Отчёт по курсовому проекту

(Бородин Дмитрий Сергеевич, Федоров Павел Вячеславович)

12 января 2019 г.

Оглавление

Введение

Сервер

Клиент

Задание. 26

Используется вызов pselect и рабочие потоки. Журналирование в отдельном потоки. Не обязательно пытаться отправлять все сообщения для одного MX за одну сессию.

Цели и задачи

Цель: Разработать **SMTP-клиент** с использованием рабочих потоков и pselect.
Задачи:

- Проанализировать архитектурное решение maildir
- Разработать подход для обработки писем в maildir
- Рассмотреть **SMTP**-протокол
- Проанализировать способы получения и обработки **MX**-записей
- Реализовать программу для отправки писем по протоколу **SMTP**

Глава 1

Конструкторский раздел

1.1 Сервер

1.1.1 Конечный автомат состояний сервера

Рис. 1.1 нагенерил самодельный *fsm2dot* из *autogen* и *dot2tex* на пару *dot*. Никто не мешает изменить параметры типа *rankdir* прямо в *fsm2dot*, если он будет лучше смотреться, например, сверху-вниз.

1.2 Клиент

1.2.1 Принцип действия

Работу клиента можно разделить на 2 отдельные сущности: работа основного потока, работа рабочих потоков *(далее воркеры)* В основном потоке решено производить следующие действия

1. Создание при инициализации и удаление при завершении списка воркеров
2. Циклический поиск новых писем
3. Распределение писем по воркерам
4. Перехват сигналов на завершение работы

При запуске сервера, главный поток создает пул воркереров и помещает в него логгер и рабочие потоки. Количество рабочих потоков напрямую зависит от количества процессоров на устройстве. Однако, для устройств в которых процессоров меньше чем 3 создается обязательно 1 рабочих процесс Для *Изящного завершения* необходимо подписаться на перехват сигналов. Для более корректного и однозначного управления потоками программы необходимо, чтобы рабочие потоки не перехватывали общие сигналы с главным потоком. Для этого решено использовать 2 сигнала **SIGINT** для завершения программы – перехватывается главным потоком и **SIGUSR1** для управления воркерами. Во время корректного завершения главный поток оповещает всех воркеров по очереди в своем спике с помощью сигнала и ожидает их завершения. После завершения необходимо удалить данные воркера и освободить все ресурсы занимаемые главным потоком.

