Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

В. С. ЮДЕНКОВ, Д. М. СЕВОСТЬЯН

ИНФОРМАЦИОННЫЕ СЕТИ

Тексты лекций
по одноименному курсу для студентов специальности
1-40 01 02-03 «Информационные системы и технологии
(издательско-полиграфический комплекс)»

Минск 2006

УДК 681.518(075.8)
ББК 32.98я7
Ю 16

Рассмотрены и рекомендованы к изданию редакционно-издательским советом университета

Рецензенты:
старший научный сотрудник лаборатории вычислительных сетей ОИПИ НАН Беларуси кандидат технических наук, доцент *А. А. Несенчук*;
заведующий кафедрой программного обеспечения вычислительной техники и автоматизированных систем БНТУ кандидат технических наук, доцент *Н. А. Разоренов*

Юденков, В. С.

Ю 16 Информационные сети : тексты лекций по одноименному курсу для студентов специальности 1-40 01 02-03 «Информационные системы и технологии (издательско-полиграфический комплекс)» / В. С. Юденков, Д. М. Севостьян. – Мн. : БГТУ, 2006. – 144 с.

ISBN 985-434-578-5

Тексты лекций содержат теоретический и практический материал, необходимый для изучения основ проектирования локальных вычислительных сетей и работы в сети Internet на основе технологии серверного программирования (PHP, ASP-NET, JavaScript).

В данном пособии рассматривается программная модель системы массового обслуживания (СМО), система команд и операторы языка GPSS, приводятся примеры программ для моделирования ЛВС и методы их отладки на базе ПЭВМ IBM PC/AT.

УДК 681.518(075.8)
ББК 32.98я7

ISBN 985-434-578-5

ПРЕДИСЛОВИЕ

Тексты лекций включают приемы моделирования систем массового обслуживания, в том числе АСУ полиграфическими предприятиями и сетей ЭВМ на базе пакета GPSS для персональных ЭВМ типа IBM PC/AT.

В пособии рассматриваются формы представления моделей систем массового обслуживания для реализации их на ЭВМ.

В разделах 1–7 приводятся архитектура и топология вычислительных сетей, характеризуются сетевые операционные системы, сетевое программное обеспечение, а также аппаратное обеспечение вычислительных сетей.

В разделе «Программирование имитационных моделей информационных сетей в среде GPSS/PC» рассматриваются описание и работа операторов и блоков пакета GPSS для составления программ статистического моделирования систем массового обслуживания.

Описание блоков и операторов сопровождается большим количеством примеров, облегчающих процесс усвоения материала.

Раздел «Серверное программирование на основе языка PHP» посвящен изучению программирования в вычислительных сетях на основе языка PHP.

Тексты лекций предназначены для обучения студентов специальности 1-40 01 02-03 «Информационные системы и технологии (издательско-полиграфический комплекс)».

1. ОБЗОР И АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ СЕТЕЙ

1.1. Основные определения и термины

Сеть – это совокупность объектов, образуемых устройствами передачи и обработки данных. Международная организация по стандартизации определила вычислительную сеть как последовательную бит-ориентированную передачу информации между связанными друг с другом независимыми устройствами.

Сети обычно находятся в частном ведении пользователя и занимают некоторую территорию. По территориальному признаку они разделяются:

– на локальные вычислительные сети (ЛВС), или Local Area Network (LAN), расположенные в одном или нескольких близко стоящих зданиях. ЛВС обычно размещаются в рамках какой-либо организации (корпорации, учреждения), поэтому их называют корпоративными;

– на распределенные компьютерные сети, глобальные, или Wide Area Network (WAN), расположенные в разных зданиях, городах и странах. Они бывают территориальными, смешанными и глобальными. Глобальные сети бывают четырех основных видов: городские, региональные, национальные и транснациональные. В качестве примеров распределенных сетей очень большого масштаба можно назвать Internet, EUNET, Relcom, FIDO.

В состав сети в общем случае включаются следующие элементы:

– сетевые компьютеры (оснащенные сетевым адаптером);

– каналы связи (кабельные, спутниковые, телефонные, цифровые, волоконно-оптические, радиоканалы и др.);

– различного рода преобразователи сигналов;

– сетевое оборудование.

Различают два понятия: коммуникационная сеть и информационная сеть.

Коммуникационная сеть предназначена для передачи данных, также она выполняет задачи, связанные с преобразованием данных. Эти сети различаются по типу используемых физических средств соединения. На базе коммуникационной сети может быть построена группа информационных сетей (рис. 1).

Информационная сеть предназначена для хранения информации. Она состоит из информационных систем.

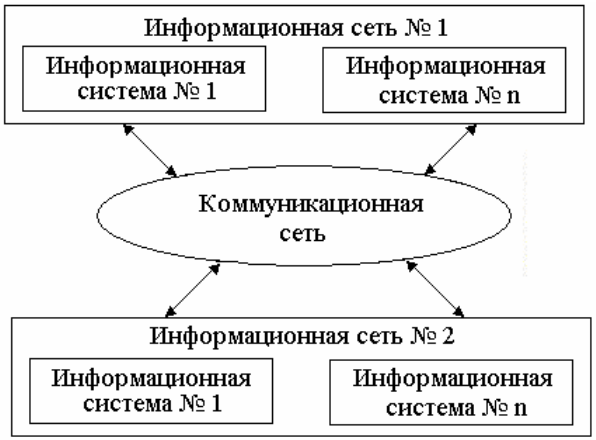


Рис. 1. Информационные и коммуникационные сети

Под информационной системой следует понимать систему, которая является поставщиком или потребителем информации.

Компьютерная сеть состоит из информационных систем и каналов связи.

Под **информационной системой** следует понимать объект, способный осуществлять хранение, обработку или передачу информация. В состав информационной системы входят: компьютеры, программы, пользователи и другие составляющие, предназначенные для процесса обработки и передачи данных. В дальнейшем информационная система, предназначенная для решения задач пользователя, будет называться рабочая станция (client). Рабочая станция в сети отличается от обычного персонального компьютера (ПК) наличием сетевой карты (сетевого адаптера), канала для передачи данных и сетевого программного обеспечения.

Под **каналом связи** следует понимать путь или средство, по которому передаются сигналы. Средство передачи сигналов называют абонентским, или физическим, каналом.

Каналы связи (data link) создаются по линиям связи при помощи сетевого оборудования и физических средств связи. Физические средства связи построены на основе витых пар, коаксиальных кабелей, оптических каналов или эфира. Между взаимодействующими информационными системами через физические каналы коммуникационной сети и узлы коммутации устанавливаются логические каналы.

Логический канал – это путь для передачи данных от одной системы к другой. Его можно охарактеризовать как маршрут, проложенный через физические каналы и узлы коммутации.

Информация в сети передается блоками данных по процедурам обмена между объектами. Эти процедуры называются протоколами передачи данных.

Протокол – это совокупность правил, устанавливающих формат и процедуры обмена информацией между двумя или несколькими устройствами.

Загрузка сети характеризуется параметром, называемым трафиком. **Трафик** (traffic) – это поток сообщений в сети передачи данных. Под ним понимается количественное измерение в выбранных точках сети числа проходящих блоков данных и их длины, выраженное в битах в секунду.

Существенное влияние на характеристику сети оказывает метод доступа. **Метод доступа** – это способ определения того, какая из рабочих станций сможет следующей использовать канал связи и как управлять доступом к каналу связи (кабелю).

В сети все рабочие станции физически соединены между собою каналами связи, построенными по определенной структуре, называемой топологией. **Топология** – это описание физических соединений в сети, указывающее, какие рабочие станции могут связываться между собой. Тип топологии определяется производительностью, работоспособностью и надежностью эксплуатации рабочих станций, а также временем обращения к файловому серверу. В зависимости от топологии сети используется тот или иной метод доступа.

Состав основных элементов в сети зависит от ее архитектуры. **Архитектура** – это концепция, определяющая взаимосвязь, структуру и функции взаимодействия рабочих станций в сети. Она предусматривает логическую, функциональную и физическую организацию технических и программных средств сети. Архитектура определяет принципы построения и функционирования аппаратного и программного обеспечения элементов сети.

Современные сети можно классифицировать по различным признакам: по удаленности компьютеров, топологии, назначению, перечню предоставляемых услуг, принципам управления (централизованные и децентрализованные), методам коммутации, методам доступа, видам среды передачи, скоростям передачи данных и т. д. Все эти понятия будут рассмотрены более подробно при дальнейшем изучении курса.

1.2. Преимущества использования сетей

Компьютерные сети представляют собой вариант «сотрудничества» людей и компьютеров, обеспечивающего ускорение доставки и обработки информации. Объединять компьютеры в сети начали более 30 лет назад. В результате усовершенствования компьютеров, а также в результате того, что ПК стали доступны каждому, развитие сетей значительно ускорилось.

Соединенные в сеть компьютеры обмениваются информацией и совместно используют периферийное оборудование и устройства хранения информации рис. 2.

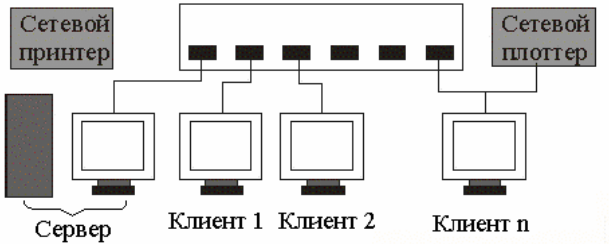


Рис. 2. Использование периферийного оборудования

С помощью сетей можно разделять ресурсы и информацию. Ниже перечислены основные задачи, которые решаются с помощью рабочей станции в сети и которые трудно решить с помощью отдельного компьютера.

Компьютерная сеть позволяет совместно использовать:

1) периферийные устройства:
– принтеры;
– плоттеры;
– дисковые накопители;
– приводы CD-ROM;
– дисководы;
– стримеры;
– сканеры;
– факс-модемы;

2) информационные ресурсы:
– каталоги;

– файлы;
– прикладные программы;
– игры;
– базы данных;
– текстовые процессоры.

Компьютерная сеть позволяет работать с многопользовательскими программами, обеспечивающими одновременный доступ всех пользователей к общим базам данных с блокировкой файлов и записей, гарантирующей целостность данных. Любые программы, разработанные для стандартных ЛВС, можно использовать в других сетях.

Совместное использование ресурсов обеспечивает существенную экономию средств и времени. Например, можно коллективно использовать один лазерный принтер, который подключен к любому из компьютеров, входящих в сеть.

Можно использовать ЛВС как почтовую службу и рассылать служебные записки, доклады и сообщения другим пользователям.

1.3. Архитектура сетей

Архитектура сети определяет основные элементы сети, характеризует ее общую логическую организацию, техническое обеспечение, программное обеспечение, описывает методы кодирования. Архитектура также определяет принципы функционирования и интерфейс пользователя.

В основном выделяются три вида архитектур:
архитектура «терминал – главный компьютер»;
одноранговая архитектура;
архитектура «клиент – сервер».

Архитектура «терминал – главный компьютер». Архитектура «терминал – главный компьютер» (terminal / host computer architecture) – это концепция информационной сети, в которой вся обработка данных осуществляется одним или группой главных компьютеров (рис. 3).

Рассматриваемая архитектура предполагает наличие двух типов оборудования:

1) главного компьютера, с которого осуществляется управление сетью, хранение и обработка данных;

2) терминалов, предназначенных для передачи главному компьютеру команд на организацию сеансов, для выполнения заданий ввода данных и получения результатов.

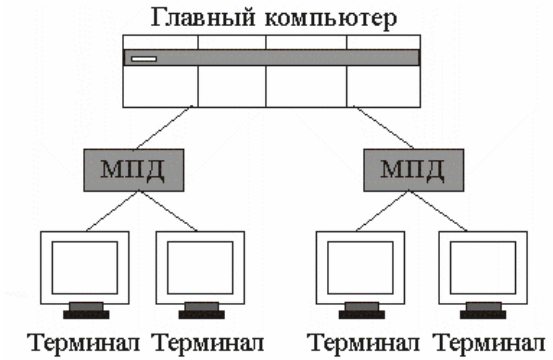


Рис. 3. Архитектура «терминал – главный компьютер»

Главный компьютер через мультиплексоры передачи данных (МПД) взаимодействует с терминалами по схеме, представленной на рис. 3.

Основное достоинство данной архитектуры состоит в организации многопользовательского режима работы только с одним главным компьютером. Основным же недостатком этой системы является невозможность хранения данных на индивидуальных жестких дисках.

Классический пример архитектуры сети с главными компьютерами – системная сетевая архитектура (System Network Architecture – SNA).

Одноранговая архитектура. Одноранговая архитектура (peer-to-peer architecture) – это концепция информационной сети, в которой ее ресурсы рассредоточены по всем системам. Данная архитектура характеризуется тем, что в ней все системы равноправны (рис. 4).

К одноранговым сетям относятся малые сети, где любая рабочая станция может выполнять одновременно функции файлового сервера и рабочей станции.

В одноранговых ЛВС дисковое пространство и файлы на любом компьютере могут быть общими. Чтобы ресурс стал общим, его необходимо отдать в общее пользование, применяя службы удаленного доступа сетевых одноранговых операционных систем. В зависимости от того, как будет установлена защита данных, другие пользователи смогут пользоваться файлами сразу же после их создания. Одноранговые ЛВС достаточно хороши только для небольших рабочих групп.

Рис. 4. Одноранговая архитектура

Одноранговые ЛВС являются наиболее легким и дешевым типом сетей для установки. Они требуют, кроме сетевой карты и сетевого носителя, только операционную систему Windows NT или Windows XP. При соединении компьютеров пользователи могут предоставлять ресурсы и информацию в совместное пользование.

Одноранговые сети имеют следующие преимущества:

– они легко устанавливаются и настраиваются;

– отдельные ПК не зависят от выделенного сервера;

– пользователи могут самостоятельно контролировать свои ресурсы;

– имеют низкую стоимость и легкость в эксплуатации;

– требуют минимум оборудования и программного обеспечения;

– не нуждаются в администраторе;

– хорошо подходят для сетей с количеством пользователей, не превышающим десяти.

Недостатками данных сетей являются следующие:

– в случаях, когда компьютеры отключаются от сети, из сети исчезают виды сервиса, которые они предоставляли;

– возможность применения сетевой безопасности только к одному ресурсу. В этом случае пользователь должен помнить столько паролей, сколько сетевых ресурсов. При получении доступа к разделяемому ресурсу ощущается падение производительности компьютера;

– отсутствие централизованного администрирования.

Использование одноранговой архитектуры не исключает применения в той же сети и архитектуры «терминал – главный компьютер» или архитектуры «клиент – сервер».

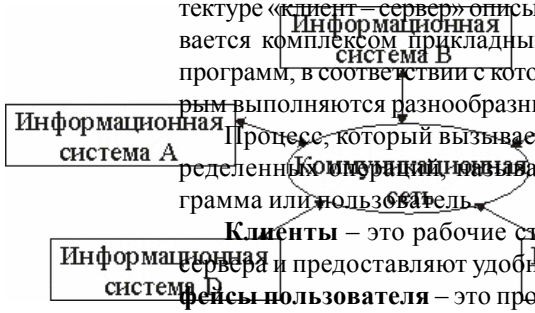
Архитектура «клиент – сервер». Архитектура «клиент – сервер» (client / server architecture) – это концепция информационной сети, в которой основная часть ее ресурсов сосредоточена в серверах, обслуживающих своих клиентов (рис. 5). Рассматриваемая архитектура определяет два типа компонентов: серверы и клиенты.

Сервер – это объект, предоставляющий сервис другим объектам сети по их запросам. Сервис – это процесс обслуживания клиентов.

Сервер работает по заданиям клиентов и управляет выполнением их заданий. После выполнения каждого задания он посылает полученные результаты клиенту, выславшему это задание.

Рис. 5. Архитектура «клиент – сервер»

Сервисная функция в архитектуре «клиент – сервер» описывается комплексом прикладных программ, в соответствии с которым выполняются разнообразные прикладные процессы.



Процесс, который вызывает сервисную функцию с помощью определенных операций, называется клиентом. Им может быть программа или пользователь.

Клиенты – это рабочие станции, которые используют ресурсы сервера и предоставляют удобные интерфейсы пользователя. **Интерфейсы пользователя** – это процедуры взаимодействия пользователя с системой или сетью.

Клиент является инициатором и использует электронную почту или другие сервисы сервера. В этом процессе клиент запрашивает вид обслуживания, устанавливает сеанс, получает нужные ему результаты и сообщает об окончании работы.

В сетях с выделенным файловым сервером на выделенном автономном ПК устанавливается серверная сетевая операционная система. Этот ПК становится сервером. Программное обеспечение (ПО), установленное на рабочей станции, позволяет ей обмениваться дан-

ными с сервером. Наиболее распространенные сетевые операционная системы: NetWare фирмы Novel; Windows NT фирмы Microsoft; UNIX фирмы AT&T; Linux.

Помимо сетевой операционной системы необходимы сетевые прикладные программы, реализующие преимущества, предоставляемые сетью.

Сети на базе серверов имеют лучшие характеристики и повышенную надежность. Сервер владеет главными ресурсами сети, к которым обращаются остальные рабочие станции.

В современной клиентско-серверной архитектуре выделяется четыре группы объектов: клиенты, серверы, данные и сетевые службы. Клиенты располагаются в системах на рабочих местах пользователей. Данные в основном хранятся в серверах. Сетевые службы являются совместно используемыми серверами и данными. Кроме того, службы управляют процедурами обработки данных.

Сети клиентско-серверной архитектуры имеют следующие преимущества:

– позволяют организовывать сети с большим количеством рабочих станций;

– обеспечивают централизованное управление учетными записями пользователей, безопасностью и доступом, что упрощает сетевое администрирование;

– предоставляют эффективный доступ к сетевым ресурсам;

– требуют один пароль для входа в сеть и для получения доступа ко всем ресурсам, на которые распространяются права пользователя.

Наряду с преимуществами сети клиентско-серверной архитектуры имеют и ряд недостатков:

– неисправность сервера может сделать сеть неработоспособной, как минимум, с потерей сетевых ресурсов;

– требуют квалифицированного персонала для администрирования;

– имеют более высокую стоимость сетей и сетевого оборудования.

Выбор архитектуры сети. Выбор архитектуры сети зависит от назначения сети, количества рабочих станций.

Следует выбрать одноранговую сеть, если:

– количество пользователей не превышает десяти;

– все машины расположены близко друг к другу;

– финансовые возможности ограничены;

– нет необходимости в специализированном сервере, таком, как сервер БД, факс-сервер или какой-либо другой;

– нет возможности или необходимости в централизованном администрировании.

Следует выбрать клиентско-серверную сеть, если:

– количество пользователей превышает десяти;

– требуется централизованное управление, безопасность, управление ресурсами или резервное копирование;

– необходим специализированный сервер;

– нужен доступ к глобальной сети;

– требуется разделять ресурсы на уровне пользователей.

2. СЕМИУРОВНЕВАЯ МОДЕЛЬ OSI

Для единого представления данных в сетях с неоднородными устройствами и программным обеспечением Международная организация по стандартизации (International Standardization Organization – ISO) разработала базовую модель связи открытых систем OSI (Open System Interconnection). Эта модель описывает правила и процедуры передачи данных в различных сетевых средах при организации сеанса связи. Основными элементами модели являются уровни, прикладные процессы и физические средства соединения. Каждый уровень модели OSI выполняет определенную задачу в процессе передачи данных по сети. Базовая модель является основой для разработки сетевых протоколов. В сети OSI разделяет коммуникационные функции на семь уровней, каждый из которых обслуживает различные части процесса области взаимодействия открытых систем.

Модель OSI описывает только системные средства взаимодействия, не касаясь приложений конечных пользователей. Приложения реализуют свои собственные протоколы взаимодействия, обращаясь к системным средствам. Если приложение может взять на себя функции некоторых верхних уровней модели OSI, то для обмена данными оно обращается напрямую к системным средствам, выполняющим функции оставшихся нижних уровней модели OSI.

Модель OSI можно разделить:

– на горизонтальную модель на базе протоколов, обеспечивающую механизм взаимодействия программ и процессов на различных машинах;

– на вертикальную модель на основе услуг, обеспечиваемых соседними уровнями друг другу на одной машине.

Каждый уровень компьютера-отправителя взаимодействует с таким же уровнем компьютера-получателя, как будто он связан напрямую. Такая связь называется логической, или виртуальной, связью. В действительности взаимодействие осуществляется между смежными уровнями одного компьютера.

Итак, информация на компьютере-отправителе должна пройти через все уровни. Затем она передается по физической среде компьютеру-получателю и опять проходит через все слои, пока не доходит до того же уровня, с которого она была послана.

В горизонтальной модели двум программам требуется общий протокол для обмена данными. В вертикальной модели соседние уровни обмениваются данными с использованием интерфейсов прикладных программ API (Application Programming Interface).

Перед подачей в сеть данные разбиваются на пакеты. **Пакет (packet)** – это единица информации, передаваемая между станциями сети. При отправке данных пакет проходит последовательно через все уровни программного обеспечения. На каждом уровне к пакету добавляется управляющая информация данного уровня (заголовок), которая необходима для успешной передачи данных по сети. На принимающей стороне пакет проходит через все уровни в обратном порядке. На каждом уровне протокол этого уровня читает информацию пакета, затем удаляет информацию, добавленную к пакету на этом же уровне отправляющей стороной, и передает пакет следующему уровню. Когда пакет дойдет до прикладного уровня, вся управляющая информация будет удалена из пакета, и данные примут свой первоначальный вид.

Каждый уровень модели выполняет свою функцию. Чем выше уровень, тем более сложную задачу он решает.

Отдельные уровни модели OSI удобно рассматривать как группы программ, предназначенных для выполнения конкретных функций. Один уровень, к примеру, отвечает за обеспечение преобразования данных из ASCII в EBCDIC и содержит программы, необходимые для выполнения этой задачи.

Каждый уровень обеспечивает сервис для вышестоящего уровня, запрашивая, в свою очередь, сервис у нижестоящего уровня. Верхние уровни запрашивают сервис почти одинаково: как правило, это требование маршрутизации каких-то данных из одной сети в другую. Практическая реализация принципов адресации данных возложена на нижние уровни.

Рассматриваемая модель определяет взаимодействие открытых систем разных производителей в одной сети. Поэтому для них она совершает координирующие действия:

– по взаимодействию прикладных процессов;

– по формам представления данных;

– по единообразному хранению данных;

– по управлению сетевыми ресурсами;

– по безопасности данных и защите информации;

– по диагностике программ и технических средств.

3. ТОПОЛОГИЯ ВЫЧИСЛИТЕЛЬНОЙ СЕТИ. МЕТОДЫ ДОСТУПА

Топология (конфигурация) – это способ соединения компьютеров в сеть. Тип топологии определяет стоимость, защищенность, производительность и надежность эксплуатации рабочих станций, для которых имеет значение время обращения к файловому серверу.

Понятие топологии широко используется при создании сетей. Одним из подходов к классификации топологий ЛВС является выделение двух основных классов топологий: широковещательных и последовательных.

В широковещательных топологиях ПК передает сигналы, которые могут быть восприняты остальными ПК. К таким топологиям относятся топологии: «общая шина», «дерево», «звезда».

В последовательных топологиях информация передается только одному ПК. Примерами таких топологий являются: произвольная (произвольное соединение ПК), «кольцо», «цепочка».

При выборе оптимальной топологии преследуются три основные цели:

– обеспечение альтернативной маршрутизации и максимальной надежности передачи данных;

– определение оптимального маршрута передачи блоков данных;

– предоставление приемлемого времени ответа и нужной пропускной способности.

При выборе конкретного типа сети важно учитывать ее топологию. Основными сетевыми топологиями являются: шинная (линейная) топология, звездообразная, кольцевая и древовидная.

Например, в конфигурации сети ARCNet используются одновременно и линейная, и звездообразная топология. Сети Token Ring физически выглядят как звезда, но логически их пакеты передаются по кольцу. Передача данных в сети Ethernet происходит по линейной шине, так что все станции видят сигнал одновременно.

3.1. Виды топологий

Существует пять основных топологий (рис. 6):

– «общая шина» (Bus);

– «кольцо» (Ring);

– «звезда» (Star);

– «дерево» (Tree).

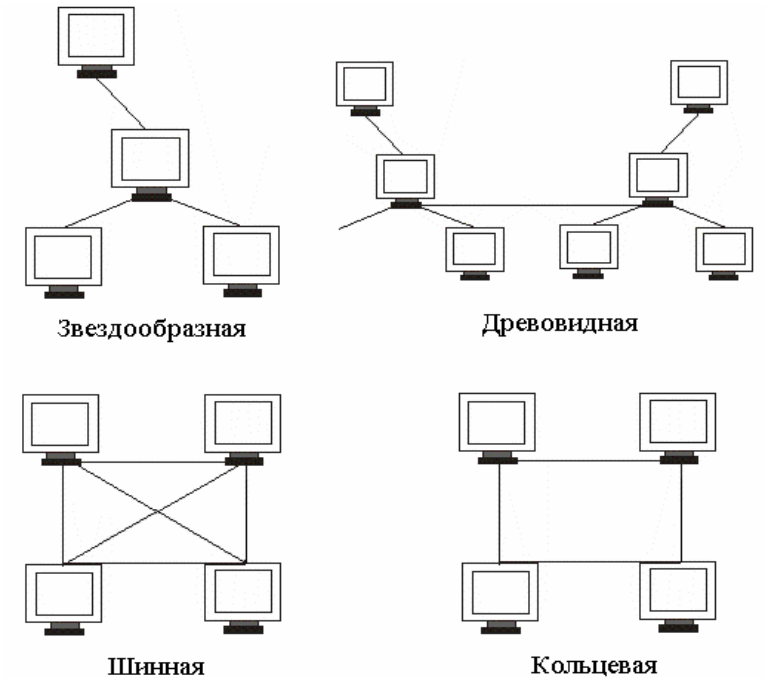


Рис. 6. Типы топологий

Общая шина. Это тип сетевой топологии, в которой рабочие станции расположены вдоль одного участка кабеля, называемого сегментом.

Топология «общая шина» (рис. 7) предполагает наличие одного кабеля, к которому подключаются все компьютеры сети. Кабель используется всеми станциями по очереди. Принимаются специальные меры для того, чтобы при работе с общим кабелем компьютеры не мешали друг другу передавать и принимать данные. Все сообщения, посылаемые отдельными компьютерами, принимаются и прослушиваются всеми остальными компьютерами, подключенными к сети. Рабочая станция отбирает адресованные ей сообщения, пользуясь адресной информацией. Надежность здесь выше, так как выход из строя отдельных компьютеров не нарушит работоспособность сети в целом. Поиск неисправности в сети затруднен. Кроме того, поскольку исполь-

зуется только один кабель, в случае обрыва нарушается работа всей сети. Шинная топология – это наиболее простая и наиболее распространенная топология сети.

Рис. 7. Топология «общая шина»

Примерами использования топологии «общая шина» является сеть 10Base-5 (соединение ПК толстым коаксиальным кабелем) и 10Base-2 (соединение ПК тонким коаксиальным кабелем).

Кольцо. Это топология ЛВС, в которой каждая станция соединена с двумя другими станциями, образуя кольцо (рис. 8). Данные передаются от одной рабочей станции к другой в одном направлении (по кольцу). Каждый ПК работает как повторитель, ретранслируя сообщения к следующему ПК, т. е. данные передаются от одного компьютера к другому как бы по эстафете. Если компьютер получает данные, предназначенные для другого компьютера, он передает их дальше по кольцу. Очень просто делается запрос на все станции одновременно.

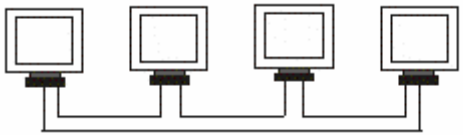


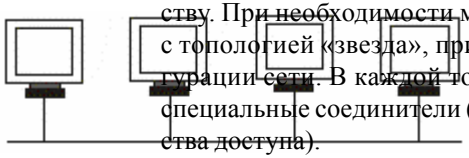
Рис. 8. Топология «кольцо»

Основная проблема при кольцевой топологии заключается в том, что каждая рабочая станция должна активно участвовать в пересылке информации, и в случае выхода из строя хотя бы одной из них, вся сеть парализуется. Подключение новой рабочей станции требует кратковременного выключения сети, так как во время установки кольцо должно быть разомкнуто. Топология «кольцо» имеет хорошо предсказуемое время отклика, определяемое числом рабочих станций.

Чистая кольцевая топология используется редко. Чаще кольцевая топология играет транспортную роль в схеме метода доступа. Кольцо описывает логический маршрут, а пакет передается от одной станции к другой, совершая в итоге полный круг. В сетях Token Ring кабельная ветвь из центрального концентратора называется MAU (Multiple Access Unit). MAU имеет внутреннее кольцо, соединяющее все подключенные к нему станции, и используется как альтернативный путь, когда оборван или отсоединен кабель одной рабочей станции. Когда кабель рабочей станции подсоединен к MAU, он просто образует расширение кольца: сигналы поступают к рабочей станции, а затем возвращаются обратно во внутреннее кольцо.

