НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Мегафакультет: Компьютерных технологий и управления

Направление: 09.03.04 «Программная инженерия»

Лабораторная работа №3

По дисциплине: «Системное программное обеспечение»

Вариант 1

Выполнил:

студент группы P33112

Адамов Степан Сергеевич

Преподаватель: Кореньков Ю.Д

Санкт-Петербург 2021

Цели

изучение способов взаимодействия между сетевыми службами низкого уровня в асинхронном режиме

Задачи

Разработать клиент-серверное приложение. Для организации взаимодействия по сети, поддержки множества соединений использовать программные интерфейсы (API) операционной системы.

Сервер и клиент взаимодействуют по протоколу, реализованному на базе сокетов (использовать TCP, если в варианте задания не указано обратное). Сервер должен поддерживать условно неограниченное количество клиентов. На всех этапах взаимодействия клиента и сервера должна быть предусмотрена обработка данных, независимо от их размера

Описание работы.

Программа может выполняться в двух режимах: сервер или клиент. Режим определяется аргументом командной строки. В режиме сервера линейно отображается журнал всех входящих и исходящих сообщений, завершение программы-сервера выполняется по нажатию ключевой клавиши (например, Q).

При запуске в режиме клиента программе в качестве аргументов командной строки также передается имя пользователя и адрес сервера.Необходимо предусмотреть возможность отправки «приватного» сообщения, которое увидит только адресат, которому оно предназначено. При подключении отображать последние 20 сообщений,предусмотреть возможность просмотра историисообщений(не сохраняя её при этом на стороне клиентского приложения).

Новые сообщения выводятся в конце списка с автопрокруткой по мере появления новых сообщений. Клавишами «стрелка вверх/вниз», PageUp/PageDown выполняется прокрутка списка сообщений, отменяющая автопрокрутку при появлении новых сообщений. Клавишей End автопрокрутка возобновляется. Собственное сообщение вводится в отдельнойстроке в нижней части окна,не блокируя историю сообщений

Аспекты реализации

Сервер

При запуска запускаются 2 потока :

1. `init_connection`
    a. `выполняет подключение новых пользователей`
    b. `запускает новый поток connection_receive_text, который принимает сообщения от пользователей`
    c. `уведомляет всех пользователей о новом подключении`
2. `queue_routine – рассылает всем пользователям приятые сообщения`

Клиент

При запуска запускаются 3 потока :

1. display_output_routine –выводит данные на экран
2. sender_routine – отвечает за ввод сообщения и отправку их на сервер
3. receiver_routine – принимает сообщения с сервера

Для гарантии, что работа будет завершена корректным образом с помощью “/q” на клиенте или “q” на сервере и не будет прекращена с помощью «CTRL+C” используется следующий код:

```
struct sigaction sigIntHandler;
sigIntHandler.sa_handler = my_handler;
sigemptyset(&sigIntHandler.sa_mask);
sigIntHandler.sa_flags = 0;
sigaction(SIGINT, &sigIntHandler, NULL);
```

При использовании многопоточного программирования используются condition variables. Так, к примеру, у клиента обновление экрана выполняется только после сигнала из принимающего потока.

Параметры запуска:
s port – сервер
c ClientID localhost(127.0.0.1) port – клиент

отправка приватного сообщения: /private receiver ms

Вывод:

В ходе лабораторной работы было разработано клиент серверное приложение «Чат», для реализации многопоточности в котором были использованы мьютексы и condition variables