10. Компьютерные сети: коммутация пакетов, классификации (по топологии, по способам администрирования, по архитектуре и т. д.). Модели сетевых коммуникаций OSI и TCP/IP. IP-адресация. MAC-адрес. Мост. Режимы передачи. Понятие маршрутизации (статическая, динамическая).

10.1 Компьютерные сети: Коммутация пакетов

Коммута́ция — процесс соединения абонентов коммуникационной сети через транзитные узлы.

Виды коммутации

Существует четыре принципиально различные схемы коммутации абонентов в сетях:

- **Коммутация каналов**
- **Коммутация сообщений**
- **Коммутация пакетов** (КП, *packet switching*) — разбиение сообщения на «пакеты», которые передаются отдельно. Разница между сообщением и пакетом: размер пакета ограничен технически, сообщения — логически. При этом, если маршрут движения пакетов между узлами определён заранее, говорят о *виртуальном канале* (с установлением соединения). Пример: коммутация IP-пакетов. Если же для каждого пакета задача нахождения пути решается заново, говорят о *датаграммном* (без установления соединения) способе пакетной коммутации.
- **Коммутация ячеек** (

Коммутация пакетов - это техника коммутации абонентов, которая была специально разработана для эффективной передачи компьютерного трафика. Эксперименты по созданию первых компьютерных сетей на основе техники коммутации каналов показали, что этот вид коммутации не позволяет достичь высокой общей пропускной способности сети. Суть проблемы заключается в пульсирующем характере трафика, который генерируют типичные сетевые приложения. Например, при обращении к удаленному файловому серверу пользователь сначала просматривает содержимое каталога этого сервера, что порождает передачу небольшого объема данных. Затем он открывает требуемый файл в текстовом редакторе, и эта операция может создать достаточно интенсивный обмен данными, особенно если файл содержит объемные графические включения. После отображения нескольких страниц файла пользователь некоторое время работает с ними локально, что вообще не требует передачи данных по сети, а затем возвращает модифицированные копии страниц на сервер - и это снова порождает интенсивную передачу данных по сети.
Коэффициент пульсации трафика отдельного пользователя сети, равный отношению средней интенсивности обмена данными к максимально возможной, может составлять 1:50 или 1:100. Если для описанной сессии организовать коммутацию канала между компьютером пользователя и сервером, то большую часть времени канал будет простаивать. В то же время коммутационные возможности сети будут использоваться - часть тайм-слотов или частотных полос коммутаторов будет занята и недоступна другим пользователям сети.
При коммутации пакетов все передаваемые пользователем сети сообщения разбиваются в

исходном узле на сравнительно небольшие части, называемые пакетами. Напомним, что сообщением называется логически завершенная порция данных - запрос на передачу файла, ответ на этот запрос, содержащий весь файл, и т. п. Сообщения могут иметь произвольную длину, от нескольких байт до многих мегабайт. Напротив, пакеты обычно тоже могут иметь переменную длину, но в узких пределах, например от 46 до 1500 байт. Каждый пакет снабжается заголовком, в котором указывается адресная информация, необходимая для доставки пакета узлу назначения, а также номер пакета, который будет использоваться узлом назначения для сборки сообщения . Пакеты транспортируются в сети как независимые информационные блоки. Коммутаторы сети принимают пакеты от конечных узлов и на основании адресной информации передают их друг другу, а в конечном итоге - узлу назначения.
Коммутаторы пакетной сети отличаются от коммутаторов каналов тем, что они имеют внутреннюю буферную память для временного хранения пакетов, если выходной порт коммутатора в момент принятия пакета занят передачей другого пакета. В этом случае пакет находится некоторое время в очереди пакетов в буферной памяти выходного порта, а когда до него дойдет очередь, то он передается следующему коммутатору. Такая схема передачи данных позволяет сглаживать пульсации трафика на магистральных связях между коммутаторами и тем самым использовать их наиболее эффективным образом для повышения пропускной способности сети в целом.

10.2 Компьютерные сети: классификации

Компьютерная сеть (КС) являет собой систему распределенной обработки информации, которая составляет как минимум из двух компьютеров, взаимодействующих между собой с помощью специальных средств связи. Или другими словами сеть являет собой совокупность соединенных друг с другом ПК и других вычислительных устройств, таких как принтеры, факсимильные аппараты и модемы. Сеть дает возможность отдельным сотрудникам организации взаимодействовать друг с другом и обращаться к совместно используемым ресурсам; позволяет им получать доступ к данным, что сохраняется на персональных компьютерах в удаленных офисах, и устанавливать связь с поставщиками.

По назначению КС распределяются на:

1. вычислительные;
2. информационные;
3. смешанные (информационно-вычислительные).

Смешанные сети совмещают функции первых двух.

По типа компьютеров, которые входят в состав КС, различают:

1. однородные компьютерные сети, которые состоят из программно общих ЭВМ;
2. неоднородные, в состав которых входят программно-несовместительные компьютеры.

Классифицируя сети по территориальному признаку, различают:

1. локальные (*LocalAreaNetworks– LAN*) сети;
2. глобальные (*WideAreaNetworks– WAN*) сети;
3. городские (*MetropolitanAreaNetworks– MAN*) сети.

LAN– сосредоточены на территории не больше 1–2 км; построенные с использованием дорогих высококачественных линий связи, которые позволяют, применяя простые методы передачи данных, достигать высоких скоростей обмена данными порядка 100 Мбит/с, предоставленные услуги отличаются широкой разнообразностью и обычно предусматривают реализацию в режиме on-line.
WAN– совмещают компьютеры, рассредоточенные на расстоянии сотен и тысяч километров. Часто используются уже существующие не очень качественные линии связи. Больше низкие, чем в локальных сетях, скорости передачи данных (десятки килобит в секунду) ограничивают набор предоставленных услуг передачей файлов, преимущественно не в оперативном, а в фоновом режиме, с использованием электронной почты. Для стойкой передачи дискретных данных применяются более сложные методы и оборудование, чем в локальных сетях.
MAN– занимают промежуточное положение между локальными и глобальными сетями. При достаточно больших расстояниях между узлами (десятки километров) они имеют качественные линии связи и высоких скоростей обмена, иногда даже больше высокими, чем в классических локальных сетях. Как и в случае локальных сетей, при построении *MAN*уже существующие линии связи не используются, а прокладываются заново.
Также дополнительно выделяют:
4. кампусные сети (*Campus Area Network – CAN*), которые совмещают значительно удаленные друг от друга абонентские системы или локальные сети, но еще не требуют отдаленных коммуникаций через телефонные линии и модемы;
5. широкомасштабные сети (*WideAreaNetwork – WAN*), которые используют отдаленные мосты и маршрутизаторы с возможно невысокими скоростями передачи данных.

Отличительные **признаки локальной сети**:

1. высокая скорость передачи, большая пропускная способность;
2. низкий уровень ошибок передачи (или, что то же, высококачественные каналы связи). Допустимая вероятность ошибок передачи данных должна быть порядку 10-7 – 10-8;
3. эффективный, быстродействующий механизм управления обменом;
4. ограничено, точно определенное число компьютеров, которые подключаются к сети.

Глобальные сети отличаются от локальных тем, которые рассчитаны на неограниченное число абонентов и используют, как правило, не слишком качественные каналы связи и сравнительно низкую скорость передачи, а механизм управления обменом, у них в принципе не может быть гарантировано скорым. В глобальных сетях намного более важное не качество связи, а сам факт ее существования.
Правда, в настоящий момент уже нельзя провести четкий и однозначный предел между локальными и глобальными сетями. Большинство локальных сетей имеют выход в глобальную сеть, но характер переданной информации, принципы организации обмена, режимы доступа, к ресурсам внутри локальной сети, как правило, сильно отличаются от тех, что принято в глобальной сети. И хотя все компьютеры локальной сети в данном случае включены также и в глобальную сеть, специфику локальной сети это не отменяет. Возможность выхода в глобальную сеть остается всего лишь одним из ресурсов, поделенные пользователями локальной сети.
Следующей не менее распространенной классификацией есть классификация КС по типа топологии

Широковещательные сети – характерен
широковещательный режим работы, когда на передачу может работать только
одна рабочая станция, а все остальные станции сети – на прием. Это

локальные сети с селекцией информации: общая шина, дерево, звезда с пассивным центром.

- Основные преимущества ЛКС с общей шиной – простота расширения сети путем подключения к шине новых рабочих станций, простота управления сетью, минимальный расход кабеля.

- ЛКС с топологией типа «дерево» - это более развитый вариант сети с шинной топологией. Дерево образуется путем соединения нескольких шин активными повторителями или пассивными размножителями («хабами»), каждая ветвь дерева представляет собой сегмент.

- В ЛКС с топологией типа «звезда» в центре находится пассивный соединитель или активный повторитель –
- Последовательные сети – осуществляется маршрутизация информации, передача данных
  производится последовательно от одной станции к соседней

- В сетях с кольцевой топологией информация чаще всего передается только в одном направлении, обычно против часовой стрелки.

10.3 Компьютерные сети: Модели сетевых коммуникаций OSI и TCP/IP

Модель представляет стандарт для связи сетевых систем. Эта модель не определяет в точности способы, с помощью которых связываются отдельные системы, а служит описанием уровней, функций и услуг для этих систем.

Модель OSI состоит из семи уровней, каждый из которых, как уже отмечалось, получает услуги от расположенного ниже уровня и предоставляет услуги уровню, расположенному выше.

1. Физический уровень
2. Канальный уровень
3. Сетевой уровень
4. Транспортный уровень
5. Сеансовый уровень
6. Уровень представления
7. Прикладной уровень

Описание отдельных уровней

Физический уровень

Физический уровень представляет собой самый нижний уровень модели OSI. Его задача состоит в установлении, поддержании и разъединении физического канала типа "точка-

точка" и "точка - многоточка" для эффективного распределения коммуникационных ресурсов между пользователями и в преобразовании цифровых данных в сигналы, предназначенные для передачи посредством физической среды передачи.