Звезда. Это топология ЛВС, в которой все рабочие станции присоединены к центральному узлу (например, к концентратору), который устанавливает, поддерживает и разрывает связи между рабочими станциями (рис. 9). Преимуществом такой топологии является возможность простого исключения неисправного узла. Однако, если неисправен центральный узел, вся сеть выходит из строя.

В этом случае каждый компьютер через специальный сетевой адаптер подключается отдельным кабелем к объединяющему устройству. При необходимости можно объединять вместе несколько сетей с топологией «звезда», при этом получаются разветвленные конфигурации сети. В каждой точке ветвления необходимо использовать специальные соединители (распределители, повторители или устройства доступа).



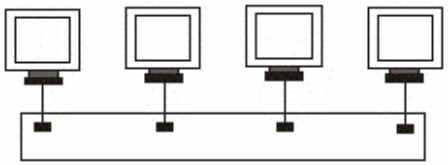


Рис. 9. Топология «звезда»

Примером звездообразной топологии является топология Ethernet с кабелем типа «витая пара» (10BASE-T), центром звезды обычно является Hub.

Звездообразная топология обеспечивает защиту от разрыва кабеля. Если кабель рабочей станции будет поврежден, это не приведет

к выходу из строя всего сегмента сети. Также она позволяет легко диагностировать проблемы подключения, так как каждая рабочая станция имеет свой собственный кабельный сегмент, подключенный к концентратору. Для диагностики достаточно найти разрыв кабеля, который ведет к неработающей станции. Остальная часть сети продолжает нормально работать.

Однако звездообразная топология имеет и недостатки. Во-первых, она требует много кабеля. Во-вторых, концентраторы довольно дорогие. В-третьих, кабельные концентраторы при большом количестве кабеля трудно обслуживать. Однако в большинстве случаев в такой топологии используется недорогой кабель типа «витая пара». В некоторых случаях можно даже использовать существующие телефонные кабели. Кроме того, для диагностики и тестирования выгодно собирать все кабельные концы в одном месте. По сравнению с концентраторами ARCNet концентраторы Ethernet и MAU Token Ring достаточно дорогие. Новые подобные концентраторы включают в себя средства тестирования и диагностики, что делает их еще более дорогими.

3.2. Методы доступа

Метод доступа – это способ определения того, какая из рабочих станций сможет следующей использовать ЛВС. То, как сеть управляет доступом к каналу связи (кабелю), существенно влияет на ее характеристики.

Примерами методов доступа являются:

– множественный доступ с прослушиванием несущей и разрешением коллизий (Carrier Sense Multiple Access with Collision Detection – CSMA/CD);

– множественный доступ с передачей полномочия (Token Passing Multiple Access – TPMA), или метод с передачей маркера;

– множественный доступ с разделением во времени (Time Division Multiple Access – TDMA);

– множественный доступ с разделением частоты (Frequency Division Multiple Access – FDMA) или множественный доступ с разделением длины волны (Wavelength Division Multiple Access – WDMA).

CSMA/CD. Метод множественного доступа с прослушиванием несущей и разрешением коллизий устанавливает следующий порядок: если рабочая станция хочет воспользоваться сетью для передачи данных, то

сначала она должна проверить состояние канала, поскольку начинать передачу станция может, только если канал свободен.

В процессе передачи станция продолжает прослушивание сети для обнаружения возможных конфликтов. Если возникает конфликт из-за того, что два узла пытаются занять канал, то обнаружившая конфликт интерфейсная плата выдает в сеть специальный сигнал, и обе станции одновременно прекращают передачу. Принимающая станция отбрасывает частично принятое сообщение, а все рабочие станции, желающие передать сообщение, в течение некоторого, случайно выбранного промежутка времени выжидают.

Все сетевые интерфейсные платы запрограммированы на разные псевдослучайные промежутки времени. Если конфликт возникнет во время повторной передачи сообщения, этот промежуток времени будет увеличен. Стандарт типа Ethernet определяет сеть с конкуренцией, в которой несколько рабочих станций должны конкурировать друг с другом за право доступа к сети.

TPMA. Метод с передачей маркера – это метод доступа к среде, при котором от одной рабочей станции к другой передается маркер, дающий разрешение на передачу сообщения. При получении маркера рабочая станция может передавать сообщение, присоединяя его к маркеру, который переносит это сообщение по сети. Каждая станция между передающей станцией и принимающей видит это сообщение, но только станция-адресат принимает его. При этом она создает новый маркер.

Маркер (token), или полномочие, – уникальная комбинация битов, позволяющая начать передачу данных.

Каждый узел принимает пакет от предыдущего, восстанавливает уровни сигналов до номинального уровня и передает дальше. Передаваемый пакет может содержать данные или являться маркером. Когда рабочей станции необходимо передать пакет, ее адаптер дожидается поступления маркера, а затем преобразует его в пакет, содержащий данные, отформатированные по протоколу соответствующего уровня, и передает результат далее по ЛВС.

Пакет распространяется по ЛВС от адаптера к адаптеру, пока не найдет своего адресата, который установит в нем определенные биты для подтверждения того, что данные достигли адресата, и ретранслирует его вновь в ЛВС. После этого пакет возвращается в узел, из которого был отправлен. Здесь после проверки безошибочной передачи пакета узел освобождает ЛВС, выпуская новый маркер. Таким образом, в ЛВС с

передачей маркера невозможны коллизии (конфликты). Метод с передачей маркера в основном используется в кольцевой топологии.

Данный метод характеризуется следующими достоинствами:

– гарантирует определенное время доставки блоков данных в сети;

– дает возможность предоставления различных приоритетов передачи данных.

Вместе с тем он имеет существенные недостатки:

– в сети возможны потеря маркера, а также появление нескольких маркеров, при этом сеть прекращает работу;

– включение и отключение новой рабочей станции связаны с изменением адресов всей системы.

4. ЛОКАЛЬНАЯ ВЫЧИСЛИТЕЛЬНАЯ СЕТЬ (ЛВС)

Локальная вычислительная сеть (ЛВС) – это совокупность компьютеров, каналов связи, сетевых адаптеров, работающих под управлением сетевой операционной системы и сетевого программного обеспечения.

В ЛВС каждый ПК называется рабочей станцией, за исключением одного или нескольких компьютеров, которые предназначены для выполнения функций файл-серверов. Каждая рабочая станция и файл-сервер имеют сетевые карты (адаптеры), которые посредством физических каналов соединяются между собой. В дополнение к локальной операционной системе на каждой рабочей станции активизируется сетевое программное обеспечение, позволяющее станции взаимодействовать с файловым сервером.

Компьютеры, входящие в ЛВС клиентско-серверной архитектуры, делятся на два типа: рабочие станции, или клиенты, предназначенные для пользователей, и файловые серверы, которые, как правило, недоступны для обычных пользователей и предназначены для управления ресурсами сети.

Аналогично на файловом сервере запускается сетевое программное обеспечение, которое позволяет ему взаимодействовать с рабочей станцией и обеспечить доступ к своим файлам.

4.1. Основные компоненты

Компьютерная сеть состоит из трех основных аппаратных и двух программных компонентов, которые должны работать согласованно. Для корректной работы устройств в сети их нужно правильно инсталлировать и установить рабочие параметры.

Основными аппаратными компонентами сети являются следующие:

1. Абонентские системы:

– компьютеры (рабочие станции или клиенты и серверы);

– принтеры;

– сканеры и др.

2. Сетевое оборудование:

– сетевые адаптеры;

– концентраторы (хабы);

– мосты;
– маршрутизаторы и др.
3. Коммуникационные каналы:
– кабели;
– разъемы;
– устройства передачи и приема данных в беспроводных технологиях.

Основными программными компонентами сети являются следующие:

1. Сетевые операционные системы, где наиболее известные из них: Windows NT, Windows for Workgroups, LANtastic, NetWare, UNIX, Linux и др.

2. Сетевое программное обеспечение (сетевые ресурсы): клиент сети; сетевая карта; протокол; служба удаленного доступа.

4.2. Рабочие станции

Рабочая станция (workstation) – это абонентская система, специализированная для решения определенных задач и использующая сетевые ресурсы. К сетевому программному обеспечению рабочей станции относятся следующие службы:

– клиент для сетей;
– служба доступа к файлам и принтерам;
– сетевые протоколы для данного типа сетей;
– сетевая плата;
– контроллер удаленного доступа.

Рабочая станция отличается от обычного автономного персонального компьютера следующими признаками:

– она имеет сетевую карту (сетевой адаптер) и канал связи;
– на экране во время загрузки ОС появляются дополнительные сообщения, которые информируют о том, что загружается сетевая операционная система;
– перед началом работы необходимо сообщить сетевому программному обеспечению имя пользователя и пароль. Это называется процедурой входа в сеть;
– после подключения к ЛВС появляются дополнительные сетевые дисковые накопители;
– появляется возможность использования сетевого оборудования, которое может находиться далеко от рабочего места.

4.3. Сетевые адаптеры

Для подключения ПК к сети требуется устройство сопряжения, которое называют сетевым адаптером, интерфейсом, модулем, или картой. Оно вставляется в гнездо материнской платы. Карты сетевых адаптеров устанавливаются на каждой рабочей станции и на файловом сервере. К файловому серверу рабочая станция отправляет запрос и получает ответ через сетевой адаптер, когда файловый сервер готов.

Сетевые адаптеры вместе с сетевым программным обеспечением способны распознавать и обрабатывать ошибки, которые могут возникнуть из-за электрических помех, коллизий или плохой работы оборудования.

Последние типы сетевых адаптеров поддерживают технологию Plug and Play (вставляй и работай). Если сетевую карту установить в компьютер, то при первой загрузке система определит тип адаптера и запросит для него драйверы.

Типы сетевых адаптеров различаются не только методами доступа к каналу связи и протоколами, но еще и такими параметрами, как:

– скорость передачи;
– объем буфера для пакета;
– тип шины;
– быстродействие шины;
– совместимость с различными микропроцессорами;
– использование прямого доступа к памяти (DMA);
– адресация портов ввода/вывода и запросов прерывания;
– конструкция разъема.

4.4. Файловые серверы

Сервер – это компьютер, предоставляющий свои ресурсы (диски, принтеры, каталоги, файлы и т. п.) другим пользователям сети.

Файловый сервер обслуживает рабочие станции. В настоящее время это обычно быстродействующий ПК на базе процессоров Pentium, работающие с тактовой частотой 500 МГц и выше, с объемом ОЗУ 128 Мбайт или более. Чаще всего файловый сервер выполняет только эти функции. Но иногда в малых ЛВС файл-сервер используется еще и в качестве рабочей станции. На файловом сервере должна стоять сетевая операционная система, а также сетевое программное обес-

печение. К сетевому программному обеспечению сервера относятся сетевые службы и протоколы, а также средства администрирования сервера.

Файловые серверы могут контролировать доступ пользователей к различным частям файловой системы. Это обычно осуществляется разрешением присоединить некоторую файловую систему (или каталог) к рабочей станции пользователя для дальнейшего использования как локального диска.

По мере усложнения возлагаемых на серверы функций и увеличения числа обслуживаемых ими клиентов происходит все большая специализация серверов. Существует множество типов серверов.

– Первичный контроллер домена – сервер, на котором хранится база бюджетов пользователей и поддерживается политика защиты.

– Вторичный контроллер домена – сервер, на котором хранится резервная копия базы бюджетов пользователей и политики защиты.

– Универсальный сервер, предназначенный для выполнения несложного набора различных задач обработки данных в локальной сети.

– Сервер базы данных, выполняющий обработку запросов, направляемых базе данных.

– Прокси-сервер, подключающий локальную сеть к сети Internet.

– Web-сервер, предназначенный для работы с Web-информацией.

– Файловый сервер, обеспечивающий функционирование распределенных ресурсов, включая файлы, программное обеспечение.

– Сервер приложений, предназначенный для выполнения прикладных процессов. С одной стороны, он взаимодействует с клиентами, получая задания, а с другой стороны, работает с базами данных, подбирая данные, необходимые для обработки.

– Сервер удаленного доступа, обеспечивающий сотрудникам, работающим дома, торговым агентам, служащим филиалов, лицам, находящимся в командировках, возможность работы с данными сети.

– Телефонный сервер, предназначенный для организации в локальной сети службы телефонии. Этот сервер выполняет функции голосовой почты, осуществляет автоматическое распределение вызовов, учет стоимости телефонных разговоров, связь интерфейса с внешней телефонной сетью. Наряду с телефонией сервер может также передавать изображения и сообщения факсимильной связи.

– Почтовый сервер, предоставляющий сервис в ответ на запросы, присланные по электронной почте.

– Сервер доступа, дающий возможность коллективного использования ресурсов пользователями, оказавшимися вне своих сетей (например, пользователями, которые находятся в командировках и хотят работать со своими сетями). Для этого пользователи через коммуникационные сети соединяются с сервером доступа и последний предоставляет нужные ресурсы, имеющиеся в сети.

– Терминальный сервер, объединяющий группу терминалов, упрощающий переключения при их перемещении.

– Коммуникационный сервер, выполняющий функции терминального сервера, но осуществляющий также маршрутизацию данных.

– Видеосервер, который в наибольшей степени приспособлен к обработке изображений, снабжает пользователей видеоматериалами, обучающими программами, видеоиграми, обеспечивает электронный маркетинг. Имеет высокую производительность и большую память.

– Факс-сервер, обеспечивающий передачу и прием сообщений в стандартах факсимильной связи.

– Сервер защиты данных, оснащенный широким набором средств обеспечения безопасности данных и, в первую очередь, идентификации паролей.

4.5. Сетевые операционные системы

Сетевые операционные системы (Network Operating System – NOS) – это комплекс программ, обеспечивающих в сети обработку, хранение и передачу данных.

Для организации сети кроме аппаратных средств необходима также сетевая операционная система. Операционные системы сами по себе не могут поддерживать сеть. Для дополнения какой-нибудь ОС сетевыми средствами необходима процедура инсталляции сети.

Сетевая операционная система необходима для управления потоками сообщений между рабочими станциями и файловым сервером. Она является прикладной платформой, предоставляет разнообразные виды сетевых служб и поддерживает работу прикладных процессов, реализуемых в сетях. NOS используют архитектуру «клиент – сервер» или одноранговую архитектуру.

NOS определяет группу протоколов, обеспечивающих основные функции сети. К ним относятся:

– адресация объектов сети;

– функционирование сетевых служб;
– обеспечение безопасности данных;
– управление сетью.

4.6. Сетевое программное обеспечение

Для сетей программа-клиент обеспечивает связь с другими компьютерами и серверами, а также доступ к файлам и принтерам.

Сетевая карта является устройством, физически соединяющим компьютер с сетью. Для каждой сетевой карты устанавливаются свои драйверы, значение IRQ (требования к прерыванию) и адреса ввода/вывода.

Протоколы используются для установления правил обмена информацией в сетях.

Служба удаленного доступа позволяет делать файлы и принтеры доступными для компьютеров в сети.

Применение многопользовательских версий прикладных программ резко увеличивает производительность. Многие системы управления базами данных позволяют нескольким рабочим станциям работать с общей базой данных. Большинство деловых прикладных программ также являются многопользовательскими.

4.7. Защита данных

Защита данных от несанкционированного доступа при работе в ЛВС необходима для обеспечения:

– гарантии от разрушений. При работе в сети неопытных пользователей возможно уничтожение файлов и каталогов;

– защиты конфиденциальности;

– защиты от мошенничества. Некоторые расчетные ведомости несут в себе большие денежные суммы, и бывает, пользователи поддаются искушению выписать чек на свое имя;

– защиты от преднамеренных разрушений. В некоторых случаях раздраженный работник может разрушить какую-нибудь информацию.

4.8. Использование паролей. Ограничение доступа

Первый шаг к безопасности – это введение пароля. Каждому пользователю ЛВС присваивается пароль – секретное слово, известное только этому пользователю. При вводе пароля высвечиваются звездочки. Сетевая операционная система хранит информацию по всем именам и паролям (в закодированной форме), а также о правах доступа к директориям и другие атрибуты пользователей.

Еще одна возможность защиты данных заключается в ограничении доступа к определенным директориям или определенным серверам. Доступ к дискам рабочих станций выбирается посредством вкладки «Управление доступом» в программе «Сетевое окружение». Взаимодействие между серверами организуется посредством установки доверительных отношений между серверами.

4.9. Типовой состав оборудования локальной сети

Фрагмент вычислительной сети включает основные типы коммуникационного оборудования, применяемого сегодня для образования локальных сетей и соединения их через глобальные связи друг с другом.

Для построения локальных связей между компьютерами используются различные виды кабельных систем, сетевые адаптеры, концентраторы, повторители. Для связей между сегментами локальной вычислительной сети используются концентраторы, мосты, коммутаторы, маршрутизаторы и шлюзы.

Подключение локальных сетей к глобальным связям осуществляется с помощью:

– специальных выходов (WAN-порты) мостов и маршрутизаторов;

– аппаратуры передачи данных по длинным линиям – модемы (при передачи по аналоговым линиям);

– устройства подключения к цифровым каналам (ТА – терминальные адаптеры сетей ISDN, устройства обслуживания цифровых выделенных каналов типа CSU/DSU и т. п.).

5. ФИЗИЧЕСКАЯ СРЕДА ПЕРЕДАЧИ ДАННЫХ

Физическая среда является основой, на которой строятся физические средства соединения. Сопряжение с физическими средствами соединения посредством физической среды обеспечивает **физический уровень.** В качестве физической среды широко используются эфир, металлы, оптическое стекло и кварц. На физическом уровне находится носитель, по которому передаются данные. Среда передачи данных может включать как кабельные, так и беспроводные технологии. Хотя физические кабели являются наиболее распространенными носителями данных для сетевых коммуникаций, беспроводные технологии все более внедряются благодаря их способности связывать глобальные сети.

На физическом уровне для физических кабелей определяются механические и электрические (оптические) свойства среды передачи, которые включают:

– тип кабелей и разъемов;

– разводку контактов в разъемах;

– схему кодирования сигналов для значений 0 и 1.

Канальный уровень определяет доступ к среде и управление передачей посредством процедуры передачи данных по каналу. В локальных сетях протоколы канального уровня используются компьютерами, мостами, коммутаторами и маршрутизаторами. В компьютерах функции канального уровня реализуются совместными усилиями сетевых адаптеров и их драйверов.

5.1. Кабель связи, линия связи, канал связи

Для организации связи в сетях используются следующие понятия:

– кабель связи;

– линия связи;

– канал связи.

Кабель связи – это длинномерное изделие электротехнической промышленности. Из кабелей связи и других элементов (монтаж, крепеж, кожухи и т. д.) строят **линии связи**. Прокладка линии внутри здания – задача достаточно серьезная. Длина линий связи колеблется от десятков метров до десятков тысяч километров. В любую более или менее серьезную линию связи кроме кабелей входят: траншеи, колодцы, муфты, переходы через реки, море и океаны, а также грозозащита (равно как и другие виды защиты) линий. Очень сложны охрана, эксплуатация, ремонт линий связи; содержание кабелей связи под избыточным давлением, профилактика (в снег, дождь, на ветру, в траншее и в колодце, в реке и на дне моря). Большую сложность представляют собой юридические вопросы, включающие согласование прокладки линий связи, особенно в городе. Вот чем линия (связи) отличается от кабеля.

По уже построенным линиям организуют **каналы связи**. Причем если линию, как правило, строят и сдают сразу всю, то каналы связи вводят постепенно. Уже по линии можно установить связь, но такое использование крайне дорогостоящих сооружений очень неэффективно. Поэтому применяют аппаратуру каналообразования (или, как раньше говорили, уплотнение линии). По каждой электрической цепи, состоящей из двух проводов, обеспечивают связь не одной паре абонентов (или компьютеров), а сотням или тысячам: по одной коаксиальной паре в междугородном кабеле может быть образовано до 10 800 каналов тональной частоты (0,3–3,4 КГц) или почти столько же цифровых, с пропускной способностью 64 Кбит/с.

При наличии кабелей связи создаются линии связи, а уже по линиям связи создаются каналы связи. Линии связи и каналы связи подключаются к узлам связи. Линии, каналы и узлы образуют первичные сети связи.

5.2. Типы кабелей и структурированные кабельные системы

В качестве среды передачи данных используются различные виды кабелей: коаксиальный кабель, кабель на основе экранированной и неэкранированной витой пары и оптоволоконный кабель. Наиболее популярным видом среды передачи данных на небольшие расстояния (до 100 м) становится неэкранированная витая пара, которая включена практически во все современные стандарты и технологии локальных сетей и обеспечивает пропускную способность до 100 Мбайт/с (на кабелях категории 5). Оптоволоконный кабель широко применяется как для построения локальных связей, так и для образования магистралей глобальных сетей. Оптоволоконный кабель может обеспечить очень высокую пропускную способность канала (до нескольких Гбайт в секунду) и передачу на значительные расстояния (до нескольких десятков километров без промежуточного усиления сигнала).

Для передачи данных в вычислительных сетях используются электромагнитные волны различных частот – КВ, УКВ, СВЧ. Однако пока в локальных сетях радиосвязь используется только в тех случаях, когда оказывается невозможной прокладка кабеля, например, в зданиях. Это объясняется недостаточной надежностью сетевых технологий, построенных на использовании электромагнитного излучения. Для построения глобальных каналов этот вид среды передачи данных используется шире: при построении спутниковых каналов связи и наземных радиорелейных каналов, работающих в зонах прямой видимости в СВЧ диапазонах.

Очень важно правильно построить фундамент сети – кабельную систему. В последнее время в качестве такой надежной основы все чаще используется структурированная кабельная система.

Структурированная кабельная система (Structured Cabling System – SCS) – это набор коммутационных элементов (кабелей, разъемов, коннекторов, кроссовых панелей и шкафов), а также методика их совместного использования, которая позволяет создавать регулярные, легко расширяемые структуры связей в вычислительных сетях.

Преимущества структурированной кабельной системы:

– Универсальность. Структурированная кабельная система при продуманной организации может стать единой средой для передачи компьютерных данных в локальной вычислительной сети.

– Увеличение срока службы. Срок старения хорошо структурированной кабельной системы может составлять 8–10 лет.

– Уменьшение стоимости добавления новых пользователей и изменения их мест размещения. Стоимость кабельной системы в основном определяется не стоимостью кабеля, а стоимостью работ по его прокладке.

– Возможность легкого расширения сети. Структурированная кабельная система является модульной, поэтому ее легко наращивать, позволяя легко и ценой малых затрат переходить на более совершенное оборудование, удовлетворяющее растущим требованиям к системам коммуникаций.

– Обеспечение более эффективного обслуживания. Структурированная кабельная система облегчает обслуживание и поиск неисправностей.

– Надежность. Структурированная кабельная система имеет повышенную надежность, поскольку обычно производство всех ее компонентов и техническое сопровождение осуществляется одной фирмой-производителем.

5.3. Кабельные системы

Выделяют два больших класса кабелей: электрические и оптические, которые принципиально различаются по способу передачи по ним сигнала.

Особенность оптоволоконных систем – высокая стоимость как самого кабеля (по сравнению с медным), так и специализированных установочных элементов (розеток, разъемов, соединителей и т. п.). Правда, главный вклад в стоимость сети вносит цена активного сетевого оборудования для оптоволоконных сетей.

Оптоволоконные сети применяются для горизонтальных высокоскоростных каналов, а также все чаще стали применяться для вертикальных каналов связи (межэтажных соединений).

Оптоволоконные кабели в будущем смогут составить реальную конкуренцию медным высокочастотным, поскольку стоимость производства медных кабелей снижаться не будет, ведь для него нужна очень чистая медь, запасов которой на земле гораздо меньше, чем кварцевого песка, из которого производят оптоволокно.

Основные поставщики оптоволоконного кабеля – Mohawk/CDT, Lucent Technologies и AMP.

5.4. Типы кабелей

Существует несколько различных типов кабелей, используемых в современных сетях. Множество разновидностей медных кабелей составляют класс электрических кабелей, используемых как для прокладки телефонных сетей, так и для инсталляции ЛВС. По внутреннему строению различают кабели на витой паре и коаксиальные кабели.

Кабель типа «витая пара» (Twisted Pair). Витой парой называется кабель, в котором изолированная пара проводников скручена с небольшим числом витков на единицу длины. Скручивание проводов уменьшает электрические помехи извне при распространении сигналов по кабелю, а экранированные витые пары еще более увеличивают степень помехозащищенности сигналов.

Кабель типа «витая пара» используется во многих сетевых технологиях, включая Ethernet, ARCNet и IBM Token Ring.

Кабели на витой паре подразделяются: на неэкранированные (Unshielded Twisted Pair – UTP) и экранированные медные кабели. Последние подразделяются на две разновидности: с экранированием каж-

дой пары и общим экраном (Shielded Twisted Pair – STP) и только с одним общим экраном (Foiled Twisted Pair – FTP). Наличие или отсутствие экрана у кабеля вовсе не означает наличия или отсутствия защиты передаваемых данных, а говорит лишь о различных подходах к подавлению помех. Отсутствие экрана делает неэкранированные кабели более гибкими и устойчивыми к изломам. Кроме того, они не требуют дорогостоящего контура заземления для эксплуатации в нормальном режиме, в отличие от экранированных кабелей. Неэкранированные кабели идеально подходят для прокладки в помещениях внутри офисов, а экранированные лучше использовать для установки в местах с особыми условиями эксплуатации, например, рядом с очень сильными источниками электромагнитных излучений, которых в офисах обычно нет. Кабели классифицируются по категориям, указанным в табл. 1. Основанием для отнесения кабеля к одной из категорий служит максимальная частота передаваемого по нему сигнала.

Таблица 1

Классификация кабелей типа «витая пара»

Категория	Частота передаваемого сигнала, МГц
3	16
4	20
5	100
5+	300
6	200
7	600

Коаксиальные кабели. Коаксиальные кабели используются в радио- и телеаппаратуре. Коаксиальные кабели могут передавать данные со скоростью 10 Мбит/с на максимальное расстояние от 185 до 500 м. В зависимости от толщины они разделяются на толстые и тонкие. Типы коаксиальных кабелей приведены в табл. 2.

Таблица 2

Типы коаксильных кабелей

Тип	Название, значение сопротивления
RG-8 и RG-11	Thicknet, 50 Ом
RG-58/U	Thinnet, 50 Ом, сплошной центральный медный проводник
RG-58 A/U	Thinnet, 50 Ом, центральный многожильный проводник
RG-59	Broadband/Cable television (широковещательное и кабельное телевидение), 75 Ом
RG-59/U	Broadband/Cable television (широковещательное и кабельное телевидение), 50 Ом
RG-62	ARCNet, 93 Ом

Кабель Thinnet, известный как кабель RG-58, является наиболее широко используемым физическим носителем данных. Сети при этом не требуют дополнительного оборудования и являются простыми и недорогими. Тонкий коаксиальный кабель (Thin Ethernet) не позволяет передавать информацию на расстояние большее, чем толстый кабель. Для соединений с тонким кабелем применяются стандартные байонетные разъемы BNC типа CP-50 и ввиду его небольшой стоимости он становится фактически стандартным для офисных ЛВС. Используется в технологии Ethernet 10Base-2, описанной ниже.

Толстый коаксиальный кабель (Thick Ethernet) имеет большую степень помехозащищенности, большую механическую прочность, но требует специального приспособления для прокалывания кабеля, чтобы создать ответвления для подключения к ЛВС. Он более дорогой и менее гибкий, чем тонкий. Используется в технологии Ethernet 10Base-5. Сети ARCNet с посылкой маркера обычно используют кабель RG-62 A/U.

5.5. Оптоволоконный кабель

Оптоволоконный кабель (Fiber Optic Cable) обеспечивает высокую скорость передачи данных на большое расстояние. Он также невосприимчив к интерференции и подслушиванию. В оптоволоконном кабеле для передачи сигналов используется свет. Благодаря волокну, применяемому в качестве световода, возможна передача сигналов на большие расстояния с огромной скоростью, но оно дорогое, и с ним трудно работать.

Для установки разъемов, создания ответвлений, поиска неисправностей в оптоволоконном кабеле необходимы специальные приспособления и высоко квалифицированный персонал. Оптоволоконный кабель состоит из центральной стеклянной нити толщиной в несколько микрон, покрытой сплошной стеклянной оболочкой. Все это, в свою очередь, спрятано во внешнюю защитную оболочку.

Оптоволоконные линии очень чувствительны к плохим соединениям в разъемах. В качестве источника света в таких кабелях применяются светодиоды (Light Emitting Diode – LED), а информация кодируется путем изменения интенсивности света. На приемном конце кабеля детектор преобразует световые импульсы в электрические сигналы.

Существует два типа оптоволоконных кабелей – одномодовые и многомодовые. Одномодовые кабели имеют меньший диаметр, боль-

шую стоимость и позволяют осуществлять передачу информации на большие расстояния. Поскольку световые импульсы могут двигаться в одном направлении, системы на базе оптоволоконных кабелей должны иметь входящий кабель и исходящий кабель для каждого сегмента. Оптоволоконный кабель требует специальных коннекторов и высококвалифицированной установки.

5.6. Кабельные системы Ethernet

10Base-T, 100Base-TX. Неэкранированная витая пара (Unshielded Twisted Pair – UTP) – это кабель из скрученных пар проводов.

