вероятности имеем:

$$t = t1((1 - p) + t2(p)$$

Из нее видно, что среднее время доступа к данным в системе с кэш-памятью линейно зависит от вероятности попадания в кэш и изменяется от среднего времени доступа в основное ЗУ (при p=0) до среднего времени доступа непосредственно в кэш-память (при p=1).

В реальных системах вероятность попадания в кэш составляет примерно 0,9. Высокое значение вероятности нахождения данных в кэш-памяти связано с наличием у данных объективных свойств: пространственной и временной локальности.

- *Пространственная локальность*. Если произошло обращение по некоторому адресу, то с высокой степенью вероятности в ближайшее время произойдет обращение к соседним адресам.
- *Временная локальность*. Если произошло обращение по некоторому адресу, то следующее обращение по этому же адресу с большой вероятностью произойдет в ближайшее время.

Все предыдущие рассуждения справедливы и для других пар запоминающих устройств, например, для оперативной памяти и внешней памяти. В этом случае уменьшается среднее время доступа к данным, расположенным на диске, и роль кэш-памяти выполняет буфер в оперативной памяти.

IV. УПРАВЛЕНИЕ УСТРОЙСТВАМИ

Фактическое выполнение операций ввода-вывода для внешних устройств реализуется специальными модулями ОС, именуемыми *драйверами*. Ввиду существенных отличий внешних устройств каждый из типов внешних устройств, подключенных к ВС, имеет свой собственный драйвер в составе ОС. Кроме того, значительно отличаются драйверы устройств с блочным (МД) либо байтовым (клавиатура) вводом-выводом. В своей работе драйверы используют физические регистры контроллеров внешних устройств либо канальные программы (при подключении устройств посредством каналов), а также аппаратные прерывания внешних устройств. Драйверы, как правило, имеют несколько точек входа и одна из них – по аппаратному прерыванию. Подробное изучение организации драйверов выходит за рамки настоящего пособия.

В базе данных ОС каждое устройство представлено специальным блоком управления, в котором хранится его описание. Кроме того, создаются блоки управления контроллером (каналом), которые содержат указатели на блоки управления подключенных к ним устройств и рабочие данные текущей операции.

1. Классификация внешних устройств

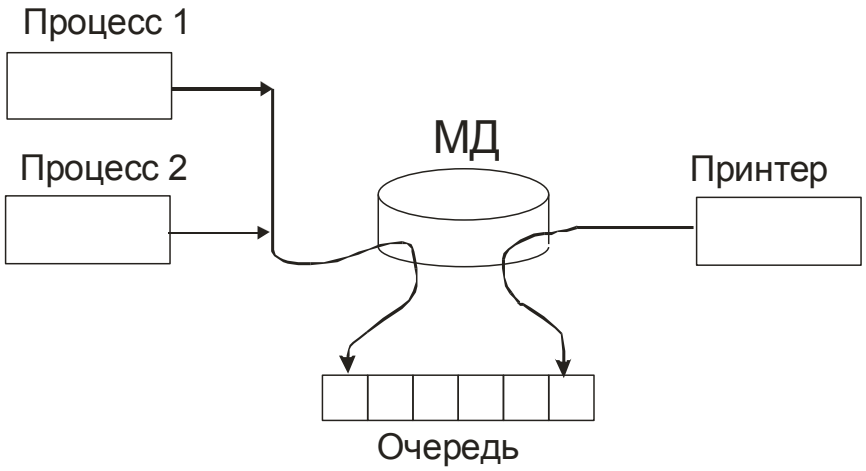
- байт-ориентированные, блок-ориентированные

Обмен – посимвольно блоками (массив, байт)

Пример байт-ориентированного – клавиатура
блок-ориентированного – МД
Разделяемые – допускает совместное использование. Применение – МД
Выделяемые – устройство захватывается процессом, выполняется последовательность операций, устройство освобождается. Пример – магнитная лента

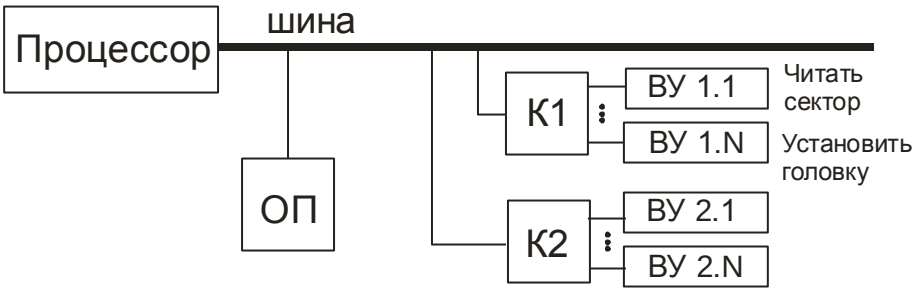
По скорости выполнения операций:
- **низкоскоростные** (печатающее устройство)
- **высокоскоростные** – МД

Пример согласования скоростей – печатающее устройство использует метод ввода/вывода – *спулинг*.



2. Подключение внешних устройств: контроллеры и каналы

1. Подключение с помощью контроллеров



К - контроллер
ОП - оперативная память

Контроллер – аппаратное средство для управления группы однотипных устройств (Пример – контроллер МД)

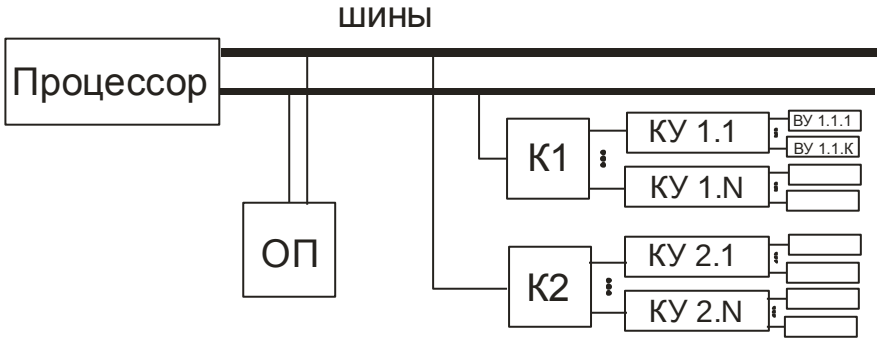
Как правило, только одно устройство, подключаемое к контроллеру может выполнять обмен данными, остальные могут выполнять операции, несвязанные с обменом данными.

Контроллер имеет представление в компьютере в виде набора управляющих регистров. Иногда эти регистры называют портами.

Типовая структура регистров контроллера

- регистр состояния ВУ
- регистр команд ВУ
- регистр данных

2. С помощью каналов



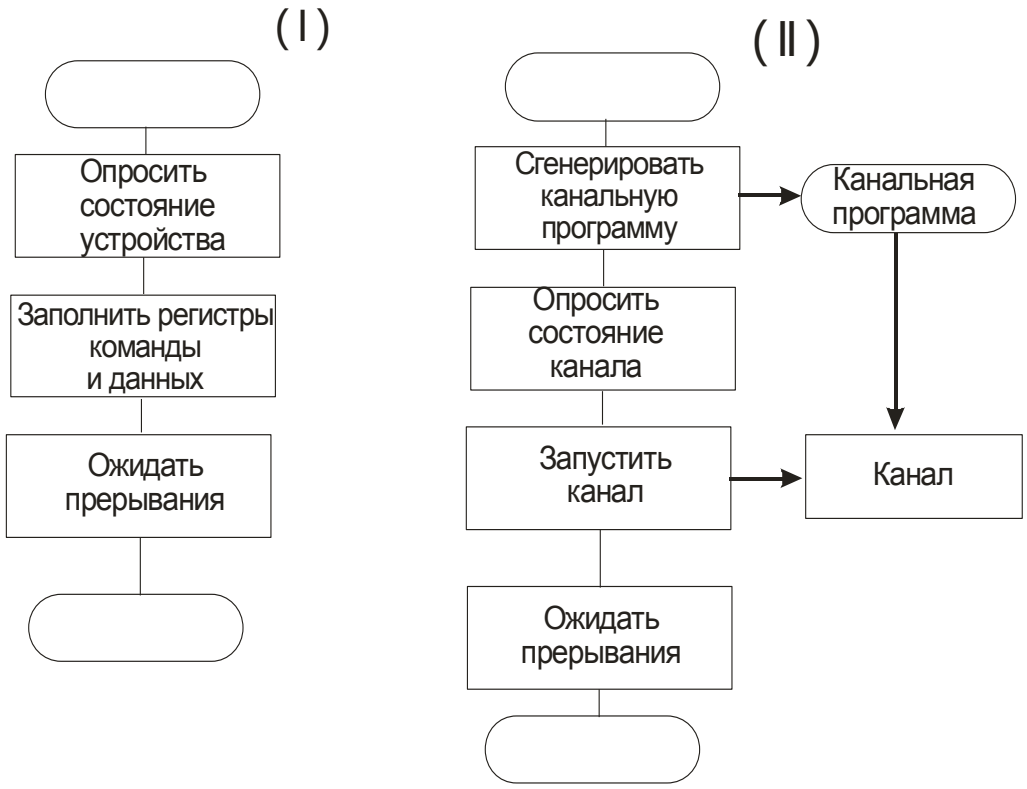
К - контроллер
ОП - оперативная память

Канал – специализированный процессор, предназначенный для выполнения программ, описывающих ввод/вывод для внешнего устройства.

Программы для каналов ввода/вывода – командные программы, используется специальный язык CCL.

Сравнительная характеристика организации ввода/вывода

Общий алгоритм управления ввода/вывода с помощью контроллера (I)
Алгоритм ввода/вывода для канала (II)



Контроллер выполняет одну команду.
Канал выполняет последовательность команд ввода/вывода.

Упрощенным устройством для подключения внешних устройств является контроллер прямого доступа к памяти (ПДП).

3. Алгоритмы работы драйверов внешних устройств

Синхронная и асинхронная операция ввода/вывода

Драйвер – модуль ОС, непосредственно управляющий работой внешнего устройства.

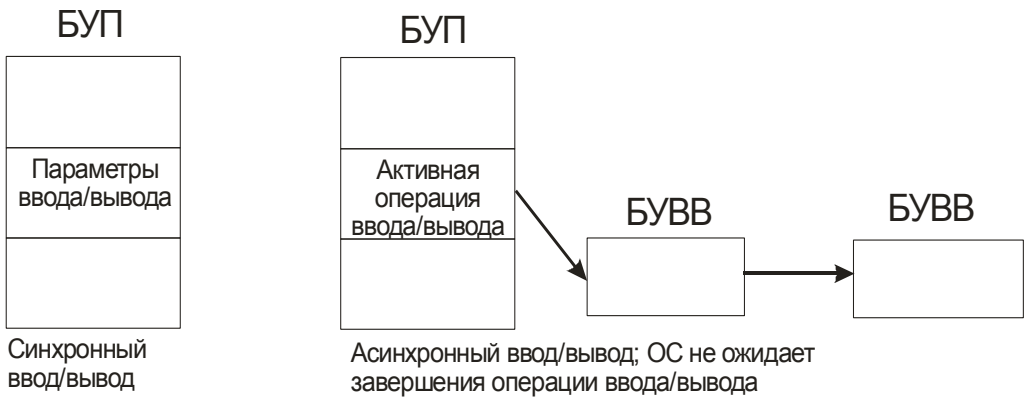
Драйвер – единственная программа, которая знает организацию регистра контроллера внешнего устройства.

Драйвер имеет не менее 2х точек входа:
- запуск операции ввода/вывода
- прерывание внешнего устройства

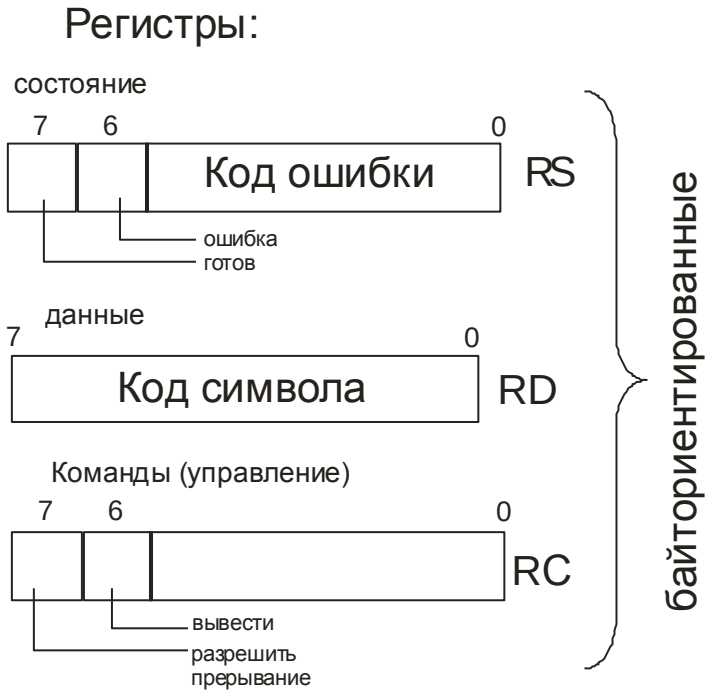
Для организации работы используют управляющие блоки:
- блок управления устройством (**БУУ**)
- блок управления контроллером (**БУК**)

Оба блока содержат описание и состояние устройств.

