TCP-чат 1.0

Документация

Галимуллин Амир

Github: https://github.com/fdhjdzcb/TCP_Chat

ОГЛАВЛЕНИЕ

КРАТКОЕ ОПИСАНИЕ

Для сборки проекта необходимо наличие библиотек Google Test (GTest) и Google Logging (glog).

Проект содержит четыре исполняемых файла.

Исполняемый файл Server использует код серверной части приложения и использует файл code/server.cpp.

Исполняемый файл ServerTests содержит юнит-тесты для серверной части приложения и использует файл test/server_test.cpp.

Исполняемый файл Client использует код клиентской части приложения и использует файл code/client.cpp.

Исполняемый файл ClientTests содержит юнит-тесты для клиентской части приложения и использует файл test/client_test.cpp.

СЕРВЕР

Описание

Серверная часть приложения представляет собой простой TCP сервер, который прослушивает определенный порт и поддерживает соединения с клиентами в разных потоках.

Глобальные переменные

pthread_t thread_id

Переменная, которая используется для хранения идентификатора потока. Переменная создается перед запуском нового потока и используется для управления этим потоком.

std::unordered_map<int, std::string> Connections

Хэш-таблица, которая используется для хранения информации о клиентах, подключенных к серверу. Ключом является целочисленный идентификатор сокета, а значением - имя клиента, связанное с этим сокетом.

Функции

void configLOGS();

Функция настраивает логирование с помощью библиотеки glog:

- Создает директорию "logs" и устанавливает ее в качестве директории для хранения лог-файлов.
- Отключает вывод логов в стандартный поток вывода с помощью FLAGS_logtostderr = false
- Устанавливает минимальный уровень логирования для вывода в файл с помощью FLAGS_stderrthreshold = 3
- Устанавливает интервал времени для автоматической записи буферизованных лог-сообщений в файл с помощью FLAGS_logbufsecs = 1.
- Устанавливает путь и имя файла для вывода логов в файл "logs/server_log_file".

void configPort(sockaddr_in &addr);

Функция настраивает сокет для прослушивания входящих соединений на определенном IP-адресе и порте. Использует структуру sockaddr_in, которая содержит информацию об IP-адресе и порте, заполняет поля структуры необходимыми значениями.

int createListenSocket(sockaddr_in &addr);

- Создает новый сокет для прослушивания входящих соединений, связывает его с заданным IP-адресом и портом
- Устанавливает опцию SO_REUSEADDR для сокета
- Вызывает функции bind() и listen() для связывания и прослушивания сокета

void addName(std::string &msg, int socketID);

Добавляет имя клиента из unordered_map Connections к передаваемому сообщению msg и сохраняет результат в msg.

std::string receiveMsgFromClient(int socketID);

- Получает размер сообщения, отправленного клиентом, через вызов функции recv() и проверяет на ошибки.
- Создает буфер для сообщения, получает само сообщение через recv(), добавляет завершающий нулевой символ и возвращает сообщение в виде строки.
- Если произошла ошибка при получении сообщения или размер сообщения не положительный, функция выбрасывает исключение для отсоединения клиента.

void sendMsg(int socketID, std::string &msg);

Отправляет сообщение msg, представленное в виде строки, через сокет с идентификатором socketID. Сначала она определяет размер сообщения, затем отправляет его размер и само сообщение через вызов функции send().

void sendMsgToClients(std::string &msg);

Отправляет сообщение msg всем подключенным клиентам, используя функцию sendMsg().

void addClientToList(std::string &msg, const auto &name);

Добавляет имя клиента к сообщению msg в формате "Сообщение, имя_клиента,".

std::string createWelcomeMsg();

Создает приветственное сообщение для клиента, которое будет отправлено при подключении. Если в словаре Connections нет подключенных клиентов, сообщение будет содержать информацию о том, что в чате никого нет. Если же клиенты уже подключены, сообщение будет содержать список их имен.

void welcome(int socketID);

Отправляет приветственное сообщение новому клиенту, используя функцию sendMsg.

void deleteClient(int socketID);

Закрывает сокет с идентификатором socketID и удаляет информацию о клиенте из словаря Connections.

