Сети ЭВМ и телекоммуникации

Лабораторная работа №3.

«Разработка многопоточного сервера с использованием механизма сокетов»

Цель:

Научиться работать с сокетами и строить многопоточные серверные приложения на примере сервера шифрования.

Задание:

1. Разработать серверное приложение которое осуществляет взаимодействие при помощи механизма сокетов TCP/IP. Сервер работает в локальной сети по определенному IP-адресу и порту, которые настраиваются при запуске сервера, и отвечает на запросы клиента. Формат команд, которые должен обрабатывать сервер определяется согласно варианту задания. Основное требование к серверу, это обеспечение многопоточности, т.е. серверное приложение должно быть разработано таким образом, чтобы одновременно могло обрабатывать запросы нескольких пользователей и возможность работы со стандартными клиентами.
2. Клиентская часть должна реализовывать функции, соответствующие номеру варианта. Обмен информацией с сервером осуществляется в текстовом режиме посредством определенных команд, каждая из которых выполняет определенное действие. Каждая команда сервера состоит из служебного слова и параметров. Сервер не должен быть чувствителен к регистру команд. Запрос должен завершаться символом конца строки (0x10) для обеспечения совместимости сервера со стандартными клиентами, такими как Telnet.

   Примечание:

   1. Задание является дифференцированным.
      - На оценку «Удовлетворительно» достаточно реализовать многопоточный сервер, который работает в режиме командной строки и позволяет осуществлять шифрование/дешифрование текстовых сообщений
      - На оценку «Хорошо» необходимо реализовать многопоточный сервер с графическим интерфейсом, который позволяет осуществлять шифрование/дешифрование текстовых сообщений
      - На оценку «Отлично» необходимо реализовать многопоточный сервер с графическим интерфейсом, который позволяет осуществлять шифрование/дешифрование текстовых сообщений, а также бинарных файлов произвольного размера.

Варианты заданий

Для всех вариантов заданий должны поддерживаться команды:

- hello <номер варианта> - команда регистрации пользователя с которой начинается работа с сервером. Сервер должен ответить hello variant <номер варианта>.

  Например: запрос клиента - hello 5

  ответ сервера - hello variant 5

- bye <номер варианта> - команда отключения клиента от сервера. Сервер должен ответить bye variant <номер варианта> и завершить работу.

  Например: запрос клиента - bye 10

  ответ сервера - bye variant 10

- encrypt <сообщение>, <пароль> - должен реализовывать метод шифрования из лабораторной работы №3 или №4 (на усмотрение студента) по дисциплине «Информационная безопасность и защита информации» согласно варианту

- decrypt <сообщение>, <пароль> - должен реализовывать метод дешифрования из лабораторной работы №3 или №4 (на усмотрение студента) по дисциплине «Информационная безопасность и защита информации» согласно варианту

Теоретическая часть

Какие бы замечательные идеи в области телекоммуникаций, распределенных баз знаний или поисковых систем вам не пришли в голову, реализовать их на практике можно, лишь написав соответствующую программу. Основные операционные среды (Unix или Windows) базируются в настоящее время на идеологии сокетов (socket). Эта технология была разработана в университете г. Беркли (США) для системы Unix, поэтому соединители (сокеты) иногда называют сокетами Беркли (berkeley sockets). Сокеты реализуют механизм взаимодействия не только партнеров по телекоммуникациям, но и процессов в ЭВМ вообще. Технология сокетов лежит в основе современного сетевого программирования.

Работа с сокетами содержит ряд этапов: сокет создается, настраивается на заданный режим работы, применяется для организации обмена и, наконец, ликвидируется. Технология сокетов поддерживает работу с любыми стеками протоколов, совмещенные процедуры ввода/вывода, использование большого числа сервиспровайдеров (серверов услуг), возможность группирования сокетов, что позволяет реализовать их приоритетное обслуживание, и многое другое. Набор операторов, поддерживающих интерфейс сервис провайдера, образует отдельную динамическую библиотеку.

