САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ТЕХНИЧЕСКОЙ КИБЕРНЕТИКИ

В. М. Ицыксон

ТЕХНОЛОГИИ КОМПЬЮТЕРНЫХ СЕТЕЙ
ПРОГРАММИРОВАНИЕ СЕТЕВЫХ ПРИЛОЖЕНИЙ

Методические указания к лабораторным работам

Санкт-Петербург

2006

УДК 004.77

Ицыксон В.М. **Технологии компьютерных сетей. Программирование сетевых приложений. Методические указания к лабораторным работам**. СПб: СПбГПУ, 2006, 91 с.

Методические указания предназначены для выполнения курса лабораторных работ по дисциплине «Технологии компьютерных сетей», читающейся на специальностях 230101 «Вычислительные машины, комплекса, системы и сети» и 220201 «Управление и информатика в технических системах».

Описываются базовые понятия технологий BSD-сокетов, показывается типовая структура сетевых приложений. Приводятся варианты заданий для лабораторных работ по проектированию сетевых приложений с использованием технологий сокетов.

Рассматриваются технологии проектирования сетевых приложений на уровне команд прикладных протоколов на примере библиотеки Internet Direct. Приводятся варианты заданий для лабораторных работ по проектированию клиентских и серверных приложений.

Описываются сетевые утилиты операционных систем, применяемые при разработке и отладке сетевых приложений.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Бурно развивающиеся в последнее время глобальные и локальные сети вызывают постоянное совершенствование предоставляемых сетевых сервисов и появление новых распределённых приложений, функционирующих в сетях TCP/IP. Цель данных методических указаний – научить базовым подходам, реализующим основные технологии, используемые при проектировании сетевых приложений.

Разработка сетевого программного обеспечения требует от программиста детальной реализации всех клиентских и серверных функции указанного протокола. Выполнение этих функций можно обеспечивать на разных уровнях детализации:

- на уровне TCP/UDP сокетов;
- на уровне обработки команд прикладных протоколов;
- на уровне специализированных прикладных библиотек;
- с помощью различных распределённых объектных технологий.

Реализация функций прикладных протоколов на уровне сокетов подробно описана в разделе 1. Реализация сетевых приложений на уровне обработки команд прикладных протоколов изложена в разделе 2. Реализация функций прикладных протоколов с помощью специализированных библиотек, а также с помощью распределённых объектных технологий, таких как, CORBA, DCOM, RMI и другие, выходит за рамки данных методических указаний.

Для разработки и отладки сетевых приложений следует использовать специальные утилиты и программы операционных систем. Перечень ключевых утилит операционных систем Linux и MS Windows и их основные опции приведены в разделе 3.

Требования к содержанию и структуре отчета по лабораторным работам изложены в разделе 4.

Автор выражает благодарность сотрудникам кафедры автоматики и вычислительной техники СПбГПУ Ельцову А.А. и Зозуле А.В. за ценные замечания, а также за помощь в подготовке первого раздела.

1. ИЗУЧЕНИЕ ИНТЕРФЕЙСА СОКЕТОВ

1.1. Разработка сетевых приложений с помощью технологии сокетов

С точки зрения архитектуры TCP/IP сокетом называется пара (IP-адрес, порт), однозначно идентифицирующая прикладное приложение в сети Internet.

С точки зрения операционной системы BSD-сокет (Berkley Software Distribution) или просто сокет — это выделенные операционной системой набор ресурсов, для организации сетевого взаимодействия. К таким ресурсам относятся, например, буфера для приёма/посылки данных или очереди сообщений. Технология сокетов была разработана в университете Беркли и впервые появилась в операционной системе BSD-UNIX.

В операционной системе MS Windows имеется аналогичная библиотека сетевого взаимодействия WinSock, реализованная на основе библиотеки BSD-сокетов. Существует две версии библиотеки WinSock1 и WinSock2. В подавляющем большинстве случаев функции и типы библиотеки WinSock совпадают с функциями и типами BSD-сокетов. Об имеющихся отличиях будет сказано отдельно.

1.1.1. Создание сокета

Для создания сокета в библиотеках BSD-socket и WinSock имеется системный вызов socket:

```
int socket(int domain, int type, int protocol);
```

В случае успеха результат вызова функции – дескриптор созданного сокета, в случае ошибки (-1) в библиотеке BSD-socket и INVALID_SOCKET в библиотеке WinSock.

Параметр domain указывает на домен, в пространстве которого создаётся данный сокет. Домен AF_UNIX используется для межпроцессного

взаимодействия, домен AF_INET – для передачи с использованием стека протоколов TCP/IP.

Параметр type определяет тип создаваемого сокета. Этот параметр может принимать значения:

- SOCK_STREAM – для организации надёжного канала связи с установлением соединений
- SOCK_DGRAM – для организации ненадёжного дейтаграммного канала связи
- SOCK_RAW – для организации низкоуровнего доступа на основе «сырых» сокетов

Параметр protocol – идентификатор используемого протокола. В большинстве случаев протокол однозначно определяется типом создаваемого сокета, и передаваемое значение этого параметра – 0. Если же это не так (например, в случае SOCK_RAW), то необходимо явно задавать идентификатор протокола. Для его получения имеются системные вызовы getprotobyname и getprotobynumber, которые разбирают файл /etc/protocols и получают идентификатор сетевого протокола:

```
struct protoent* getprotobyname(const char *name);
struct protoent* getprotobynumber(int proto);
```

Эти функции заполняют структуру protoent, поле p_proto которой следует использовать в качестве параметра protocol вызова socket:

```
struct protoent
{
  char*   p_name;     // имя протокола из файла protocols
  char**  p_aliases;  // список псевдонимов
  int     p_proto;    // идентификатор протокола
}
```

Структура protoent описана в файле /usr/include/netdb.h

1.1.2. Организация соединения

Для установления TCP-соединения используется вызов connect:

```
int connect(int s, const struct sockaddr* serv_addr,
            int addr_len);
```

Результатом выполнения функции является установление TCP-соединения с TCP-сервером. Функция возвращает значение 0 в случае успеха и -1 в случае ошибки.

Параметр s – дескриптор созданного сокета.

Параметр serv_addr – указатель на структуру, содержащую параметры удалённого узла.

Параметр addr_len – размер в байтах структуры, на которую указывает параметр serv_addr.

При программировании сокетов из домена AF_INET вместо структуры sockaddr используется приводимая к ней структура sockaddr_in, находящаяся в подключаемом файле /usr/include/linux/in.h:

```
struct sockaddr_in
{
  sa_family_t           sin_family;  // Коммуникационный домен
  unsigned short int    sin_port;    // Номер порта
  struct in_addr        sin_addr;    // IP-адрес
  ...
};
```

Успех выполнения функции connect означает корректное установление логического канала связи и возможность начала передачи и приёма данных по протоколу TCP.

В случае использования вызова connect для протокола UDP установления соединения не происходит, а адрес и порт из структуры serv_addr используется как адрес по умолчанию для последующих вызовов send и recv.

Для более простого заполнения параметров структуры sockaddr_in используются системные вызовы htons и inet_addr, осуществляющие замену порядка следования байт в номере порта и перевод IP-адреса из строкового вида в числовой соответственно, например:

```
serv_addr.sin_port = htons(3128);
```

```
serv_addr.sin_addr.s_addr = inet_addr(“192.168.1.1”);
```

1.1.3. Передача и приём данных по протоколу TCP

Передача и приём данных в рамках установленного TCP-соединения осуществляется вызовами send и recv:

```
int send(int s, const void *msg, size_t len, int flags);
int recv(int s, void *msg, size_t len, int flags);
```

Параметр s – дескриптор сокета, параметр msg – указатель на буфер, содержащий данные (вызов send), или указатель на буфер, предназначенный для приёма данных (вызов recv). Параметр len – длина буфера в байтах, параметр flags – опции посылки или приёма данных.

Возвращаемое значение – число успешно посланных или принятых байтов, в случае ошибки функция возвращает значение -1.

1.1.4. Передача и приём данных по протоколу UDP

В случае установленного адреса по умолчанию для протокола UDP (вызов connect) функции для передачи и приёма данных по протоколу UDP можно использовать вызовы send и recv.

Если адрес и порт по умолчанию для протокола UDP не установлен, то параметры удалённой стороны необходимо указывать или получать при каждом вызове операций записи или чтения. Для протокола UDP имеется два аналогичных вызова sendto и recvfrom:

```
int sendto(int s, const void *buf, size_t len, int flags,
           struct sockaddr *to, int* tolen);
int recvfrom(int s, void *buf, size_t len, int flags,
           struct sockaddr *from, int* fromlen);
```

Параметры s, buf, len и flags имеют тот же смысл, что и в случае использования функций send и recv, параметры to и tolen – атрибуты адреса удалённого сокета при посылке данных, параметры from и fromlen – атрибуты структуры данных в которую помещаются параметры удалённого сокета при получении данных.

1.1.5. Завершение TCP-соединения

Завершение установленного TCP-соединения осуществляется в библиотеке BSD-socket с помощью вызова shutdown:

```
int shutdown(int s, int how);
```

Параметр s – дескриптор сокета, параметр how – определяет способ закрытия:

- SHUT_RD – запрещён приём данных
- SHUT_WR – запрещена передача данных
- SHUT_RDWR – запрещены и приём и передача данных.

В библиотеке WinSock семантика вызова несколько отличается:

```
int shutdown(SOCKET s, int how);
```

Параметр s – дескриптор сокета, параметр how – определяет способ закрытия:

- SD_RECEIVE – запрещён приём данных. В случае наличия данных в очереди соединение разрывается.
- SD_SEND – запрещена передача данных.
- SD_BOTH – запрещены и приём и передача данных.

1.1.6. Закрытие сокета

По окончании работы следует закрыть сокет, для этого в библиотеке BSD-socket предусмотрен вызов close:

```
int close(int s);
```

Аналогичный вызов в библиотеке WinSock имеет название closesocket:

```
int closesocket(SOCKET s);
```

Параметр s – дескриптор сокета.

Возвращаемое значение – 0, в случае успеха.

1.1.7. Привязывание сокета

Созданный сокет является объектом операционной системы, использующим её отдельные ресурсы. В то же время в большинстве случаев не-

достаточно просто выделить ресурсы операционной системы, а следует также связать эти ресурсы с конкретными сетевыми параметрами: сетевым адресом и номером порта. Особенно это важно для серверных сокетов, для которых такая связь – необходимое требование доступности разрабатываемого сетевого сервиса.

Организация привязки созданного вызовом socket() сокета к определённым IP-адресам и портам осуществляется с помощью функция bind:

```
int bind(int s, struct sockaddr *addr, socklen_t addrlen);
```

Параметр s – дескриптор сокета, параметр addr задаёт указатель на структуру, хранящую параметры адреса и порта, addrlen – размер структуры addr в байтах.

1.1.8. Перевод TCP-сокета в состояние прослушивания

Для перевода сокета в состояние прослушивания служит системный вызов listen:

```
int listen(int s, int backlog);
```

Параметр s – дескриптор сокета, параметр backlog – задаёт максимальную длину, до которой может расти очередь ожидающих соединений.

В случае успеха возвращаемое значение – 0. При ошибке возвращается -1.

1.1.9. Приём входящего TCP-соединения

В случае, когда сокет находится в состоянии прослушивания (listen) необходимо отслеживать поступление входящих соединений. Для этого предусмотрен системный вызов accept:

```
int accept(int s, struct sockaddr *addr, socklen_t *addrlen);
```

Параметр s – дескриптор прослушивающего сокета, параметр addr – указатель на структуру, содержащую параметры сокета, инициирующего соединение, addrlen – размер структуры addr в байтах.

Возвращаемое значение – дескриптор сокета, созданного для нового соединения. Большинство параметров нового сокета соответствуют параметрам слушающего сокета. Полученный сокет в дальнейшем может использоваться для передачи и приёма данных.

В случае если входящих соединений нет, то функция accept ожидает поступления запроса на входящее соединение.

1.1.10. Дополнительные функции

Для получения текстового представления ошибки в операционной системе Linux используется функция strerror:

```
char* strerror(int errnum);
```

Функция слежения за изменением файлового дескриптора select:

```
int select(int n, fd_set *readfds, fd_set *writefds,
           fd_set *exceptfds, struct timeval *timeout);
```

Эта функция может использоваться для определения наличия информации в приёмном сокете.

Функции, предназначенные для установки и чтения значений опций сокетов setsockopt и getsockopt:

```
int setsockopt(int s, int level, int optname, const void *optval,
               socklen_t optlen);
int getsockopt(int s, int level, int optname, void *optval,
               socklen_t *optlen);
```

Параметр s – дескриптор сокета, level – уровень, на котором должна происходить интерпретация флага, optname – идентификатор опции. Параметры optval и optlen используются в функции setsockopt для доступа к значениям флагов, для getsockopt они задают буфер, в который нужно поместить запрошенное значение.

1.1.11. Структура TCP-клиента

Перечисленные в предыдущих параграфах функции используются для организации клиентских и серверных приложений.

Клиент протокола TCP создаёт экземпляр сокета, необходимый для взаимодействия с сервером, организует соединение, осуществляет обмен данными, в соответствии с протоколом прикладного уровня.

Типичная структура TCP-клиента представлена на рис. 1.1.