Характеристики:

– диаметр проводников 0.4–0.6 мм, 4 скрученные пары (8 проводников, из которых для 10Base-T и 100Base-TX используются только 4). Кабель должен иметь категорию 3 или 5 и качество data grade или выше;

– максимальная длина сегмента 100 м;

– разъемы восьмиконтактные RJ-45.

10Base-2. Тонкий коаксиальный кабель.

Характеристики:

– диаметр 0.2 дюйма, RJ-58A/U, 50 Ом;

– приемлемые разъемы – BNC;

– максимальная длина сегмента – 185 м;

– минимальное расстояние между узлами – 0,5 м;

– максимальное число узлов в сегменте – 30.

10Base-5. Толстый коаксиальный кабель.

Характеристики:

– волновое сопротивление – 50 Ом;

– максимальная длина сегмента – 500 м;

– минимальное расстояние между узлами – 2.5 м;

– максимальное число узлов в сегменте – 100.

5.7. Беспроводные технологии

Беспроводные технологии передачи данных являются удобным, а иногда незаменимым средством связи. Они различаются по типам сигнала, частоте (большая частота означает большую скорость передачи) и расстоянию передачи. Важное значение имеют помехи и стоимость. Можно выделить три основных типа беспроводной технологии:

– радиосвязь;

– связь в микроволновом диапазоне;

– инфракрасная связь.

Радиосвязь. Технологии радиосвязи (Radio Waves) пересылают данные на радиочастотах и практически не имеют ограничений по дальности. Она используется для соединения локальных сетей, расположенных друг от друга на больших географических расстояниях. Радиопередача в целом имеет высокую стоимость и чувствительна к электронному и атмосферному наложению, а также подвержена перехватам, поэтому требует шифрования для обеспечения уровня безопасности.

Связь в микроволновом диапазоне. Передача данных в микроволновом диапазоне (Microwaves) основана на использовании высоких частот и осуществляется как на короткие, так и на большие расстояния. Главное ограничение заключается в том, чтобы передатчик и приемник были в зоне прямой видимости. Используется в местах, где применение физического носителя затруднено. Передача данных в микроволновом диапазоне с помощью спутников может быть очень дорогой.

Инфракрасная связь. Инфракрасные технологии (Infrared Transmission) функционируют на очень высоких частотах, приближающихся к частотам видимого света. Они могут быть использованы для установления двусторонней или широковещательной передачи на близкие расстояния. При инфракрасной связи обычно применяют светодиоды (LED) для передачи инфракрасных волн приемнику. Инфракрасная передача ограничена малым расстоянием в прямой зоне видимости и может быть использована в офисных зданиях.

6. СЕТЕВЫЕ ОПЕРАЦИОННЫЕ СИСТЕМЫ

Сетевые операционные системы (Network Operating System – NOS) – это комплекс программ, обеспечивающих обработку, хранение и передачу данных в сети.

Сетевая операционная система выполняет функции прикладной платформы, предоставляет разнообразные виды сетевых служб и поддерживает работу прикладных процессов, выполняемых в абонентских системах. Сетевые операционные системы используют клиентско-серверную либо одноранговую архитектуру. Компоненты NOS располагаются на всех рабочих станциях, включенных в сеть.

NOS определяет взаимосвязанную группу протоколов верхних уровней, обеспечивающих выполнение основных функций сети. К ним, в первую очередь, относятся:

– адресация объектов сети;

– функционирование сетевых служб;

– обеспечение безопасности данных;

– управление сетью.

При выборе NOS необходимо рассматривать множество факторов. Среди них:

– набор сетевых служб, которые предоставляет сеть;

– возможность наращивания имен, определяющих хранимые данные и прикладные программы;

– механизм рассредоточения ресурсов по сети;

– способ модификации сети и сетевых служб;

– надежность функционирования и быстродействие сети;

– используемые или выбираемые физические средства соединения;

– типы компьютеров, объединяемых в сеть, их операционные системы;

– предлагаемые системы, обеспечивающие управление сетью;

– используемые средства защиты данных;

– совместимость с уже созданными прикладными процессами;

– число серверов, которое может работать в сети;

– перечень ретрансляционных систем, обеспечивающих сопряжение локальных сетей с различными территориальными сетями;

– способ документирования работы сети, организация подсказок и поддержек.

6.1. Структура сетевой операционной системы

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

В соответствии со структурой в сетевой операционной системе отдельной машины можно выделить несколько частей.

1. Средства управления локальными ресурсами компьютера: функции распределения оперативной памяти между процессами, планирования и диспетчеризации процессов, управления процессорами, управления периферийными устройствами и другие функции управления ресурсами локальных ОС.

2. Средства предоставления собственных ресурсов и услуг в общее пользование – серверная часть ОС (сервер). Эти средства обеспечивают, например, блокировку файлов и записей, ведение справочников имен сетевых ресурсов, обработку запросов удаленного доступа к собственной файловой системе и базе данных, управление очередями запросов удаленных пользователей к своим периферийным устройствам.

3. Средства запроса доступа к удаленным ресурсам и услугам – клиентская часть ОС (редиректор). Эта часть выполняет распознавание и перенаправление в сеть запросов к удаленным ресурсам от приложений и пользователей. Клиентская часть также осуществляет прием ответов от серверов и преобразование их в локальный формат, так что для приложения выполнение локальных и удаленных запросов неразличимо.

4. Коммуникационные средства ОС, с помощью которых происходит обмен сообщениями в сети. Эта часть обеспечивает адресацию и буферизацию сообщений, выбор маршрута передачи сообщения по сети, надежность передачи и т. п., иными словами, является средством транспортировки сообщений.

6.2. Клиентское программное обеспечение

Для работы с сетью на клиентских рабочих станциях должно быть установлено клиентское программное обеспечение. Это программное

обеспечение делает возможным доступ к ресурсам, расположенным на сетевом сервере. Тремя наиболее важными компонентами клиентского программного обеспечения являются редиректоры, распределители и имена UNC (UNC pathnames). [5]

Редиректоры. Редиректор (Redirector) – сетевое программное обеспечение, которое принимает запросы ввода/вывода для удаленных файлов, именованных каналов или почтовых слотов и затем переназначает их сетевым сервисам другого компьютера. Редиректор перехватывает все запросы, поступающие от приложений, и анализирует их.

Фактически существует два типа редиректоров, используемых в сети:

– клиентский (client);

– серверный (server).

Оба редиректора функционируют на представительском уровне модели OSI. Когда клиент делает запрос к сетевому приложению или службе, редиректор перехватывает этот запрос и проверяет, является ли ресурс локальным (находящимся на запрашивающем компьютере) или удаленным (в сети). Если редиректор определяет, что это локальный запрос, он направляет запрос центральному процессору для немедленной обработки. Если запрос предназначен для сети, редиректор направляет запрос по сети к соответствующему серверу. По существу, редиректоры скрывают от пользователя сложность доступа к сети. После того как сетевой ресурс определен, пользователи могут получить к нему доступ без знания его точного расположения.

Распределители. Распределитель (Designator) представляет собой часть программного обеспечения, управляющую присвоением букв накопителя (drive letter) как локальным, так и удаленным сетевым ресурсам или разделяемым дисководам, что помогает во взаимодействии с сетевыми ресурсами. Когда между сетевым ресурсом и буквой локального накопителя создана ассоциация, известная также как отображение дисковода (mapping a drive), распределитель отслеживает присвоение такой буквы дисковода сетевому ресурсу. Затем, когда пользователь или приложение получат доступ к диску, распределитель заменит букву дисковода на сетевой адрес ресурса прежде, чем запрос будет послан редиректору.

Серверное программное обеспечение. Для того чтобы компьютер мог выступать в роли сетевого сервера, необходимо установить серверную часть сетевой операционной системы, которая позволит поддерживать ресурсы и распространять их среди сетевых клиентов. Главным требованием, предъявляемым к сетевым серверам, является возможность

ограничения доступа к сетевым ресурсам. Это называется сетевой защитой (Network Security). Она предоставляет средства управления доступом пользователей к ресурсам, степенью этого доступа, а также количеством пользователей, которые смогут получить такой доступ одновременно. Этот контроль обеспечивает конфиденциальность и защиту и поддерживает эффективную сетевую среду.

В дополнение к обеспечению контроля над сетевыми ресурсами сервер выполняет следующие функции:

– предоставляет проверку регистрационных имен (logon identification) для пользователей;

– управляет пользователями и группами;

– хранит инструменты сетевого администрирования для управления, контроля и аудита;

– обеспечивает отказоустойчивость для защиты целостности сети.

6.3. Клиентское и серверное программное обеспечение

Некоторые из сетевых операционных систем, в том числе Windows NT, имеют программные компоненты, обеспечивающие компьютеру как клиентские, так и серверные возможности. Это позволяет компьютерам поддерживать и использовать сетевые ресурсы; преобладает в одноранговых сетях. В общем, этот тип сетевых операционных систем не так мощен и надежен, как законченные сетевые операционные системы.

Главное преимущество комбинированной клиентско-серверной сетевой операционной системы заключается в том, что важные ресурсы, расположенные на отдельной рабочей станции, могут быть разделены с остальной частью сети. Недостаток состоит в том, что если рабочая станция поддерживает много активно используемых ресурсов, она испытывает серьезное падение производительности. Если такое происходит, то необходимо перенести эти ресурсы на сервер для увеличения общей производительности.

В зависимости от функций, возлагаемых на конкретный компьютер, в его операционной системе может отсутствовать либо клиентская, либо серверная часть.

Если сделан запрос к ресурсу данного компьютера, то он переадресовывается локальной операционной системе. Если же это запрос к удаленному ресурсу, то он переправляется в клиентскую часть, где преобразуется из локальной формы в сетевой формат, и передается коммуникационным средствам. Серверная часть ОС компьютера принимает зап-

рос, преобразует его в локальную форму и передает для выполнения своей локальной ОС. После того как результат получен, сервер обращается к транспортной подсистеме и направляет ответ клиенту, выдавшему запрос. Клиентская часть преобразует результат в соответствующий формат и адресует его тому приложению, которое выдало запрос.

6.4. Выбор сетевой операционной системы

При выборе сетевой операционной системы необходимо учитывать:
– совместимость оборудования;
– тип сетевого носителя;
– размер сети;
– сетевую топологию;
– требования, предъявляемые к серверу;
– операционные системы на клиентах и серверах;
– сетевую файловую систему;
– соглашения об именах в сети;
– организацию сетевых устройств хранения.

6.5. Требования, предъявляемые к сетям

При организации и эксплуатации сети важными требованиями при работе являются следующие:
– производительность;
– надежность и безопасность;
– расширяемость и масштабируемость;
– прозрачность;
– поддержка разных видов трафика;
– управляемость;
– совместимость.

6.5.1. Производительность

Производительность – это характеристика сети, позволяющая оценить, насколько быстро информация передающей рабочей станции достигнет приемной рабочей станции.

На производительность сети влияют следующие ее характеристики:
– конфигурация;

– скорость передачи данных;
– метод доступа к каналу;
– топология сети;
– технология.

Если производительность сети перестает отвечать предъявляемым к ней требованиям, то администратор сети может прибегнуть к различным приемам:

– изменить конфигурацию сети таким образом, чтобы структура сети более соответствовала структуре информационных потоков;

– перейти к другой модели построения распределенных приложений, которая позволила бы уменьшить сетевой трафик;

– заменить мосты более скоростными коммутаторами.

Но самым радикальным решением в такой ситуации является переход на более скоростную технологию. Если в сети используются традиционные технологии Ethernet или Token Ring, то переход на Fast Ethernet, FDDI или 100VG-AnyLAN позволит сразу в 10 раз увеличить пропускную способность каналов.

С ростом масштаба сетей возникла необходимость в повышении их производительности. Одним из способов достижения этого стала их микросегментация. Она позволяет уменьшить число пользователей на один сегмент и снизить объем широковещательного трафика, а значит, повысить производительность сети.

Первоначально для микросегментации использовались маршрутизаторы, которые, вообще говоря, не очень приспособлены для этой цели. Решения на их основе были достаточно дорогостоящими и отличались большой временной задержкой и невысокой пропускной способностью. Более подходящими устройствами для микросегментации сетей стали коммутаторы. Благодаря относительно низкой стоимости, высокой производительности и простоте в использовании они быстро завоевали популярность.

Таким образом, сети стали строить на базе коммутаторов и маршрутизаторов. Первые обеспечивали высокоскоростную пересылку трафика между сегментами, входящими в одну подсеть, а вторые передавали данные между подсетями, ограничивали распространение широковещательного трафика, решали задачи безопасности и т. д.

Виртуальные ЛВС (VLAN) обеспечивают возможность создания логических групп пользователей в масштабе корпоративной сети. Виртуальные сети позволяют организовать работу в сети более эффективно.

6.5.2. Надежность и безопасность

Важнейшей характеристикой вычислительных сетей является **надежность**. Повышение надежности основано на принципе предотвращения неисправностей путем снижения интенсивности отказов и сбоев за счет применения электронных схем и компонентов с высокой и сверхвысокой степенью интеграции, снижения уровня помех, облегченных режимов работы схем, обеспечения тепловых режимов их работы, а также за счет совершенствования методов сборки аппаратуры.

Отказоустойчивость – это такое свойство вычислительной системы, которое обеспечивает ей как логической машине возможность продолжения действий, заданных программой, после возникновения неисправностей. Введение отказоустойчивости требует наличия дополнительного аппаратного и программного обеспечения. Направления, связанные с предотвращением неисправностей и отказоустойчивостью, – основные в проблеме надежности. В параллельных вычислительных системах достигается как наиболее высокая производительность, так и, во многих случаях, очень высокая надежность. Имеющиеся в избытке ресурсы в параллельных системах могут использоваться как для повышения производительности, так и для повышения надежности.

Главной целью повышения надежности систем является обеспечение целостности хранимых в них данных.

Безопасность – одна из основных задач, решаемых любой нормальной компьютерной сетью. Под безопасностью может пониматься: защита от злонамеренной порчи данных, несанкционированного доступа к конфиденциальности информации, хищения и т. п.

Обеспечить защиту информации в условиях локальной сети всегда легче, чем при наличии на фирме десятка автономно работающих компьютеров. Практически в вашем распоряжении один инструмент – резервное копирование (backup). Для простоты назовем этот процесс резервированием. Суть его состоит в создании в безопасном месте полной копии данных, обновляемой как можно чаще. Для персонального компьютера более или менее безопасным носителем данных служат дискеты. Возможно использование стримера, но это уже дополнительные затраты на аппаратуру.

Легче всего обеспечить защиту данных от самых разных неприятностей в случае сети с выделенным файловым сервером. На сервере сосредоточены все наиболее важные файлы, а уберечь одну машину куда

проще, чем десять. Концентрированность данных облегчает и резервирование, так как не требуется собирать их по всей сети.

Экранированные линии позволяют повысить безопасность и надежность сети. Экранированные системы гораздо более устойчивы к внешним радиочастотным полям.

6.5.3. Прозрачность

Прозрачность – это такое состояние сети, когда пользователь, работая в сети, не видит ее.

Коммуникационная сеть является прозрачной относительно проходящей сквозь нее информации, если выходной поток битов в точности повторяет входной поток. Но сеть может быть непрозрачной во времени, если из-за меняющихся размеров очередей блоков данных изменяется и время прохождения различных блоков через узлы коммутации. Прозрачность сети по скорости передачи данных указывает, что данные можно передавать с любой скоростью.

Если в сети по одним и тем же маршрутам передаются информационные и управляющие (синхронизирующие) сигналы, то говорят, что сеть прозрачна по отношению к типам сигналов.

Если передаваемая информация может кодироваться любым способом, то это означает, что сеть прозрачна для любых методов кодировок.

Прозрачная сеть является простым решением, в котором для взаимодействия локальных сетей, расположенных на значительном расстоянии друг от друга, используется принцип Plug and Play (подключись и работай).

Служба прозрачных локальных сетей обеспечивает сквозное (end-to-end) соединение, связывающее между собой удаленные локальные сети. Привлекательность данного решения состоит в том, что эта служба объединяет удаленные друг от друга на значительное расстояние узлы как части локальной сети. Поэтому не нужно вкладывать средства в изучение новых технологий и создание территориально распределенных сетей (Wide Area Network – WAN). От пользователей требуется только поддерживать локальное соединение, а провайдер службы прозрачных сетей обеспечит беспрепятственное взаимодействие узлов через сеть масштаба города (Metropolitan Area Network – MAN) или сеть WAN.

Службы прозрачной локальной сети имеют много преимуществ. Например, пользователь может быстро и безопасно передавать боль-

шие объемы данных на значительные расстояния, не обременяя себя сложностями, связанными с работой в сетях WAN.

6.5.4. Поддержка разных видов трафика

Трафик в сети складывается случайным образом, однако в нем отражены и некоторые закономерности. Как правило, некоторые пользователи, работающие над общей задачей (например, сотрудники одного отдела), чаще всего обращаются с запросами либо друг к другу, либо к общему серверу, и только иногда они испытывают необходимость доступа к ресурсам компьютеров другого отдела. Желательно, чтобы структура сети соответствовала структуре информационных потоков. В зависимости от сетевого трафика компьютеры в сети могут быть разделены на группы (сегменты сети). Компьютеры объединяются в группу, если большая часть порождаемых ими сообщений адресована компьютерам этой же группы.

Для разделения сети на сегменты используются мосты и коммутаторы. Они экранируют локальный трафик внутри сегмента, не передавая за его пределы никаких кадров, кроме тех, которые адресованы компьютерам, находящимся в других сегментах. Таким образом, сеть распадается на отдельные подсети. Это позволяет более рационально выбирать пропускную способность имеющихся линий связи, учитывая интенсивность трафика внутри каждой группы, а также активность обмена данными между группами.

Однако локализация трафика средствами мостов и коммутаторов имеет существенные ограничения. С другой стороны, использование механизма виртуальных сегментов, реализованного в коммутаторах локальных сетей, приводит к полной локализации трафика; такие сегменты полностью изолированы друг от друга, даже в отношении широковещательных кадров. Поэтому в сетях, построенных только на мостах и коммутаторах, компьютеры, принадлежащие разным виртуальным сегментам, не образуют единой сети.

Для того чтобы эффективно консолидировать различные виды трафика в сети ATM, требуется специальная предварительная подготовка (адаптация) данных, имеющих различный характер: кадры – для цифровых данных; сигналы импульсно-кодовой модуляции – для голоса; потоки байтов – для видео. Эффективная консолидация трафика требует также учета и использования статистических вариаций интенсивности различных типов трафика.

6.5.5. Управляемость

ISO внесла большой вклад в стандартизацию сетей. Модель управления сетью является основным средством для понимания главных функций систем управления сетью. Эта модель состоит из пяти концептуальных областей:

– управление эффективностью;
– управление конфигурацией;
– управление учетом использования ресурсов;
– управление неисправностями;
– управление защитой данных.

Управление эффективностью. Цель этой области – измерение и обеспечение различных аспектов эффективности сети для того, чтобы межсетевая эффективность могла поддерживаться на приемлемом уровне. Примерами переменных эффективности, которые могли бы быть обеспечены, являются пропускная способность сети, время реакции пользователей и коэффициент использования линии.

Управление эффективностью включает несколько этапов:

1) сбор информации об эффективности по тем переменным, которые представляют интерес для администраторов сети;

2) анализ информации для определения нормальных (базовая строка) уровней;

3) определение соответствующих порогов эффективности для каждой важной переменной таким образом, что превышение этих порогов указывает на наличие в сети проблемы, достойной внимания.

Управление конфигурацией. Цель данной области – контролирование информации о сетевой и системной конфигурации для того, чтобы можно было отслеживать воздействие на работу сети различных версий аппаратных и программных элементов и управлять им. Так как все аппаратные и программные элементы имеют эксплуатационные отклонения, погрешности (или то и другое вместе), которые могут влиять на работу сети, такая информация важна для поддержания гладкой работы сети.

Каждое устройство сети располагает разнообразной информацией о версиях, ассоциируемых с ним. Чтобы обеспечить легкий доступ, подсистемы управления конфигурацией хранят эту информацию в базе данных. Когда возникает какая-нибудь проблема, в этой базе данных может быть проведен поиск ключей, которые могли бы помочь решить эту проблему.

Управление учетом использования ресурсов. Цель управления учетом использования ресурсов – измерить параметры использова-

ния сети так, чтобы можно было соответствующим образом регулировать ее использование индивидуальными или групповыми пользователями. Такое регулирование минимизирует число проблем в сети (поскольку ресурсы сети могут быть поделены исходя из возможностей источника) и максимизирует равнодоступность к сети для всех пользователей.

Управление неисправностями. Цель управления неисправностями – выявить, зафиксировать проблемы в сети, уведомить пользователей о них и (в пределах возможного) автоматически их устранить, с тем чтобы эффективно поддерживать работу сети. Так как неисправности могут привести к простоям или недопустимой деградации сети, управление неисправностями, по всей вероятности, является наиболее широко используемым элементом модели управления сети ISO.

Управление неисправностями включает в себя несколько шагов:

1) определение симптомов проблемы;

2) изолирование проблемы;

3) устранение проблемы;

4) проверка устранения неисправности во всех важных подсистемах;

5) регистрация обнаружения проблемы и ее решения.

Управление защитой данных. Цель управления защитой данных – контроль доступа к сетевым ресурсам в соответствии с местными руководящими принципами, для того чтобы сделать невозможными саботаж сети и доступ к чувствительной информации лицам, не имеющим соответствующего разрешения. Например, одна из подсистем управления защитой данных может контролировать регистрацию пользователей ресурса сети, отказывая в доступе тем, кто вводит коды доступа, не соответствующие установленным.

Подсистемы управления защитой данных работают путем разделения источников на санкционированные и несанкционированные области. Для некоторых пользователей доступ к любому источнику сети является несанкционированным.

Подсистемы управления защитой данных выполняют следующие функции:

– идентифицируют чувствительные ресурсы сети (включая системы, файлы и другие объекты);

– определяют отображения в виде карт между чувствительными источниками сети и набором пользователей;

– контролируют точки доступа к чувствительным ресурсам сети;

– регистрируют несанкционированный доступ к чувствительным ресурсам сети.

6.5.6. Совместимость программного обеспечения

В широких масштабах концепция программной совместимости впервые была применена разработчиками системы IBM/360. Основная задача при проектировании всего ряда моделей этой системы заключалась в создании такой архитектуры, которая с точки зрения пользователя была бы одинаковой для всех моделей системы независимо от цены и производительности каждой из них. Огромные преимущества такого подхода, позволяющего сохранять существующий задел программного обеспечения при переходе на новые (как правило, более производительные) модели, были быстро оценены как производителями компьютеров, так и пользователями, и с этого времени практически все фирмы-поставщики компьютерного оборудования взяли на вооружение эти принципы, поставляя серии совместимых компьютеров. Однако следует заметить однако, что со временем даже самая передовая архитектура неизбежно устаревает и возникает потребность во внесении радикальных изменений в архитектуру и способы организации вычислительных систем.

В настоящее время одним из наиболее важных факторов, определяющих современные тенденции в развитии информационных технологий, является ориентация компаний-поставщиков компьютерного оборудования на рынок прикладных программных средств.

Этот переход выдвинул ряд новых требований. Во-первых, такая вычислительная среда должна позволять изменять количество и состав аппаратных средств и программного обеспечения в соответствии с изменяющимися требованиями решаемых задач. Во-вторых, она должна обеспечивать возможность запуска одних и тех же программных систем на различных аппаратных платформах, т. е. обеспечивать мобильность программного обеспечения. В-третьих, эта среда должна гарантировать возможность применения одних и тех же человеко-машинных интерфейсов на всех компьютерах, входящих в неоднородную сеть. В условиях жесткой конкуренции производителей аппаратных платформ и программного обеспечения сформировалась концепция открытых систем, представляющая собой совокупность стандартов на различные компоненты вычислительной среды, предназначенных для обеспечения мобильности программных средств в рамках неоднородной, распределенной вычислительной системы.

7. АППАРАТНОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СЕТЕЙ

7.1. Планирование сети с хабом

При выборе места для установки концентратора необходимо принять во внимание следующие аспекты:

– местоположение;

– расстояние;

– питание.

Выбор места установки концентратора является наиболее важным этапом при планировании небольшой сети. Разумно хаб расположить вблизи геометрического центра сети (на одинаковом расстоянии от всех компьютеров). Такое расположение позволит минимизировать расход кабеля. Длина кабеля от концентратора до любого из подключаемых к сети компьютеров или периферийных устройств не должна превышать 100 м.

Концентратор можно поставить на стол или закрепить его на стене с помощью входящих в комплект хаба скоб. Установка хаба на стене позволяет упростить подключение кабелей, если они уже проложены в офисе.

При планировании сети есть возможность наращивания (каскадирования) хабов.

7.2. Преимущества концентратора

Концентраторы имеют много преимуществ. Прежде всего, в сети используется топология «звезда», при которой соединения с компьютерами образуют лучи, а хаб является центром звезды. Такая топология упрощает установку сети и управление ею. Любые перемещения компьютеров или добавление в сеть новых узлов при такой топологии выполнить совсем несложно. Кроме того, эта топология значительно надежнее, поскольку при любом повреждении кабельной системы сеть сохраняет работоспособность (перестает работать лишь поврежденный луч). Светодиодные индикаторы хаба позволяют контролировать состояние сети и легко обнаруживать неполадки.

Различные производители концентраторов реализуют в своих устройствах различные наборы вспомогательных функций, но наиболее часто встречаются следующие:

– объединение сегментов с различными физическими средами (например, коаксиал, витая пара и оптоволокно) в единый логический сегмент;

– автосегментация портов, т. е. автоматическое отключение порта при возникновении неисправностей (повреждение кабеля, интенсивная генерация пакетов ошибочной длины и т. п.);

– поддержка между концентраторами резервных связей, которые используются при отказе основных;

– защита передаваемых по сети данных от несанкционированного доступа (например, путем искажения поля данных в кадрах, повторяемых на портах, не имеющих компьютера с адресом назначения);

– поддержка средств управления сетями – протокола SNMP, баз управляющей информации MIB.

7.3. Мост

Мост (bridge) – ретрансляционная система, соединяющая каналы передачи данных.

В соответствии с базовой эталонной моделью взаимодействия открытых систем мост описывается протоколами физического и канального уровней, над которыми располагаются канальные процессы. Мост опирается на пару связываемых им физических средств соединения, которые в этой модели представляют физические каналы. Мост преобразует физический (1A, 1B) и канальный (2A, 2B) уровни различных типов. Что касается канального процесса, то он объединяет разнотипные каналы передачи данных в один общий.

Мост, а также его быстродействующий аналог – коммутатор (switching hub), делят общую среду передачи данных на логические сегменты. Логический сегмент образуется путем объединения нескольких физических сегментов (отрезков кабеля) с помощью одного или нескольких концентраторов. Каждый логический сегмент подключается к отдельному порту моста/коммутатора. При поступлении кадра на какой-либо из портов мост/коммутатор повторяет этот кадр, но не на всех портах, как это делает концентратор, а только на том порту, к которому подключен сегмент, содержащий компьютер-адресат.

Мосты могут соединять сегменты, использующие разные типы носителей, например 10Base-T (витая пара) и 10Base-2 (тонкий коакси-

альный кабель). Они могут соединять сети с разными методами доступа к каналу, например сети Ethernet (метод доступа CSMA/CD) и Token Ring (метод доступа TPMA).

7.4. Коммутатор

Коммутатор (switch) – устройство, осуществляющее выбор одного из возможных вариантов направления передачи данных.

В коммуникационной сети коммутатор является ретрансляционной системой (система, предназначенная для передачи данных или преобразования протоколов), обладающей свойством прозрачности (т. е. коммутация осуществляется здесь без какой-либо обработки данных). Коммутатор не имеет буферов и не может накапливать данные. Поэтому при использовании коммутатора скорости передачи сигналов в соединяемых каналах передачи данных должны быть одинаковыми. Канальные процессы, реализуемые коммутатором, выполняются специальными интегральными схемами. В отличие от других видов ретрансляционных систем, здесь, как правило, не используется программное обеспечение.

Вначале коммутаторы использовались лишь в территориальных сетях. Затем они появились и в локальных сетях, например, частные учрежденческие коммутаторы. Позже появились коммутируемые локальные сети. Их ядром стали коммутаторы локальных сетей.

Коммутатор может соединять серверы в кластер и служить основой для объединения нескольких рабочих групп. Он направляет пакеты данных в узлы ЛВС. Каждый коммутируемый сегмент получает доступ к каналу передачи данных без конкуренции и видит только тот трафик, который направляется в его сегмент. Коммутатор должен предоставлять каждому порту возможность соединения с максимальной скоростью без конкуренции со стороны других портов (в отличие от совместно используемого концентратора). Обычно в коммутаторах имеются один или два высокоскоростных порта, а также хорошие инструментальные средства управления. Коммутатором можно заменить маршрутизатор, дополнить им наращиваемый маршрутизатор или использовать его в качестве основы для соединения нескольких концентраторов. Коммутатор может служить отличным устройством для направления трафика между концентраторами ЛВС рабочей группы и загруженными файл-серверами.

7.5. Различие между мостом и коммутатором

Разница между мостом и коммутатором состоит в том, что мост в каждый момент времени может осуществлять передачу кадров только между одной парой портов, а коммутатор одновременно поддерживает потоки данных между всеми своими портами. Другими словами, мост передает кадры последовательно, а коммутатор параллельно.

