Отчёт по курсовой работе

Горин Дмитрий Игоревич

28 декабря 2020 г.

Введение

Задание на курсовую работу: Реализовать клиента SMTP-сервера с использованием системного вызова select и использующего один рабочий поток с журналированием в отдельном процессе. Данный отчет содержит информацию о протоколе SMTP, а так же информацию о реализации и разработке курсового проекта.

Глава 1

Аналитический раздел

1.1 Протокол SMTP

SMTP (Simple Mail Transfer Protocol) — это широко используемый сетевой протокол, предназначенный для передачи электронной почты в сетях TCP/IP. SMTP впервые был описан в RFC 821 (1982 год); последнее обновление в RFC 5321 (2008) включает масштабируемое расширение — ESMTP (Extended SMTP). В настоящее время под «протоколом SMTP» как правило подразумевают и его расширения. Протокол SMTP предназначен для передачи исходящей почты с использованием порта TCP 25.

1.1.1 SMTP-команды

Ниже представлен неплоный список SMTP-команд:

- HELO - Открывает SMTP-сессию
- EHLO - Открывает ESMTP-сессию
- MAIL - Определяет отправителя сообщения
- RCPT - Определяет получателей сообщения
- DATA - Определяет начало сообщения
- RSET - Сброс SMTP-соединения
- VRFY - Проверяет имя пользователя системы
- QUIT - Остановить сеанс SMTP

1.1.2 SMTP-сессия

SMTP-сессия состоит из команд, посылаемых SMTP-клиентом, и соответствующих ответов SMTP-сервера. Когда сессия открыта, сервер и клиент обмениваются её параметрами. Сессия может включать ноль и более SMTP-операций (транзакций).

Сессия начинается с команды HELO или EHLO, и заканчивается командой QUIT. Сбросить состояние сессии можно командой RSET.

Отправка сообщения состоит из трёх последовательностей команда/ответ:

1. MAIL FROM — устанавливает обратный адрес
2. RCPT TO — устанавливает получателя данного сообщения. Эта команда может быть дана несколько раз, по одной на каждого получателя.
3. DATA — для отправки текста сообщения.

1.2 Плюсы и минусы использования однопоточной схемы обработки подключений с использованием системного вызова select

Плюсы:

- Упрощение логики программы
- select принимает аргументы одной длинны, что упрощает работу с памятью

Минусы:

- select может обработать до 1024 (FD_SETSIZE) подключений максимум
- Невозможна одновременная (параллельная) обработка нескольких подключений, что увеличит время обработки каждого подключения

Глава 2

Конструкторский раздел

2.1 Конечный автомат состояний сервера

На Рис. 2.1 представлен конечный автомат клиентской части курсовой работы.

2.2 Описание основных структур данных

- String – Структура строки текста
- SMTPMessage – SMTP-сообщение. Содержит весь текст сообщения, а так же спарсенные данные для команд MAIL FROM и RCPT TO
- SMTPMessageQueue – Очередь SMTP-сообщений
- SMTPConnection – SMTP-соединение. Содержит сокет, буферы чтения и записи и очередь сообщений
- SMTPConnectionList – Список SMTP-подключений