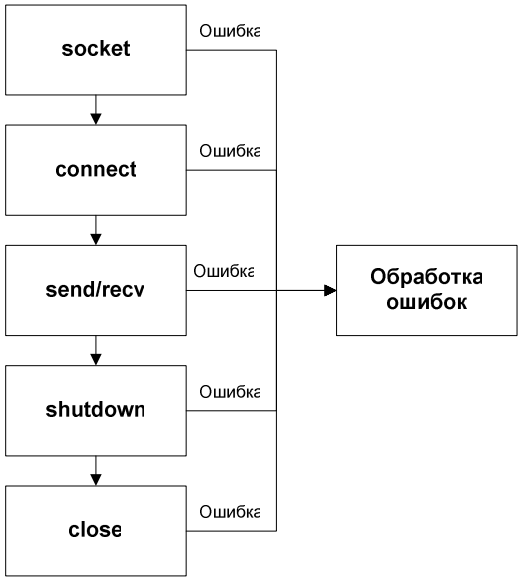


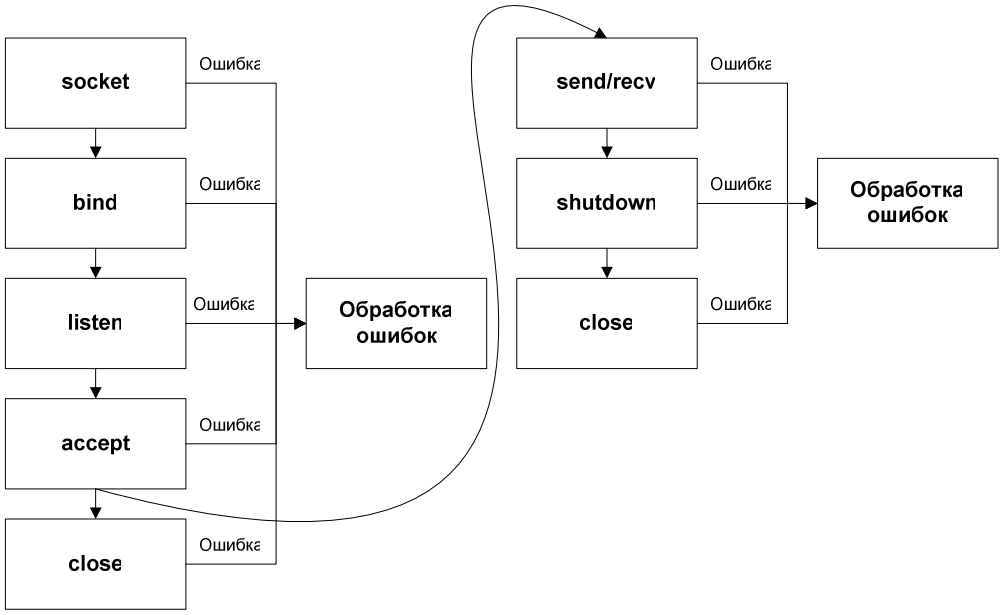
Рис. 1.1. Типичная структура TCP-клиента

Если инициатором разрыва соединения является клиентское приложение (например, канал данных протокола FTP), то далее следует вызвать функцию shutdown и после этого закрыть сокет.

Каждый вызов функций библиотеки сокетов должен сопровождаться проверкой на наличие ошибочной ситуации и обработкой этой ситуации.

1.1.12. Структура TCP-сервера

Организация TCP-сервера отличается от TCP-клиента в первую очередь созданием слушающего сокета (см. рис. 1.2 а). Такой сокет находится в состоянии listen и предназначен только для приёма входящих соединений. В случае прихода запроса на соединение создаётся дополнительный сокет, который и занимается обменом данными с клиентом. Типичная структура TCP-сервера и взаимосвязь сокетов изображена на рис. 1.2 а) и б).



а) Слушающий сокет б) Порождённый сокет

Рис. 1.2. Типичная структура TCP-сервера

1.1.13. Структура UDP-клиента

Структура UDP-клиента ещё более простая, чем у TCP-клиента, так как нет необходимости создавать и разрывать соединение. Варианты организации UDP-клиента изображены на рис. 1.3.

Наличие двух вариантов организации связано с возможностью в UDP-приложениях использовать вызов connect, устанавливающий значения по умолчанию для IP-адреса и порта сервера.

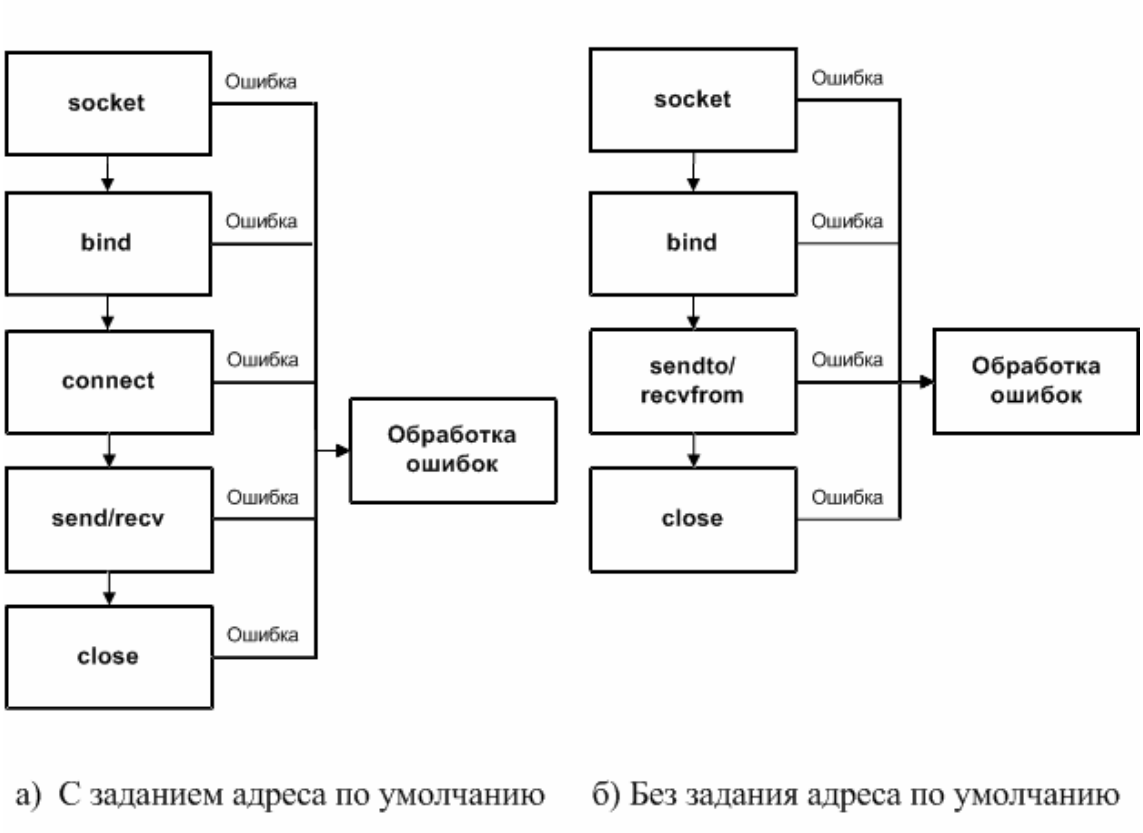


Рис. 1.3. Типичная структура UDP-клиента

1.1.14. Структура UDP-сервера

Ввиду того, что в протоколе UDP не устанавливается логический канал связи между клиентом и сервером, то для обмена данными между несколькими клиентами и сервером нет необходимости использовать со стороны сервера несколько сокетов. Для определения источника полученной дейтаграммы серверный сокет может использовать поля структуры from вызова recvfrom. Типичный способ организации UDP-сервера приведён на рис. 1.4.

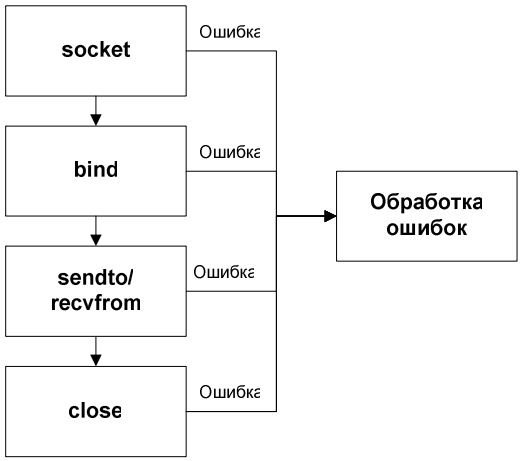


Рис. 1.4. Типичная структура UDP-сервера

1.2. Лабораторная работа №1. Типовые задания

Приведенные ниже задания являются базовыми для проведения первой части лабораторных работ по курсу «Технологии компьютерных сетей». Каждое задание может быть выполнено с использованием одного из двух протоколов: TCP или UDP, а также с различным выбором операционных систем для программы-клиента и программы-сервера. Все комбинации вариантов протоколов обмена и распределения клиент-серверных функций приведены в табл. 1.

Таблица 1. Варианты заданий

№	Сетевой протокол	Серверная операционная система	Клиентская операционная система
1	TCP	Linux	Windows
2	TCP	Windows	Linux
3	UDP	Linux	Windows
4	UDP	Windows	Linux

1.2.1. Система обмена сообщениями

Задание: разработать приложение-клиент и приложение сервер, обеспечивающие функции мгновенного обмена сообщений между пользователями.

Основные возможности. Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту от клиентов
3) Поддержка одновременной работы нескольких клиентов через механизм нитей
4) Передача текстового сообщения одному клиенту
5) Передача текстового сообщения всем клиентам
6) Прием и ретрансляция входящих сообщений от клиентов
7) Обработка запроса на отключение клиента
8) Принудительное отключение указанного клиента

Клиентское приложение должно реализовывать следующие функции:

1) Установление соединения с сервером
2) Передача сообщения всем клиентам
3) Передача сообщения указанному клиенту
4) Прием сообщения от сервера с последующей индикацией
5) Разрыв соединения
6) Обработка ситуации отключения клиента сервером

Настройки приложений. Разработанное клиентское приложение должно предоставлять пользователю настройку IP-адреса или доменного имени сервера сообщений и номера порта сервера.

Методика тестирования. Для тестирования приложений запускается один экземпляр серверного приложения и несколько экземпляров клиентских. В процессе тестирования проверяются основные возможности при-

ложений по передаче-приёму сообщений, ситуации самостоятельного и принудительного отключения клиента.

Источники: конспект лекций [1], [2]

1.2.2. Система передачи файлов

Задание: разработать приложение-клиент и приложение сервер, обеспечивающие функции обмена файлами.

Основные возможности. Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту от клиентов
3) Поддержка одновременной работы нескольких клиентов через механизм нитей
4) Приём файла от клиента
5) Передача по запросу клиента списка файлов текущего каталога
6) Приём запросов на передачу файла и передача файла клиенту
7) Навигация по системе каталогов
8) Обработка запроса на отключение клиента
9) Принудительное отключение клиента

Клиентское приложение должно реализовывать следующие функции:

1) Установление соединения с сервером
2) Получение от сервера списка файлов каталога
3) Операции навигации по системе каталогов
4) Передача файла серверу
5) Приём файла от сервера
6) Разрыв соединения
7) Обработка ситуации отключения клиента сервером

Настройки приложений. Разработанное клиентское приложение должно предоставлять пользователю настройку IP-адреса или доменного имени файл-сервера и номера порта, используемого сервером.

Разработанное серверное приложение должно предоставлять пользователю настройку корневого каталога для клиентских приложений.

Методика тестирования. Для тестирования приложений запускается файловый сервер и несколько клиентов. В процессе тестирования проверяются основные возможности приложения по передаче файлов и навигации по системе каталогов.

Источники: конспект лекций [1], [2]

1.2.3. Система электронной почты

Задание: разработать приложение-клиент и приложение сервер электронной почты.

Основные возможности. Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту от клиентов
3) Поддержка одновременной работы нескольких почтовых клиентов через механизм нитей
4) Приём почтового сообщения от одного клиента для другого
5) Хранение электронной почты для клиентов
6) Посылка клиенту почтового сообщения по запросу с последующим удалением сообщения
7) Посылка клиенту сведений о состоянии постового ящика
8) Обработка запроса на отключение клиента
9) Принудительное отключение клиента

Клиентское приложение должно реализовывать следующие функции:

1) Установление соединения с сервером

2) Передача электронного письма на сервер для другого клиента
3) Проверка состояния своего почтового ящика
4) Получение конкретного письма с сервера
5) Разрыв соединения
6) Обработка ситуации отключения клиента сервером

Настройки приложений. Разработанное клиентское приложение должно предоставлять пользователю настройку IP-адреса или доменного имени сервера электронной почты, номера порта, используемого сервером, и идентификационной информации пользователя.

Разработанное серверное приложение должно предоставлять пользователю настройку списка пользователей почтового сервера.

Методика тестирования. Для тестирования приложений запускается сервер электронной почты и несколько клиентов. В процессе тестирования проверяются основные возможности приложений по передаче и приёму сообщений.

Источники: конспект лекций [1], [2]

1.2.4. Информационная система

Задание: разработать распределённую информационную систему, состоящую из приложения-сервера и приложения-клиента. Информационная система является иерархическим хранилищем статей, каждая из которых состоит из названия, автора и текста статьи. Информационная система должна обеспечивать параллельный доступ к информации нескольким клиентам.