Для общей синхронизации работы сервиспровайдеров и приложений в winsock введено понятие объектов событий. Объекты событий служат, в частности, для организации работы совмещенных по времени процессов информационного обмена. Здесь уместно замечание об использовании стандартных номеров портов. В многозадачных, многопользовательских системах стандартные номера портов используются при инициализации процесса. Так как допускается несколько идентичных соединений (например, несколько одновременных сессий FTP) между клиентом и сервером, стандартными номерами портов здесь не обойтись. Ведь P_SIP_S сервера могут соответствовать несколько P_CIP_C клиента.

В системах, ориентированных на соединение, пара комбинаций IP-адресов и номеров портов однозначно определяет канал связи между двумя процессами в ЭВМ. Такая комбинация называется сокетом (socket). Номера портов могут и совпадать, так как относятся к разным машинам, но IP-адреса должны быть обязательно разными. Впервые идея сокета была использована в системе BSD4.3 Unix для организации сетевого ввода/вывода. В Unix внешнее устройство и файл с точки зрения системного программиста эквивалентны. Сетевые процедуры несколько сложнее и не укладываются в такую простую схему. Из этой схемы выпадают, прежде всего, операции, при которых

сервер пассивно ожидает обращения, особенно операции обмена, не ориентированные на соединение. Сокет является пограничным понятием между протоколами телекоммуникаций и операционной системой ЭВМ. Сокеты играют важную роль при написании прикладных программ (API).

Сокет отправителя = IP-адрес отправителя + номер порта отправителя

Сокет адресата = IP-адрес адресата + номер порта адресата

При обменах, ориентированных на соединение, формируется ансамбль (IP_SP_S + IP_DP_D), где IP_SP_S — адрес и порт отправителя, а IP_DP_D − адрес и порт места назначения.

Межкомпьютерные коммуникации не сводятся к знакомству с соседским депозитарием, к выполнению операций Telnet/ssh, FTP/scp и т.д. Одной из важнейших задач является удаленный контроль за процессами в больших распределенных системах, когда обмен информацией активизируется не человеком, а ЭВМ. Примерами таких задач могут служить управление современными высокотехнологичными производствами, сбор метео- или другой геофизической информации в реальном масштабе времени, эксперименты в области физики высоких энергий, где для контроля установки и сбора экспериментальных данных используются десятки (а иногда и сотни) вычислительных машин, которые обмениваются диагностической информацией и данными. Именно для решения таких задач и применяются идеи сокетов, "труб" и т.д.. Понятие сокета в прикладных программах — это не просто комбинация IP-адресов и номеров портов, это указатель на структуру данных, где хранятся параметры виртуального канала. Прежде чем воспользоваться сокетом, его нужно сформировать. Оператор формирования сокета имеет вид:

s=socket(INT AF, INT type, INT protocol);

где все параметры целочисленные, AF (address_family) характеризует набор протоколов, соответствующий данному сокету (это может быть набор Internet, Unix, Appletalk и т.д.). Для Интернет AF может принимать только значение PF_INET, для Unix PF_UNIX. Аргумент type определяет тип коммуникаций (SOCK_STREAM, SOCK_RAW, и SOCK_DGRAM). Аргумент protocol задает код конкретного протокола из указанного набора (заданного AF), который будет реализован в данном соединении. Протоколы обозначаются символьными константами с префиксом IPPROTO_ (например, IPPROTO_TCP или IPPROTO_UDP). Допускается значение protocol=0 (протокол не указан), в этом случае используется значение по умолчанию для данного вида соединений. Значения AF и type можно обычно найти в файле <sys/socket.h>. Возвращаемый параметр S представляет собой дескриптор сокета. Параметр SOCK_STREAM говорит о том, что вы намерены создать надежный двунаправленный канал обмена, ориентированный на соединение (TCP для Интернет). Связь с другим процессом в этом случае устанавливается оператором connect . После установления соединения данные могут посылаться оператором send или получаться посредством оператора recv. Параметр SOCK_DGRAM характеризует канал, не ориентированный на соединение, с пакетами фиксированного размера (например, UDP в случае AF= PF_INET). Такой канал позволяет использовать операторы sendto и recvfrom. Параметр SOCK_RAW определяет третий режим, при котором возможно использование протоколов нижнего уровня, например, ICMP или даже IP. Таким образом, формирование сокета — это создание описывающей его структуры данных.