Параметры текущей операции размещаются либо в БУП (процессом), либо в БУВВ (ввода/вывода).



Пример: Алфавитно-цифровой терминал (АЦТ), вывод последовательности символов



Алгоритм работы драйвера:

```
Function OutString(str, n)
  ;вывод строки из n-символов
  RC[7]:=0;
  ;запрещаем прерывание
  for i:=1 to n
  begin
    while RS[7]=1 do;
    ;ожидание готовности
    RD:=str[I];
    RC[6]:=1;
  End
End.
```

Был показан пример синхронного ввода/вывода.

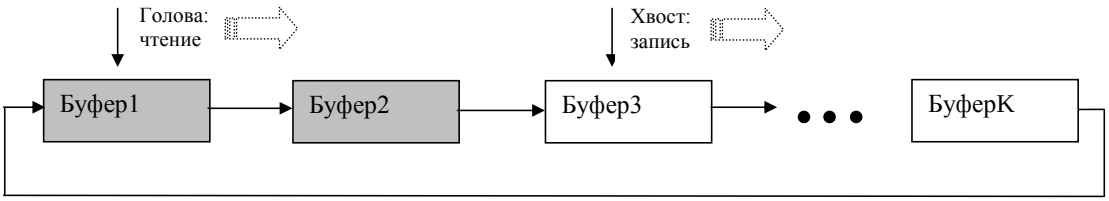
Недостаток: непроизводительное использование ЦП для готовности ВУ.

Асинхронный ввод/вывод: 2 точки входа:

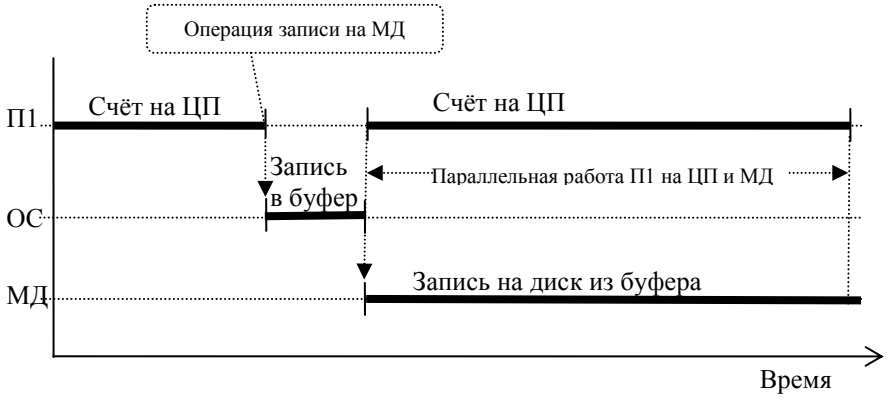
```
1.function OutStringAc(str, n);
;выделить БУВВ;
БУВВ[1]:=addr(str);
БУВВ[2]:=n;
БУВВ[3]:=1;
Текущее БУВВ:= addr(БУВВ) – сохранение адресного блока
ввода/вывода
RC[f]:=1;
Выход

2.Прерывание АУТ:
if ^(ТЕК БУВВ)[3]= ^( ТЕК БУВВ)[2]
then
begin
разблокировать БУП
Выход
end
RD:=^(^( ТЕК БУВВ)[1])[^( ТЕК БУВВ)[3]);
RC[6]:=1;
Выход
```

4. Буферизация ввода/вывода

Буферизация является стандартным средством современных ОС для согласования скоростей работы устройств ВС. *Буфер* представляет собой участок памяти для промежуточного хранения данных. Различают буферы постоянной и переменной длины. Отдельные буферы объединяются в связные списки. Наиболее распространённой является *циклическая буферизация*:



Согласование скоростей обеспечивается, например, при выводе на магнитный диск, за счёт того, что после быстрого копирования ОС выводимых данных в буфер процесс считает операцию завершённой и продолжает своё выполнение. Таким образом, фактический вывод данных на устройство выполняется *параллельно с работой процесса*:



Достаточно большой размер буфера позволяет значительно ускорить выполнение процесса. В том случае, если записанные в буфер данные считываются до их фактической записи на диск, ОС обеспечивает их быстрое извлечение из буфера. Следует отметить, что интенсивный ввод-вывод может привести к заполнению буферов и неизбежному ожиданию их освобождения, но в среднем буферизация существенно повышает эффективность работы ВС. Для *оценки эффективности* можно использовать сравнение времён работы процесса без буферизации и с использованием буферизации.

К *недостаткам буферизации* можно отнести возможность потери данных при крахах ОС; в современных надёжных ОС при использовании бесперебойного питания ВС вероятность такой ситуации сведена к минимуму. Тем не менее, большинство современных ОС предусматривают операцию принудительной записи содержимого буферов.

Для согласования скоростей и ускорения работы внешних устройств ВС используют также *кэширование*. В этом случае блоки данных размещаются в промежуточном высокоскоростном хранилище (кэше) несвязно; обеспечивается быстрый ассоциативный поиск по ключевой информации, например, номерам логических (физических) блоков внешних устройств.

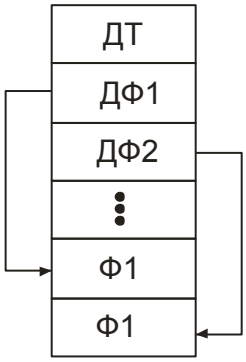
V. УПРАВЛЕНИЕ ИНФОРМАЦИЕЙ

Подсистема управления устройствами (также как и подсистема управления ОП) является защищённой и скрытой от конечного пользователя и прикладных процессов в современных ОС. Пользователю предоставляются средства управления информацией ОС с единицей хранения – *файл*. Следует отметить также тенденцию представления устройств в виде специальных файлов (например, в ОС Unix) для обеспечения ограниченного (и безопасного) доступа пользователей.

В базе данных ОС объекты файловой системы представлены блоками управления томом, каталогом, файлом, организованные в многосвязные списки. Заметим, что ОС обеспечивает «горячую» защиту информации на фактически подключенных устройствах. Безопасность в случае возможной несанкционированной передачи устройства (диска) злоумышленникам может быть обеспечена только при использовании дополнительных возможностей шифрования хранимой информации.

1. Планирование пространства тома

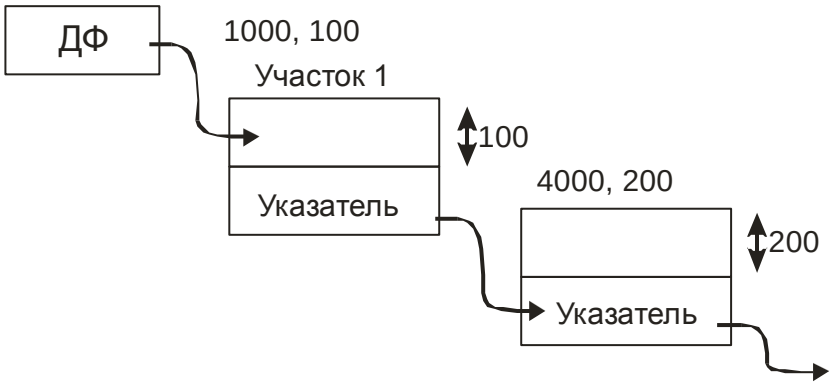
1. Связное выделение



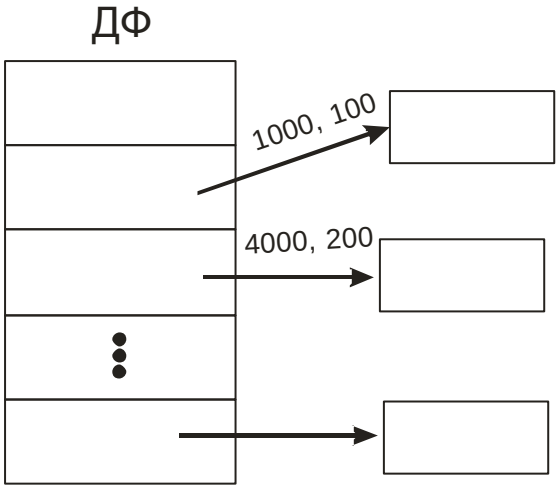
ДТ – дескриптор тома
ДФ – дескриптор файла

В случае связного распределения на пространство выделяется файлу одним непрерывным участком
Недостаток – необходимость перемещения при возникновении фрагментации

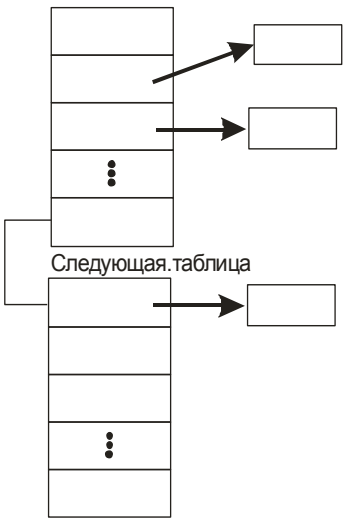
2. Несвязное распределение



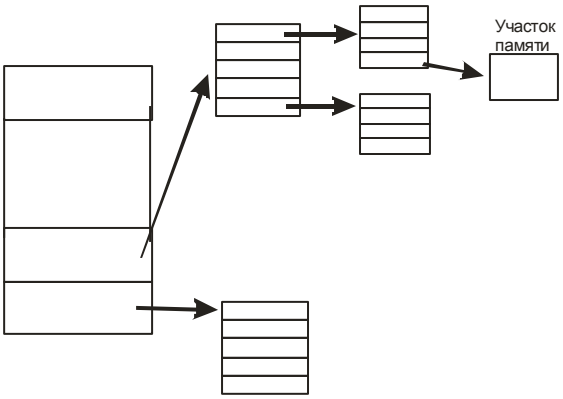
Модификация несвязного распределения – таблица указателей устройств.



Проблема – фиксированный размер таблицы.
а) дополнительная таблица



б) многоуровневая таблица



Для учета свободного дискового пространства применяются:

1. Битовые кадры

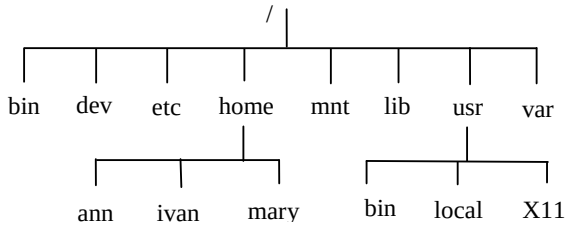
0	0	1	-

1- занято
бит соответствует сектору

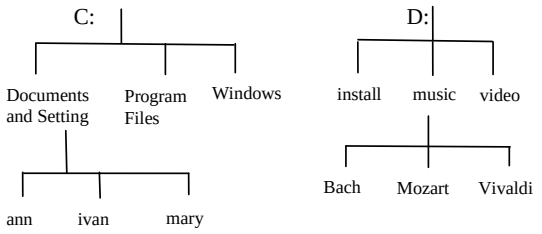
2. специальные системные файлы
free_blk – занимает все свободные кластеры диска.

2. Файловая структура диска

Древовидная (иерархическая) файловая система, образованная такими элементами как *том, каталог, файл,* является стандартной для современных ОС. Файл представляет собой поименованную единицу хранения информации; файлы объединяются в каталоги, причём каталог может содержать как файлы, так и другие каталоги; том представляет собой устройство с файловой системой. Единственная разница между семействами доминирующих в настоящее время ОС Unix и MS Windows состоит в представлении томов: Unix использует общую иерархию всех томов, MS Windows представляет файловую систему с разбивкой по томам (устройствам):



