Оглавление

Введение

0.0.1 Цели и задачи

Цель: Разработать **SMTP-клиент** используя вызов select (или poll) и единственный рабочий процесс. Журналирование в отдельном процессе.

Задачи:

- Проанализировать архитектурное решение
- Разработать подход для обработки исходящих соединений и отправки (пересылки) писем MTA-сервера из директории maildir
- Рассмотреть **SMTP**-протокол
- Реализовать программу для отправки писем по протоколу **SMTP**
- Реализовать метод журналирования в отдельном процессе**SMTP**

Глава 1

Аналитический раздел

Предметная область

Согласно обозначенному протоколу в рамках данной работы, в системе устанавливаются отношения "отправитель - получатель причем отправитель может отправить письмо нескольким получателям. Основная единица данных, передаваемая по протоколу - письмо, которое включает в себя отправителя и получателя, причем получателей может быть несколько. Также письмо содержит в себе единственное тело, которое может быть использовано как для последующей передачи, так и для хранения на сервере. Таким образом, в рамках предметной области можно выделить 3 вида сущностей:

- 1. Сервер
- 2. Клиент
- 3. Письмо

Клиент

Преимущества и недостатки условия задачи

Согласно условию задачи, в работе клиента предлагается исполльзовать однопроцессную асинхронную систему. Данная архитектура имеет следующие преимущества:

- 1. Лучшая производительность по сравнению с простой многопроцессной (как и многопоточной) схемой за счёт оптимального использования ресурсов процесса в моменты его вынужденных "простоев"на какаом-либо соединении из-за неизбежных временных потерь ввиду "латентности"сетевой передачи данных.
- 2. Возможность не использовать потенциально подверженные сложноотлаживаемым ошибкам средства межпроцессного взаимодействия.
- 3. Любая прикладная задача представима в однопроцессной схеме работы (в отличие от параллельной, многопроцессной или многопоточной).

При этом данная архитектура имеет следующие недостатки:

– 1. Сложность реализации. В отличие от многопроцессной или многопоточной схемы, асинхронная система подразумевает "пересечение"в одном потоке кода для выполнения сразу нескольких, логически независимых задач, что затрудняет их проектирование и отладку

– 2. Простая асинхронная обработка имеет свои границы масштабирования, за которыми необходимо объединять асинхронность с многопоточностью.

Глава 2

Конструкторский раздел

2.1 Конечный автомат состояний клиента

Конечный автомат состояний клиента представлен на Рис. **??**

2.1.1 Синтаксис команд протокола

Ниже приведен формат команд сообщений протокола в виде регулярных выражений

1. **EHLO**: *EHLO [w+]+*
2. **HELO**: *HELO [w+]+*
3. **MAIL**: *MAIL FROM <[\w]+@[\w]+[\w]+>*
4. **RCPT**: *RCPT <[\w]+@[\w]+[\w]+>*
5. **DATA**: *DATA*
6. **RSET**: *RSET*
7. **QUIT**: *QUIT*

Представление данных

Ниже приведена диаграмма представления данных в системе

2.1.2 Алгоритм обработки соединений

```
ПОКА (процесс_работает == 1)
     Добавить дескрипторы слушающих сокетов в множество_клиентских_сокетов
     Ожидать соединения на одном из сокетов (время = 5с)
     ЕСЛИ есть запрос ТО
       ДЛЯ каждого слушающего сокета
          ЕСЛИ действие на одном из слушающих сокетов ТО
```

