Отчет курсовой работе
по курсу "Протоколы вычислительных сетей"
по теме "Проектирование, реализация и тестирование почтового сервера (MTA)"

Студент: Спасенов И.В. ИУ7-51

Преподаватель: Оленев А. А.

2022 г.

Оглавление

Введение

Серверная часть SMTP агента

Задание. Вариант 10

Используется вызов pselect и единственный рабочий поток. Журналирование в отдельном процессе. Нужно проверять обратную зону днс.

Цель и задачи

Цель: Разработать **SMTP-сервер** с использованием одного потока и метода pselect(). Проверять обратную зону днс.

Задачи:

- проанализировать **SMTP**-протокол и разработать конечный автомат обработки SMTP-сообщений;
- реализовать программу для получения и сохранения писем по протоколу **SMTP** на языке программирования **C**;
- оформить расчетно-пояснительную записку.

Глава 1

Аналитический раздел

1.1 SMTP

SMTP (англ. Simple Mail Transfer Protocol — простой протокол передачи почты) — это широко используемый сетевой протокол, предназначенный для передачи электронной почты в сетях TCP/IP.

SMTP впервые был описан в RFC 821 ([1]); последнее обновление в RFC 5321 ([2]) включает масштабируемое расширение — ESMTP (англ. 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 Реализуемые команды протокола SMTP

1. *EHLO* - открывает приглашение от клиента
2. *MAIL FROM* - определяет отправителя сообщения
3. *RCPT TO* - определяет получателей сообщения
4. *DATA* - определяет начало сообщения
5. *RSET* - сбросить сеанс до состояния EHLO
6. *QUIT* - завершает сеанс SMTP

1.1.2 Метод мониторинга сетевых событий pselect

Функция pselect как и select ждет изменения статуса нескольких файловых описателей. Эти функции идентичны, за исключением 3-х отличий между ними:

1. Функция select использует время ожидания, которое задано в структуре struct timeval (с секундами и микросекундами), тогда как pselect использует struct timespec (с секундами и наносекундами).
2. Функция select может обновить параметр timeout, который показывает сколько времени прошло. Функция pselect не изменяет этот параметр.

3. Функция select не имеет параметра sigmask, и т.о. ведет себя также как функция pselect вызванная с этим параметром, установленным в NULL.

В pselect отслеживаются 3 независимых набора описателей. Те, что перечислены в readfds, будут отслеживаться для того, чтобы обнаружить появление символов, доступных для чтения (говоря более точно, чтобы узнать, не будет ли блокировано чтение; описатель файла также будет указывать на конец файла); те описатели, которые указаны в writefds, будут отслеживаться для того, чтобы узнать, не заблокирован ли процесс записи; те же, что указаны в параметре exceptfds, будут отслеживаться для обнаружения исключительных ситуаций. При возврате из функции наборы описателей модифицируются, чтобы показать, какие описатели фактически изменили свой статус.

1.1.3 ER-диаграмма предметной области

В результате проведенного исследования были выявлены следующие сущности предметной области:

1. Клиент.
2. Сервер.
3. Логгер.
4. Письмо.
5. Отправитель.
6. Получатель.
7. Данные письма.

Зависимость между сущностями предметной области может быть описана ER-диаграммой (1.1).

Рис. 1.1: ER-диаграмма предметной области

1.2 Достоинства и недостатки реализуемой архитектуры

1.2.1 Серверная часть SMTP агента

Согласно условию задачи, в работе сервера предлагается использовать один поток выполнения и один отдельный поток журналирования.

Достоинства варианта реализации:

- простота реализации, отсутствует необходимость реализации разделяемой памяти и взаимодействия между процессами или потоками;
- отсутствие времени на переключение контекстов;
- благодаря неблокирующему вводу/выводу, сервер может обслуживать множество клиентов с достаточно высокой производительностью, при условии, что обработка занимает мало времени;
- логирование в отдельном процессе позволяет не блокироваться на операциях ввода/вывода при записи в файл или в терминал;

Недостатки данной архитектуры:

- низкая производительность при длительной обработке клиентских команд;
- низкая отказоустойчивость (использование одного потока является менее надежным при возникновении фатальных ошибок в приложении, чем при наличии нескольких взаимозаменяемых потоков,);
- сложность масштабирования и использования всех аппаратных ресурсов системы.

