Отчёт по курсовому проекту

(Федоров Павел Вячеславович)

11 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.

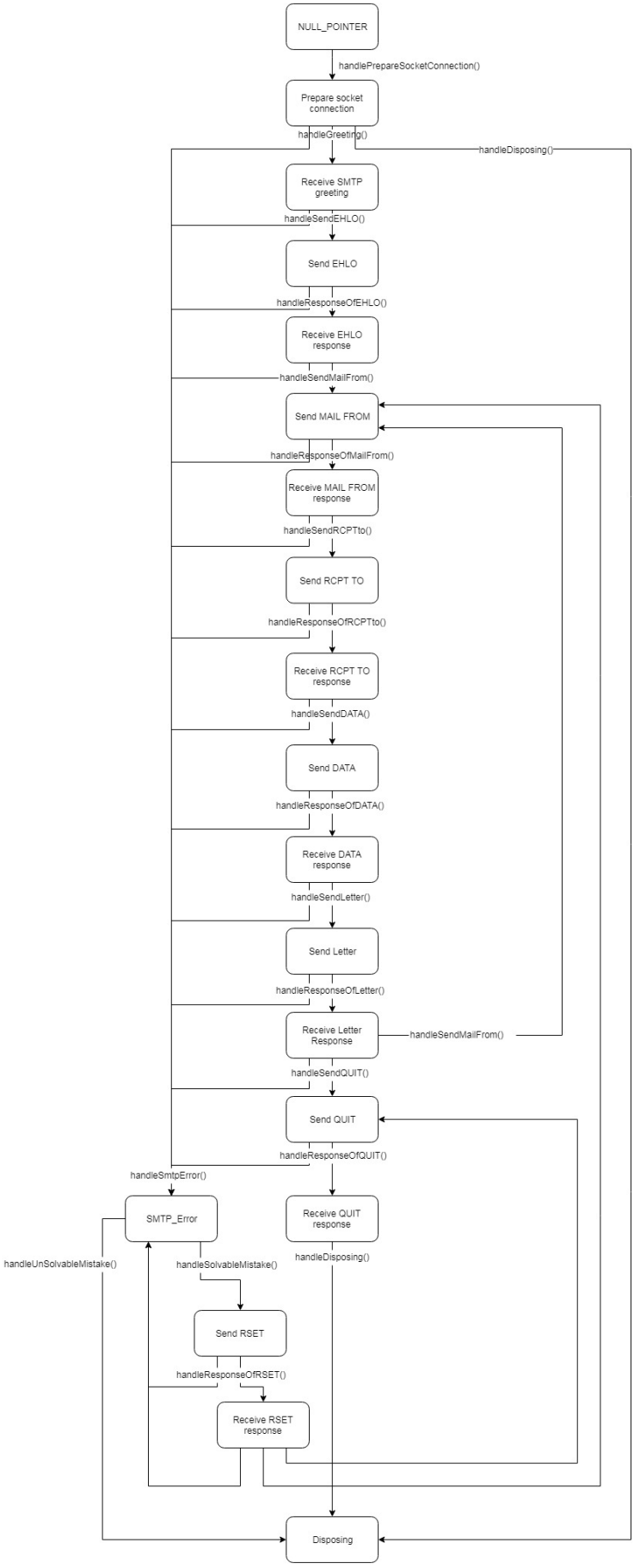

Рис. 1.1: Состояния сервера

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.

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. Когда

1.3 Синтаксис команд протокола

Команда выхода из сеанса

Команда передачи имени пользователя

Для грамматики можно использовать вставку из файла и оформление \begin{verbatim} и \end{verbatim} или пакет *listings*[1].

Для примера воспользуемся автоматической вставкой файла описания параметров программы (не забудьте перенести это в технологический раздел) через утилитку *src2tex*.

[1]На дворе XXI век, но пакет *listings* всё ещё не пашет с русскими комментариями без бубна, и лично я его пока не победил.

Глава 2

Технологический раздел

Нужно отметьть, что символ «_» необходимо оформлять как «_».

2.1 Сборка программы

Сборка программы описана в файле *Makefile* системы сборки *make*. Рис. 2.1 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*.

2.2 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*.

2.3 Графы вызова функций

Поскольку функций много, графы вызовов разбиты на два рисунка. На рис. 2.2 показаны основные функции, на рис. 2.5 – функции обработки команд. Файл **cflow.ignore** содержит список функций (точнее, шабловнов поиска), использыемых программой *grep* для удаления малоинтересных стандартных функций[1].

Графы созданы с помощью *cflow*, *cflow2dot*, *dot*.