Канальный уровень

Канальный уровень обеспечивает канал связи между двумя смежными элементами сети, устанавливает параметры передачи и сигналы, ошибки в которых на физическом уровне не могут быть скорректированы. Канальный уровень упаковывает биты в кадры данных и отождествляет их с адресами MAC. MAC создает базу для связи локального устройства (локальных устройств), формируя домен для одноадресной и многоадресной передачи. На этом уровне работают мосты и коммутаторы.

Сетевой уровень

Сетевой уровень предоставляет функциональные средства передачи данных от источника до пункта назначения (при различных расстояниях между ними) по одной или нескольким сетям при поддержании качества услуг, требуемого транспортным уровнем. Сетевой уровень осуществляет функции маршрутизации, пользуясь услугами работающих на этом уровне маршрутизаторов.

Маршрутизаторы работают со схемой иерархической адресации. Наиболее известный протокол сетевого уровня - протокол Интернета (IP).

Транспортный уровень

Транспортный уровень обеспечивает передачу данных между конечными пользователями. Главные протоколы транспортного уровня - TCP и UDP.

TCP

"Протокол надежной передачи данных

"Протокол управления обменом данных регулирует прием пакетов данных и предохраняет приемник TCP от переполнения

"Режим "скользящего" окна - после того, как определенное количество данных получили пометку "принятые" (объем таких данных определяется техническими условиями передачи), приемная сторона может дать запрос передающей на отправку дополнительных данных.

UDP

" протокол "ненадежной" передачи данных

" используется Интернет-приложениями, не нуждающимися в получении приемным хостом всех без исключения пакетов данных (Интернет-радио, интерактивные игры, потоковое видео и т.п.)

Сеансовый уровень

Сеансовый уровень координирует пересылку данных между взаимодействующими сеансовыми уровнями.

Уровень представления

Этот уровень отвечает за форматирование данных для объектов прикладного уровня. Он выполняет кодовое и алфавитное шифрование, изменяет представление графики и т.п.

Прикладной уровень

Прикладной уровень открывает объектам прикладного уровня доступ к сетям связи, таким образом обеспечивая их взаимодействие.

ПОЛНЫЙ ОТВЕТ

Стек протоколов TCP/IP (англ. *Transmission Control Protocol/Internet Protocol* — протокол управления передачей) — набор сетевых протоколов разных уровней модели сетевого взаимодействия DOD, используемых в сетях. Протоколы работают друг с другом в стеке (англ. *stack*, стопка) — это означает, что протокол, располагающийся на уровне выше, работает «поверх» нижнего, используя механизмы инкапсуляции. Например, протокол TCP работает поверх протокола IP.

10.4 Компьютерные сети: IP адресация

Типы адресов: физический (MAC-адрес), сетевой (IP-адрес) и символьный (DNS-имя)

Каждый компьютер в сети TCP/IP имеет адреса трех уровней:

- Локальный адрес узла, определяемый технологией, с помощью которой построена отдельная сеть, в которую входит данный узел. Для узлов, входящих в локальные сети - это MAC-адрес сетевого адаптера или порта маршрутизатора, например, 11-A0-17-3D-BC-01. Эти адреса назначаются производителями оборудования и являются уникальными адресами, так как управляются централизовано. Для всех существующих технологий локальных сетей MAC-адрес имеет формат 6 байтов: старшие 3 байта - идентификатор фирмы производителя, а младшие 3 байта назначаются уникальным образом самим производителем. Для узлов, входящих в глобальные сети, такие как X.25 или frame relay, локальный адрес назначается администратором глобальной сети.
- IP-адрес, состоящий из 4 байт, например, 109.26.17.100. Этот адрес используется на сетевом уровне. Он назначается администратором во время конфигурирования компьютеров и маршрутизаторов. IP-адрес состоит из двух частей: номера сети и номера узла. Номер сети может быть выбран администратором произвольно, либо назначен по рекомендации специального подразделения Internet (Network Information Center, NIC), если сеть должна работать как составная часть Internet. Обычно провайдеры услуг Internet получают диапазоны адресов у подразделений NIC, а затем распределяют их между своими абонентами.

Номер узла в протоколе IP назначается независимо от локального адреса узла. Деление IP-адреса на поле номера сети и номера узла - гибкое, и граница между этими полями может

устанавливаться весьма произвольно. Узел может входить в несколько IP-сетей. В этом случае узел должен иметь несколько IP-адресов, по числу сетевых связей. Таким образом IP-адрес характеризует не отдельный компьютер или маршрутизатор, а одно сетевое соединение.

- Символьный идентификатор-имя, например, SERV1.IBM.COM. Этот адрес назначается администратором и состоит из нескольких частей, например, имени машины, имени организации, имени домена. Такой адрес, называемый также DNS-именем, используется на прикладном уровне, например, в протоколах FTP или telnet.

Три основных класса IP-адресов

IP-адрес имеет длину 4 байта и обычно записывается в виде четырех чисел, представляющих значения каждого байта в десятичной форме, и разделенных точками, например:

128.10.2.30 - традиционная десятичная форма представления адреса,

10000000 00001010 00000010 00011110 - двоичная форма представления этого же адреса.

На рисунке 3.1 показана структура IP-адреса.

Класс A

0	N сети	N узла

Класс B

1	0	N сети	N узла

Класс C

1	1	0	N сети	N узла

Класс D

1	1	1	0	адрес группы multicast

Класс E

1	1	1	1	0	зарезервирован

Рис. 3.1. Структура IP-адреса

Адрес состоит из двух логических частей - номера сети и номера узла в сети. Какая часть адреса относится к номеру сети, а какая к номеру узла, определяется значениями первых битов адреса:

- Если адрес начинается с 0, то сеть относят к классу A, и номер сети занимает один байт, остальные 3 байта интерпретируются как номер узла в сети. Сети класса A имеют номера в диапазоне от 1 до 126. (Номер 0 не используется, а номер 127 зарезервирован для специальных целей, о чем будет сказано ниже.) В сетях класса A количество узлов должно быть больше 2^{16} , но не превышать 2^{24}.
- Если первые два бита адреса равны 10, то сеть относится к классу B и является сетью средних размеров с числом узлов 2^8 - 2^{16}. В сетях класса B под адрес сети и под адрес узла отводится по 16 битов, то есть по 2 байта.
- Если адрес начинается с последовательности 110, то это сеть класса C с числом узлов не больше 2^8. Под адрес сети отводится 24 бита, а под адрес узла - 8 битов.
- Если адрес начинается с последовательности 1110, то он является адресом класса D и обозначает особый, групповой адрес - multicast. Если в пакете в качестве адреса назначения указан адрес класса D, то такой пакет должны получить все узлы, которым присвоен данный адрес.
- Если адрес начинается с последовательности 11110, то это адрес класса E, он зарезервирован для будущих применений.

В таблице приведены диапазоны номеров сетей, соответствующих каждому классу сетей.

10.5 Компьютерные сети: MAC

MAC-адрес (от англ. *Media Access Control* — управление доступом к среде, также **Hardware Address**) — это уникальный идентификатор, присваиваемый каждой единице оборудования компьютерных сетей. Большинство сетевых протоколов канального уровня используют одно из трёх пространств MAC-адресов, управляемых IEEE: MAC-48, EUI-48 и EUI-64. Адреса в каждом из пространств теоретически должны быть глобально уникальными. Не все протоколы используют MAC-адреса, и не все протоколы, использующие MAC-адреса, нуждаются в подобной уникальности этих адресов.

В широковещательных сетях (таких, как сети на основе Ethernet) MAC-адрес позволяет уникально идентифицировать каждый узел сети и доставлять данные только этому узлу. Таким образом, MAC-адреса формируют основу сетей на канальном уровне, которую используют протоколы более высокого (сетевого) уровня. Для преобразования MAC-адресов в адреса сетевого уровня и обратно применяются специальные протоколы (например, ARP и RARP в сетях IPv4 и NDP в сетях на основе IPv6).

Стандарты IEEE определяют 48-разрядный (6 октетов) MAC-адрес, который разделен на четыре части.

Первые 3 октета (в порядке их передачи по сети; старшие 3 октета, если рассматривать их в традиционной бит-реверсной шестнадцатеричной записи MAC-адресов) содержат 24-битный уникальный идентификатор организации (OUI)[1], или (Код MFG — Manufacturing, производителя), который производитель получает в IEEE. При этом используются только младшие 22 разряда (бита), 2 старшие имеют специальное назначение:

- первый бит указывает, для одиночного (0) или группового (1) адресата предназначен кадр
- следующий бит указывает, является ли MAC-адрес глобально (0) или локально (1) администрируемым.

Следующие три октета выбираются изготовителем для каждого экземпляра устройства. За исключением сетей системной сетевой архитектуры SNA.

Таким образом, **глобально администрируемый MAC-адрес** устройства **глобально уникален** и обычно «зашит» в аппаратуру.

10.6 Компьютерные сети: Мост

Мост, **сетевой мост**, **бридж** (жарг., калька с англ. *bridge*) — сетевое устройство 2 уровня модели OSI, предназначенное для объединения сегментов (подсети) компьютерной сети разных топологий и архитектур.

Различия между коммутаторами и мостами

В общем случае коммутатор (свитч) и мост аналогичны по функциональности; разница заключается во внутреннем устройстве: мосты обрабатывают трафик, используя

центральный процессор, коммутатор же использует коммутационную матрицу (аппаратную схему для коммутации пакетов). В настоящее время мосты практически не используются (так как для работы требуют производительный процессор), за исключением ситуаций, когда связываются сегменты сети с разной организацией первого уровня, например, между xDSL соединениями, оптикой, Ethernet'ом. В случае SOHO-оборудования, режим прозрачной коммутации часто называют «мостовым режимом» (bridging).