Недостатки программной реализации с одним потоком выполнения и мультиплексированием можно уменьшить с помощью создания нескольких (пула) потоков с неблокирующим вводом/выводом и распределения нагрузки между ними.

1.2.2 Клиентская часть SMTP агента

Глава 2

Конструкторский раздел

2.1 Конечный автомат состояний сервера

На рис. 2.1 представлен сгенерированный с использованием *fsm2dot* скрипта из *autogen* файла конфигурации конечного автомата *serverfsm.def* и *dot*.

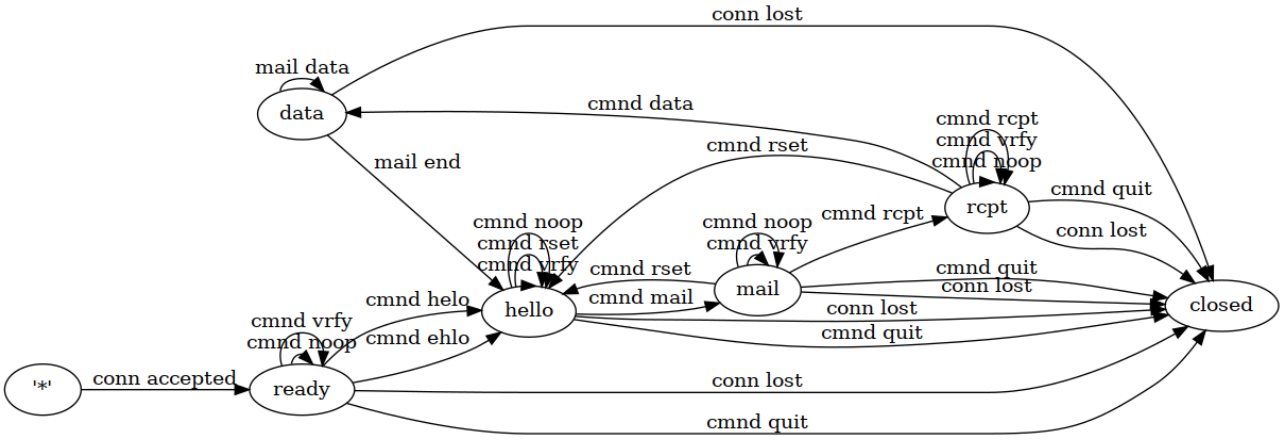

Рис. 2.1: Построенный граф конечного автомата SMTP сервера

2.2 Синтаксис команд протокола

Ниже приведен формат команд сообщений протокола в виде регулярных выражений: Регулярные выражения SMTP команд:

NOOP `[Nn][Oo][Oo][Pp]\\r\\n`

HELO `[Hh][Ee][Ll][Oo]\\s*(?<domain>.+)\\r\\n`

EHLO `[Ee][Hh][Ll][Oo]\\s*(?<domain>.+)\\r\\n`

MAIL `[Mm][Aa][Ii][Ll] [Ff][Rr][Oo][Mm]:\\s*<(?<address>.+@.+)?>\\r\\n`

RCPT `[Rr][Cc][Pp][Tt] [Tt][Oo]:\\s*<(?<address>.+@.+)>\\r\\n`

VRFY `[Vv][Rr][Ff][Yy]\\s*(?<domain>.+)\\r\\n`

DATA `[Dd][Aa][Tt][Aa]\\r\\n`

RSET `[Rr][Ss][Ee][Tt]\\r\\n`

QUIT `[Qq][Uu][Ii][Tt]\\r\\n`

Окончание данных письма `^\\.\\r\\n`

2.3 Представление данных в системе

На рис. 2.3 и на рис. 2.2 представлены логическая и физическая диаграммы представления данных в системе соответственно.