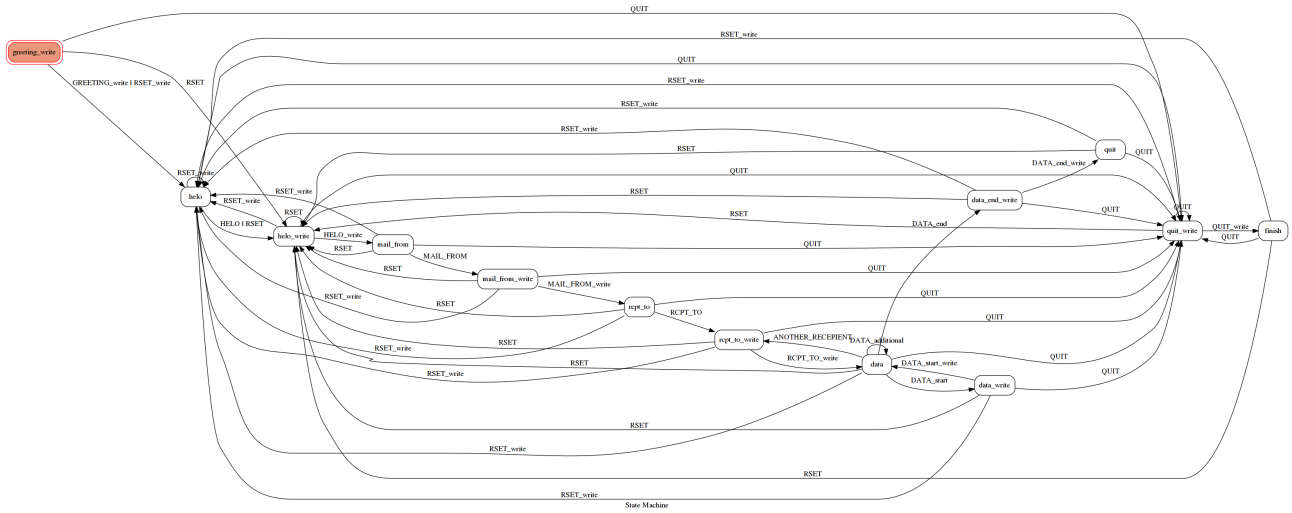

Рис. 2.1: Состояния клиента

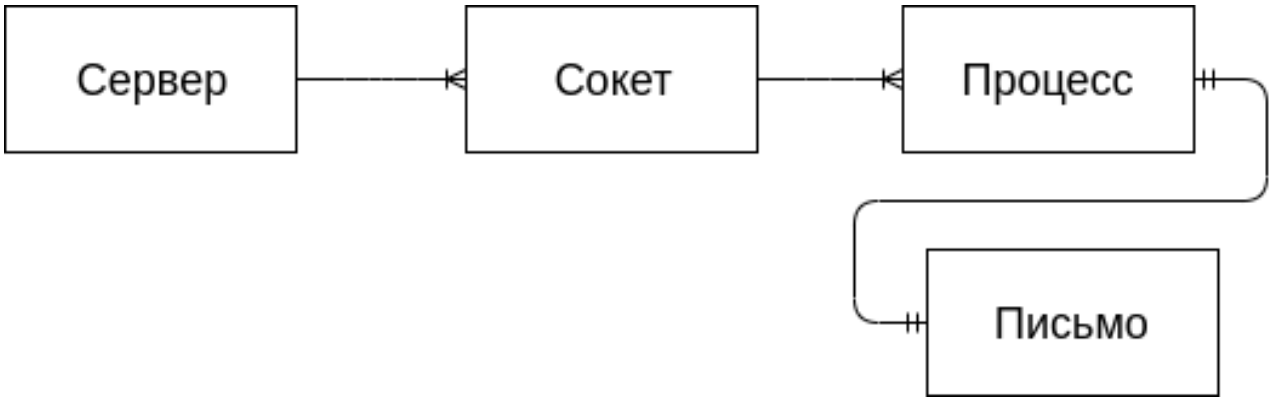

Рис. 2.2: Физическая диаграмма сущностей

```
            Принять новое соединение
            Инициализировать новый сокет
         КОНЕЦ ЕСЛИ
         ЕСЛИ действие на одном из клиентских сокетов ТО
          Обработать действие в соответствии с протоколом
         КОНЕЦ ЕСЛИ
       КОНЕЦ ДЛЯ
       ДЛЯ каждого сокета записи
         ЕСЛИ действие на одном из клиентских сокетов ТО
            Обработать действие в соответствии с протоколом
         КОНЕЦ ЕСЛИ
       КОНЕЦ ДЛЯ
     КОНЕЦ ЕСЛИ
КОНЕЦ ПОКА
```