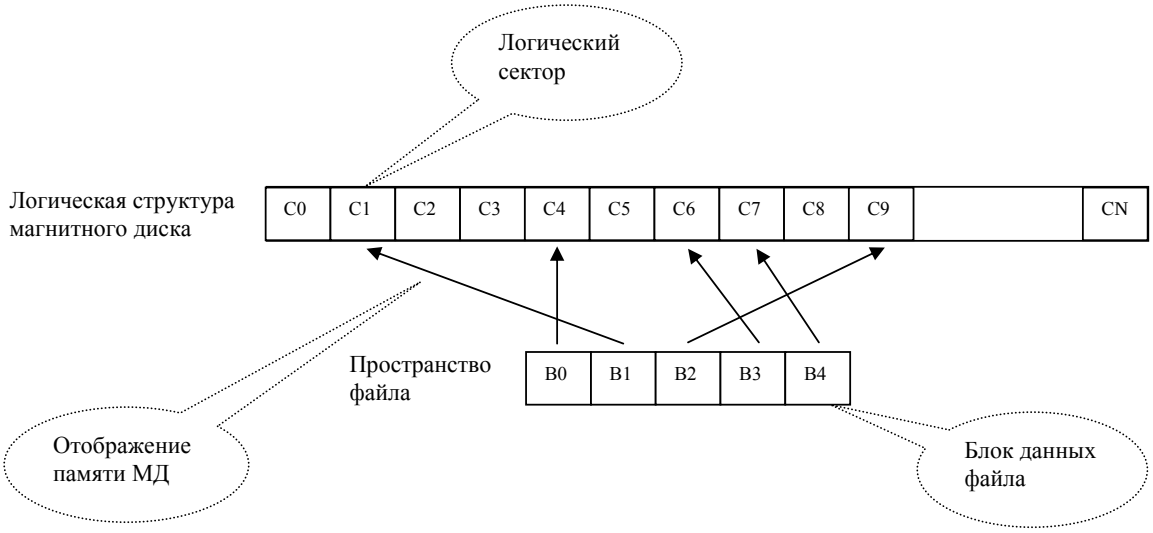
Unix



MS Windows

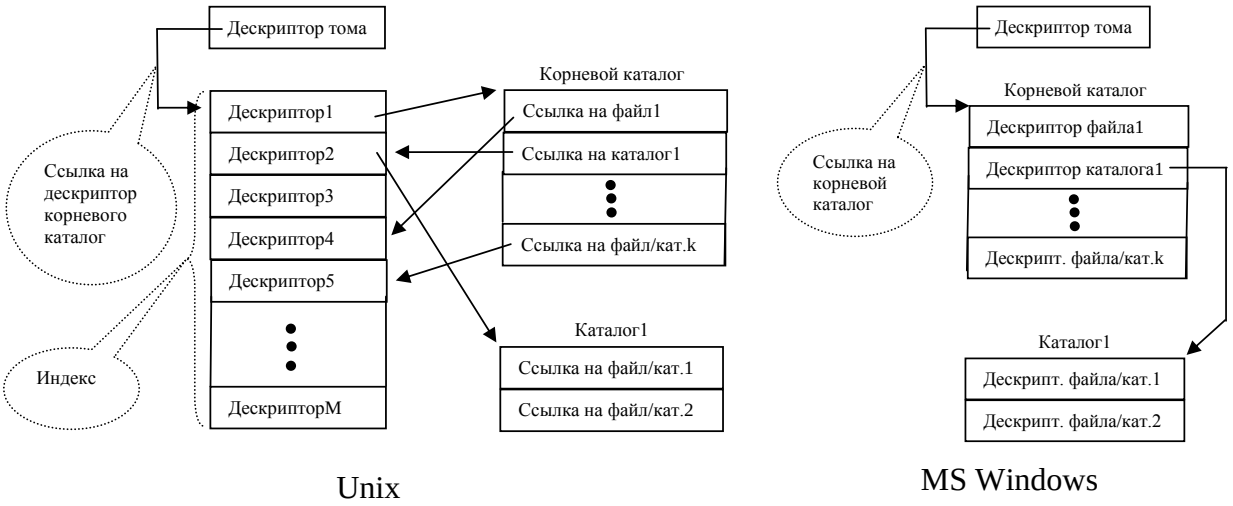
Как правило, современные ОС реализуют такие основные файловые операции как чтение-запись и позиционирование внутри файла, обеспечивая *последовательный и прямой доступ* к информации. Более сложные методы доступа реализуются СУБД на основе указанных операций ОС.

Логическая структура диска может быть представлена одномерным массивом секторов образованным занумерованными физическими секторами всех поверхностей, цилиндров и дорожек. Возникает *проблема отображения* файловой системы на логическую структуру диска для обеспечения доступа к файлам:



Для обеспечения отображения создаются специальные структуры данных, описывающие отображение и невидимые для конечного пользователя. Способ организации таких данных принято называть *файловой структурой*. Файловая структура обеспечивает решение двух основных задач: нахождение логических секторов заданного файла/каталога на диске; учёт свободных/занятых секторов.

Большинство файловых структур использует специальный блок – *дескриптор* для каждого из своих объектов: том, каталог, файл. Дескриптор файла/каталога содержит имя, владельца, права доступа, даты создания/корректировки, длину. Дескриптор тома содержит также указатель на дескриптор корневого каталога. Дескрипторы файла/каталога имеют фиксированную длину. ОС используют два основных похода к размещению дескрипторов: в ОС семейства Unix дескрипторы файлов/каталогов размещаются в общем одномерном массиве – *индексе*, в этом случае каталог содержит список указателей на дескрипторы его файлов/каталогов; в ОС семейства MS Windows дескрипторы файлов/каталогов размещаются внутри содержащего их каталога:



При решении задачи отображения для уменьшения размера структур данных применяют *кластеризацию*. Кластер состоит из нескольких блоков (секторов) и является единицей выделения пространства диска. В настоящее время доминируют два подхода к описанию размещения файла на диске: использование общей таблицы размещения для всех кластеров (в ОС семейства MS Windows); использование массива описателей связных участков в дескрипторе файла (в ОС семейства Unix).

Общая таблица размещения содержит по одной записи для каждого кластера. Запись равняется нулю, если соответствующий кластер свободен. Если кластер принадлежит некоторому файлу, то соответствующая запись содержит номер следующего кластера; запись последнего кластера файла имеет специальное значение – признак конца цепочки (например, все двоичные единицы).

Использование *таблицы связных участков*, описанных указанием начального кластера участка и длины в дескрипторе файла, обеспечивает более быстрое отображение. Однако существенным недостатком этого метода является фиксированное число участков в дескрипторе файла. Размещение больших фрагментированных файлов решается в ОС Unix за счёт выделения дополнительных дескрипторов.

Таблица размещения файлов
(File Allocation Table)

0	1	2	3	4	5	6	7	8	9
0	0	3	4	8	0	-	0	9	6

Таблица связных участков
(в дескрипторе)

Нач.	2	8	6
Дл.	3	2	1

Использование единой таблицы размещения обеспечивает также решение задачи *учёта свободных/занятых кластеров*: записи свободных кластеров имеют нулевое значение в таблице. К дополнительным способам учёта свободного/занятого пространства можно отнести битовые карты и специальный (фиктивный) файл, занимающий все свободные кластеры. В битовой карте каждый бит соответствует кластеру; значение 0 – кластер свободен, значение 1 – занят. Применение специального файла обеспечивает преимущества быстрого поиска наиболее подходящего участка при выделении пространства и способствует уменьшению фрагментации.

VI. ОС UNIX

Unix является стандартом де-факто современной сетевой операционной системы. Он был создан как первая сетевая ОС общего назначения в 1969 – 1971 гг. в лаборатории Bell Labs, фирмы AT&T.

В середине 70-х была выполнена первая реализация стека протоколов TCP/IP в UNIX.

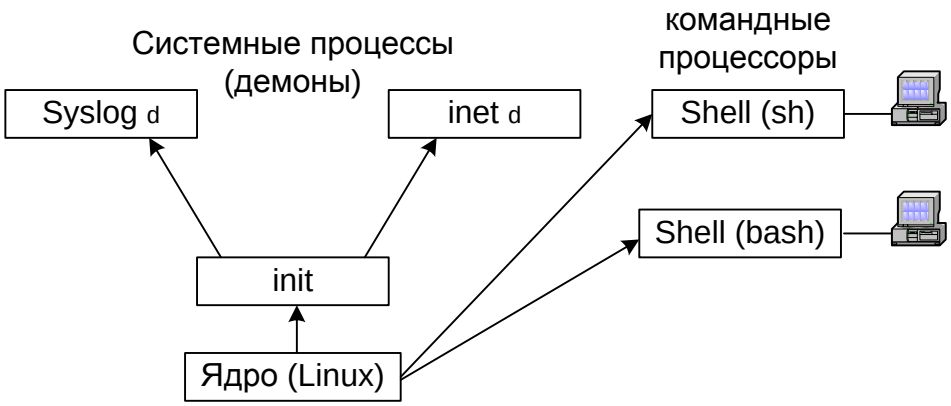
В 80-х создана графическая оболочка XWindows.

В 1991 году появилась система Linux – наиболее популярная в настоящее время ОС семейства Unix.

Существуют:

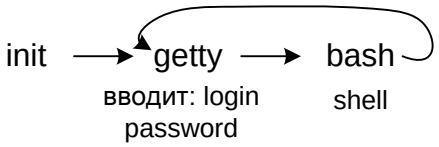
- коммерческие реализации: Solaris, HP-UX, IPTX, AIX.
- свободно распространяющиеся системы: LINUX, Free BSD.

1. Общая организация работы ОС Unix



Ядро не является процессом в Unix. После загрузки в ОП ядро создаёт начальный процесс Init, который является предком всех остальных процессов системы.

Для диалогового взаимодействия с пользователем init периодически запускает командные процессоры (оболочки shell):



init – «родитель» всех процессов.
getty порождает login.

Сначала init запускает на каждый подключенный терминал системный процесс getty, который вводит и проверяет имя пользователя и пароль, а затем запускает на терминале командный процессор. Командный процессор вводит и интерпретирует команды пользователя и позволяет также для графических терминалов запустить графическую командную оболочку KDE, Gnome в среде XWindows.

Командные процессоры:
sh
ksh
csh
bash
отличие, синтаксис, команды и специальные ключи управления.

Алгоритм работы командного процессора

Пользователи и группы

Пользователь идентифицируется своим именем, подтверждённым вводом пароля; кроме того, для краткого обозначения каждый пользователь имеет свой идентификатор в виде целого числа UID.

Изменить пароль можно с помощью password

Информация о пользователях хранится в файле /etc/password/

Пользователь является членом одной или нескольких групп.

Пользователь → группа

/etc/group – список групп хранится в этом файле.

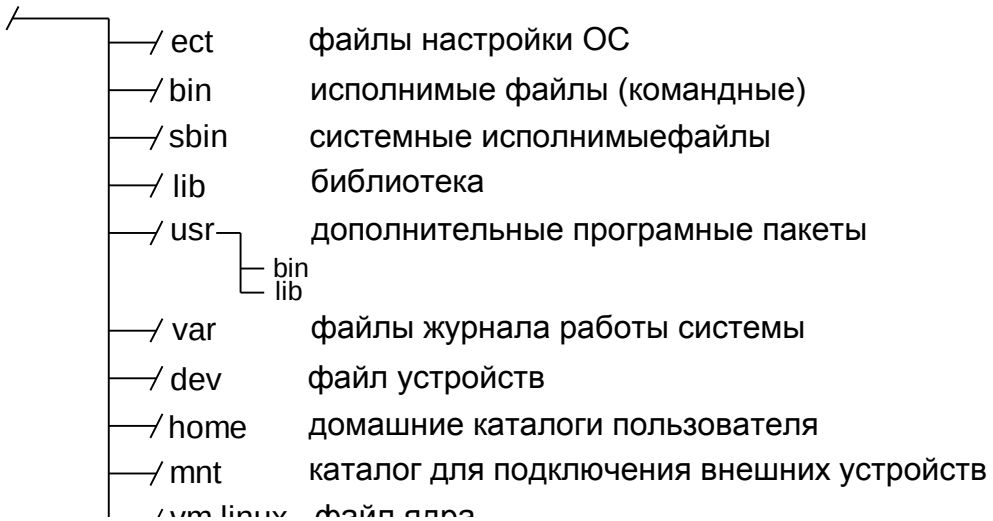
1: 25: user 1, user 2, user 3…

Группа задаётся именем; кроме того, для краткого обозначения каждая группа имеет свой идентификатор в виде целого числа GID.

Организация защиты в ОС Unix основана на действиях суперпользователя – root – наделённого неограниченными полномочиями; суперпользователем, как правило, является администратор ОС.

2. Файловая система UNIX

ОС Unix имеет древовидную файловую систему с единой иерархией; подключение файловых систем отдельных томов и их отображение на каталоги файловой системы контролируется администратором с помощью специальных команд монтирования/демонтирования тома (mount/umount). Рассмотрим основные системные каталоги:



Файл (каталог):

- собственник (пользователь)
- группа пользователей
- права доступа
- дата, время, время последней корректировки
- размер
- имя

Права доступа:

	R (чтение)	W (запись)	E (запуск)
U (user)	-	W	-
G (grope)	R	-	E
O (other)	-	-	E

При распечатке шифр прав записывается в строке: p – w – r – e – – e

u g o

p – префикс типа записи:
– обычный файл (–)
– каталог (d)

Команды указания модификации прав:

$whoam – текущее имя группы;
$newgrp <имя группы> – изменить текущую группу пользователей
$chown <имя пользователя> <имя файла> – изменение собственника
$chgrp <имя группы> <имя файла> – изменение группы
$cnmod <права> <имя файла> – изменение прав доступа к текущему файлу

права: одно число в восьмеричной системе счисления, например:

– w – r – e – – e
0 1 0 1 0 1 0 0 1
2 5 1 = 251 – задание прав.

Добавить/удалить.
U+W – добавление права на запись собственника.
go - e – отображать права на исполнение у группы и у всех остальных.
<кому> ± <что>
a - rwx – запрет всякого доступа всех и себя.

Имена файлов

– абсолютные – полный перечень имён от корневого каталога.
Например: /home/kulikov/text/ book1.txt
каталоги имя файла
Абсолютное имя однозначно идентифицирует файл в системе Unix.
– относительное – от текущего каталога.
Например: $pwd – указывает текущий каталог
/home/kulikov
text/book1.txt
– псевдоимена
специальные символы: .– текстовый каталог
.. – вышестоящий каталог
$HOME – имя домашнего каталог пользователя
– скрытые файлы: имя начинается с «.» для распечатки их используют специальные ключи.