Мосты используются только для связи локальных сетей с глобальными, т. е. как средства удаленного доступа, поскольку в этом случае необходимости в параллельной передаче между несколькими парами портов просто не возникает.

Когда появились первые устройства, позволяющие разъединять сеть на несколько доменов коллизий (по сути, фрагменты ЛВС, построенные на хабах), они были двухпортовыми и получили название мостов. По мере развития данного типа оборудования, они стали многопортовыми и получили название коммутаторов. Некоторое время оба понятия существовали одновременно, а позднее вместо термина «мост» стали применять «коммутатор». Далее в этой теме будет использоваться термин «коммутатор» для обозначения этих обеих разновидностей устройств, поскольку все сказанное ниже в равной степени относится и к мостам, и к коммутаторам. Следует отметить, что в последнее время локальные мосты полностью вытеснены коммутаторами.

Нередки случаи, когда необходимо соединить локальные сети, в которых различаются лишь протоколы физического и канального уровней. Протоколы остальных уровней в этих сетях приняты одинаковыми. Такие сети могут быть соединены мостом. Часто мосты наделяются дополнительными функциями. Такие мосты обладают определенным интеллектом (интеллектом в сетях называют действия, выполняемые устройствами) и фильтруют сквозь себя блоки данных, адресованные абонентским системам, расположенным в той же сети. Для этого в памяти каждого моста имеются адреса систем, включенных в каждую из сетей. Блоки, проходящие через интеллектуальный мост, дважды проверяются, на входе и выходе. Это позволяет предотвращать появление ошибок внутри моста.

Мосты не имеют механизмов управления потоками блоков данных. Поэтому может оказаться, что входной поток блоков будет большим, чем выходной. В этом случае мост не справится с обработкой входного потока, и его буферы могут переполняться. Чтобы этого не произошло, избыточные блоки выбрасываются. Специфические функции выполня-

ет мост в радиосети. Здесь он обеспечивает взаимодействие двух радиоканалов, работающих на разных частотах. Его именуют ретранслятором.

Мосты оперируют данными на высоком уровне и имеют совершенно определенное назначение. Прежде всего, они предназначены для соединения сетевых сегментов, имеющих различные физические среды, например для соединения сегмента с оптоволоконным кабелем и сегмента с коаксиальным кабелем. Мосты также могут быть использованы для связи сегментов, имеющих различные протоколы низкого уровня (физического и канального).

7.6. Коммутатор локальной сети

Коммутатор локальной сети (local area network switch) – устройство, обеспечивающее взаимодействие сегментов одной либо группы локальных сетей.

Коммутатор локальной сети, как и обычный коммутатор, обеспечивает взаимодействие подключенных к нему локальных сетей (рис. 10). Но в дополнение к этому, если соединяются различные типы сегментов локальной сети, он осуществляет преобразование интерфейсов. Чаще всего это сети Ethernet, кольцевые сети IBM, сети с оптоволоконным распределенным интерфейсом данных.

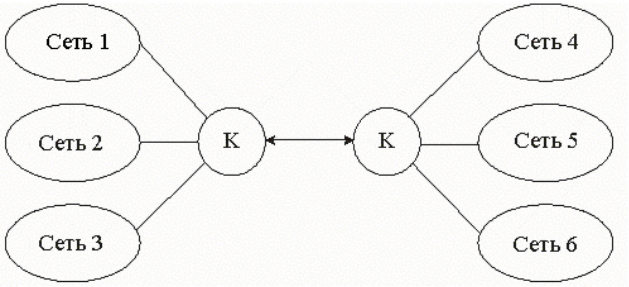


Рис. 10. Схема подключения локальных сетей к коммутаторам

В перечень функций, выполняемых коммутатором локальной сети, входят:

– обеспечение сквозной коммутации;

– наличие средств маршрутизации;

– поддержка простого протокола управления сетью;

– имитация моста либо маршрутизатора;

– организация виртуальных сетей;

– скоростная ретрансляция блоков данных.

7.7. Маршрутизатор

Маршрутизатор (router) – ретрансляционная система, соединяющая две коммуникационные сети либо их части.

Каждый маршрутизатор реализует протоколы физического (1А, 1В), канального (2А, 2В) и сетевого (3А, 3В) уровней. Специальные сетевые процессы соединяют части коммутатора в единое целое. Физический, канальный и сетевой протоколы в разных сетях различны. Поэтому соединение пар коммуникационных сетей осуществляется через маршрутизаторы, которые осуществляют необходимое преобразование указанных протоколов. Сетевые процессы выполняют взаимодействие соединяемых сетей.

Маршрутизатор работает с несколькими каналами, направляя в какой-нибудь из них очередной блок данных.

Маршрутизаторы обмениваются информацией об изменениях структуры сетей, трафике и их состоянии. Благодаря этому выбирается оптимальный маршрут следования блока данных в разных сетях от абонентской системы-отправителя к системе-получателю. Маршрутизаторы обеспечивают также соединение административно независимых коммуникационных сетей.

Архитектура маршрутизатора также используется при создании узла коммутации пакетов.

7.8. Различие между маршрутизатором и мостом

Маршрутизаторы превосходят мосты своей способностью фильтровать и направлять пакеты данных на сети. Так как маршрутизаторы работают на сетевом уровне, они могут соединять сети, использующие разную сетевую архитектуру, методы доступа к каналам связи и протоколы.

Маршрутизаторы не обладают такой способностью к анализу сообщений, как мосты, но зато могут принимать решение о выборе оптимального пути передачи данных между двумя сетевыми сегментами.

Мосты принимают решение по поводу адресации каждого из поступивших пакетов данных, переправлять его через мост или нет в зависимости от адреса назначения. Маршрутизаторы же выбирают из таблицы маршрутов наилучший для данного пакета.

В поле зрения маршрутизаторов находятся только пакеты, направленные к ним предыдущими маршрутизаторами, в то время как мосты должны обрабатывать все пакеты сообщений в сегменте сети, к которому они подключены.

Тип топологии или протокола уровня доступа к сети не имеет значения для маршрутизаторов, так как они работают на уровень выше, чем мосты (сетевой уровень модели OSI). Маршрутизаторы часто используются для связи между сегментами с одинаковыми протоколами высокого уровня. Наиболее распространенным транспортным протоколом, который используют маршрутизаторы, является IPX фирмы Novell или TCP фирмы Microsoft.

Необходимо запомнить, что для работы маршрутизаторов требуется один и тот же протокол во всех сегментах, с которыми он связан. При связывании сетей с различными протоколами лучше использовать мосты. Для управления загруженностью трафика сегмента сети также можно использовать мосты.

7.9. Шлюз

Шлюз (gateway) – ретрансляционная система, обеспечивающая взаимодействие информационных сетей.

Шлюз является наиболее сложной ретрансляционной системой, обеспечивающей взаимодействие сетей с различными наборами протоколов всех семи уровней. В свою очередь, наборы протоколов могут опираться на различные типы физических средств соединения.

В тех случаях, когда информационные сети соединяются, в них часть уровней может иметь одни и те же протоколы. Тогда сети соединяются не при помощи шлюза, а на основе более простых ретрансляционных систем, именуемых маршрутизаторами и мостами.

Шлюзы оперируют на верхних уровнях модели OSI (сеансовом, представительском и прикладном) и представляют наиболее развитый метод подсоединения сетевых сегментов и компьютерных сетей. Необходимость в сетевых шлюзах возникает при объединении двух систем, имеющих различную архитектуру. Например, шлюз приходится использовать для соединения сети с протоколом TCP/IP и большой ЭВМ со стандартом SNA. Эти две архитектуры не имеют ничего общего, и потому требуется переводить весь поток данных, проходящих между двумя системами.

В качестве шлюза обычно используется выделенный компьютер, на котором запущено программное обеспечение шлюза и производятся преобразования, позволяющие взаимодействовать нескольким системам в сети. Другой функцией шлюзов является преобразование протоколов. При получении сообщения IPX/SPX для клиента TCP/IP шлюз преобразует сообщения в протокол TCP/IP.

Шлюзы сложны в установке и настройке. Шлюзы работают медленнее, чем маршрутизаторы.

8. ПРОГРАММИРОВАНИЕ ИМИТАЦИОННЫХ МОДЕЛЕЙ ИНФОРМАЦИОННЫХ СЕТЕЙ В СРЕДЕ GPSS/PC

8.1. Непрерывно-стохастические модели систем массового обслуживания

В информационной сети поток заявок на обслуживание носит случайный характер. Поэтому информационные сети можно рассматривать как системы массового ослуживания (СМО). Графическая интерпретация моделей систем массового обслуживания осуществляется с помощью Q-схем. На входы Q-схемы подаются потоки заявок с различными законами распределения вероятности. Структурная схема элементарной ячейки СМО представляется Q-схемой (рис. 11).

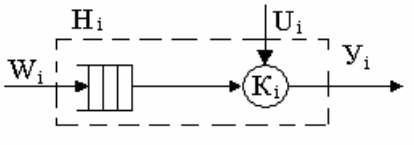
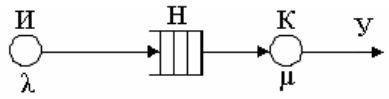


Рис. 11. Q-схема элементарной ячейки

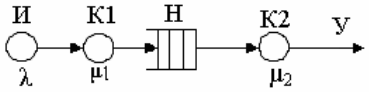
Обозначения, принимаемые в Q-схеме элементарной ячейки:
H_i – i-й накопитель заявок емкостью L1;
K_i – канал обслуживания заявок;
W_i – входной поток поступающих заявок;
U_i – поток обслуживаний для i-й заявки;
$У_i$ – выходной поток обслуженных заявок.
Существуют следующие варианты Q-схем систем массового обслуживания:

1. Схема с одним источником заявок, с одним накопителем и с одним каналом обслуживания:

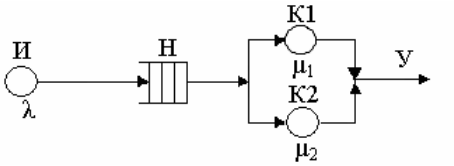


Принятые обозначения:
И – источник заявок;

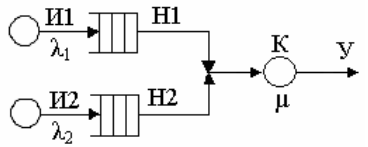
λ – интенсивность поступления заявок;
μ – интенсивность обслуживания заявок в канале.
2. Схема с одним источником заявок и с двумя каналами обслуживания:



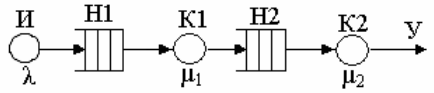
3. Схема с двумя каналами, включенными параллельно:



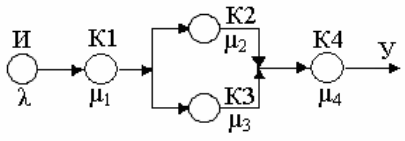
4. Схема с двумя источниками заявок:



5. Схема с последовательным включением накопителей и каналов обслуживания:



6. Схема СМО без накопителей заявок:



8.2. Описание языка моделирования GPSS

Основная программа (файл) – интерпретатор (GPSSPC.EXE). Основным динамическим объектом языка GPSS является транзакт. Моделирующая система GPSS (General Purpose Simulation System) основана на предположении, что моделью любой сложной системы является совокупность ее абстрактных элементов, называемых объектами, и логических правил их взаимодействий, называемых операциями.

Объекты языка GPSS разделяются на три класса:

– динамические;

– аппаратно-ориентированные;

– статические.

К динамическим объектам, представляющим собой поток обслуживаний, относятся транзакты. Они создаются, приводят в действия некоторые элементарные модели и уничтожаются.

В реальной системе транзактам могут соответствовать детали, работники, сообщения вычислительной сети и т. д.

Аппаратно-ориентированные объекты – это абстрактные элементы (устройства, накопители, ключи), на которые может быть разделено оборудование системы. Одновременно устройство может обслужить только один транзакт.

Ключи предназначены для переключения направления движения транзакта и имеют два состояния: включено и выключено.

К статическим объектам относятся очереди заявок и таблицы.

Логику функционирования модели задают операционные объекты или блоки.

Транзакты моделей входят в цепи. Существует пять типов цепей:

1. Цепь текущих событий. Включает транзакты, планируемое время поступления которых равно или меньше текущего модельного времени.

2. Цепь будущих событий. Включает транзакты, планируемое время поступления которых больше текущего модельного времени.

3. Цепь прерывания. Включает транзакты, обслуживание которых было прервано.

4. Цепь парных транзактов. Включает транзакты, которые ожидают появления синхронизирующих транзактов.

5. Цепь пользователя. Включает те транзакты, которые пользователь удалил из цепи текущих событий на некоторое время.

Как правило, цепь текущих событий организуется в порядке убывания приоритетов.

8.2.1. Форма представления моделей в языке GPSS

Модель системы в языке GPSS может быть описана в виде блок-схемы, либо в виде программы.

Блок-схема – это графическое представление программы в виде условных изображений блоков.

Программы модели представляются в виде операторов языка GPSS.

8.2.2. Транзакты

Транзакты – это динамические, движущиеся элементы моделей. В начале моделирования нет ни одного транзакта. В процессе моделирования транзакты (сообщения) входят в модель в моменты времени, определяемые логикой работы самой системы.

В общем случае в модели существует множество транзактов, но активным может быть только один. Перемещение транзакта по модели определяется блок-схемой. Транзакт движется до тех пор, пока не наступит одно из 3-х событий:

1. Транзакт входит в блок, функцией которого является задержка.

2. Транзакт входит в блок, функцией которого является удаление.

3. Транзакт пытается войти в блок, но тот его не может принять. При этом движение данного транзакта прекращается, и по модели может двигаться уже другой транзакт.

В модели на языке GPSS может быть несколько независимых программных сегментов. Активным будет тот сегмент, в котором есть движущийся транзакт.

Основным атрибутом транзакта являются его параметры, число которых достигает 1000. Наиболее важным атрибутом транзакта является уровень его приоритета.

Уровень приоритета транзакта обозначается целым числом от 0 до 127.

8.2.3. Таймер модельного времени

Интерпретатор GPSS автоматически генерирует и обслуживает таймер модельного времени. Переключение таймера от одного момента времени к другому происходит в зависимости от наступающих событий.

В начальный момент моделирования значение таймера равно 0. Следующим значением таймера будет время наступления первого события.

Дискретность изменения показаний таймера определяется частотой наиболее быстрых событий.

Пример функционирования некоторой рабочей станции, на вход которой поступают сообщения (рис. 12).

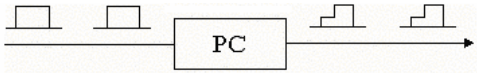


Рис. 12. Функциональная схема рабочей станции вычислительной сети

На выходе имеем обработанные сообщения. Сообщения поступают с интервалом 20 мин, а время обработки сообщения 15 мин. Статистика событий, формируемая GPSS, приведена в табл. 3.

Таблица 3

Статистика событий для рабочей станции

Последовательность возникновения событий	События	Фактическое время	Модельное время
1	Включение PC	8 ч 00 мин	0
2	1-я заявка PC	8 ч 20 мин	20
3	1-я заявка обработана	8 ч 35 мин	35
4	2-я заявка PC	8 ч 40 мин	40
5	2-я заявка обработана	8 ч 55 мин	55

Особенности таймера GPSS:

1. Таймер регистрирует только целое значение.

2. Единица времени выбирается разработчиком, и она должна быть единой для всех данных моделей.

3. GPSS ориентирован на последующие события, поэтому время прогона моделей не зависит от масштаба выбранной единицы времени, т. е. имеется возможность сжимать или растягивать время.

8.3. Основные сведения о блоках языка GPSS

Интерпретатор GPSS читает операторы, вызывает к исполнению определенные подпрограммы, формирует файл статистики и строит гистограммы событий.

На рис. 13 представлен алгоритм программы обслуживания транзакта в устройстве.

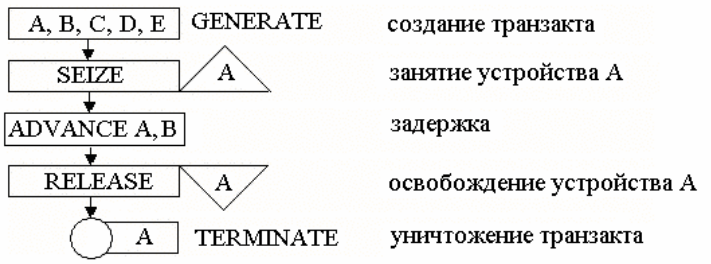


Рис. 13. Алгоритм программного сегмента

Для организации переходов к подпрограммам и ссылок на них блокам присваивают имена (метки).

Блоки содержат различные операнды (A, B, C, D, E, F и т. д.), которые используются при обращении к подпрограммам и для условий ветвления программы. Максимальное число операндов 7, но чаще всего 1 или 2.

Если некоторые операнды не заданы в явном виде, то интерпретатор принимает их значения по умолчанию. Информация в блоке распределена по трем полям:

1-е поле занимает вторую – шестую позицию. В это поле заносится имя блока (метка перехода);

2-е поле – поле операции, занимает с 8 по 18 позицию курсора. В нем содержится код блока в виде символического имени;

3-е поле – поле операндов. Может занимать с 19 по 71 позицию курсора. В него заносятся аргументы блока. Аргументы отделяются друг от друга запятыми. Если операнд в блоке отсутствует, то подряд надо поставить две запятые.

Формат написания блока:

1 2 3 4 5 6 7|8|9 10 11 12 13 14 15...19 20 21 22 23 24 25...

|G|ENERATE 20,5,3; блок Generate с аргументами A, B, D.

- - - - - - - - -

SB12 SEIZE O T O; блок с именем SB12 и операндом OTO.

8.3.1. Транзактно-ориентированные блоки GPSS

Блок GENERATE. Этим блоком производится ввод транзакта в модель (на рис. 14 представлено условно-графическое обозначение (УГО) блока).

Формат блока: GENERATE <A>,<B>,<C>,<D>,<E>.

Назначение параметров:

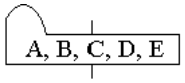


Рис. 14. УГО блока Generate

< A > – определяет среднее время между приходами транзактов;

< B > – задает половину поля допуска в интервалах поступления транзактов (доверительный интервал);

< C > – задает смещение во временной области (т. е. время появления первого транзакта). Если < C > отсутствует, то смещение не задано и первый транзакт войдет в модель в момент времени, определенный операндами < A > и < B >;

< D > – задает общее число транзактов, которое поступит в модель. Если его нет, формируется поступление бесконечного числа транзактов;

< E > – устанавливает уровень приоритета транзакта. По умолчанию уровень приоритета наивысший (0).

В модели может быть несколько блоков GENERATE.

Например,

GENERATE 20,,3,2, транзакт поступает с интервалом 20 единиц, смещение равно 0. Отсутствие операнда < B > задает детерминированный закон поступления; при D = 3 в систему будет подано 3 транзакта, приоритет которых равен 2.

Блок TERMINATE. Этот блок удаляет транзакт из модели (рис. 15).

Формат блока: TERMINATE < A >

Рис. 15. УГО блока TERMINATE

В опреранде < A > указывается удаляемый из модели транзакт.

Имеется только один операнд, который является указателем уменьшения счетчика завершения. Счетчик завершения – это ячейка памяти, в которой хранится целое положительное число, записанное в начале моделирования с помощью управляющего блока START < A >, < B >. В модели имеется один счетчик завершений и может быть несколько блоков TERMINATE. Транзакты, проходящие через блоки TERMINATE будут уменьшать содержимое счетчика завершений на суммарное количество единиц, записанных в операнде < A >.

Пример на составление сегмента таймера некоторой модели. Требуется промоделировать сегмент таймера при работе вычислительной сети в течение смены. Выбранная единица времени моделирования 1 мин, а все модельное время – 480 ед. времени.

Это соответствует программе:

1-й вариант: GENERATE 480
TERMINATE 1
START 1

2-й вариант: GENERATE 1
TERMINATE 1
START 480

Блок ADVANCE. Этот блок реализует задержку транзакта во времени (рис. 16).

Формат блока: ADVANCE < A >,< B >

Рис. 16. УГО блока ADVANCE

Назначение параметров:

< A > – среднее время задержки транзакта;

< B > – величина доверительного интервала времени задержки.

Например,

ADVANCE 20,5; задержка на 20 ± 5 ед.

При этом доверительный интервал соответствует равномерному закону распределения вероятности.

8.3.2. Аппаратно-ориентированные блоки GPSS

Блок SEIZE. Этим блоком производится занятие устройства (рис. 17).

Формат блока: SEIZE < A >

В операнде < A > указывается числовое либо символьное имя устройства.

Например:

Рис. 17. УГО блока SEIZE

SEIZE 12; занять устройство 12;

SEIZE OTO; занять устройство с символьным именем OTO.

Если присутствует попытка занять уже занятое устройство, то транзакт не попадает в устройство и остается в предыдущем блоке.

Блок RELEASE. Данный блок освобождает занятый некоторым транзактом прибор или устройство (рис. 18).

Рис. 18. УГО блока RELEASE

Формат блока: RELEASE < A >

В операнде < A > указывается имя устройства, которое необходимо освободить.

При работе модели автоматически собирается статистика в виде ответов на вопросы:

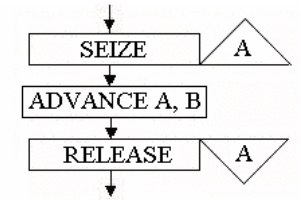


Рис. 19. Моделирование обработки в устройстве

1. Какую часть времени устройство было занято?

2. Сколько раз было занято устройство?

3. Какой средний интервал задержки устройства?

Обычно в программе 3 блока – занять, освободить, задержка – используются в определенной последовательности (рис. 19).

Это соответствует программе:

SEIZE PRT; занять устройство

ADVANCE 15,5; сформировать задержку 15 ± 5 ед.

RELEASE PRT; освободить устройство

8.4. Блоки QUEUE и DEPART

Блоки QUEUE и DEPART используются для сбора статистики об очередях.

Блок QUEUE. Этот блок ставит транзакт в очередь (рис. 20).

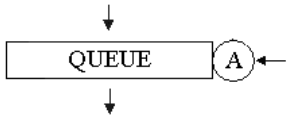


Рис. 20. УГО блока QUEUE

Формат блока: QUEUE < A >

В операнде < A > указывается цифровое или символьное имя очереди.

Напимер:

QUEUE 34; стать в очередь с номером 34.

QUEUE RS; стать в очередь с символьным именем RS

Блок DEPART. При выполнении этот блока транзакт покидает очередь с символьным или цифровым именем < A > (рис. 21).

Формат блока: DEPART < A >.

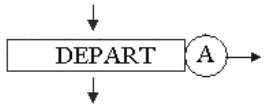


Рис. 21. УГО блока DEPART

Как правило, в модели бывает несколько очередей. При входе транзакта в блок QUEUE интерпретатор выполняет действия:

1. Счетчик входов увеличивается на 1.

2. Счетчик текущего содержимого очереди увеличивается на 1.

3. Производится привязка транзакта к очереди по ее имени.

4. Производится временная привязка транзакта к очереди.

При переходе транзакта в блок DEPART происходит следующее:

1. Счетчик текущего содержания очереди уменьшается на 1.

2. Интерпретатор определяет время пребывания транзакта в очереди, и если оно оказывается нулевым, то увеличивает счетчик нулевых вхождений на 1.

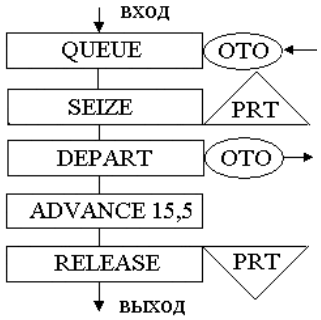


Рис. 22. Алгоритм программы моделирования очереди

3. Ликвидирует привязку данного транзакта к данной очереди.

Пример моделирования устройства с одной очередью (рис. 22)

8.5. Операторы управления программой SIMULATE, START, END

Оператор SIMULATE необходим для указания интерпретатору начала модели.

Формат записи: SIMULATE < A >.

Операнд < A > указывает продолжительность моделирования в минутах (A – целое число).

Оператор START начинает прогон модели.

Формат записи: START < A >,[< B >],[< C >],[< D >],

Назначение параметров:

< A > – задает начальное значение счетчика завершений (т. е. условное время моделирования в условных единицах);

< B > – управляет сбором статистики (если принимает значение NP, то сбор статистики не производится);

< C > – резервный операнд;

< D > – устанавливает интервал промежуточных выдач списков текущих и будущих событий.

Оператор END ставится в самом конце программы. Его выполнение прекращает работу в среде GPSS, и управление передается ОС. При этом

результаты сбора статистики о модели автоматически сохраняются в файле с именем REPORT.GPS.

Например:

SIMULATE 10; время моделирования 10 мин.

GENERATE 200; транзакт в модель генерируется с интервалом 200 ед.

TERMINATE 1; уменьшение счетчика завершения на 1 ед.

START 3,,1; начальное значение счетчика завершения 3 и статистика собирается на каждом шаге.

Пример моделирования системы с одним устройством и одной очередью. На обработку к серверу с именем PRT заявки поступают равномерно с интервалами 18 ± 6 с. Время обработки распределено также равномерно и составляет 16 ± 4 с. Порядок обслуживания: первым пришел – первым и обслуживается. Требуется смоделировать работу сервера в течение 1 смены и собрать статистику об очереди.

Блок-схема программы имеет следующий вид (рис. 23).

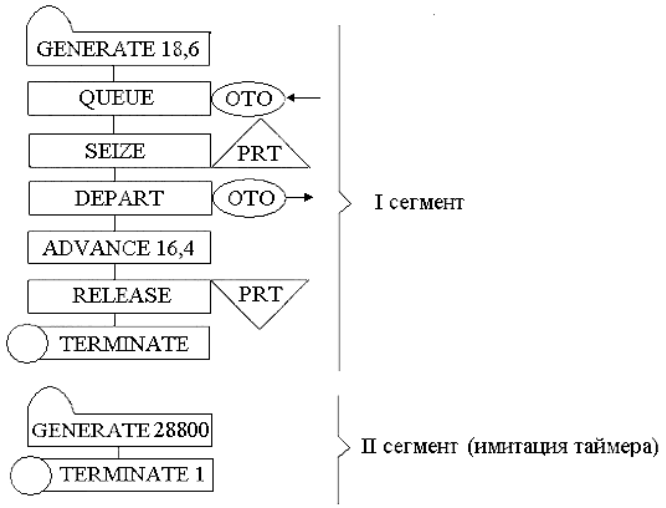


Рис. 23. Алгоритм программы моделирования системы с одним устройством и одной очередью

Текст программы имеет вид:

```
100 SIMULATE
; I сегмент
110 GENERATE 18,6; генерация транзактов
120 QUEUE  OTO; поставить в очередь OTO транзакт
130 SEIZE  PRT; занять устройство PRT
140 DEPART OTO; освободить очередь на один транзакт
150 ADVANCE 16,4; задержка 16 ±4 ед. для обработки заявки
160 RELEASE PRT; освободить устройство (сервер) от транзакта (заявки)
170 TERMINATE; уничтожить транзакт
; II сегмент
180 GENERATE  28800; транзакт в модель генерируется с интервалом 28 800 ед. (1 смена)
190 TERMINATE 1; уменьшение счетчика завершения на 1 ед.
200 START 1; начальное значение счетчика завершения 0, статистика собирается на каждом шаге.
```

Распечатка статистики. В версии GPSS/PC учитываются следующие статистические данные:

1. Начальное и конечное время моделирования.
2. Общее количество блоков, оборудования, каналов, очередей.
3. Описание блоков моделей с указанием имени, количества входов, текущего содержимого счетчика входов, коэффициента загрузки.
4. Описание очереди.

8.6. Использование интерпретатором цепей событий

Из пяти цепей событий в модели есть только одна цепь текущих событий и одна будущих.

В цепь текущих событий включаются транзакты, время наступления которых запланировано как текущее или ближайшее.

В цепь будущих событий включаются транзакты, попавшие в блок задержки. В этот же блок включаются транзакты, которые должны войти в модель в будущем через блок GENERATE.

Когда транзакт прекращает движение в модели, интерпретатор выполняет следующие действия:

1. Продолжает просмотр цепи текущих событий, выбирает следующий транзакт и продвигает его в модель.
2. Если нет такого транзакта в цепи текущих событий или предыдущий транзакт находится в блоках SEIZE или RELEASE, то просмотр цепи текущих событий начинается с ее начала.
3. Если в цепи текущих событий транзакта нет, то наступает фаза

корректировки таймера. При этом проверяется цепь будущих событий, таймер продвигается к ближнему значению, и этот транзакт вводится в модель из цепи будущих событий в цепь текущих.

Если в цепи будущих событий окажется несколько таких транзактов, то все они переносятся в цепь текущих событий в соответствии со своим уровнем приоритета.

Интерпретатор помечает в модели все транзакты, которые можно рассмотреть при распечатке цепей событий. Всего для транзактов есть 5 типов записи:

1. Номер транзакта.
2. Время движения.
3. Номер текущего блока, где находится транзакт.
4. Уровень приоритета.
5. Номер следующего блока.