Функциональные возможности

Мост обеспечивает:

- ограничение домена коллизий
- задержку фреймов, адресованных узлу в сегменте отправителя
- ограничение перехода из домена в домен ошибочных фреймов:
  -

Мосты «изучают» характер расположения сегментов сети путем построения адресных таблиц вида «Интерфейс:MAC-адрес», в которых содержатся адреса всех сетевых устройств и сегментов, необходимых для получения доступа к данному устройству.

10.7 Компьютерные сети: Режимы передачи

Симплекс
Протокол Aloha, изначально использовал симплекс метод передачи данных, это означало, что одновременно по каналу связи информация может передаваться только в одном направлении. Если подтверждение о получении данных не поступает в заданный интервал времени, то передающий узел, полагает, что в сети произошла коллизия из-за того, что данные передавались по сети одновременно и данные были утеряны. В случае обнаружения коллизии, оба узла прекращают передачу данных по сети на произвольный период времени, по истечении которого, узлы снова возобновят передачу данных.
Однако, чем большие объемы данных начинают передаваться через сеть Aloha, тем больше коллизий возникает.

Полу-дуплекс
Для улучшения сетей Aloha, Меткалф разработал новую систему, которая включала механизм, который обнаруживал коллизии (collision detect). С данным механизмом, узлы отслеживают загруженность сети перед передачей данных, а также используют общий канал для передачи данных. Полу-дуплекс, означает, что данные могут предаваться между узлами в обоих направлениях, но одна транзакция данных может передаваться только в одном направлении.

Протокол CSMA/CD состоит из следующих частей:
◆ Carrier Sense – определяет процесс прослушивания сети, перед началом передачи данных. Каждый узел должен определить активные передачи данных по сети, перед началом передачи данных. Если в настоящий момент имеется активная передача, это означает, что линия занята и узел должен дождаться, когда она освободится.

◆ Multiple Access – термин, созданный для определения доступа множества узлов к сети Ethernet. Он определяет возможность узлов передавать данные по общим каналам связи без использования приоритетов и привилегий. Данная возможность определяется соответствующим алгоритмом управления доступами.

◆ Deferral или back-off счетчики определяют время, которое узлы должны ожидать, перед повторной передачей данных по сети Ethernet. Если счетчик попыток отправки пакета, превысит значение в 15 раз, то узел решит, что канал связи недоступен и прекратит попытки отправки пакета. В данной ситуации узел отбросит (потеряет) пакет.

Такая ситуация может произойти, в случае когда слишком много узлов работают в одной физической сети или существенно не хватает пропускной способности сети. Решением такой проблемы может стать разбитие большой сети на меньшие сегменты.

Полный дуплекс
передавать данные между узлами сети в обе стороны одновременно. При полном дуплексе, соединение Ethernet осуществляется по двум витым парам проводов, где одна пара используется для получения данных, а вторая для передачи данных на устройство, подключенное к другому концу провода. Эта технология помогла увеличить пропускную способность соединения и уменьшить вероятность возникновения коллизий.

10.8 Компьютерные сети: маршрутиза́ция

Стати́ческая маршрутиза́ция - вид маршрутизации, при котором маршруты указываются в явном виде при конфигурации маршрутизатора. Вся маршрутизация при этом происходит без участия каких-либо протоколов маршрутизации.

При задании статического маршрута указывается:

- Адрес сети (на которую маршрутизируется трафик), маска сети
- Адрес шлюза (узла), который отвечает за дальнейшую маршрутизацию (или подключен к маршрутизируемой сети напрямую)
- (опционально) метрика (иногда именуется также "ценой") маршрута. При наличии нескольких маршрутов на одну и ту же сеть некоторые маршрутизаторы выбирают маршрут с минимальной метрикой (однако, например, ядро Linux просто игнорирует параметр metric в таблице маршрутизации, и предназначается он только для протоколов маршрутизации, наподобие RIP).

В некоторых маршрутизаторах возможно указывать интерфейс, на который следует направить трафик сети и указать дополнительные условия, согласно которым выбирается маршрут (например, SLA в маршрутизаторах cisco).

Лёгкость отладки и конфигурирования в малых сетях.

- Отсутствие дополнительных накладных расходов (из-за отсутствия протоколов маршрутизации)
- Мгновенная готовность (не требуется интервал для конфигурирования/подстройки)
- Низкая нагрузка на процессор маршрутизатора

Очень плохое масштабирование (добавление (N+1)-ой сети потребует сделать 2*(N+1) записей о маршрутах, причём на большинстве маршрутизаторов таблица маршрутов будет различной, при N>3-4 процесс конфигурирования становится весьма трудоёмким).

- Низкая устойчивость к повреждениям линий связи (особенно, в ситуациях, когда

обрыв происходит между устройствами второго уровня и порт маршрутизатора не получает статус down).

Динамическая (обычно используются именно они, наиболее известным разработчиком является компания CISCO):

В маршрутизаторе с динамическим протоколом (например, BGP-4) резидентно загруженная программа-драйвер изменяет таблицы маршрутизации на основе информации, полученной от соседних маршрутизаторов. В ЭВМ, работающей под UNIX и выполняющей функции маршрутизатора, эту задачу часто решает резидентная программа gated или routed (демон). Последняя - поддерживает только внутренние протоколы маршрутизации.

Применение динамической маршрутизации не изменяет алгоритм маршрутизации, осуществляемой на IP-уровне. Программа-драйвер при поиске маршрутизатора-адресата по-прежнему просматривает таблицы. Любой маршрутизатор может использовать два протокола маршрутизации одновременно, один для внешних связей, другой - для внутренних. Для стыковки внешнего маршрута с внутренним в большинстве протоколов предусматриваются специальные средства.

10. Язык ассемблер для архитектуры i386. Кэш-память. Шины: понятие, синхронный и асинхронный принцип работы, арбитраж

10.1 Ассемблер

Ассе́мблер (от англ. *assembler* — сборщик) — компьютерная программа, компилятор исходного текста программы, написанной на языке ассемблера, в программу на машинном языке.

Как и сам язык (ассемблер), ассемблеры, как правило, специфичны конкретной архитектуре, операционной системе и варианту синтаксиса языка. Вместе с тем существуют мультиплатформенные или вовсе универсальные (точнее, ограниченно-универсальные, потому что на языке низкого уровня нельзя написать аппаратно-независимые программы) ассемблеры, которые могут работать на разных платформах и операционных системах. Среди последних можно также выделить группу *кросс-ассемблеров*, способных собирать машинный код и исполняемые модули (файлы) для других архитектур и ОС.

Ассемблирование может быть не первым и не последним этапом на пути получения исполнимого модуля программы. Так, многие компиляторы с языков программирования высокого уровня выдают результат в виде программы на языке ассемблера, которую в дальнейшем обрабатывает ассемблер. Также результатом ассемблирования может быть не исполнимый, а *объектный* модуль, содержащий разрозненные и непривязанные друг к другу части машинного кода и данных программы, из которого (или из нескольких объектных модулей) в дальнейшем с помощью программы-компоновщика («линкера») может быть скомпонован исполнимый файл.

10.2 Кэш

Кэш[1][2][3] или **кеш**[4][5][6] (англ. *cache*, от фр. *cacher* — «прятать»; произносится [kæʃ] — «кэш») — промежуточный буфер с быстрым доступом, содержащий информацию, которая может быть запрошена с наибольшей вероятностью. Доступ к данным в кэше идёт быстрее, чем выборка исходных данных из оперативной (ОЗУ) и быстрее внешней (жёсткий диск или твердотельный накопитель) памяти, за счёт чего уменьшается среднее время доступа и увеличивается общая производительность компьютерной системы. Доступ к данным, хранящимся в кэше программным путем на процессорах линейки [x86] невозможен.

Кэш — это память с большей скоростью доступа, предназначенная для ускорения обращения к данным, содержащимся постоянно в памяти с меньшей скоростью доступа (далее «основная память»). Кэширование применяется ЦПУ, жёсткими дисками, браузерами, веб-серверами, службами DNS и WINS.

Кэш состоит из набора записей. Каждая запись ассоциирована с элементом данных или блоком данных (небольшой части данных), которая является копией элемента данных в основной памяти. Каждая запись имеет идентификатор, определяющий соответствие между элементами данных в кэше и их копиями в основной памяти.

Когда клиент кэша (ЦПУ, веб-браузер, операционная система) обращается к данным, прежде всего исследуется кэш. Если в кэше найдена запись с идентификатором, совпадающим с идентификатором затребованного элемента данных, то используются элементы данных в кэше. Такой случай называется *попаданием кэша*. Если в кэше не найдена запись, содержащая затребованный элемент данных, то он читается из основной памяти в кэш, и становится доступным для последующих обращений. Такой случай называется *промахом кэша*. Процент обращений к кэшу, когда в нём найден результат, называется *уровнем попаданий* или *коэффициентом попаданий* в кэш.

Например, веб-браузер проверяет локальный кэш на диске на наличие локальной копии веб-страницы, соответствующей запрошенному URL. В этом примере URL — это идентификатор, а содержимое веб-страницы — это элементы данных.

Кэш центрального процессора

Ряд моделей центральных процессоров (ЦП) обладают собственным кэшем, для того чтобы минимизировать доступ к оперативной памяти (ОЗУ), которая медленнее, чем регистры. Кэш-память может давать значительный выигрыш в производительности, в случае когда тактовая частота ОЗУ значительно меньше тактовой частоты ЦП. Тактовая частота для кэш-памяти обычно ненамного меньше частоты ЦП.