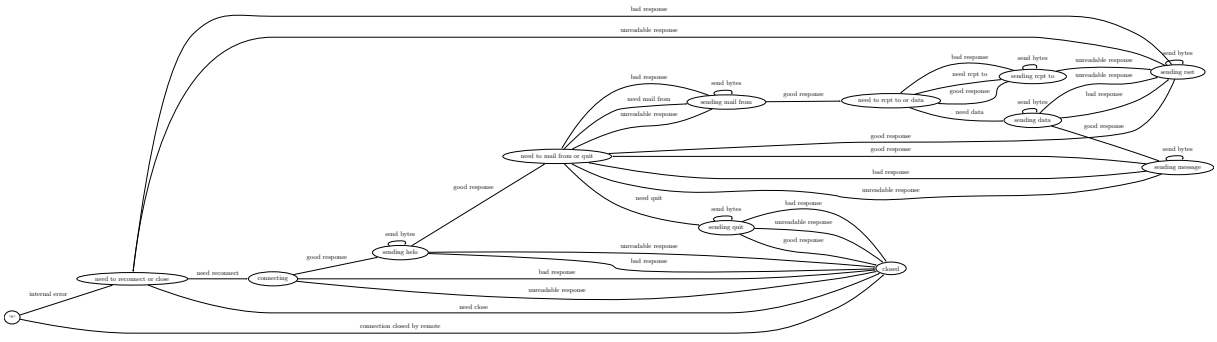

Рис. 2.1 Состояния клинетской части

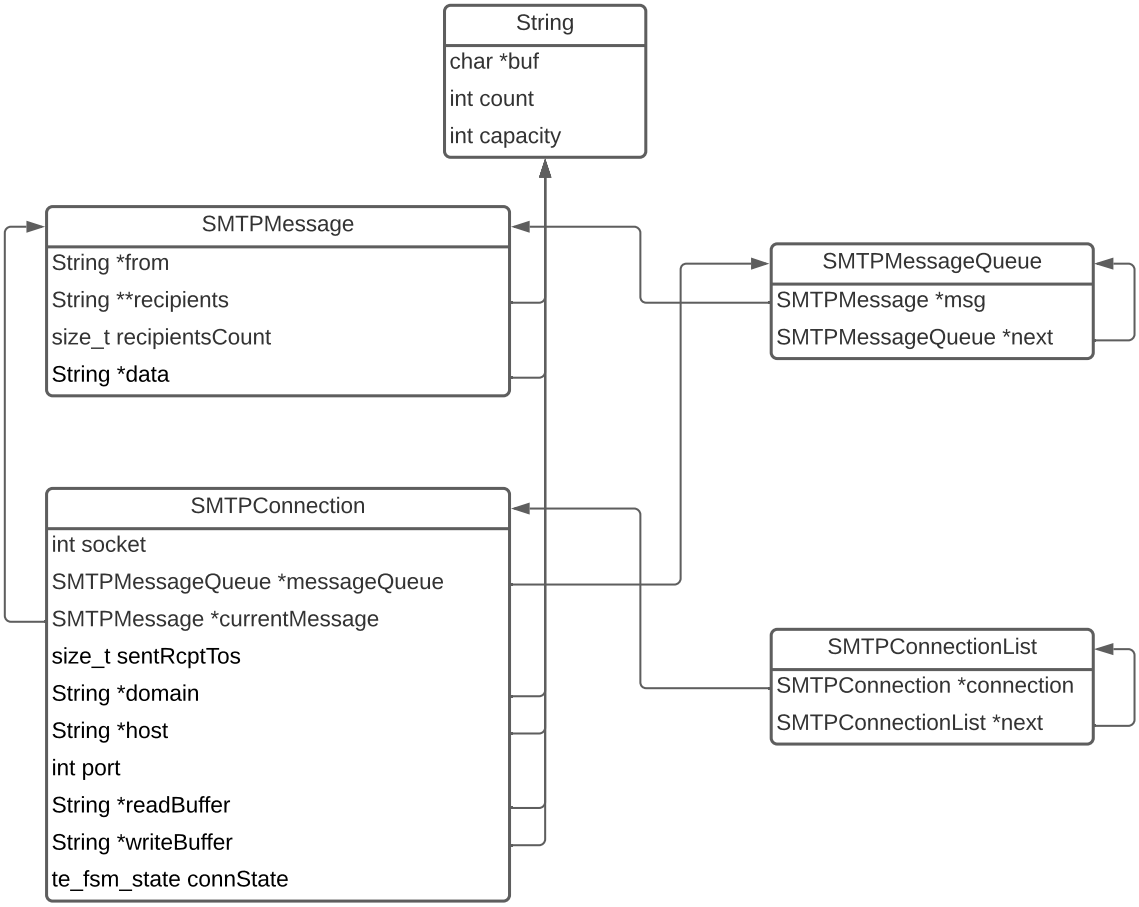

Рис. 2.2 ER-диаграмма

2.3 Связь логгера и основной программы

Программа-клиент отправляет сообщения программе-логгеру через технологию IPC message queue (sys/msg.h)

Глава 3

Технологический раздел

3.1 Доступ к данным на диске

Сервер записывает все письма в директорию Client/mails в виде текстовых файлов. Для корректной работы команд MAIL FROM и RCPT TO сервер записывает нужные данные в заголовки X-KIMI-From и X-KIMI-To соответственно.