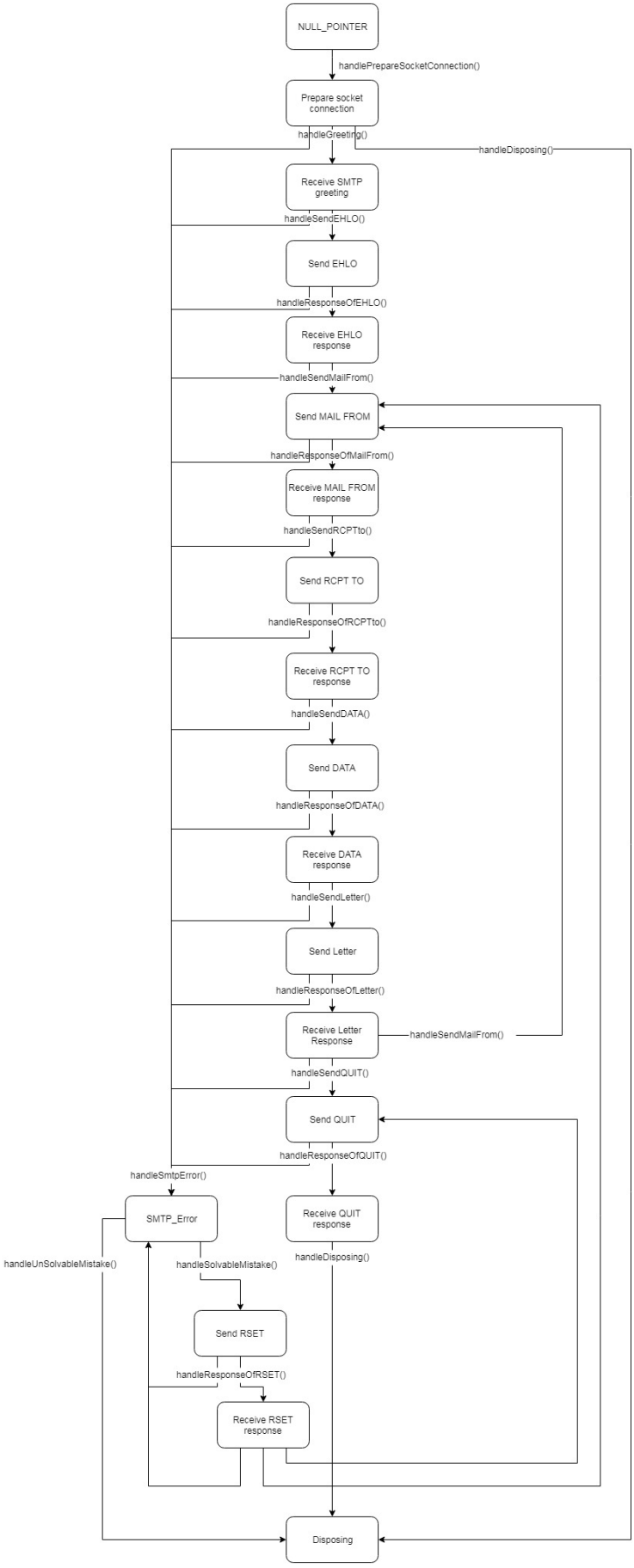

Рис. 1.1: Состояния сервера

Рабочие процессы в данном случае выполняют роль связных. Их деятельность можно описать следующими действиями:

1. Контролирует список соединений
2. Получает и обрабатывает задания от главного потока
3. Отправляет письма

Воркер во время инициализации получает служебную информацию для поддержания связи с главным потоком. Так как в требованиях к заданию указывается, что соединение должно быть неблокируемым, каждый воркер обладает множествами пишущих, читающих и обработчиков прерываний. Помимо указаных множеств в данном блоке информации хранится список активных соединений (сокетов) Во время работы воркер ожидает на **pselect** готовых к работе соединений. Однако, перед этим необходимо распределить новые задания от главного потока. После обработки всех заданий воркер для каждого готового соединения передает управление SMTP контроллеру до тех пор, пока:

- Текущее состояние соединения - в процессе
- Текущее состояние соединения - заблокирован
- Дальнейших действий в SMTP контроллере по данному соединению произвести невозможно.

Если по текущемо соединению все задания на текущий момент были завершены, то активное соединение удаляется из списка. Стоит отметить что, для того чтобы новые заявки не накапливались во время ожидания активных соединений, решено использовать таймер. Таким образом, все новые заявки будут распределяться по своим соединениям даже когда они не готовы. Ранее, отмечалось, что воркер управляется главным потоком через сигнал. Он используется как для оповещения новых заданий, так и для завершения работы. Принцип *"Изящного завершения"* должен учитывать состояния соединений воркера. Поэтому заверешние воркера заключается в уменьшении текущих задач. Для всех соединений задания, которые необходимо будет выполнить(не текущее) удаляется из списка заданий, а само письмо переноситься в начальную директорию (/new) пользователя. В случае, если соединение еще не было корректно установлено (для данного соединения еще не было приветствия от SMTP сервера), тогда первое задание тоже удаляется, файл переноситься, а соединение закрывается и удаляется. Текущие же задания, если стадия "приветствия"прошла успешно обрабатываются до конца. После обработки всех оставшихся соединений, воркер освобождает занимаемые им ресурсы и сообщает о завершении работы.

1.2.2 Обработка новых писем

Для отправки писем, необходимо найти письма в maildir и отправить их воркерам на выполнение. Функцию поиска писем выполняет главный поток. После инициализации всех необходимых структур и данных главный поток проверяет maildir на наличие новых писем. Сначала формируется список пользователей SMTP клиента. После, в каждой директории из списка проверяется директория **new** на наличие новых писем. Если в данном каталоге есть файлы, необходимо выделить информацию о адресате письма. Адресат

письма получается из поля **To** письма. На его основе формируется домен, ответственный за прием писем пользователей. Найденное письмо перемещается в директорию **tmp** и для него формируется заявка на отправку. В заявке указан ответственный домен и путь к письму. После производится поиск подходящего воркера и задание на оптравку делегируется. Когда список файлов на отправку освобождается главный поток ожидает 30 секунд и повторяет поиск заново. Для корректного распределения используется метрика загруженности воркеров. Она отображает сколько задач уже было делегировано воркеру.

1.2.3 Словарь

Для контроля распределения заявок между воркерами, а так же отправки писем через одну соединение, хоть это и не обязательно, решено использовать словарь. В данном словаре хранится информация о распределении доменов между воркерами. Во время поиска писем, главный поток узнает ответственный домен и смотрит свой словарь доменов. Если в нем нет воркера, который уже работает с данным доменом, то выбирается наимение загруженный воркер. В случае если есть воркер, то новое письмо делегируется ему. Рабочий процесс также взаимодействует с данным словарем. Когда список заданий для какого-либо определенного домена закончился, соединение переходит в режим "завершения"отправки писем. В данный момент и воркер удаляет запись в словаре для данного домена. Таким образом, система старается(не гарантирует) отправить все письма адресованные одному домену через одно активное соединение.

