Курсовая работа

Зеленовой Юлии, ИУ7-31М

26 декабря 2020 г.

Оглавление

Введение

Сервер SMTP (Simple Mail Transfer Protocol) - это приложение, основной целью которого является отправка, получение и/или пересылка исходящей почты между отправителями и получателями электронной почты.

Должен быть реализован SMTP-сервер, обеспечивающий локальное получение сообщений. Должен использоваться вызов select и рабочие процессы. Журналирование должно вестись в отдельном потоке основного процесса.

Глава 1

Аналитический раздел

1.1 Многопроцессность

В данной работе взаимодействие с сокетами происходит через рабочие процессы. Порождение новых процессов происходит через функцию fork() в родительском процессе. Порожденные процессы выполняют работу с сокетами, а родительский процесс занимается только журналированием и очищением ресурсов, в случае закрытия процессов. Журналирование ведется в отдельном потоке родительского процесса.

К плюсам такого решения можно отнести:

– Порожденные процессы независимы друг от друга;

– Высокая скорость работы.

К недостаткам можно отнести:

– Необходимо больше ресурсов для работы;

– Выше скорость переключения между процессами, по сравнению с потоками.

1.2 Разделение нагрузки

Разделение нагрузки между процессами осуществляется следующим способом. У каждого процесса есть набор контекстов соединения, в котором хранится информация о текущих соединениях (файловый дескриптор, состояние сокета и т.д.). Каждый процесс самостоятельно определяет необходимость чтения или записи, выполняет операции обработки данных. Процесс работает только с теми сокетами, которые выполняют общение в данный момент.

Выбор сокета, с которым происходит соединение, отдается на откуп ядру.

1.3 Сущности предметной области

В качестве письма считается текстовый файл. В файле содержится такая информация как: адрес отправителя, адреса получателей и содержимое письма. Сервер получает содержимое файла во время общения с клиентом и хранит их у себя в папке, предназначенной для хранения полученных писем.

Клиент. Данная сущность является инициатором общения. Во время общения клиент отправляет содержимое письма серверу для дальнейшей передачи получателю, указанному в письме. Команды SMTP генерируются отправителем и посылаются получателю, который на них отвечает.

Сервер. Является получателем почтового сообщения и отвечает на запросы клиента. Хранит в себе все письма, предназаначенные для пользователей, использующих его домен в своем адресе. Сервер может быть либо конечным адресатом, либо промежуточным звеном в цепи передачи письма.

Глава 2

Конструкторский раздел

2.1 Конечный автомат состояний сервера

На рисунке ?? представлен конечный автомат состояний сервера.

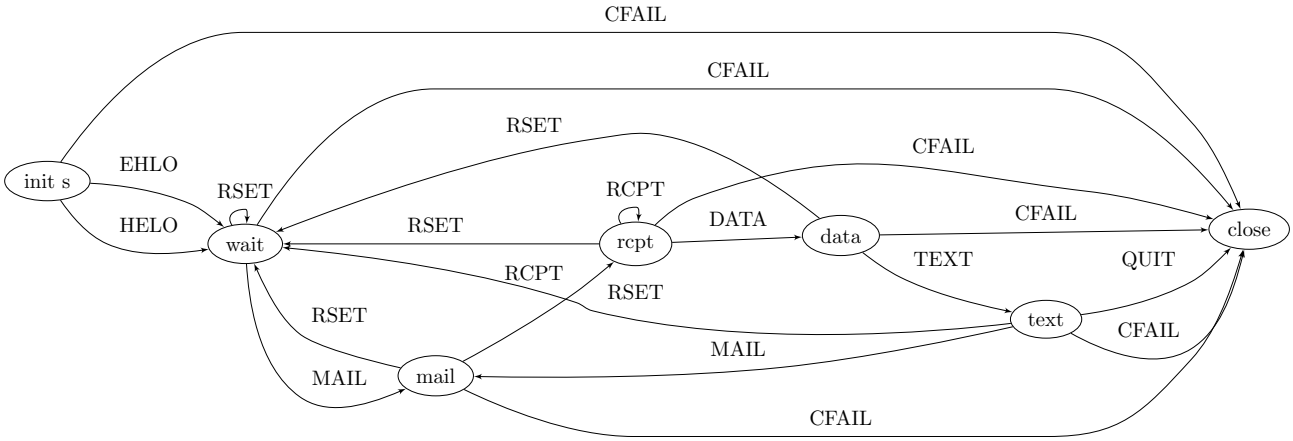

Рис. 2.1: Состояния сервера

При установке соединения и после отрпавки приветственного сообщения, сервер переходит в состояние *init s*. Находясь в данном сотоянии сервер ожидает только команды приветствия от клиента.