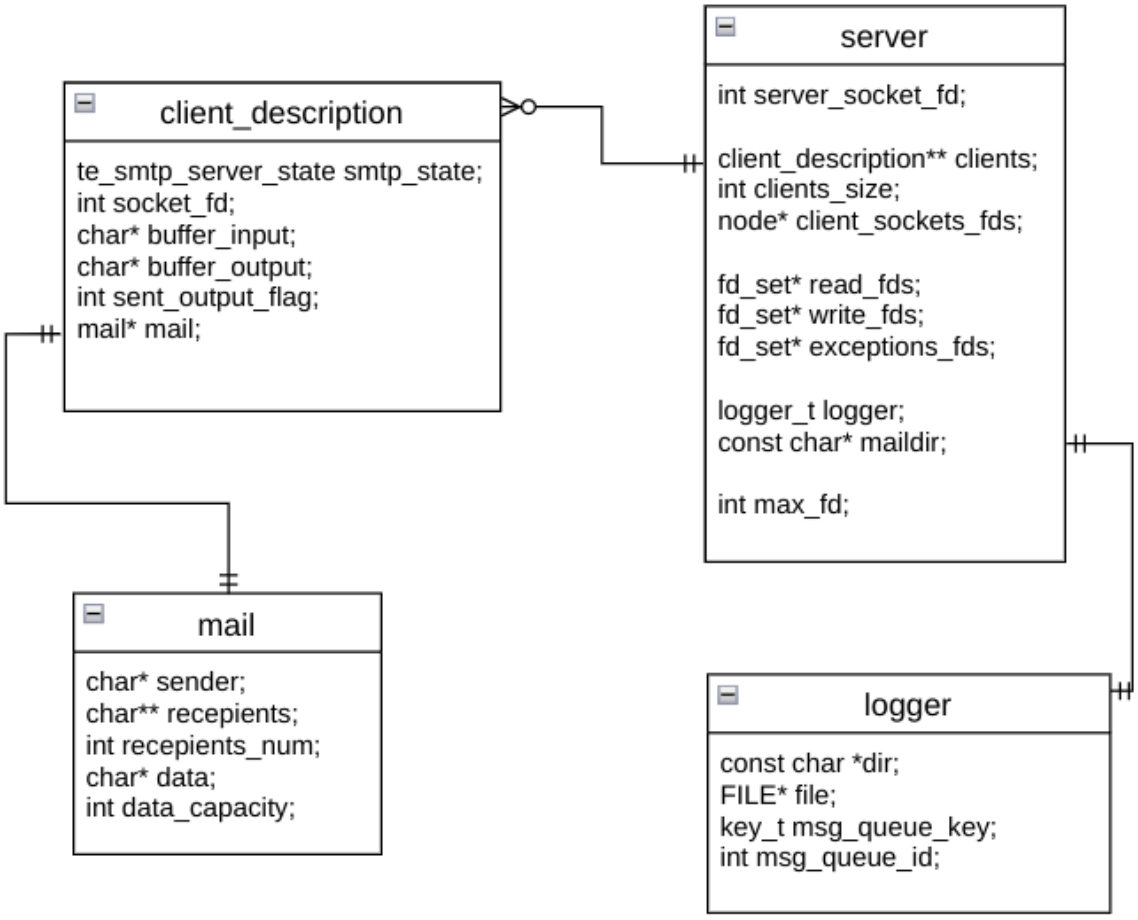

Рис. 2.2: Логическая диаграмма представления данных в серверной части системы

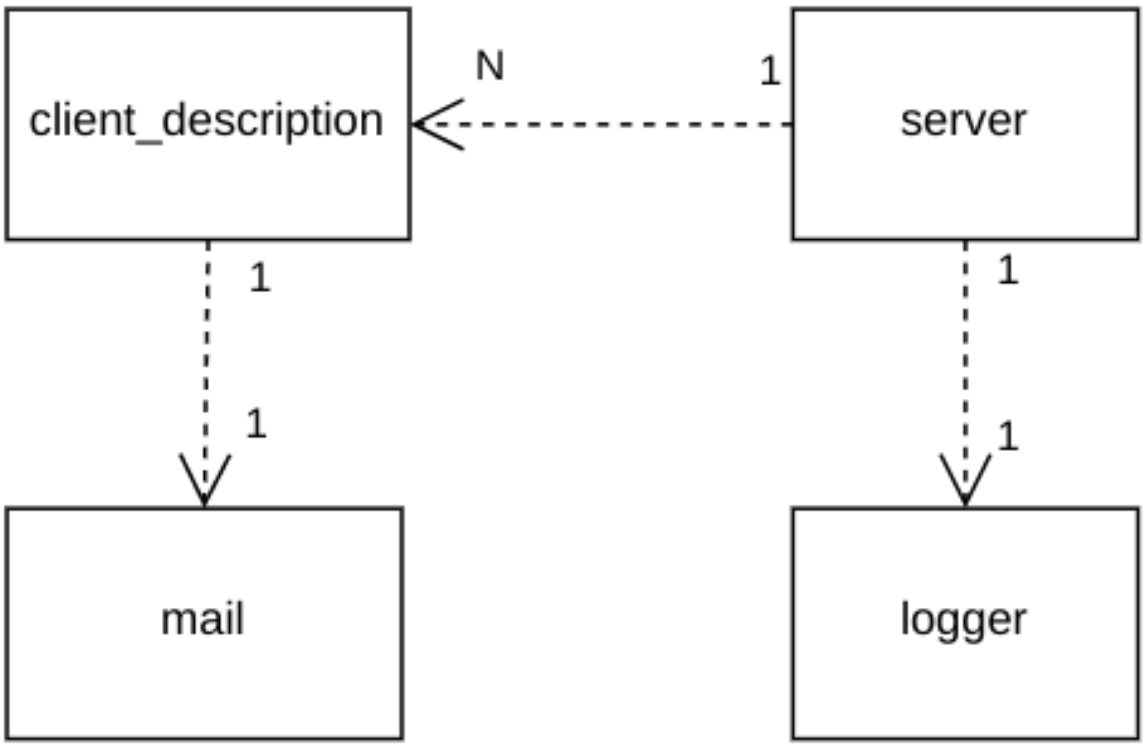

Рис. 2.3: Физическая диаграмма представления данных в серверной части системы

2.4 Обработка соединений в одном потоке выполнения

Псевдокод обработки сервером клиентских соединений в одном потоке выполнения:

```
Если пришло сообщение на сокет сервера
    Обработать новое подключение и добавить нового клиента

Цикл по всем дескрипторам сокетов клиента
     Если истек таймер ожидания команды от клиента
        Закрыть соединение с клиентом
    Если пришло сообщение на сокет клиента
        Обработать сообщение от клиента
    Если нужно отправить сообщение клиенту
        Отослать сообщение клиенту
    Если возникло исключение
        Закрыть соединение с клиентом
```

2.5 Обработка отправки в тексте письма символа окончания письма

Если клиент в тексте письма добавит EOL.EOL то сервер закончит чтение, что будет ошибкой, поэтому в текщей реализации клиент заменяет все вхождения EOL. в тексте письма клиента на EOL.., сервер делает обратную замену.

2.6 Осуществление журналирования

Процесс журналирования запускается с помощью функции fork(), взаимодействие с процессом из родительского процесса осуществляется с помощью очереди сообщений.

2.7 Хранение почты

Для хранения локальной почты, то есть предназначенной для клиентов с почтовым доменом сервера, используется упрощенный формат Maildir. Maildir - формат хранения электронной почты, не требующий монопольного захвата файла для обеспечения целостности почтового ящика при чтении, добавлении или изменении сообщений. Каждое сообщение хранится в отдельном файле с уникальным именем, а каждая папка представляет собой каталог, имя которого совпадает с адресом получателя письма.

Первоначально файл заполняется во временном месте (папка tmp), чтобы его невозможно было прочитать (из папки new) пока тот не будет полностью записан. Затем файл переносится в директорию new.

Для того, чтобы имена файлов были уникальными, в их качестве используется следующий формат: "(текущее занчение секунд).(текущее занчение миллисекунд).(случайное число)".

Глава 3

Технологический раздел

3.1 Выбор языка и библиотек для разработки клиента

В качестве языка реализации серверной части выступал C стандарта c99. В качестве компилятора выбран gcc версии 9.3.0. Для выполнения IO мультиплексировании использовалась библиотека sys/select.h. Для автогенерации конечного автомата состояний сервера и структуры параметров для запуска сервера из командной строки использовался autogen.

