Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Сети и телекоммуникации

Отчет по лабораторной работе
"Сетевые технологии"

Работу выполнил:
Миносян Э.К.
Группа: 43501/1
Преподаватель:
Алексюк А.О.

Санкт-Петербург
2018

1 Система терминального доступа

1.1 Цель работы

Разработать приложение «Система терминального доступа». Задание: разработать клиент-серверную систему терминального доступа, позволяющую клиентам подсоединиться к серверу и выполнить элеметарные 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 клиентов через механизм нитей
4. Проведение аутентификации клиента на основе полученных имени пользователься 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 - Привелигированная 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) серыерц
4. Получение ответа от сервера
5. Разрыв соединения
6. Обработка ситуации отключения клиента сервером или другим клиентом

2 Протокол команд

2.1 Форматы команд

Для получения или изменения информации на сервере клиент посылает серверу текстовые команды. Набор команд для TCP клиента должен удовлетворять следующим требованиям:

- Все команды пишутся в нижнем регистре;
- Соответствие шаблону сообщения.

Протокол TCP имеет следующий шаблон сообщения:
<команда> <атрибут> <атрибут>
В начале сообщения, всегда присутствует требуемая для выполнения команда, далее в зависимости от нее могут идти атрибуты, которые отделены друг от друга пробелом. В случае, если пересылаемое клиентом сообщение не соотвествует шаблону, сервер сообщает о неверно введенной команде.

- Команды доступные серверу:

  При подключении на сервер, клиенту необходимо пройти аутентификацию

  <l> - вывод подключенных пользователей

  <q> - отключение сервера

- Команды доступные клиенту:

  При подключении на сервер, клиенту необходимо пройти аутентификацию

  <login "login" "password"> - команда для аутентификации существующих пользователей

  <addusr "login" "password"> - команда для добавления и последующей аутентификации нового пользователя

  После аутентификации клиенту доступны следующие команды

  <ls> - вывод содержимого в текущей директории

  <cd> - перемещение в заданную директорию

  <logout> - смена пользователя

  Если пользователь обладает правами суперпользователя, ему доступные следуюшие команды

  <who> - вывод списка пользователей, их статус(онлайн или не онлайн), и текущая директория

  <kill "login"> - команда позволяющая отключить соответствуюего клиента

Обмен сообщений происходит по следующему алгоритму:
Со стороны клиента:

1. Прием сообщения от сервера
2. Отправка сообщения серверу

Со стороны серера:

1. Отправка сообщения клинету
2. Прием сообщения от клиента
3. Обработка команды
4. Отправка ответа клиенту

2.2 Прототипы функций, реализованных на сервере:

1. Функции сервера

   int MessageReceive(SOCKET socket, char *buf, string &line); - функция получения сообщения сервером

   int MessageSend(SOCKET socket, string buffer); - функция отправки сообщения сервером

   DWORD WINAPI ProcessClient(void* socket); - функция обработки команд отосланных клиентами

   DWORD WINAPI ProcessServer(); - функция обработки сообщений отосланных сервером

   DWORD WINAPI ConnectionsAccept(void *listenSocket); - функция обработки подключения по сокетам

   void WSAStart(); - функция инициализации сокетов

   void closeSocket(int ind); - функция закрытия сокетов

2. Функции терминала

   int UserFileRewrite(); - функция перезаписи файла listusers.txt

   int getIndexUser(string login); - функция позволяющая выдавать индексы пользователям

   int UserFileRead(); - функция считывания файла listusers.txt

   int addusrImpl(string login, string password); - функция обработки команды addusr

   int loginImpl(string login, string password); - функция обработки команды login

   int killImpl(string login); - функция обработки команды kill

   string getServerPath();- функция получения пути до сервера

   string cdImpl(SOCKET sock, string dir, string path); - функция обработки команды cd

   vector<string> lsImpl(string folder); - функция обработки команды ls

   bool DirCompare(string i, string j);- функция сравнения каталогов(необходима для реазизации функции обработки команды ls)

2.3 Формат хранимых файлов на сервере

Серверная программа TCP позволяет сохранять, изменять и загружать информацию о пользователях. На сервере обязательно присутствует файл listusers.txt, в котором хранятся существующие пользователи в следующем виде:
"login"||"password"||"Права пользователя(sudo - если суперпользователь)"||"Начальный каталог"

Исходный код можно посмотреть на https://github.com/emil151997/NetworksLab2018

3 Выводы

В данной лабораторной работе были реализованы клиент-серверная программа терминального доступа с разработкой собственного протокола на основе TCP.Протокол был реализован на языке C++ для операционных систем Windows. На примере данной разработки были изучены основные приемы использования протокола транспортного уровня TCP — транспортного механизма, предоставляющего поток данных с предварительной установкой соединения. Его преимуществом является достоверность получаемых данных за счет осуществления повторного запроса данных в случае их потери, устранения дублирования при получении копий одного пакета.

Однако TCP может не подойти в некоторых ситуациях обмена по сети вследствие медленной (по сравнению с UDP) работы. Например, передавая по сети данные, требующие быстрого отклика в реальном времени, необходимо соблюдать жесткие временные рамки, с которыми протокол TCP может не справиться.