3.2 Сборка программы

Сборка программы описана в файле Makefile системы сборки make.

3.3 Основные функции программы

Весь это раздел сгеренерировал doxygen из части комментированных исходников программы. В файле конфигурации doxyggen.cfg был отключён параметр HAVE_DOT, поскольку для рисования графов вызовов используется cflow.

3.4 Файл client/client.c

Функции клиента

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
```

#include <unistd.h>
#include <signal.h>
#include <sys/socket.h>
#include "client.h"
#include "../logger/logger.h"
#include "../errors/client_errors.h"
#include "../bytes/bytes.h"
#include "../smtp/smtp_connection_list.h"
#include "../autogen/fsm-fsm.h"

Функции

- ssize_t readFromFd (SMTPConnection *connection)
- ssize_t sendThroughSocket (SMTPConnection *connection, int flags)
- int parseResponseCode (const String *responseString)
- int clientMain (int needLoopback)

3.4.1 Подробное описание

Функции клиента

3.4.2 Функции

3.4.2.1 clientMain()

int clientMain (
int needLoopback)

Основная функция клинета

Аргументы

needLoopback	Нужена ли отправка письма обратно на сторону сервера

Возвращает

Код ошибки (0 – успешное завершение)

3.4.2.2 parseResponseCode()

int parseResponseCode (
const String * responseString)

Парсинг кода возврата от SMPT-сервера

Аргументы

responseString	- Строка, прочитанная из подключения (До CRLF)

Возвращает

Код возврата (0 если код возврата нечитаемый)

3.4.2.3 readFromFd()

ssize_t readFromFd (
SMTPConnection * connection)

Чтение из сокета

Аргументы

connection	SMTP-соединение

Возвращает

Длинна прочитанного сообщения

3.4.2.4 sendThroughSocket()

ssize_t sendThroughSocket (
SMTPConnection * connection,
int flags)

Отправка данных через сокет

Аргументы

connection	SMTP-подключение
flags	Флаги send

Возвращает

Количество отправленых байт

3.5 Файл logger/logger.c

Функции логгера

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <unistd.h>
#include <errno.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include "logger.h"
#include "../errors/client_errors.h"
```

Функции

- int logConnectingTo (const String *domain, const String *host)
- int logError (const char *file, const char *func, int line)
- int logCantRmFile (const char *filepath)
- int logNoConnectionForFdFound (int fd)
- int logResponseForFdAndDomain (int fd, const String *domain, const String *response, const char *command, LogMessageType messageType)
- int logGoodResponse (int fd, const String *domain, const String *response, const char *command)
- int logBadResponse (int fd, const String *domain, const String *response, const char *command)
- int logUnreadableResponse (int fd, const String *domain, const String *response, const char *command)
- int logClosedByRemote (int fd, const String *domain)
- int logInternalError (int fd, const String *domain)
- int logInvalidTransition (int fd, const String *domain)
- int logDecidedTo (int fd, const String *domain, const char *command)
- int logChangeState (int fd, const String *domain, int oldState, const char *old↩ StateName, int newState, const char *newStateName)
- int logMessage (const char *message, LogMessageType messageType)
- pid_t loggerMain (void)

3.5.1 Подробное описание

Функции логгера

3.5.2 Функции

3.5.2.1 loggerMain()

```
pid_t loggerMain (
            void  )
```

Основная функция логгера (с созданием процесса)