3.2 Выбор средств разработки

В качестве среды разработки выступил Visual Studio Code 1.64.0

3.2.1 Программное обеспечение, необходимое для сборки и запуска

Для сборки и запуска сервера должны быть загружены следующие зависимости:

- gcc version 9.3.0
- GNU Make 4.2.1
- libconfig++-dev - требуется для libconfig.h;
- libedit-dev - требуется для arpa/nameser.h;
- autogen и autogen-libopts-devel

3.3 Сборка программы

Сборка производится с помощью утилиты make([9]). make — утилита, автоматизирующая процесс преобразования файлов из одной формы в другую. Чаще всего это компиляция исходного кода в объектные файлы и последующая компоновка в исполняемые файлы или библиотеки.

Утилита использует специальные make-файлы, в которых указаны зависимости файлов друг от друга и правила для их удовлетворения. На основе информации о времени

последнего изменения каждого файла make определяет и запускает необходимые программы.

Сборка SMTP сервера состоит из следующих целей:

1. генерация исходных кодов конечного автомата и опций с помощью *autogen*;
2. сборка сервера;
3. сборка рпз, в котором помимо сборки pdf из tex файлов, осуществляется создание изображения графа конечного автомата состояний, графа cflow и графического описания make файлов;

Полная сборка осуществляется с помощью следующей команды:

```
make autogen_files && make server && make report
```

Также в make файле присутствуют цели на запуск сервера, запуск сервера под valgrind и запуск тестов.

На рисунках 3.1 и 3.2 представлено графическое описание сборки программы.

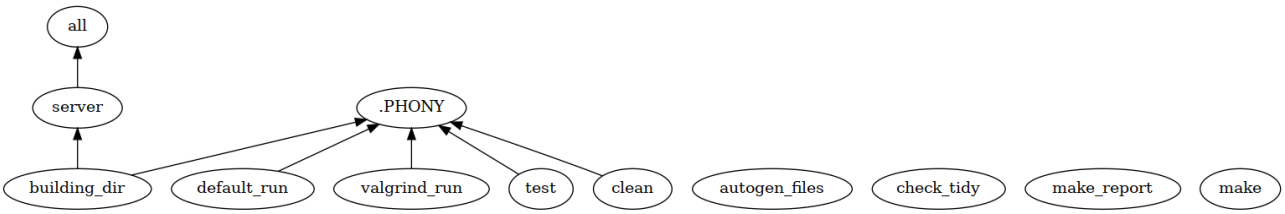

Рис. 3.1: Графическое описание сборки программы часть 1

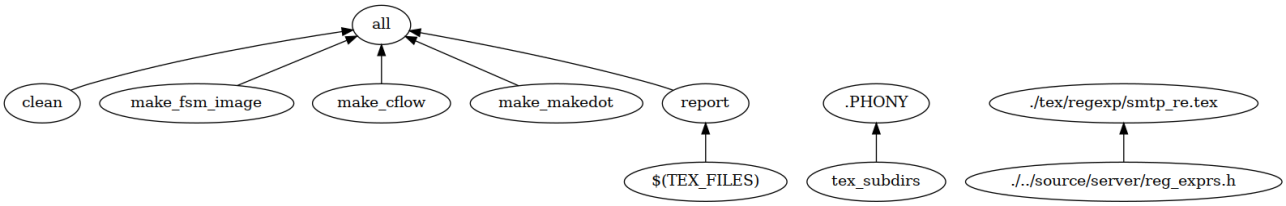

Рис. 3.2: Графическое описание сборки программы часть 2

Запуск программы сервера содержит следующие параметры:

1. -p - порт;
2. -d - путь к директории с почтой;
3. -d - путь к директории с почтой;
4. -l - Путь к директории с лог файлами;

3.3.1 Проверка обратной зоны dns

Проверка осуществлется следующим образом: сначала по номеру дескриптора клиента с помощью функции getpeername заполняется структура sockaddr, затем из нее с помощью функции getnameinfo находится имя хоста, которое сравнивается с именем пришедшем в команде helo / ehlo.

3.3.2 Граф вызова основных функций программы

Для построения графа вызова была использована утилита cflow([11]). cflow - это генератор потоковых графиков, который является частью проекта GNU. Он считывает коллекцию исходных файлов C и генерирует потоковый график C внешних ссылок. Использует только исходные тексты и не нуждается в запуске программы.

На рисунке 3.3 представлен граф вызова программы.