Работа с файлами:

(Команды имеют формат: <имя команды> [ключи] [операнды])

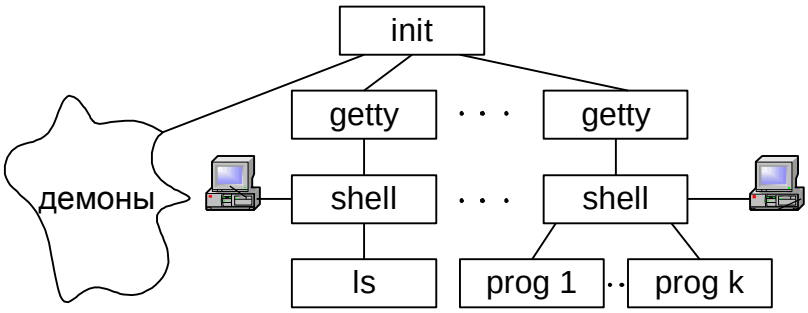
$ls – вывести список файлов текущего каталога.
$ ls -l – печать атрибутов.
$ ls -a – вывод скрытых файлов.
$ ls -al – печать атрибутов и вывод скрытых файлов.

$man <имя команды> – получение справки.
$cat имя – вывод на экран файла, конкатенация файлов
$cp имя1имя2 – копирование файла

$mv имя1имя2 – перемещение, переименование файла
$rm имя – удаление файла
$mkdir – создать каталог
$rmdir – удалить каталог
$pwd – вывести имя текущего каталога
$cd имя – изменить текущий каталог.

3. Управление процессами в Unix

В ОС Unix множество процессов организовано иерархически на основе отношения «родитель – потомок». Ядро не является процессом. Прародитель всех процессов – процесс init.



Атрибуты процесса:

- ресурсы ОГ
- идентификатор PID
- идентификатор родительского процесса PPID
- собственник (пользователь) UID
- группа GID
- начальный приоритет PRF
- модификатор приоритета NI
- состояние процессов
- время использования процессора

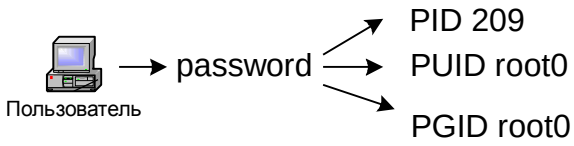
PUID: правила формирования:
PGID
1. Права пользователя, запустившего процесс

	user	group
prog1	root	root

<имя пользователя> prog1 (PID) 205
<имя группы> (PUID) <имя пользователя>
(PGID) <имя группы>

Для установления бита перезаписи прав (8-бит): процесс получает права владельца исполнимого файла.
/etc/password – только чтение
$ password – можно изменить пароль:

$ password root root s r w x r – x r – x
user group



В системе Unix используются динамические приоритеты. Первоначальные приоритеты при этом задаются пользователем, либо по умолчанию. Пользователь не может назначить уровень приоритета выше назначаемого администратором.

Модификатор назначается ОС по результатам наблюдения за поведением процесса.

Наблюдение за поведением процесса:

Актуальный приоритет = первоначальный +модификатор.

STATE
R – готов к выполнению
W – выгружен
B – заблокирован
Z – зомби (уничтожение, не остающийся в системе процесс)

Команды работы с процессом:

$ имя – породить процесс, для выполнения файла с указанным именем.

имя:
– абсолютное имя /…/…/файл
– относительное имя:
– префикс относительного имени
– PATM префикс.

Префикс: ../prog1 – взять prog1 из вышестоящего файла
../../prog2 – prog2 на уровень выше

PATH – префикс: PATH=.;/bin/usr/bin

$ prog1 – поиск следующих окон:
- ./prog1
- /bin/prog1
- /bin/usr/prog1

$ prog2
- unknounfile $echo$PATH – (результат) .;/bin/usr/bin/host1 user2>
$echo PATH – (результат) PATH

Для ключевых команд администрирования желательно использовать абсолютное имя файла, т.к. распространённым видом является подмена префикса и исполнимого файла.

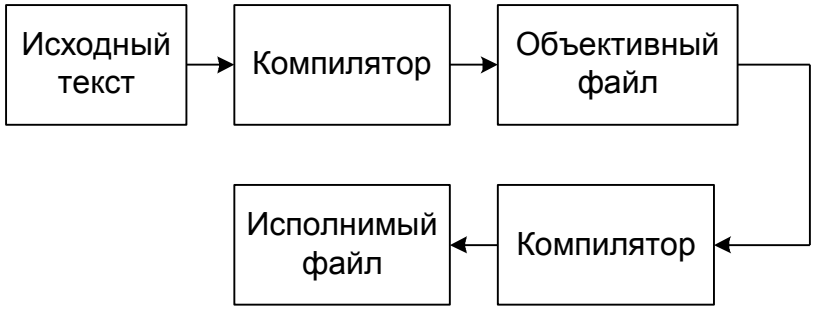
Пример:
/sbir/ifconfig – конфигурация сетевых интерфейсов; злоумышленник подменяет ifconfig, указав PATH=/Mydir;
Администратор вводит команду ifconfig и запускает вредоносный файл, находящийся в /Mydir.

Появляются процессы с правами root (PUID root).

$ps – вывести список процессов.
Как правило, выводятся процессы текущего пользователя: PID 213; TIME: 00:00; NAME: PS.

$ps -e – выводит список всех процессов
$ps -l – подробная информация (большинство атрибутов).
$ps -le – подробный вывод информации обо всех процессах (комбинация 1 или 2)
$ps –le|more (| – конвейер)

$имя & – запуск процесса без ожидания завершения – параллельный запуск процесса.
Примечание:
Тип запускаемого файла:
- двоичный используется файл в формате ОС.
- командный файл shell (текст)



Примечание: у запускаемого файла д.б. установлен атрибут, позволяющий его исполнение текущему пользователю.
Пример:

	user	group	name
– r w x z – – – – –	petia	svoi	grop1

$chmod U+X proge –право их исполнения – запуск файла.

$kill big PID – послать сигнал процессу с указанным идентификатором PID.
$kill -9 PID – сигнал принудительного завершения указанного процесса.
$kill -1 PID – сбросить установку текущего процесса.
$kill HUP – повторить запуск.

Примечание:
При посылке сигналов сопоставляются права процесса и пользователя.

$nice PID priotiry – изменён приоритет указанного процесса
renice

Комментарий: значение приоритета изменено в приделах модификатора. В большинстве систем Unix, значение приоритета не может превышать максимальный установленный для класса пользовательских процессов.

$nohup <command> – запустить программу в непрерываемом режиме.

Комментарии: в качестве команды указывается простая команда, либо командный файл; команда, запущенная с помощью nohup продолжает исполняться по завершению родительского процесса. По завершению Shell завершились все процессы, кроме nohup. Используют (nohup) для запуска файловых процессов на длительные периоды. Файловые процессы используют файловые ввод/вывод.

$at <time> <command> – запустить команду в указанное время; не требует наличие пользователя в момент запуска.

4. Программирование на языке командного процессора (Shell)

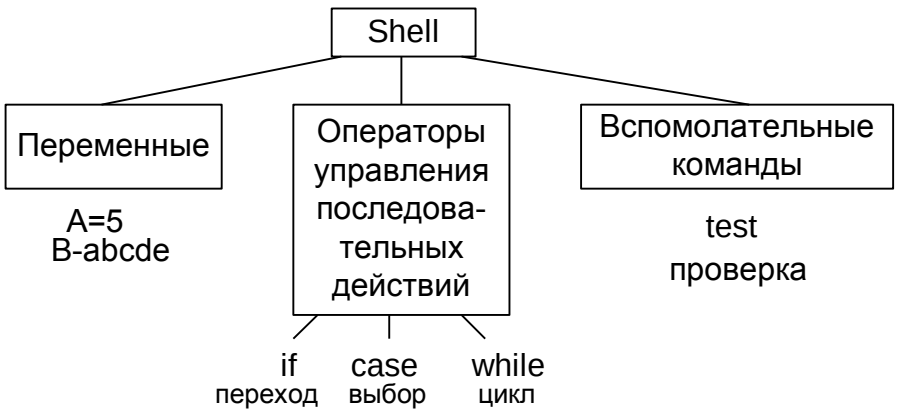
Последовательность команд Shell можно записать в текстовый файл, добавить атрибут разрешения запуска x и таким образом получит простейшую программу на языке Shell. Для написания более сложных программ (скриптов) используются дополнительные управляющие структуры, характерные для языков программирования: переходы, ветвления, циклы.

Язык Shell:
- файлы начального запуска Unix;
- файлы администрирования;
- командные файлы пользователей;

Bash-shell в системе Linux и Free BSD
! /bin/bash – первая строка.
/bin/sh

Способ ввода команд:
1. указание файлы с командами.
   $ prog
2. перечисление в строке;
   команда 1; команда 2;…; команда k.
3. непосредственный ввод с экрана клавиатуры
   $ if (незавершенная)
   if > (продолжение ввода на нескольких строках)

Общая характеристика средств программирования Shell



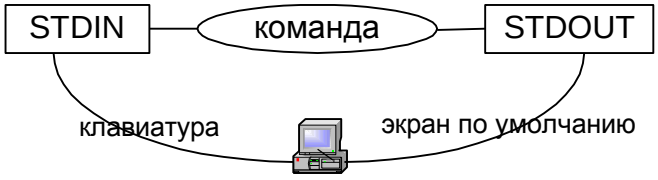
Примечание: использование кавычек в командных строках Shell:
”” – текстовая строка без интерпретации содержимого.

’ ’ – текст строки с интерпретацией содержимого.
` ` – подстановка результатов выполнения указанной команды.

$ A=abcde
$ echo “echo $A” на экране: echo $A
$ echo ‘echo $A’ на экране: echo abcde
$ echo `echo $A` на экране: abcde

Переадресация ввода-вывода

Стандартная (G) организация:

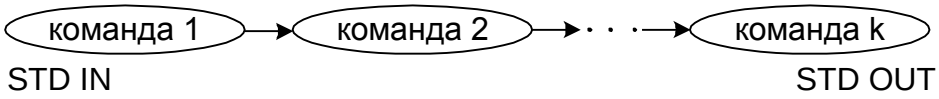


Пользователь имеет возможность указать в качестве SID IN и STD OUT свои собственные файлы команда>файл – переадресация ввода.
Результаты работы команды направляются в файл с указанным именем.
Команда<файл – переадресация ввода. Исходные данные для работы команды выбираются из файлов с указанным именем.

$ cat file1 – вывод файла на экран
$ cat file1 – file2 } cp file1 file2.
$ wc (word counter) – подсчет количества слов и строк во входном потоке.
$ wc < file2 подсчет количества слов в файле 2.

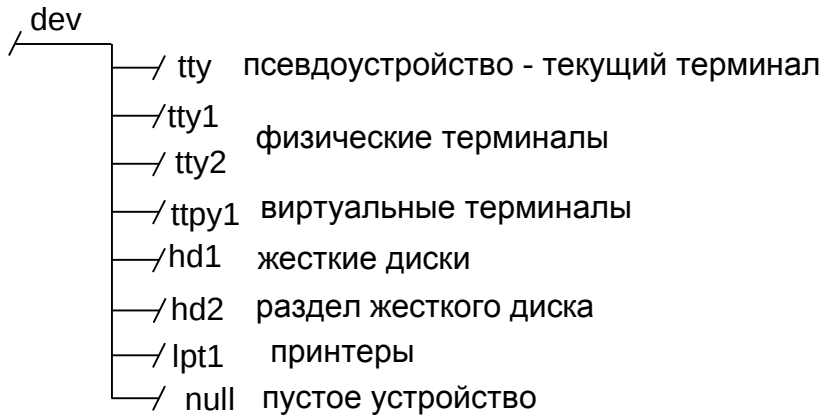
Организация конвейеров

При организации конвейера в Unix, результат работы некоторой программы направляется во входной поток следующей команды.



Символ конвейера: |
Пример: $ ls –l
$ ls –l|more – вывод по частям.
Фильтры – набор команд, предназначены для работы внутри конвейера.

more – вывод по частям
ws – подсчёт слов
sort – сортировка строк
grep – поиск внутри текста.



Примечание: как правило, фильтры имеют широкий набор дополнительных возможностей, задаваемых с помощью ключей.

$ man имя_фильтра
$ имя --help – подробная справка.

Имена устройств Unix

Каждое из устройств ОС Unix представлено специальным файлом, в каталоге /dev.

Устройство null является эффективным средством избавления от нежелательных выходных данных.

$ ls > /dev/null

Фильтры. Псевдо символы. Генерация имён файла.

/home/test/file1 – точное имя.

Назначение символов.

* – произвольная последовательность символов.
? – произвольный символ (единичный).
[abc] – множество символов
[0-9] – диапазон.

Пример: home/test/* --все файлы, которые находятся в каталоге test
beg? – все имена, которые начинаются с beg и имеют 4 символа.

Средства построения программ

– переменные.

$ имя = значение – задание переменной
$ echo $ имя – ссылка.
Предопределённые переменные.

$1 $2 … – параметры командной строки Shell
$* – все параметры
$ # – код возврата.
$$ – идентификатор предыдущего процесса.

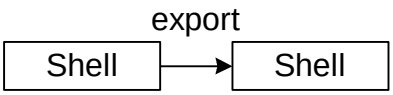
Переменные окружения (позволяет хранить какие-то параметры)

PATH – путь поиска файла.
FS1 – подписка (приглашение) основная $
FS2 – подписка (приглашение) дополнительная >
TTY – параметры терминала.
Средства указания переменной окружения.

profile – файл содержит начальные установки Shell (bash profile)
Все переменные Shell являются локальные.