Почти все переходы между состояниями происходят после обработки сообщения от клиента. Как видно, в случае возникновения ошибки при переходе, происходит событие, приводящее к разрыву соединения. Это может случиться на любом этапе работы.

Чтобы сервер мог отправить несколько сообщений за одну сессию, то после ввода текста сообщения он может ввести нового отправителя, что означет начало передачи нового письма.

2.2 Структуры данных

На рисунке ?? представлена ER-диаграмма основных структур данных программы.
Структура *server_t* отвечает за хранение информации о сервере:

– *workers* хранит информацию о дочерних процессах;

– *nworkers* – количество запущенных дочерних процессов;

– *master_ socket* содержит файловый дескриптор мастера-сокета;

– *l_ context* содержит иформацию, необходимую дляя логгирования.

Структура *process_ t* хранит в себе такую информацию о процессах, как:

– *client_ socket* – информация о запущенных сокетах;

– *readfds* – файловый дескриптор на чтение из сокета;

– *writefds* – файловый дескритпор на запись информации в сокет;

– *pid* – идентификатор текущего процесса;

– *l_ context* содержит иформацию, необходимую дляя логгирования;

– *running* – переменная для остлеживания состояния работы процесса.

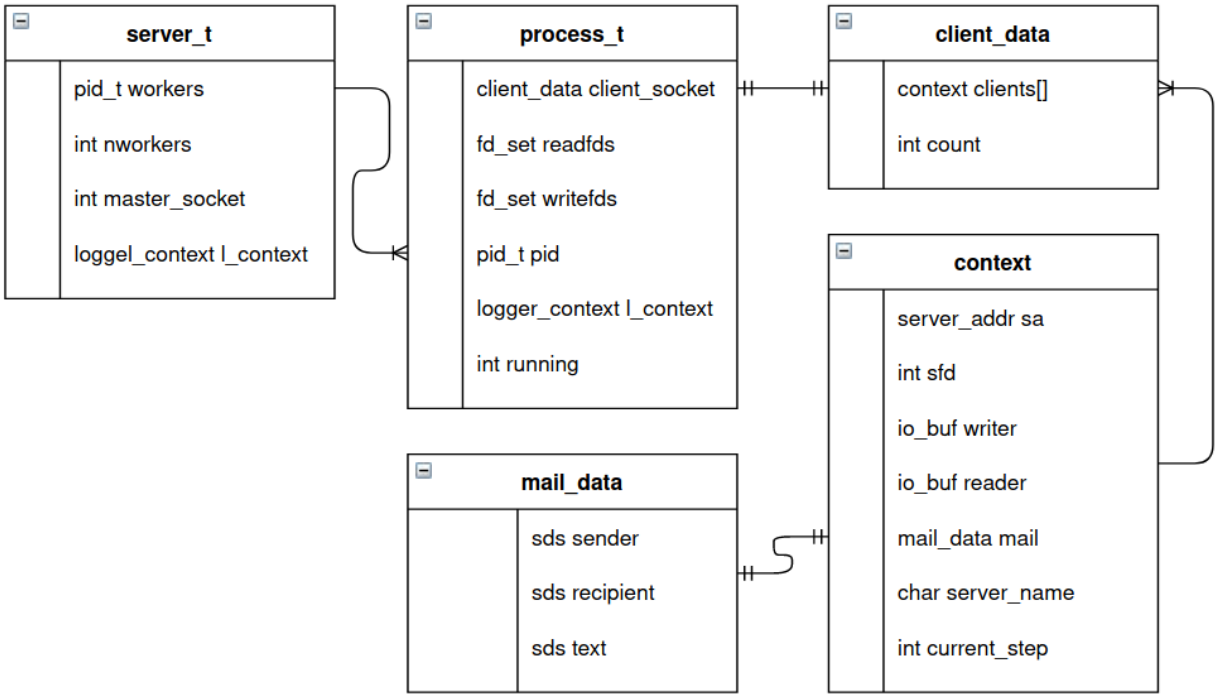

Рис. 2.2: ER-диаграмма основных структур данных

Структура *client_ data* хранит в себе информацию о соединениях:

– *clients* – список установленных соединений;

– *count* – количество соединений.

