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:
Шайхенуров Р.Р.
Группа: 43501/1
Преподаватель:
Алексюк А.О.

Санкт-Петербург
2018

1 Сетевой форум

1.1 Цель работы

Разработать приложение «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

1.2 Методы обмена сообщениями

- void sendErrorCode(int newsockfd, int type); - отправка кода ошибки
- void sendSuccessCode(int newsockfd); - отправка "успешного"кода
- void writing(int newsockfd, char *buffer); - чтение сообщения от клиента
- void reading(int newsockfd, char *buffer); - отправка сообщения клиенту

1.3 Формат команды

Так как протокол выбран синхронный, то реализация команды представляет из себя текстовое сообщение, в котором содержится цифра для взаимодействия с сервером.

[INT]

Обмен сообщений происходит по следующему алгоритму:
Со стороны клиента:

1. Прием инструкции от сервера со STATUS-кодом
2. Отправка сообщения

Со стороны серера:

1. Отправка STATUS-кода клиенту
2. Отправка сообщения
3. Прием сообщения
4. Выполнения команды
5. Отправка ответа клиенту

1.4 Прототипы функций, реализованных на сервере:

1. void actionLogin(char *buffer, int newsockfd);
2. void actionMenu(char *buffer, int newsockfd);
3. char *actionThemes(char *buffer, int newsockfd);
4. void actionMessages(char *theme, char *buffer, int newsockfd);
5. void actionNewMessage(char *buffer, int newsockfd);
6. void checkOnline(char *buffer, int newsockfd);
7. void putNewMsg(char *newMessage);

1.5 Описание функций:

1. Авторизация клиента на форуме
2. Вывод клиенту список меню и ожидание действия
3. Вывод клиенту списка тем форума, ожидание выбора
4. Прием сообщения в ветку форума, вывод всех сообщений ветки
5. Вывод списка последних сообщений из всех тем форума
6. Вывод списка пользователей, находящихся в данный момент онлайн
7. Вставка сообщения в список последних сообщений форума

1.6 Описание

Для запуска севера используется строка вида:

[server.exe]

где:
server.exe - скомпилированый исполняемый файл сервера
Для запуска клиента используется строка вида:

[client.exe addres port]

где:
client.exe - скомпилированый исполняемый файл клиента
addres - адрес на котором запущен TCP-сервер(localhost)
port - порт который будет прослушивать сервер(8080)
При первом запуске потребуется ввести имя пользователя и пароль в виде username_passwd, после прохождения проверки клиент будет подключен к серверу. Максимальное количество выводимых последних сообщений:

```
int countOfNewMsg = 3;
```

Максимальная длина сообщения:

```
\#define MAX_LENGTH 256
```

Используемые библиотеки сервера:

```
\#include <stdio.h>
\#include <pthread.h>
\#include <unistd.h>
\#include <cygwin/socket.h>
\#include <sys/socket.h>
\#include <strings.h>
\#include <stdlib.h>
\#include <cygwin/in.h>
\#include <memory.h>
\#include <stdbool.h>
```

Тип сокета:

```
int sockfd;
sockfd = socket(AF_INET, SOCK_STREAM, 0);
```

Сервер реализован с помощью enum'а, то есть имеет константные состояния для каждого клиента. Это возможно, так как используется механизм нитей, поэтому у каждого клиента своя область видимости и свое состояние сервера.

1.7 Функции, для механизма нитей

1. void *connection_f(); - нить, подключающая клиента к серверу, выделяя ему

   ```
   socketId
   ```

2. void *connect_handler(void *args); - нить, выдающая собственную область видимости состояний сервера клиенту

Исходный код можно посмотреть на https://github.com/Pox3R/NetworksLab2018

2 Прикладной протокол HTTP

2.1 Цель работы

Разработать «Прикладной протокол HTTP». Задание: предоставить серверную реализацию протокола HTTP обрабатывающего следующие методы HTTP:

1. GET
2. POST
3. HEAD

Клиентским приложением является браузер или программа Postman

2.2 Методы обмена сообщениями

Для обмена сообщениями используется браузер или программа Postman, с их помощью можно формировать запросы нужных видов и отправлять серверу

2.3 Формат команды

Так как протокол выбран синхронный, то реализация команды представляет из себя текстовое сообщение, в котором содержится цифра для взаимодействия с сервером.