В процессорах с поддержкой виртуальной адресации часто вводят небольшой быстродействующий буфер трансляций адресов (TLB). Его скорость важна, т.к. он опрашивается на каждом обращении в память.

Уровни кэша

Кэш центрального процессора разделён на несколько уровней. В универсальном процессоре

в настоящее время число уровней может достигать 3. Кэш-память уровня N+1 как правило больше по размеру и медленнее по скорости доступа и передаче данных, чем кэш-память уровня N.

Самой быстрой памятью является кэш первого уровня — L1-cache. По сути, она является неотъемлемой частью процессора, поскольку расположена на одном с ним кристалле и входит в состав функциональных блоков. В современных процессорах обычно кэш L1 разделен на два кэша, кэш команд (инструкций) и кэш данных (Гарвардская архитектура). Большинство процессоров без L1 кэша не могут функционировать. L1 кэш работает на частоте процессора, и, в общем случае, обращение к нему может производиться каждый такт. Зачастую является возможным выполнять несколько операций чтения/записи одновременно. Латентность доступа обычно равна 2−4 тактам ядра. Объём обычно невелик — не более 384 Кбайт.

Вторым по быстродействию является L2-cache — кэш второго уровня, обычно он расположен на кристалле, как и L1. В старых процессорах — набор микросхем на системной плате. Объём L2 кэша от 128 Кбайт до 1−12 Мбайт. В современных многоядерных процессорах кэш второго уровня, находясь на том же кристалле, является памятью раздельного пользования — при общем объёме кэша в nM Мбайт на каждое ядро приходится по nM/nC Мбайта, где nC количество ядер процессора. Обычно латентность L2 кэша, расположенного на кристалле ядра, составляет от 8 до 20 тактов ядра.

Кэш третьего уровня наименее быстродействующий, но он может быть очень внушительного размера — более 24 Мбайт. L3 кэш медленнее предыдущих кэшей, но всё равно значительно быстрее, чем оперативная память. В многопроцессорных системах находится в общем пользовании и предназначен для синхронизации данных различных L2.

Иногда существует и 4 уровень кэша, обыкновенно он расположен в отдельной микросхеме. Применение кэша 4 уровня оправдано только для высоко производительных серверов и мейнфреймов.

Проблема синхронизации между различными кэшами (как одного, так и множества процессоров) решается когерентностью кэша. Существует три варианта обмена информацией между кэш-памятью различных уровней, или, как говорят, кэш-архитектуры: инклюзивная, эксклюзивная и неэксклюзивная.

Инклюзивная архитектура предполагает дублирование информации кэша верхнего уровня в нижнем (предпочитает фирма Intel).

Эксклюзивная кэш-память предполагает уникальность информации, находящейся в различных уровнях кэша (предпочитает фирма AMD).

В неэксклюзивной кэши могут вести себя как угодно.

Ассоциативность кэша

Одна из фундаментальных характеристик кэш-памяти — уровень ассоциативности — отображает её логическую сегментацию. Дело в том, что последовательный перебор всех строк кэша в поисках необходимых данных потребовал бы десятков тактов и свёл бы на нет весь выигрыш от использования встроенной в ЦП памяти. Поэтому ячейки ОЗУ жёстко привязываются к строкам кэш-памяти (в каждой строке могут быть данные из фиксированного набора адресов), что значительно сокращает время поиска. С каждой ячейкой ОЗУ может быть связано более одной строки кэш-памяти: например, N-канальная ассоциативность обозначает, что информация по некоторому адресу оперативной памяти может храниться в N местах кэш-памяти.

При одинаковом объеме кэша схема с большей ассоциативностью будет наименее быстрой,

но наиболее эффективной (после 4-way эффективность на 1 поток растет мало).

Шины данных

5.4 Организация шин. Синхронный и асинхронный протоколы.

Совокупность трактов, объединяющих между собой основные устройства ВМ (центральный процессор, память и модули ввода/вывода), образует структуру взаимосвязей вычислительной машины. Структура взаимосвязей должна обеспечивать обмен информацией между:

- центральным процессором и памятью;
- центральным процессором и модулями ввода/вывода;
- памятью и модулями ввода/вывода.

С развитием вычислительной техники менялась и структура взаимосвязей устройств ВМ (рис. 4.2). На начальной стадии преобладали непосредственные связи между взаимодействующими устройствами ВМ. С появлением мини-ЭВМ, и особенно первых микроЭВМ, все более популярной становится схема с одной общей шиной. Последовавший за этим быстрый рост производительности практически всех устройств ВМ привел к неспособности единственной шины справиться с возросшим трафиком, и ей на смену приходят структуры взаимосвязей на базе нескольких шин. Дальнейшие перспективы повышения производительности вычислений связаны не столько с однопроцессорными машинами, сколько с многопроцессорными вычислительными системами. Способы взаимосвязей в таких системах значительно разнообразнее, и их рассмотрению посвящен один из разделов учебника. Возвращаясь к вычислительным машинам, более внимательно рассмотрим вопросы, связанные с организацией взаимосвязей на базе шин.

DMA – direct memory access, прямой доступ к памяти; режим обмена данными между памятью и устройством ввода/вывода.

Взаимосвязь частей ВМ и ее «общение» с внешним миром обеспечиваются системой шин. Большинство машин содержат несколько различных шин, каждая из которых оптимизирована под определенный вид коммуникаций. Часть шин скрыта внутри интегральных микросхем или доступна только в пределах печатной платы. Некоторые шины имеют доступные извне точки, с тем чтобы к ним легко можно было подключить дополнительные устройства, причем большинство таких шин не просто доступны, но и отвечают определенным стандартам, что позволяет подсоединять к шине устройства различных производителей. , Чтобы охарактеризовать конкретную шину, нужно описать (рис. 4.3):

- совокупность сигнальных линий;
- физические, механические и электрические характеристики шины;
- используемые сигналы арбитража, состояния, управления и синхронизации;
- правила взаимодействия подключенных к шине устройств (протокол шины).

Шину образует набор коммуникационных линий, каждая из которых способна передавать сигналы, представляющие двоичные цифры 1 и 0. По линии может пересылаться развернутая во времени последовательность таких сигналов. При совместном использовании несколько линий могут обеспечить одновременную (параллельную) передачу двоичных чисел. Физически линии шины реализуются в виде отдельных проводников, как полоски

проводящего материала на монтажной плате либо как алюминиевые или медные проводящие дорожки на кристалле микросхемы.

Операции на шине называют транзакциями. Основные виды транзакций — транзакции чтения и транзакции записи. Если в'обмене участвует устройство ввода/вывода, можно говорить о транзакциях ввода и вывода, по сути эквивалентных транзакциям чтения и записи соответственно. Шинная транзакция включает в себя две части: посылку адреса и прием (или посылку) данных.

Когда два устройства обмениваются информацией по шине, одно из них должно инициировать обмен и управлять им. Такого рода устройства назывют ведущими (bus master). В компьютерной терминологии «ведущий» — это любое устройство, способное взять на себя владение шиной и управлять пересылкой данных. Ведущий не обязательно использует данные сам. Он, например, может захватить управление шиной в интересах другого устройства. Устройства, не обладающие возможностями инициирования транзакции, носят название ведомых (bus slave). В принципе к шине может быть подключено несколько потенциальных ведущих, но в любой момент времени активным может быть только один из них: если несколько устройств передают информацию одновременно, их сигналы перекрываются и искажаются. Для предотвращения одновременной активности нескольких ведущих в любой шине предусматривается процедура допуска к управлению шиной только одного из претендентов (арбитраж). В то же время некоторые шины допускают широковещательный режим записи, когда информация одного ведущего передается сразу нескольким ведомым (здесь арбитраж не требуется). Сигнал, направленный одним устройством, доступен всем остальным устройствам, подключенным к шине.

Английский эквивалент термина «шина» — «bus» — восходит к латинскому слову omnibus, означающему «для всего». Этим стремятся подчеркнуть, что шина, ведет себя как магистраль, способная обеспечить всевозможные виды трафика.

Типы шин

Важным критерием, определяющим характеристики шины, может служить ее це- ; левое назначение. По этому критерию можно выделить:

- шины «процессор-память»;
- шины ввода/вывода;
- системные шины.

Протокол шины – метод информирования устройств о достоверности адреса, данных, управляющей информации, информации состояния и т.п.

Существует 2 основных класса протоколов:

1 – синхронный;

2 – асинхронный.

Синхронный протокол.

Признаком синхронного действия шины является наличие генератора тактовой частоты. Тактовые импульсы генерируются генератором и распространяются по специальной сигнальной линии, представляя собой последовательность логических нулей и единиц.

Сигнальная линия:

Один период тактовой последовательности называется тактовым периодом шин и является минимальным квантом действия шин.

Все устройства, подключенные к шине, считывают состояние тактовой сигнальной линии и все события, происходящие на шине, отсчитываются от начала тактового импульса.

Как правило, изменение управляющих сигналов на шине совпадают с передним или задним фронтом тактового импульса.

Стартовым сигналом начала транзакции является присутствие на шине адресной или управляющей информации.

Когда ведомое устройство распознаёт свой адрес, выставленный на шину, оно находит запрошенные данные и помещает их на шину вместе с информацией состояния, сигнализируя ведущему о наличии данных на шине.

Сигналы управления и адреса всегда распространяются от ведущего устройства к ведомому; в обратном направлении распространяются сигналы состояния и подтверждения.

Данные распространяются в обоих направлениях.

Шины проектируются таким образом, что тактовые импульсы, распространяясь вдоль шин, одновременно достигали всех устройств.

Частота тактового генератора должна выбираться таким образом, чтобы сигнал от любой точки шины до другой доходил в пределах одного тактового импульса.

Чем короче синхронная шина, тем выше возможная частота.

Асинхронный протокол.