Глобальные переменные:

export имя1,имя2,…



дополнительные способы указания значений переменных.
$ (имя – значение) – исполнитель указал значение, если переменная не определена, или имеет пустое значение.
$ (имя = значение) – присваивает.

Управление последовательных действий

Условный оператор.
if команда then команда2 fi
if команда then команда2 else команда fi
if команда1 then команда2 elseif команда3 then команда4 elseif команда5 … fi

комментарии:

1. Везде, где указанна команда можно, указывать последовательность команд.
2. для проверки условия имеется набор средств проверки test.

Test {ключ} file

- r – разрешено чтение
- w – разрешена запись
- x
- f – существование файла.

Проверка выражений:

test выражение1 {ключ} выражение

- eq (=)
- qt (>)
- le (<=)

Пример:

if test –f names then cat names (если существует файл names, то выводит его на экран)

```
if test $A -eq 7
  then prog1
   else prog2
fi
```

Вычислить значение алгоритма проверок:

```
$ expr ‘выражение’
A=5
B=2
$expr ‘$A+$B’
$ 7
$ c= `expr ‘$A+$B’`
$echo $c
$7
```

Операторы цикла.

for переменная in значение do команда done.

Пример:

```
$ls
file1 file2 prog
$for $c in ls
do
   echo $c
   cat $c
done
```

Дополнительные операторы цикла:

```
while команда1
   do команда2
done
```

команда2 повторяется пока код возвращения команды1 не будет равен нулю.

```
do
   команда1
until команда2 done
```

Оператор выбора.

```
case переменная
   вариант 1|команда1;
   вариант 2|команда2;
   …
   esac.
```

Редактирование текстовых файлов

$ ed – текстовый редактор командной строки
$ vi – экранный редактор (визуальный)

Редактор ed не требует специфических настроек терминального оборудования его целесообразно применять на незнакомое терминальное оборудование.
vi обеспечивает 3 режима:

1 ввод текста
2 исполнение команд редактирования
3 исполнение файловых команд

$ vi file – редактирование файла. При этом на экране появляется текст файла, по которому можно перемещаться с помощью клавишей. При этом по умолчанию времени 2
Команды редактирования.

x –удалить текущий символ,
d –удалить текущую строку,
i –вставить текст перед курсором
a –вставить текст после курсора.

Команда i и a приводят в режим 1, при нажатии ESC – перейдём в режим 2. переход из 2 в 3 – нажатие Ctrl: или Shift.

Файловые команды:
:w – сохранить файл
:q – завершить работу
:wname – сохранить файл с указанным именем
:rname – читать файл с указанным именем
q!-q – завершить работу, несмотря на то, что текст не сохранён

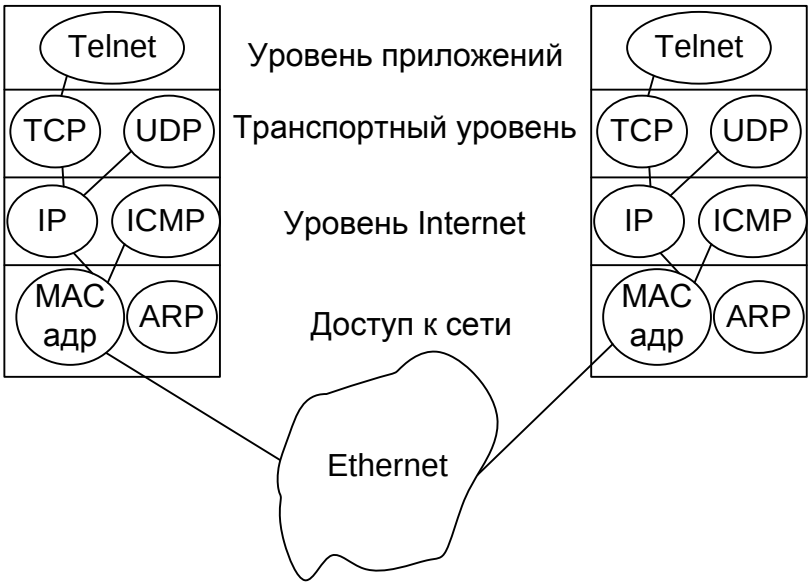
Типы терминалов.
Для работы vi используют терминалы vt100. Для установки параметров терминалов исполняют команду settee.

5. Организация доступа к сетевым ресурсам в Unix

Основные сетевые протоколы ОС Unix – TCP/IP –это семейство протоколов впервые было реализовано в ОС Unix (1976 г).

Общая характеристика протоколов семейства TCP/IP

Используется четырёхуровневая модель протокола.



TCP/IP не зависит от среды передачи данных, это решается за счёт того, что протоколы физического уровня не входят в это семейство. Поэтому в качестве физической среды могут использовать Ethernet, Token Ring, ISDN, Leased Line (выделение линии).

Особенности реализации TCP/IP для Ethernet.

В Ethernet каждое устройство имеет MAC адрес, 6-байт: 3-байта – фирма производитель, 3 – номер сети. В TCP/IP каждый хост, устройство доступа к TCP/IP имеет IP адрес.
ARP – протокол, сопоставляющий IP адресу MAC-адрес.
RARP – протокол сопоставляющий MAC-адресу IP адрес.

IP	MAC
193.63.10.11	00:00:20:if:21:31
193.63.10.15	00:00:12:22:32:40

Если требуется передать пакет по заданному IP адресу:
Протокол IP

Обеспечив доставку пакетов «хост-хост» возможно с использованием промежуточных шлюзов с помощью маршрутизаторов. Основу протокола составляет IP адрес.

IP адрес однозначно идентифицирует хост подключения к сети.

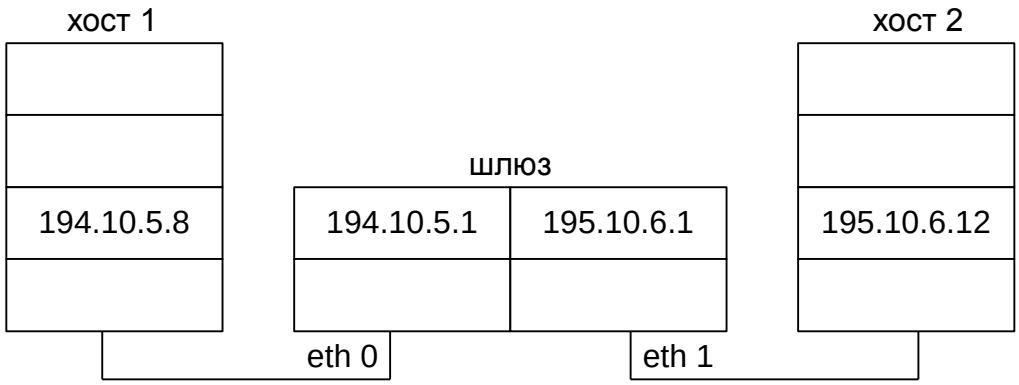


Таблица маршрутизации.

Сеть	интерфейс
194.105.0	194.10.5.1
195.106.0	195.10.6.1

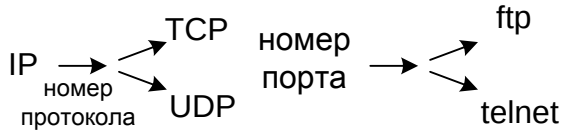
Маршрутизация.

Транспортные протоколы:

– TCP – протокол доставки пакетов, ориентированный на установление соединение в сети.

– UDP – для передачи отдельных сообщений небольшого размера.

Взаимодействие протоколов:

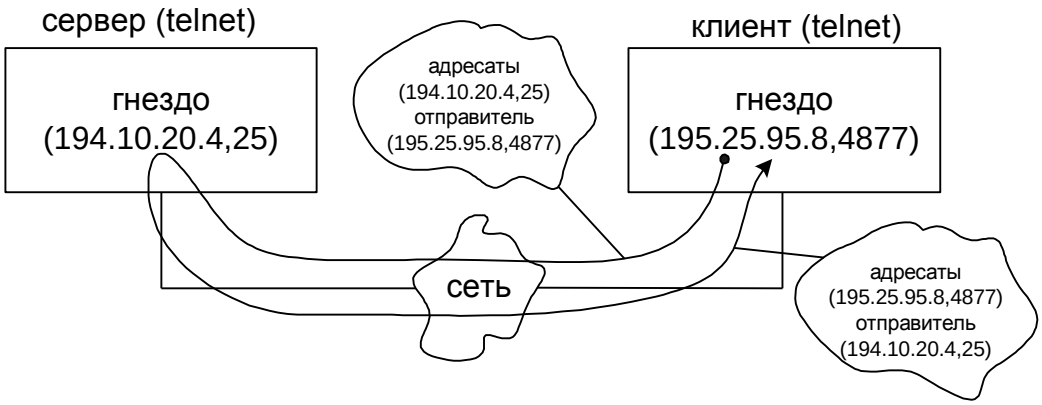


TCP состоит из трёх этапов:

- установление соединения
- передача
- разрыв соединения.

Номер порта – логический номер, используемый ОС для обеспечения взаимодействия сетевых приложений, номера портов меньше 1024 – зарезервированы, как правило, они назначаются сервером. Порты больше 1024 используются по запросам и используются клиентами приложений.

Гнездо (socket) – пара: IP адрес + порт. Гнездо однозначно идентифицировано приложениями функцией в сети. Сервер, как правило, прослушивает информацию, поступающую в его гнездо, для обнаружения запросов. Исполняет их и отправляет запрашивающему клиенту.



Запуск приложения $telnet yahoo.com – номер порта 25 закреплён за telnet.

6. Конфигурирование DNC

(**Domain Names Services** – служба доменных имен)

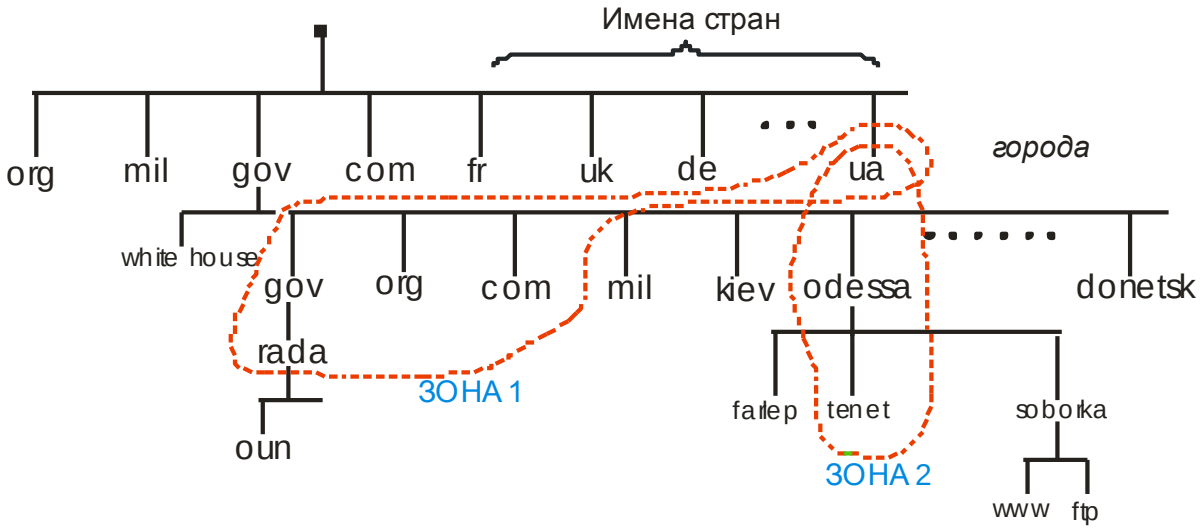
Первичный сервер зоны – сервер, обслуживающий группу доменных имен, именуемых зоной

Вторичный сервер зоны – зеркальная копия первичного сервера, созданная для обеспечения надежности системы DNS

Кэшируемый сервер – сервер, хранящий доменные имена наиболее часто используемых ресурсов.