Основные возможности. Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту от клиентов

3) Поддержка одновременной работы нескольких клиентов через механизм нитей
4) Хранение иерархической структуры статей информационной системы
5) Передача пользователю списка текущих разделов системы, списка статей
6) Переход в конкретный раздел системы по запросу клиента
7) Возврат на предыдущий уровень по запросу клиента
8) Передача пользователю конкретной статьи по названию
9) Передача пользователю всех статей текущего раздела, принадлежащих определенному автору
10) Приём от клиента новой статьи и сохранение в информационной системе
11) Обработка запроса на отключение клиента
12) Принудительное отключение клиента

Клиентское приложение должно реализовывать следующие функции:

1) Установление соединения с сервером
2) Получение и печать списка подразделов и статей раздела
3) Передача команды на переход в конкретный раздел
4) Передача команды на переход в раздел на уровень выше
5) Получение конкретной статьи из информационной системы
6) Получение статей конкретного автора
7) Посылка новой статьи в систему
8) Разрыв соединения
9) Обработка ситуации отключения клиента сервером

Настройки приложений. Разработанное клиентское приложение должно предоставлять пользователю возможность введения идентификационной информации, настройки IP-адреса или доменного имени, а также номера порта сервера информационной системы.

Разработанное серверное приложение должно предоставлять пользователю возможность настройки начальной точки входа в информационную систему каждого пользователя.

Методика тестирования. Для тестирования приложений запускается сервер информационной системы и несколько клиентов. В процессе тестирования проверяются основные возможности приложений по передаче статей и навигации по разделам информационной системы.

Источники: конспект лекций [1], [2]

1.2.5. Калькулятор

Задание: разработать приложение-сервер «Удаленный калькулятор», позволяющее по запросу выполнять математические операции, и удаленный клиент для сервера.

Основные возможности. Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту от клиентов
3) Поддержка одновременной работы нескольких клиентов через механизм нитей
4) Приём «быстрых» операций с аргументами от клиента. Должны поддерживаться следующие операции: сложение, вычитание, умножение, деление.
5) Вычисление «долгих» математических операций (факториал, квадратный корень) с последующей отложенной посылкой результата клиенту (отдельная операция, инициируемая сервером).
6) Обработка запроса на отключение клиента
7) Принудительное отключение клиента

Клиентское приложение должно реализовывать следующие функции:

1) Установление соединения с сервером
2) Посылка операции с аргументами на вычисление
3) Получение результата вычислений «быстрых» операций
4) Получения результата вычислений «долгих» операций
5) Разрыв соединения
6) Обработка ситуации отключения клиента сервером

Настройки приложений. Разработанное клиентское приложение должно предоставлять пользователю настройку IP-адреса или доменного имени удалённого калькулятора и номера порта, используемого сервером.

Разработанное серверное приложение должно предоставлять пользователю настройку времени выполнения «долгих» операций.

Методика тестирования. Для тестирования приложений запускается сервер «Удаленного калькулятора» и несколько клиентов. В процессе тестирования проверяются основные возможности калькулятора по мгновенному и отложенному выполнению удалённых операций.

Источники: конспект лекций [1], [2]

1.2.6. Система дистанционного тестирования

Задание: разработать клиент-серверную систему дистанционного тестирования знаний, состоящую из централизованного сервера тестирования и клиентов тестирования.

Основные возможности. Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту от клиентов
3) Поддержка одновременной работы нескольких клиентов через механизм нитей
4) Регистрация клиента, выдача клиенту результата его последнего теста, выдача клиенту списка тестов

5) Получение от клиента номера теста
6) Последовательная выдача клиенту вопросов теста и получение ответов на вопросы
7) После прохождения теста – выдача клиенту его результата
8) Обработка запроса на отключение клиента
9) Принудительное отключение клиента

Клиентское приложение должно реализовывать следующие функции:

1) Установление соединения с сервером
2) Посылка регистрационных данных клиента
3) Выбор теста
4) Последовательная выдача ответов на вопросы сервера
5) Индикация результатов теста
6) Разрыв соединения
7) Обработка ситуации отключения клиента сервером

Настройки приложений. Разработанное клиентское приложение должно предоставлять пользователю настройку IP-адреса или доменного имени удалённого сервера тестов и номера порта, используемого сервером.

Разработанное серверное приложение должно хранить вопросы и правильные ответы нескольких тестов.

Методика тестирования. Для тестирования приложений запускается сервер «Удаленного тестирования» и несколько клиентов. В процессе тестирования проверяются основные возможности сервера по параллельному тестированию нескольких пользователей.

Источники: конспект лекций [1], [2]

1.2.7. Сетевой форум

Задание: разработать клиент-серверную систему сетевого форума, состоящую из сервера форума и пользовательских клиентов.

Основные возможности. Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту от клиентов
3) Поддержка одновременной работы нескольких клиентов через механизм нитей
4) Регистрация подключившегося клиента
5) Выдача клиенту перечня новых сообщений («постов») форума
6) Выдача клиенту иерархического представления форума
7) Прием от клиента сообщения в ветку форума
8) Выдача списка текущих активных пользователей форума
9) Обработка запроса на отключение клиента
10) Принудительное отключение клиента

Клиентское приложение должно реализовывать следующие функции:

1) Установление соединения с сервером
2) Посылка регистрационных данных клиента
3) Получение и вывод перечня новых сообщений
4) Получение и вывод иерархии форума
5) Выбор текущей ветки форума
6) Посылка сообщения в текущую ветку форума
7) Запрос текущих активных пользователей форума
8) Разрыв соединения
9) Обработка ситуации отключения клиента сервером

Настройки приложений. Разработанное клиентское приложение должно предоставлять пользователю настройку IP-адреса или доменного имени удалённого форум-сервера и номера порта, используемого сервером.

Разработанное серверное приложение должно хранить иерархию форума, а также историю появления сообщений в форуме.

Методика тестирования. Для тестирования приложений запускается сервер «Форум» и несколько клиентов. В процессе тестирования проверяются основные возможности сервера по параллельной работе на форуме нескольких клиентов.

Источники: конспект лекций [1], [2]

1.2.8. Сетевая игра «Рулетка»

Задание: разработать клиент-серверную игру, имитирующую сетевую игру «Рулетка», состоящую из игрового сервера и клиентов.

Основные возможности. Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту от клиентов
3) Поддержка одновременной работы нескольких клиентов через механизм нитей
4) Регистрация подключившегося клиента в качестве крупье (один) или в качестве игрока (несколько)
5) Выдача клиенту (крупье и игроку) списка ставок текущей игры (ставок других игроков)
6) Получение от клиента ставки: суммы ставки и типа ставки (чет, нечет, номер)
7) Получение от крупье команды на начало розыгрыша.
8) Уведомление всех клиентов о результате розыгрыша
9) Обработка запроса на отключение клиента
10) Принудительное отключение клиента

Клиентское приложение должно реализовывать следующие функции:

1) Установление соединения с сервером
2) Посылка регистрационных данных клиента (как крупье или как игрока)
3) Получение и вывод списка ставок
4) Для игрока: посылка своей ставки (сумма и ставки и тип ставки)
5) Для крупье: посылка команды начала розыгрыша
6) Получение и вывод результатов розыгрыша
7) Разрыв соединения
8) Обработка ситуации отключения клиента сервером

Настройки приложений. Разработанное клиентское приложение должно предоставлять пользователю настройку IP-адреса или доменного имени удалённого игрового сервера и номера порта, используемого сервером.

Разработанное серверное приложение должно хранить пароль клиента-крупье.

Методика тестирования. Для тестирования приложений запускается игровой сервер и несколько клиентов. В процессе тестирования проверяются основные возможности сервера по параллельной работе нескольких клиентов и одного крупье. Проверяется блокировка входа в игру второго крупье, попытки сделать повторную ставку и т.п.

Источники: конспект лекций [1], [2]

1.2.9. Сетевой сервер учета ошибок «Bug tracking»

Задание: разработать клиент-серверную систему регистрации, учета и управления ошибками в программных проектах. Система должна состоять из сервера, хранящего репозиторий ошибок и рабочих клиентских мест, позволяющих программистам и тестерам управлять ошибками, найденными в программных проектах.

Основные возможности. Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту от клиентов
3) Поддержка одновременной работы нескольких клиентов через механизм нитей
4) Регистрация подключившегося клиента в качестве тестера или разработчика
5) Выдача тестеру списка исправленных ошибок
6) Выдача тестеру списка активных (неисправленных) ошибок
7) Прием от тестера новой ошибки с указанием разработчика, проекта, идентификатора ошибки, текста ошибки
8) Прием от тестера команды о подтверждении или отклонении исправления ошибки
9) Выдача разработчику списка найденных в его проектах ошибок: идентификаторов и текстов
10) Прием команды разработчика об исправлении ошибки
11) Обработка запроса на отключение клиента
12) Принудительное отключение клиента

Клиентское приложение должно реализовывать следующие функции:

1) Установление соединения с сервером
2) Посылка регистрационных данных клиента (как тестера или как разработчика)
3) Для тестера: получение списка активных ошибок
4) Для тестера: получение списка исправленных ошибок
5) Для тестера: добавление новой ошибки
6) Для тестера: посылка команды подтверждения или отклонения исправления активной ошибки

7) Для разработчика: получение списка активных ошибок
8) Для разработчика: посылка команды исправления активной ошибки
9) Разрыв соединения
10) Обработка ситуации отключения клиента сервером

Настройки приложений. Разработанное клиентское приложение должно предоставлять пользователю настройку IP-адреса или доменного имени удалённого сервера учета ошибок и номера порта, используемого сервером.

Разработанное серверное приложение должно хранить списки тестеров и разработчиков.

Методика тестирования. Для тестирования приложений запускается сервер учета ошибок и несколько клиентов. В процессе тестирования проверяются основные возможности сервера по параллельной работе нескольких клиентов-тестеров и клиентов-разработчиков. Проверяется корректность изменения статуса ошибки разработчиком и тестером.

Источники: конспект лекций [1], [2]

1.2.10. Система терминального доступа

Задание: разработать клиент-серверную систему терминального доступа, позволяющую клиентам подсоединяться к серверу и выполнять элементарные команды операционной системы.

Основные возможности. Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту от клиентов
3) Поддержка одновременной работы нескольких терминальных клиентов через механизм нитей
4) Проведение аутентификации клиента на основе полученных имени пользователя и пароля

5) Выполнение команд пользователя:

- ls – выдача содержимого каталога
- cd – смена текущего каталога
- who – выдача списка зарегистрированных пользователей с указанием их текущего каталога
- kill – Привилегированная команда. Завершение сеанса другого пользователя.
- logout – выход из системы

6) Принудительное отключение клиента

Клиентское приложение должно реализовывать следующие функции:

1) Установление соединения с сервером
2) Посылка аутентификационных данных клиента (имя и пароль)
3) Посылка одной из команд (ls, cd, who, kill, logout) серверу
4) Получение ответа от сервера
5) Разрыв соединения
6) Обработка ситуации отключения клиента сервером или другим клиентом

Настройки приложений. Разработанное клиентское приложение должно предоставлять пользователю настройку IP-адреса или доменного имени удалённого терминального сервера и номера порта, используемого сервером.

Разработанное серверное приложение должно хранить аутентификационные данные для вех пользователей, а также списки разрешенных каждому пользователю команд.

Методика тестирования. Для тестирования приложений запускается терминальный сервер и несколько клиентов. В процессе тестирования проверяются основные возможности сервера по параллельной работе нескольких клиентов, имеющих различные привилегии (списки разрешенных ко-

манд). Проверяется корректность выполнения всех команда в различных ситуациях.

1.2.11. Электронный магазин

Задание: разработать приложение–клиент и приложение–сервер электронного магазина. Товар в электронном магазине имеет уникальный идентификатор, именование, цену.

Основные возможности. Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту от клиентов электронного магазина
3) Поддержка одновременной работы нескольких клиентов электронного магазина через механизм нитей
4) Прием запросов на добавление или покупку товара
5) Осуществление добавления товара, учет количества единиц товара
6) Передача клиенту электронного магазина информации о товарах и подтверждений о совершении покупки
7) Обработка запроса на отключение клиента
8) Принудительное отключение клиента

Клиентское приложение должно реализовывать следующие функции:

1) Установление соединения с сервером
2) Передача запросов о добавлении, покупке товаров серверу
3) Получение ответов на запросы от сервера
4) Разрыв соединения
5) Обработка ситуации отключения клиента сервером

Настройки приложений. Разработанное клиентское приложение должно предоставлять пользователю настройку IP–адреса или доменного

имени сервера электронного магазина и номера порта, используемого сервером.

Методика тестирования. Для тестирования приложений запускается сервер электронного магазина и несколько клиентов. В процессе тестирования проверяются основные возможности приложений по передаче и приему сообщений.

Источники: конспект лекций [1], [2]

1.2.12. Система торгов

Задание: разработать приложение–клиент и приложение–сервер системы торгов. На торги выставляются лоты, имеющие начальную стоимость. Участники торгов могут повышать стоимость лота. Распорядитель может прекратить торги. При окончании торгов всем участникам рассылаются результаты торгов.

Основные возможности. Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту от клиентов системы торгов (как распорядитель или участник торгов)
3) Поддержка одновременной работы нескольких клиентов системы торгов через механизм нитей
4) От участников торгов: прием запросов на передачу списка лотов
5) От участников торгов: прием запросов на повышение стоимости лота
6) От распорядителя: прием запроса на добавление нового лота с первоначальной стоимостью
7) От распорядителя: прием запроса об окончании торгов
8) Осуществление добавления лота, учет повышения стоимости лота участниками, завершение торгов и рассылка результатов торгов всем участникам

9) Обработка запроса на отключение клиента
10) Принудительное отключение клиента

Клиентское приложение должно реализовывать следующие функции:

1) Установление соединения с сервером
2) Регистрация в качестве распорядителя или участника
3) Участник: передача запроса о выводе списка лотов
4) Участник: передача запроса о повышении стоимости лота
5) Распорядитель: передача запроса о добавлении лота
6) Распорядитель: передача запроса о прекращении торгов
7) Получение результатов торгов от сервера
8) Разрыв соединения
9) Обработка ситуации отключения клиента сервером

Настройки приложений. Разработанное клиентское приложение должно предоставлять пользователю настройку IP–адреса или доменного имени сервера системы торгов и номера порта, используемого сервером.