Транзакты могут находиться в пассивном или активном состоянии.

В пассивном состоянии они располагаются в виде столбика стека и нумеруются начиная с вершины. При необходимости ввода транзакта в модель интерпретатор извлекает первый транзакт из вершины и через цепь будущих событий переводит его в активный буфер. После вывода из модели транзакт помещается в вершину пассивного буфера.

Первым действием интерпретатора является ввод модели. До фазы ввода модели цепи текущих и будущих событий пусты. В фазе ввода интерпретатор проверяет все блоки, отыскивая блок GENERATE. Найдя его, он определяет время прихода транзакта в модель. Если время равно нулю, то интерпретатор устанавливает время прихода транзакта в единицу.

Допустим, при чтении блока GENERATE 18,6 разыгранное значение времени равно 16. Интерпретатор в этом случае выбирает 1-й транзакт из пассивного буфера и помещает его в цепь будущих событий. Планируемое время появления транзакта составляет 16 мин.

Интерпретатор читает блок GENERATE 28800, извлекает из пассивного буфера транзакт N2 и помещает его в цепь будущих событий за транзактом N1. Читая блок START 1, интерпретатор устанавливает единицу в счетчик завершений. В момент окончания фазы ввода цепи событий имеют вид:

– цепь текущих событий (ЦТС) – пусто;

– цепь будущих событий (ЦБС) – <1 16 1>

<2 480 8>.

Пример моделирования системы с учетом цепей событий. Пусть на вычислительный центр в течение смены поступают заявки двух типов:

1) на обработку только по 1-му алгоритму;

2) на обработку по 1-му и 2-му алгоритму.

Распределение интервалов поступления заявок первого типа 35 ± 10 мин. Поступление заявок второго типа 60 ± 20 мин. На 1-й алгоритм затрачивается 10 ± 2 мин, а на процедуру 2-го алгоритма 18 ± 6 мин. Порядок обработки заявок: первый пришел, первый обслужен. Требуется составить программу модели, предусмотрев в конце моделирования распечатку информации об общей очереди и об ЦТС и ЦБС. Программа должна состоять из двух сегментов (по числу типов заявок) и в программу нужно включить сегмент таймера.

1. Алгоритм программы имеет вид:

а) для первого типа заявок:

```
GENERATE 35,10
QUEUE OTO_1
SEIZE SERV
DEPART OTO_1
ADVANCE 10,2
RELEASE SERV
TERMINATE
```

б) для второго типа:

```
GENERATE 60,20
QUEUE OTO2
SEIZE SERV
DEPART OTO_2
ADVANCE 10,2
ADVANCE 18,6
RELEASE SERV
TERMINATE
```

в) блок таймера

```
GENERATE 480
TERMINATE 1
```

Информация об очередях каждого типа заявок собирается двумя парами блоков:

```
1: QUEUE OTO_1
DEPART OTO_1
2: QUEUE OTO_2
DEPART OTO_2
```

2. Полный текст программы имеет вид:

```
SIMULATE
```

```
; 1-я группа деталей
GENERATE 35,10
QUEUE OTO_1
SEIZE SERV
DEPART OTO_1
ADVANCE 10,2
RELEASE SERV
TERMINATE
; 2-я группа деталей
GENERATE 60,20
QUEUE OTO_2
SEIZE SERV
DEPART OTO_2
ADVANCE 10,2
ADVANCE 18,6
RELEASE SERV
TERMINATE
; таймер
GENERATE  480
TERMINATE 1
START 1,,,1
END
```

Обозначения:

OTO_1 и OTO_2 – символические имена очередей заявок на вычислительный центр;

SERV – символическое имя устройства (сервера ВЦ)

Имитация обработки заявок на сервере моделируется блоками задержки ADVANCE.

Сбор статистики о цепях будущих и текущих событий осуществляется установкой единицы в параметр < D > оператора START.

8.7. Изменение дисциплины обслуживания транзактов. Блок TRANSFER

Пример моделирования процесса обработки сообщений 2-х типов на одном сервере. Сообщения 1-го типа имеют более низкий приоритет, чем сообщения 2-го типа. Это значит, что сообщения 2-го типа обрабатываются прежде, чем сообщения 1-го типа. Для сообщений 1-го типа:

– время обработки 300 ± 90 с;

– интервал поступления 420 ± 360 с.

Для сообщений 2-го типа:

– время обработки 100 ± 30 с;

– интервал поступления 360 ± 240 с.

Приоритет:

– сообщения 1-го типа – 1;

– сообщения 2-го типа – 2.

Сообщения 1-го типа поступают медленно и долго обрабатываются. Поэтому если исключить приоритет в обработке деталей, то детали 2-й группы будут накапливаться. Время моделирования – одна смена длительностью 28 800 с. Требуется оценить потери от задержки деталей при обработке.

При приоритетном обслуживании организуется дисциплина обслуживания: первым пришел – первым обслужен внутри приоритетного класса.

Схематически это можно показать следующим образом (рис. 24).

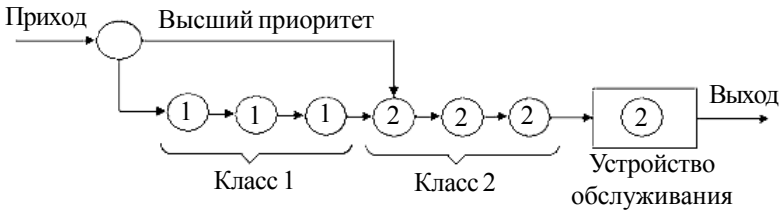


Рис. 24. Схема обслуживания транзактов с приоритетами

Текст программы:

```
SIMULATE
GENERATE 420,360,,,1; генерация транзакта с низшим уровнем приоритета 1
 QUEUE OTO_1; постановка в очередь приоритетного класса 1
SEIZE OTO; занятие сервера на обработку сообщения
DEPART OTO_1; освобождение приоритетной очереди на один транзакт класса 1
ADVANCE  300,90; обработка сообщения с приоритетом 1
RELEASE  OTO; освобождение устройства от транзакта
TERMINATE; вывод транзакта класса 1 из модели
GENERATE 360,240,,,2; генерация транзактов с высшим приоритетом 2
```

QUEUE OTO_2; организация очереди из сообщений приоритета обслуживания 2

SEIZE OTO; занятие устройства OTO на обработку

DEPART OTO_2; освобождение приоритетной очереди на один транзакт класса 2

ADVANCE 100,30; обработка

RELEASE OTO; освобождение устройства

TERMINATE; вывод транзакта из модели

; таймер

GENERATE 28800

TERMINATE 1

START 1

END

Результаты моделирования приведены в табл. 4.

Таблица 4

Результаты моделирования работы системы с приоритетами

Показатель (признак)	Количественное значение	
	с учетом приоритета	без учета приоритета
Коэффициент использования сервера OTO	0.952	0.932
Число обработанных сообщений	142	140
Среднее содержимое очереди	0.77	2.73

Блок TRANSFER. Этот блок обеспечивает безусловную передачу транзакта в блок, отличный от последующего по программе (рис. 25).

Формат блока: TRANSFER < A >,< B >

Назначение параметров:

< A > – не используется. На его место ставится «,»;

< B > – определяет имя блока, в который должен быть направлен транзакт.

Имя блока может иметь символическое либо цифровое обозначение.

Рис. 25. УГО блока TRANSFER

TRANSFER, < B >

Например,

TRANSFER, BET; безусловная передача транзакта на блок, имеющий символьное имя BET.

Блок TRANSFER никогда не отказывает транзакту во входе и сразу передает управление на блок, имя которого указано в операнде < B >.

Если нет условий для входа в указанный блок, то транзакт остается на входе и выводится из цепи текущих событий.

Пример моделирования процесса обработки изображений. Время обработки T = 30 ± 5 мин. Процесс обработки заканчивается сборкой изображения в течении 8 ± 2 мин. Работает несколько операторов этих изображений (транзакты).

Требуется смоделировать работу системы в течение 40 ч модельного времени. Оценить производительность участков при различном числе операторов.

Алгоритм программы представлен на рис. 26.

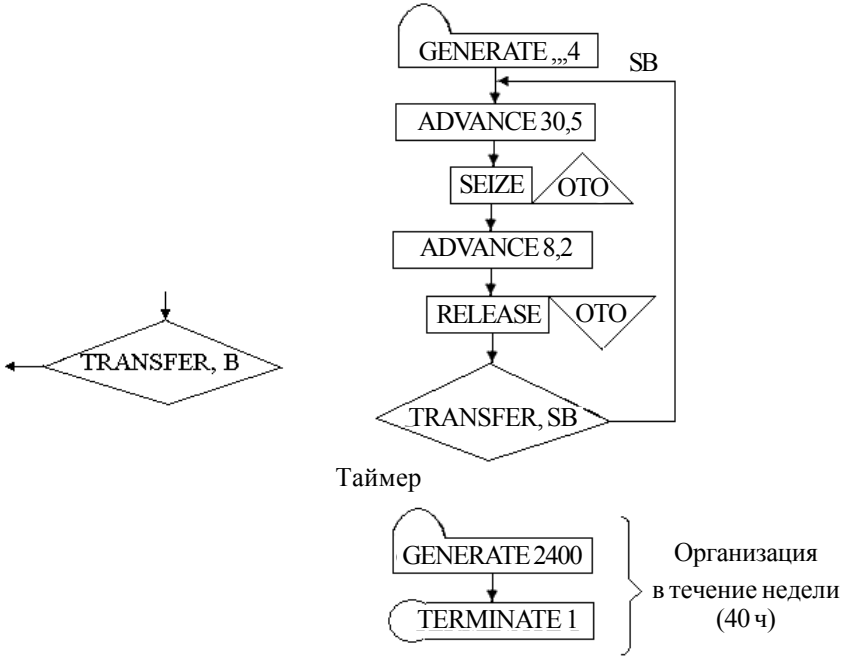


Рис. 26. Алгоритм моделирования процессов ветвления

В блоке GENERATE ,,,4 одновременно вырабатываются четыре транзакта (четыре оператора вступают в работу одновременно). В программе использованы блоки:

ADVANCE 30,5; задержка на обработку изображения

SEIZE OTO; занятие устройства (РС)
ADVANCE 8,2; непосредственная задержка на сборку
RELEASE OTO; освобождение устройства (РС)
TRANSFER, SB; передача изображений соответствующему оператору.
Текст программы:

```
SIMULATE
GENERATE ,,,4
SB ADVANCE 30,5
SEIZE OTO
ADVANCE 8,2
RELEASE OTO
TRANSFER, SB
; таймер
GENERATE 2400
TERMINATE 1
START 1
END
```

Для оценки производительности работы системы необходимо изменять параметр < D > в блоке GENERATE. После каждой замены запускать модель.

Статистика трех прогонов модели приведена в табл. 5.

Таблица 5

Показатели работы системы

Число операторов	Коэффициент использования OTO	Количество изображений
4	0.785	236
5	0.952	286
6	0.988	295

Блок TRANSFER в режиме случайной передачи. В этом случае на место операнда < A > записывается десятичная точка и до 3 цифр после точки, показывающих долю (частоту), с которой транзакты попадают в блок с именем, указанным в операнде < C >. Оставшаяся часть транзактов попадает в блок с именем, указанным в операнде < B > (рис. 27).

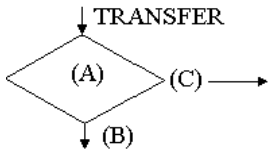


Рис. 27. УГО блока TRANSFER в режиме случайной передачи

Например,

TRANSFER .25,ALF,BET; 25 транзактов из 100, поступивших на вход блока TRANSFER попадут в блок с символьным именем BET, а остальные 75 попадут в блок ALF.

Блок TRANSFER в режиме условной передачи. В этом случае в операнде < A > записывается слово BOTH. Входящие в блок транзакты делают попытку войти сначала по адресу, указанному в операнде < B >. Если попытка неудачна (т. е. блок занят), то транзакт передается по адресу, указанному в операнде < C > (рис. 28).

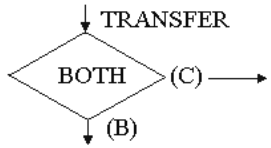


Рис. 28. УГО блока в режиме условной передачи TRANSFER

Например,

TRANSFER BOTH,GAM,DEL; транзакт поступает на вход блока с символьным именем GAM, а если он занят, то на вход блока DEL.

Если блоки GAM и DEL заняты, то транзакт остается на входе блока TRANSFER, ожидая готовности этих блоков.

Управляющий блок CLEAR. Блок очистки CLEAR используется для подготовки модели к повторному прогону после переопределения некоторых операндов блоков.

Когда интерпретатор GPSS выполняет операцию CLEAR, то в ноль сбрасывается вся статистика, включая таймер. Все транзакты возвращаются в пассивный буфер и заново выполняется фаза ввода транзакта в модель. Не возвращаются в начальные состояния лишь датчики случайных чисел. Поэтому результаты моделирования при многократных прогонах могут отличаться друг от друга. Далее интерпретатор читает следующий блок (чаще всего блок START), устанавливая этим счетчик завершения в нужное значение.

П р и м е ч а н и е. Обязательно блоку, в котором переопределяется операнд, необходимо присвоить имя. При распечатке статистики, с использованием блока CLEAR, после блоков с переопределенными операндами печатается текст, указывающий, что в предыдущем блоке повторно используется имя.

Пример (сборка изображений).

```
SIMULATE
VAR GENERATE ,,,4
SB ADVANCE 30,5
SEIZE OTO
ADVANCE 8,2
RELEASE OTO
```

```
TRANSFER, SB
; таймер
GENERATE 2400
TERMINATE 1
;
START 1
;
VAR GENERATE ,,,5
CLEAR
START 1
;
VAR GENERATE ,,,6
CLEAR
START 1
END
```

8.8. Моделирование многоканальных устройств

Многоканальное устройство имеет возможность одновременного обслуживания нескольких транзактов или сообщений. Под емкостью многоканального устройства понимают число одновременно обрабатываемых в этом устройстве транзактов.

Блок ENTER. Он предназначен для оптимальной работы многоканальных устройств (рис. 29). При входе транзактов в блок ENTER происходит занятие одного канала многоканального устройства.

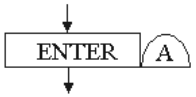


Рис. 29. УГО блока ENTER

Формат блока: ENTER < A >,< B >

Назначение параметров:

< A > – имя многоканального устройства;

< B > – указывает, на сколько уменьшается емкость многоканального устройства при поступлении транзакта.

Блок LEAVE. Этот блок обеспечивает освобождение многоканального устройства от транзакта (рис. 30).

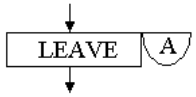


Рис. 30. УГО блока LEAVE

Формат блока: LEAVE < A >,< B >

Назначение операндов < A >,< B > аналогично назначению операндов блока ENTER.

При поступлении транзакта в блок ENTER происходит следующее:

1) счетчик входов многоканального устройства увеличивается на 1;

2) текущее содержимое многоканального устройства увеличивается на 1;

3) доступная емкость многоканального устройства уменьшается на 1.

Когда транзакт входит в блок LEAVE:

1) текущее содержимое счетчиков входов уменьшается на 1;

2) доступная емкость увеличивается на 1.

В процессе моделирования автоматически формируется текущая статистика, включающая:

1) содержимое счетчиков входов;

2) текущее содержимое многоканального устройства;

3) максимальное содержимое многоканального устройства;

4) среднее время пребывания, определяемое отношением полного времени занятости многоканального устройства к значению счетчиков входов.

Например:

```
:
ENTER MAP,3
ADVANCE 10,5
:
LEAVE MAP, 3
```

В блоке ENTER занимается многоканальное устройство с именем MAP. При входе транзакта в этот блок происходит занятие сразу 3-го канала многоканального устройства и текущее содержимое многоканального устройства уменьшается сразу на 3.

Когда транзакт поступает в блок LEAVE, происходит следующее:

1) счетчик входов уменьшается на 1;

2) текущее содержимое многоканального устройства увеличивается на 3 (доступная емкость увеличивается на 3).

Блок STORAGE. Этим блоком задается емкость многоканального устройства.

Формат блока: < NAME > STORAGE < A >

В поле < NAME > указывается имя многоканального устройства. В поле операций пишется слово STORAGE. В поле операнда на месте операнда < A > – емкость многоканального устройства.

Например,

MAP STORAGE 10; многоканальное устройство MAP может обрабатывать 10 транзактов одновременно.

Если емкость многоканального устройства не указана, то она является бесконечно большой (2^31–1).

Блок STORAGE всегда записывается в начале программы за оператором SIMULATE.

Если транзакт пытается войти в многоканальное устройство, содержимое которого равно заданной емкости, то попытка будет неудачна и транзакт останется в предыдущем блоке.

В случае когда используется многоканальное устройство, в системе с очередью без учета приоритета реализуется принцип: если пришла очередь транзакта, то он идет на обслуживание к любому не занятому каналу.

В случае когда учитываются приоритеты канала многоканального устройства, транзакты выбираются по принципу: первый пришел – первый обслужен внутри приоритетного класса.

Пример моделирования многоканальных устройств. На вычислительном центре работают 50 ЭВМ по 8 ч в день и по 5 дней в неделю. Время наработки на отказ каждой машины составляет 150 ± 25 ч. Время ремонта машины 7 ± 3 ч.

Ограничения:

а) численность ремонтного персонала;

б) количество резервных машин.

Требуется промоделировать работу системы, определив необходимое соотношение «число резервных машин – число ремонтников». Время моделирования – 50 недель (2000 ч). В задаче принять 2 многоканальных устройства, имитирующих работу системы машин и наладчиков.

Допустим, что в результате предварительной оценки оказалось, что область изменения числа ремонтников от 3 до 5, а область изменения резерва от 3 до 5 машин.

Текст программы:

```
SIMULATE
MAP STORAGE 50; определение емкости многоканальных
NAL STORAGE 3; устройств
VAR GENERATE ,,,53
RES ENTER MACH; занятие устройства
ADVANCE 150,25
LEAVE MACH; освобождение одной из неисправных машин
ENTER NAL
```



```
ADVANCE 7,3
LEAVE NAL
TRANSFER, RES
GENERATE 2000
TERMINATE 1
START 1
; продолжение программы для моделирования различных сочетаний числа наладчиков и резервных машин
VAR GENERATE ,,,54; моделирование при резерве 4 машины и числе наладчиков 3
CLEAR
START 1
VAR GENERATE ,,,55; моделирование при резерве 5 машин и числе наладчиков 3
CLEAR
START 1
; моделирование при изменении числа наладчиков
NAL STORAGE 4; 4 наладчика и 5 резервных машин
VAR GENERATE ,,,53
CLEAR
START 1
:
NAL STORAGE 4
VAR GENERATE ,,,55;
CLEAR
START 1
; моделирование при резерве 3 машины и числе наладчиков 5
NAL STORAGE 5
VAR GENERATE ,,,53
CLEAR
START 1
```

П р и м е ч а н и е. При переопределении операндов блоков последним присваиваются имена.

8.9. Управляющий оператор RESET

Оператор RESET осуществляет сброс статистики.

В предыдущем примере имеется этап моделирования переход-

ного процесса до выхода хотя бы одной машины из строя. Этот режим является нетипичным. Поэтому необходимо производить учет этих переходных процессов следующими способами:

1. Выбор достаточно большого промежутка времени моделирования (время переходного процесса не должно превышать 5% от общего времени моделирования).

2. Отбросить статистику переходного режима с помощью оператора RESET.

Когда выполняется оператор RESET, статистика сбрасывается в ноль. При этом поток случайных чисел не устанавливается в исходное состояние (в отличие от CLEAR). Счетчик текущих значений устанавливается равным числу транзактов, находящихся в блоках. Счетчик числа входов в блоке сбрасывается в ноль. Таймер относительного времени (C1) устанавливается в ноль, таймер абсолютного времени не сбрасывается в ноль.

Пример. Используя предыдущий пример проиллюстрируем действие оператора RESET, , в котором сброс статистики осуществляется через 40 ч модельного времени (см. с. 80). Программа для 4-х различных машин и 4-х наладчиков примет вид:

```
SIMULATE
MACH STORAGE 50
NAL STORAGE 4
VAR GENERATE „„54
RES ENTER MACH
ADVANCE 150,25
LEAVE MACH
ENTER NAL
ADVANCE 7,3
LEAVE NAL
TRANSFER, RES
GENERATE 40
TERMINATE 1
START 1
RESET
START 1
24 раза:
RESET
START 1
```

Результаты моделирования представлены в табл. 6.

Таблица 6

Результаты моделирования

№ недели	Использование MACH (RESET)	Использование MACH (без RESET)
1	0.974	0.984
2	1	0.987
3	1	0.991
4	0.905	0.970
5	0.632	0.922
6	0.901	…
7	1	…
8	0.987	…
9	0.984	…
10	0.999	…
…	0.999…1.000	0.972

8.10. Использование дискретных распределений в блоках GENERATE и ADVANCE

8.10.1. Равномерное дискретное распределение

Сущность дискретного равномерного распределения сводится к двум процедурам:

1) разыгрывается случайное число из выборки равномерно распределенных чисел в интервале от 0 до 1;

2) полученное число преобразовывается в эквивалентное значение, принадлежащее заданной выборке.

В языке GPSS имеется 8 генераторов случайных чисел, стандартные числовые атрибуты которых имеют вид: RN1, RN2, ..., RN8.

Пусть при первом обращении к датчику случайного числа RN1 разыграно число 0.000573. Далее эта величина преобразуется в значение из другого распределения, которое задано программистом. Формула преобразований реализуется по пропорции:

1 – – > (2B+1)

X – – > N,

где B – доверительный интервал;

X – разыгранное число;

N – число, используемое при моделировании.

В случае когда генератор случайного числа используется в качестве аргумента функции, он выдает значения 0.000000–0.999999. Во всех остальных случаях: 000–999.

Например:

GENERATE 420,360; транзакты генерируются с интервалом 420 ± 360

ADVANCE 360,240; задержка транзакта 360 ± 240.

Пример. Рассчитаем момент прихода в модель транзакта, который генерируется блоком GENERATE, при условии, что разыграна случайная величина с RN1 = 0.000573. Время появления транзакта в модели формируется в соответствии с рис. 31.

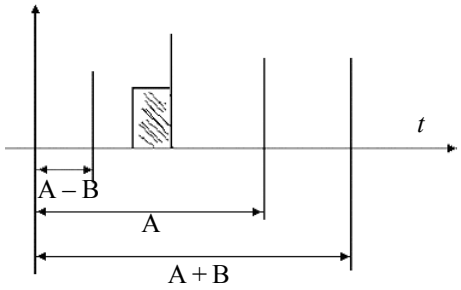


Рис. 31. Временная диаграмма поступления транзактов в модель

Время прихода транзакта:

$$T = A \pm B = A - B + RN1 * (2B + 1) =$$

$$= 420 - 360 + 0.000573 * (2 * 360 + 1) = 60.4.$$

Интерпретатор принимает время, равное 60 ед.

8.10.2. Неравномерное дискретное распределение

Пусть случайная переменная принимает следующие значения: 2, 5, 8, 9, 12 соответственно с относительной частотой 0.15, 0.2, 0.25, 0.22, 0.18 (неравномерное распределение). Распределение случайной величины приведено в табл. 7.

Таблица 7

Вероятностные характеристики случайной величины

Значение случайной переменной	Относительная частота	Суммарная частота	Диапазон	Интервал
2	0.15	0.15	0.00–0.15	1
5	0.20	0.35	0.15–0.35	2
8	0.25	0.60	0.35–0.60	3
9	0.22	0.82	0.60–0.82	4
12	0.18	1.00	0.82–1.00	5

Пусть генератор случайных чисел с равномерным распределением из интервала (0..1) выбрал число 0.528637. Оно относится к третьему интервалу таблицы, поэтому случайной переменной будет число 8.

Итак, интерпретатору для формирования случайных функций необходимы:

– генератор случайного числа;

– диапазон изменения суммарной частоты;

– соответствующее значение случайной функции.

Формат случайной функции в GPSS имеет следующий вид:

< номер строки > < имя функции > FUNCTION < A >,< B >

Операнд < A > задает номер генератора случайных чисел (из набора: RN1, RN2, ..., RN8).

В поле операнда < B > для дискретной функции пишется буква «D», после которой указывается число дискретных значений функции.

Например:

200 MAP FUNCTION RN1,D5

.15,2/.35,5/.6,8/.82,9/1,12

В 200-й строке определена случайная дискретная функция с неравномерным распределением. В качестве источника случайных чисел будет генератор RN1, число дискретных опорных точек функции – 5.

В качестве аргумента функции во второй строке записывается суммарная частота появления событий, а в качестве 2-го значения в строке используется значение самой функции.

Примечания:

1) аргумент и сама функция должны быть монотонно возрастающими (нелинейно);

2) описание самой функции должно быть в начале программы, сразу за блоком SIMULATE.

Пример сборки изображений на компьютере (см. с. 75). Изменим распределение интервала сборки с равномерного на неравномерное.

Закон распределения времени сборки изображений запишем в табл. 8.

Таблица 8

Распределение времени сборки изображения

Время сборки, мин	6	7	8	9	10
Относительная частота	0.05	0.25	0.40	0.25	0.05
Суммарная частота	0.05	0.30	0.70	0.95	1.00

Текст программы имеет вид:

```
SIMULATE
ABC FUNCTION RN1,D5
.05,6/.3,7/.7,8/.95,9/1,10
GENERATE ,,,4
SBOR  ADVANCE 30,5
SEIZE OTO
ADVANCE FN$ABC
RELEASE OTO
TRANSFER, SBOR
; таймер
GENERATE 2400
TERMINATE 1
START 1
```

В этой программе процесс сборки изображений имитируется блоком задержки ADVANCE FN$ABC, т. е. носит случайный характер с неравномерным законом распределения в соответствии с дискретной случайной функцией ABC, заданной пятью опорными точками.

8.10.3. Моделирование непрерывных случайных функций

При моделировании дискретной функции разыгрывается случайное число, которое преобразуется в аргумент функции. При просмотре таблицы отыскивается интервал суммарной частоты, в который попадает разыгранное число. Второй элемент пары чисел будет искомым значением функции.

В случае непрерывной функции производится линейная интерполяция для пары точек, находящихся на краях интервала значений суммарной частоты, на которую выпало случайное число, являющееся аргументом.

Если непрерывное распределение равномерно, то функция задается двумя парами точек. Причем, первое значение функции входит в искомый интервал, а второе – не входит.

Пример функции для равномерного непрерывного распределения. Пусть задан интервал функции [2..10]. Тогда задание этой функции на языке GPSS будет иметь вид:

```
ABC FUNCTION RN3,C2
0,2 / 1,10
```



```
GENERATE FN$ABC
:
```

Вероятность того, что выпадет значение <2, равна 0, а вероятность того, что выпадет значение от 2 до 10, равна 1.

Транзакты поступают в модель от блока GENERATE в случайные моменты времени из диапазона 2, 3, 4, 5, 6, 7, 8, 9, 10 (диапазон целых чисел) с равномерным распределением.

Представление непрерывной функции с неравномерным распределением приведены в табл. 9.

Таблица 9

Непрерывные функции с неравномерным распределением

Интервал	Относительная частота событий	Суммарная частота
<15	0.0	0.0
15..30 –	0.07	0.07
30..45 –	0.25	0.32
45..60 –	0.41	0.73
60..75 –	0.19	0.92
75..90 –	0.08	1.00

П р и м е ч а н и е. Знак (–) показывает, что случайные значения не включены в данный интервал.

Например:

```
MAK FUNCTION RN2,C6
```

0,15/.07,30/.32,45/.73,60/.92,75/1,90; будем описывать непрерывную случайную функцию MAK шестью точками (C6).

8.10.4. Моделирование пуассоновских потоков и экспоненциально распределенных интервалов времени обслуживания

Пуассоновский поток заявок – это такой поток, при котором выполняются три условия:

1) вероятность того, что событие (заявка) поступит на малом интервале, пропорциональна длине этого интервала;

2) вероятность одновременного поступления нескольких заявок пренебрежительно мала на данном малом интервале;

3) интервалы не зависят друг от друга.

При пуассоновском законе поступления заявок распределение вероятности является экспоненциальным. В GPSS для моделирования пуассоновского потока используется экспоненциальная функция, реали-

зованная 24-мя парами чисел. Причем значения функции берутся от 0 до 8. Эта функция имеет единичное среднее значение.

При использовании этой функции в блоках GENERATE и ADVANCE в качестве операнда <A> берут среднее масштабированное значение функции, а операнд <B> указывает значение единичной экспоненциальной функции. При необходимости производится масштабирование.

Задание экспоненциальной функции:

XPDIS FUNCTION RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509
.5,.69/.6,.915/.7,1.2/.75,1.38/.8,.1.6
.84,.1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81
.95,2.99/.96,3.2/.97,3.5/.98,3.9/.99,4.6
.995,5.3/.998,6.2/.999,7/.9998,8

Например,

GENERATE 200,FN$XPDIS; транзакты поступают в модель со средним значением интервала 200 ед. времени и величиной отклонения, соответствующей экспоненциальному закону распределения.