Возвращает

Pid процесса-логгера

3.5.2.2 logMessage()

```
int logMessage (
            const char * message,
            LogMessageType messageType )
```

Логгирование

Аргументы

message	Сообщение
messageType	Тип сообщения (LogMessageType)

Возвращает

Код ошибки (0 – успешно)

3.6 Файл smtp/smtp_connection.c

SMTP-подключение

```
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <resolv.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <errno.h>
#include "../errors/client_errors.h"
#include "../bytes/bytes.h"
#include "../logger/logger.h"
#include "smtp_command.h"
#include "smtp_message.h"
#include "smtp_connection.h"
```

Функции

- String * getIpByHost (const String *host, int *port, int needConnect)
- SMTPConnection * smtpConnectionInitEmpty (const String *domain, int need↩ Connect)
- int smtpConnectionReconnect (SMTPConnection *self, int needClose)
- int smtpConnectionIsNeedToWrite (const SMTPConnection *self)
- int smtpConnectionIsHaveMoreMessages (const SMTPConnection *self)
- int smtpConnectionPushMessage (SMTPConnection *self, SMTPMessage *message)
- int smtpConnectionSetCurrentMessage (SMTPConnection *self)
- int smtpConnectionClearCurrentMessage (SMTPConnection *self)
- String * smtpConnectionGetLatestMessageFromReadBuf (SMTPConnection *self, int *exception)
- void smtpConnectionDeinit (SMTPConnection **self, int needClose)

3.6.1 Подробное описание

SMTP-подключение

3.6.2 Функции

3.6.2.1 getIpByHost()

```
String* getIpByHost (
            const String * host,
            int * port,
            int needConnect )
```

Получение IP-адреса почтового сервера

Аргументы

host	Хост
port	Порт
needConnect	Нужно ли делать DNS-запрос (если нет, то вернется 127.0.0.1)

Возвращает

IP-адрес почтового сервера. и установка порта

3.6.2.2 smtpConnectionClearCurrentMessage()

```
int smtpConnectionClearCurrentMessage (
        SMTPConnection * self )
```

Очистка текущего SMTP-сообщения

Аргументы

self	SMTP-подключение

Возвращает

Код ошибки (0 – успех)

3.6.2.3 smtpConnectionDeinit()

```
void smtpConnectionDeinit (
        SMTPConnection ** self,
        int needClose )
```

Деструктор SMTP-подключения

Аргументы

self	SMTP-подключение
needClose	Нужно ли закрывать сокет

3.6.2.4 smtpConnectionGetLatestMessageFromReadBuf()

```
String* smtpConnectionGetLatestMessageFromReadBuf (
        SMTPConnection * self,
        int * exception )
```

Получение самого старого сообщения из полученных через SMTP-подключение (через readBuffer)

Аргументы

self	SMTP-подключение
exception	Код ошибки (0 – успех)

Возвращает

Строка самого старого сообщения

3.6.2.5 smtpConnectionInitEmpty()

```
SMTPConnection* smtpConnectionInitEmpty (
        const String * domain,
        int needConnect )
```

Создание структуры SMTP-подключения

Аргументы

domain	Домен, к которому нужно подключиться
needConnect	Нужно ли подключаться через connect()

Возвращает

SMTP-подключение

3.6.2.6 smtpConnectionIsHaveMoreMessages()

```
int smtpConnectionIsHaveMoreMessages (
        const SMTPConnection * self )
```

Предикат наличия сообщений для отправки в SMTP-подключение

Аргументы

self	SMTP-подключение

Возвращает

Есть или нет письма в очереди на отправку

3.6.2.7 smtpConnectionIsNeedToWrite()

```
int smtpConnectionIsNeedToWrite (
        const SMTPConnection * self )
```

Предикат необходимости записи в сокет

Аргументы

self	SMTP-подключение

Возвращает

Нужно или нет писать в сокет

3.6.2.8 smtpConnectionPushMessage()

```
int smtpConnectionPushMessage (
        SMTPConnection * self,
        SMTPMessage * message )
```