Глава 3

Технологический раздел

3.1 Сборка программы

Сборка программы описана в файле *Makefile* системы сборки *make*. Рис. 3.1. Изображение с конечным автоматом генерируется срествами библиотеки transitions.

3.2 Графы вызова функций

Поскольку функций много, графы вызовов разбиты на два рисунка. На рис. 3.2 показаны основные функции, на рис. **??** – функции обработки команд.

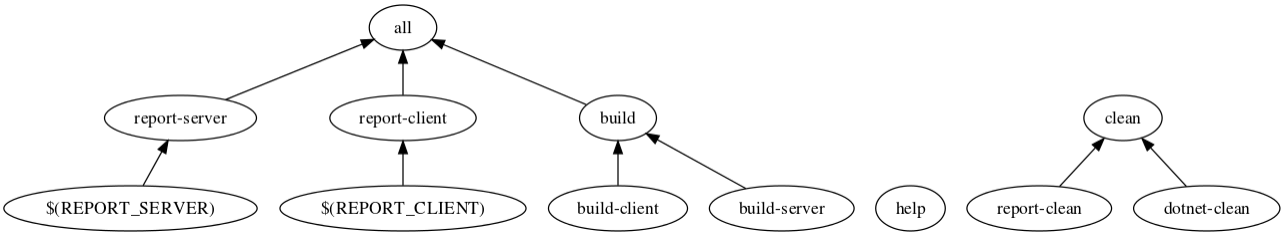

Рис. 3.1: Сборка программы

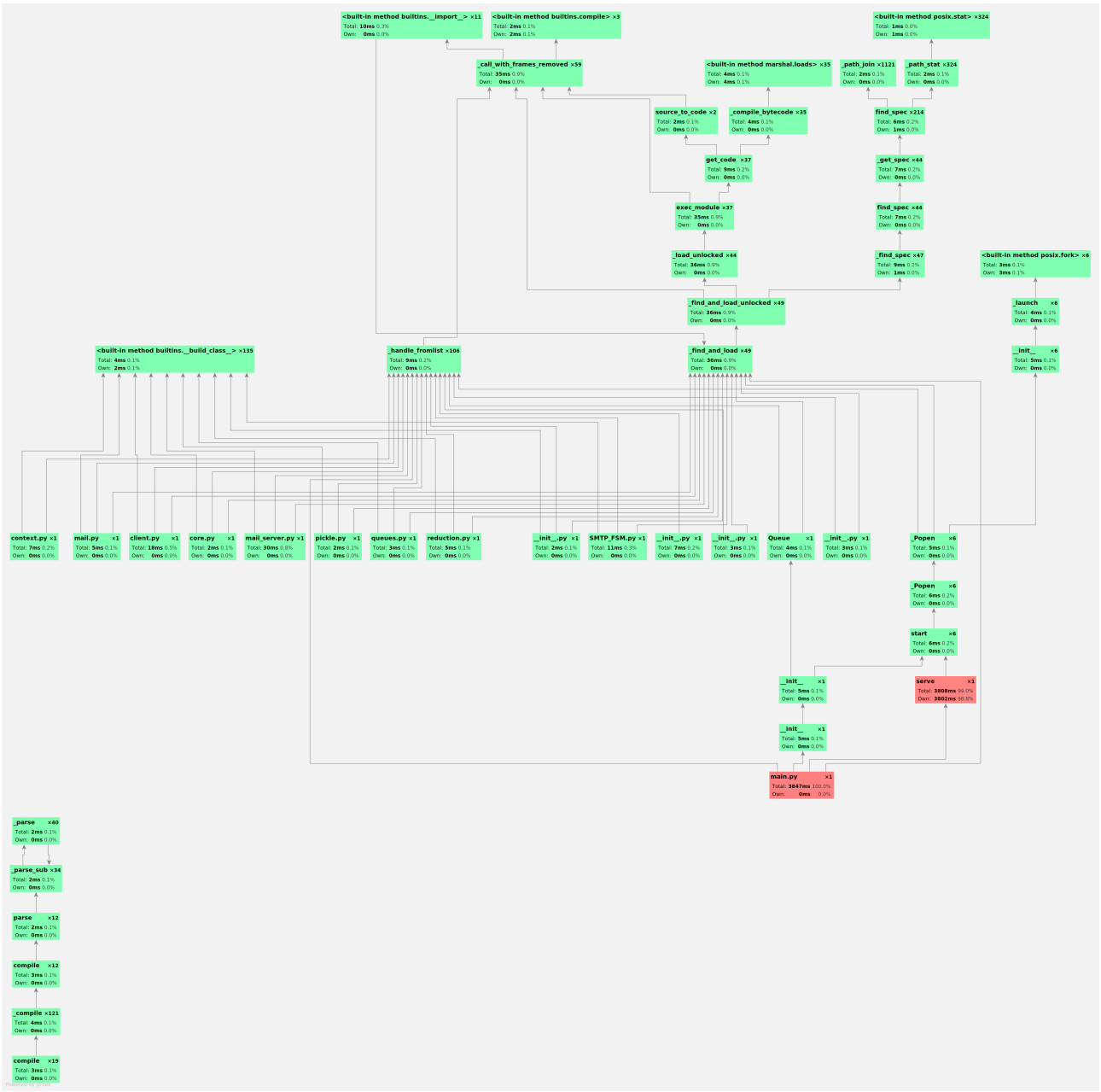

Рис. 3.2: Граф вызовов

3.3 Тестирование

Ниже приведён отчет о модульном тестировании.

```
Testing started at 10:44 ...
/home/sbn/PycharmProjects/smtp-course-work/venv/bin/python /home/sbn/Downloads/pycharm-c
Launching pytest with arguments test_sockets.py::test_send_simple_message_socket in /hom

============================= test session starts ==============================
platform linux -- Python 3.8.0, pytest-5.3.2, py-1.8.1, pluggy-0.13.1 -- /home/sbn/Pycha
cachedir: .pytest_cache
rootdir: /home/sbn/PycharmProjects/smtp-course-work/client/tests
plugins: cov-2.8.1
collecting ... collected 2 items

test_sockets.py::test_send_simple_message_socket PASSED                      [100%]
test_reg_exp_s.py::test_ALL_patterns_CLIENT PASSED                         [100%]

============================== 2 passed in 10.60s==============================
```

Выводы

В рамках предложенной работы нами был реализован SMTP-клиент в соответствии со стандартами RFC. В ходе работы реализованы следующие задачи:

1. Проанализировали архитектурное решение
2. Разработан подход для обработки исходящих соединений и отправки исходящих писем MTA-сервкера из директории maildir
3. Рассмотрен **SMTP**-протокол
4. Реализована программа для отправки писем по протоколу **SMTP**
5. Рассмотрена работа с неблокирующими сокетами и их взаимодействие
6. Реализован метод журналирования в отдельном процессе**SMTP**
7. Разработана система, работающая в многозадачном режиме (основной процесс, процесс журналирования и т.д.)
8. Произведено ознакомление с утилитами автоматической сборки и тестирования

В ходе работы получены следующие навыки:

1. проектирования реализации сетевого протокола по имеющейся спецификации;
2. реализации сетевых приложений на языке программирования;
3. реализации сетевой службы без создания нити на каждое соединение;
4. автоматизированного системного тестирования ПО сетевой службы;
5. групповой работы с использованием системы контроля версий;