8.10.5. Выборка из нормального распределения

Нормальное распределение (распределение Гаусса) случайной величины (СВ) – это такое распределение, плотность вероятности $f(x)$ которого имеет вид:

$$f(x) = \frac{1}{\sqrt{2\pi}\cdot\sigma} e^{-(x-a)^2/2\sigma^2}$$

где a – точка максимума и одновременно центр симметрии кривой;

σ – расстояние от центра симметрии до точки перегиба.

Плотность нормированного и центрированного нормального распределения получается при $a = 0, \sigma = 1$: $\varphi(x) = 1/\sqrt{2\pi}\cdot e^{-0,5x^2}$.

СВ с нормальным распределением описывается математическим ожиданием и стандартным отклонением (доверительный интервал).

Нормированная СВ нормального распределения имеет математическое ожидание, равное 0, и стандартное отклонение, равное 1.

Для организации ненормированной выборки с нормальным распределением в языке GPSS используется следующая формула:

GNORMвыб = GNORMст.откл * SNORMвыб + GNORMмо,

где G – ненормированная выборка;

S – нормированная выборка.

Нормированная табличная выборка по закону Гауса записывается в следующем виде:

SNORM FUNCTION RN1,C25
0,– 5/.00003,– 4/.00135,– 3/.00621,– 2.5/.02275,– 2/.6681,– 1.5
.11507,– 1.2/.15866,– 1/.21186,– .8/.27425,– .6/.34458,– .4
.42074,– .2/.5,0/.57926,.2/.65542,.4/.72575,.6/.788814,.8
.84134,1/.88493,1.2/.93319,1.5/.97725,2/.99379,2.5
.99865,3/.99997,4/1,5

Доверительный интервал равен 0..1.

Значение функции находится в интервале 5..5.

Например:

GNORM VARIABLE 2#FN$SNORM+10; описание переменной GNORM с нормальным распределением

⋮

GENERATE V$GNORM; источник транзактов с нормальным распределением времени поступления.

Блоком GENERATE генерируется транзакт с нормальным законом распределения вероятности.

Математическое ожидание равно 10, а отклонение (доверительный интервал) равен 2.

Необходимо следить, чтобы не получилось отрицательное значение величины 2*FN$SNORM+10.

8.11. Стандартные числовые атрибуты (СЧА)

Стандартные числовые атрибуты – это условное обозначение объектов модели, к которым могут обращаться программы в процессе моделирования.

Задание операнда в виде СЧА включает:

в 1-й части – групповое имя или тип объекта;

во 2-й части – идентификатор конкретного члена группы.

При определении функции и использовании в качестве операндов СЧА применяется косвенное задание операнда.

Различают:

– системные СЧА;

– СЧА объектов моделей.

К системным относятся СЧА:

1) обозначения относительного системного времени – C1;

2) абсолютного системного времени – AC1;

3) генератора случайных чисел равномерного распределения – RN;

4) приоритета транзакта – PR;

5) возвращающий номер активного сообщения – XN1.

СЧА объектов моделей включают стандартные числовые атрибуты:

– устройств;

– многоканальных устройств;

– очередей;

– блоков;

– транзактов: Pj, MPj, MBj;

– функций: FNj;

– переменных: Vj, BVj;

– ячеек памяти: Xj, MXj;

– таблиц: TBj, TCj, TDj,

где j – номер либо символическое имя.

СЧА устройств, многоканальных устройств, очередей и блоков приведены в табл. 10.

Таблица 10

Стандартные числовые атрибуты транзактов

Наименование	Значение
СЧА устройств	
Fj (F$< имя >)	1 – занято; 0 – свободно
FCj (FC$< имя >)	Число занятий устройства
FRj (FR$< имя >)	Нагрузка устройства (в долях от тысячи)
FTj (FT$< имя >)	Целая часть среднего времени задержки транзакта в устройстве
СЧА многоканальных устройств	
Rj (R$< имя >)	Доступная емкость
Sj (S$< имя >)	Текущее содержимое многоканального устройства
SAj (SA$< имя >)	Целая часть среднего содержимого многоканального устройства
SCj (SC$< имя >)	Число входов в многоканальное устройство
SMj (SM$< имя >)	Максимальное содержимое многоканального устройства
SRj (SR$< имя >)	Нагрузка в долях тысячи устройства
STj (ST$< имя >)	Целая часть средней задержки транзакта многоканальным устройством

Наименование	Значение
СЧА очередей	
Qj (Q$< имя >)	Текущее содержимое очереди
QAj (QA$< имя >)	Целая часть средней длины очереди
QCj (QC$< имя >)	Число входов в очередь с заданным именем
QMj (QM$< имя >)	Максимальная длина очереди за время моделирования
QTj (QT$< имя >)	Целая часть среднего времени пребывания в очереди
QXj (QX$< имя >)	Целая часть среднего времени пребывания в очереди без нулевых вхождений
QZj (QZ$< имя >)	Число нулевых вхождений в очередь
СЧА блоков	
Wj (W$< имя >)	Текущее содержимое транзактов в блоке
Nj (N$< имя >)	Число входов в блок

Например,

100 PRIM ADVANSE FC$ABC; осуществляется задержка транзакта на число занятий устройства с именем ABC.

Пример программы моделирования системы с пуассоновским входящим потоком требований. Пусть в системе имеется один прибор с очередью, в котором осуществляется обслуживание требований с экспоненциальным законом распределения. Интенсивность поступлений – 12 заявок в час (1 заявка в 300 с). Среднее время обслуживания зависит от числа заявок в очереди (табл. 11).

Таблица 11

Среднее время обслуживания заявок

Длина очереди	Среднее время обслуживания, мин
0	5.5 (330 с)
1 или 2	5.0 (300 с)
3, 4, 5	4.5 (270 с)
6 и более	4 (240 с)

Время моделирования – одна смена (28800 с).

Программа имеет вид:

```
SIMULATE
XPDIS FUNCTION RN1,C24
0,0/ .....
...../0.9998, 8
```

QUEF FUNCTION Q$ABC,D4; определение функции OUEF 4-мя дискретными значениями (функции времени обслуживания)

; аргумент функции QUEF – длина очереди с именем ABC
0,330/2,300/5,270/6,240
GENERATE 300,FN$XPDIS
QUEUE ABC
SEIZE OTO
DEPART ABC; освобождение очереди на 1 транзакт
ADVANCE FN$QUEF,FN$XPDIS; задержка транзакта на величину, равную значению длины очереди с экспоненциальным распределением
RELEASE OTO; освобождение устройства OTO
TERMINATE; транзакт уничтожен, но счетчик завершения не изменяется
GENERATE 28800
TERMINATE 1
START 1

8.12. Параметры транзактов

При прохождении транзактов через отдельные блоки их характеристики могут изменяться. Для фиксации этих изменений используются параметры транзактов, число которых задается операндом < D > блока GENERATE. Если этот операнд опущен, то по умолчанию считается, что существует 12 параметров транзактов. Для обозначения параматров транзактов применяется СЧА Pj (P1, P2, ..., P12). В качестве параметров могут использоваться символьные обозначения: P$PRI1, P$PRI2, ...

Например,

ADVANCE P5,2; блок осуществляет задержку транзакта на время, среднее значение которого находится в пятом параметре текущего транзакта, 2 – доверительный интервал равномерного распределения.

Блок ASSIGN. Он изменяет значения параметров транзактов (рис. 32).

Формат блока: ASSIGN < A >,< B >

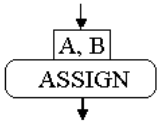


Рис. 32. УГО блока ASSIGN

Блок используется в 3-х режимах:

1. **Режим замещения**. В этом случае в операнде < A > указан параметр, подлежащий замещению. В операнде < B > задается новое значение параметра.

Например:

ASSIGN P2,83; при вхождении транзакта параметр P2 принимает значение 83.

ASSIGN 34,FN$EX3; при вхождении транзакта его 34-й параметр изменяется на значение функции с именем EX3.

2. **Режим приращения** .

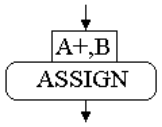


Рис. 33. УГО блока ASSIGN в режиме приращений

Формат блока: ASSIGN A+,B

В режиме приращения текущее значение параметра транзакта увеличивается на величину операнда < B > (рис. 34).

Например,

ASSIGN 48+,V$PC3; при вхождении транзакта текущее значение 48-го параметра увеличится на величину переменной PC3.

3. **Режим вычитания.**

Формат блока: ASSIGN < A– >,< B >

В режиме вычитания текущее значение параметра транзакта уменьшается на величину операнда < B > (рис. 34).

Например:

ASSIGN 3–,83; 3-й параметр транзакта уменьшится на 83 ед.

ASSIGN 34–,X$EX1; 34-й параметр транзакта уменьшится на величину ячейки памяти с именем EX1.

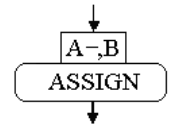


Рис. 34. УГО блока ASSIGN в режиме вычитания

Дополнительная форма записи блока: ASSIGN < A >,< B >,< C >

Назначение параметров:

< A > – номер, символьное имя или СЧА параметра транзакта;

< B > – среднее значение этого параметра;

< C > – доверительный интервал равномерного распределения вероятности.

Приоритет транзакта. В процессе моделирования с помощью блоков GPSS можно изменять уровень приоритета транзактов. Уровень приоритета транзактов задается или изменяется с помощью блока PRIORITY (рис. 35).

Формат блока: PRIORITY < A >

В операнде < A > указывается новое значение приоритета транзакта.

Например,

120 M1 PRIORITY 2; транзакт, проходя через блок, изменяет свой уровень приоритета, который становится равным 2.

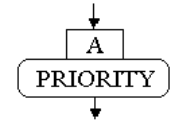


Рис. 35. УГО блока PRIORITY

8.13. Блок MARK

Блок MARK либо заменяет значение отметки времени сообщения на текущее значение абсолютного модельного времени (операнд А не определен), либо записывает значение модельного времени в заданный параметр транзакта (при использовании операнда < A >).

Формат блока: MARK < A >

Поле < A > содержит ссылку на номер параметра транзакта, в который записывается значение абсолютного модельного времени. Если такого параметра не существует, то он создается. Операнд < A > может быть именем, положительным целым числом, СЧА или СЧА*< параметр >.

Например:

MARK BEGINNING AC1; когда транзакт входит в блок MARK, его параметру с именем BEGINNING присваивается значение абсолютного модельного времени AC1.

MARK; значение отметки времени обрабатываемого в данный момент транзакта становится равным значению абсолютного модельного времени.

Исходное значение времени создания транзакта может быть заменено на текущее значение абсолютного модельного времени при прохождении сообщения через блок MARK.

Каждое сообщение имеет следующие два стандартных числовых атрибута, связанных с временем прохождения участков модели данным транзактом:

M1 – время прохождения сообщением модели. Вычисление значения этого СЧА производится следующим образом:

$$M1 = \left|\begin{array}{l}\text{Текущее значение}\\ \text{абсолютного модельного}\\ \text{времени}\end{array}\right| - \left|\begin{array}{l}\text{Отметка времени}\\ \text{обрабатываемого в}\\ \text{данный момент сообщения}\end{array}\right|;$$

MPj – промежуточное время прохождение сообщением участка модели. Вычисление значения этого СЧА производится следующим образом:

$$MPj = \left|\begin{array}{l}\text{Текущее значение}\\ \text{абсолютного модельного}\\ \text{времени}\end{array}\right| - \left|\begin{array}{l}\text{Значение параметра}\\ \text{обрабатываемого в}\\ \text{данный момент сообщения}\end{array}\right|.$$

8.14. Таблицы

8.14.1. Оператор описания таблицы

Оператор TABLE определяет аргумент, а также число и ширину частотных классов.

Формат оператора: < NAME > TABLE < A >,< B >,< C >,< D >

Метка < NAME > определяет имя таблицы.

Назначение параметров:

< A > – задается аргумент таблицы (элемент данных, частотное распределение которого будет табулироваться). Операнд может быть именем, целым числом, СЧА или СЧА*< параметр >;

< B > – задается верхний предел первого частотного класса. Операнд может быть целым числом или именем;

< C > – задается ширина частотного класса (разница между верхней и нижней границей каждого частотного класса). Операнд может быть положительным целым числом;

< D > – задается число частотных классов. Это число не может превышать 8191. Операнд может быть положительным целым числом.

Для сбора элементов данных сообщение должно войти в блок TABULATE с тем же именем таблицы, которое определено в блоке TABLE.

Когда транзакт входит в блок TABULATE, оценивается аргумент таблицы (операнд < A > в операторе TABLE). Если он меньше или равен операнду < B > в операторе TABLE, то выбирается первый частотный класс таблицы. Если аргумент таблицы не подходит для этого класса, то значение операнда B выбирается путем деления значения аргумента на операнд < C > оператора TABLE. Нижняя граница частотного класса включается в предыдущий класс. Если таблицы недостаточно для размещения значения аргумента, то выбирается последний частотный класс. После выбора частотного класса счетчик числа попаданий в него увеличивается на величину, определяемую операндом < B > оператора TABULATE. По умолчанию увеличение происходит на 1. В конце работы оператора TABULATE изменяются среднее значение и значение стандартного отклонения аргумента таблицы.

Таблица может быть переопределена или переинициализирована другим оператором TABLE, с той же самой меткой, что и первая.

Стандартные числовые атрибуты, связанные с описываемым оператором, следующие:

TB – среднее значение аргумента;
TC – число входов в таблицу;
TD – стандартное отклонение.
Блок, связанный с оператором TABLE, является оператор TABULATE.

8.14.2. Оператор описания Q-таблицы

Оператор QTABLE является средством получения распределения времени пребывания сообщений в очереди.

Формат оператора: < NAME > QTABLE < A >,< B >,< C >,< D >

Метка < NAME > определяет имя таблицы.

Назначение параметров:

< A > – имя очереди. Операнд может быть положи тельным целым или именем.

< B > – верхняя граница первого частотного класса. Операнд может быть нулем или положительны целым числом.

< C > – размер частотного класса – разница между верхней и нижней границей каждого частотного класса. Операнд может быть положительным целым числом.

< D > – число частотных классов. Оно не может превышать 8191. Операнд может быть положительным целым числом.

В число стандартных статистических данных об очереди входит среднее время задержки сообщения в очереди. Это значение равно частному от деления временного интервала длины очереди на общее число входов в очередь.

Q-таблица может быть переопределена другим QTABLE оператором с той же меткой, что и первый.

Время пребывания транзакта в очереди записывается в Q-таблицу после прохождения транзактом соответствующего блока DEPART.

8.14.3. Блок TABULATE

Блок TABULATE табулирует текущее значение заданного аргумента. Способ табуляции зависит от режима работы таблицы, который определяется оператором описания таблицы TABLE.

Формат блока: TABULATE < A >,[< B >]

Назначение параметров:

< A > – номер или имя таблицы, в которой табулируется значение

аргумента. Операнд должен быть именем, положительным целым, СЧА или СЧА*< параметр >. Таблица должна быть определена оператором описания TABLE;

< B > – число единиц, которые должны быть занесены в тот частотный класс, в который попало значение аргумента. Если поле < B > пусто, эта величина полагается равной единице. Операнд < B > может быть именем, положительным целым числом, СЧА или СЧА*< параметр >.

Когда транзакт входит в блок TABULATE, то для нахождения таблицы используется операнд <A>. Если такой таблицы нет, то возникает ошибка выполнения. Таблица должна быть определена оператором TABLE. Таблица изменяется в соответствии с операндами оператора TABLE.

Например:

```
RTIME TABLE M1,100,100,9
INQUE QTABLE ONE,0,600,20
:
QUEUE  ONE
ENTER TERL
DEPART ONE
TABULATE RTIME
:
```

В примере в таблице с именем RTIME табулируется время прохождения сообщением части модели. В таблице с именем INQUE табулируется время пребывания сообщений в очереди ONE.

Например:

```
ABC TABLE MP5,100,50,6
:
MARK 5
:
TABULATE ABC
```

В первой строке описана таблица с именем ABC. В таблицу будет занесен СЧА MP5. Начальное значение первого поддиапазона 100, величина диапазона 50, число диапазонов 6.

Блок MARK 5 заносит значение абсолютного модельного времени в параметр 5-го транзакта.

Блок TABULATE заносит в таблицу в соответствующий диапазон разницу dT между текущим значением модельного времени и значением параметра 5-го текущего транзакта: dT = AC1 – < P5 >.

8.15. Арифметические переменные

Для обозначения и использования арифметических переменных применяют СЧА – Vj (V$< имя >). Индекс j – номер арифметической переменной. Арифметическая переменная задает выражение, которое представляет собой набор данных, связанных арифметическими операциями. В качестве данных могут использоваться СЧА.

GPSS поддерживает следующие арифметические операции:

«–» – вычитание;

«+» – сложение;

«#» – умножение;

«@» – деление по модулю;

«^» – возведение в степень.

Например,

c1/100#V$PR34

В примере относительное модельное время делится нацело на 100 и умножается на переменную с символьным именем PR34. Остатки при вычислении отбрасываются всегда, кроме случая деления по модулю. Остаток записывается в результат.

Переменная должна быть определена с помощью оператора:

< имя переменной > VARIABLE < выражение >.

Например,

MAP1 VARIABLE 2#Q1–(Q1+Q2+10)/P1.

В поле метки пишется имя переменной MAP1. В поле операции записано служебное слово VARIABLE, а в поле операндов – записано арифметическое выражение, по которому вычисляется значение переменной. В данном случае текущие значения очередей Q1 и Q2 складываются, прибавляется 10 и полученное значение нацело делится на текущее значение первого параметра транзакта. Полученная величина вычитается из удвоенного значения очереди Q1. В результате получается целое число. Отбрасывание остатков производится в процессе вычисления.

Для получения более точного результата используется оператор FVARIABLE.

При вычислении промежуточный результат может быть вещественным, однако конечный результат округляется до целого числа.

Например,

S48 FVARIABLE 70*FN$XPDIS

8.16. Сохраняемые величины

Транзакты не могут передавать друг другу свои параметры, к тому же не существует прямого способа распечатки переменных. Для устранения этих недостатков используются ячейки памяти с установленными значениями. К этим ячейкам можно обращаться из любой точки модели. Такие ячейки называются сохраняемыми величинами, которые обозначаются:

Xj (X$< имя ячейки памяти >).

Оператор INITIAL используется для задания начальных значений ячейкам памяти. Перед началом моделирования интерпретатор GPSS сбрасывает в ноль все ячейки памяти. Затем для устанавления новых значений сохраняемых величин используется оператор INITIAL.

Формат оператора: INITIAL < A > [< B >]

Назначение параметров:

< A > – имя ячейки памяти, которое может выражаться следующим образом:

X< число >, X$< имя > – скалярные числа;

MX< число (n,m) >, MX$ < имя (n,m) > – служат для задания начальных значений в ячейках памяти матрицы;

< B > – сохраняемое числовое значение.

Имя матрицы может быть либо числом, либо символом.

Например,

INITIAL MX$MATR2 (2,4),–33; элемент матрицы с именем MATR2 с координатами (2,4), где 2 – строка, 4 – столбец, принимает значение – 33.

Оператор SAVEVALUE используется для записи значений переменных в ячейки памяти (рис. 36).

Формат оператора: SAVEVALUE < A >,< B >

Назначение параметров:

< A > – имя ячейки памяти (номер либо символьное имя);

< B > – значение, которое сохраняется в ячейках памяти.

Например:

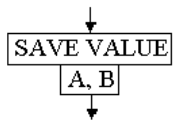


Рис. 36. УГО оператора SAVEVALUE

SAVEVALUE 10,V$ALFA; в 10-й ячейке памяти сохраняется значение переменной ALFA.

SAVEVALUE BETA,Q$AB12; в ячейке памяти с символьным именем BETA сохраняется длина очереди (текущее значение) с именем AB12.

Возможно использование оператора SAVEVALUE с суммированием и вычитанием.

Например:

SAVEVALUE 10+,V$ALFA; к текущему содержимому 10-й ячейки прибавляется текущее значение переменной ALFA.

SAVEVALUE 12–,V$PSI; из содержимого 12-й ячейки вычитается значение переменной PSI.

Примечание. Если в программе имеется оператор RESET, то содержимое сохраняемых величин не изменяется; если имеется оператор CLEAR, то содержимое всех сохраняемых величин сбрасывается в ноль.

Пример использования арифметических выражений. Смоделируем процесс сборки изображений на ПЭВМ. Время сборки изображения 30 ± 5 мин. Время обработки 8 ± 2 мин. Зарплата оператора 30 р. в день. Стоимость аппаратно-программного обеспечения, пересчитанного на один день, 80 р./день. Чистая стоимость изображения 5 р. Требуется организовать выдачу информации о средней дневной прибыли. При этом необходимо сделать с помощью сохраняемых величин число моделируемых дней переменным. В одном прогоне модели исследовать случаи с 4-мя, 5-ю и 6-ю операторами.

В блоке GENERATE сегмента таймера в качестве оператора A используется сохраняемая величина с именем TIMER. Средняя дневная прибыль рассчитывается по формуле

$$П = 5 * \text{Число изображений в день} - \text{Расходы},$$

где

$$\text{Расходы} = \text{Стоимость аппаратно-программного обеспечения} + \text{Зарплата}.$$

Количество изображений будем определять по количеству транзактов, которые прошли через блок RELEASE OTO, имеющий метку PLAN. Количество обработанных изображений определится количеством входов в транзакт N$PLAN.

Обозначим через переменную KDNI количество рабочих дней за моделируемый период.

Средняя дневная выручка рассчитывается при использовании выражения

5#N$PLAN/KDNI.

Дневная зарплата работников

30#X$RAB.

Тогда прибыль вычисляется с помощью выражения

PRIB=5#N$PLAN/KDNI–80–30#X$RAB.

В сегмент таймера введем сохраняемую величину INDEX для занесения в нее значения дневной прибыли.

Программа имеет вид:

```
SIMULATE
INITIAL X$RAB,4
INITIAL X$TIMER,2400; запись в ячейку памяти числа 2400 мин
KDNI VARIABLE X$TIMER/480; количество дней работы PRIB
VARIABLE 5#N$PLAN/KDNI–80–30#X$RAB
GENERATE ,,,X$RAB; генерация транзактов, количество которых равно содержимому ячейки X$RAB
SBOR ADVANCE 30,5
SEIZE OTO; занятие устройства с именем OTO
ADVANCE 8,2
PLAN RELEASE OTO
TRANSFER, SBOR
;
GENERATE X$TIMER
SAVEVALUE INDEX,V$PRIB; запись в ячейку памяти переменной
TERMINATE 1
START 1; моделирование для 5-ти операндов
CLEAR
INITIAL X$RAB,5
INITIAL X$TIMER,2400
START 1; моделирование для 6-ти операндов
CLEAR
INITIAL X$RAB,6
INITIAL X$TIMER,2400
START 1
```

Матричные сохраняемые переменные. Язык GPSS поддерживает матричную форму сохраняемой переменной.

Для того чтобы воспользоваться матрицами, необходимо использовать оператор MATRIX.

Формат оператора: < NAME > MATRIX < A >,< B >,< C >.

В поле метки < NAME > записывается имя матрицы, в поле операнда < A > ставится «,», а в полях < B > и < C > – соответственно число строк и число столбцов.

Например,

PRM1 MATRIX ,3,2; определена матричная переменная PRM1 с числом строк 3 и числом столбцов 2.

Оператор INITIAL используется для записи элементов матрицы.

Например, пусть нужно определить матрицу с именем 1, размером 3×3, элементы которой равны:

2 5 6
3 4 8
1 0 9

Тогда

```
1  MATRIX  ,3,3
INITIAL MX1(1,1),2/MX1(1,2),5/MX1(1,3),6/
MX1(2,1),3/MX1(2,2),4/MX1(2,3)8/
MX1(3,1),1/MX1(3,2),0/MX1(3,3),9,
```

где MX – СЧА матричной переменной.

Формат обращения имеет вид MX< число >. Если имя матрицы символ, тогда формат обращения иммет вид MX$< имя >. Для сохранения информации в матричных ячейках памяти применяется блок вида:

MSAVEVALUE < A >,< B >,< C >,< D >,

где < A > – имя матрицы;

< B > – номер строки;

< C > – номер столбца;

< D > – значение элемента матрицы.

Например,

MSAVEVALUE MATR 2,2,3,–100

Элементы матрицы могут использоваться в качестве операндов в блоках: ADVANCE MX$MATR12(1,2),P5. Осуществляется задержка транзакта на среднее время, определяемое элементом матрицы MATR12 с координатами (1,2). При этом доверительный интервал находится в 5-м параметре текущего транзакта (P5).

Примечание. Оператор RESET не изменяет матричную величину, а оператор CLEAR обнуляет все матрицы.

8.17. Организация ветвлений транзактов в модели

Блок TEST. Этот блок используется для изменения направления движения транзактов в модели, осуществляющий проверку числовых выражений (рис. 37).

Формат блока: TEST < X > < A >, < B >,< C >.

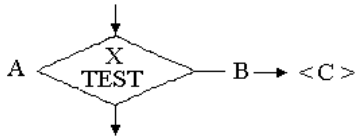


Рис. 37. УГО блока TEST

Блок проверяет некоторое условие X для операндов A и B. Если это условие ложно, то транзакт направляется в блок с именем, записанным в параметре < C >. Если условие истинно, то транзакт направляется в следующий блок.

Блок TEST может работать в двух режимах:

1) без параметра < C >:

TEST < X > < A >, < B >

Здесь сравниваются значения операторов < A > и < B > по условию < X >. Если условие ложно, то транзакт задерживается в данном блоке TEST, а если условие истинно, то транзакт поступает в следующий блок.

Значения условия < X >:

G – больше; LE – меньше или равно;
GE – больше или равно; L – меньше;
E – равно; NE – не равно.

Например,

TEST E Q$OTO,5; проверяется по условию равно ли содержимое очереди OTO непосредственно величине операнда 5. Если условие ложно, то транзакт задерживается, иначе он идет в следующий блок;

2) в режиме условной передачи транзакта в блок с именем, находящимся в параметре < C >.

Например,

TEST LE P1,Q1,ALFA; если содержимое 1-го параметра транзакта меньше или равно величине очереди 1, то транзакт проходит в следующий по порядку блок. Иначе транзакт поступает на вход блока, имеющего имя ALFA.

Моделирование цикла. При моделировании цикла производится уменьшение некоторых параметров или атрибутов на единицу и проверяется условие на нулевой результат. Если условие ложно, то цикл повторяется и транзакт направляется к началу цикла. Если счетчик обнуляется, то транзакт переводится в следующий по порядку блок.

Например:

```
MET22  ENTER  DRIVER
  :
ASSIGN 22–,1
TEST E P22,0,MET22
```

После каждого прохождения цикла содержимое 22-го параметра текущего транзакта уменьшается на 1. В блоке TEST проверяется на равенство нулю значения параметра 22-го текущего транзакта.

Блок LOOP. Он используется для организации цикла.

Формат блока: LOOP < A >,< B >

Назначение параметров:

< A > – номер параметра транзакта, в который должно быть занесено необходимое количество повторений (циклов);

< B > – имя блока, куда передается транзакт, если цикл не окончен.

Например,

LOOP P2,PRR34; проверить сдержание 2-го параметра текущего транзакта, и если оно не равно 0, то транзакт передается в блок с именем PRR34.

8.18. Моделирование логического управления

Для моделирования логического управления используются логические переключатели. Они имеют два устойчивых положения «выключено» и «включено». В начале моделирования все логические переключатели находятся в положении «сброшено» («выключено»).

Оператор INITIAL используется для перевода некоторых логических переключателей в начале моделирования в положение «установлено» («включено»).

Формат блока: INITIAL < LSj >

В поле < LSj > указывется стандартный числовой атрибут логического переключателя (j может быть номером или символическим именем).

Например,

INITIAL LS232, INITIAL LS$RP1

П р и м е ч а н и е. Оператор RESET не действует на переключатели, а оператор CLEAR переводит их в состояние «сброшено».

8.18.1. Блок LOGIC

Блок LOGIC предназначен для установки логического переключателя в заданное состояние в любой точке программы (рис. 38)

Формат блока: LOGIC < X > < A >

Назначение параметров:

< X > – логическое состояние ключа. Условие < X > может принимать значения:

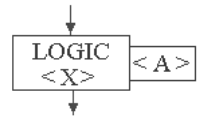


Рис. 38. УГО блока LOGIC

S – установка ключа в состояние «установлено»,

R – установка ключа в состояние «сброшено»,

I – инвертировать состояние ключа,

< A > – операнд для задания имени логического переключателя.

Изображение блока LOGIC на структурных схемах:

Например,

S22 LOGIC R KL1; блок имеет имя в модели S22 и устанавливает в положение «сброшено» логический переключатель KL1.

8.18.2. Блок GATE

Блок GATE предназначен для проверки состояния логических переменных и для организации ветвлений в программе (рис. 39).

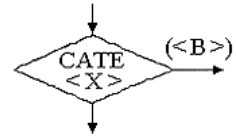


Рис. 39. УГО блока GATE

Формат блока: GATE < X > < A >, [< B >]

Назначение параметров:

< X > – записывается логическое условие, на которое проверяется логический переключатель с именем, находящимся в операнде < A >.