Методика тестирования. Для тестирования приложений запускается сервер системы торгов и несколько клиентов. В процессе тестирования проверяются основные возможности приложений по передаче и приему сообщений.

Источники: конспект лекций [1], [2]

1.2.13. Система поиска/публикации новостей

Задание: разработать приложение–клиент и приложение–сервер системы поиска/публикации новостей. Новости сгруппированы по темам. Каждая новость имеет уникальный идентификатор, название и текст новости.

Основные возможности. Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта

2) Обработка запросов на подключение по этому порту от клиентов системы поиска/публикации новостей
3) Поддержка одновременной работы нескольких клиентов системы поиска/публикации новостей через механизм нитей
4) Прием запросов от клиента на передачу списка тем, списка новостей по теме, текста новости, добавление новости по теме
5) Осуществление добавления тем, новостей по темам
6) Передача списков тем, списков новостей и текстов новостей
7) Обработка запроса на отключение клиента
8) Принудительное отключение клиента

Клиентское приложение должно реализовывать следующие функции:

1) Установление соединения с сервером
2) Передача запросов на передачу списка тем, списка новостей по теме, текста новости, добавление новости по теме
3) Получение результатов от сервера
4) Разрыв соединения
5) Обработка ситуации отключения клиента сервером

Настройки приложений. Разработанное клиентское приложение должно предоставлять пользователю настройку IP–адреса или доменного имени сервера системы поиска/публикации новостей и номера порта, используемого сервером.

Методика тестирования. Для тестирования приложений запускается сервер системы поиска/публикации новостей и несколько клиентов. В процессе тестирования проверяются основные возможности приложений по передаче и приему сообщений.

Источники: конспект лекций [1], [2]

1.2.14. Платежная система

Задание: разработать приложение–клиент и приложение–сервер платежной системы. Участники платежной системы имеют электронные кошельки. Электронный кошелек имеет уникальный номер. При регистрации пользователя в платежной системе на его счет зачисляется определенная сумма. Пользователя платежной системы могут осуществлять платежи друг другу через приложение–сервер.

Основные возможности. Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту от клиентов платежной системы
3) Поддержка одновременной работы нескольких клиентов платежной системы через механизм нитей
4) Прием запросов от клиента на регистрацию пользователя, передачу списка электронных кошельков пользователей платежной системы, осуществление платежей одного пользователя другому, проверка состояния счета кошелька
5) Осуществление добавления пользователя в платежную систему, хранение и изменение состояния электронных кошельков в зависимости от платежей пользователей
6) Передача запросов на платежи от одного пользователя другому, подтверждений платежей, номера нового кошелька при регистрации пользователя, списка электронных кошельков
7) Обработка запроса на отключение клиента
8) Принудительное отключение клиента

Клиентское приложение должно реализовывать следующие функции:

1) Установление соединения с сервером

2) Передача запросов на передачу списка электронных кошельков пользователей платежной системы, платежи одного пользователя другому, проверку состояния счета кошелька
3) Получение от сервера запросов на платеж от другого пользователя, результатов платежа
4) Разрыв соединения
5) Обработка ситуации отключения клиента сервером

Настройки приложений. Разработанное клиентское приложение должно предоставлять пользователю настройку IP–адреса или доменного имени сервера платежной системы и номера порта, используемого сервером.

Методика тестирования. Для тестирования приложений запускается сервер платежной системы и несколько клиентов. В процессе тестирования проверяются основные возможности приложений по передаче и приему сообщений.

Источники: конспект лекций [1], [2]

1.2.15. Служба каталогов x.500 (LDAP)

Задание: разработать приложение–клиент и приложение–сервер службы каталогов. Всякая запись в каталоге состоит из одного или нескольких атрибутов и обладает уникальным именем. Уникальное имя состоит из одного или нескольких относительных уникальных имен, разделённых запятой (например, “ `cn=Users, dc=myserver, dc=myprovider, dc=ru` ”). Относительное уникальное имя имеет вид `имяАтрибута=значение`. Значение атрибута отделяется от имени атрибута двоеточием (например, “ `objectClass: top` ”). На одном уровне каталога не может существовать двух записей с одинаковыми относительными уникальными именами. В силу такой структуры уникального имени записи в каталоге можно легко представить в виде дерева. Первая строка каждого блока определяет так назы-

ваемый узел поиска. От этого узла производится поиск. Пример трех записей:

```
dn: cn=Users, dc=myserver, dc=myprovider, dc=ru
objectClass: top

dn: cn=ldapuser1, cn=Users, dc=myserver, dc=myprovider, dc=ru
objectClass: posixAccount
cn: LDAP User 1
uid: ldapuser1
uidNumber: 1001
gidNumber: 10
homeDirectory: /home/ldapuser1
loginShell: /bin/bash

dn: cn=ldapuser2, cn=Users, dc=myserver, dc=myprovider, dc=ru
objectClass: posixAccount
cn: LDAP User 2
uid: ldapuser2
uidNumber: 1002
gidNumber: 10
homeDirectory: /home/ldapuser2
loginShell: /bin/bash
```

Основные возможности. Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту от клиентов службы каталогов
3) Поддержка одновременной работы нескольких клиентов службы каталогов через механизм нитей
4) Прием запросов на поиск, добавление и удаление записей службы каталогов
5) Осуществление поиска, добавления, удаления записей службы каталогов

6) Передача клиенту записей службы каталогов и подтверждений о вставке и удалении записей
7) Обработка запроса на отключение клиента
8) Принудительное отключение клиента

Клиентское приложение должно реализовывать следующие функции:

1) Установление соединения с сервером
2) Передача запросов о поиске, добавлении, удалении записей серверу
3) Получение ответов на запросы от сервера
4) Разрыв соединения
5) Обработка ситуации отключения клиента сервером

Настройки приложений. Разработанное клиентское приложение должно предоставлять пользователю настройку IP–адреса или доменного имени сервера службы каталогов и номера порта, используемого сервером.

Методика тестирования. Для тестирования приложений запускается сервер службы каталогов и несколько клиентов. В процессе тестирования проверяются основные возможности приложений по передаче и приему сообщений.

Источники: конспект лекций [1], [2], [15]

1.2.16. Система голосования/рейтингов

Задание: Разработать распределенную систему, состоящую из приложений клиента и сервера, для формирования рейтингов на заданную тему. Подразумевается использование одноуровневой системы рейтингов (без сложной вложенной иерархии): имеется набор тем, в каждой из тем - некоторое заданное фиксированное количество альтернатив выбора, за каждую из которых можно отдать некоторое количество голосов. Информационная система должна обеспечивать параллельную работу нескольких клиентов.

Основные возможности:

Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту клиентов
3) Поддержка одновременной работы нескольких клиентов с использованием механизма нитей и средств синхронизации доступа к разделяемым между нитями ресурсам.
4) Принудительное отключение конкретного клиента
5) Добавление новой темы рейтинга (с указанием количества альтернатив выбора)
6) Удаление темы рейтинга вместе со всеми данными
7) Добавление альтернативы выбора в рейтинг (если количество альтернатив меньше заданного)
8) Открытие/закрытие голосования по теме
9) Учет голосов за конкретную альтернативу в конкретном рейтинге
10) Выдача пользователю списка имеющихся тем рейтингов.
11) Выдача пользователю состояния конкретного рейтинга: закрыт/открыт, таблицу рейтинга в абсолютном и относительном выражении
12) * Сохранение состояния при выключении сервера

Клиентское приложение должно реализовывать следующие функции:

1) Возможность параллельной работы нескольких клиентов с одного или разных IP-адресов
2) Установление соединения с сервером (возможно, с регистрацией на сервере)
3) Разрыв соединения
4) Обработка ситуации отключения сервером
5) Получение и вывод списка тем рейтингов

6) Получение и вывод состояния конкретного рейтинга
7) Передача команды на создание темы рейтинга
8) Передача команды на удаление темы рейтинга
9) Передача команды на добавление альтернативы в рейтинг
10) Передача команды на открытие/закрытие голосования по теме рейтинга
11) Подача нескольких голосов за конкретную альтернативу конкретного рейтинга

Настройка приложений. Разработанное клиентское приложение должно предоставлять пользователю возможность задания IP-адреса или доменного имени сервера, а также номера порта сервера.

Тестирование. Для тестирования запускается сервер рейтинговой системы и несколько клиентов. В процессе тестирования проверяются основные функциональные возможности разработанной рейтинговой системы.

Источники: конспект лекций [1], [2]

1.2.17. Система публикации и поиска вакансий

Задание: Разработать распределенную систему, состоящую из приложений клиента и сервера, для ведения базы вакансий. Подразумевается использование одноуровневой системы вакансий (без сложной вложенной иерархии): имеется набор специальностей, в каждую из них можно подать вакансию. Кроме подачи вакансий должна осуществляться возможность многокритериального поиска по вакансиям. Информационная система должна обеспечивать параллельную работу нескольких клиентов.

Основные возможности:

Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту клиентов

3) Поддержка одновременной работы нескольких клиентов с использованием механизма нитей и средств синхронизации доступа к разделяемым между нитями ресурсам.
4) Принудительное отключение конкретного клиента
5) Добавление новой специальности
6) Удаление специальности вместе со всеми поданными вакансиями
7) Добавление вакансии (ID, специальность, компания, должность, границы возраста, заработная плата)
8) Удаление вакансии (по ID)
9) Выдача пользователю списка специальностей.
10) Выдача пользователю списка вакансий, удовлетворяющего запросу на поиск (специальность, возраст, должность, заработная плата)
11) *Сохранение состояния при выключении сервера

Клиентское приложение должно реализовывать следующие функции:

1) Возможность параллельной работы нескольких клиентов с одного или разных IP-адресов
2) Установление соединения с сервером (возможно, с регистрацией на сервере)
3) Разрыв соединения
4) Обработка ситуации отключения сервером
5) Получение и вывод списка специальностей
6) Передача команды на добавление специальности
7) Передача команды на удаление специальности
8) Передача команды на добавление вакансии
9) Получение и вывод списка вакансий по условиям поиска (при этом необходимо организовать поиск по полному и неполному набору критериев)

Настройка приложений. Разработанное клиентское приложение должно предоставлять пользователю возможность задания IP-адреса или доменного имени сервера, а также номера порта сервера.

Тестирование. Для тестирования запускается сервер системы управления вакансиями и несколько клиентов. В процессе тестирования проверяются основные функциональные возможности разработанной информационной системы.

Источники: конспект лекций [1], [2]

1.2.18. Система оповещения о событиях и подписки на них

Задание. Разработать распределенную систему, состоящую из приложений клиента и сервера, для распределенного оповещения о событиях по принципу подписки. Подразумевается разовая или многократная генерация событий сервером (с некоторым заданным интервалом), с информированием подписанных клиентов об этих событиях. Информационная система должна обеспечивать параллельную работу нескольких клиентов.

Основные возможности:

Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту клиентов
3) Поддержка одновременной работы нескольких клиентов с использованием механизма нитей и средств синхронизации доступа к разделяемым между нитями ресурсам.
4) Принудительное отключение конкретного клиента
5) Добавление нового события (однократного или многократного с указанием периода генерации)
6) Удаление события
7) Генерация многократных событий с заданным интервалом по таймеру

8) Генерация однократных событий по команде от клиента
9) Выдача пользователю списка имеющихся событий
10) Получение и обработка команд подписки/снятия с подписки на события
11) Рассылка уведомлений о событиях клиентам с фильтрацией по факту подписки
12) * Сохранение состояния при выключении сервера – списка событий и расписания многократных событий

Клиентское приложение должно реализовывать следующие функции:

1) Возможность параллельной работы нескольких клиентов с одного или нескольких IP-адресов
2) Установление соединения с сервером (возможно, с регистрацией на сервере)
3) Разрыв соединения
4) Обработка ситуации отключения сервером
5) Получение и вывод списка событий
6) Передача команды на добавление события (с указанием типа и периода для многократных событий)
7) Передача команды на удаление события
8) Передача команды на генерацию однократного события
9) Передача команды подписки на событие
10) Передача команды отмены подписки на событие
11) Вывод реакции на наступление событий, на которые клиент подписан

Настройка приложений. Разработанное клиентское приложение должно предоставлять пользователю возможность задания IP-адреса или доменного имени сервера, а также номера порта сервера.

Тестирование. Для тестирования запускается сервер системы подписки на события и несколько клиентов. В процессе тестирования проверяются основные функциональные возможности разработанной системы.

Источники: конспект лекций [1], [2]

1.2.19. Система выставления курсов валют

Задание: Разработать распределенную систему, состоящую из приложений клиента и сервера, для распределенного выставления/просмотра курсов валют. Информационная система должна обеспечивать параллельную работу нескольких клиентов.

Основные возможности:

Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту клиентов
3) Поддержка одновременной работы нескольких клиентов с использованием механизма нитей и средств синхронизации доступа к разделяемым между нитями ресурсам.
4) Принудительное отключение конкретного клиента
5) Добавление новой валюты (кода валюты)
6) Удаление валюты
7) Добавление курса конкретной валюты
8) Выдача пользователю списка имеющихся валют с текущими курсами и абсолютными/относительными приращениями к предыдущим значениям
9) Выдача пользователю истории курса конкретной валюты
10) * Сохранение состояния при выключении сервера

Клиентское приложение должно реализовывать следующие функции:

1) Возможность параллельной работы нескольких клиентов с одного или нескольких IP-адресов
2) Установление соединения с сервером (возможно, с регистрацией на сервере)
3) Разрыв соединения
4) Обработка ситуации отключения сервером
5) Получение и вывод списка валют с котировками/изменениями
6) Передача команды на добавление валюты
7) Передача команды на удаление валюты
8) Передача команды на добавление курса валюты
9) Получение и вывод истории котировок валюты

Настройка приложений. Разработанное клиентское приложение должно предоставлять пользователю возможность задания IP-адреса или доменного имени сервера, а также номера порта сервера.