1.2.4 Рабочий поток

Каждый рабочий поток обладает записью с необходимой информацией о себе. Данная информация передается ему при создании. С помощью общей с главным потоком блоком информации, воркер может получать новые задания узнавать о текущем состоянии работы программы. В данном блоке целесообразно хранить следующую информацию:

- ссылку на очередь новых заданий
- личный идентификатор (удобно для логгирования)
- состояние работы
- количество заданий

Доступ к очереди заданий рабочего процесса имею два потока: главный и сам рабочий поток. Для исключения коллизий, потери данных необходимо использовать объект синхронизации. В качестве такого объекта синхронизации является семафор. Для корректной работы, каждый воркер создает во время подготовки структуру, с помощью которой будет осуществляться контроль соединений и их состояний. Разграничение между состояниями соединений необходимо учитывать во время передачи данных. Для этого в данной структуре содержатся множества ожидающих на запись и ожидающих на чтение соединений.

Параметр состояния работы, указывает рабочему процессу на необходимость корректного и быстрого завершения работы. В связи с этим, рабочий процесс, узнав о необходимости завершения, убирает все задания, которые еще не выполнялись у каждого соединения. Также изменяется время ожидания активизации соединений. Это необходимость обусловлена неблокируемостью сокетов, из-за чего у соединения нет как такого *timeout*

на получение или отправку. Стоит отметить, что у каждого соединения текущее задание остается для корректного завершения обмена информацией с серверами.

Каждое активное соединение обрабатывается с помощью SMTP контроллера

1.2.5 Активное соедиение

Каждое соединенние должно быть однозначно определено. Для контроля соединений целесообразно описать соединение следующим образом

- идентификатор соединения
- домен сервера
- адрес сервера
- текущее состояние соединения
- предыдущее состояние соединения
- количество попыток отправки одного письма
- список писем
- данные текущего письма

Во время подготовки к соединению необходимо разрешить адрес сервера. Для этого для указанного домена необходимо запросить MX запись. Из этой записи получается доменное имя самого SMTP сервера, с которым необходимо обмениваться данными. Однако, для корректного обмена необходимо узнать его IP адрес в сети. Для этого полученной записи запрашивается список IP адресов и выбирается первый успешно обработанный. Для оптимальной обработки писем, целесообразно хранить в памяти только служебную информацию, которая необходима для передачи письма. Для этих целей выделена отдельный блок "данные письма". В данной блоке хранится адресат и адресант письма. Это позволяет сократить время на открытия файл и взаимодействия с ним во время установленного соединения

1.2.6 SMTP контроллер

SMTP контроллер отвечает за корректность общения с серверами. Для контроля обмена информацией данный контроллер ослеживает состояния общения и ведет их учет. Таким образом исключается возможность непредвиденных переходов, отправки некорректного сообщения, а так же исключается очередность отправки/получения сообщений между SMTP клиентом и SMTP сервером. Состояния соединений и переходы между ними отображены на Рис. 1.2

Стоит отметить, что во при переходе к завершению обмена данными по данному соединение именно SMTP контроллер удаляет запись из *словаря*. Это необходимо для исключения появления писем по закрывающемуся соединению. Таким образом, главный поток не будет делегировать письма воркеру, который завершает процесс передачи для данного домена и выберет наиболее свободного воркера. Отправка команд протокола SMTP сопровождается верификацией ответов от сервера. Каждый ответ сервера проверяется