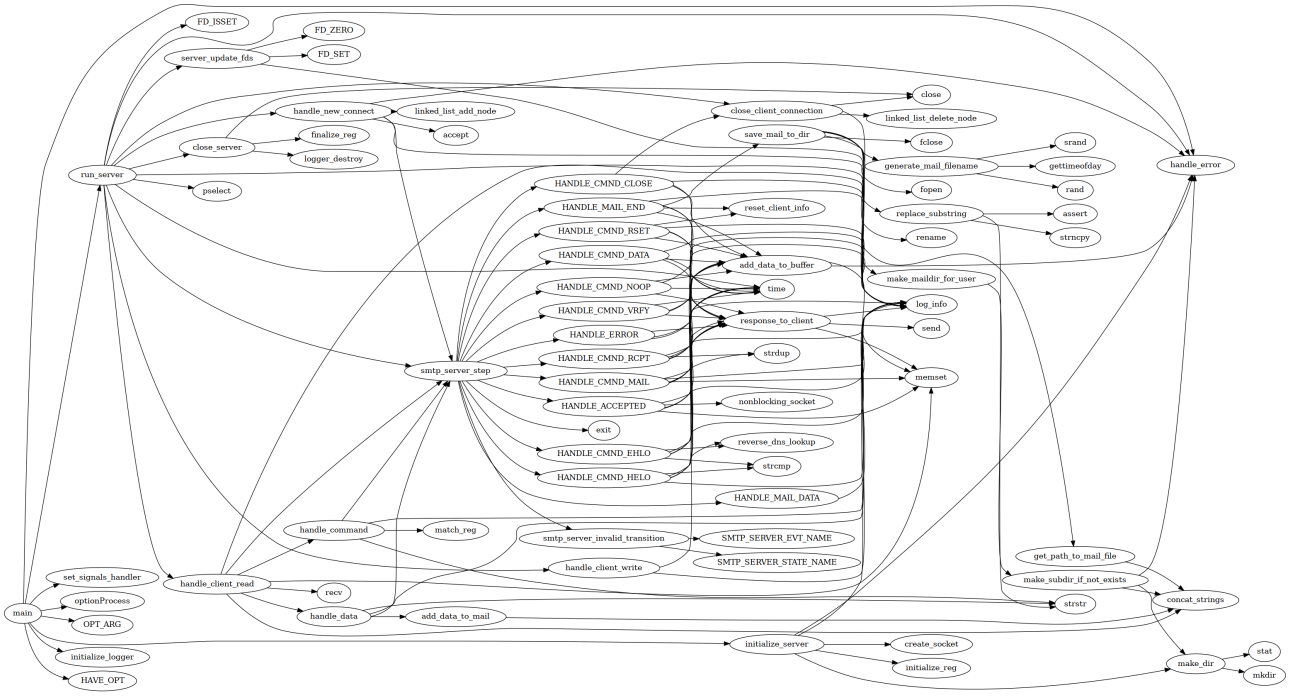

Рис. 3.3: Граф вызовов основных функций программы

3.4 Тестирование

Статический анализ кода проводился с помощью утилиты ctidy([12]). clang-tidy - это инструмент “линтера” на C++ на основе clang. Его цель - предоставить расширяемую платформу для диагностики и исправления типичных ошибок программирования, таких как нарушения стиля, неправильное использование интерфейса или ошибки, которые могут быть выявлены с помощью статического анализа. clang-tidy является модульным и предоставляет удобный интерфейс для написания новых проверок.

Проводилось интеграционное тестирование сервера по следующим пунктам:

1. передача одному получателю маленького сообщения;
2. передача нескольким получателям маленького сообщения;
3. передача одному получателю маленького сообщения с пустым адресом отправителя;
4. передача одному получателю большого сообщения;
5. передача одному получателю несколько сообщений подряд;
6. передача неавалидных команд;
7. передача rset после команд mail и rcpt;
8. передача quit после команд helo, mail, rcpt;
9. передача в тексте сообщения EOL.. и замену его на EOL.;

Тестирование проводилось с помощью скрипта, написанного на языке Python в файле test.py, для соединения с сервером использовалась библиотека socket.