В асинхронном протоколе начало очередного события на шине определяется предшествующим событием и следует непосредственно за ним.

Каждая совокупность сигналов, помещаемых на шину, сопровождается синхронизирующим сигналом-стробом.

Ведущее выставляет на шину адрес и управляющие сигналы, выжидает время перекоса сигналов, и затем выдаёт строб адреса, подтверждающий достоверность адреса.

Ведомые устройства следят за адресной шиной, чтобы определить, должны ли они реагировать на очередной запрос. Ведомое, распознавшее свой адрес, отвечает ведущему выставлением на шину информации о своём состоянии и сигналом подтверждения адреса.

Когда ведущее обнаруживает сигнал подтверждения адреса, соединение считается установленным.

С этого момента присутствие адреса на шине не обязательно, поскольку адрес уже есть у ведомого.

Ведущее устройство меняет управляющую информацию на шине, выжидает время перекоса сигналов и выдаёт строб данных.

Когда ведомое подготовит запрошенные данные, оно выставляет их на шину вместе с информацией по состоянии и подтверждением достоверности данных.

Когда ведущее обнаружит сигнал подтверждения достоверности данных, оно считывает данные с шины и снимает строб данных, показывая этим, что чтение завершено. Ведомое устройство, обнаружив отсутствие строба данных, снимает выставленные данные с шины, а также информацию о своём состоянии, тем самым освобождая шину.

АРБИТРАЖ

Все асинхронные сигналы шинного арбитража, поступающие в процессор, синхронизируются для внутреннего использования. Как показано на рис.5.17, синхронизация требует максимум одного такта системной

синхронизации, предполагая, что время установки сигнала асинхронного ввода (#47 в разделе электрических характеристик) было воспринято. Сигнал асинхронного ввода опрашивается по срезу такта синхронизации и является
внутренне доступным после следующего среза.

Управление шинным арбитражем реализовано по принципу конечного автомата. Диаграмма состояний (а) на рис.5.18 применима ко всем процессорам
использующим техпроводный шинный арбитраж, а диаграмма состояний (б) применима к процессорам, использующим двухпроводный шинный арбитраж, при котором BGACK# постоянно снят внутренне или внешне. Использован тот же конечный автомат, но с двумя состояниями, так как BGACK# неактивен все время.

На рис.5.18 входные сигналы R и A представляют собой внутренне синхронизированные BR# и BGACK#. Выход BG# показан как G, а внутренний трехстабильный управляющий сигнал как T. Если значение T истинно, шины адреса, данных и управления находятся в состоянии высокого импеданса при пассивном AS#. Все сигналы показаны в положительной логике (активный уровень высокий) независимо от их действительных активных уровней напряжения. Состояние выходов изменяется по фронту тактового импульса, поступающего после того, как внутренний сигнал был разрешен.

Временная диаграмма процесса арбитража в шинном цикле процессора показана на рис.5.19. Временные соотношения для шинного арбитража при неактивной шине (например, когда процессор выполняет внутреннюю обработку инструкции умножения) показаны на рис.5-20.

Если запрос сделан после того, как процессор начал шинный цикл, и до формирования AS# (S0), используется специальная последовательность, показанная на рис.5.21. Вместо активизации по следующему фронту импульса тактирования BG# задерживается до поступления второго фронта после его активизации внутри процессора.

12. Диспетчер. Привилегированные команды. Виртуальная система команд. Защита памяти. Стратегии управления физической памятью. Виртуальная память. Управление виртуальной памятью. Прерывания.

12.1 Диспетчер

Планирование выполнения задач — одна из ключевых концепций в многозадачности и многопроцессорности как в операционных системах общего назначения, так и в операционных системах реального времени. Планирование заключается в назначении приоритетов процессам в очереди с приоритетами. Программный код, выполняющий эту задачу, называется **планировщиком**.

Самой важной целью планирования задач является наиболее полная загрузка процессора. *Производительность* — количество процессов, которые завершают выполнение за единицу времени. *Время ожидания* — время, которое процесс ожидает в очереди готовности. *Время отклика* — время, которое проходит от начала запроса до первого ответа на запрос.

В средах вычислений реального времени, например, на мобильных устройствах, предназначенных для автоматического управления в промышленности (например, робототехника), планировщик задач должен обеспечить отработку процессов в течение заданных временны́х промежутков (время отклика); это критично для поддержания корректной работы системы реального времени.

Долговременный планировщик

Долговременный планировщик решает, какие задачи или процессы будут добавлены в очередь процессов, готовых к выполнению; то есть, когда производится попытка запуска процесса, долговременный планировщик или добавляет новый процесс в очередь готовых процессов (допускает к выполнению), или откладывает это действие. Таким образом, долговременный планировщик решает, какие процессы будут выполняться одновременно, тем самым контролируя степень параллелизма и пропорцию между процессами, интенсивно выполняющими ввод-вывод, и процессами, интенсивно использующими процессор. Обычно в настольных компьютерах не применяется долговременный планировщик и новые процессы допускаются к выполнению автоматически. Но данный планировщик очень важен для систем реального времени, так как при чрезмерной нагрузке системы параллельно выполняющимися процессами время отклика системы может стать больше требуемого, что недопустимо. *[источник не указан 646 дней]*

Среднесрочный планировщик

Во всех системах с виртуальной памятью среднесрочный планировщик временно перемещает (выгружает) процессы из основной памяти во вторичную (например, на жёсткий диск), и наоборот. Эти действия называются подкачкой или свопингом. Среднесрочный планировщик может принять решение выгрузить процесс из основной памяти если:

- процесс был неактивен некоторое время;
- процесс имеет низкий приоритет;
- процесс часто вызывает ошибки страниц (page fault);
- процесс занимает большое количество основной памяти, а системе требуется свободная память для других целей (например, чтобы удовлетворить запрос выделения памяти для другого процесса).

Процесс будет возвращён в основную память, когда будет доступно необходимое количество свободной памяти или когда процесс выйдет из режима ожидания (в этом случае планировщик выгрузит из основной памяти другой процесс для освобождения основной памяти). *[источник не указан 646 дней]*

Во многих современных системах, поддерживающих отображение виртуального адресного пространства на вторичную память, отличную от файла подкачки, среднесрочный планировщик может одновременно играть роль и долговременного планировщика, рассматривая новые процессы как процессы, которые были выгружены из основной памяти. Таким образом система может подгружать в основную память программный код только тогда, когда он понадобится процессу для выполнения (это называется загрузкой по требованию или «ленивой загрузкой»). *[источник не указан 646 дней]*

Краткосрочный планировщик

Планировщик на этом уровне решает, какие из готовых и загруженных в память процессов будут запущены на ЦПУ после прерывания (по времени, операции ввода-вывода, вызову операционной системы или другому сигналу). Решения на этом уровне приходится принимать очень часто (как минимум, каждый временной отрезок). Также планировщик может поддерживать или не поддерживать вытесняющую многозадачность (то есть иметь возможность прервать исполнение какого-либо процесса).

Диспетчер

Диспетчер — это еще один компонент системы планирования. Это модуль, который передает

управление процессором тому процессу, который был выбран на уровне кратковременного планирования. В его задачи входит переключение контекста, переключение в пользовательский режим и прыжок к нужному месту пользовательской программы, чтобы начать или продолжить ее исполнение. Главное требование к диспетчеру — это быстродействие, поскольку он осуществляет каждое переключение процессов.

12.2 Привилегированные команды

Термин “привилегия” подразумевает права или возможности, которые обычно не разрешаются, а разрешаются только в порядке исключения из общих правил. Введение неравноправия программ в виде уровней привилегий (уровней PL) является средством защиты программных сегментов и сегментов данных операционной системы. Защищаются программы ОС различных уровней иерархии от ошибок в пользовательских программах и в программах операционной системы более низких уровней иерархии.

Операционные системы обычно имеют свою иерархическую структуру, свои логические уровни управления. Программы более высокого логического уровня управления активно используют программы более низкого уровня. Таким образом, ОС изначально имеют свои уровни управления, где каждый логический уровень является сервисным для программ более высокого уровня. При этом, чем ниже логический уровень программ, тем большими возможностями по управлению они обладают. Поэтому сбои в этих программах могут привести к более тяжелым последствиям, и степень их защиты должна быть выше.

В микропроцессоре установлено 4 уровня привилегий (PL) , которые задаются номерами от 0 до 3.

Наиболее привилегированным является уровень с наименьшим номером. Степень защищенности сегмента также имеет 4 уровня, которые схематически представляются в виде вложенных колец защиты.

3 – программы пользователя

2 – служебные программы

1 – Драйверы устройств и утилиты ОС

0 – ядро ОС

Чем меньше номер уровня, тем меньше его логический уровень управления, но тем более он привилегирован и имеет большую степень защиты.

Наименее защищенными являются прикладные программы пользователя, которым присваивается уровень с номером 3. Остальные уровни отводятся для системных программ, которые разделяются на 3 уровня в зависимости от требований к их защищенности. Наиболее защищенная часть - это ядро ОС, которой присвоен уровень 0. В ядро входит часть ОС, обеспечивающая инициализацию работы, а также управление доступом к памяти и другие функции, нарушение которых может полностью вывести ОС из строя. Основная часть программ ОС имеет уровень 1. К этому уровню, в частности, относятся драйверы устройств и утилиты.

На втором и третьем уровнях располагаются программы разработчиков комплексных систем,

например СУБД, и программы пользователей соответственно.

Проверка защиты по уровням привилегий осуществляется при выполнении почти каждой машинной команды во время работы микропроцессора в защищенном режиме (P – режиме).