Тестирование. Для тестирования запускается сервер системы котировок валют и несколько клиентов. В процессе тестирования проверяются основные функциональные возможности разработанной системы.

Источники: конспект лекций [1], [2]

1.2.20. Система распределенных математических расчетов

Задание: Разработать распределенную систему, состоящую из приложений клиента и сервера, для распределенного расчета простых чисел. Информационная система должна обеспечивать параллельную работу нескольких клиентов.

Основные возможности:

Серверное приложение должно реализовывать следующие функции:

1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту клиентов

3) Поддержка одновременной работы нескольких клиентов с использованием механизма нитей и средств синхронизации доступа к разделяемым между нитями ресурсам.
4) Принудительное отключение конкретного клиента
5) Хранение рассчитанных клиентами простых чисел, а также текущей нижней границы диапазона для нового запроса на расчет
6) Выдача клиентам максимального рассчитанного простого числа
7) Выдача клиентам последних N рассчитанных простых чисел
8) Выдача клиентам необходимого диапазона расчета чисел
9) * Сохранение состояния при выключении сервера

Клиентское приложение должно реализовывать следующие функции:

1) Возможность параллельной работы нескольких клиентов с одного или нескольких IP-адресов
2) Установление соединения с сервером (возможно, с регистрацией на сервере)
3) Разрыв соединения
4) Обработка ситуации отключения сервером
5) Получение от сервера и вывод максимально рассчитанного простого числа
6) Получение от сервера и вывод последних N рассчитанных простых чисел
7) Получение от сервера диапазона расчета простых чисел (нижнюю грань выдает сервер, количество проверяемых чисел - клиент)
8) Расчет простых чисел в требуемом диапазоне (имеет смысл проверять остатки от деления на все нечетные числа в пределах $sqrt(N_{max})+1$)
9) Передача серверу набора рассчитанных простых чисел

Настройка приложений. Разработанное клиентское приложение должно предоставлять пользователю возможность задания IP-адреса или доменного имени сервера, а также номера порта сервера.

Тестирование. Для тестирования запускается сервер системы расчета простых чисел и несколько клиентов. В процессе тестирования проверяются основные функциональные возможности разработанной системы.

Источники: конспект лекций [1], [2]

2. ИЗУЧЕНИЕ ПРИКЛАДНЫХ ПРОТОКОЛОВ TCP/IP

В первом разделе был представлен низкоуровневый способ организации сетевых приложений на основе программирования BSD-сокетов. В реальных условиях не всегда необходима столь глубокая детализация описания, чаще требуются более высокоуровневые программные интерфейсы, упрощающие проектирование TCP- и UDP-приложений.

Данный раздел посвящен реализации прикладных протоколов на уровне обработки команд прикладных протоколов. В качестве механизма реализации обработки команд используется свободно распространяемая компонентная библиотека Internet Direct версии 9 (Indy9) [5]. Эта библиотека содержит широкий набор компонентов для реализации клиентских и серверных приложений, использующих протоколы TCP и UDP, и реализующих все необходимые функции по подготовке, передаче, получению и обработке данных.

2.1. Разработка клиента прикладного протокола

2.1.1. Общие положения

Для организации клиентского программного обеспечения библиотека Internet Direct предоставляет программисту компонент TIdTCPClient, инкапсулирующий в себе все возможности, необходимые для построения полноценного сетевого приложения-клиента.

2.1.2. Установление соединения

Библиотека предоставляет два способа организации клиентского соединения: статический и динамический.

Статический способ заключается в размещении во время разработки (в design time) на форме (наследнике класса TForm) или в модуле данных (наследнике класса TDataModule) экземпляра компонента TIdTCPClient. Па-

раметры организуемого соединения задаются статически свойствами компонента в инспекторе объектов (Object Inspector).

Динамический способ создания соединения заключается в создании во время исполнения (в run time) экземпляра объекта TIdTCPClient. Параметры организуемого соединения задаются с помощью вызова методов объекта и задания значений соответствующих свойств.

2.1.3. Задание параметров соединения

Параметры организуемого соединения задаются с помощью задания свойств компонента TIdTCPClient.

Для задания параметров *удаленного серверного приложения* используются свойства Host и Port:

- свойство ***Host*** задаёт в текстовом виде IP-адрес или доменное имя узла сети, на котором функционирует серверное приложение.
- свойство ***Port*** определяет в TCP-порт, используемый серверным приложением.

Для задания *привязки клиентского приложения* к конкретным IP-адресам и TCP-портам используются следующие свойства:

- ***BoundIP*** – задаёт в текстовом виде IP-адрес используемый клиентским приложением. Пустая строка обозначает использование любого адреса, имеющегося у данного сетевого узла.
- ***BoundPort*** – задаёт номер TCP-порта, выделяемый для клиентского приложения. Значение 0 обозначает использование номера из диапазона свободных непривилегированных портов, предлагаемых операционной системой.
- ***BoundPortMax*** – верхняя граница диапазона непривилегированных портов, предлагаемых операционной системой (в случае, когда нулевого значения свойства BoundIP)

- ***BoundPortMin*** – нижняя граница диапазона непривилегированных портов, предлагаемых операционной системой (в случае, нулевого значения свойства BoundIP)

2.1.4. Установление и разрыв соединения

Для установления соединения с удаленным приложением используется метод Connect:

```
procedure Connect(const ATimeout: Integer = IdTimeoutDefault);
```

Параметр – предельное время установления соединения, по умолчанию – время не ограничено.

Для разрыва установленного соединения используется метод Disconnect:

```
procedure Disconnect;
```

Для установления соединения и получения всех данных из сокета (например, при чтении файла в протоколе FTP) используется метод ConnectAndGetAll:

```
function ConnectAndGetAll: string;
```

2.1.5. Посылка RFC-команд

Посылка команд прикладных протоколов (команд RFC) осуществляется вызовом метода SendCmd:

```
function SendCmd(const AOut: string;
         const AResponse: SmallInt = -1): SmallInt;
function SendCmd(const AOut: string;
         const AResponse: array of SmallInt): SmallInt;
```

Параметр AOut задаёт команду протокола, параметр AResponse – возможный ответ или список ответов сервера.

2.1.6. Получение данных

Для чтения данных компонент имеет набор методов и свойств, реализующих получение типизированных и нетипизированных данных от серверной стороны:

1) для получения целочисленных значений используются методы ReadInteger, ReadSmallInt, ReadCardinal:

```
function ReadInteger(const AConvert: boolean = True): Integer;
function ReadCardinal(const AConvert: boolean = True): Cardinal;
function ReadSmallInt(const AConvert: boolean = True): SmallInt;
```

2) Для получения символьных и строковых значений используются методы ReadChar, ReadString, ReadLn, ReadLnWait и свойство AllData:

```
function ReadChar: Char;
function ReadLn(ATerminator: string = LF;
         const ATimeout: Integer = IdTimeoutDefault;
         AMaxLineLength: Integer = -1): string;
function ReadLnWait(AFailCount: Integer = MaxInt): string;
function ReadString(const ABytes: Integer): string;
property AllData: string;
```

3) Для получения буферов, потоков данных и списков строк используются методы ReadBuffer, ReadStream и ReadStrings:

```
procedure ReadBuffer(var ABuffer; const AByteCount: Longint);
procedure ReadStream(AStream: TStream; AByteCount: LongInt = -1;
          const AReadUntilDisconnect: boolean = False);
procedure ReadStrings(ADest: TStrings; AReadLinesCount: Integer = -1);
```

2.1.7. Посылка данных

Для записи данных в сокет компонент имеет набор методов, реализующих посылку типизированных и нетипизированных данных серверной стороне:

1) для посылки целочисленных значений используются методы WriteInteger, WriteSmallInt и WriteCardinal:

```
procedure WriteInteger(AValue: Integer;
                       const AConvert: Boolean = True);
procedure WriteCardinal(AValue: Cardinal;
                        const AConvert: Boolean = True);
procedure WriteSmallInt(AValue: SmallInt;
                        const AConvert: Boolean = True);
```

2) Для посылки символьных и строковых значений используются методы WriteChar, Write и WritelLn:

```
procedure WriteChar(AValue: Char);
procedure Write(const AOut: string);
procedure WriteLn(const AOut: string = '');
```

3) Для посылки буферов, потоков данных, файлов и списков строк используются методы WriteBuffer, WriteStream, WriteFile и WriteStrings:

```
procedure WriteBuffer(const ABuffer; AByteCount: Longint;
                      const AWriteNow: Boolean = False);
procedure WriteStream(AStream: TStream;
                      const AAll: Boolean = True;
                      const AWriteByteCount: Boolean = False;
                      const ASize: Integer = 0);
function WriteFile(const AFile: string;
                   const AEnableTransferFile: Boolean = False):Cardinal;
procedure WriteStrings(AValue: TStrings;
                       const AWriteLinesCount: Boolean = False);
```

4) Для посылки списков строк в формате RFC и стандартного ответа RFC используются методы WriteHeader, WriteRFCStrings и WriteRFCReply:

```
procedure WriteHeader(AHeader: TStrings);
procedure WriteRFCStrings(AStrings: TStrings);
procedure WriteRFCReply(AReply: TIdRFCReply);
```

2.1.8. Обработка ответов сервера

Обработка ответов сервера заключается в анализе результата посланной команды и в обработке возникающих событий:

1) Событие OnConnected. Возникает после успешного установления соединения клиентом.
2) Событие OnDisconnected возникает после разрыва соединения.
3) Свойство LastCmdResult содержит текстовый и числовой результат выполнения последней команды.

2.1.9. Лабораторная работа №2. Варианты заданий

2.1.9.1 Клиент протокола POP-3

Разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции клиента протокола POP-3.

Основные возможности. Приложение должно реализовывать следующие функции:

1) Подключение к указанному серверу по IP-адресу или доменному имени
2) Получение состояния ящика (количество новых писем, их суммарная длина)
3) Получение списка заголовков всех новых писем сервера без предварительной загрузки
4) Загрузка всех новых или конкретных выбранных писем
5) Пометка конкретных писем для последующего удаления
6) Удаление помеченных писем
7) Выход из приложения без удаления помеченных писем
8) Подробное протоколирование соединения сервера с клиентом

Поддерживаемые команды. Разработанное приложение должно реализовывать следующие команды протокола POP-3:

- USER – передача серверу идентификационной информации пользователя
- PASS – передача серверу пароля пользователя
- STAT – получение состояния почтового ящика
- LIST – получение списка сообщения почтового ящика
- RETR – получение сообщения
- DELE – пометка сообщения на удаление
- TOP – получение первых нескольких строк сообщения
- UIDL – получение уникального идентификатора сообщения

- RSET – сброс всех пометок на удаление сообщений
- QUIT – удаление всех помеченных сообщений и завершение сеанса

Настройки приложения. Разработанное приложение должно предоставлять пользователю настройку следующих параметров:

1) IP-адрес или доменное имя почтового сервера
2) Номер порта сервера (по умолчанию - 110)
3) Имя пользователя
4) Пароль пользователя

Методика тестирования. Для тестирования приложения следует использовать почтовые серверы, имеющиеся в лаборатории, а также бесплатные почтовые серверы, имеющиеся в сети Internet (http://www.mail.ru, http://www.yandex.ru, http://www.rambler.ru и т.п.).

Штатными клиентами электронной почты (Mozilla Thunderbird, MS Outlook Express, The Bat, Netscape Messenger) на сервер помещаются тестовые сообщения. В процессе тестирования проверяются основные возможности приложения.

Источники: конспект лекций [1], [6], RFC [7].

2.1.9.2 Клиент протокола SMTP

Разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции клиента протокола SMTP.

Основные возможности. Приложение должно реализовывать следующие функции:

1) Создание нового письма, включающего такие поля, как **From** (отправитель), **To** (получатель), **Subject** (тема), **Carbon copy** (дополнительные адресаты), **Blind copy** (дополнительные скрытые адресаты), **Body** (текст)
2) Формирование всех необходимых заголовков письма, с тем, чтобы приёмная сторона не рассматривала данное письмо как спам.

3) Подключение к указанному SMTP-серверу и отсылка созданного письма
4) Подробное протоколирование соединения клиента с сервером

Поддерживаемые команды. Разработанное приложение должно реализовывать следующие команды протокола SMTP:

- HELO – передача серверу информации о домене пользователя
- MAIL FROM – передача серверу адреса отправителя письма
- RCPT TO – передача серверу адреса получателя письма
- DATA – передача серверу тела письма
- QUIT – завершение сеанса связи

Настройки приложения. Разработанное приложение должно обеспечивать настройку следующих параметров:

1) Собственное доменное имя для передачи в команде HELO
2) Адрес отправителя
3) IP-адрес или доменное имя сервера SMTP

Методика тестирования. Для тестирования приложения следует использовать почтовые серверы, имеющиеся в лаборатории, а также бесплатные почтовые серверы, имеющиеся в сети Internet (http://www.mail.ru, http://www.yandex.ru, http://www.rambler.ru и т.п.).

Средствами разработанного приложения осуществляется передача письма на указанные ящики электронной почты. Штатными клиентами электронной почты проверяется корректность доставки почты и правильность параметров письма.