Результат тестирования:

```
PASS test 1
PASS test 2
PASS test 3
PASS test 4
PASS test 5
PASS test 6
PASS test 7
PASS test 8
PASS test 9
```

Для поиска утечек использовась утилита valgrind([10]). Valgrind — инструментальное программное обеспечение, предназначенное для отладки использования памяти, обнаружения утечек памяти, а также профилирования.

Valgrind по сути является виртуальной машиной, использующей методы JIT-компиляции, среди которых — динамическая перекомпиляция. То есть, оригинальная программа не выполняется непосредственно на основном процессоре. Вместо этого Valgrind сначала транслирует программу во временную, более простую форму, называемую промежуточным представлением (Intermediate Representation, сокр. IR), которая сама по себе не зависит от процессора и находится в SSA-виде. После преобразования инструмент может

выполнять любое необходимое преобразование IR до того, как Valgrind оттранслирует IR обратно в машинный код и позволит основному процессору его исполнить. Её используют, даже несмотря на то, что для этого может использоваться динамическая трансляция (то есть, когда основной и целевой процессоры принадлежат к разным архитектурам). Valgrind перекомпилирует двоичный код для запуска на основном и целевом (или его симуляторе) процессорах одинаковой архитектуры.

При проведении тестирования valgrind был запущен с настройками, представленными ниже

```
 --tool=memcheck --track-origins=yes  --leak-check=full --show-reachable=yes
```

Отчет valgrind после прогона тестов:

```
==166488==
==166488== HEAP SUMMARY:
==166488==     in use at exit: 0 bytes in 0 blocks
==166488==   total heap usage: 34 allocs, 34 frees, 536,445 bytes allocated
==166488==
==166488== All heap blocks were freed -- no leaks are possible
==166488==
==166488== For lists of detected and suppressed errors, rerun with: -s
==166488== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
==166487==
==166487== HEAP SUMMARY:
==166487==     in use at exit: 0 bytes in 0 blocks
==166487==   total heap usage: 25,774 allocs, 25,774 frees, 13,498,089 bytes allocated
==166487==
==166487== All heap blocks were freed -- no leaks are possible
==166487==
==166487== For lists of detected and suppressed errors, rerun with: -s
==166487== ERROR SUMMARY: 298 errors from 1 contexts (suppressed: 0 from 0)
```

Выводы

3.5 Серверная часть SMTP агента

В результате выполнения курсового проекта была достугнута поставленная цель, а именно разработан **SMTP-сервер** с использованием одного потока и метода pselect(), осуществляющее прием и сохранение писем для дальнейшей поставки их пользователям, с проверкой обратной зоны dns.

Во время выполнения работы были выполнены следующие задачи:

- проанализирован **SMTP**-протокол и разработан конечный автомат обработки SMTP-сообщений;
- реализована программу для получения и сохранения писем по протоколу **SMTP** на языке программирования **C**;
- оформлена расчетно-пояснительную записка.

В ходе работы не были реализованы следующие пункты, планируемые к разработке в дальнейшем:

- UNIT тестирование;

Литература

[1] Протокол передачи данных SMTP в редакции RFC821 https://www.ietf.org/rfc/rfc821.txt

[2] Протокол передачи данных SMTP в редакции RFC5321 https://www.ietf.org/rfc/rfc5321.txt

[3] Формат SMTP сообщений в редакции RFC2045 https://www.ietf.org/rfc/rfc2045.txt

[4] Обзор методов IO мультиплексирования https://habr.com/ru/company/infopulse/blog/415259/

[5] Устройство системного вызова poll https://programming.vip/docs/i-o-multiplexing-poll-system-call.html

[6] C99 standard (ISO/IEC 9899:1999): 7.23.2.4 The time function (p: 341) http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf

[7] C11 standard (ISO/IEC 9899:201x): http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf

[8] представлено MX записи https://semantica.in/blog/chto-takoe-mx-zapis-domena.html

[9] официальный сайт разработчиков make https://www.gnu.org/software/make/manual/make.html

[10] официальный сайт разработчиков valgrind https://valgrind.org/

[11] официальный сайт разработчиков cflow https://www.gnu.org/software/cflow/manual/cflow.html

[12] официальный сайт разработчиков clang tidy https://clang.llvm.org/extra/clang-tidy/