Операционная система необязательно должна поддерживать все четыре уровня привилегий. Например, система UNIX имеет всего 2 уровня: операционной системе присвоен номер 0, а программам пользователей – уровень 3. Система OS/2 поддерживает три уровня: программы ОС работают на 0 уровне, специальные процедуры для обращения к устройствам ввода/вывода действуют на уровне 2, а прикладные программы пользователей выполняются на уровне 3.

3.2 Задание уровней привилегий.

Основными объектами механизма защиты по привилегиям являются сегменты программ и данных, а также шлюзы. Именно им назначаются уровни привилегий. Уровень привилегий относительно содержимого сегмента следует считать глобальным, т.е. не может быть, чтобы какая-то часть сегмента данных была более привилегированной, чем другие части. Но ОС имеет возможность определять один сегмент несколькими дескрипторами с разными уровнями привилегий и даже с разными размерами самого сегмента.

Уровень привилегий сегмента или шлюза определяет поле DPL , которое находится в байте прав доступа AR соответствующего дескриптора. Таким образом, каждый сегмент или шлюз имеет свой персональный уровень привилегий DPL.

Кроме уровня привилегий сегмента DPL существует понятие текущего уровня привилегий CPL (от англ. Current Privilege Level или Code Privilege Level). Текущий уровень привилегий CPL определяется полем DPL дескриптора текущего программного сегмента. После загрузки программного сегмента в ОЗУ и передачи на него управления значение DPL копируется в поле CPL регистра программного сегмента CS. После этого уровень привилегий программного сегмента DPL становится текущим уровнем привилегий CPL. Таким образом, текущий уровень привилегий является уровнем привилегий исполняемого программного сегмента.

Кроме уровней привилегий DPL и CPL , механизм защиты использует понятие уровня привилегий запроса RPL (Requested Privilege Level). Этот уровень привилегий задается двумя младшими разрядами селектора. Уровень привилегий запроса RPL селекторов сегментов данных сохраняется в соответствующих сегментных регистрах SS, DS, ES, FS, GS. Понятие уровня привилегий запроса RPL служит для защиты программ и данных операционной системы от ошибок программ, изменивших уровень привилегий при межуровневых передачах управления.

Защита памяти

- набор регистров процессора, используемый им при проверке достоверности доступа при каждом обращении к памяти и, собственно, механизм ее проверки. Различают три основных

вида защиты:
а) Защита памяти по граничным регистрам. В процессор специально добавляется два дополнительных регистра, в которые ОС при выделении некоторого куска памяти какому-либо процессу записывает начальные и конечные адреса данного куска памяти.
При попытке обращения процессом к какой-либо ячейке памяти, проверяется, попадает ли данный адрес в пределы между значениями, хранящимися в данный момент в граничных регистрах.
В случае непопадания в данный интервал вызывается прерывание по защите памяти, которое, в свою очередь, обрабатывается операционной системой, и она решает, что в данном случае предпринять. Как правило, в случае нарушения защиты памяти ОС аварийно завершает процесс.
б) Защита памяти по ключу. Оперативная память (ОЗУ) разбивается на некоторые сегменты, каждому из которых присваивается некоторый уникальный номер - ключ. В свою очередь, у каждого из процессов имеется также свой уникальный номер, называемый идентификатором процесса. Для каждого из процесов, каждому сегменту памяти, принадлежащему данному процессу, присваивается его (процесса) идентификатор. При обращении процесса к какому-либо сегменту памяти сличаются оба ключа. При несовпадении генерируется прерывание.
в) Защита по базированию. В процессоре существует дополнительный регистр, называемый базисным (базовым), в который записывается адрес того куска памяти, который принадлежит данному процессу. Процесс, в свою очередь, всегда «считает», что ему доступна вся физическая память, и что он может использовать ее всю, начиная с нулевого адреса. На самом деле, при любом обращении к памяти, к тому адресу, по которому обращается процесс,
добавляется значение базового регистра, после чего получается реальный адрес в физической памяти.
3. Защита команд. Команды делятся на непривилегированные (команды, которыми может пользоваться любая программа) и привилегированные (команды, которые может использовать только операционная система). При попытке использования процессом привилегированной команды вызывается прерывание по защите команд. В этом случае управление передается операционной системе, которая решает, что делать с данным процессом - либо аварийно завершить, либо произвести определенные действия (системные вызовы).

12.4 Управление физической памятью

Управление физической памятью заключается в распределении страниц и отдельных блоков памяти и их освобождении. За распределение страниц памяти отвечает менеджер основной физической памяти (primary physical memory manager) ядра Linux. В его функции входит
• Выделение/освобождение физических страниц
• Распределение областей физически непрерывных страниц по запросу

Учет доступных физических страниц осуществляется менеджером по собственному алгоритму, известному как принцип buddy-heap (смежных куч, куч близнецов). Его идея в том, что при освобождении двух смежных областей памяти, доступных для использования, происходит объединение этих областей в результате чего появляется новая область большего размера. При обработке запроса на выделение небольшого блока памяти, Страничный менеджер разделяет доступную свободную область памяти на две, меньшего размера. Системой ведутся списки свободных областей памяти, по размеру доступных областей.

Наименьшей выделяемой областью памяти в Linux является физическая страница. На рис.9 показан процесс разделения памяти в buddy-heaps.
Как правило, гораздо чаще требуются небольшие области памяти. Для обработки запросов произвольного размера ядро поддерживает функцию kmalloc, выделяющую память постранично, разделяя области на более мелкие. Ядро ведет учет страниц распределяемых с помощью функции kmalloc. Память, распределенная с помощью функции kmalloc , может быть освобождена лишь в явном виде. При недостатке памяти, эти области не могут быть использованы повторно при помощи kmalloc.

4) Связные и несвязные распределения памяти. Стратегия управления физической памятью: мультипрограммирование с фиксированными разделами. Фрагментация памяти.

Определение 14. Управление памятью - это распределение памяти между процессами, т.е. выдача или изъятие некоторого объема памяти у процессов и, быть может, защита этой памяти от посягательств иных процессов.
Различают связные и несвязные распределения памяти. При связном распределении каждому процессу при запросе на какой-либо объем памяти (в том числе при инициализации процесса) выдается один-единственный
непрерывный раздел. В данном случае процесс предполагает, что выданная память абсолютная - все адреса, которые он использует, равны либо адресам физической памяти, либо отличаются одной константой. В этом случае говорят о распределении физической памяти. При несвязном распределении при запросе может выдаваться несколько разных разделов памяти, которые имеют право находиться в разных частях физической оперативной памяти. В этом случае при обращении процессом к какому-либо адресу алгоритм вычисления настоящего физического адреса усложняется. В этом случае говорят о распределении виртуальной памяти.

Управление физической памятью с фиксированными разделами
При генерации системы вся физическая память делится на разделы различной или одинаковой длины. При инициализации процесса ищется свободный сегмент с подходящей длиной (больше или равной). Если такого
сегмента нет, то данный процесс откладывается в очередь и дожидается освобождения раздела. Важно заметить, что для любой задачи должен существовать раздел подходящей длины. Существуют два подхода при создании вышеупомянутых очередей:
1) очередь к каждому из сегментов. В очереди к какому-либо разделу находятся только процессы, длина которых подходит под данный раздел. Адрес процесса может устанавливаться уже на этапе компиляции. В этом случае возможен вырожденный случай - у одного из разделов выстраивается огромная очередь, остальные простаивают;
2) очередь одна для всех процессов. При освобождении какого-либо участка памяти, из очереди выбирается первый процесс, подходящий для данного раздела.
Среди недостатков данного подхода можно отметить:
1) никогда заведомо не известен объем нужного участка памяти;
2) существуют большие потери оперативной памяти - фрагментация:
а) внутренняя - простаивает некоторое количество памяти внутри раздела;
б) внешняя - простаивают целые разделы, которые не подходят по размеру.

5) Стратегии управления физической памятью. Мультипрограммирование с переменными разделами. Динамическое распределение памяти. Методы первого подходящего и наиболее подходящего.

При инициализации системы предполагается, что есть непрерывный кусок памяти (возможно вся оперативная память). Задачей системы управления памятью с переменными разделами является реализация двух процедур:
1)выдача некоторого объема памяти по запросу;
2) освобождение уже ненужного участка памяти.
Управление физической памятью делят на:
1. Динамическое управление с явным освобождением.
1) Метод первого подходящего. При запросе некоторого объема памяти выдается первый встретившийся свободный участок памяти достаточного размера. Выбор первого подходящего по размеру участка является достаточно рациональным, поскольку он позволяет быстро принять решение о размещении задания.
2) Метод наилучшего подходящего. При запросе выдается тот свободный участок основной памяти, в котором остается минимально возможное неиспользуемое пространство.
В качестве достоинства явного освобождения можно отметить то, что не требуется время на запуск и выполнение так называемого сборщика мусора (он может работать вхолостую). Главный недостаток состоит в том, что не все процессы освобождают память.
2. Динамическое управление с неявным освобождением - время от времени запускается сборщик мусора, который освобождает уже неиспользуемую память.

6) Динамическое распределение памяти. Метод близнецов.
1. Динамическое управление с явным освобождением.
Вся память делится на куски, размеры которых кратны степени двойки. При запросе некоторого объема памя-
ти, требуемая длина округляется до ближайшей степени двойки.
Близнецами называются два стоящих рядом куска памяти одинакового размера. Если известен адрес одного из близнецов размера 2^k - A1, то адрес другого вычисляется по формуле A2 = A1 XOR 2^k

В качестве достоинства явного освобождения можно отметить то, что не требуется время на запуск и выполнение так называемого сборщика мусора (он может работать вхолостую). Главный недостаток состоит в том, что не все процессы освобождают память.
2. Динамическое управление с неявным освобождением - время от времени запускается сборщик мусора, который освобождает уже неиспользуемую память.

Виртуальная память. Основные концепции. Страничная организация памяти.