Если операция socket завершилась успешно, s равно дескриптору сокета, в противном случае s=INVALID_SOCKET (1). С помощью оператора WSAGetLastError можно получить код ошибки, проясняющий причину отрицательного результата.

Дескриптор сокета указывает на элемент таблицы дескрипторов, соответствующий данному сокету. Оператор socket отводит место в этой таблице. Элемент такой таблицы имеет вид:

- код семейства протоколов
- код типа сервиса
- локальный IP-адрес
- удаленный IP-адрес
- номер локального порта
- номер удаленного порта.

IP-адрес определяет интерфейс ЭВМ, а номер порта в данном случае характеризуют сетевую процедуру (процесс). Эта структура данных позволяет осуществлять несколько соединений между рабочей станцией и, например, WEBсервером, в том числе имеющих разный уровень приоритета.

Так как в Unix возможно формирование сокета без IP-адресов, а для практической работы они нужны, имеется оператор bind, который позволяет присвоить определенный IP-адрес заданному сокету:

r=bind(s, const struct socketaddr far*name, int namelen),

где s — целочисленный код дескриптора, параметр name (идентификатор локального адреса) обычно (для Интернет) содержит три величины: IP-адрес ЭВМ, код протокольного набора, номер порта, который определяет характер приложения. Структура адресной информации имеет вид:

```
struct sockaddr {
u_short sa_family;
char sa_data[14];
};
```

Параметр namlen определяет длину второго параметра. В рамках этой идеологии легко реализовать систему клиент-сервер. IP-адрес может быть сделан равным INADDR_ANY (или =0), если ЭВМ имеет несколько интерфейсов. При номере порта, равном нулю, windows socket присвоит порту уникальный номер в диапазоне 1024-5000. Приложение может выполнить операцию getsockname после bind, чтобы определить присвоенный адрес. Оператор bind выполняется до операций connect или listen. При корректном выполнении оператор bind возвращает код 0 (r=0), в противном случае SOCKET_ERROR=1. Команда bind выдается для записи собственного номера порта. Сервер генерирует команду bind, чтобы подготовить определенный вид связи (например, FTP), и пассивно ожидает запроса connect со стороны клиента:

R=connect(s, const struct socketaddr FAR*name, int namelen)

где s — дескриптор сокета, name — идентификатор адреса места назначения (указатель на структуру данных), а namelen — длина этого адреса. Таким образом, оператор connect сообщает IP-адрес и номер порта удаленной ЭВМ. Если адресное поле структуры name содержит нули, оператор connect вернет ошибку WSAEADDRNOTAVAIL (или SOCKET_ERROR = 1).

Установка в режим ожидания осуществляется командой listen, которая организует очередь запросов:

R=listen(s, int backlog)

где backlog задает максимальный размер очереди для приходящих запросов соединения (то есть, сколько запросов может быть принято на обслуживание без потерь; обычно этот параметр равен 5). При переполнении очереди будет послано сообщение об ошибке. Следует иметь в виду, что клиент, ориентированный на соединение, также должен прослушивать порт протокола, ожидая появления дейтограммоткликов. Ожидающий сокет посылает каждому отправителю сообщениеотклик, подтверждающее получение запроса на соединение. Оператор listen подготавливает сокет к обработке потока запросов, система должна быть достаточно быстродействующей. Запросы из очереди извлекаются оператором accept :

R=accept(s, struct sockaddr FAR*addr, int FAR*addrlen),

где s — дескриптор сокета, который прослушивает соединение (тот же, что и в listen), addr — опционный указатель на структуру, которая содержит адрес, addrlen — код длины адреса. Оператор accept позволяет серверу принять запрос от клиента. Когда входная очередь сформирована, программа реализует процедуру accept и переходит в режим ожидания запросов. Программа извлекает первый элемент очереди, создает новый сокет со свойствами, идентичными s, и при успешном выполнении возвращает дескриптор нового сокета. При возникновении ошибки возвращается код INVALID_SOCKET. По окончании обработки запроса сервер вновь вызывает accept, который возвращает ему дескриптор сокета очередного запроса, если таковой имеется. Если очередь пуста, accept блокирует программу до получения связи. Существуют серверы с параллельной и последовательной обработкой запросов. Параллельный обработчик запросов не ждет завершения обработки предшествующего запроса и вызывает оператор accept немедленно. В системе Unix используются обычно параллельные обработчики запросов.