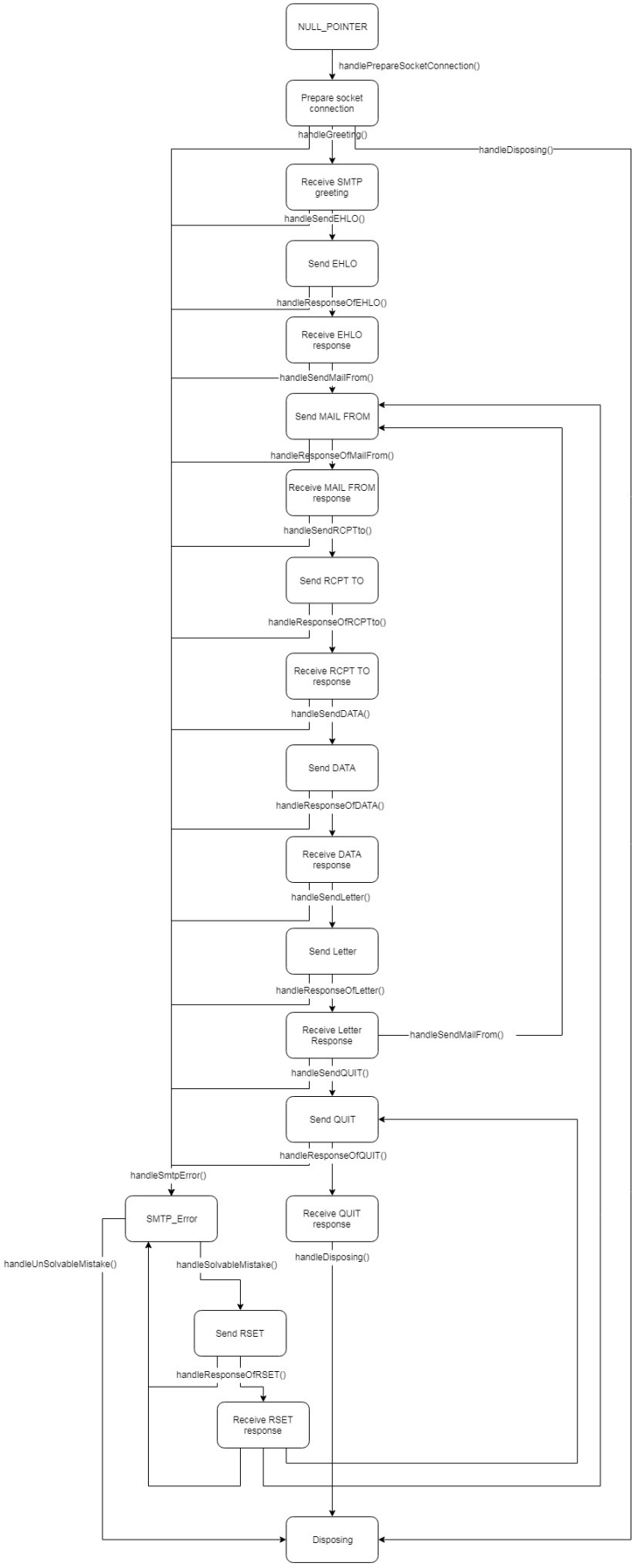

Рис. 1.2: Конечный автомат состояний соединений SMTP клиента

на соблюдение регламента передачи. Так, во время подключения к SMTP серверу и получения сообщения от сервера с доменным именем за который он отвечает, происходит верификация статуса ответа и самого домена. В случае нарушения соглашения: некорректный ответ или указанный в сообщение домен не совпадает с указанным в соединение, возникает ошибка, которая разрешается закрытием соединения и освобождению ресурсов.

1.2.7 Синтаксис поддерживаемых команд протокола

В данном разделе приведены поддерживаемые клиентом команды протокола SMTP.

- **EHLO**: *EHLO [w+]+*
- **MAIL**: *MAIL FROM: <[\w]+@[\w]+[\.\w]+>*
- **RCPT**: *RCPT To:<[\w]+@[\w]+[\.\w]+>*
- **DATA**: *DATA*
- **RSET**: *RSET*
- **QUIT**: *QUIT*

Глава 2

Технологический раздел

2.1 Сервер

2.2 Клиент

2.2.1 Графы вызова функций

2.2.2 Основные данные

Стркутура рабочего процесса

```
struct worker
{
    pthread_t thread;
    sem_t lock;
    struct worker_task *tasks;
    int workerId;
    int count_task;
    bool worked;
};
```

2.3 Сборка программы

Сборка программы описана в файле *Makefile* системы сборки *make*. Рис. 2.3 нагенерили самодельные *makesimple* и *makefile2dot*, а также *dot2tex* и *dot*.

Отмечу, что за исключения целей типа *all*, *install*, *clean*, *tests*, все имена целей в файле систем сборки *make* обычно совпадают с именами файлов (такой вот низкоуровневый инструмент). То есть вместо цели *lexer* следует использовать цель *src/lexer.c*.

2.4 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*.