< B > – может задаваться имя блока, в который будет направляться транзакт, если условие < X > ложно. Если в результате проверки это условие оказалось истинно, то транзакт поступает в следующий блок. Если < B > опущен, то реализуется режим «отказа». В этом случае, если условие < X > ложно, то транзакт задерживается блоком GATE до тех пор, пока условие не станет истинно.

Например:

GATE LR ALF1; проверяется, находится ли переключатель ALF1 в «сброшенном» состоянии.

MET1 GATE LS 2; проверка ключа 2 по условию «включено».

MET33 GATE LR BETA,MET1; если логический переключатель BETA сброшен, то транзакт идет в следующий по порядку блок, а если BETA установлен, то транзакт направляется в блок с именем MET1.

Блок можно применять для проверки логических условий, связанных с состоянием устройств или многоканальных устройств.

В этом случае операнд < X > принимает следующие значения:

1) для устройств:

U – проверка по условию занятости устройства;

NU – проверка по условию незанятости устройства;

2) для многоканальных устройств:

SF – проверка занятости многоканального устройства;

SNF – проверка незанятости многоканального устройства;

SE – проверка на пустоту многоканального устройства;

SNE – проверка на непустоту многоканального устройства.

Пример моделирования работы локальной вычислительной сети (ЛВС). Интервалы прибытия заявок соответствуют указанным в табл. 12.

Таблица 12

Интервалы времени прибытия заявок

Интервал, с	<0	100	200	300	>400
Суммарная частота	0,0	0,2	0,48	0,69	1,0

Интервалы времени обслуживания заявок соответствуют приведенным табл. 13.

Таблица 13

Интервалы времени обслуживания заявки

Интервал, с	<100	200	300	400	>500
Суммарная частота	0,0	0,06	0,21	0,48	1,0

Заявка принимается на обслуживание, если число ожидаемых заявок меньше числа обрабатываемых или равно ему. ЛВС работает по 12 ч в сутки (43 200 с). Все заявки, попавшие за 12 ч на станцию, должны быть обслужены. Условная прибыль с заявки 1 р. На каждой ПЭВМ работает один оператор с заработком 30 руб. в день. Заработок другого персонала, отнесенного к этой ПЭВМ – 75 р. в день. Определить, сколько должно быть ПЭВМ в ЛВС для получения максимальной прибыли.

1) Смоделируем работу ЛВС в течении 5 дней. Условия во все дни одинаковы. При возникновении временных неувязок завершение обслуживания должно иметь более высокий приоритет, чем прибывающие заявки. Прибывшие в момент закрытия ЛВС заявки имеют приоритет выше, чем приоритет закрытия ЛВС.

Закрытие ЛВС моделируется блоком GATE. Для этого на входе системы используется блок GATE LR ABC,ALFA, который прверяет состояние логического переключателя ABC по условию «сброшено». Если условие «ложно», т. е. переключатель установлен, то транзакт поступает в блок ALFA.

В сегменте таймера используется блок LOGIC S ABC. Этот блок, при входе в него транзакта, устанавливает переключатель ABC (условие закрытия ЛВС) в состояние «включено».

2) Для обслуживания всех заявок, попавших на станцию, в сегменте таймера установим блок TEST, который будет работать в режиме «отказа». Он сравнивает количество заявок, присоединившихся к очереди, с количеством обрабатываемых заявок.

Для развязки временных неувязок назначим различные приоритеты транзактам:

а) прибывающим заявкам – 1-й приоритет;

б) обслуживаемым заявкам – 2-й приоритет;

в) транзактам, генерируемым в сегменте таймера, – 0-й приоритет.

3) Обеспечение одинаковых условий при моделировании в различные сутки осуществляется с помощью блока PMULT < A >, < B >, < C >, < D >, < E >, < F >, < G >, где в операндах < A >, < B >, < C >, < D >, < E >, < F >, < G > указываются любые нечетные целые числа до трех знаков.

Блоком PMULT эти числа задают начальные значения генератору случайных величин.

PMULT 121,,331

Этим блоком генераторы случайных чисел 1 и 3 в начале моделирования будут вырабатывать числа 121 и 331 соответственно.

4) Обозначим переменную для вычисления прибыли NET:

NET VARIABLE SC1–75–30#R1,

где SC1 – СЧА для определения числа входов в многоканальное устройство 1;

R1 – доступная емкость многоканального устройства 1.

Текст программы:

```
 SIMULATE
 PMULT   111
 IAT FUNCTION  RN1,C5
 0,0/.2,100/.48,200/.69,300/1,400
 STIME FUNCTION  RN1,C5
 0,100/.06,200/.21,300/.48,400/1,500
```

```
SS1  STORAGE 1; число ПЭВМ в ЛВС
NET VARIABLE  SC$SS1–75–30#R$SS1;  дневная прибыль
    GENERATE  FN$IAT
    GATE  LR ABC,ALFA; проверка открытия ЛВС
    TEST  LE Q$SS1,S$SS1,ALFA; проверка величины очереди на
обслуживание
MET1  QUEUE   SS1
    ENTER   SS1
    DEPART  SS1
    PRIORITY  2; приоритет 2 заявки при обслуживании
    ADVANCE  FN$STIME; обслуживание заявки
MET2  LEAVE   SS1
    SAVEVALUE 2+,V$NET
ALFA  TERMINATE
; сегмент таймера
    GENERATE  43200
    LOGIC   S ABC;  закрытие ЛВС
    TEST  E N$MET1,N$MET2
    SAVEVALUE 1, V$NET
    TERMINATE 1
    PLOT X2,50,0,45000; построение гистограмм процесса
    PLOT X1,200,0,45000
    START   1
; второй день моделирования с одной ПЭВМ
    PMULT   333
    CLEAR
    START   1
    ⋮
; моделирование 5-го дня работы с одной ПЭВМ
    PMULT   999
    CLEAR
    START   1
; моделирование первого дня работы с 2-мя ПЭВМ
    PMULT   111
SS1  STORAGE 2
    CLEAR
    START   1
    ⋮
```



```
; моделирование 5-го дня с двумя ПЭВМ
    PMULT   999
    CLEAR
    START   1
    ⋮
; моделирование 1-го дня с 3-мя ПЭВМ
    PMULT   111
    CLEAR
S1  STORAGE 3
    START   1
    ⋮
    PMULT   999
    CLEAR
    START   1
```

8.18.3. Булевские переменные

Для операций с булевскими переменными в GPSS вводится СЧА – BVj (j – номер либо символьное имя). Булевская переменная принимает значение равное 1, если значение выражения истинно, в противном случае – 0.

Для описания булевских переменных применяются операторы 3-х типов:

– логические;

– булевские;

– операторы отношения.

Логические применяются для ссылок на логическое состояние в устройствах, в многоканальных устройствах и в логических ключах.

Для ключей используются СЧА: LSj, LRj.

Устройства описываются СЧА Fj. Если устройство занято, то Fj = 1, иначе Fj = 0. СЧА FVj – доступность устройства. Если FVj = 1, то устройство доступно. СЧА FIj – определение нахождения устройства в состоянии прерывания.

Для многоканальных устройств:

SFj – заполненность многоканального устройства (SF = 1, если многоканальное устройство полностью заполнено).

SEj – проверка на пустоту многоканального устройства (SE = 1, если многоканальное устройство пусто).

SVj – проверка на непустоту (SV = 1, если устройство доступно).

Например,

STO1 BVARIABLE SF$PRI22; определение булевской переменной STO1, которая определяет заполненность многоканального устройства PRI22.

Операторы отношения. Операторы отношения определяют условия, которые могут существовать между двумя численными величинами. Операторы отношения производят алгебраическое сравнение операндов путем следующих условий:

G – больше; LE – меньше или равно;
GE – больше или равно; L – меньше;
E – равно; NE – не равно.

Например:

ATEST BVARIABLE V$FIXED 'G' 5; определена переменная ATEST, которая вычисляет отношение переменной с символьным именем FIXED и числа 5. В том случае, если значение V$FIXED > 5, то ATEST = 1, а если V$FIXED = 5, то ATEST = 0.

BTEST BVARIABLE FN3 'LE' P4; описание булевской переменной BTEST, которая принимает значение 1, если вычисленное значение функции 3 ≤ содержимого параметра P4 текущего транзакта.

Булевские операторы. Для булевских операторов существуют следующие условия:

AND– «и»
OR – «или»
NOT – «инверсия»

При обработке булевских переменных выражения в скобках вычисляются в первую очередь.

Например,

ALFA BVARIABLE (Q1 'GE' S1) 'OR' (N$ABC 'LE' N$BETA)

Здесь описана булевская переменная ALFA, которая может принимать значение 1, либо 0. Значение 1 она принимает, если выражение Q1 'GE' S1 или выражение N$ABC 'LE' N$BETA истинно. Выражение Q1 'GE' S1 будет истинно, если очередь устройства 1 будет больше емкости многоканального устройства 1. Выражение N$ABC 'LE' N$BETA будет истинно, если число входов в блок ABC будет меньше числа входов в блок BETA.

П р и м е ч а н и е. В том случае, если булевская переменная представляет собой простое арифметическое выражение, значение этой переменной равно 0 тогда, когда арифметическое выражение равно 0. В противном случае булевская переменная равна 1.

Например:

PST1 BVARIABLE V$PRT2+10
PST1=0; если V$PRT1 + 10 = 0
PST=1; если V$PRT + 10 не равно 0.

Для проверки булевских переменных используется блок TEST.

В качестве операндов, с которыми сравнивают булевскую переменную, выступает 0 или 1.

Например:

TEST E BV$PRR12,1
B22 TEST E BV$PRR12,0

8.19. Блок SPLIT

Блок выполняет копирование входящего в него транзакта, который называется исходным (порождающим). В результате выполнения блока порождается семейство транзактов (рис. 40).

Формат блока: SPLIT < A >, [< B >], [< C >]

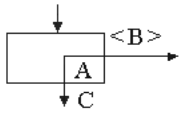


Рис. 40. УГО блока SPLIT

Назначение параметров:

< A > – число создаваемых копий транзактов (может быть просто числом, либо СЧА, либо именем, либо арифметическим выражением). Если вычисленное значение A не равно 0, то блок SPLIT выполняет копирование. После создания копий исходное сообщение или транзакт переходит к следующему по порядку блоку;

< B > – номер блока, к которому переходят копии исходного сообщения. Операнд < B > вычисляется для каждой копии транзакта отдельно (может организовать ветвление);

< C > – номер параметра, используемого для присвоения копиям последовательных номеров.

Например,

SPLIT 3,AL1; создается 3 копии исходного транзакта, которые затем поступят на вход блока с именем AL1. Если исходный транзакт имеет параметр (до входа в блок SPLIT) и содержимое этого параметра до входа в блок SPLIT было равно 0, тогда после выполнения блока SPLIT значение параметра исходного транзакта будет равно 1, а в транзактах-копиях значение параметра будет соответственно равно 2 (1-я копия), 3 (2-я копия), 4 (3-я копия). Для того чтобы направить поток транзактов по заданным адресам, можно использовать функцию с именем, указанным в операнде B блока SPLIT.

Например:

GAM FUNCTION P5,D3; функция определена на 3-х дискретных диапазонах.

2,BLOK1/3,BLOK2/4,BLOK3; в качестве значений функций использованы имена блоков.

SPLIT 3,FN$GAM, 5; создаются 3 копии транзакта с параметром 5, в котором записывается номер копии. Направление передачи копий определяется значением функции GAM.

8.20. Цепи пользователя

Для ускорения работы модели, если имеется много блоков типа SEIZE, ENTER, TEST, GATE, вводятся так называемые цепи пользователя, куда переводятся транзакты из цепи текущих событий. После снятия блокировки эти транзакты выводятся из цепи пользователя в цепь текущих событий модели.

Блок LINK. Этот блок вводит транзакт в цепь пользователя (рис. 41).

Формат блока: LINK < A >, < B >, [< C >]

Назначение параметров:

< A > – имя цепи (списка) пользователя;

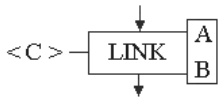


Рис. 41. УГО блока LINK

< B > – порядок ввода транзакта в цепь пользователя. <B> может принимать значения:

FIFO – транзакт помещается в конец цепи пользователя;

LIFO – транзакт помещается в начало цепи пользователя;

Pj – транзакты располагаются в списке в соответствии со значением указанного параметра Pj (по возрастанию);

< C > – имя блока, куда попадает транзакт, если он не присоединился к цепи пользователя.

Если присутствует < C >, то не все транзакты, вошедшие в блок LINK, попадут в цепь пользователя. Выбор между цепью и блоком, находящимся в < C >, осуществляется интерпретатором автоматически в результате анализа положения индекатора цепи пользователя. Если при входе в блок LINK индикатор выключен, то транзакт включает индикатор и направляется по адресу, указанному в операнде < C >. Если при входе в блок LINK индикатор включен, то транзакт направляется в цепь пользователя.

Блок UNLINK. Извлекает транзакт из списка пользователя (рис. 42).

Формат блока: UNLINK [< X >] < A >, < B >, [< C >], [< D >], [< E >], [< F >]

Назначение параметров:

< A > – имя цепи пользователя;

< B > – имя блока, в который направляются выведенные из списка пользователя транзакты;

< C > – задает число выводимых транзактов (ALL – все транзакты);

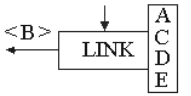


Рис. 42. УГО блока UNLINK

< E > – СЧА, значение которого сравнивается со значением параметров сообщения в списке пользователя;

< D > – номер параметра транзакта, содержимое которого сравнивается с содержимым операнда < E >;

< F > – имя следующего блока для входящих в блок UNLINK транзактов в случае, если:

1) список пуст;

2) не выполнено условие X при сравнении операндов < D > и < E >;

3) D = 0.

Если операторы < D > и < E > опущены, то вывод происходит из начала цепи. При окончании выборки транзактов из списка пользователя интерпретатор выключает индикатор цепи пользователя, снимая тем самым блокировку.

Например:

```
   GENERATE  10,5
   LINK  PR22,FIFO,LIPS;
   LIPS  SEIZE  OTO
   ADVANCE 8,2
   RELEASE OTO
   UNLINK  PR22,LIPS,1
   TERMINATE
   :
```

Блоком LINK создается цепь пользователя с именем PR22. Порядок обслуживания – FIFO (в конец цепи), имя блока, в который направляются транзакты, не присоединившиеся к цепи, – LIPS.

П р и м е ч а н и е. При входе транзактов в блок LINK включается индикатор цепи, блокируя дальнейшее продвижение транзакта. Пока он включен все транзакты поступают в цепь пользователя.

8.21. Цепи прерывания

При моделировании систем часто возникает потребность прервать обслуживание некоторого транзакта в устройстве в случае прихода другого транзакта с более высоким приоритетом. Тогда происходит захват устройства другим транзактом. После завершения обслуживания транзакта с высшим приоритетом, транзакт с низшим приоритетом дообслуживается данным устройством.

Блоки PREEMPT и RETURN используются для реализации этого режима (рис. 43).

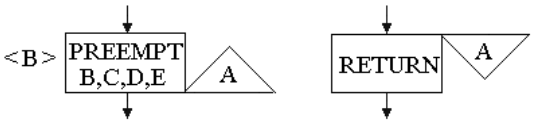


Рис. 43. УГО блоков PREEMPT и RETURN

Формат блоков: PREEMPT <A>, [<B>], [<C>], [<D>], [<E>];
RETURN <A>

Назначение параметров:

<A> – имя устройства, которое нужно захватить;

<B> – условие захвата;

B = PR – захват по приоритету. При захвате по приоритету прерванный транзакт помещается во внутреннюю цепь прерываний и находится там до тех пор, пока не освободится захваченное устройство;

<C> – имя блока, в который пересылается прерванный транзакт;

<D> – номер параметра прерванного транзакта, в котором заносится время его дообслуживания;

<E> – право на дообслуживание. По умолчанию дообслуживание требуется. Если E = RE – дообслуживание не требуется.

Примечание. Если используется операнд <E>, то игнорируются операнды <C> и <D>.

Блок RETURN предназначен для посылки на доработку прерванных транзактов в том случае, если транзакт входит в этот блок. Устройство может быть освобождено только тем транзактом, которым оно было захвачено.

8.22. Блок ASSEMBLE

Блок объединяет несколько транзактов из одного семейства в одно сообщение, которое поступает в следующий блок. Блок ASSEMBLE накапливает заданное число транзактов и после этого на его выходе генерируется один транзакт (рис. 44).

Формат блока: ASSEMBLE <A>

В операнде <A> указывется число соединяемых транзактов.

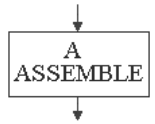


Рис. 44. УГО блока ASSEMBLE

Например:

```
SEIZE  CPU
SPLIT 1,PCA2; генерация семейства транзактов с двумя транзактами: оригинал + копия
ADVANCE FN$TIM1; задержка на обработку транзакта оригинала (копии идут на блок PCA2)
RELEASE CPU
PAE1 ASSEMBLE 2; объединение 2-х транзактов в один
TERMINATE
PCA2 SEIZE  ARM; занятие устройства для обработки копий
ADVANCE FN$TIME2
SEIZE  CHAN
ADVANCE FN$TIME3
RELEASE CHAN
RELEASE ARM
TRANSFER ,PAE1
```

8.23. Блок GATHER

Блок накапливает заданное количество сообщений, принадлежащих к одному семейству, но не объединяет их в один транзакт (рис. 45). Когда в блоке GATHER накопится заданное количество сообщений, все эти сообщения одновременно поступают на вход следующего по порядку блока. В одном блоке GATHER может накапливаться несколько семейств транзактов.

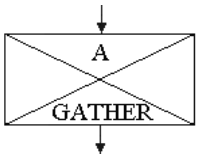


Рис. 45. УГО блока GATHER

Формат блока: GATHER <A>

В операнде <A> указывется число накапливаемых транзактов (A > 1).

Транзакты, накапливаемые в блоке, удаляются из списка текущих событий и переходят в состояние синхронизации.

8.24. Блок MATCH

Блок MATCH используется для синхронизации движения 2-х транзактов, принадлежащих одному семейству. Блок MATCH не объединяет синхронизируемые сообщения. Синхронизация осуществляется путем подбора пар сообщений из одного семейства и задержки этих сообщений до тех пор, пока оба сообщения из одной пары не поступят в заданные точки модели.

Формат блока: MATCH < A >

В операнде < A > указывется имя другого блока MATCH, называемого сопряженным.

Для синхронизации пар транзактов всегда надо использовать два блока MATCH.

Например:

```
:
ABC1  MATCH  AF1
:
AF1 MATCH  ABC1
:
```

Когда транзакт из одного семейства поступает в блок ABC1, интерпретатор проверяет, есть ли в сопряженном блоке AF1 второй транзакт из этого семейства. Если такового там нет, то первый транзакт переводится в цепь синхронизации и ожидает прибытия в блок MATCH второго транзакта из этого же семейства с именем AF1.

9. СЕРВЕРНОЕ ПРОГРАММИРОВАНИЕ НА ОСНОВЕ ЯЗЫКА PHP

Язык PHP был разработан как инструмент для решения чисто практических задач. Его создатель, Расмус Лердорф, хотел знать, сколько людей читают его online-резюме, и написал для этого простенькую CGI-оболочку на языке Perl, т. е. это был набор Perl-скриптов, предназначенных исключительно для определенной цели – сбора статистики посещений.

PHP способен решать те же задачи, что и любые другие CGI-скрипты, в том числе обрабатывать данные HTML-форм, динамически генерировать HTML-страницы и т. п. Но есть и другие области, где может использоваться PHP. Всего выделяют три основные области применения PHP.

Первая область. Это создание приложений (скриптов), которые исполняются на стороне сервера. PHP наиболее широко используется именно для создания такого рода скриптов. Для того чтобы работать таким образом, понадобится PHP-парсер (т. е. обработчик PHP-скриптов), и Web-сервер для обработки скрипта, браузер для просмотра результатов работы скрипта и, конечно, какой-либо текстовый редактор для написания самого PHP-кода. Парсер PHP распространяется в виде CGI-программы или серверного модуля.

Вторая область. Это создание скриптов, выполняющихся в командной строке. Иными словами, с помощью PHP можно создавать такие скрипты, которые будут исполняться, вне зависимости от web-сервера и браузера, на конкретной машине. Для такой работы потребуется лишь парсер PHP (в этом случае его называют интерпретатором командной строки (Command Line Interpreter – CLI). Этот способ работы подходит, например, для скриптов, которые должны выполняться регулярно с помощью различных планировщиков задач или для решения задач простой обработки текста.

Третья область. Это создание GUI-приложений (графических интерфейсов), выполняющихся на стороне клиента. В принципе это не самый лучший способ использовать PHP, особенно для начинающих, но если вы уже досконально изучили PHP, то такие возможности языка могут оказаться весьма полезны. Для применения PHP в этой области потребуется специальный инструмент – PHP-GTK, который является расширением PHP.

В PHP сочетаются две самые популярные парадигмы программирования – объектная и процедурная. В PHP 4 более полно поддерживается процедурное программирование, но есть возможность писать программы и в объектном стиле. Уже в первых пробных версиях PHP 5 большинство недочетов в реализации объектно-ориентированной модели языка, существующих в PHP 4, устранены. Таким образом, можно выбрать наиболее привычный стиль работы.

Если говорить о возможностях сегодняшнего PHP, то они выходят далеко за рамки тех, что были реализованы в его первых версиях. С помощью PHP можно создавать изображения, PDF-файлы, флэш-ролики, в него включена поддержка большого числа современных баз данных, встроены функции для работы с текстовыми данными любых форматов, включая XML, и функции для работы с файловой системой. PHP поддерживает взаимодействие с различными сервисами посредством соответствующих протоколов, таких, как протокол управления доступом к директориям LDAP, протокол работы с сетевым оборудованием SNMP, протоколы передачи сообщений IMAP, NNTP и POP3, протокол передачи гипертекста HTTP и т. д.

Обращая внимание на взаимодействие между различными языками, следует упомянуть о поддержке объектов Java и возможности их использования в качестве объектов PHP. Для доступа к удаленным объектам можно использовать расширение CORBA.

Для работы с текстовой информацией PHP унаследовал (с небольшими изменениями) механизмы работы с регулярными выражениями из языка Perl и UNIX-систем. Для обработки XML-документов можно использовать как стандарты DOM и SAX, так и API для XSLT-трансформаций.

Для создания приложений электронной коммерции существует ряд полезных функций, таких, как функции осуществления платежей Cybercash, CyberMUT, VeriSign Payflow Pro и CCVS.

Установка Apache 1.3.29 под Windows XP. Чтобы установить какую-либо программу, нужно для начала иметь соответствующее программное обеспечение (ПО). Скачать ПО для установки Apache можно, например, с его официального сайта http://www.apache.org. Мы скачали файл apache_1.3.29-win3x86-no_src.exe. Это автоматический установщик (иначе – wizard) сервера Apache под Windows. Эта программа попытается почти самостоятельно (а точнее, с минимальными усилиями с вашей стороны) установить на компьютер какое-либо программное обеспечение, а в данном случае сервер. После запуска файла-установщика появляется диалоговое окно (рис. 46).

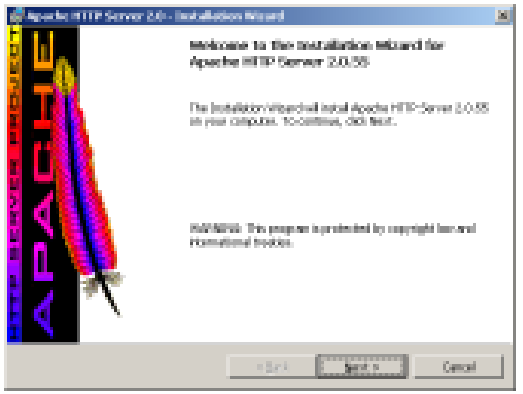


Рис. 46. Автоматическая установка сервера Apache

Чтобы установить HTTP-сервер Apache версии 1.3.29 на свой компьютер, нужно нажать на кнопку Next. Кстати говоря, эта же программа позволит изменить или удалить уже установленный Web-сервер.

После нажатия кнопки Next программа предложит согласиться с условиями лицензии (рис. 47).

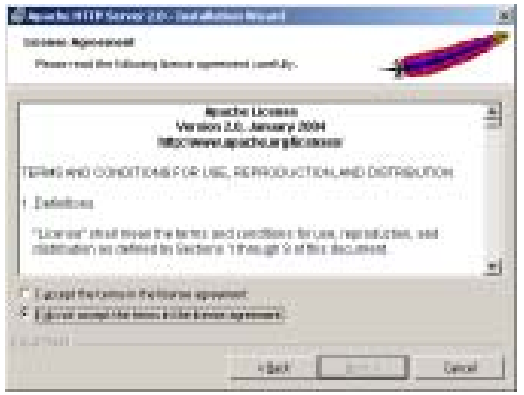


Рис. 47. Лицензионное соглашение

Следующий экран будет содержать информацию о сервере Apache, и в частности о его Windows-реализации (его изображение не приводим).

На следующем шаге нужно ввести имя сетевого домена, имя сервера и e-mail администратора. Программа попытается автоматически определить ваш домен и хост по настройкам компьютера (рис. 48).

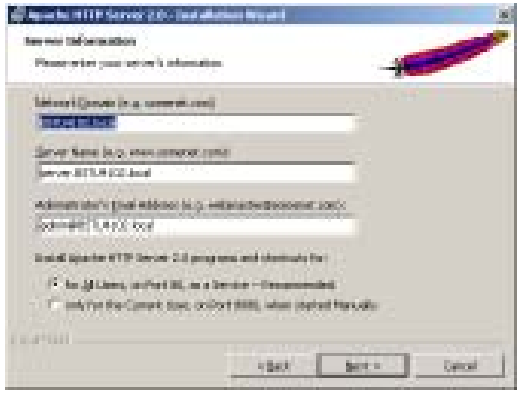


Рис. 48. Основная информация о сервере

После того как вы ввели данные в вышеприведенную форму, нужно выбрать тип установки: полная (устанавливаются все компоненты сервера) или определяемая пользователем (можно выбрать компоненты для установки) (рис. 49).

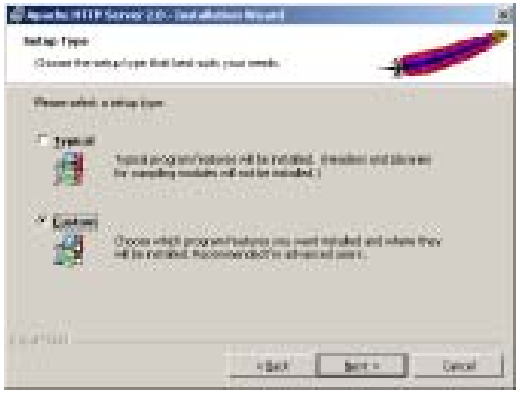


Рис. 49. Тип установки

Выбор компонентов сервера (инструменты, необходимых для работы сервера, и документация к нему) не очень большой (рис. 50).

Рис. 50. Выбор компонентов пользоватательской установки

Мы выберем полную установку. Тогда на следующем шаге программа предложит выбрать папку, в которую будет установлен сервер. По умолчанию сервер устанавливается в папку c:\Program Files\Apache Group\ (рис. 51).

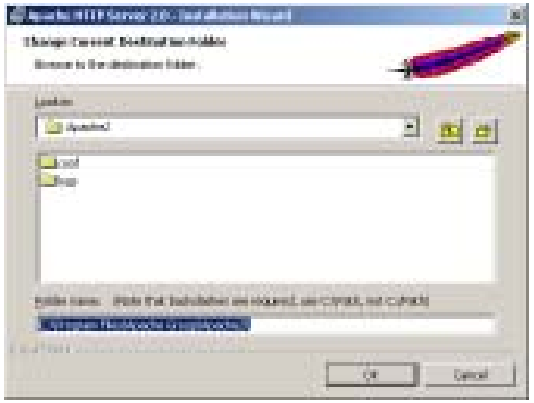


Рис. 51. Папка, в которую будет установлен сервер

В следующем окне (рис. 52) потребуется подтвердить правильность введенных данных и начать установку. Из любого окна установки, включая и это, можно вернуться назад и изменить введенные ранее данные.

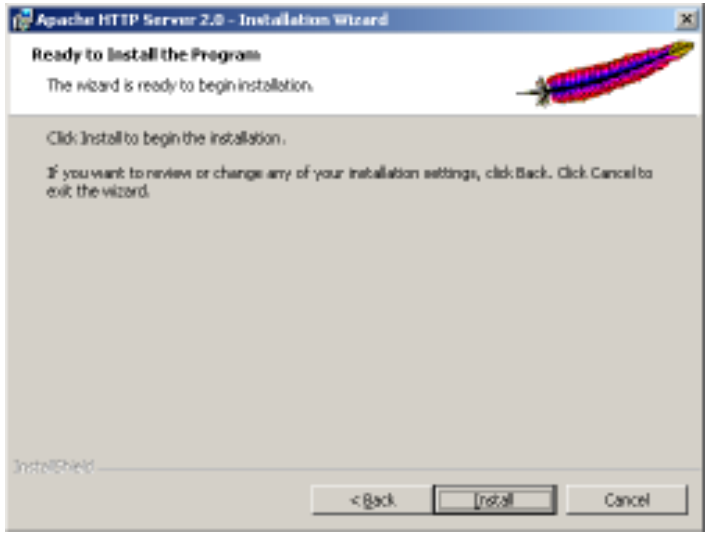


Рис. 52. Начало установки

После нажатия кнопки Install начнется непосредственная установка сервера. От пользователя никаких дополнительных действий больше не требуется. В целом это хорошо, но и у такой автоматизированной установки есть некоторые недостатки. Например, домашние директории пользователей оказываются там же, где и файлы настроек сервера (c:\Program Files\Apache Group\Apache\users\). Это небезопасно, если на компьютере работает несколько пользователей, не являющихся администраторами сервера. Но для начала можно ничего не менять. Допустим, мы запустили установщик, ввели все необходимые данные, он выполнил все операции успешно и говорит, что сервер установлен. Как проверить, действительно ли сервер установлен? Набираем в окне браузера http://localhost/. Если все установлено правильно, то на экран выводится страничка приветствия сервера Apache (рис. 53).