Система доменных имен

Это иерархическая система



Полное доменное имя отражается как путь дерева «от листа к корню»
www.soborka.odessa.ua

Домен – путь из некоторой вершины в корень этого дерева:

odessa.ua
whitehouse.gov
ua

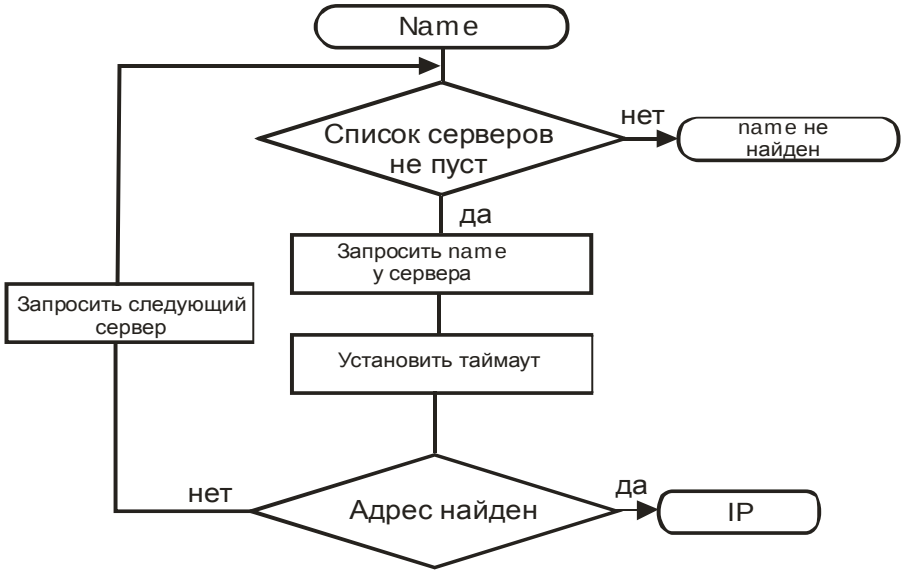
Зона – один либо несколько доменов

Zona1:	Zona2:
org.ua	tenet.odessa.ua
com.ua	
gov.ua	

Конфигурирование клиента DNS

Для конфигурирования необходимо создать файл:
/etc / resolv.conf
nameserver IP адрес
nameserver 193.20.10.54 – первичный
nameserver 225.195.40.18 - вторичный
nameserver 211.11.125.15 – вторичный

Алгоритм разрешения имен



Конфигурирование сервера DNS

Основной файл:

/ etc / named.boot – (файл прямой зоны, обратной зоны, файл кэша)

Обобщенный формат файла прямой и обратной зоны

ИМЯ	***ТИП***	***АДРЕС***
www	IN	194.80.15.10

	ftp	IN	195.30.15.5	
	mail	IN	202.2.5.6	
~10	…	IN	Адрес	Адреса корневых серверов DNS

Использование алгоритма аналогично прямому отображению обеспечения определенного доменного имени по заданному адресу

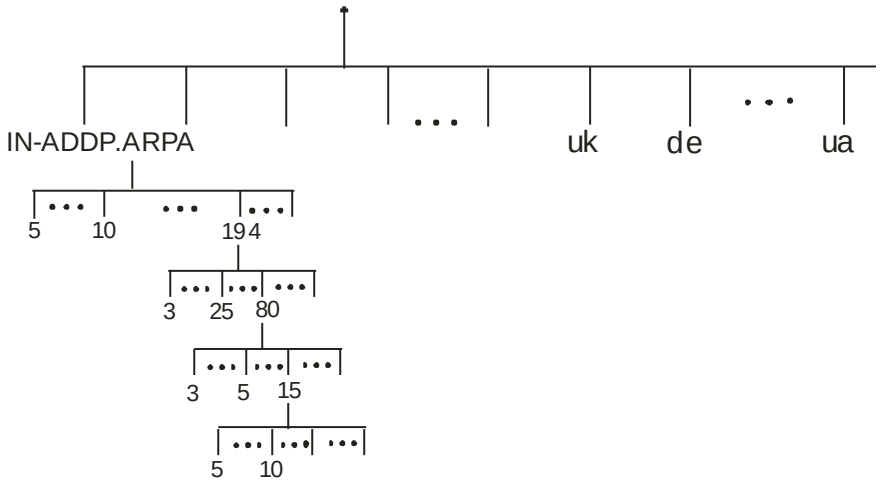
Алгоритм разрешения имен (детальный)

1. Сервер имен по resolv.conf
2. Поиск в КЭШе (www.yahoo.com)
3. Обращение к корневому серверу
4. Последовательное обращение к серверу нижележащих зон

Пример: oun.rada.gov.ua
1. Локальный сервер имен
2. Корневой сервер (•)
3. Сервер зоны, включающий ua
4. Сервер зоны, содержащий домен gov.ua
5. Сервер зоны, содержащий домен rada.gov.ua

Обратное отображение имен

Оно обеспечивает нахождение доменного имени по заданному IP – адресу



7. Конфигурирование и использование сетевых сервисов в Unix

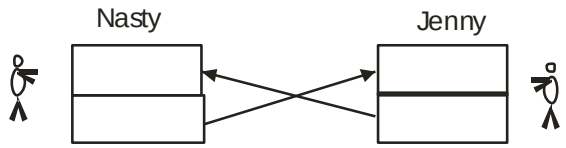
1. finger – средство указания компьютеров и пользователей
finger пользователь @ компьютер
информация об указанном ресурсе

finger mike@ftp8.yahoo.com
Сообщает стандартную информацию учетной записи пользователя. Для помещения собственной информации необходимо создать файл .plan

urathen@london.uk
.planrc – командный файл shell

Сервисы Unix

2. Talk – диалог в форме обмена сообщениями
#talk @host
#talk jenry@m3.yahoo.com



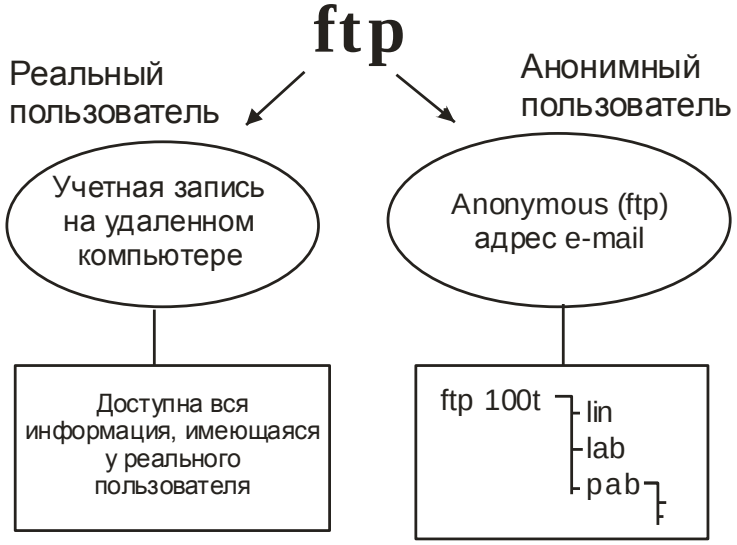
3. Telnet – удаленный или виртуальный терминал
#telnet Host
#команда ⟶ результат

Клиент telnet – ПО для Windows xTerminal

4. FTP – копирование файлов
#ftp host

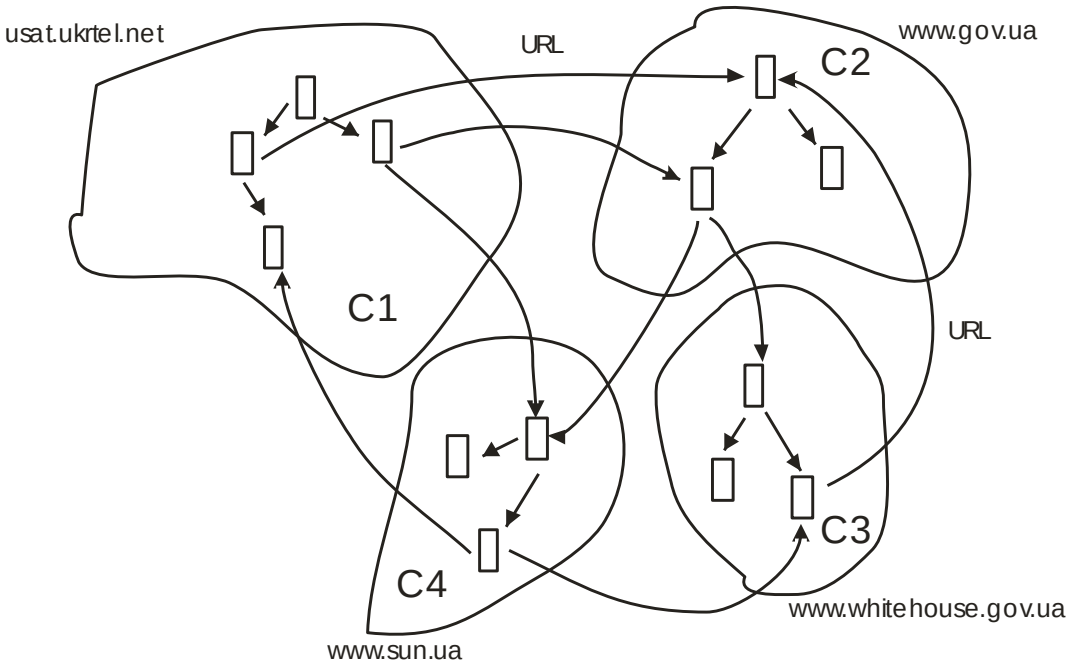
password
>command
>quit
#

Существует 2 вида ftp



5. Gopher – система доступа на основе текстовых меню
6. Wais – поисковый сервис
7. Web – всемирная паутина
1. Гипертекст – текст + ссылки; в масштабах сети
2. URL

Протокол	Хост	Файл



http://www/sun.com/downloads
ftp://ftp.uu.net/pub/file.txt

Для описания странички используются HTML документы

www

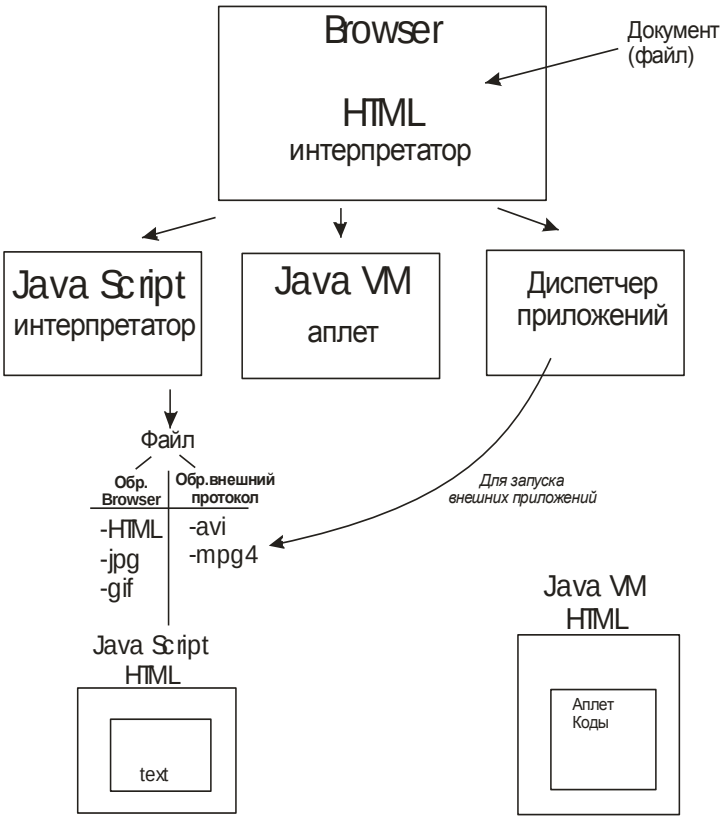
HTML – набор управляющих элементов

<head>
<head>
<title> My page <title>
<head>
<body>
<ing src=”face.jpg”>
I was born …
I am study at …

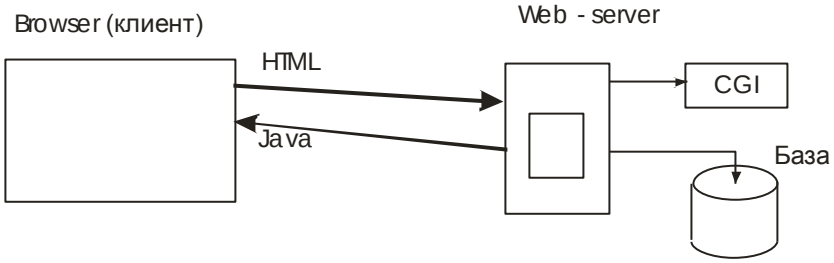
Веб-браузер как универсальный сетевой клиент

Browser – клиент для WWW сервиса:

– Netscape (Solaris,HP-4X,HEX)
– MSIE
– Opera



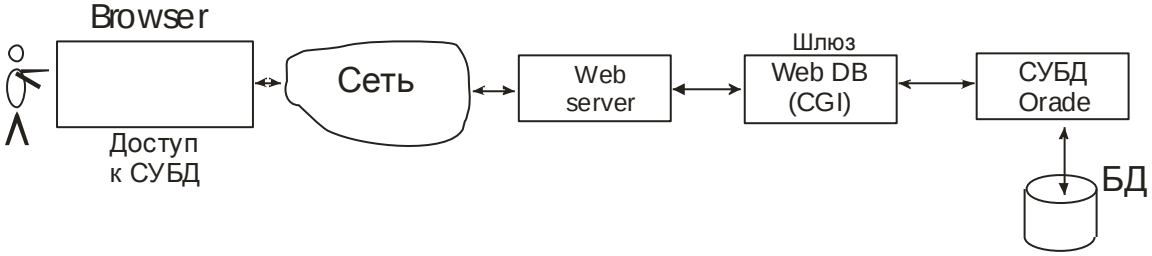
WWW – как универсальный интерфейс сетевых приложений



Browser – универсальный сетевой клиент
Common Gateway Interface

- удаленный запуск приложения;
- передача параметров
- передача резервных работающих приложений (HTML-файл)

Пример использования СУБД Oracle



- программный шлюз Web DB
- Web-server + сервер СУБД

Концепция «тонкого» клиента. Сетевой компьютер

Проект СОС:
- толстый клиент
- тонкий клиент

Толстый клиент – максимум ПО, данные размещены на рабочих станциях

Достоинства:
- возможность автоматического функционирования
- высокая надежность
- малый трафик сети

Недостатки:
- значительные затраты на ресурсы
- затраты на администрирование и сохранение ПО
- трудно обеспечить целым ПО в сети

