Курс «Информационные сети и телекоммуникации»

Лабораторная работа № 2
«Блокирующие потоковые сокеты»

Цель работы

Изучение передачи данных с применением блокирующих потоковых сокетов. Приобретение опыта реализации разделения сообщений в потоке данных при помощи двоичного протокола прикладного уровня.

Задание на лабораторную работу

Необходимо разработать систему для загрузки файлов по сети.

Пользователь программы-клиента вводит имена файлов на сервере, которые требуется загрузить, и после получения очередного из запрошенных файлов может распечатать содержимое на экране или сохранить в одноименный локальный файл. Программа-сервер ожидает подключения клиента, считывает список файлов и отправляет их содержимое клиенту (или сообщает об ошибке, например, если файла не существует).

Обмен данными требуется вести по протоколу, описанному в указаниях к выполнению лабораторной работы (там же приведены и другие полезные сведения).

1. **Реализовать программу-клиент,** которая совершает следующее:
    1.1. Запрашивает у пользователя адрес IP и порт сервера.
    1.2. Устанавливает соединение с указанным сервером.
    1.3. Циклически:
        1.3.1. Запрашивает у пользователя имя очередного файла для загрузки.
        1.3.2. При вводе строки `/quit` отправляет на сервер уведомление об отключении и завершает цикл.
        1.3.3. Отправляет на сервер пакет TLV-запрос (см. ниже) на получение файла.
        1.3.4. Принимает ответ сервера с содержимым файла или с текстом сообщения об ошибке.
        1.3.5. В случае ошибки печатает текст сообщения и продолжает цикл.
        1.3.6. При успешном приеме файла запрашивает пользователя, следует ли напечатать содержимое на экране или сохранить в локальный файл, и выполняет указанное.
    1.4. Корректно (gracefully) завершает соединение с сервером.

2. **Реализовать программу-сервер,** которая совершает следующее:
    2.1. Запрашивает у пользователя адрес IP и порт.
    2.2. Создает сокет и выполняет его привязку к заданному адресу.
    2.3. Циклически:
        2.3.1. Принимает входящее подключение на созданный сокет, печатает на экране адрес и порт подключившегося клиента.
        2.3.2. Пока клиент не отсоединился:
            2.3.2.1. Принимает очередную команду клиента.
            2.3.2.2. Если принято имя файла для загрузки и удается считать файл, отправляет клиенту пакет TLV с содержимым файла. В противном случае отправляет пакет TLV с сообщением об ошибке.
            2.3.2.3. Если принята команда «отключение», перейти к п. 2.3.3.
        2.3.3. Корректно (gracefully) завершает соединение с клиентом.
3. **Проверить корректность работы программ** с файлами размером не менее сотен килобайт (например, с изображениями).

Указания к выполнению лабораторной работы

Протокол

Следует (и это целесообразно) использовать протокол прикладного уровня, пакет которого включает последовательно:

1) длину оставшейся части пакета (беззнаковое целое, 8 байт, тип `uint64_t`);
2) тип пакета (1 байт);
3) данные (байты в количестве, заданном длиной, минус 1 байт).

Пакеты подобного формата принято называть TLV (Type, Length, Value).

Используются четыре вида пакетов:

Смысл	**Тип**	**Данные**	**Источник**
запрос файла	`R`	имя файла (строка с `'\0'`)	клиент
передача файла	`F`	содержимое файла (массив байт)	сервер
сообщение об ошибке	`E`	сообщение об ошибке (строка с `'\0'`)	сервер
отключение	`Q`	нет (длина нулевая)	клиент

Пример пакета (байты в шестнадцатеричном виде):

```
0A 00 00 00 00 00 00   52   74 65 73 74 2E 74 78 74 00
```

Пакет представляет собой запрос на файл `test.txt`. Размер строки «test.txt» — 9 байт (8 символов и завершающий `'\0'`), еще один байт занимает тип пакета (52 — код симовла `R`), итого 10 байт (в шестнадцатеричном виде — A).

План решения

Целесообразно разбить задачу на более простые:

1. Реализовать функции приема и отправки блока данных заданного размера:

```
bool receiveSome(SOCKET from, char* data, size_t size);
bool sendSome(SOCKET to, const char* data, size_t size);
```

2. Реализовать отправку пакета TLV как отправку всего содержимого при помощи функции `sendSome()`, написанной в п. 1.
3. Реализовать прием пакет TLV в два этапа при помощи функции `receiveSome()`:
   a) прием типа и размера данных (их длины известны);
   b) прием данных, размер которых получен в п. 3a).

Схема работы функции `receiveSome()` может быть приблизительно такой:

1. Принято `0` байт..
2. Пока не принято `size` байт:
   2.1. Осталось принять (`size - `*`принято`*) байт.
   2.2. Принять очередную порцию данных в область `data`, начиная с первого еще не принятого байта. Возможно принять столько байт, сколько осталось. Пусть удалось принять очередные `received` байт.
   2.3. Стало принято на `received` байт больше, чем ранее.

В пункте 2.2 возможна ошибка функции `recv()`, в этом случае можно вернуть **`false`**.

В помощь при разработке

- Предоставляются эталонные реализации, которые корректно работают друг с другом и должны — с написанными решениями. Программа-клиент сохраняет загруженные файлы всегда в свой рабочий каталог.
- Функция для считывания содержимого файла в `std::vector<`**`char`**`>`. Записать в поток (`std::cout` или `std::ofstream`) данные можно методом `write()`.

Контрольные вопросы

1. В чем отличия работы дейтаграммных и потоковых сокетов?
2. В каких случаях следует использовать потоковые сокеты и почему?
3. Почему невозможна broadcast рассылка при помощи потоковых сокетов?
4. Как обеспечить разделение сообщений, передаваемых через потоковый сокет?
5. Опишите схему работы клиента, использующего потоковые сокеты.
6. Опишите схему работы (синхронного) сервера, использующего потоковые сокеты.