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

Рис. 2.1: Сборка программы

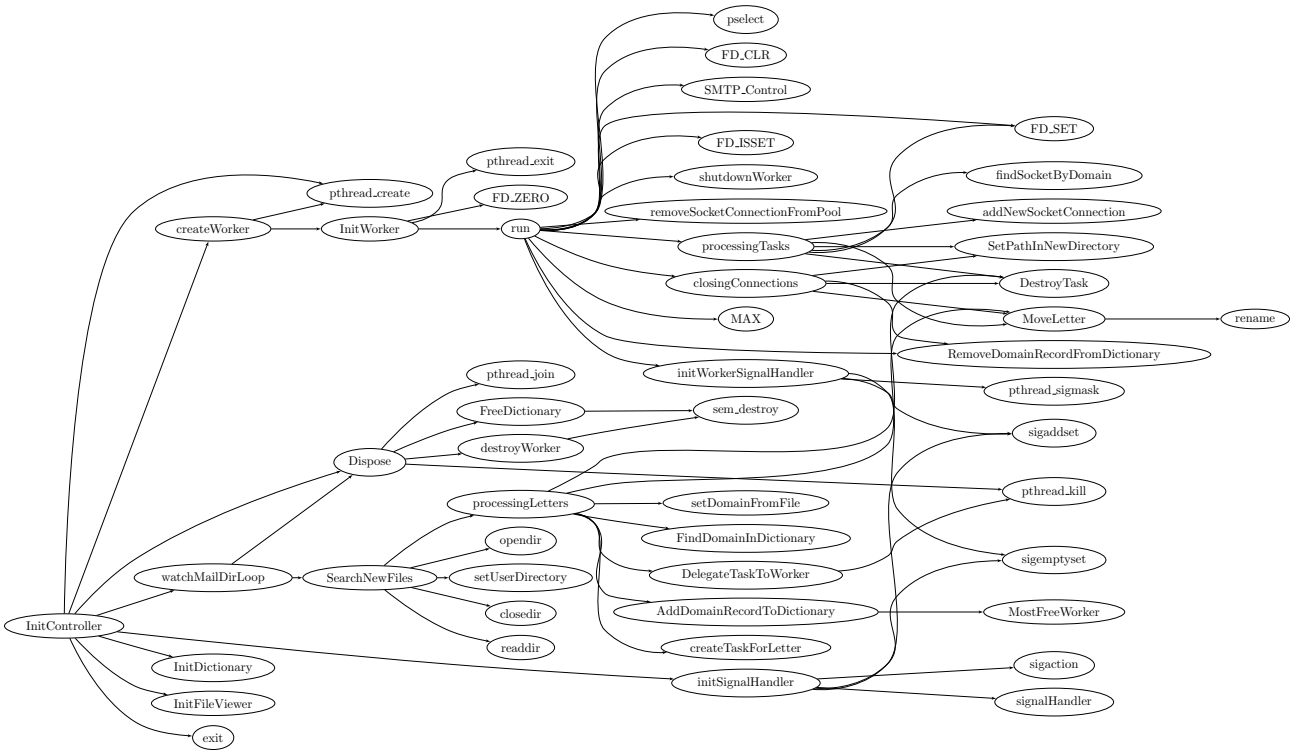

Рис. 2.2: Граф вызовов, основные функции

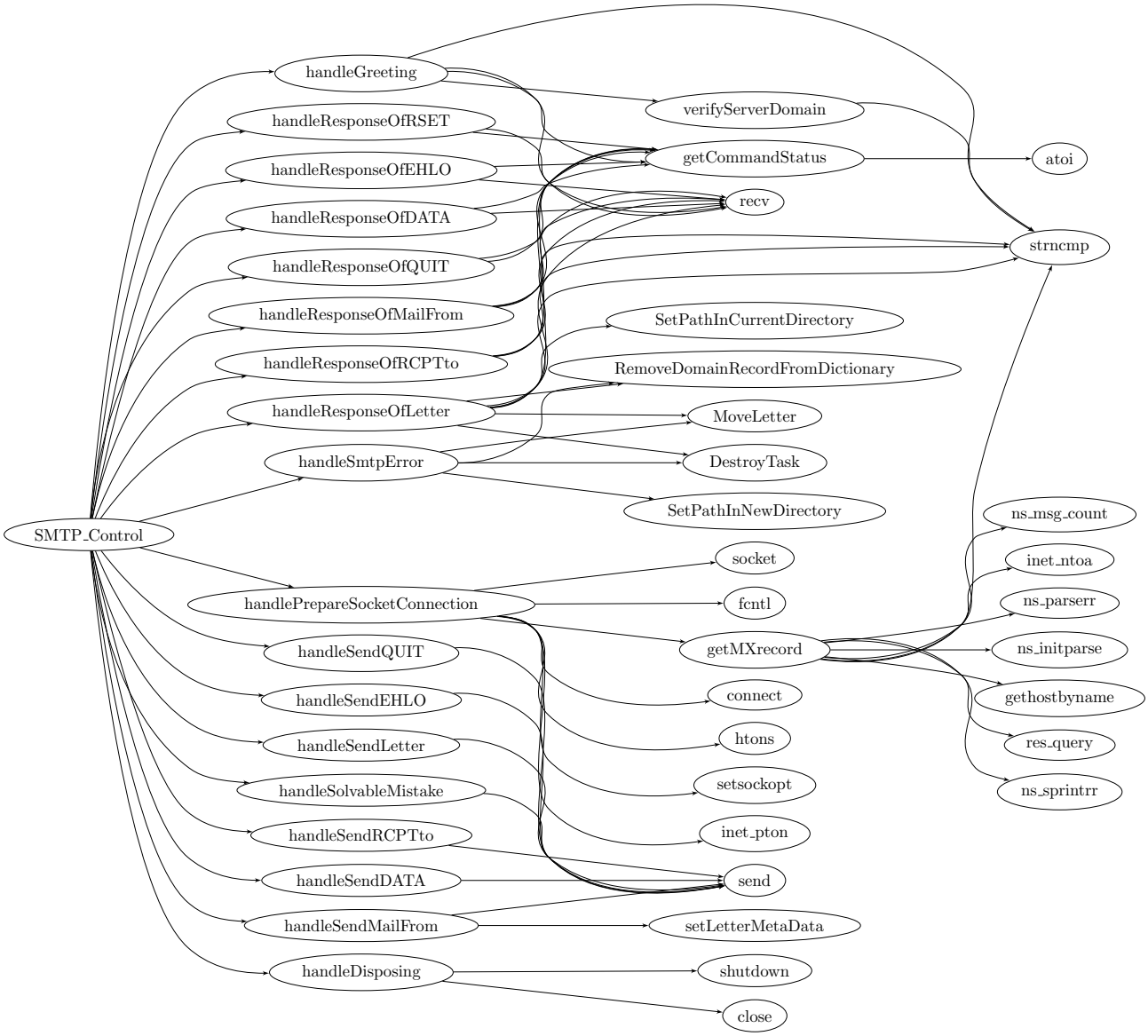