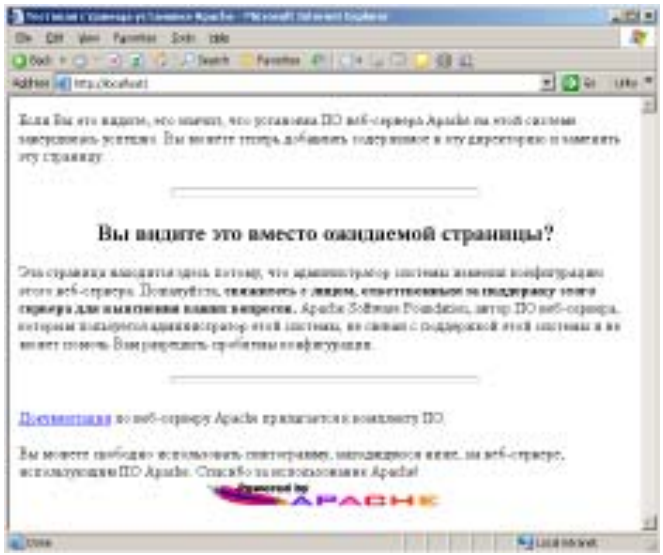


Рис. 53. Страница приветствия

Итак, сервер установлен. Как теперь с ним работать? Откуда можно запускать скрипты и где должны находиться файлы пользователей? Файлы, которые должны быть обработаны сервером, можно сохранять либо в корне сервера (в нашем случае это c:\Program Files\Apache Group\Apache\htdocs), либо в директориях пользователей (в нашем случае это c:\Program Files\Apache Group\Apache \users\). Местоположение корня сервера и директорий пользователей прописано в настройках сервера, а точнее, в файле конфигурации httpd.conf (найти его можно в c:\Program Files\Apache Group \Apache\conf). Для изменения этих путей нужно изменить соответствующие переменные в файле конфигурации сервера. Переменная в файле конфигурации ServerRoot отвечает за корневую директорию сервера, а переменная UserDir – за директорию, где будут располагаться файлы пользователей сервера (для более безопасной работы советуем изменить переменную UserDir на что-нибудь типа c:\users\). Чтобы получить доступ к файлу test.html, находящемуся в корне сервера, нужно набрать в браузере http://localhost/test.html (т. е. имя хоста, имя файла). Если же файл test.html находится в директории пользователя user, то для его просмотра нужно набрать в браузере http://localhost/~user/test.html.

9.1. Основной синтаксис

Первое, что нужно знать относительно синтаксиса PHP, – это то, как он встраивается в HTML-код, как интерпретатор узнает, что это код на языке PHP. В предыдущей лекции мы уже говорили об этом. Повторяться не будем, отметим только, что в примерах мы чаще всего будем использовать вариант <?php ?>, а иногда сокращенный вариант <? ?>.

Разделение конструкций. Программа на языке PHP (да и на любом другом языке программирования) – это набор команд (инструкций). Обработчику программы (парсеру) необходимо как-то отличать одну команду от другой. Для этого используются специальные символы – разделители. В PHP инструкции разделяются так же, как и в Си или Perl, – каждое выражение заканчивается точкой с запятой.

Закрывающий тег «?>» также подразумевает конец инструкции, поэтому перед ним точку с запятой не ставят.

Например, два следующих фрагмента кода эквивалентны:

```
<?php
echo "Hello, world!"; // точка с запятой
                // в конце команды
                // обязательна
?>
<?php
echo "Hello, world!" ?>
<!— точка с запятой
  опускается из-за "?>" —>
```

Комментарии. Часто при написании программ возникает необходимость делать какие-либо комментарии к коду, которые никак не влияют на сам код, а только поясняют его. Это важно при создании больших программ и в случае, если несколько человек работают над одной программой. При наличии комментариев в программе в ее коде разобраться гораздо проще. Кроме того, если решать задачу по частям, недоделанные части решения также удобно комментировать, чтобы не забыть о них в дальнейшем. Во всех языках программирования предусмотрена возможность включать комментарии в код программы. PHP поддерживает несколько видов комментариев: в стиле Си, C++ и оболочки UNIX. Символы «//» и «#» обозначают начало однострочных комментариев, «/*» и «*/» обозначают соответственно начало и конец многострочных комментариев.

Например,

```
<?php
echo "Меня зовут Вася";
  // это однострочный комментарий
  // в стиле C++
echo "Фамилия моя Петров";
/* это многострочный комментарий.
Здесь можно написать несколько строк.
При исполнении программы все, что
находится в пределах этих символов (закомментировано),
будет игнорироваться. */
echo "Я изучаю PHP в INTUIT.ru";
  # это комментарий в стиле
  # оболочки UNIX
?>
```

9.2. Переменные, константы

Важным элементом каждого языка являются переменные, константы и применяемые к ним операторы. Рассмотрим, как выделяются и обрабатываются эти элементы в PHP.

Переменные. Переменная в PHP обозначается знаком доллара, за которым следует ее имя.

Например: $my_var

Имя переменной чувствительно к регистру, т. е. переменные $my_var и $My_var различны.

Имена переменным задаются по тем же правилам, что и остальные наименования в PHP: правильное имя переменной должно начинаться с буквы или символа подчеркивания с последующими в любом количестве буквами, цифрами или символами подчеркивания.

В PHP 3 переменные всегда присваивались по значению. Иными словами, когда вы присваиваете выражение переменной, все значения оригинального выражения копируются в эту переменную. Это означает, к примеру, что после присвоения одной переменной значения другой изменение одной из них не влияет на значение другой.

Пример присваивания по значению (html, txt)

```
<?php
$first = "Text"; // присваиваем $first
```

```
                // значение
                //  "Text "
$second = $first; // присваиваем $second
                // значение
                // переменной $first
$first =  "New text"; // изменяем
                    // значение
                    // $first
                    // на  "New text"
echo "Переменая с именем first".
   "равна $first <br>";
     // выводим значение $first
echo "Переменая с именем second".
    "равна $second";
      // выводим значение $second
?>
```

Результат работы этого скрипта будет следующим:

Переменная с именем first равна New text.

Переменная с именем second равна Text.

Кроме этого, PHP 4 предлагает еще один способ присвоения значений переменным: присвоение по ссылке. Для того чтобы присвоить значение переменной по ссылке, это значение должно иметь имя, т. е. оно должно быть представлено какой-либо переменной. Чтобы указать, что значение одной переменной присваивается другой переменной по ссылке, нужно перед именем первой переменной поставить знак «&» (амперсанд).

Рассмотрим тот же пример, что и выше, только будем присваивать значение переменной first переменной second по ссылке.

Пример присваивания по ссылке (html, txt)

```
<?php
$first = 'Text'; // присваиваем $first
                // значение 'Text'
$second = &$first;
/* делаем ссылку на $first через $second.
  Теперь значения этих переменных
  будут всегда совпадать */
// изменим значение $first
// на ′New text′
$first = 'New text';
```



```
echo "Переменная с именем first".
   "равна $first <br>";
// выведем значения обеих переменных
echo "Переменная с именем second".
    "равна $second";
?>
```

Этот скрипт выведет следующее:

Переменная с именем first равна New text.

Переменная с именем second равна New text.

Таким образом, вместе с переменной $first изменилась и переменная $second.

Константы. Для хранения постоянных величин, т. е. таких величин, значение которых не меняется в ходе выполнения скрипта, используются константы. Такими величинами могут быть математические константы, пароли, пути к файлам и т. п. Основное отличие константы от переменной состоит в том, что ей нельзя присвоить значение больше одного раза и ее значение нельзя аннулировать после ее объявления. Кроме того, у константы нет приставки в виде знака доллара и ее нельзя определить простым присваиванием значения. Как же тогда можно определить константу? Для этого существует специальная функция define(). Ее синтаксис таков:

```
define("Имя_константы",
    "Значение_константы",
    [Нечувствительность_к_регистру])
```

По умолчанию имена констант чувствительны к регистру. Для каждой константы это можно изменить, указав в качестве значения аргумента «Нечувствительность_к_регистру» значение True. Существует соглашение, по которому имена констант всегда пишутся в верхнем регистре.

Получить значение константы можно указав ее имя. В отличие от переменных, не нужно предварять имя константы символом $. Кроме того, для получения значения константы можно использовать функцию constant() с именем константы в качестве параметра.

Пример определения константы в PHP (html, txt)

```
<?php
// определяем константу
// PASSWORD
define("PASSWORD","qwerty");
// определяем регистронезависимую
// константу PI со значением 3.14
```

```
define("PI","3.14", True);
// выведет значение константы PASSWORD,
// т. е. qwerty
echo (PASSWORD);
// тоже выведет qwerty
echo constant(″PASSWORD″);
echo  (password);
/* выведет password и предупреждение,
  поскольку мы ввели регистрозависимую
  константу PASSWORD */
// выведет 3.14, поскольку константа PI
// регистронезависима по определению
echo pi;
?>
```

Кроме переменных, объявляемых пользователем, о которых мы только что рассказали, в PHP существует ряд констант, определяемых самим интерпретатором. Например, константа _FILE_ хранит имя файла программы (и путь к нему), которая выполняется в данный момент, _FUNCTION_ содержит имя функции, _CLASS_ – имя класса, PHP_VERSION – версия интерпретатора PHP. Полный список предопределенных констант можно получить прочитав руководство по PHP.

9.3. Операторы

Операторы позволяют выполнять различные действия с переменными, константами и выражениями. Мы еще не упоминали о том, что такое выражение. Выражение можно определить как все то, что имеет значение. Переменные и константы – это основные и наиболее простые формы выражений. Существует множество операций (и соответствующих им операторов), которые можно производить с выражениями. Операторы языка GPSS сведены в табл. 14–19.

Таблица 14

Арифметические операторы

Обозначение	Название	Пример
+	Сложение	$a + $b
–	Вычитание	$a – $b
*	Умножение	$a * $b
/	Деление	$a / $b
%	Остаток от деления	$a % $b

Таблица 15

Операторы присваивания

Обозначение	Название	Описание	Пример
=	Приравнивание	Переменной слева от оператора будет присвоено значение, полученное в результате выполнения каких-либо операций, или переменной/константы с правой стороны	$a = ($b = 4) +5; ($a будет равна 9)
+=	Сокращение	Прибавляет к переменной число и затем присваивает ей полученное значение	$a += 5; (эквивалентно $a = = $a + 5;)
.=	Присваивание	Сокращенно обозначает комбинацию операций конкатенации и присваивания (сначала добавляется строка, потом полученная строка записывается в переменную)	$b = "Привет"; $b .= "всем"; (эквивалентно $b = = $b. "всем";) В результате: $b= "Привет всем"

Таблица 16

Строковые операторы

Обозначение	Название	Описание	Пример
.	Конкатенация (сложение строк)	$a. $b (это строка, состоящая из $a и $b)	$a. $b

Таблица 17

Логические операторы

Обозначение	Название	Описание	Пример
and	И	Истина, если $a и $b истинны	$a and $b
&&	И		$a && $b
or	Или	Хотя бы одна из переменных $a или $b истинна (возможно, что и обе)	$a or $b
||	Или	Логическая операция «или»	$a || $b
xor	Исключающее «или»	Одна из переменных истинна. Случай, когда они обе истинны, исключается	$a xor $b
!	Инверсия (NOT)	Если $a=True, то !$a=False, и наоборот	! $a

Таблица 18

Операторы сравнения

Обозначение	Название	Описание	Пример
==	Равенство	Значения переменных равны	$a == $b
===	Эквивалентность	Равны значения и типы переменных	$a = = = $b
!=	Неравенство	Значения переменных не равны	$a != $b
<>	Неравенство	Условие не равно	$a <> $b
!==	Неэквивалентность	Переменные не эквивалентны	$a !== $b
<	Меньше	Условие строго меньше	$a < $b
>	Больше	Условие строго больше	$a > $b
<=	Меньше или равно	Условие меньше или равно	$a <= $b
>=	Больше или равно	Условие больше или равно	$a >= $b

Таблица 19

Операторы инкремента и декремента

Обозначение	Название	Описание	Пример
++$a	Пре-инкремент	Увеличивает $a на единицу и возвращает $a	<? $a=4; echo "Должно быть 5:" . ++$a; ?>
$a++	Пост-инкремент	Возвращает $a, затем увеличивает $a на единицу	<? $a=4; echo "Должно быть 4:" .$a++; echo "Должно быть 5:" .$a; ?>
--$a	Пре-декремент	Уменьшает $a на единицу и возвращает $a	<? $a=4; echo "Должно быть 3:" . --$a; ?>
$a--	Пост-декремент	Возвращает $a, затем уменьшает $a на единицу	<? $a=4; echo "Должно быть 4:" . $a--; echo "Должно быть 5:" .$a; ?>

9.4. Типы данных

PHP поддерживает восемь простых типов данных.

Четыре скалярных типа:

– boolean (логический);

– integer (целый);

– float (с плавающей точкой);

– string (строковый).

Два смешанных типа:

– array (массив);

– object (объект).

И два специальных типа:

– resource (ресурс);

– NULL.

В PHP не принято явное объявление типов переменных. Предпочтительнее, чтобы это делал сам интерпретатор во время выполнения программы в зависимости от контекста, в котором используется переменная. Рассмотрим по порядку все перечисленные типы данных.

Тип boolean (булев, или логический тип). Этот простейший тип выражает истинность значения, т. е. переменная этого типа может иметь только два значения – истина True или ложь False.

Чтобы определить булев тип, используют ключевое слово True или False. Оба регистронезависимы. *Например*,

```
<?php
$test=True;
?>
```

Логические переменные используются в различных управляющих конструкциях (циклах, условиях и т. п.). Иметь логический тип, т. е. принимать только два значения, истину или ложь, могут также и некоторые операторы (например, оператор равенства). Они также используются в управляющих конструкциях для проверки каких-либо условий. Например, в условной конструкции проверяется истинность значения оператора или переменной, и в зависимости от результата проверки выполняются те или иные действия. Здесь условие может быть истинно или ложно, что как раз и отражает переменная и оператор логического типа.

Пример использования логического типа (html, txt)

```
<?php
// оператор '==' проверяет равенство
```

```
// и возвращает
// булево значение
if ($know == False) { // если $know
                      // имеет значение
                      // false
echo "Изучай PHP!";
}
if (!$know) { // то же самое, что
              // и выше, т. е. проверка
              // имеет ли $know значение
              // false
echo "Изучай PHP!";
}
/* оператор '==' проверяет, совпадает ли
  значение переменной $action со строкой
  "Изучить PHP". Если совпадает, то
  возвращает true, иначе – false.
  Если возвращено true, то выполняется
  то, что внутри фигурных скобок */
if ($action == "Изучить PHP")
{ echo "Начал изучать"; }
?>
```

Тип integer (целые). Этот тип задает число из множества целых чисел Z = {..., -2, -1, 0, 1, 2, ...}. Целые числа могут быть указаны в десятичной, шестнадцатеричной или восьмеричной системе исчисления, по желанию, с предшествующим знаком «-» или «+».

Если вы используете восьмеричную систему счисления, вы должны предварить число нулем, для использования шестнадцатеричной системы нужно поставить перед числом 0x.

Например,

```
<?php
# десятичное число
$a = 1234;
# отрицательное число
$a = -123;
# восьмеричное число (эквивалентно
# 83 в десятичной системе)
$a = 0123;
```



```
# шестнадцатеричное число (эквивалентно
# 26 в десятичной системе)
$a = 0x1A;
?>
```

Размер целого числа зависит от платформы, хотя, как правило, максимальное значение около двух миллиардов (это 32-битное знаковое число). Беззнаковые целые числа PHP не поддерживает.

Если вы определите число, превышающее пределы целого типа, оно будет интерпретировано как число с плавающей точкой. Также если вы используете оператор, результатом работы которого будет число, превышающее пределы целого, вместо него будет возвращено число с плавающей точкой.

В PHP не существует оператора деления целых. Результатом 1/2 будет число с плавающей точкой 0.5. Вы можете привести значение к целому, что всегда округляет его в меньшую сторону, либо использовать функцию round(), округляющую значение по стандартным правилам. Для приведения переменной к конкретному типу необходимо перед переменной указать в скобках нужный тип. Например, для приведения переменной $a=0.5 к целому типу необходимо написать (integer)(0.5), или (integer) $a, или использовать сокращенную запись (int)(0.5). Возможность явного преобразования типов по такому принципу существует для всех типов данных (конечно, не всегда значение одного типа можно перевести в другой тип). Мы не будем углубляться во все тонкости приведения типов, поскольку PHP делает это автоматически в зависимости от контекста.

Тип float (числа с плавающей точкой). Числа с плавающей точкой (они же числа двойной точности или действительные числа) могут быть определены при помощи любого из следующих синтаксисов.

Например,

```
<?php
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
?>
```

Размер целого числа зависит от платформы, хотя максимум, как правило, ~1.8e308 с точностью около 14 десятичных цифр.

Тип string (строки). Строка – это набор символов. В PHP символ – это то же самое, что байт, т. е. существует ровно 256 различных символов. Это

также означает, что PHP не имеет встроенной поддержки Unicode. В PHP практически не существует ограничений на размер строк, поэтому нет абсолютно никаких причин беспокоиться об их длине.

Строка в PHP может быть определена тремя различными способами:

– с помощью одинарных кавычек;

– с помощью двойных кавычек;

– heredoc-синтаксисом.

Одинарные кавычки. Простейший способ определить строку – это заключить ее в одинарные кавычки «'». Чтобы использовать одинарную кавычку внутри строки, как и во многих других языках, перед ней необходимо поставить символ обратной косой черты «\», т. е. экранировать ее. Если обратная косая черта должна идти перед одинарной кавычкой либо быть в конце строки, необходимо продублировать ее «\\'».

Если внутри строки, заключенной в одинарные кавычки, обратный слэш «\» встречается перед любым другим символом (отличным от «\» и «'»), то он рассматривается как обычный символ и выводится, как и все остальные. Поэтому обратную косую черту необходимо экранировать, только если она находится в конце строки, перед закрывающей кавычкой.

В PHP существует ряд комбинаций символов, начинающихся с символа обратной косой черты. Их называют управляющими последовательностями, и они имеют специальные значения. Итак, в отличие от двух других синтаксисов, переменные и управляющие последовательности для специальных символов, встречающиеся в строках, заключенных в одинарные кавычки, не обрабатываются.

Пример использования управляющих последовательностей (html, txt)

```
<?php
echo 'Таким же образом вы можете вставлять в строки
   символ новой строки,
   поскольку это нормально';
// Выведет: Чтобы вывести ' надо
// перед ней поставить \
echo 'Чтобы вывести \' надо перед'.
   'ней поставить \\';
// Выведет: Вы хотите удалить C:\*.*?
echo 'Вы хотите удалить C:\\*.*?';
// Выведет: Это не вставит: \n
// новую строку
echo 'Это не вставит: \n новую строку';
// Выведет: Переменные $expand также
// $either не подставляются
echo 'Переменные $expand также $either'.
   'не подставляются';
?>
```

Двойные кавычки. Если строка заключена в двойные кавычки «"», PHP распознает большее количество управляющих последовательностей для специальных символов. Некоторые из них приведены в табл. 20.

Таблица 20

Управляющие последовательности

Последовательность	Значение
\n	Новая строка (LF или 0x0A (10) в ASCII)
\r	Возврат каретки (CR или 0x0D (13) в ASCII)
\t	Горизонтальная табуляция (HT или 0x09 (9) в ASCII)
\\	Обратная косая черта
\$	Знак доллара
\"	Двойная кавычка

Повторяем, если вы захотите экранировать любой другой символ, обратная косая черта также должна быть напечатана!

Самым важным свойством строк в двойных кавычках является обработка переменных.

Heredoc. Другой способ определения строк – это использование heredoc-синтаксиса. В этом случае строка должна начинаться с символа <<<, после которого идет идентификатор. Заканчивается строка этим же идентификатором. Закрывающий идентификатор должен начинаться в первом столбце строки. Кроме того, имя идентификатору должно присваиваться по тем же правилам именования, что и для остальных меток в PHP: содержать только буквенно-цифровые символы и знак подчеркивания и начинаться не с цифры или знака подчеркивания.

Heredoc-текст ведет себя так же, как и строка в двойных кавычках, при этом их не имея. Это означает, что вам нет необходимости экранировать кавычки в heredoc, но вы по-прежнему можете использовать перечисленные выше управляющие последовательности. Переменные внутри heredoc тоже обрабатываются.

Пример использования heredoc-синтаксиса (html, txt)

```
<?php
$str = <<<EOD
```

```
Пример строки, охватывающей несколько
строчек, с использованием
heredoc-синтаксиса
EOD;
// здесь идентификатор – EOD, а ниже
// идентификатор EOT
$name = 'Вася';
echo <<<EOT
Меня зовут "$name".
EOT;
// это выведет "Меня зовут "Вася".
?>
```

П р и м е ч а н и е: Поддержка heredoc была добавлена в PHP 4.

9.5. Массивы данных

Массив в PHP представляет собой упорядоченную карту – тип, который преобразует значения в ключи. Этот тип оптимизирован в нескольких направлениях, поэтому его можно использовать как собственно массив, список (вектор), хеш-таблицу (являющуюся реализацией карты), стэк, очередь и т. д. Поскольку вы можете иметь в качестве значения другой массив PHP, можно также легко эмулировать деревья.

Определить массив можно с помощью конструкции array() или непосредственно задавая значения его элементам.

Определение при помощи array( ).

```
array ([key] => value,
    [key1] => value1, ... )
```

Языковая конструкция array( ) принимает в качестве параметров пары «ключ => значение», разделенные запятыми. Символ «=>» устанавливает соответствие между значением и его ключом. Ключ может быть как целым числом, так и строкой, а значение может быть любого имеющегося в PHP типа. Числовой ключ массива часто называют индексом. Индексирование массива в PHP начинается с нуля. Значение элемента массива можно получить указав после имени массива в квадратных скобках ключ искомого элемента. Если ключ массива представляет собой стандартную запись целого числа, то он рассматривается как число, в противном случае – как строка. Поэтому запись $a["1"] равносильна записи $a[1], так же как и $a["-1"] равносильно $a[-1].

Например,

```
<?php
$books = array ("php" =>
            "PHP users guide",
          12 => true);
echo  $books["php"];
// выведет "PHP users guide"
echo $books[12];    // выведет 1
?>
```

Если для элемента ключ не задан, то в качестве ключа берется максимальный числовой ключ, увеличенный на единицу. Если указать ключ, которому уже было присвоено какое-то значение, то оно будет перезаписано. Начиная с PHP 4.3.0, если максимальный ключ – отрицательное число, то следующим ключом массива будет ноль (0).

Напимер,

```
<?php
// массивы $arr и $arr1 эквиваленты
$arr = array(5 => 43, 32, 56, "b" => 12);
$arr1 = array(5 => 43, 6 => 32,
        7 => 56, "b" => 12);
?>
```

Если использовать в качестве ключа True или False, то его значение переводится соответственно в единицу и ноль типа integer. Если использовать NULL, то вместо ключа получим пустую строку. Можно использовать и саму пустую строку в качестве ключа, при этом ее надо брать в кавычки. Так что это не то же самое, что использование пустых квадратных скобок. Нельзя использовать в качестве ключа массивы и объекты.

Определение с помощью синтаксиса квадратных скобок. Создать массив можно просто записывая в него значения. Как мы уже говорили, значение элемента массива можно получить с помощью квадратных скобок, внутри которых нужно указать его ключ, например, $book["php"]. Если указать новый ключ и новое значение, например, $book["new_key"]="new_value", то в массив добавится новый элемент. Если мы не укажем ключ, а только присвоим значение $book[]="new_value", то новый элемент массива будет иметь числовой ключ, на единицу больший максимального существующего. Если массив, в который мы добавляем значения, еще не существует, то он будет создан.

Например,

```
<?
$books["key"] = value; // добавили в массив
                       // $books значение
                       // value с ключом key
$books[] = value1; /* добавили в массив
                      значение value1 с
                      ключом 13, поскольку
                      максимальный ключу
                      нас был 12 */
?>
```

Для того чтобы изменить конкретный элемент массива, нужно просто присвоить ему с его ключом новое значение. Изменить ключ элемента нельзя, можно только удалить элемент (пару «ключ – значение») и добавить новый. Чтобы удалить элемент массива, нужно использовать функцию unset().

Например,

```
<?php
$books = array ("php" =>
                "PHP users guide",
               12 => true);
$books[] =
 "Book about Perl"; // добавили элемент
                    // с ключом (индексом)
                    // 13 – это эквивалентно
                    // $books[13] =
                     // "Book about Perl";
$books["lisp"] =
 123456; /* это добавляет к массиву новый
            элемент с ключом "lisp" и
           значением 123456 */
unset($books[12]); // это удаляет элемент
                   // с ключом 12 из массива
unset ($books); // удаляет массив полностью
?>
```

Заметим, что когда используются пустые квадратные скобки, максимальный числовой ключ ищется среди ключей, существующих в массиве с момента последнего переиндексирования. Переиндексировать массив можно с помощью функции array_values().

Пример переиндексации массива (html, txt).

```
<?php
$arr =
 array ("a","b","c"); /* создаем массив
                         со значениями
                          "a", "b" и "c".
                         Поскольку ключи
                          не указаны, они
                         будут 0,1,2
                          соответственно */
print_r($arr); // выводим массив (и ключи,
               // и значения)
unset($arr[0]);
unset($arr[1]);
unset($arr[2]);
       // удаляем из него все значения
print_r($arr);  // выводим массив (и ключи,
                // и значения)
$arr[] = "aa";  // добавляем новый элемент
                // в массив.
                // Его индексом (ключом)
                // будет 3, а не 0
print_r($arr);

$arr =
 array_values($arr); // переиндексируем
                     // массив
$arr[] = "bb"; // ключом этого элемента
               // будет 1
print_r($arr);
?>
```

Результатом работы этого скрипта будет:

```
Array ([0] => a [1] => b [2] => c)
Array ()
Array ([3] => aa )
Array ([0] => aa [1] => bb)
```

ЛИТЕРАТУРА

1. Шрайбер Т. Дж. Моделирование на GPSS. – М.:, Мир, 1980.

2. Советов Б. Я., Яковлев С. А. Моделирование систем. Лабораторный практикум: Учеб. пособие для вузов. – М.: Высш. шк., 1989.

3. Советов Б. Я. Моделирование систем. Практикум: Учеб. пособие для вузов / Б. Я. Советов, С. А. Яковлев. – 2-е изд., перераб. и доп. – М.: Высш. шк., 2003.

3. Якубайтис Э. А. Информационные сети и системы: Справочная книга. – М.: Финансы и статистика, 1996.

5. Бэрри Нанс. Компьютерные сети: Пер. с англ. – М.: БИНОМ, 1996.

6. Основы современных компьютерных технологий / Под ред. А. Д. Хомоненко. – СПб.: КОРОНА принт, 1998.

7. Ресурсы Microsoft Windows NT Workstation 4.0: Пер. с англ. – СПб.: BNV, 1998.

8. Networking Essentials / Титтел Эд и др. – СПб.: ПИТЕР, 1999.

9. TCP/IP / Титтел Эд и др. – СПб.: ПИТЕР, 1999.

10. Компьютерные сети: Учеб. курс Microsoft Corporation. – М.: Русская редакция, 1999.

ОГЛАВЛЕНИЕ

Учебное издание

Юденков Виктор Степанович
Севостьян Дмитрий Михайлович

ИНФОМАЦИОННЫЕ СЕТИ

Тексты лекций

Редактор *Ю. В. Кравцова*
Компьютерная верстка *О. Ю. Шантарович*

Подписано в печать 04.04.2006. Формат 60×84 $^{1}/_{16}$.
Бумага офсетная. Гарнитура Таймс. Печать офсетная.
Усл. печ. л. 8,4. Уч.-изд. л. 8,6.
Тираж 150 экз. Заказ 164.

Учреждение образования
«Белорусский государственный технологический университет».
220050. Минск, Свердлова, 13а.
ЛИ № 02330/0133255 от 30.04.2004.

Отпечатано в лаборатории полиграфии учреждения образования
«Белорусский государственный технологический университет».
220050. Минск, Свердлова, 13.
ЛП № 02330/0056739 от 22.01.2004.