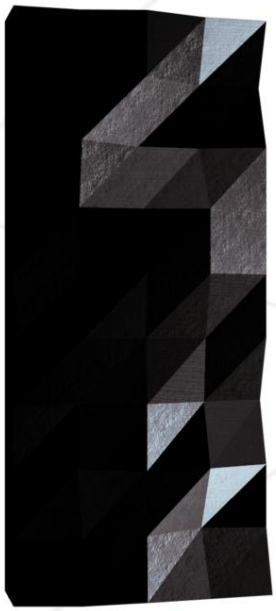

# ft_irc

## Интернет-релейный чат

Описание:

Этот проект посвящен созданию собственного IRC-сервера.

Вы будете использовать настоящий IRC-клиент для подключения к вашему серверу и его тестирования.

Интернет управляется протоколами твердых стандартов, которые позволяют подключенным компьютерам взаимодействовать друг с другом.

Всегда полезно знать.

Версия: 6

# Содержание

# Глава I

## Введение

Internet Relay Chat или IRC — это текстовый протокол связи в Интернете.
Он предлагает обмен сообщениями в режиме реального времени, который может быть как публичным, так и частным. Пользователи могут обмениваться прямыми сообщениями и присоединяться к групповым каналам.

Клиенты IRC подключаются к серверам IRC, чтобы присоединиться к каналам. IRC-сервера подключены вместе, чтобы сформировать сеть.

# Глава II

## Основные правила

- Ваша программа не должна давать сбой ни при каких обстоятельствах (даже когда памяти) и не должен неожиданно завершать работу.
  Если это произойдет, ваш проект будет считаться нефункциональным, а ваша оценка будет равна 0.

- Вы должны сдать Makefile, который скомпилирует ваши исходные файлы. Это не должно пересвязать.

- Ваш Makefile должен как минимум содержать правила:
  $(NAME), all, clean, fclean и re.

- Скомпилируйте свой код с помощью C++ и флагов -Wall -Wextra -Werror

- Ваш код должен соответствовать стандарту C++ 98. Затем он все равно должен скомпилироваться если вы добавите флаг -std=c++98.

- Старайтесь всегда разрабатывать с использованием как можно большего числа возможностей C++ (например, выбирайте <cstring> вместо <string.h>). Вам разрешено использовать функции C, но всегда предпочитайте их версии C++, если это возможно.

- Запрещены любые внешние библиотеки и библиотеки Boost.

# Глава III

## Обязательная часть

| | |
|---|---|
| Название программы | ircserv |
| Сдать файлы | Makefile, *.{h, hpp}, *.cpp, *.tpp, *.ipp, необязательный файл конфигурации NAME, all, clean, fclean, re port: Пароль порта |
| Makefile | прослушивания: Пароль соединения Все в C++ 98. socket, |
| Аргументы | setsockopt, getsockname, getprotobyname, gethostbyname, getaddrinfo, freeaddrinfo, bind, connect, listen, accept, htons, |
| Внешние функции. | htonl, ntohs, ntohl, inet_addr, inet_ntoa, send, recv, signal, lseek, fstat, fcntl, poll (или аналогичный ) н/д IRC-сервер на C++ 98 |
| Либфт авторизован | |
| Описание | |

Вам нужно разработать IRC-сервер на C++ 98.

Вы не должны развивать клиента.
Вы не должны обрабатывать связь между серверами.

Ваш исполняемый файл будет запущен следующим образом:

./ircserv <порт> <пароль>

• порт: номер порта, на котором ваш IRC-сервер будет прослушивать входящие IRC-соединения.

• пароль: пароль подключения. Он понадобится любому IRC-клиенту, который попытается подключиться к вашему серверу.

Даже если в теме и шкале оценки упоминается poll(), вы можете использовать любой эквивалент, такой как select(), kqueue() или epoll().

# III.1 Требования

• Сервер должен быть способен обслуживать несколько клиентов одновременно и никогда не вешать.