Рис. 2.3: Граф вызовов, основные функции

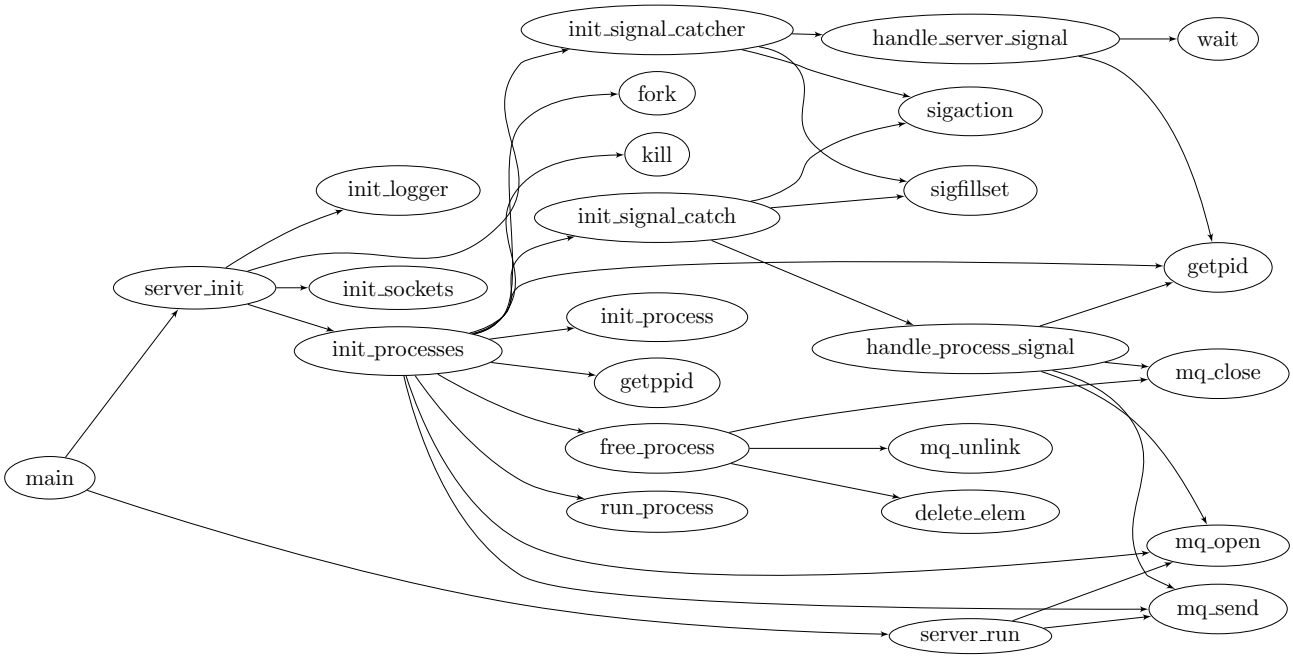

Рис. 2.4: Граф вызовов, функции обработки команд

Рис. 2.5: Граф вызовов, функции обработки команд

Выводы

Что вы сделали и поняли.