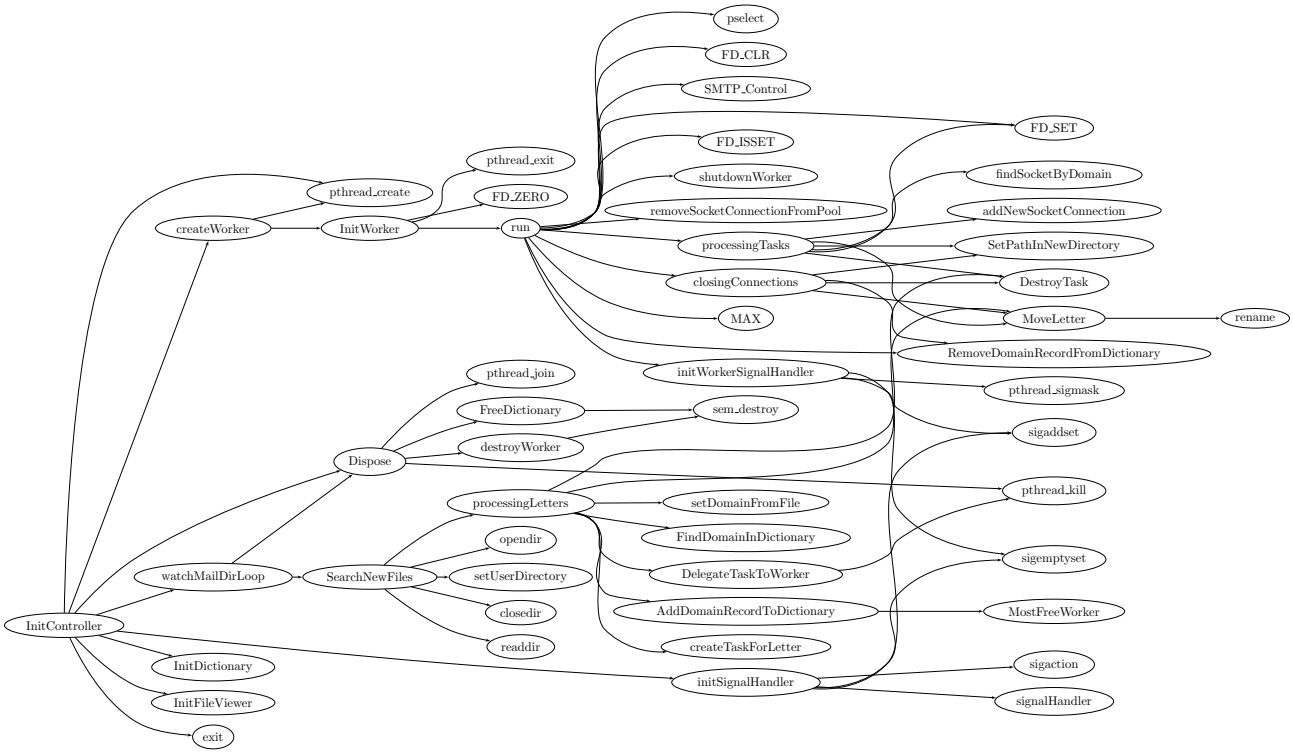

Рис. 2.1: Граф вызовов, основные функции

2.5 Графы вызова функций

Поскольку функций много, графы вызовов разбиты на два рисунка. На рис. 2.4 показаны основные функции, на рис. 2.7 – функции обработки команд. Файл **cflow.ignore** содержит список функций (точнее, шабловнов поиска), использыемых программой *grep* для удаления малоинтересных стандартных функций[1].

[1] Функции по работе с сокетами, ipc и привилегиями к малоинтересным ни в коем случае не относятся.

