

# Оглавление

|  |           |
|--|-----------|
| [Pleaseinsertintopreamble][Pleaseinsertintopreamble][Pleaseinsertintopreamble][Pleaseinsertintopreamble] |           |
| 0.0.1 Цели и задачи . . . . .  | 4         |
| <b>1 Аналитический раздел</b>  | <b>5</b>  |
| <b>2 Конструкторский раздел</b>  | <b>7</b>  |
| 2.1 Конечный автомат состояний клиента . . . . .   | 7         |
| 2.1.1 Синтаксис команд протокола . . . . .   | 7         |
| 2.1.2 Алгоритм обработки соединений . . . . .  | 8         |
| <b>3 Технологический раздел</b>  | <b>9</b>  |
| 3.1 Сборка программы . . . . .   | 9         |
| 3.2 Графы вызова функций . . . . .   | 9         |
| 3.3 Тестирование . . . . .   | 11        |
| <b>Выводы</b>  | <b>11</b> |

# 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 МТА-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 использовать 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 каком-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.

# 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

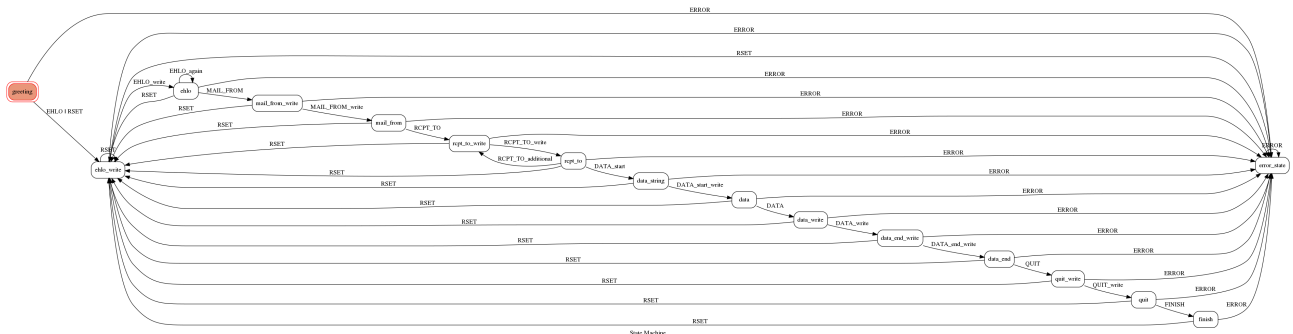


Рис. 2.1: Состояния клиента

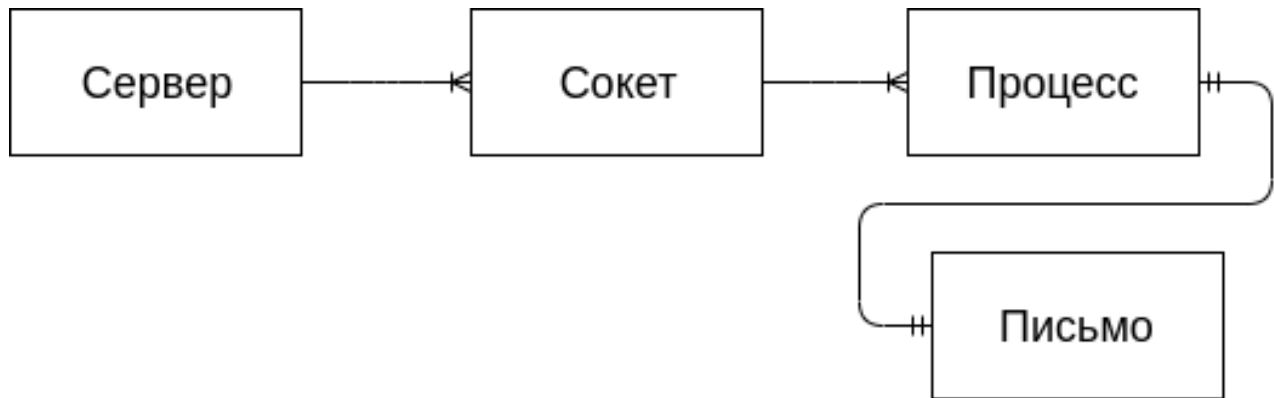


Рис. 2.2: Физическая диаграмма сущностей

### 2.1.2 Алгоритм обработки соединений

```

ПОКА (процесс_работает == 1)
    Проверить директорию почты maildir на наличие необработанных сообщений
    ЕСЛИ сообщения есть, создать новый клиентский_сокет
    КОНЕЦ ЕСЛИ

    ОБНУЛИТЬ список_сокетов
    ДЛЯ каждого клиентского_сокета:
        добавить клиентский_сокет в список_сокетов,
        ЕСЛИ список_сокетов НЕ пуст:
            применить SELECT (или POLL) для получения списка дескрипторов на чтение
            ДЛЯ каждого дескриптора в ДЧ:
                выбрать клиентский_сокет из списка_сокетов
                запустить для него обработчик_состояний_FSM для чтения_из_сокета
            КОНЕЦ ДЛЯ
        ДЛЯ каждого дескриптора в ДЗ:
            выбрать клиентский_сокет из списка_сокетов
            запустить для него обработчик_состояний_FSM для записи_в_сокет
        КОНЕЦ ДЛЯ
    КОНЕЦ ЕСЛИ
    КОНЕЦ ДЛЯ
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 средствами 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.

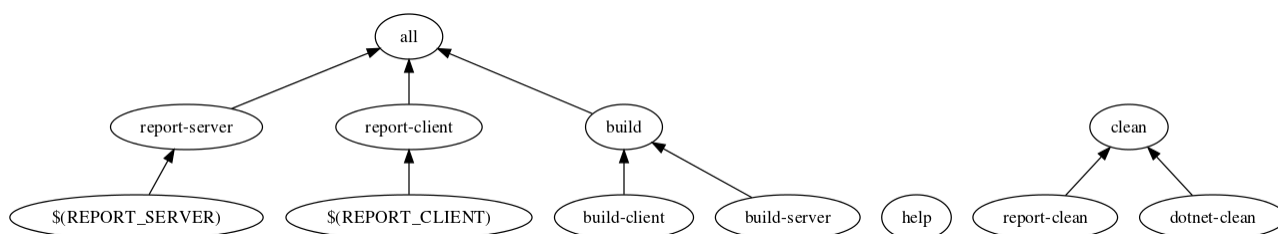


Рис. 3.1: Сборка программы





### 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 /home

===== 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 МТА-сервера 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;