Тонкий клиент – РС операционная среда (СОС)

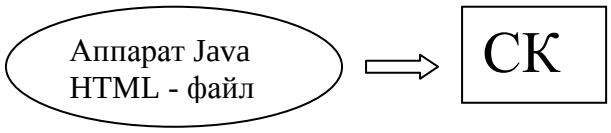
Приложения – сервер приложений
Данные – сервер БД
Рабочая область – сервер каталогов

Преимущества:
- min стоимость рабочих станций
- отсутствие затрат на сопровождение прикладного ПО в РС
- целостность информации

Недостатки:
- необходимость высокопроизводительной надежной сети
- специальная технология прикладных программ
- повышение требования к безопасности

В середине 90-х годов Sun попытались объединить концепцию тонкого клиента и технологию Web, Java и предложили концепцию сетевого компьютера

Сетевой компьютер:
- аппаратная реализация Java VM;
- встроенный Browser



Сетевой компьютер не требует практически никаких настроек:
- интерфейс удаленного приложения
- исполнение небольших приложений
- аплетов на языке Java

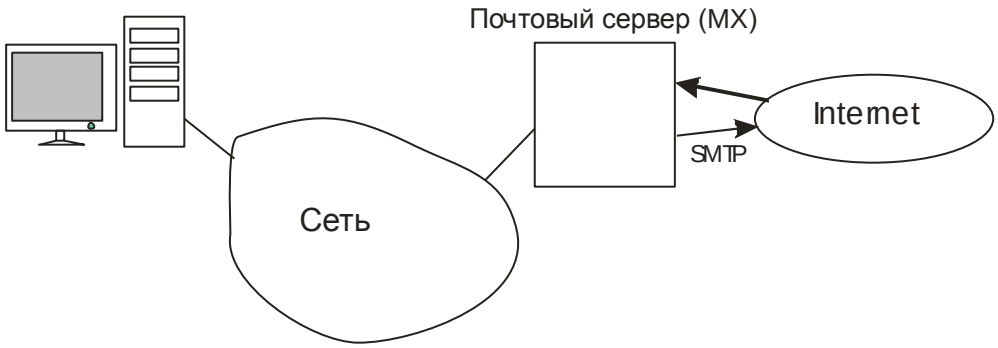
E – mail

Первой Unix доставлен e-mail в домашний каталог пользователя.

$ mail – отправитель или получатель
>S user
<M – получатель
$ mail адрес

Доставка
sendmail
протокол SMTP – (TCP/IP, MNCP, unix – unix copy)

Гетерогенная сеть



Почтовый сервер:
- авторизация пользователя
- хранение корреспонденции
- прием/отправление корреспонденции

Сетевая файловая система NFS

Демоны NFS

	сервер	клиент	
Nfsd	V		Чтение/запись, передана в сеть
Biod		V	Чтение/запись, обмен с сетью
Lockd	V	V	Блокировка ресурсов
Statd	V	V	Сообщение информации о состоянии
Mountd	V		Подключение ресурса

Удаленный запуск процедур RPC
(Remote Procedure Call)

Механизм RPC используется для запуска данных программ схемы NFS.
Сервер слушает socket (IP, pat)

NFS насчитывает десятки стандартных процедур. Каждая процедура NFS однозначно идентифицируется своим номером portmap – стандартная привязка к порту.

Все составные части NFS перечислены в файле /etc/rpc.

Формат:

#name	number	alias-синонимы	
portmap	100000	rps bind	bind
rststd	100001	rem state	

…

Клиент при первом обращении запрашивает номер порта; **portmap** – сообщает номер порта.

Полное обращение к системе RPC состоит из:

- номер программы
- номер версии
- номер процедур
- номер параметров

NIS (NIS+) - информационная система сети

Назначение:

- распространение основных конфигурационных файлов в пределах корпоративной сети;
- обеспечение системы каталогов пользователя;
- обеспечение безопасности сети.

Конфигурационные файлы, которые распространяет система NIS.

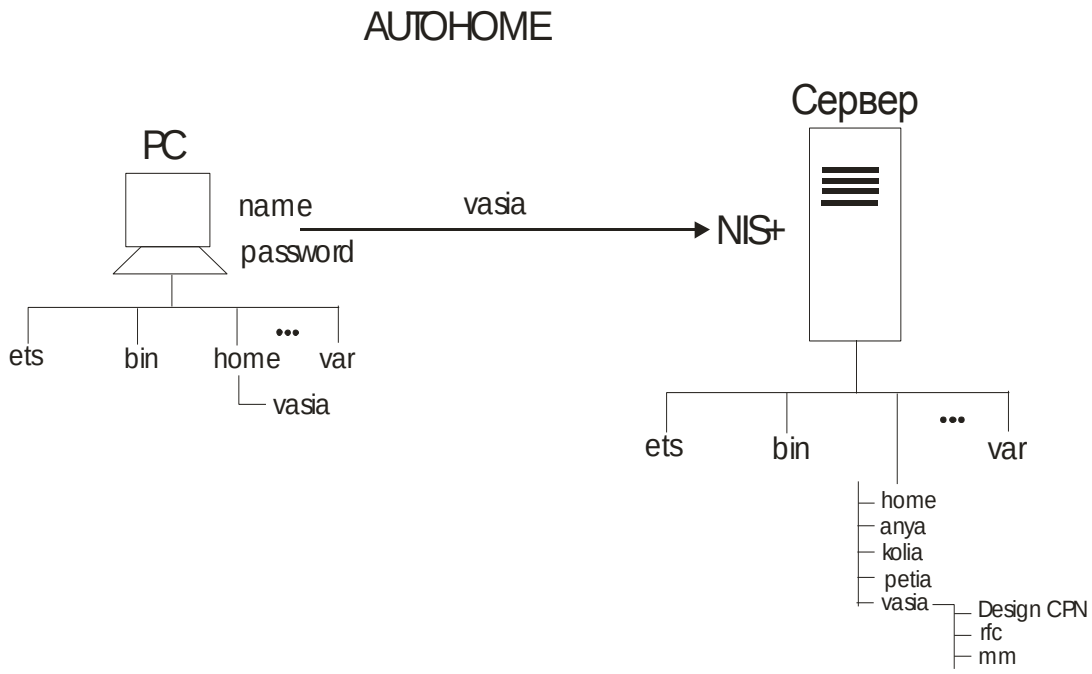
/etc:	
ethers	
protocols	
hosts	/etc/nisswitch.conf
services	protocols nisplus local
networks	names hosts dns nisplus
password	password nisplus local
group	

Администрирование выполняется на ***nisplus***.

Преимущества:

- снижение трудоемкости администрирования сети;
- возможность использования производительной PC;
- повышение безопасности.

Автоматическое отображение протокола



Все, что храниться на сервере в каталоге vasia при входе пользователя на сервер отображается в его виртуальном каталоге.

8. Обзор средств Unix для администрирования сетей

Протокол SNMP (Simple Network Management Protocol)

Администрирование сети включает администрирование рабочих станций и серверов; администрирование сетевых устройств (концентраторов, коммутаторов, маршрутизаторов), и администрирование сети в целом.

Для работы с удаленными серверами и PC есть средства:

- **telnet** – текстовая среда
- **x терминал** – графическая среда Xwindows
  Remoto Admin
  PC Angivare - удаленные графические терминалы
- **Web** – шлюзы администрирования
  HTML, Java, Sun, Solaris

Устройства SNMP – автоматическое получение информации о работе устройства
Web – формы – коммутаторы
telnet – маршрутизаторы

ОП для администрирования сетей

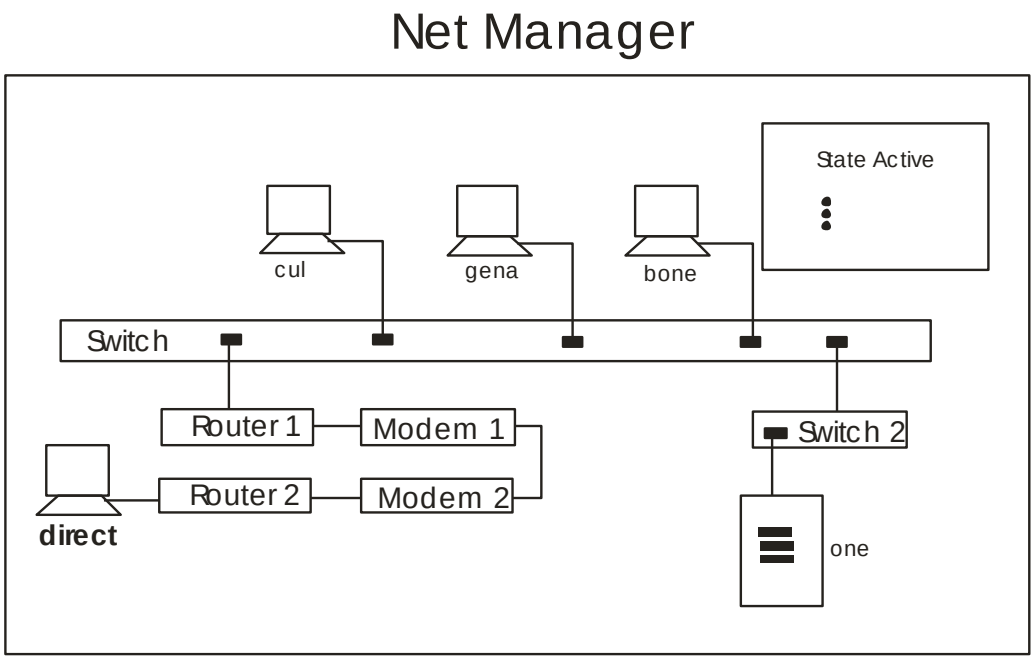
Разрабатывается специальное ПО:

Sun Net Manager
HP Open View

Они позволяют:

- ввод в специальном редакторе, либо автоматически определение конфигурации сети;
- отслеживание состояния сетевых устройств
- анализ трафика

- поиск и установление неполадок
- удобный доступ к средствам администрирования устройств, PC и серверов.



Приложение: Классификация сетевых операционных систем

Операционные системы могут различаться особенностями реализации внутренних алгоритмов управления основными ресурсами компьютера (процессорами, памятью, устройствами), особенностями использованных методов проектирования, типами аппаратных платформ, областями использования и многими другими свойствами.

Ниже приведена классификация ОС по нескольким наиболее основным признакам.

Особенности алгоритмов управления ресурсами

От эффективности алгоритмов управления локальными ресурсами компьютера во многом зависит эффективность всей сетевой ОС в целом. Поэтому, характеризуя сетевую ОС, часто приводят важнейшие особенности реализации функций ОС по управлению процессорами, памятью, внешними устройствами автономного компьютера. Так, например, в зависимости от особенностей использованного алгоритма управления процессором, операционные системы делят на многозадачные и однозадачные, многопользовательские и однопользовательские, на системы, поддерживающие многонитевую обработку и не поддерживающие ее, на многопроцессорные и однопроцессорные системы.

Поддержка многозадачности. По числу одновременно выполняемых задач операционные системы могут быть разделены на два класса:

- однозадачные (например, MS-DOS, MSX) и
- многозадачные (ОС ЕС, OS/2, UNIX, Windows 95).

Однозадачные ОС в основном выполняют функцию предоставления пользователю виртуальной машины, делая более простым и удобным процесс взаимодействия пользователя с компьютером. Однозадачные ОС включают средства управления периферийными устройствами, средства управления файлами, средства общения с пользователем.

Многозадачные ОС, кроме вышеперечисленных функций, управляют разделением совместно используемых ресурсов, таких как процессор, оперативная память, файлы и внешние устройства.

Поддержка многопользовательского режима. По числу одновременно работающих пользователей ОС делятся на:

- однопользовательские (MS-DOS, Windows 3.x, ранние версии OS/2);
- многопользовательские (UNIX, Windows NT).

Главным отличием многопользовательских систем от однопользовательских является наличие средств защиты информации каждого пользователя от несанкционированного доступа других пользователей. Следует заметить, что не всякая многозадачная система является многопользовательской, и не всякая однопользовательская ОС является однозадачной.

Вытесняющая и невытесняющая многозадачность. Важнейшим разделяемым ресурсом является процессорное время. Способ распределения процессорного времени между несколькими одновременно существующими в системе процессами (или нитями) во многом определяет специфику ОС. Среди множества существующих вариантов реализации многозадачности можно выделить две группы алгоритмов:

- невытесняющая многозадачность (NetWare, Windows 3.x);
- вытесняющая многозадачность (Windows NT, OS/2, UNIX).

Основным различием между вытесняющим и невытесняющим вариантами многозадачности является степень централизации механизма планирования процессов. В первом случае механизм планирования процессов целиком сосредоточен в операционной системе, а во втором - распределен между системой и прикладными программами. При невытесняющей многозадачности активный процесс выполняется до тех пор, пока он сам, по собственной инициативе, не отдаст управление операционной системе для того, чтобы та выбрала из очереди другой готовый к выполнению процесс. При вытесняющей многозадачности решение о переключении процессора с одного процесса на другой принимается операционной системой, а не самим активным процессом.

Поддержка многонитевости. Важным свойством операционных систем является возможность распараллеливания вычислений в рамках одной задачи. Многонитевая ОС разделяет процессорное время не между задачами, а между их отдельными ветвями (нитями).