Возможная схема использования сокетов в случае работы N клиентов с одним WEB-сервером показана на рис. 1. Клиент 1 сформировал два соединения с сервером.

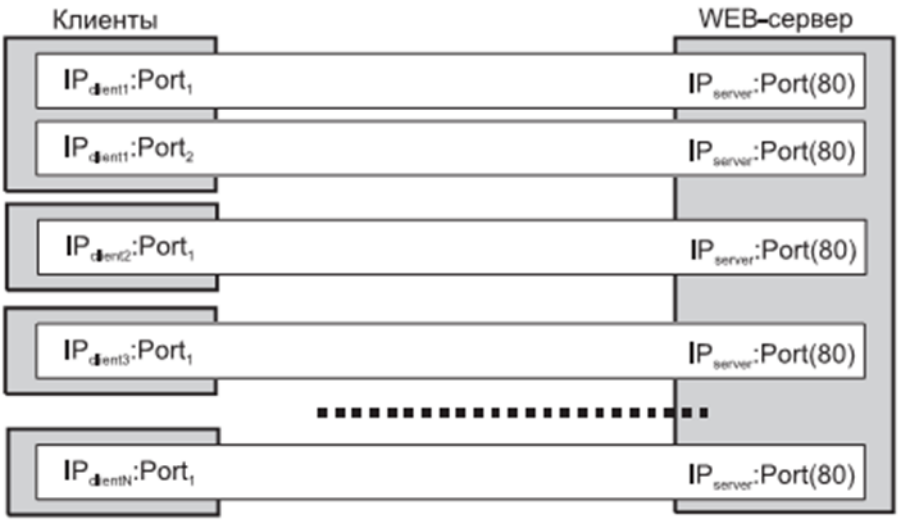

Рис. 1. Формирование сокетов при подключении к одному WEBсерверу

Схема взаимодействия различных операторов winsock в рамках идеологии клиент/сервер для случая процедур, ориентированных на соединение, показана на рис 2. Горизонтальными стрелками обозначены направления посылки сетевых сообщений.

Следует иметь в виду, что программе клиента (выделена рамкой) в этом режиме не нужно знать номер порта, поэтому она не обращается к процедуре bind, а для установления связи сразу вызывает оператор connect. Современные распределенные информационные системы, WWW-серверы, поисковые системы и т.д. эффективно используют механизмы формирования сокетов и многие процедуры, описанные в данном разделе. Из литературы [2.15, 2.24] известно, что для многих видов услуг в Интернет выделены строго определенные номера портов. Доступ же к этим услугам должен быть обеспечен достаточно большому числу пользователей. С клиентской стороны при этом используются номера портов со значениями из диапазона 1024-5000. Для каждого нового клиентского запроса в ЭВМ-сервере, как правило, формируется новый процесс.