• Разветвление не допускается. Все операции ввода-вывода должны быть неблокирующими.

• Для обработки всех этих операций (чтение, запись, а также прослушивание и т. д.) может использоваться только 1 poll() (или аналогичный).

Поскольку вы должны использовать неблокирующие файловые дескрипторы, можно использовать функции чтения/получения или записи/отправки без опроса() (или эквивалента), и ваш сервер не будет блокироваться.
Но это будет потреблять больше системных ресурсов.
Таким образом, если вы попытаетесь прочитать/получить или записать/отправить любой файловый дескриптор без использования poll() (или его эквивалента), ваша оценка будет равна 0.

• Существует несколько клиентов IRC. Вы должны выбрать один из них в качестве эталона. Твой эталонный клиент будет использоваться в процессе оценки.

• Ваш эталонный клиент должен иметь возможность подключаться к вашему серверу, не встречая любая ошибка.

• Связь между клиентом и сервером должна осуществляться через TCP/IP (v4 или v6).

• Использование эталонного клиента на вашем сервере должно быть таким же, как и на любом официальном IRC-сервере. Однако вам нужно реализовать только следующие функции:

  - Вы должны иметь возможность пройти аутентификацию, установить псевдоним, имя пользователя, присоединиться к каналу, отправлять и получать личные сообщения с помощью вашего эталонного клиента.
  - Все сообщения, отправленные одним клиентом в канал, должны каждый второй клиент, присоединившийся к каналу.
  - У вас должны быть операторы и обычные пользователи.
  - Затем необходимо реализовать команды, относящиеся к операторам.

• Конечно, вы должны писать чистый код.

## III.2 Только для MacOS

Поскольку MacOS не реализует write() так же, как другие ОС Unix, вам разрешено использовать fcntl().

Вы должны использовать файловые дескрипторы в неблокирующем режиме, чтобы получить поведение аналогично другим ОС Unix.

Однако вам разрешено использовать fcntl() только следующим образом: fcntl(fd, F_SETFL, O_NONBLOCK);

Любой другой флаг запрещен.

## III.3 Пример испытания

Проверьте абсолютно все возможные ошибки и проблемы (получение неполных данных, низкая пропускная способность и т. д.).

Чтобы ваш сервер корректно обрабатывал все, что вы ему отправляете, можно выполнить следующий простой тест с использованием nc:

```
\$> nc 127.0.0.1 6667
com^Dman^Dd \$>
```

Используйте ctrl+D, чтобы отправить команду в несколько частей: «com», затем «man», затем «d\n».

Чтобы обработать команду, вы должны сначала агрегировать полученные пакеты в для того, чтобы восстановить его.

# Глава IV

## Бонусная часть

Вот дополнительные функции, которые вы можете добавить к своему IRC-серверу, чтобы он выглядел еще более похожим на настоящий IRC-сервер:

- Управление передачей файлов.
- Бот.

Бонусная часть будет оцениваться только в том случае, если обязательная часть будет ИДЕАЛЬНОЙ. Идеальный означает, что обязательная часть была полностью выполнена и работает без сбоев. Если вы не выполнили ВСЕ обязательные требования, ваша бонусная часть вообще не будет оцениваться.

# Глава V

## Представление и экспертная оценка

Сдайте задание в своем репозитории Git, как обычно. Во время защиты будет оцениваться только работа внутри вашего репозитория. Не стесняйтесь перепроверять имена ваших файлов, чтобы убедиться, что они верны.

Вам рекомендуется создавать тестовые программы для вашего проекта, даже если они не будут отправлены и не будут оцениваться. Эти тесты могут быть особенно полезны для проверки вашего сервера во время защиты, а также сервера вашего коллеги, если вам однажды придется оценить другой ft_irc. Действительно, вы можете использовать любые тесты, которые вам нужны в процессе оценки.

Ваш эталонный клиент будет использоваться в процессе оценки.