Пользовательская программа составляется в предположении, что все используемое ею адресное пространство непрерывно.
В системах с управлением виртуальной памятью вся ОП разбивается на участки одинаковой или различной длины и используется так называемое несвязное распределение памяти, поэтому память, принадлежащая какому-либо процессу, может быть неадекватно воспринята с точки зрения пользователя.
Различают два вида организации виртуальной памяти:
1) в случае, когда непрерывные участки физической памяти имеют одинаковый размер, говорят о страничной организации виртуальной памяти, а сами участки памяти называют страницами;
2) если же непрерывные участки памяти имеют различную длину, то говорят о сегментной организации виртуальной памяти, а участки памяти называются сегментами.

Страничная организация виртуальной памяти

При данной организации виртуальной памяти вся физическая память разбивается на страницы, размер которых задается в системе раз и навсегда.
Когда пользовательская программа обращается к памяти, реально происходит обращение не к физической памяти, а к так называемому виртуальному адресному пространству, т.е. адрес, по которому обращается пользователь, не является физическим адресом, а состоит из двух частей:
1) номер виртуальной страницы (как правило, старшие разряды);
2) смещение относительно нее.
В системе с каждым из процессов связана таблица виртуальных страниц этого процесса. Эта таблица представляет из себя массив, в котором хранятся адреса физических страниц.
Для получения необходимого адреса в физической памяти происходит индексация по массиву (т.е. по таблице виртуальных страниц), после чего к выбранному оттуда физическому адресу дописывается смещение.
Для преобразования виртуальных адресов в физические процессор снабжен соответствующим аппаратным механизмом.
Для более эффективного преобразования адресов размер страницы выбирается, как правило, равным степени двойки.
При обращении пользователя к какому-либо адресу памяти происходит как бы двойное обращение к оперативной памяти: сначала обращение в ОП к таблице виртуальных страниц, а затем еще одно обращение - уже по необходимому адресу (рис. 2).
Для повышения эффективности данных обращений используется так называемая ассоциативная память - сверхбыстрая кэш-память (cache). Отличия этой памяти от обычной следующие:
1) скорость доступа к кеш-памяти значительно выше, но, в силу сложности технологического процесса, она и более дорогая (почему ее, соответственно, обычно бывает мало);
2) оперативная память представляет собой фиксированный набор ячеек (набор пар «аргумент-значение»), при этом множество аргументов (адресов) - константны, а множество аргументов в ассоциативной памяти изменяется во времени. Ассоциативная память - не массив (по ней нельзя индексироваться), а набор пар «ключ-значение». При обращении к АП на вход подается ключ, процессор ищет его в АП и выдает соответствующее данному ключу значение. Поиск по АП реализуется методом бинарного поиска (поиска по двоичному ключу).
При использовании ассоциативной памяти алгоритм поиска адреса физической памяти будет следующим:
1) берем из адреса номер виртуальной страницы и сначала обращаемся к АП. Используя его в качестве ключа, смотрим, есть ли в АП такой ключ, и, если есть, то значение адреса физической страницы берем как соответствующее данному ключу значение;
2) если в АП не оказалось нужного ключа, то обращаемся к оперативной памяти (в таблицу виртуальных страниц) и берем физический адрес оттуда. После чего:
- если в ассоциативной памяти есть хотя бы одна свободная ячейка, то она заполняется соответствующими номером виртуальной страницы и адресом физической страницы;
- если же свободных ячеек в АП нет, то с помощью некоторого алгоритма (называемого алгоритмом замещения или стратегией замещения) одна из ячеек АП убирается, и на ее место записываются новые значения.
Существует четыре вида стратегий замещения:
1) случайное замещение - ячейка для замещения выбирается случайным образом;
2) стратегия FIFO выгружается ячейка, которая дольше всего находится в памяти;
3) стратегия LFU (Least Frequently Used) - замещается та ячейка, к которой реже всего обращаются;
4) стратегия LRU (Least Recently Used) - замещается ячейка, к которой дольше всего не обращаются.

8) Виртуальная память: сегментная и сегментно-страничная организация.
При сегментной организации виртуальной памяти вся физическая память делится на сегменты. Ответственность за разбиение всех данных на сегменты лежит на пользователе. Сегментная организация памяти почти ничем не отличается от страничной, за исключением следующих моментов:
1) вместо таблицы страниц существует таблица сегментов;
2) наряду с характеристиками сегментов, такими же, как у страниц, добавляется размер сегмента, характеристика расширяемости (максимальная длина), местоположение сегмента в физической памяти.
Каждый сегмент рассматривается как отдельное виртуальное адресное пространство, и обращение к нему идет с помощью виртуального адреса, который реально получается из двух составляющих номера сегмента и смещения внутри его.
Сегментно-страничная организация памяти
При сегментно-страничной организации существуют сегменты, которые рассматриваются как отдельное виртуальное адресное пространство, которое в свою очередь отображается на физические страницы, т.е. в системе на аппаратном уровне организована страничная организация памяти, и виртуальный адрес складывается из двух составлящих: смещение в таблице сегментов и виртуальное смещение внутри сегмента.
При обращении по виртуальному адресу сначала идет индексация по таблице сегментов, затем выбирается адрес таблицы страниц данного сегмента, после этого производится индексирование по этой таблице страниц (индекс берется из второй составляющей виртуального адреса, т.е. реально вторая часть сама состоит из двух частей: номер страницы и смещение внутри нее), после чего для получения физического адреса, вышеописанное смещение добавляется к адресу, взятому из таблицы страниц (рис. 3).
При наличии кэш-памяти обращение сначала идет к ней, причем ключом служит пара (номер сегмента, номер страницы). В остальном способ получения физического адреса аналогичен соотвествующему способу, используемому в случае страничной организации памяти.

12.5 Управление виртуальной памятью

Существуют следующие стратегии управления памятью:
1. Стратегия размещения. Определение местоположения виртуальной страницы (сегмента). Стратегия размещения решает вопрос о том, в какое место оперативной памяти поместить данную страницу (сегмент). В случае страничной организации памяти задача является тривиальной - происходит поиск и выдача первой свободной физической страницы. В случае сегментной организации обычно используют динамическое распределение памяти (например, метод наилучшего подходящего).
2. Стратегия замещения. Определение страницы (сегмента), которую необходимо заменить. Стратегия замещения решает вопрос о том, какую страницу (сегмент) временно удалить из памяти, чтобы на ее (его) место записать данные. В данном случае используются описанные выше алгоритмы замещения. При использовании этой стратегии система может решить вообще не подкачивать данную страницу в данный момент времени, а подождать, когда какой-либо из процессов освободит свой сегмент или страницу. Реально этот механизм в операционных системах не используется.
Различают два вида стратегий замещения:
1) локальная стратегия. В данном случае замещаются страница или сегмент, принадлежащие только данному процессу;
2) глобальная стратегия. Замещение происходит среди всего множества страниц.
3. Стратегия подкачки. Существуют две стратегии подкачки. Если существует алгоритм,

который позволяет определить, какая из виртуальных страниц данного процесса потребуется в ближайшее время, то можно принять решение о заблаговременной подкачке.
Данная стратегия называется подкачкой с упреждением. Если такого алгоритма не существует, то говорят о подкачке по требованию. Стратегия подкачки по требованию более проста в реализации, но при хорошем алгоритме предсказания стратегия с упреждением дает значительный выигрыш во времени.

2) Прерывания. Действия аппаратуры и ОС при прерывании.

Наступление того или иного события сигнализируется прерываниями Interrupt. Источниками прерываний могут быть как аппаратура (HardWare), так и программы (SoftWare).
Аппаратура сообщает о прерывании асинхронно (в любой момент времени) путем пересылки в CPU через общую шину сигналов прерываний.
Программа сообщает о прерывании путем выполнения операции System Call. Примеры событий, вызывающих прерывания:
1) попытка деления на 0;
2) запрос на системное обслуживание;
3) завершение операции ввода-вывода;
4) неправильное обращение к памяти.
Каждое прерывание обрабатывается соответственно обработчиком прерываний (Interrupt handler), входящим в состав ОС.
Главные функции механизма прерываний это:
1) распознавание или классификация прерываний;
2) передача управления соответственно обработчику прерываний;
3) корректное возвращение к прерванной программе.
Переход от прерываемой программы к обработчику и обратно должен выполняться как можно быстрей. Одним из быстрых методов является использование таблицы, содержащей перечень всех допустимых для компьютера прерываний и адреса соответствующих обработчиков. Такая таблица называется вектором прерываний (Interrupt vector) и хранится в начале адресного пространства основной памяти (UNIX/MS DOS).
Для корректного возвращения к прерванной программе перед передачей управления обработчику прерываний содержимое регистров процессора запоминается либо в памяти с прямым доступом, либо в системном стеке System Stack.
Обычно запрещаются прерывания обработчика прерываний. Однако в некоторых ОС прерывания снабжаются приоритетами, то есть работа обработчика прерывания с более низким приоритетом может быть прервана,
если произошло прерывание с более высоким приоритетом.

2) Прерывания. Действия аппаратуры и ОС при прерывании.