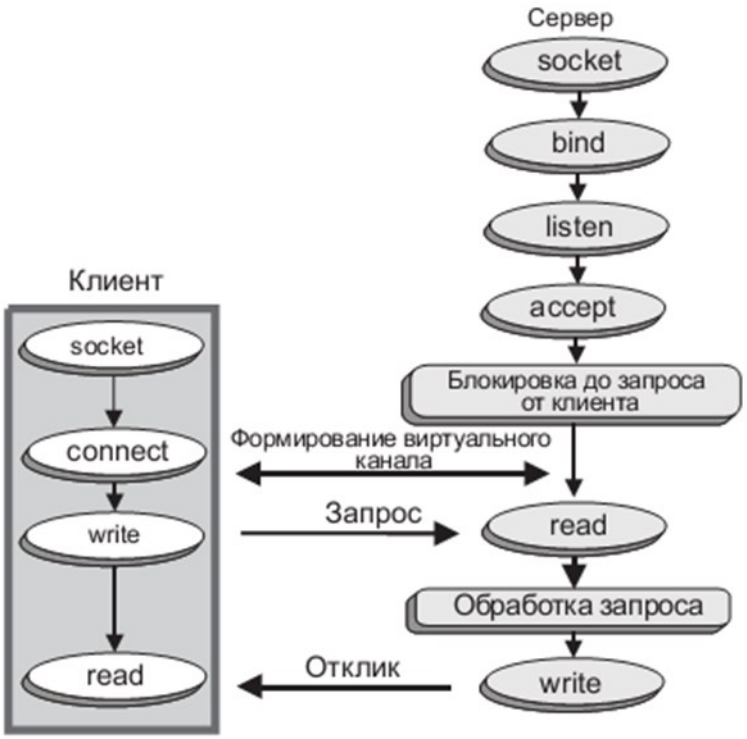

Рис. 2. Схема взаимодействия операторов winsock для процедур, ориентированных на соединение

Лишь при успешной реализации всех перечисленных операций может начаться обмен данными. Для пересылки информации могут использоваться команды write, read, send, recv. Команды write и read имеют форму вызова:

R=write(s, buf, len) или R=read(s, buf, len),

где s — дескриптор сокета, buf — имя массива, подлежащего пересылке (или предназначенного для приема), len — длина этого массива. Оператор writev отличается от write тем, что данные могут не лежать в виде непрерывного массива:

R=writev(s, io_vect, vectlen) или R=readv(s, io_vect, vectlen),

где s — дескриптор сокета, io_vect — векторуказатель на список указателей, vectlen — длина списка указателей. Команда выполняется медленнее, чем write или read. Список указателей имеет формат (рис. 3):

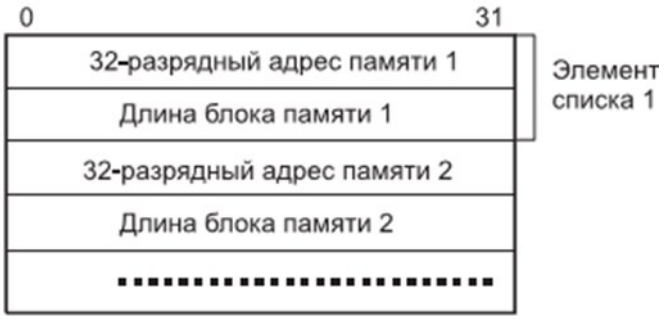

Рис. 3. Формат списка указателей для функций readv и writev

Команды send(s, msg_buf, buflen, flags) и recv имеют аналогичный формат, но среди параметров обращения содержат переменную flags, которая служит для целей диагностики и управления передачей данных (например, пересылка информации с высоким приоритетом (MSG_OOB — Message Out Of Band), что используется, в частности, при передаче звуковых сообщений). При работе с операторами send или recv надо быть уверенным, что принимающая сторона знает, что ей следует делать с этими приоритетными сообщениями. Другой возможный флаг, определяемый константой MSG_PEEK, позволяет анализировать запросы из входной очереди транспортного уровня. Обычно после считывания данных из входной очереди они уничтожаются. Когда MSG_PEEK=1, данные из входной очереди не стираются. Этот флаг используется, например, программой FTP. При успешном выполнении команды будет возвращено число переданных байтов, в противном случае —1.

Все перечисленные выше операторы рассчитаны на применение в рамках протоколов, ориентированных на установление соединения (TCP), где не требуется указание адреса места назначения. В протоколах типа UDP (не ориентированных на соединение) для передачи информации используются операторы sendto, recvfrom или sendmsg:

R=sendto(s, msg_buf, buflen, flags, adr_struc, adr_struc_len)

или

recvfrom(s, msg_buf, buflen, flags, adr_struc, adr_struc_len),

где s — дескриптор сокета, msg_buf — указатель на буфер, где лежит сообщение, buflen — длина этого буфера (длина сообщения), adr_struc — адресная структура, содержащая исчерпывающую информацию об адресате, adr_struc_len — длина этой структуры. Оператор recvfrom принимает все данные, приходящие на его порт. Приняв дейтограмму, recvfrom записывает также адрес, откуда эта дейтограмма получена. Сервер может посылать по этому адресу дейтограммуотклик. Вызов оператора sendmsg имеет форму:

R=sendmsg(s, msg_struc, flags) [или recvmsg(s, msg_struc, flags)],

где s — дескриптор сокета, msg_struc — информационная структура, формат которой показан ниже на Рис. 4. Применение структур делает программирование пересылки сообщений более гибким. Следует учитывать, что для обменов, не ориентированных на соединение, сокет как бы состоит лишь из одной половины (IP-адрес

и номер порта). Сокеты, созданные однажды для обмена (UDP), далее могут жить своей жизнью. Они могут принимать пакеты от других аналогичных "сокетов" и сами посылать им дейтограммы (кавычки здесь связаны с тем, что это не реальный сокет и никакого соединения здесь не осуществляется).

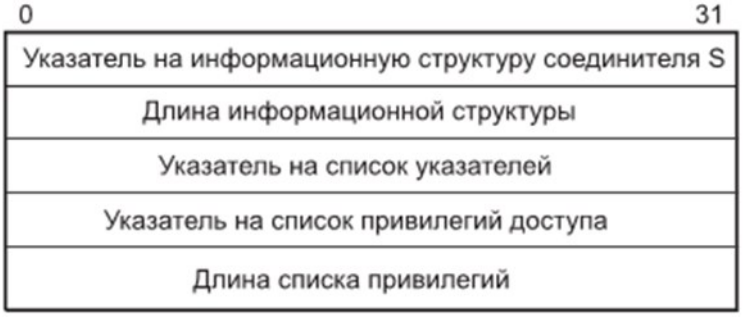

Рис. 4. Формат информационной структуры msg_struc

Взаимодействие операторов winsock для систем, не ориентированных на соединение, показано на рисунке 5. Здесь так же, как и в случае, ориентированном на соединение, сервер вызывает socket и bind, после чего обращается к процедуре recvfrom (вместо read или recv). Программаклиент в данной схеме обращается к оператору bind и совсем не использует оператор connect (ведь предварительного соединения не нужно). Для передачи запросов и приема откликов здесь служат операторы sendto и recvfrom, соответственно.

Помимо уже описанных операторов для работы с сокетами имеется еще один — select, довольно часто используемый серверами. Оператор select позволяет процессу отслеживать состояние одного или нескольких сокетов. Для каждого сокета вызывающая программа может запросить информацию о статусе read, write или error. Форма обращения имеет вид:

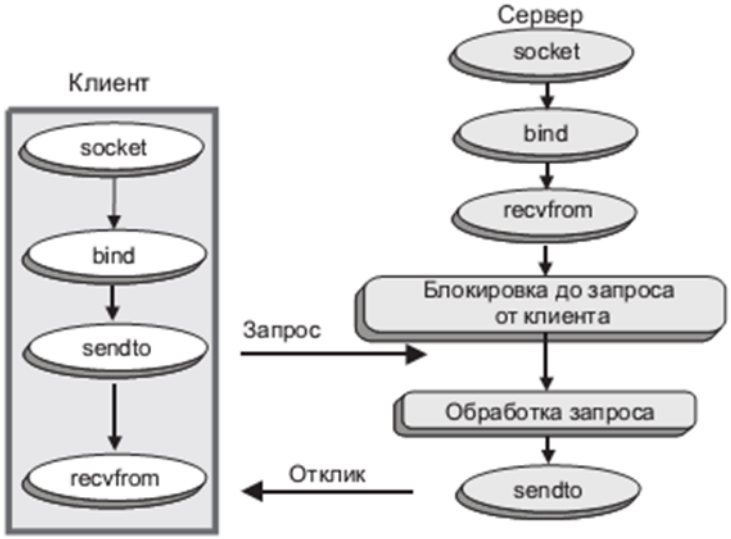

Рис. 5. Схема взаимодействия операторов winsock для процедур, не ориентированных на соединение

R=select(num_of_socks, read_socks, write_socks, error_socks, max_time),

где num_of_socks — число контролируемых сокетов (в некоторых реализациях не используется и является необязательным; по умолчанию это число не должно превышать 64).