Источники: конспект лекций [1]по курсу «Технологии компьютерных сетей», [6], RFC ([8, 9, 10, 11])

2.1.9.3 Клиент протокола FTP, работающий с сервером в активном режиме

Разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции клиента протокола FTP.

Основные возможности. Приложение должно реализовывать следующие функции:

1) Подключение к указанному серверу
2) Получение списка файлов в каталоге
3) Навигация по системе каталогов
4) Копирование файла на сервер
5) Копирование файла с сервера
6) Создание и удаление каталогов
7) Удаление файлов
8) Обеспечение работы со ссылками на файлы и каталоги
9) Поддержка передачи в бинарном и текстовом режиме
10) Протоколирование соединения сервера с клиентом

Поддерживаемые команды. Разработанное приложение должно реализовывать следующие команды протокола FTP:

- USER – передача серверу идентификационной информации пользователя
- PASS – передача серверу пароля пользователя
- CWD – смена текущего каталога сервера
- MKD – создание каталога
- RMD – удаление каталога
- DELE – удаление фала на сервере
- PORT – передача на сервер параметров (адреса и порта) сокета, осуществляющего приём и передачу данных

- LIST – получение списка файлов в текущем каталоге сервера в расширенном формате
- NLST – получение списка файлов в текущем каталоге сервера в сокращённом формате
- RETR – получение файла с сервера
- STOR – посылка файла на сервер
- TYPE – задание режима передачи данных
- QUIT – завершение сеанса

Настройки приложения. Разработанное приложение должно обеспечивать настройку следующих параметров:

1) IP-адрес или доменное имя файлового сервера
2) Имя пользователя
3) Пароль пользователя
4) Режим передачи: текстовый / бинарный

Методика тестирования. Для тестирования приложения следует использовать файловые серверы, имеющиеся в лаборатории, а также открытые файловые серверы, имеющиеся в сети Internet (ftp://ftp.funet.fi, ftp://ftp.relcom.ru и т.п.).

Для разработанных приложений проверяются возможности подсоединения к серверу, копирования файла на сервер, копирования файла с сервера, удаления файла на сервере, переименовании файла на сервере, создания каталога, удаления каталога.

Источники: конспект лекций [1], [6], RFC [12].

2.1.9.4 Клиент протокола FTP, работающий с сервером в пассивном режиме

Разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции клиента протокола FTP.

Основные возможности. Приложение должно реализовывать следующие функции:

1) Подключение к указанному серверу
2) Получение списка файлов в каталоге
3) Навигация по системе каталогов
4) Копирование файла на сервер
5) Копирование файла с сервера
6) Создание и удаление каталогов
7) Удаление файлов
8) Обеспечение работы со ссылками на файлы и каталоги
9) Поддержка передачи в бинарном и текстовом режиме
10) Протоколирование соединения сервера с клиентом

Поддерживаемые команды. Разработанное приложение должно реализовывать следующие команды протокола FTP:

- USER – передача серверу идентификационной информации пользователя
- PASS – передача серверу пароля пользователя
- CWD – смена текущего каталога сервера
- MKD – создание каталога
- RMD – удаление каталога
- DELE – удаление фала на сервере
- PASV – переключение сервера в пассивный режим
- LIST – получение списка файлов в текущем каталоге сервера в расширенном формате
- NLST – получение списка файлов в текущем каталоге сервера в сокращённом формате
- RETR – получение файла с сервера
- STOR – посылка файла на сервер

- QUIT – завершение сеанса

Настройки приложения. Разработанное приложение должно обеспечивать настройку следующих параметров:

1) IP-адрес или доменное имя файлового сервера
2) Имя пользователя
3) Пароль пользователя
4) Режим передачи: текстовый / бинарный

Методика тестирования. Для тестирования приложения следует использовать файловые серверы, имеющиеся в лаборатории, а также открытые файловые серверы, имеющиеся в сети Internet (ftp://ftp.funet.fi, ftp://ftp.relcom.ru и т.п.).

Для разработанных приложений проверяется возможности подсоединения к серверу, копирования файла на сервер, копирования файла с сервера, удаления файла на сервере, переименовании файла на сервере, создания каталога, удаления каталога.

Источники: конспект лекций [1], [6], RFC [12].

2.1.9.5 Сборщик электронной почты

Разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции сборщика почты по протоколу POP-3 с нескольких почтовых серверов.

Основные возможности. Приложение должно реализовывать следующие функции:

1) Обработка запроса на подключения клиента
2) Подключение к указанным серверам
3) Получение списка всех новых писем всех почтовых серверов без их предварительной загрузки
4) Загрузка всех новых или конкретных выбранных писем
5) Протоколирование соединения клиента с серверами

Поддерживаемые команды. Разработанное приложение должно реализовывать следующие команды протокола POP-3:

- USER – передача серверу идентификационной информации пользователя
- PASS – передача серверу пароля пользователя
- STAT – получение состояния почтового ящика
- LIST – получение списка сообщения почтового ящика
- RETR – получение сообщения
- TOP – получение первых нескольких строк сообщения
- UIDL – получение уникального идентификатора сообщения
- QUIT – удаление всех помеченных сообщений и завершение сеанса

Настройки приложения. Разработанное приложение должно обеспечивать настройку следующих параметров:

1) IP-адреса или доменные имена всех почтовых серверов
2) Номера портов всех серверов (по умолчанию - 110)
3) Аутентификационную информацию пользователя для всех серверов (имя пользователя и пароль)

Методика тестирования. Для тестирования приложения следует использовать почтовые серверы, имеющиеся в лаборатории, а также бесплатные почтовые серверы, имеющиеся в сети Internet (http://www.mail.ru, http://www.yandex.ru, http://www.rambler.ru и т.п.).

Штатными клиентами электронной почты на серверы помещаются тестовые сообщения. В процессе тестирования проверяются основные возможности приложения.

Источники: конспект лекций [1], [6], RFC [7].

2.1.9.6 Программа, обеспечивающая закачку всех файлов FTP- каталога

Разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции получения содержимого указанного каталога сервера FTP и всех его подкаталогов.

Основные возможности. Приложение должно реализовывать следующие функции:

1) Подключение к указанному FTP-серверу
2) Выбор режима работы: активный или пассивный
3) Выбор каталога для закачивания
4) Копирование всех файлов каталога и рекурсивно всех его подкаталогов с сервера
5) Обеспечение корректности работы со ссылками на файлы и каталоги
6) Протоколирование соединения сервера с клиентом

Поддерживаемые команды. Разработанное приложение должно реализовывать следующие команды протокола FTP:

- USER – передача серверу идентификационной информации пользователя
- PASS – передача серверу пароля пользователя
- CWD – смена текущего каталога сервера
- PORT – передача на сервер параметров (адреса и порта) сокета, осуществляющего приём и передачу данных
- PASV – переключение сервера в пассивный режим
- LIST – получение списка файлов в текущем каталоге сервера в расширенном формате
- NLST – получение списка файлов в текущем каталоге сервера в сокращённом формате
- RETR – получение файла с сервера

- STOR – посылка файла на сервер
- QUIT – завершение сеанса

Настройки приложения. Разработанное приложение должно обеспечивать настройку следующих параметров:

1) IP-адрес или доменное имя файлового сервера
2) Имя пользователя
3) Пароль пользователя

Методика тестирования. Для тестирования приложения следует использовать файловые серверы, имеющиеся в лаборатории, а также открытые файловые серверы, имеющиеся в сети Internet (ftp://ftp.funet.fi, ftp://ftp.relcom.ru и т.п.). Для разработанных приложений проверяются возможности подсоединения к серверу, выбора требуемого каталога, копирования содержимого каталога и его подкаталогов на компьютер-клиент, корректность работы в активном и пассивном режимах.

Источники: конспект лекций [1], [6], RFC [12].

2.1.9.7 Программа – загрузчик содержимого Web-сайтов

Разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции получения содержимого Web-сайта.

Основные возможности. Приложение должно реализовывать следующие функции:

1) Подключение к указанному HTTP-серверу
2) Разбор кода HTML-страниц и получение списка ссылок на медиа-элементы и другие страницы
3) Копирование в локальный каталог заданного числа уровней вложенности страниц Web-сайта
4) Протоколирование соединения сервера с клиентом

Настройки приложения. Разработанное приложение должно обеспечивать настройку следующих параметров:

1) имя Web-сервера
2) глубину вложенности страниц, которую следует загрузить

Методика тестирования. Для тестирования приложения следует использовать Web-серверы, имеющиеся в сети Internet. Для разработанного приложения проверяются возможности подсоединения к серверу, копирования ресурсов, соответствующих заданному числу уровней вложенности компьютер-клиент.

Источники: конспект лекций [1], [6], RFC [13, 14].

2.2. Разработка сервера прикладного протокола

2.2.1. Общие положения

Для организации серверного программного обеспечения библиотека Internet Direct предоставляет программисту компонент TIdTCPServer, включающий в себя все возможности, необходимые для построения полнофункционального многонитиевого сетевого серверного приложения.

2.2.2. Организация соединения

Библиотека Internet Direct обеспечивает возможность организации серверной службы с помощью компонента TIdTCPServer. Для организации многонитиевой сетевой службы, прослушивающей определённый порт, необходимо разместить экземпляр компонента на форме (наследнике класса TForm) или в модуле данных (наследнике класса TDataModule). Ещё один вариант – динамическое создание экземпляра класса TIdTCPServer во время исполнения (run time).

2.2.3. Задание параметров соединения

Параметры серверного соединения задаются с помощью задания свойств компонента TIdTCPServer.

Для задания параметров прослушивания сокетов используются свойства Bindings и DefaultPort:

- свойство **Bindings** задаёт список сокетов (пар IP-адрес и TCP-порт), на которых сервер будет осуществлять прослушивание
- свойство **DefaultPort** задаёт номер порта по умолчанию, на котором будет проводиться прослушивание

Для определения параметров одновременной обработки входящих соединений используются свойства MaxConnections и ListenQueue:

- свойство **MaxConnections** определяет максимальное число соединений, обслуживаемых сервером
- свойство **ListenQueue** задаёт максимальное число необработанных соединений, которые могут находиться в очереди запросов

Запуск и останов функции прослушивания осуществляется с помощью булева свойства **Active**, которое указывает на текущее состояние сервера (запущен и прослушивает соответствующий порт или остановлен).

2.2.4. Задание списка обрабатываемых команд

Сервер протокола TCP обычно строится на основе организации обработки RFC-команд, заданных в спецификации протокола в RFC. Каждой команде из списка команд прикладного протокола необходимо поставить в соответствие экземпляр объекта TIdCommandHandler, находящийся в списке CommandHandlers компонента. Для работы со списками команд используются следующие свойства:

- булево свойство **CommandHandlersEnabled** сигнализирует об использовании механизма обработчиков команд в данном экземпляре компонента TIdTCPServer
- свойство **CommandHandlers** задаёт список обработчиков команд, созданных для этого сервера. Каждый обработчик обслуживает одну RFC команду.

- свойство **ReplyUnknownCommand** задаёт RFC-ответ на приход неизвестной команды
- событие **OnBeforeCommandHandler** позволяет определить действия, которые необходимо выполнить непосредственно перед вызовом обработчика команды
- событие **OnAfterCommandHandler** позволяет определить действия, которые необходимо выполнить сразу после вызова обработчика команды
- событие **OnNoCommandHandler** определяет действия сервера в случае прихода неизвестной команды

У каждой созданной команды (объект типа TIdCommandHandler) необходимо установить группу свойств, определяющих реакцию сервера на получение этой команды:

- булево свойство **Enabled** определяет, разрешено ли использование этой команды в данный момент
- текстовое свойство **Command** определяет синтаксис команды
- символьное свойство **CmdDelimiter** задаёт код символа, отделяющего команду от параметров
- символьное свойство **ParamDelimiter** задаёт код символа, отделяющего параметры друг от друга
- булево свойство **Disconnect** определяет необходимо ли разрывать соединение после обработки данной команды
- обработчик **OnCommand** определяет действия сервера в случае прихода данной команды.

2.2.5. Обработка запросов клиента

Кроме механизмов обработки команд существуют дополнительные способы обработки запросов клиента:

- событие **OnConnected** возникает после успешного установления соединения клиентом.
- событие **OnDisconnected** возникает после разрыва соединения
- событие **OnExecute** возникает при приходе запроса от клиента в случае, когда сервер не использует механизм обработки RFC-команд (значение свойства CommandHandlersEnabled установлено в False).

2.2.6. Обмен данными с клиентским приложением

После успешного подсоединения клиентского приложения существует два способа обмена данными клиента и сервера: с использованием механизма обработчиков RFC-команд и без него.

В случае использования механизма обработчиков RFC-команд вся процедура обмена данными реализуется в обработчике события OnCommand соответствующего экземпляра TIdCommandHandler. В качестве параметра передаётся объект ASender типа TIdCommand, содержащий необходимые свойства:

- свойство **Params** используется для доступа к списку параметров обрабатываемой команды
- свойство **Thread** содержит экземпляр нити, созданной для обработки команды от данного клиента.

Для получения и посылки данных используется объект Connection: ASender.Thread.Connection. Этот объект имеет все необходимые методы реализующие функции обмена данными с клиентским приложением.

В случае отказа от использования механизма обработчиков команд, весь обмен обеспечивается обработчиком OnExecute, параметром которого является экземпляр нити Thread. В этом варианте все функции обмена осуществляются с помощью объекта Thread.Connection аналогично предыдущему случае.

2.2.7. Лабораторная работа №3. Варианты заданий

2.2.7.1 Сервер протокола POP-3

Задание: разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции сервера протокола POP-3.