Добавление письма в очередь на отправку

Аргументы

self	SMTP-подключение
message	SMTP-сообщение

Возвращает

Код ошибки (0 – успех)

3.6.2.9 smtpConnectionReconnect()

```
int smtpConnectionReconnect (
        SMTPConnection * self,
        int needClose )
```

Переподключение сокета в SMTP-соединении

Аргументы

self	SMTP-соединение
needClose	Нужно ли закрывать текущее подключение на сокете

Возвращает

Дескриптор сокета

3.6.2.10 smtpConnectionSetCurrentMessage()

```
int smtpConnectionSetCurrentMessage (
        SMTPConnection * self )
```

Установка нового письма на отправку из очереди

Аргументы

self	SMTP-подкоючение

Возвращает

Код ошибки (0 – успех)

3.7 Файл smtp/smtp_connection_list.c

Список SMTP-подключений

```
#include <stdlib.h>
#include <errno.h>
#include "../errors/client_errors.h"
#include "../logger/logger.h"
#include "../bytes/bytes.h"
#include "smtp_connection.h"
#include "smtp_connection_list.h"
```

Функции

- SMTPConnectionList * smtpConnectionListInitEmptyNode ()
- SMTPConnection * smtpConnectionListGetConnectionWithSocket (SMTPConnectionList *head, int socket)
- SMTPConnection * smtpConnectionListGetConnectionWithDomain (SMTPConnectionList *head, const String *domain)
- SMTPConnectionList * smtpConnectionListAddMessage (SMTPConnectionList *head, const SMTPMessage *message, int ignoreKimiMimi)
- SMTPConnectionList * smtpConnectionListAddConnectionToList (SMTPConnectionList *head, SMTPConnection *conn)
- SMTPConnectionList * smtpConnectionListRemoveAndDeinitConnectionWithSocket (SMTPConnectionList *head, int socket, int needClose)
- void smtpConnectionListDeinitList (SMTPConnectionList *head, int needClose)

3.7.1 Подробное описание

Список SMTP-подключений

3.7.2 Функции

3.7.2.1 smtpConnectionListAddConnectionToList()

```
SMTPConnectionList* smtpConnectionListAddConnectionToList (
        SMTPConnectionList * head,
        SMTPConnection * conn )
```

Добавление подключения в список

Аргументы

head	Голова списка
conn	SMTP-подключение

Возвращает

Новая голова списка

3.7.2.2 smtpConnectionListAddMessage()

```
SMTPConnectionList* smtpConnectionListAddMessage (
        SMTPConnectionList * head,
        const SMTPMessage * message,
        int ignoreKimiMimi )
```

Добавление сообщения в подключения (и создание нового подключения при необходимости)

Аргументы

head	Голова списка
message	Сообщение
ignoreKimiMimi	Игнорировать или нет локальный домен

Возвращает

Новая голова списка

3.7.2.3 smtpConnectionListDeinitList()

```
void smtpConnectionListDeinitList (
        SMTPConnectionList * head,
        int needClose )
```

Деструктор списка

Аргументы

head	Голова списка
needClose	Нужно ли закрывать сокетные подключения

3.7.2.4 smtpConnectionListGetConnectionWithDomain()

SMTPConnection* smtpConnectionListGetConnectionWithDomain (
SMTPConnectionList * head,
const String * domain)

Получение подключения с доменом

Аргументы

head	Голова списка
domain	Домен

Возвращает

SMTP-подключение

3.7.2.5 smtpConnectionListGetConnectionWithSocket()

SMTPConnection* smtpConnectionListGetConnectionWithSocket (
SMTPConnectionList * head,
int socket)

Получение подключения с сокетом

Аргументы

head	Голова списка
socket	Дескриптор сокета

Возвращает

SMTP-подключение

3.7.2.6 smtpConnectionListInitEmptyNode()

SMTPConnectionList* smtpConnectionListInitEmptyNode ( )

Создание ноды списка

Возвращает

Новая нода списка

3.7.2.7 smtpConnectionListRemoveAndDeinitConnectionWithSocket()