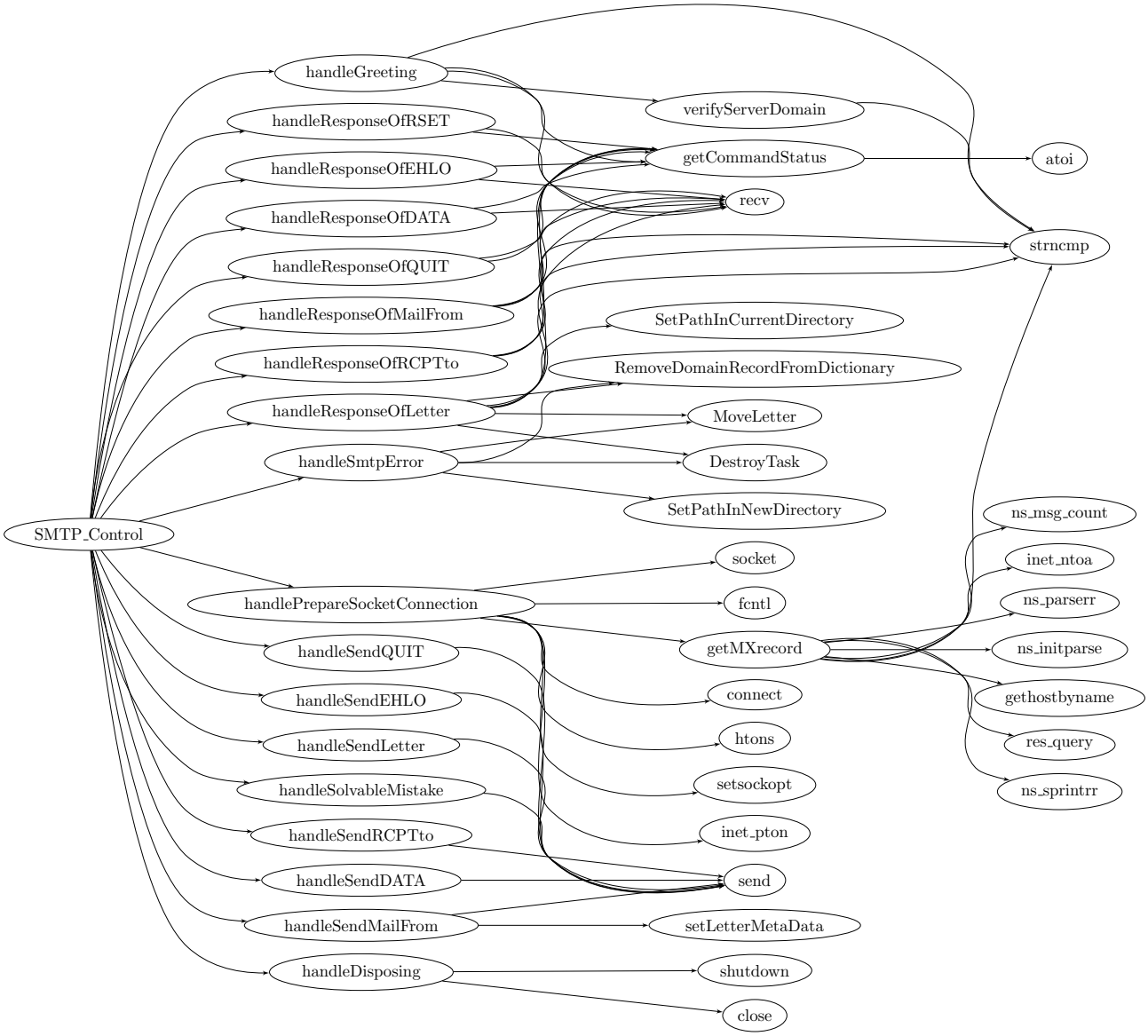

Рис. 2.2: Граф вызовов, взаимодействие с сервером

Рис. 2.3: Сборка программы

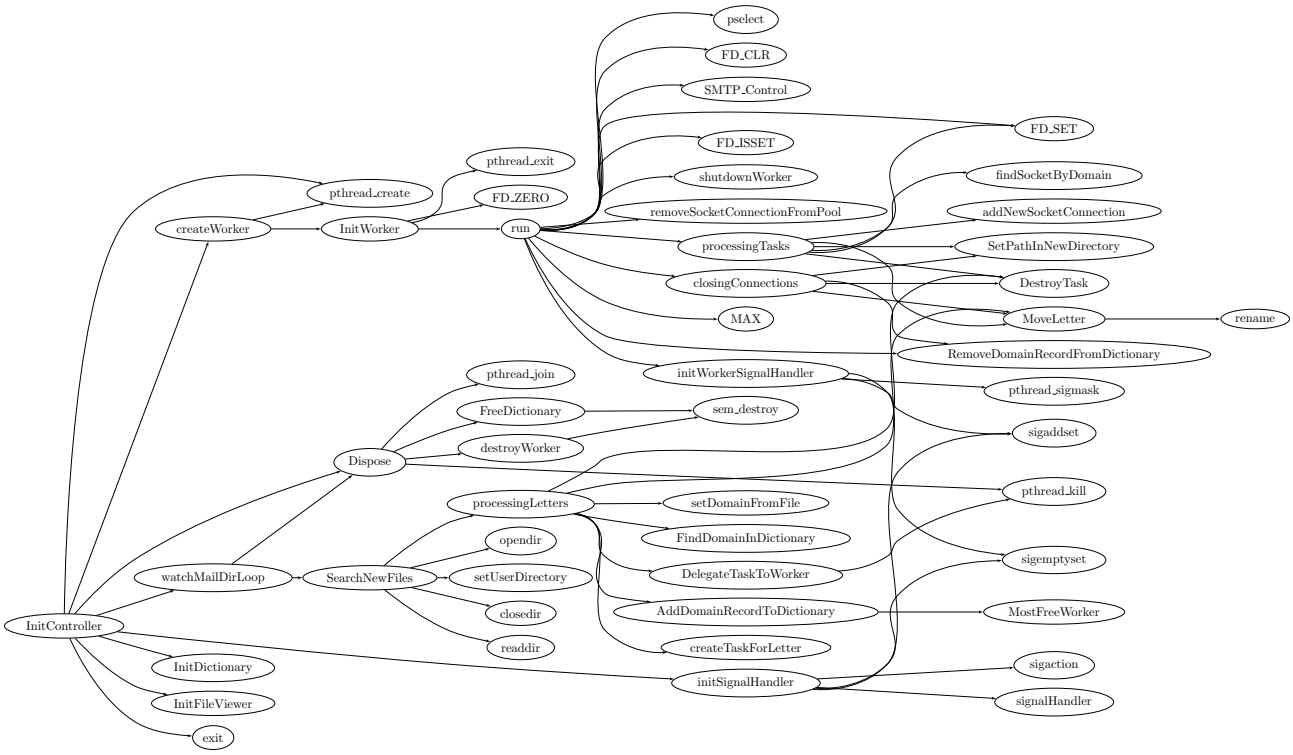