Основные возможности. Приложение должно реализовывать следующие функции:

1) Хранение идентификационной и аутентификационной информации нескольких пользователей
2) Хранение почтовых папок входящих сообщений нескольких пользователей
3) Обработка подключения клиента
4) Выдача состояния ящика (количество новых писем, их суммарная длина)
5) Выдача списка всех писем сервера с длиной в байтах
6) Выдача содержимого указанного письма
7) Пометка письма для последующего удаления
8) Удаление всех помеченных писем при разрыве соединения
9) Протоколирование соединения сервера с клиентом

Поддерживаемые команды. Разработанное приложение должно реализовывать следующие команды протокола POP-3:

- USER – получение от клиента идентификационной информации пользователя
- PASS – получение от клиента пароля пользователя
- STAT – отправка клиенту состояния почтового ящика
- LIST – отправка клиенту списка сообщения почтового ящика
- RETR – отправка клиенту сообщения
- DELE – пометка сообщения на удаление
- TOP – отправка клиенту первых нескольких строк сообщения

- UIDL – выдача уникального идентификатора сообщения
- RSET – сброс всех пометок на удаление сообщений
- QUIT – удаление всех помеченных сообщений и завершение сеанса

Настройки приложения. Разработанное приложение должно обеспечивать:

- настройку номера порта сервера (по умолчанию - 110)
- создание, редактирование и удаление пользователей почтового сервера
- настройку аутентификационной информации для пользователей почтового сервера

Методика тестирования. Для тестирования приложения следует использовать приложения-клиенты, установленные в лаборатории (Mozilla Thunderbird, The Bat, MS Outlook Express).

С помощью имеющихся клиентов электронной почты осуществляется подключение к серверу с различной аутентификационной информацией. В процессе тестирования проверяются основные возможности сервера POP3.

Источники: конспект лекций [1], [6], RFC [7].

2.2.7.2 Сервер протокола SMTP

Задание: разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции сервера протокола SMTP.

Основные возможности. Приложение должно реализовывать следующие функции:

1) Поддержку хранения почтовых папок пользователей для какого-либо домена (например, test.ru)
2) Получение почты для поддерживаемого домена (test.ru)
3) Реализация функции ретрансляции почты для других доменов. Для этого с помощью DNS-запроса необходимо определить IP-адрес SMTP-сервера, отвечающего за искомый почтовый домен, (можно

использовать компонент библиотеки InternetDirect TIdDNSResolver) и переслать по этому адресу необходимые сообщения

4) Одновременная работа с несколькими клиентами
5) Протоколирование соединения сервера с клиентом

Поддерживаемые команды. Разработанное приложение должно реализовывать следующие команды протокола SMTP:

- HELO – получение информации о домене пользователя
- MAIL FROM – получение адреса отправителя письма
- RCPT TO – получение адресата письма
- DATA – получение тела письма
- QUIT – завершение сеанса связи

Настройки приложения. Разработанное приложение должно обеспечивать:

1) задание номер порта сервера (по умолчанию - 25)
2) задание имени поддерживаемого почтового домена

Методика тестирования. Для тестирования приложения следует использовать приложения-клиенты, установленные в лаборатории (Mozilla Thunderbird, The Bat, MS Outlook Express).

С помощью имеющихся клиентов электронной почты осуществляется подключение к серверу и посылка писем на поддерживаемый домен (например, test.ru) и на другие домены электронной почты. В процессе тестирования проверяются функции хранения «своей» почты и корректная отсылка почтовых сообщений другим почтовым серверам.

Источники: конспект лекций [1], [6], RFC [8, 9, 10, 11].

2.2.7.3 Сервер протокола FTP, функционирующий в активном режиме

Задание: разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции FTP-сервера.

Основные возможности. Приложение должно реализовывать следующие функции:

1) Хранение идентификационной и аутентификационной информации нескольких пользователей
2) Поддержка анонимного входа (пользователь anonymous)
3) Обработка подключения клиента
4) Выдача по запросу клиента содержимого каталога
5) Навигация по системе каталогов
6) Создание нового каталога
7) Удаление каталога
8) Посылка по запросу клиента содержимого указанного файла
9) Приём по запросу клиента содержимого указанного файла
10) Удаление указанного файла
11) Протоколирование соединения сервера с клиентом

Поддерживаемые команды. Разработанное приложение должно реализовывать следующие команды протокола FTP:

- USER – получение от клиента идентификационной информации пользователя
- PASS – получение от клиента пароля пользователя
- LIST – отправка клиенту расширенной информации о списке файлов каталога
- NLST – отправка клиенту сокращённой информации о списке файлов каталога
- CWD – смена текущего каталога сервера
- MKD – создание каталога
- RMD – удаление каталога
- DELE – удаление файла на сервере

- PORT – получение параметров сокета клиента (адреса и порта), осуществляющего приём и передачу данных
- RETR – посылка файла клиенту
- STOR – запись полученного от клиента файла
- DELE – удаление файла
- TYPE – задание режима передачи данных
- QUIT – удаление всех помеченных сообщений и завершение сеанса

Настройки приложения. Разработанное приложение должно обеспечивать:

1) настройку номеров порта сервера (по умолчанию – 21 и 20)
2) настройку корневого каталога сервера для каждого пользователя

Методика тестирования. Для тестирования приложения следует использовать стандартные FTP-клиенты, установленные в лаборатории (Mozilla Firefox, MS Explorer, Far, Total Commander).

С помощью имеющихся клиентов протокола FTP осуществляется подключение к серверу с различной аутентификационной информацией. В процессе тестирования проверяются основные возможности сервера по передаче, приёму, удалению файлов, навигации по файловой системе, функции по работе с каталогами.

Источники: конспект лекций [1], [6], RFC [12].

2.2.7.4 Сервер протокола FTP, функционирующий в пассивном режиме

Задание: разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции FTP-сервера.

Основные возможности. Приложение должно реализовывать следующие функции:

1) Хранение идентификационной и аутентификационной информации нескольких пользователей

2) Поддержка анонимного входа (пользователь anonymous)
3) Обработка подключения клиента
4) Выдача по запросу клиента содержимого каталога
5) Навигация по системе каталогов
6) Создание нового каталога
7) Удаление каталога
8) Посылка по запросу клиента содержимого указанного файла
9) Приём по запросу клиента содержимого указанного файла
10) Удаление указанного файла
11) Протоколирование соединения сервера с клиентом

Поддерживаемые команды. Разработанное приложение должно реализовывать следующие команды протокола FTP:

- USER – получение от клиента идентификационной информации пользователя
- PASS – получение от клиента пароля пользователя
- LIST – отправка клиенту расширенной информации о списке файлов каталога
- NLST – отправка клиенту сокращённой информации о списке файлов каталога
- CWD – смена текущего каталога сервера
- MKD – создание каталога
- RMD – удаление каталога
- DELE – удаление файла на сервере
- PASV – перевод сервера в пассивный режим
- RETR – посылка файла клиенту
- STOR – запись полученного от клиента файла
- DELE – удаление файла
- TYPE – задание режима передачи данных

- QUIT – удаление всех помеченных сообщений и завершение сеанса

Настройки приложения. Разработанное приложение должно обеспечивать:

1) настройку номера порта сервера (по умолчанию – 21)

2) настройку корневого каталога сервера для каждого пользователя

Методика тестирования. Для тестирования приложения следует использовать стандартные FTP-клиенты, установленные в лаборатории (Mozilla Firefox, MS Explorer, Far, Total Commander).

С помощью имеющихся клиентов протокола FTP осуществляется подключение к серверу с различной аутентификационной информацией. В процессе тестирования проверяются основные возможности сервера по передаче, приёму, удалению файлов, навигации по файловой системе, функции по работе с каталогами.

Источники: конспект лекций [1], [6], RFC [12].

2.2.7.5 Сервер протокола HTTP

Задание: разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее базовые функции сервера протокола HTTP (Web-сервера).

Основные возможности. Приложение должно реализовывать следующие функции:

1) Обработка подключения клиента
2) Разбор строки URL
3) Выдача клиенту запрошенного ресурса
4) Обеспечение параллельной загрузки клиенту страниц и медиа-элементов
5) Обеспечение параллельной работы нескольких клиентов
6) Отображение параметров, передаваемых вместе с методом POST
7) Формирование необходимых заголовков протокола HTTP

8) Протоколирование соединения клиента с сервером

Реализуемые методы протокола:

- GET – для передачи Web-страниц и медиа-элементов
- HEAD – для передачи заголовков Web-страниц и медиа-элементов
- POST – для получения от клиента параметров Web-форм.

Настройки приложения. Разрабатываемое приложение должно обеспечивать настройку корневого каталога Web-сервера.

Методика тестирования. В рабочий каталог сервера помещается содержимое какого-либо Web-сайтов сети Internet. Используемый сайт должен иметь несколько уровней вложенности и содержать медиа-элементы (например, графические изображения).

Для тестирования приложения следует использовать стандартные браузеры, имеющиеся в лаборатории (Mozilla Firefox, Internet Explorer, Opera). В процессе тестирования проверяются основные возможности сервера по передаче Web-страниц, медиа-элементов, тестируется корректность разбора строки URL, контролируются параметры, передаваемые с помощью метода POST.

Источники: конспект лекций [1], [6], RFC [13, 14].

2.3. Разработка приложения-посредника прикладного протокола

2.3.1. Общие положения

Приложения-посредники для прикладных протоколов предназначены для организации опосредованного доступа клиентских приложений к удалённым серверным службам через промежуточные приложения. Приложения-посредники применяются в случаях невозможности установить прямое соединение между клиентом и сервером и при необходимости реализовать дополнительные возможности, такие как кэширование информации, трансляция протоколов и т.п.

В случае организации сервера-посредника (прокси-сервера) с помощью библиотеки Internet Direct следует использовать пару компонентов TIdTCPClient и TIdTCPServer. Компонент TCP-сервер будет использоваться для обслуживания подключаемых клиентов, а компонент TCP-клиент для обеспечения соединения с прикладным сервером. Собственно приложение-посредник обеспечивает трансляцию команд прикладных протоколов от клиентов к серверу, и, при необходимости, преобразование транслируемых команд и данных.

2.3.2. Лабораторная работа №4. Варианты заданий

2.3.2.1 Прокси-сервер протокола HTTP

Задание: разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции прокси-сервера для протокола HTTP.

Основные возможности. Приложение должно реализовывать следующие функции:

1) Обработка подключения клиента
2) Получение запросов от клиента и перенаправление их серверу
3) Получение от сервера файлов и кэширование их
4) Передача клиенту ответа от сервера или кэшированного запроса
5) Обеспечение одновременной работы нескольких клиентов
6) Протоколирование сеанса связи клиентом с сервером

Поддерживаемые методы протокола:

- GET – для обеспечения загрузки Web-страниц и медиа-элементов
- HEAD – для передачи заголовков Web-страниц и медиа-элементов
- POST – для получения от клиента параметров Web-форм.

Настройки приложения. Разработанное приложение должно обеспечивать настройку номера TCP-порта, используемого в прокси-сервером, объёма кэша и времени хранения информации в кэше.

Методика тестирования. Для проверки работоспособности приложения используются браузеры, установленные в лаборатории (Mozilla Firefox, Internet Explorer, Opera, Netscape). В качестве параметров прокси-сервера устанавливается IP-адрес и TCP-порт, занятые разработанным приложением.

Проверяется корректность загрузки различных Web-сайтов, имеющихся в сети Internet. Особое внимание уделяется корректности параллельной загрузки медиа-элементов.

Источники: конспект лекций [1], [6], RFC [13, 14].

2.3.2.2 Прокси-сервер для протокола FTP, функционирующий в активном режиме

Задание: разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции прокси-сервера для протокола FTP. Прокси-сервер должен использовать активный режим протокола FTP.

Основные возможности. Приложение должно реализовывать следующие функции:

1) Обработка подключения клиента
2) Разбор параметра username для определения реального имени FTP-сервера
3) Переадресация всех команд клиента FTP-серверу
4) Переадресация ответов сервера клиенту
5) Трансляция канала данных, открываемого сервером, клиенту
6) Кэширование данных, присланных сервером
7) Протоколирование соединения сервера с клиентом

Поддерживаемые команды. Разработанное приложение должно обрабатывать следующие команды протокола FTP:

- USER – передача серверу идентификационной информации пользователя вместе с параметрами FTP-сервера
- PASS – передача серверу пароля пользователя
- PORT – передача на сервер параметров (адреса и порта) сокета, осуществляющего приём и передачу данных

Остальные команды протокола FTP должны транслироваться FTP-серверу.

Настройки приложения. Разработанное приложение должно обеспечивать:

1) конфигурирование номера порта, прослушиваемого прокси-сервером
2) настройку объёма кэша

Методика тестирования. Для тестирования приложения следует использовать стандартные FTP-клиенты, установленные в лаборатории (Mozilla Firefox, MS Explorer, Far, Total Commander) и имеющиеся в сети Internet FTP-серверы (ftp://ftp.funet.fi, ftp://ftp.relcom.ru и т.п.).

С помощью имеющихся клиентов протокола FTP осуществляется подключение к прокси-серверу с указанием различных реальных FTP-серверов. В процессе тестирования проверяются основные возможности прокси-сервера по трансляции команд и данных, кэшированию информации.

2.3.2.3 Прокси-сервер для протокола FTP, функционирующий в пассивном режиме

Задание: разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции прокси-сервера для прото-

кола FTP. Приложение должно использовать пассивный режим протокола FTP.