void receiveNameFromClient(int socketID);

- Получает имя клиента через вызов функции receiveMsgFromClient
- Отправляет приветственное сообщение клиенту
- Сохраняет имя клиента в unordered_map Connections.

void *clientHandler(void *arg);

Является функцией-обработчиком для каждого подключенного клиента. получает идентификатор сокета socketID через аргумент arg

- Отправляет запрос на ввод имени клиента, используя функцию sendMsg
- Ожидает получения имени клиента через вызов функции receiveNameFromClient
- Затем бесконечном цикле
- Получает сообщение от клиента через вызов функции receiveMsgFromClient
- Добавляет имя клиента к сообщению через вызов функции addName
- Отправляет сообщение всем подключенным клиентам через вызов функции sendMsgToClients
- Если происходит ошибка при получении сообщения от клиента, функция выбрасывает исключение, удаляет клиента из Connections и закрывает сокет.

КЛИЕНТ

Описание

Клиентская часть приложения представляет собой программу, которая устанавливает соединение с сервером и позволяет отправлять и получать сообщения от сервера. После установки соединения, клиент запускает поток для приема сообщений от сервера, который позволяет клиенту получать сообщения от сервера в режиме реального времени.

```
amir@DESKTOP-TEO6PPM:/mnt/z/cprogs$ sh claunch.sh
Connected!
Введите ваше имя:
Эдвард Нортон
Привет! Сейчас в чате никого нет!
Эдвард Нортон: А где все?(((
Тайлер Дерден вошел в чат!
Тайлер Дерден: Я здесь.
Эдвард Нортон: О, привет, а ты кто?
Тайлер Дерден: А ты не знаешь?
Эдвард Нортон: Нет, не знаю. Почему меня все принимают за тебя?
Тайлер Дерден: Я это ты.
Эдвард Нортон: Чтоооооооо. Ты меня пугаешь, пока.
Чак Паланик вошел в чат!
Чак Паланик: Привет.
Тайлер Дерден: Пока
Эдвард Нортон: Пока
Чак Паланик: (((((((((
```

Рисунок 1 - Интерфейс клиента

```
amir@DESKTOP-TEO6PPM:/mnt/z/cprogs$ sh slaunch.sh
ListenSocketID: 6
Connection socket: 7
Connection socket: 8
Client from socket 8 disconnected
Client from socket 7 disconnected
```

Рисунок 2 - Интерфейс сервера

Функции

void configLOGS();

Функция настраивает логирование с помощью библиотеки glog:

void configPort(sockaddr_in &addr);

Функция настраивает сокет для прослушивания входящих соединений на определенном IP-адресе и порте. Использует структуру sockaddr_in, которая содержит информацию об IP-адресе и порте, заполняет поля структуры необходимыми значениями.

void* receiveMsgFromServer(void *arg);

Функция, которая запускается в отдельном потоке и слушает входящие сообщения от сервера. Аргумент arg является указателем на целочисленный идентификатор сокета, который используется для получения сообщений от сервера.

В бесконечном цикле

- ожидает получения сообщения от сервера
- принимает размер сообщения, который передается в виде целого числа
- выделяет память для сообщения и принимает сообщение
- выводит полученное сообщение в консоль и освобождает выделенную для него память

int createSocket();

Создает сокет с проверкой успешности создания.

int connectToServerBySocket(int socketID, sockaddr_in &addr);

Подключается к серверу по номеру сокета с проверкой успешности подключения.

int connectToServer(sockaddr_in &addr);

Инициирует создание сокета и подключение к серверу.

void sendMsg(int socketID, std::string &msg);

Отправляет сообщение серверу через указанный сокет. Аргумент socketID является идентификатором сокета, через который нужно отправить сообщение, а аргумент msg – это строка с текстом сообщения.

Сначала вычисляется размер сообщения, затем размер отправляется через сокет, после чего отправляется само сообщение.

std::string getMsg();

Используется для получения сообщения от пользователя через стандартный ввод (консоль). Функция ожидает ввод пользователем непустой строки и сохраняет ее в переменную msg.