Многопроцессорная обработка. Другим важным свойством ОС является отсутствие или наличие в ней средств поддержки многопроцессорной обработки - *мультипроцессирование*. Мультипроцессирование приводит к усложнению всех алгоритмов управления ресурсами.

Многопроцессорные ОС могут классифицироваться по способу организации вычислительного процесса в системе с многопроцессорной архитектурой: асимметричные ОС и симметричные ОС. Асимметричная ОС целиком выполняется только на одном из процессоров системы, распределяя прикладные задачи по остальным процессорам. Симметричная ОС полностью децентрализована и использует весь пул процессоров, разделяя их между системными и прикладными задачами.

Выше были рассмотрены характеристики ОС, связанные с управлением только одним типом ресурсов - процессором. Важное влияние на облик операционной системы в целом, на возможности ее использования в той или иной области оказывают особенности и других подсистем управления локальными ресурсами - подсистем управления памятью, файлами, устройствами ввода-вывода.

Специфика ОС проявляется и в том, каким образом она реализует сетевые функции: распознавание и перенаправление в сеть запросов к удаленным ресурсам, передача сообщений по сети, выполнение удаленных запросов. При реализации сетевых функций возникает комплекс задач, связанных с распределенным характером хранения и обработки данных в сети: ведение справочной информации обо всех доступных в сети ресурсах и серверах, адресация взаимодействующих процессов, обеспечение прозрачности доступа, тиражирование данных, согласование копий, поддержка безопасности данных.

Особенности аппаратных платформ

На свойства операционной системы непосредственное влияние оказывают аппаратные средства, на которые она ориентирована. По типу аппаратуры различают операционные системы персональных компьютеров, мини-компьютеров, мейнфреймов, кластеров и сетей ЭВМ. Среди перечисленных типов компьютеров могут встречаться как однопроцессорные варианты, так и многопроцессорные. В любом случае специфика аппаратных средств, как правило, отражается на специфике операционных систем.

Очевидно, что ОС большой машины является более сложной и функциональной, чем ОС персонального компьютера. Так в ОС больших машин функции по планированию потока выполняемых задач, очевидно, реализуются путем использования сложных приоритетных дисциплин и требуют большей вычислительной мощности, чем в ОС персональных компьютеров. Аналогично обстоит дело и с другими функциями.

Сетевая ОС имеет в своем составе средства передачи сообщений между компьютерами по линиям связи, которые совершенно не нужны в автономной ОС. На основе этих сообщений сетевая ОС поддерживает разделение ресурсов компьютера между удаленными пользователями, подключенными к сети. Для поддержания функций передачи сообщений сетевые ОС содержат специальные программные компоненты, реализующие популярные коммуникационные протоколы, такие как TCP/IP, IPX и другие.

Многопроцессорные системы требуют от операционной системы особой организации, с помощью которой сама операционная система, а также поддерживаемые ею приложения могли бы выполняться параллельно отдельными процессорами системы. Параллельная работа отдельных частей ОС создает дополнительные проблемы для разработчиков ОС, так как в этом случае гораздо сложнее обеспечить согласованный доступ отдельных процессов к общим системным таблицам, исключить эффект гонок и прочие нежелательные последствия асинхронного выполнения работ.

Другие требования предъявляются к операционным системам кластеров. Кластер - слабо связанная совокупность нескольких вычислительных систем, работающих совместно для выполнения общих приложений, и представляющихся пользователю единой системой. Наряду со специальной аппаратурой для функционирования кластерных систем необходима и программная поддержка со стороны операционной системы, которая сводится в основном к синхронизации доступа к разделяемым ресурсам, обнаружению отказов и динамической реконфигурации системы.

Наряду с ОС, ориентированными на совершенно определенный тип аппаратной платформы, существуют операционные системы, специально разработанные таким образом, чтобы они могли быть легко перенесены с компьютера одного типа на компьютер другого типа, так называемые *мобильные* ОС. Наиболее ярким примером такой ОС является популярная система UNIX. В этих системах аппаратно-зависимые места тщательно локализованы, так что при переносе системы на новую платформу переписываются только они. Средством, облегчающем перенос остальной части ОС, является написание ее на машинно-независимом языке, например, на С, который и был разработан для программирования операционных систем.

Особенности областей использования

Многозадачные ОС подразделяются на три типа в соответствии с использованными при их разработке критериями эффективности:

- системы пакетной обработки (например, ОС ЕС),
- системы разделения времени (UNIX, VMS, Windows),
- системы реального времени (QNX, RT/11).

Системы пакетной обработки предназначались для решения задач в основном вычислительного характера, не требующих быстрого получения результатов. Главной целью и критерием эффективности систем пакетной обработки является максимальная пропускная способность, то есть решение максимального числа задач в единицу времени. Для достижения этой цели в системах пакетной обработки используется следующая схема функционирования: в начале работы формируется пакет заданий, каждое задание содержит требование к системным ресурсам; из этого пакета заданий формируется мультипрограммная смесь, то есть множество одновременно выполняемых задач. Для одновременного выполнения выбираются задачи, предъявляющие отличающиеся требования к ресурсам, так, чтобы обеспечивалась сбалансированная загрузка всех устройств вычислительной машины; так, например, в мультипрограммной смеси желательно одновременное присутствие вычислительных задач и задач с интенсивным вводом-выводом. Таким образом, выбор нового задания из пакета заданий зависит от внутренней ситуации, складывающейся в системе, то есть выбирается "выгодное" задание. Следовательно, в таких ОС невозможно гарантировать выполнение того или иного задания в течение определенного периода времени. В системах пакетной обработки переключение процессора с выполнения одной задачи на выполнение другой происходит только в случае, если активная задача сама отказывается от процессора, например, из-за необходимости выполнить операцию ввода-вывода. Поэтому одна задача может надолго занять процессор, что делает невозможным выполнение интерактивных задач. Таким образом, взаимодействие пользователя с вычислительной машиной, на которой установлена система пакетной обработки, сводится к тому, что он приносит задание, отдает его диспетчеру-оператору, а в конце дня после выполнения всего пакета заданий получает результат. Очевидно, что такой порядок снижает эффективность работы пользователя.

Системы разделения времени призваны исправить основной недостаток систем пакетной обработки - изоляцию пользователя-программиста от процесса выполнения его задач. Каждому пользователю системы разделения времени предоставляется терминал, с которого он может вести диалог со своей программой. Так как в системах разделения времени каждой задаче выделяется только квант процессорного времени, ни одна задача не занимает процессор надолго, и время ответа оказывается приемлемым. Если квант выбран достаточно небольшим, то у всех пользователей, одновременно работающих на одной и той же машине, складывается впечатление, что каждый из них единолично использует машину. Ясно, что системы разделения времени обладают меньшей пропускной способностью, чем системы пакетной обработки, так как на выполнение принимается каждая запущенная пользователем задача, а не та, которая "выгодна" системе, и, кроме того, имеются накладные расходы вычислительной мощности на более частое переключение процессора с задачи на задачу. Критерием эффективности систем разделения времени является не максимальная пропускная способность, а удобство и эффективность работы пользователя.

Системы реального времени применяются для управления различными техническими объектами, такими, например, как станок, спутник, научная экспериментальная установка или технологическими процессами, такими, как гальваническая линия, доменный процесс и т.п. Во всех этих случаях существует предельно допустимое время, в течение которого должна быть выполнена та или иная программа, управляющая объектом, в противном случае может произойти авария: спутник выйдет из зоны видимости, экспериментальные данные, поступающие с датчиков, будут потеряны, толщина гальванического покрытия не будет соответствовать норме. Таким образом, критерием эффективности для систем реального времени является их способность выдерживать заранее заданные интервалы времени между запуском программы и получением результата (управляющего воздействия). Это время называется временем реакции системы, а соответствующее свойство системы - реактивностью. Для этих систем мультипрограммная смесь представляет собой фиксированный набор заранее разработанных программ, а выбор программы на выполнение осуществляется исходя из текущего состояния объекта или в соответствии с расписанием плановых работ.

Некоторые операционные системы могут совмещать в себе свойства систем разных типов, например, часть задач может выполняться в режиме пакетной обработки, а часть - в режиме реального времени или в режиме разделения времени. В таких случаях режим пакетной обработки часто называют фоновым режимом.

Особенности методов построения

При описании операционной системы часто указываются особенности ее структурной организации и основные концепции, положенные в ее основу.

К таким базовым концепциям относятся:

- Способы построения ядра системы - монолитное ядро или микроядерный подход. Большинство ОС использует монолитное ядро, которое компонуется как одна программа, работающая в привилегированном режиме и использующая быстрые переходы с одной процедуры на другую, не требующие переключения из привилегированного режима в пользовательский и наоборот. Альтернативой является построение ОС на базе микроядра, работающего также в привилегированном режиме и выполняющего только минимум функций по управлению аппаратурой, в то время как функции ОС более высокого уровня

выполняют специализированные компоненты ОС - серверы, работающие в пользовательском режиме. При таком построении ОС работает более медленно, так как часто выполняются переходы между привилегированным режимом и пользовательским, зато система получается более гибкой - ее функции можно наращивать, модифицировать или сужать, добавляя, модифицируя или исключая серверы пользовательского режима. Кроме того, серверы хорошо защищены друг от друга, как и любые пользовательские процессы.

- Построение ОС на базе объектно-ориентированного подхода дает возможность использовать все его достоинства, хорошо зарекомендовавшие себя на уровне приложений, внутри операционной системы, а именно: аккумуляцию удачных решений в форме стандартных объектов, возможность создания новых объектов на базе имеющихся с помощью механизма наследования, хорошую защиту данных за счет их инкапсуляции во внутренние структуры объекта, что делает данные недоступными для несанкционированного использования извне, структурированность системы, состоящей из набора хорошо определенных объектов.
- Наличие нескольких прикладных сред дает возможность в рамках одной ОС одновременно выполнять приложения, разработанные для нескольких ОС. Многие современные операционные системы поддерживают одновременно прикладные среды MS-DOS, Windows, UNIX (POSIX), OS/2 или хотя бы некоторого подмножества из этого популярного набора. Концепция множественных прикладных сред наиболее просто реализуется в ОС на базе микроядра, над которым работают различные серверы, часть которых реализуют прикладную среду той или иной операционной системы.
- Распределенная организация операционной системы позволяет упростить работу пользователей и программистов в сетевых средах. В распределенной ОС реализованы механизмы, которые дают возможность пользователю представлять и воспринимать сеть в виде традиционного однопроцессорного компьютера. Характерными признаками распределенной организации ОС являются: наличие единой справочной службы разделяемых ресурсов, единой службы времени, использование механизма вызова удаленных процедур (RPC) для прозрачного распределения программных процедур по машинам, многонитевой обработки, позволяющей распараллеливать вычисления в рамках одной задачи и выполнять эту задачу сразу на нескольких компьютерах сети, а также наличие других распределенных служб.

Список основной рекомендуемой литературы

1. Олифер В.Г., Олифер Н.А. Сетевые операционные системы. – Питер, 2001. – 544 с.
2. Дейтел Г. Введение в операционные системы: В 2-х т. – М.: Мир, 1987. – 756 с.
3. Краковяк С. Основы организации и функционирования ЭВМ. – М.: Мир, 1988. – 480 с.
4. Мэдник С., Донован Дж. Операционные системы. – М.: Мир, 1978. – 640с.

Список дополнительной литературы

1. Иртегов Д. Введение в операционные системы. – BHV: Санкт-Петербург, 2002. – 624 с.
2. Столлингс В. Операционные системы. – Вильямс , 2002. – 848 с.
3. Партыка Т.Л., Попов И.И. Операционные системы, среды и оболочки. – Форум, 2003. – 400 с.
4. Гордеев А.В. Молчанов А.Ю Системное программное обеспечение. СПб Питер, 2001.

5. Робачевский А.М. ОС UNIX. BHV: С-Петербург, 2001.
6. Кулаков Ю.А., Омелянский С.В. Компьютерные сети. К.: Юниор,1999.
7. Шоу А. Логическое проектирование операционных систем. – М.: Мир, 1981. – 360 с.
8. Соловьев Г.Н. Операционные системы ЦВМ. – М.: Машиностроение, 1977. – 136 с.
9. Татцан Г. Операционные системы.- М.: Мир, 1976.- 471с.
10. Лихачева Г.Н., Медведев В.Д. Операционные системы. – М.: Статистика, 1980. – 231с.
11. Зелковиц М., Шоу А., Гэннон Дж. Принципы разработки программного обеспечения. – М.: Мир, 1982. – 368 с.
12. Коэн Л.Дж. Анализ и разработка операционных систем. – М.: Наука, 1975. – 190 с.
13. Кристиан К. Введение в операционную систему UNIX. – М.: Финансы и статистика, 1985. – 318 с.
14. Бирюков В.В., Рыбаков А.В., Шикура Ю.П. Введение в систему программирования ОС РВ. – М.: Финансы и статистика, 1986. – 192 с.
15. Кейслер С. Проектирование операционных систем для малых ЭВМ. – М.: Мир, 1986. – 680 с.
16. Методические указания к курсовому проектированию операционных систем (для студентов специальности 22.04) / Сост.: А.В.Григорьев, Д.А.Зайцев, А.И.Слепцов. – Донецк: ДГТУ, 1994. – 29 с.