Основные возможности. Приложение должно реализовывать следующие функции:

1) Обработка подключения клиента
2) Разбор параметра username для определения реального имени FTP-сервера
3) Переадресация всех команд клиента FTP-серверу
4) Переадресация ответов сервера клиенту
5) Трансляция канала данных, открываемого клиентом, серверу
6) Кэширование данных, присланных сервером
7) Протоколирование соединения сервера с клиентом

Поддерживаемые команды. Разработанное приложение должно реализовывать следующие команды протокола FTP:

- USER – передача серверу идентификационной информации пользователя вместе с параметрами FTP-сервера
- PASS – передача серверу пароля пользователя
- PASV – переключение сервера в пассивный режим

Остальные команды протокола FTP должны транслироваться FTP-серверу.

Настройки приложения. Разработанное приложение должно обеспечивать:

1) конфигурирование номера порта, прослушиваемого прокси-сервером
2) настройку объёма кэша

Методика тестирования. Для тестирования приложения следует использовать стандартные FTP-клиенты, установленные в лаборатории

(Mozilla Firefox, MS Explorer, Far, Total Commander) и имеющиеся в сети Internet FTP-серверы (ftp://ftp.funet.fi, ftp://ftp.relcom.ru и т.п.).

С помощью имеющихся клиентов протокола FTP осуществляется подключение к прокси-серверу с указанием различных реальных FTP-серверов. В процессе тестирования проверяются основные возможности прокси-сервера по трансляции команд и данных, кэшированию информации.

Источники: конспект лекций [1], [6], RFC [12, 13, 14].

2.3.2.4 HTTP-прокси-сервер для протокола FTP

Задание: разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции HTTP-прокси-сервера для протокола FTP.

Основные возможности:

1) Подключение клиента по протоколу HTTP
2) Преобразование HTTP-запросов в FTP-команды
3) Трансляция аутентификационной информации на FTP-сервер
4) Поддержка режима анонимного пользователя
5) Протоколирование всего обмена между клиентом и сервером FTP

Поддерживаемые методы протокола HTTP и команды протокола FTP:

- GET – получение файла или каталога FTP-сервера. Преобразуется в команды RETR, LIST или NLST протокола FTP.
- PUT – запись файла. Преобразуется в команду STOR протокола FTP

Остальные команды протокола FTP реализуются по необходимости.

Методика тестирования. Для тестирования приложения следует использовать стандартные FTP-клиенты, поддерживающие механизм HTTP-прокси для протокола FTP, установленные в лаборатории (Mozilla Firefox,

MS Explorer) и имеющиеся в сети Internet FTP-серверы (ftp://ftp.funet.fi, ftp://ftp.relcom.ru и т.п.).

С помощью имеющихся клиентов протокола FTP осуществляется подключение к прокси-серверу с указанием различных реальных FTP-серверов. В процессе тестирования проверяются основные возможности прокси-сервера по трансляции команд и данных протокола FTP в HTTP и наоборот, по кэшированию информации.

Источники: конспект лекций [1], [6], RFC [12, 13, 14]

2.3.2.5 Прокси-сервер для протокола POP-3

Задание: разработать сервер-посредник для протокола POP-3, функционирующий под управлением операционных систем Windows или Linux.

Основные возможности приложения:

1) Подключение нескольких почтовых клиентов
2) Разбор аутентификационных данных и получение имени POP3-сервера
3) Трансляция команд от клиента к серверу и обратно
4) Протоколирование обмена между почтовым клиентом и сервером

Поддерживаемые команды. Разработанное приложение должно реализовывать следующие команды протокола POP-3:

- USER – получение от клиента идентификационных данных пользователя вместе с адресом реального сервера
- PASS – получение от клиента пароля пользователя
- STAT – отправка клиенту состояния почтового ящика
- LIST – отправка клиенту списка сообщения почтового ящика
- RETR – отправка клиенту текста сообщения
- DELE – пометка сообщения на удаление
- TOP – отправка клиенту первых нескольких строк сообщения
- UIDL – выдача уникального идентификатора сообщения

- RSET – сброс всех пометок на удаление сообщений
- QUIT – удаление всех помеченных сообщений и завершение сеанса

Методика тестирования. Для тестирования приложения используется стандартный клиент электронной почты, имеющийся в лаборатории (Mozilla Thunderbird, The Bat, MS Outlook Express). Параметры учетной записи настраиваются на сокет, используемый разработанным приложением. В качестве почтового сервера можно использовать лабораторные POP-3-серверы или бесплатные почтовые службы (например, www.mail.ru, www.yandex.ru и т.п.). Спроектированный прокси-сервер проверяется на выполнение набора основных функций протокола POP3.

Источники: конспект лекций [1], [6], RFC [7].

3. СИСТЕМНЫЕ ПРОГРАММНЫЕ СРЕДСТВА, ИСПОЛЬЗУЕМЫЕ ПРИ ОТЛАДКЕ СЕТЕВЫХ ПРИЛОЖЕНИЙ

Для отладки разрабатываемых сетевых приложений можно воспользоваться набором утилит операционных систем Linux [2, 3] и MS Windows [4], позволяющих проводить контроль состояния и настройку сетевых компонентов.

3.1. Утилита ifconfig

Утилита ifconfig операционной системы Linux предназначена для отображения или модификации всех параметров сетевого интерфейса.

Основные формы команды:

- ifconfig – отобразить информацию обо всех интерфейсах;
- ifconfig <interface> – отобразить информацию о конкретном интерфейсе;
- ifconfig <interface> <опции> – установить конфигурационные параметры для интерфейса.

Утилита ifconfig позволяет установить большой набор параметров сетевого интерфейса, поддерживает множество сетевых архитектур, позволяет останавливать и запускать сетевые интерфейсы. В операционной системе Linux обычно не требуется непосредственно использовать утилиту ifconfig для конфигурирования интерфейсов, в большинстве случаев для этого существуют более высокоуровневые средства: файлы сценариев, инициализационные файлы, программы-конфигураторы.

3.2. Утилита ipconfig

Аналогом утилиты ifconfig в операционной системе MS Windows XP является утилита ipconfig. В отличие от ifconfig основными функциями утилиты ipconfig являются:

- отображение текущих сетевых установок;
- отображение кэша DNS-преобразователя (DNS-resolver) и очистка его;
- операции с арендой IP-адресов, полученных через DHCP-сервер.

Установка параметров сетевых интерфейсов данной утилитой не предусмотрена.

Основные формы команды:

- ipconfig – отображение базовых сетевых параметров интерфейсов;
- ipconfig /all – отображение всех сетевых параметров интерфейсов, а также общей информации о сетевых настройках стека TCP/IP;
- ipconfig /displaydns – отображение текущего состояния кэша DNS-преобразователя;
- ipconfig /flushdns – очистка кэша DNS-преобразователя;
- ipconfig /release – освобождение полученного в аренду у DHCP-сервера IP-адреса;
- ipconfig /renew – возобновление аренды IP адреса.

3.3. Утилита arp

Утилита предназначена для взаимодействия с таблицей ARP (Address Resolution Protocol), которая хранит соответствие адресов сетевого и канального уровня. Основные формы команды:

- arp – отобразить содержимое ARP-таблицы (только для ОС Linux);
- arp –a – отобразить содержимое ARP-таблицы (для всех ОС);
- arp –s <IP> <MAC>– добавить статическую запись в ARP-таблицу;
- arp –d <IP>– удалить запись из ARP-таблицы.

3.4. Утилита netstat

Предназначена для отображения информации о статистике сетевых протоколов, текущих сетевых соединениях, состоянии сокетов, сетевых интерфейсах, таблице маршрутизации.

Основные формы команды:

- netstat – отображение информации об активных сетевых соединениях (Linux и Windows) и состоянии сокетов UNIX-домена (только в Linux);
- netstat -a – отображение информации об активных сетевых соединениях и прослушивающих сокетах;
- netstat -r – отображение таблицы маршрутизации;
- netstat -i – отображение таблицы интерфейсов (только Linux).

Дополнительно в ОС Linux утилита netstat имеет опцию “-p”, которая отображает процесс, владеющий сокетом.

3.5. Утилита route

Предназначена для отображения и модификации таблицы маршрутизации операционной системы.

Основные формы команды для ОС Linux:

- route – вывод таблицы маршрутизации;
- route add <…> – добавление маршрута;
- route del <…> – удаление маршрута;
- route flush – очистка таблицы маршрутизации.

Основные формы команды для ОС MS Windows XP:

- route print– вывод таблицы маршрутизации;
- route add <…> – добавление маршрута;
- route change <…> – изменение маршрута;
- route delete <…> – удаление маршрута;

- route –f – очистка таблицы маршрутизации;
- route –p add <…>– добавление постоянного маршрута.

3.6. Утилита ping

Утилита ping (**p**acket **in**ternet **g**roper) является одним из основных средств первичного тестирования сетевой среды и используется:

- для проверки работоспособности удалённого узла;
- для тестирования временных параметров маршрута;
- для просмотра маршрута следования пакетов.

Формат вызова: ping <опции> <адрес>.

Основные формы утилиты в операционной системе Linux:

- ping –c N <адрес>— посылка N тестовых пакетов;
- ping –S N <адрес> — установка размера посылаемых пакетов;
- ping –R <адрес> — распечатка первых 9-ти шагов маршрута.

Основные формы утилиты в операционной системе MS Windows:

- ping –n N <адрес>— посылка N тестовых пакетов;
- ping –l N <адрес> — установка размера посылаемых пакетов;
- ping –r N <адрес> — распечатка первых N шагов маршрута.

3.7. Утилита traceroute

Утилита traceroute (в MS Windows – tracert) предназначена для отображения пути следования пакета от одного узла к другому. Для построения маршрута используется цепочка тестовых UDP-дейтаграмм или ICMP-пакетов. Вариант утилиты для операционной системы MS Windows всегда использует ICMP-пакеты. В операционной системе Linux по умолчанию используются UDP-дейтаграммы, использование ICMP-пакетов включается с помощью опции –I.

Формат утилиты: traceroute <опции> <адрес>.

3.8. Программа telnet

Программа telnet (или её аналоги) может использоваться как универсальный клиент для TCP-приложений. Имитация передаваемых данных осуществляется пользователем с помощью клавиатурного ввода, ответы TCP-сервера отображаются в текстовом виде в окне программы telnet. Программа telnet входит в состав всех современных операционных систем.

4. ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА ПО ЛАБОРАТОРНЫМ РАБОТАМ

Титульный лист отчета по лабораторной работе должен быть оформлен в соответствии со стандартами СПбГПУ.

Раздел **«Техническое задание»** должен содержать подробное описание задания на разработку. Здесь следует указать все требования, предъявляемые к разрабатываемым приложениям, описать накладываемые ограничения.

В разделе **«Анализ задания и выбор способа решения»** проводится анализ поставленного технического задания, выбирается метод (способ, технология, методика) решения.

В этом же разделе описываются те технологии, протоколы, библиотеки, которые будут использоваться для решения поставленных задач.

Раздел **«Разработка сетевых приложений»** должен содержать описание архитектуры и логики взаимодействия компонентов сетевого приложения. Необходимо специфицировать разработанные прикладные протоколы, показать режимы работы всей сетевой системы. Особенности сетевого взаимодействия следует подкрепить иллюстративными материалами: структурными схемами, диаграммами.

В разделе **«Описание разработанных приложений»** следует описать архитектуру, логику функционирования и особенности реализации приложений. Здесь необходимо изобразить структуру программы, взаимосвязь ее компонентов, описать логику функционирования каждого из имеющихся режимов работы, дать краткое описание ключевых классов, методов, процедур, функций и потоков программы.

Описание проведённых процедур тестирования разработанных приложений приводится в разделе **«Тестирование приложений и анализ результатов».**

Раздел **«Выводы»** должен включать в себя ***содержательные*** выводы, сделанные именно по данной работе. Здесь следует подчеркнуть особенности используемых протоколов, режимов работы, применяемых технологий и библиотек.

В **«Приложениях»** приводятся прокомментированные исходные тексты разработанных во время лабораторных работ программ. Программы должны быть распечатаны моноширинным шрифтом (например, Courier, кегль– 9).

СПИСОК ЛИТЕРАТУРЫ

1. В.М. Ицыксон. Конспект лекций по курсу «Технологии компьютерных сетей».
2. Й. Снейдер. Эффективное программирование TCP/IP. Библиотека программиста — СПб: Питер, 2001. — 320 с: ил.
3. Э. Немет, Г. Снайдер, С. Сибасс, Т. Хейн. UNIX: Руководство системного администратора: Пер. с англ. — К.:BHV, 1996 — 832 с.
4. Титтел Э., Хадсон К., Стюарт М. TCP/IP. Сертификационный экзамен – экстерном. — СПб: Издательство «Питер», 1999. — 416 с.: ил.
5. Библиотека InternetDirect. http://www.indyproject.org/
6. Золотов С. Протоколы Internet – СПб.: BHV – Санкт-Петербург, 1998. 304 с., ил.
7. RFC 1939. Post Office Protocol - Version 3. J. Myers, M. Rose. May 1996.
8. RFC 0821. Simple Mail Transfer Protocol. J. Postel. Aug-01-1982
9. RFC 0822. Standard for the format of ARPA Internet text messages. D. Crocker. Aug-13-1982.
10. RFC 2821. Simple Mail Transfer Protocol. J. Klensin, Ed.. April 2001.
11. RFC 2822. Internet Message Format. P. Resnick, Ed.. April 2001.
12. RFC 0959. File Transfer Protocol. J. Postel, J.K. Reynolds. Oct-01-1985.
13. RFC 1945 Hypertext Transfer Protocol -- HTTP/1.0. T. Berners-Lee, R. Fielding, H. Frystyk. May 1996.
14. RFC 2616 Hypertext Transfer Protocol -- HTTP/1.1. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee. June 1999.
15. http://ru.wikipedia.org/wiki/LDAP