SMTPConnectionList* smtpConnectionListRemoveAndDeinitConnectionWithSocket (
SMTPConnectionList * head,
int socket,
int needClose)

Удаление подключения из списка и его деинициализация

Аргументы

head	Голова списка
socket	Дескриптор сокета для поиска подключения на удаление
needClose	Нужно ли закрывать сокетное подключение

Возвращает

Новая голова списка

3.8 Файл smtp/smtp_message.c

SMTP-сообщение

```
#include <stdlib.h>
#include <dirent.h>
#include <string.h>
#include <errno.h>
#include "../bytes/bytes.h"
#include "../errors/client_errors.h"
#include "../logger/logger.h"
```

#include "smtp_message.h"

Функции

- SMTPMessage * smtpMessageInit ()
- SMTPMessage * smtpMessageInitCopy (const SMTPMessage *copy)
- SMTPMessage * smtpMessageInitFromFile (const char *filename)
- SMTPMessage ** smtpMessageInitFromDir (const char *dirname, int *messages↩ Number)
- int smtpMessageAddRecipient (SMTPMessage *self, String *recipient)
- String * smtpMessageGetAnyHeader (const char *headerName, const String *headerData)
- String * getDomainFromEmailAddress (const String *emailAddress)
- String ** smtpMessageGetRecipientsDomainsDistinct (const SMTPMessage *self, size_t *domainsNum)
- String * smtpMessageGetFromHeader (const SMTPMessage *self)
- void smtpMessageDeinit (SMTPMessage **self)

3.8.1 Подробное описание

SMTP-сообщение

3.8.2 Функции

3.8.2.1 getDomainFromEmailAddress()

String* getDomainFromEmailAddress (
const String * emailAddress)

Получение домена из почтового адреса

Аргументы

emailAddress	Почтовый адрес

Возвращает

Домен

3.8.2.2 smtpMessageAddRecipient()

```
int smtpMessageAddRecipient (
        SMTPMessage * self,
        String * recipient )
```

Добавление получателя в SMTP-сообщение

Аргументы

self	SMTP-сообщение
recipient	Новый получатель

Возвращает

Новое количество получателей

3.8.2.3 smtpMessageDeinit()

```
void smtpMessageDeinit (
        SMTPMessage ** self )
```

Деструктор SMTP-сообщения

Аргументы

self	SMTP-сообщение

3.8.2.4 smtpMessageGetFromHeader()

```
String* smtpMessageGetFromHeader (
        const SMTPMessage * self )
```

Получение SMTP-хэдерв FROM

Аргументы

self	SMTP-сообщение

Возвращает

Хэдер

3.8.2.5 smtpMessageGetRecipientsDomainsDistinct()

String** smtpMessageGetRecipientsDomainsDistinct (
const SMTPMessage * self,
size_t * domainsNum)

Получение всех уникальных доменов получателей

Аргументы

self	SMTP-сообщение
domainsNum	Полученное количество доменов

Возвращает

Массив доменов

3.8.2.6 smtpMessageInit()

SMTPMessage* smtpMessageInit ( )

Создание пустого SMTP-сообщения

Возвращает

SMTP-сообщение

3.8.2.7 smtpMessageInitCopy()

SMTPMessage* smtpMessageInitCopy (
const SMTPMessage * copy)

Создание копии SMTP-сообщение

Аргументы

copy	SMTP-сообщение для копирования

Возвращает

Копия сообщения

3.8.2.8 smtpMessageInitFromDir()

```
SMTPMessage** smtpMessageInitFromDir (
        const char * dirname,
        int * messagesNumber )
```

Создание SMTP-сообщений из директории

Аргументы

dirname	Имя дериктории
messagesNumber	Количество созданных сообщений

Возвращает

Массив сообщений

3.8.2.9 smtpMessageInitFromFile()

```
SMTPMessage* smtpMessageInitFromFile (
        const char * filename )
```