Рис. 2.4: Граф вызовов, основные функции

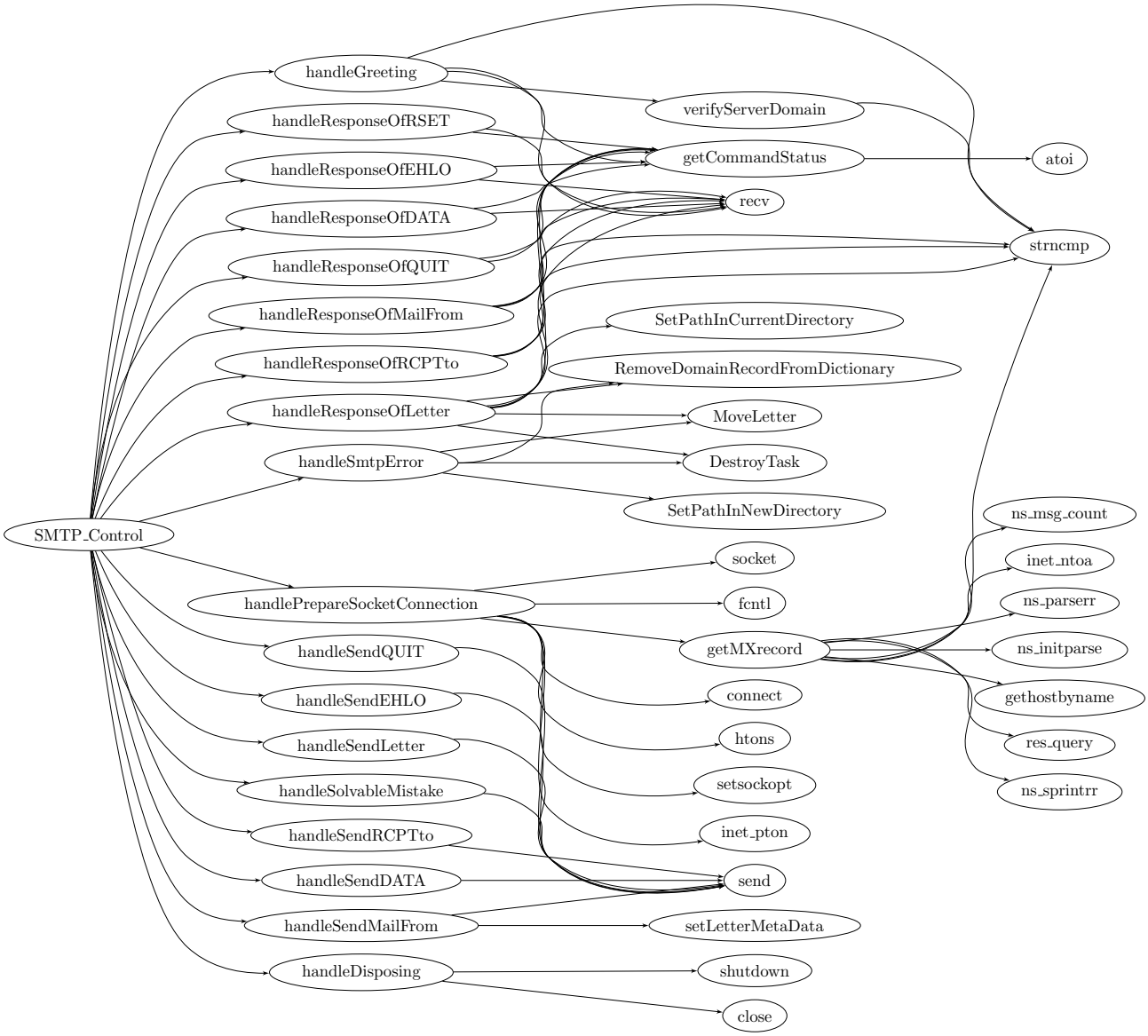

Рис. 2.5: Граф вызовов, основные функции

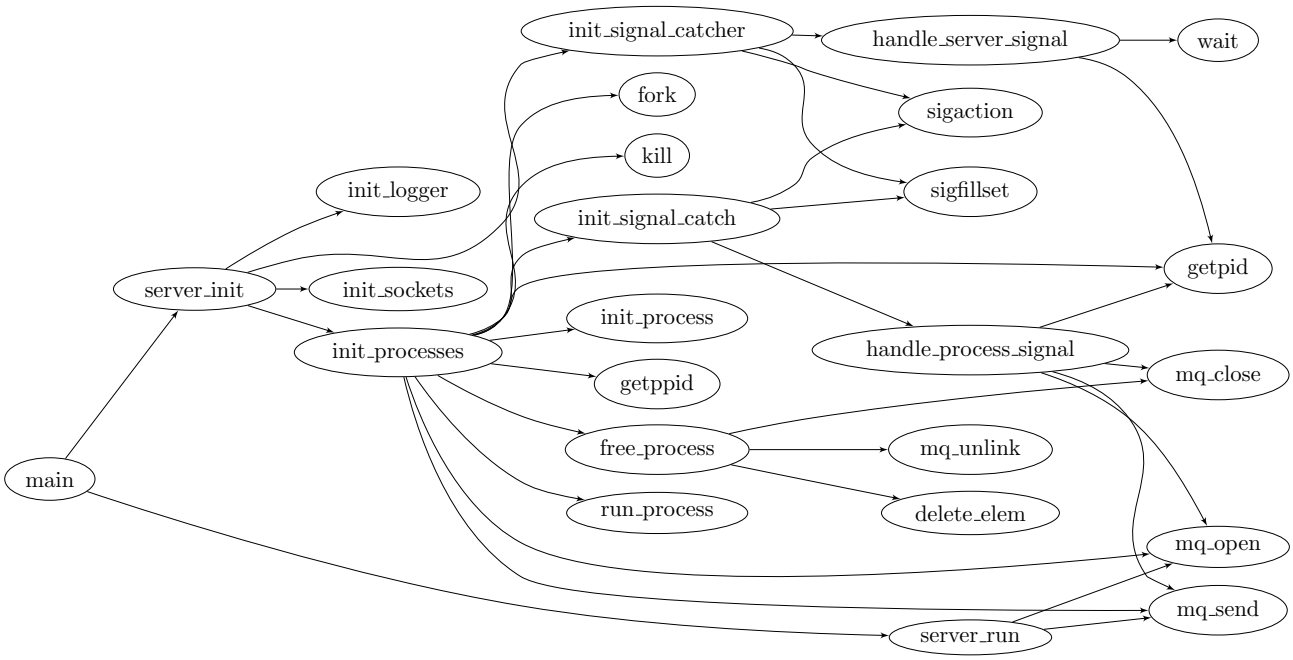