Обмен данными происходит по следующему алгоритму:
Со стороны клиента: Формируется HTTP запрос и отправляется на сервер Со стороны серера: Сервер принимает запрос, после чего обрабатывает полученное сообщение, через s.getInputStream(), после чего формирует **response header** и **response body** и отправляет через s.getOutputStream()

2.4 Прототипы функций, реализованных на сервере:

1. public static void main(String args[]) - main-функция, в которой происходит прослушка порта и создание потока, при подключении клиента
2. public HttpServer(Socket s) - конструктор класса, запускающий поток для конкретного сокета
3. public void run() - функция, необходимая для реализации интерфейса Thread, которую будет выполнять поток
4. protected String getData(String header) - получение данных, отправленных на сервер для их разбора - обрабатывает GET запрос
5. public String parseURL(String path, String URI) - парсинг URL, необходимый для извлечения пути к файлу, к которому обратился клиент
6. protected String extract(String str, String start, String end) - функция, необходимая для извлечения URL
7. public String getPOST(String URI) - обработка POST-запроса
8. public String getHEAD(String URI) - обработка HEAD запроса

2.5 Описание

Для запуска севера используется строка вида:

[server.jar]

Или запустить можно в среде разработки Intellij IDEA.
Для запуска клиента - GET запрос:

1. Открываем браузер
2. в адресную строку вводим http://localhost:80/*имя файла, который вы хотите получить*, например, http://localhost:80/logo.jpg

Для POST и HEAD запросов:

1. Открываем Postman
2. Вводим адрес, указанный выше, с файлом(Для HEAD запроса) и с Query-параметрами для POST-запроса
3. Нажимаем кнопку SEND
4. В нижней части окна получаем ответ сервера
5. В POST запросе получаем список Query-параметров и Header запроса
6. В HEAD запросе получаем Header файла, который указали

Сервер работает асинхронно, то есть выдает то, что хочет клиент и завершает поток. Поэтому при каждом новом подключении клиента происходит пересоздание сокета, это позволяет достичь того, что поключиться к серверу смогут множество клиентов. При вводе имени файла, которого нет на сервере, сервер выдаст ошибку 404, означающую "Not Found при попытке получить HEAD файла, не существующего на сервере, приходит HEADER с нулевыми значениями. При отправке POST-запроса, файлы не учитываются, считываются лишь его Query-параметры, которые отправляет сервер в Response Body

Используемые библиотеки сервера:

```
import java.io.*;
import java.net.*;
import java.text.DateFormat;
import java.util.Date;
import java.util.TimeZone;
```

3 Выводы

При реализации TCP сервера возникли проблемы с обработкой файлов и строк, так как не использовались стандартные библиотеки. Написано в ручную множество функций для строк(Например, удаление из середины строки), используя стандартные CHAR'овские функции. В итоге получили Клиент-Серверное синхронное приложение: Сетевой форум, обрабатывающий по своему протоколу сообщения клиента. С помощью STATUS-кодов можно определить, что именно не так сделал клиент и легко исправить ввод. Использовалась модель конечного автомата, которая не очень хорошо подожша бы для графического приложения, но для данного случая подходит отлично. В будущем можно будет пересмотреть логику сервера, чтобы он был асинхронным, позволяющим перейти в любое меню по командам.

При реализации HTTP протокола возникли следующие сложности: обработка принятого сообщения. В итоге была написана функция, которая обрезает Request, в связи с нужным ответом клиенту. Использовались Java-socket'ы и интерфейс Thread, которые немного упростили задачу в конечном итоге. Так же сервером формируется Response в связи со стандартом RFC так, чтобы это поддерживалось браузером.