Создание SMTP-сообщения из файла

Аргументы

filename	Имя файла

Возвращает

SMTP-сообщение

3.9 Файл smtp/smtp_message_queue.c

Очередь SMTP-сообщений

```
#include <stdlib.h>
#include <errno.h>
#include "../bytes/bytes.h"
#include "../bytes/string.h"
#include "../logger/logger.h"
#include "../errors/client_errors.h"
#include "smtp_message_queue.h"
```

Функции

- SMTPMessageQueue * smtpMessageQueueInit (const SMTPMessage *message)
- SMTPMessageQueue * smtpMessageQueuePush (SMTPMessageQueue *head, const SMTPMessage *message)
- SMTPMessageQueue * smtpMessageQueuePop (SMTPMessageQueue *head, SMTPMessage **message)
- size_t smtpMessageQueueCount (SMTPMessageQueue *head)
- void smtpMessageQueueDeinitNode (SMTPMessageQueue *node)
- void smtpMessageQueueDeinitQueue (SMTPMessageQueue *head)

3.9.1 Подробное описание

Очередь SMTP-сообщений

3.9.2 Функции

3.9.2.1 smtpMessageQueueCount()

```
size_t smtpMessageQueueCount (
            SMTPMessageQueue * head )
```

Длинна очереди

Аргументы

head	Голова очереди

Возвращает

Длинна очереди

3.9.2.2 smtpMessageQueueDeinitNode()

void smtpMessageQueueDeinitNode (
SMTPMessageQueue * node)

Деструктор ноды очереди

Аргументы

node	Нода очереди

3.9.2.3 smtpMessageQueueDeinitQueue()

void smtpMessageQueueDeinitQueue (
SMTPMessageQueue * head)

Деструктор Очереди

Аргументы

head	Голова очереди

3.9.2.4 smtpMessageQueueInit()

SMTPMessageQueue* smtpMessageQueueInit (
const SMTPMessage * message)

Создание ноды очереди сообщений

Аргументы

message	SMTP-сообщение

Возвращает

Нода очереди сообщений

3.9.2.5 smtpMessageQueuePop()

```
SMTPMessageQueue* smtpMessageQueuePop (
        SMTPMessageQueue * head,
        SMTPMessage ** message )
```

Получение сообщения из очереди

Аргументы

head	Голова очереди
message	Полученное сообщение

Возвращает

Новая голова очереди

3.9.2.6 smtpMessageQueuePush()

```
SMTPMessageQueue* smtpMessageQueuePush (
        SMTPMessageQueue * head,
        const SMTPMessage * message )
```

Добавление сообщения в очередь

Аргументы

head	Голова очереди
message	SMTP-сообщение

Возвращает

Новая голова очереди

3.10 Графы вызова функций

На рис. 3.1 показаны основные функции.