Рис. 2.6: Граф вызовов, функции обработки команд

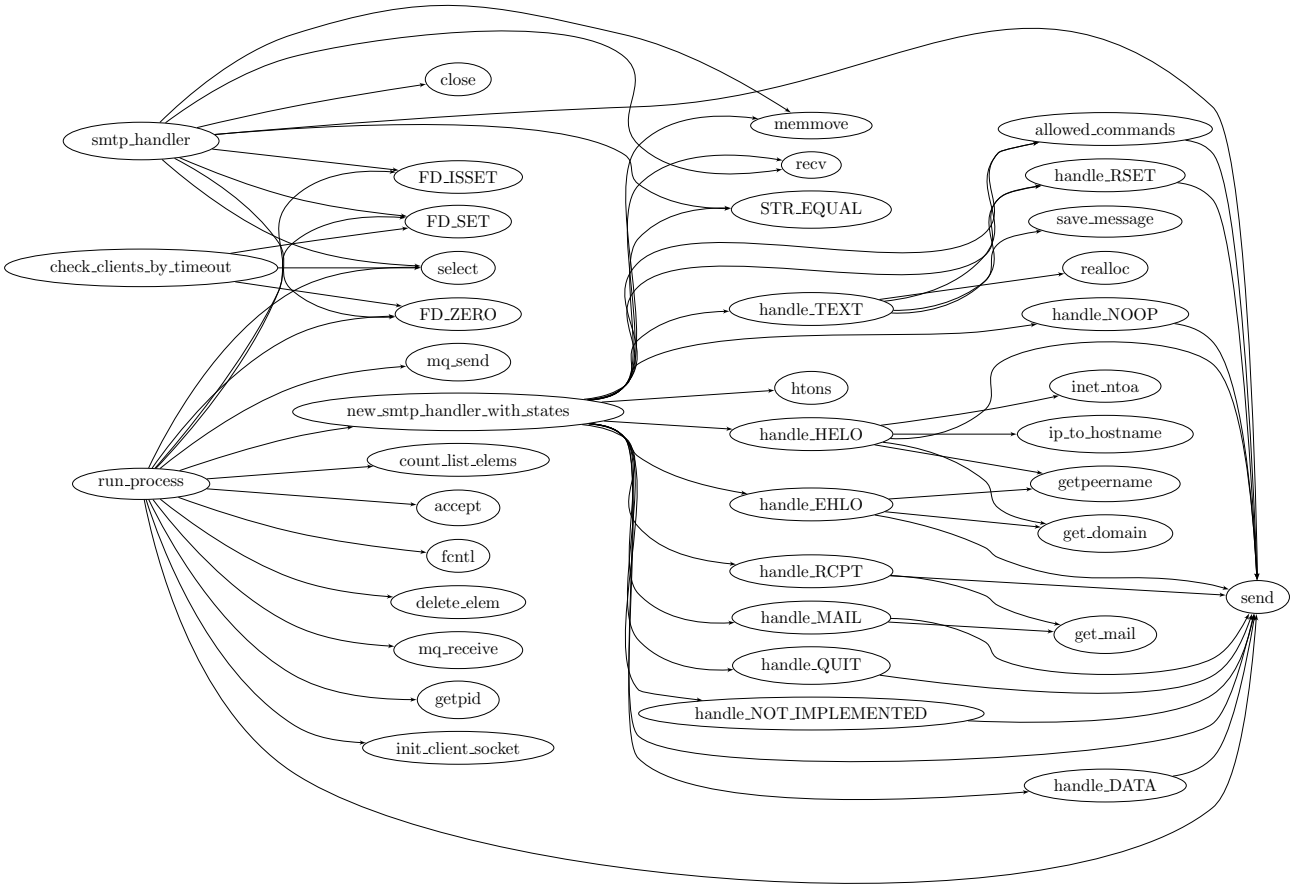

Рис. 2.7: Граф вызовов, функции обработки команд

Выводы

Что вы сделали и поняли.