Наступление того или иного события сигнализируется прерываниями Interrupt. Источниками прерываний могут быть как аппаратура (HardWare), так и программы (SoftWare).
Аппаратура сообщает о прерывании асинхронно (в любой момент времени) путем пересылки в CPU через общую шину сигналов прерываний.
Программа сообщает о прерывании путем выполнения операции System Call. Примеры событий, вызывающих прерывания:
1) попытка деления на 0;
2) запрос на системное обслуживание;
3) завершение операции ввода-вывода;

4) неправильное обращение к памяти.
Каждое прерывание обрабатывается соответственно обработчиком прерываний (Interrupt handler), входящим в состав ОС.
Главные функции механизма прерываний это:
1) распознавание или классификация прерываний;
2) передача управления соответственно обработчику прерываний;
3) корректное возвращение к прерванной программе.
Переход от прерываемой программы к обработчику и обратно должен выполняться как можно быстрей. Одним из быстрых методов является использование таблицы, содержащей перечень всех допустимых для компьютера прерываний и адреса соответствующих обработчиков. Такая таблица называется вектором прерываний (Interrupt vector) и хранится в начале адресного пространства основной памяти (UNIX/MS DOS).
Для корректного возвращения к прерванной программе перед передачей управления обработчику прерываний содержимое регистров процессора запоминается либо в памяти с прямым доступом, либо в системном стеке System Stack.
Обычно запрещаются прерывания обработчика прерываний. Однако в некоторых ОС прерывания снабжаются приоритетами, то есть работа обработчика прерывания с более низким приоритетом может быть прервана,
если произошло прерывание с более высоким приоритетом.

13.Свойство локальности программ. Основные концепции планирования загрузки процессора. Процессы. Состояния процессов. Стратегии диспетчеризации. Многоуровневые очереди с обратной связью.

13.1 Свойство локальности программ.

Существует два вида локальности: пространственная и временная.
Определение 15. Говорят, что процесс обладает пространственной локальностью, если в течение довольно длительного времени он использует небольшое количество виртуальных страниц.
Определение 16. Говорят, что процесс обладает временной локальностью, если в течение довольно длительного времени множество его виртуальных страниц остается неизменным.
Определение 17. Если процесс обладает временной локальностью, то множество страниц, которые он использует, называется его рабочим множеством.
В этом случае говорят, что рабочее множество виртуальных таблиц процесса стабильно, в противном случае - изменчиво.
Исходя из принципа локальности программ, не обязательно держать в ОП все виртуальные страницы процесса.
Как правило, в системе виртуальной памяти используется внешнее запоминающее устройство, на которое копируются неактивные страницы. При попытке обращения процесса к виртуальной странице, которая в данный момент находится на внешнем запоминающем устройстве, данная страница загружается обратно в физическую память. При этом применяется одна из вышеописанных стратегий замещения.
При выгрузке данной страницы из ОП проверяется, была ли она модифицирована, и, если да, то ее образ на внешнем устройстве обновляется.
Если процессу предоставить меньше оперативной памяти, чем объем множества его рабочих

страниц, то процесс начинает активно подкачивать свои страницы со внешнего устройства. Такая ситуация называется *пробуксовкой*. Если процесс начал активно подкачивать свои страницы, это еще не означает, что он буксует. Пробуксовку можно легко устранить, предоставив процессу дополнительный объем доступной ему оперативной памяти.
Если же подкачка страниц происходит не из-за пробуксовки, то исправить это никак нельзя, и в этом случае к процессу обычно применяют некоторые штрафные санкции.
В связи с вышесказанным, таблица виртуальных страниц должна содержать еще, как минимум:
1) адрес страницы в ОП;
2)флаг модификации страницы (была ли модификация);
3)права доступа к странице (необходимы для того, чтобы система могла разделять виртуальные страницы между процессами, например процессами, использующими одни и те же данные).
Т.к. таблица страниц может быть достаточно большой и не умещаться на одной физической странице, то иногда делают таблицу страниц таблицы страниц, т.е. таблицу страниц тоже размещают в виртуальном адресном пространстве.
В этом случае сначала происходит обращение к ассоциативной памяти (если таковая есть), после чего идет обращение к таблице страниц таблицы страниц, после этого - обращение к таблице страниц и, наконец, обращение к физической памяти.
Определение 18. Таблица страниц таблицы страниц называется таблицей страниц второго уровня.
Уровни в таблицах страниц можно повышать при условии, что таблица самого верхнего уровня полностью умещается в одну физическую страницу, а таблицы промежуточных уровней умещаются в одну виртуальную страницу.

13.2 Основные концепции планирования загрузки процессора.

1.4.1. Состояния процессов

Процессор является вторым по значимости ресурсом компьютера (первый - память). Подготовленная к счету задача помещается на так называемый буфер ввода. На ее основе формируется процесс. Данное формирование называется инициализацией процесса и заключается в присвоении уникального номера данному процессу в вычислительной смеси, который называется идентификатором процесса, и в предоставлении ему ресурсов, необходимых для начала работы процесса, кроме, быть может, самого процессора. В данном случае говорят, что процессу предоставлен виртуальный процессор.
Существующий процесс, прошедший стадию инициализации, может находиться в одном из следующих состояний:
1) текущий процесс;
2) готовый процесс;
3) заблокированный процесс;
4) приостановленный процесс.
Процесс, который в данный момент выполняется на процессоре, называется текущим. Если процесс готов к счету, но ему не предоставлен процессор, то такой процесс называется готовым и говорят, что он находится в состоянии ожидания.
Если текущий процесс освободил процессор добровольно (произошло нормальное или аварийное завершение), то система берет процесс из числа готовых и помещает его в процессор.
Процесс может добровольно освободить процессор в случае операции обмена (вводавывода). В данном случае система блокирует процесс (который, соответственно, называется заблокированным), причем все захваченные им ресурсы остаются у него. При принятии решения о разблокировании процесса (это может произойти по окончании операции вводавывода, например при сигнале контроллера), процесс становится готовым, т.е.

переводится в состояние ожидания.
Три рассмотренных выше состояния называются активными. В некоторых системах есть еще пассивное состояние. Такое состояние называется приостановленным. У приостановленного процесса отбираются все захваченные им ресурсы с условием, что они будут ему возвращены при возобновлении его работы. Это состояние полезно для корректировки баланса вычислительной смеси и разрешения некоторых тупиковых ситуаций. Если в системе существует такое состояние, то обычно после инициализации процесса, он переводится в состояние приостановленности. Иначе в состояние готовности.

1.4.3. Уровни планирования

При прохождении через компьютер процесс мигрирует между различными очередями под управлением программы, которая называется «планировщик» (scheduler). Операционная система, обеспечивающая режим мультипрограммирования, обычно включает два планировщика долгосрочный (long term scheduler) и краткосрочный (short term scheduler/CPU scheduler).
Основное отличие между долгосрочным и краткосрочным планировщиками заключается в частоте запуска, например: краткосрочный планировщик может запускаться каждые 100 мс, долгосрочный - один раз за несколько минут.
Долгосрочный планировщик решает, какой из процессов, находящихся во входной очереди, должен быть переведен в очередь готовых процессов в случае освобождения ресурсов памяти.
Долгосрочный планировщик выбирает процесс из входной очереди с целью создания неоднородной мультипрограммной смеси. Это означает, что в очереди готовых процессов должны находиться в разной пропорции
как процессы, ориентированные на ввод-вывод, так и процессы, ориентированные на преимущественную работу с процессором.
Краткосрочный планировщик решает, какой из процессов, находящихся в очереди готовых процессов, должен быть передан на выполнение в процессор. В некоторых операционных системах долгосрочный планировщик
может отсутствовать. Например, в системах разделения времени (time sharing system) каждый новый процесс сразу же помещается в основную память.

13.3 Процессы.

13.2. Состояния процессов.

13.3 Стратегии диспетчеризации

Для диспетчеризации с перераспределением процессора используются
следующие дисциплины:
1) SRT (Shortest Remaining Time Next). Дисциплина SRT аналогична SJN, но при принудительной смене контекста ожидаемое время выполнения вновь поступившей задачи сравнивается с оставшимся временем выполнения текущей. Если вновь поступившая задача короче, чем оставшееся время выполняемого процесса, то контекст сменяется.
В идеале SRN минимизирует среднее время ожидания процесса на множестве готовых задач, но значительно увеличивает дисперсию и время выполнения долгих задач.
2) RR (Round Robin). С каждым из процессов связывается некая постоянная или изменяемая во времени величина, называемая диспетчерским приоритетом процесса. Все готовые процессы складываются в очередь, в которой они упорядочены по приоритету. При смене контекста, из очереди выбираются процессы с наивысшим приоритетом.
Кроме того, для каждого процесса хранится еще одна величина q_i , тоже постоянная или изменяемая во времени, которая является квантом времени и определяет, какое максимальное

непрерывное время данный процесс может находиться в процессоре. После истечения данного кванта времени, контекст принудительно сменяется.
3

13.4 Многоуровневые очереди с обратной связью.

) Многоуровневые очереди с обратной связью. В системе существует некоторое множество очередей (n очередей), каждая из которых представляет из себя очередь FIFO. С каждой из очередей связано два параметра q_i (квант времени) и m_i (число повторений). Число повторений - это величина, обозначающая, какое количество раз задача может быть помещена в процессор для обработки, находясь в очереди данного уровня. В самом начале процесс помещают в очередь самого верхнего уровня - в наиболее приоритетную очередь. Как только число повторений для данного процесса превзошло число повторений в его очереди, его перемещают на уровень ниже в менее приоритетную очередь. Попав на нижний уровень, задача остается там до своего завершения (т.е. m n = inf).
Между q_i и m_i должны соблюдаться следующие соотношения:
q 1 << q 2 << . . . << q_n,

m 1 << m 2 << . . . << m_n.
Величины q_i и m_i стараются выбирать достаточно большими, чтобы короткие задачи успевали выполняться на очередях верхнего уровня, и достаточно малыми, чтобы дать возможность выполняться долгим процессам, но вовремя опускать их вниз.
Диспетчеризация с использованием многоуровневых очередей с обратной связью позволяет оперативно реагировать на производительность вычислительной смеси. Так, обменная задача (процесс, который очень часто занимается операциями ввода-вывода и добровольно освобождает процессор) может поощряться. Для поощрения используют два метода:
1) счетчик повторений данного процесса устанавливается в начальное состояние;
2) процесс поднимается на уровень выше в системе очередей.