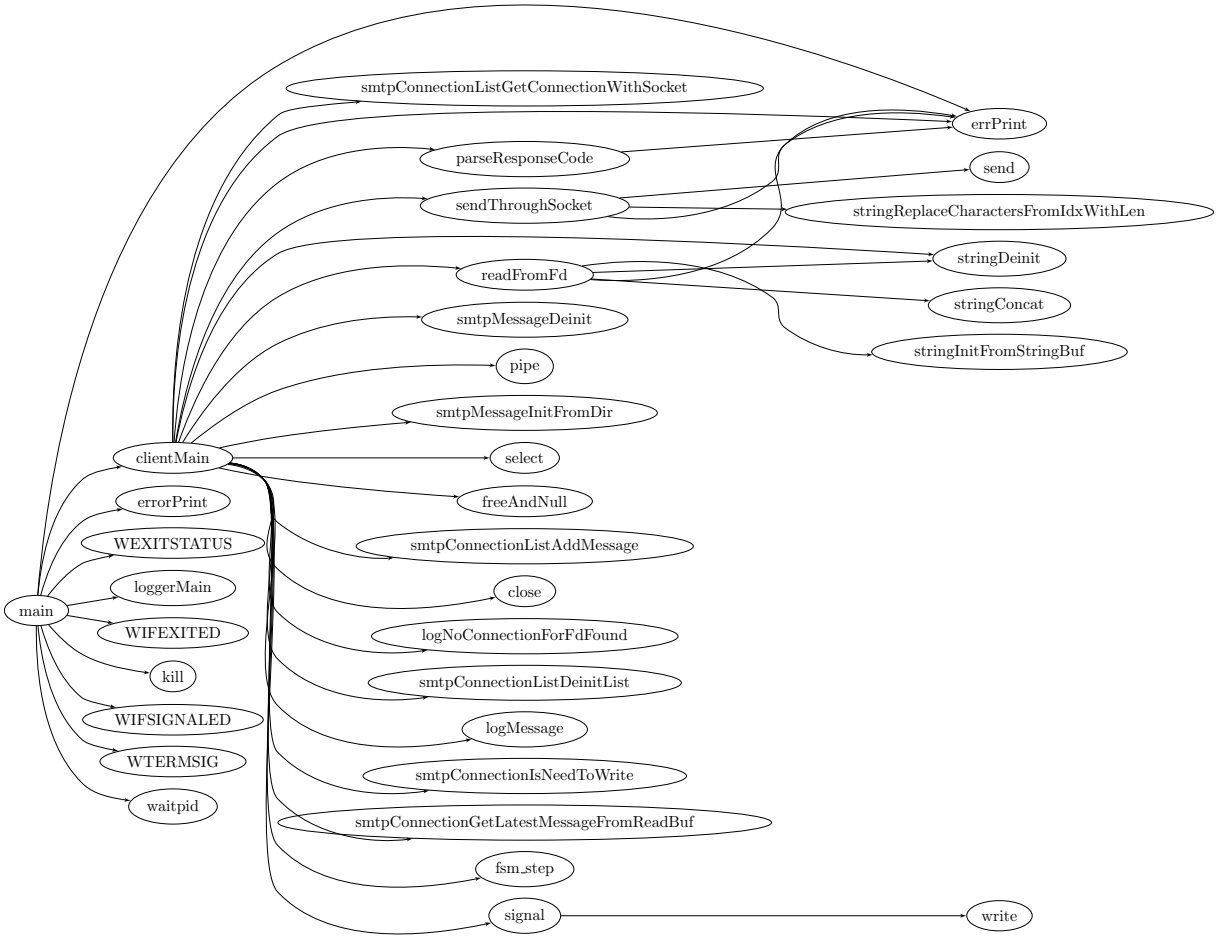

Рис. 3.1 Граф вызовов, основные функции клиента

Графы созданы с помощью cflow, cflow2dot, dot.

3.11 Модульные тесты

Для модульного тестирования используется библиотека CUnit. Всего написано 35 тестов с 205 ASSERT-вызовами.

3.12 Проверка утечек памяти с помощью valgrind

```
==33081== HEAP SUMMARY:
==33081== in use at exit: 962,565 bytes in 14,919 blocks
==33081== total heap usage: 18,623 allocs, 3,704 frees, 1,026,349 bytes allocated
==33081==
==33081== Searching for pointers to 14,919 not-freed blocks
==33081== Checked 14,649,888 bytes
==33081==
==33081== LEAK SUMMARY:
==33081== definitely lost: 0 bytes in 0 blocks
==33081== indirectly lost: 0 bytes in 0 blocks
==33081== possibly lost: 0 bytes in 0 blocks
==33081== still reachable: 944,640 bytes in 14,760 blocks
==33081== suppressed: 17,925 bytes in 159 blocks
==33081== Rerun with —leak-check=full to see details of leaked memory
==33081==
==33081== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 4 from 4)
```

3.13 Генерация документации и отчета

Отчет и документация генерируются автоматически через Makefile. Для этого в Makefile добавлена цель doxygen и report (report также генерирует doxygen).

Выводы

В ходе выполнения работы был реализован клиент серверной части SMTP-сервера. Был изучен протокол SMTP и были получены навыки в написании сетевых приложений на языке C с использованием мультиплексирования.