Федеральное государственное автономное образовательное учреждение высшего профессионального образования Национальный исследовательский университет «Высшая школа экономики»

Московский институт электроники и математики Факультет прикладной математики и кибернетики

Кафедра «Компьютерная безопасность»

ОТЧЕТ

по дисциплине «Экзамен Современные технологии программирования и обработки информации»

Реализация асинхронного сервера с использование протокола FTP и библиотеки libevent

Выполнил: студент группы
СКБ-171 Зайцева А.А

МОСКВА 2021

Задание

Общее задание: Реализовать асинхронный параллельный сервер, реализующий выбранный задачей протокол. Порт приёма входящих соединений и число потоков задавать параметрами программы. Осуществлять штатный выход по сигналам прерывания/завершения процесса.

Задание для протокола FTP: Реализовать FTP-сервер. Достаточно реализовать только анонимный доступ (без аутентификации), пассивный режим (PASV, соединение для передачи данных устанавливает клиент), только команды, необходимые для перемещения по дереву каталогов, получению списка файлов и скачке файлов в поточном двоичном режиме. Тестировать корректность с браузерами, ещё поддерживающими FTP (не Chrome) или специализированными FTP-клиентами, например, WinSCP. По логам работы с клиентами можно выяснить минимальное необходимое количество команд для корректной работы.

Код реализации можно посмотреть здесь: https://github.com/Nasty09/ISAP

Использованные функции библиотеки libevent

Папка event2 с заголовочными файлами для libevent:

- bufferevent.h:
  - *bufferevent_trigger* – запускает обратные вызовы данных bufferevent

```
void bufferevent_trigger(struct bufferevent *bufev, short iotype,
    int options);
```

  - *bufferevent_socket_new* – создает новый сокет bufferevent поверх существующего сокета

```
struct bufferevent *bufferevent_socket_new(struct event_base *base,
                                           evutil_socket_t fd, int options);
```

  - *bufferevent_socket_connect* – запускает попытку connect() с помощью bufferevent на основе сокетов

```
int bufferevent_socket_connect(struct bufferevent *,
                               const struct sockaddr *, int);
```

  - *bufferevent_set_timeouts* – устанавливает таймер на чтение или запись для bufferevent. Если чтение/запись отключены, или если операция чтения/записи bufferevent была приостановлена из-за отсутствия данных для записи, недостаточной пропускной способности и т.д., timeout не активен. timeout становится активным только тогда, когда мы действительно хотим читать или писать.

```
int bufferevent_set_timeouts(struct bufferevent *bufev,
const struct timeval *timeout_read, const struct timeval *timeout_write)
```

  - *bufferevent_write* – записывает данные в буфер bufferevent

```
int bufferevent_write(struct bufferevent *bufev,
    const void *data, size_t size);
```

  - *bufferevent_read* – читает данные из буфера bufferevent

```
size_t bufferevent_read(struct bufferevent *bufev,
                        void *data, size_t size);
```

  - *bufferevent_setcb* – изменяет обратные вызовы для события bufferevent

```
void bufferevent_setcb(struct bufferevent *bufev,
    bufferevent_data_cb readcb, bufferevent_data_cb writecb,
    bufferevent_event_cb eventcb, void *cbarg);
```

  - *bufferevent_enable* – включает buffevent

```
int bufferevent_enable(struct bufferevent *bufev, short event);
```

  - *bufferevent_free* – освобождает память, связанную со структурой bufferevent

```
void bufferevent_free(struct bufferevent *bufev);
```

- util.h:
  - *evutil_inet_pton* – преобразует сетевой адрес IPv4 или IPv6 в его стандартной текстовой форме представления в его числовую двоичную форму
    ```
    int evutil_inet_pton(int af, const char *src, void *dst);
    ```
- event.h:
  - *event_base_dispatch* – создает цикл диспетчеризации событий. Он будет запускать базу событий до тех пор, пока не кончатся ожидающие или активные события, или пока не будут вызваны event_base_loopbreak() или event_base_loopexit()
    ```
    int event_base_dispatch(struct event_base *);
    ```
  - *event_base_free* – освобождает всю память, связанную с базой событий event_base, и освобождает base
    ```
    void event_base_free(struct event_base *);
    ```
  - *event_config_new* – создает объект конфигурации события. С помощью этой конфигурации можно менять поведение базы событий (event_base)
    ```
    struct event_config *event_config_new(void);
    ```
  - *event_config_set_flag* – устанавливает один или несколько флагов для настройки того, какие части возможной event_base будут инициализированы и как они будут работать
    ```
    int event_config_set_flag(struct event_config *cfg, int flag);
    ```
  - *event_base_new_with_config* – инициализирует новую базу событий с учетом указанной конфигурации
    ```
    struct event_base *event_base_new_with_config(const struct event_config *);
    ```
  - *event_config_free* – освобождает всю память, связанную с объектом конфигурации событий
    ```
    void event_config_free(struct event_config *cfg);
    ```
  - *event_new* – выделяет и назначает новую структуру событий, готовую к добавлению. Функция возвращает новое событие, которое можно использовать в будущих вызовах event_add() и event_del(). Аргументы fd и events определяют, какие условия вызовут событие; аргументы callback и callback_arg сообщают Libevent, что делать, когда событие становится активным
    ```
    struct event *event_new(struct event_base *, evutil_socket_t,
                            short, event_callback_fn, void *);
    ```
  - *event_add* – добавляет событие в набор ожидающих событий
    ```
    int event_add(struct event *ev, const struct timeval *timeout);
    ```

- listener.h:
  - *evconnlistener_new_bind* – выделяет новый объект evconnlistener для прослушивания входящих TCP-соединений по заданному адресу

```
struct evconnlistener *evconnlistener_new_bind(struct event_base *base,
    evconnlistener_cb cb, void *ptr, unsigned flags, int backlog,
    const struct sockaddr *sa, int socklen);
```

  - *evconnlistener_free* – отключает и освобождает evconnlistener

```
void evconnlistener_free(struct evconnlistener *lev);
```

Протокол FTP

Реализация

Список литературы

1. Библиотека libevent - http://www.wangafu.net/~nickm/libevent-book/
2. Описание протокола FTP - https://tools.ietf.org/html/rfc959
3.