Структура *context* хранит в себе информацию о соединении:

– *sa* – адрес подключенного клиента;

– *sfd* – файловый дескриптор сокета;

– *writer* – структура-буфер для записи в сокет;

– *reader* – структура-буфер для чтения из сокета;

– *mail* – структура, хранящая в себе данные о получаемом письме;

– *server_name* – доменное имя сервера;

– *current_step* – текущий шаг автомата.

Структура *mail_data* хранит в себе информацию о получаемом письме:

– *sender* – строка с адресом отправителя;

– *recipient* – строка с адресами получателей;

– *text* – текст сообщения.

2.3 Синтаксис команд протокола

– HELO client_name. Команда для начала сеанса SMTP между отправителем и сервером, а также чтобы сервер мог идентифицировать отправителя. Синтаксис этой команды требует, чтобы отправитель отправил свое доменное имя в качестве параметра.

– EHLO client_name. Работает по тому же принципу, что и HELO, только сервер возвращает список команд, которые он поддерживает.

– MAIL FROM path. Команда используется чтобы начать процесс составления электронного письма и позволить серверу знать идентификатор отправителя, и его синтаксис требует обязательного использования адреса электронной почты отправителя в качестве параметра.

– RCPT TO client_path. Синтаксис этой команды требует, чтобы адрес электронной почты получателя был предоставлен в качестве обязательного параметра, в дополнение к другим необязательным. Он используется для указания получателя сообщения электронной почты.

– DATA. Эта команда уведомляет сервер о том, что он готов начать отправку электронного письма.

– QUIT. Эта команда используется для завершения сеанса SMTP.

– RSET. Эта команда используется для выполнения операции сброса; Текущий разговор завершается, или сообщение сбрасывается, и можно начинать заново.

2.4 Проектирование алгоритмов обработки соединений

Псевдокод алгоритма обработки соединений в одном потоке выполнения приведен ниже.

```
While процесс работает
    max_socket_desc = max(client_sockets->sfd)

    activity = select()
    if activity < 0
        return Сообщение об ошибке
    if activity == 0
        Сообщение о таймауте
        client_socket = ctx_new()

    if FD_ISSET(master_socket, readfds)
        if connfd < 0
            return Сообщение об ошибке
        Создание нового соединения

    for i in client_sockets
        Проверить состояние соединения
        Обработать чтение/запись с сокетом
```

Глава 3

Технологический раздел

3.1 Сборка программы

Сборка программы осуществляется с помощью gcc версии 9.3 и библиотек:

– Glib – библиотека, содержащая коллекцию различных структур данных;

– pthread – библиотека для управления потоками.

Для автоматизации и упрощения сборки использовалась утилита make.

3.2 Основные функции программы

– create_processes. Создает дочерние процессы. В качестве аргументов передаются данные сервера (такие как логер и количество процессов, которые необходимо создать), адрес сервера и структура с настройками запуска.

– run_process. Отввечает за непосредственное выполнение работы процесса. Здесь происходит подлючение новых клиентов и чтение/запись из сокета. В качестве аргументов принимает данныые процесса, файловый дескриптор главного сокета, адрес сервера и его имя.

– key_switcher. На основе входных данных определяет событие, которое должно произойти на сервере. В соответствии с событием будет вызвана соотвтествующая ему функция.

– <SMTP_keyword>_handler. Данная группа функций работает согласно тому, какая команда поступает на вход функции, описанной выше. Например, функция RCPT_handler будет запоминать пользователя, которому необходимо отправить письмо. Это происходит если пришла строка "RCPT TO <user>" доменное имя получателя совпадает с именем сервера и все шаги до этого были выполнены корректно, т.е. текущий статус сервера совпадаает с тем, который необходим для выполнения данной функции.

3.3 Файл конфигурации и параметры командной строки

В качестве файла конфигурации используется файл формата *.ini. Его пример представлен ниже.

```
[multiprocessing]
count=4

[logging]
level=0
file=tmp/logs

[server]
name=pupa.com
port=25
```

3.4 Тестирование

Для тестирования работы сервера использовадся фреймворк CUnit, с помощью которого проверялась работа отдельных функций и работы протокола.

3.5 Графы вызова функций

На рисунках ниже представлены графы вызова функций. Графы созданы с помощью *pycflow2dot*.

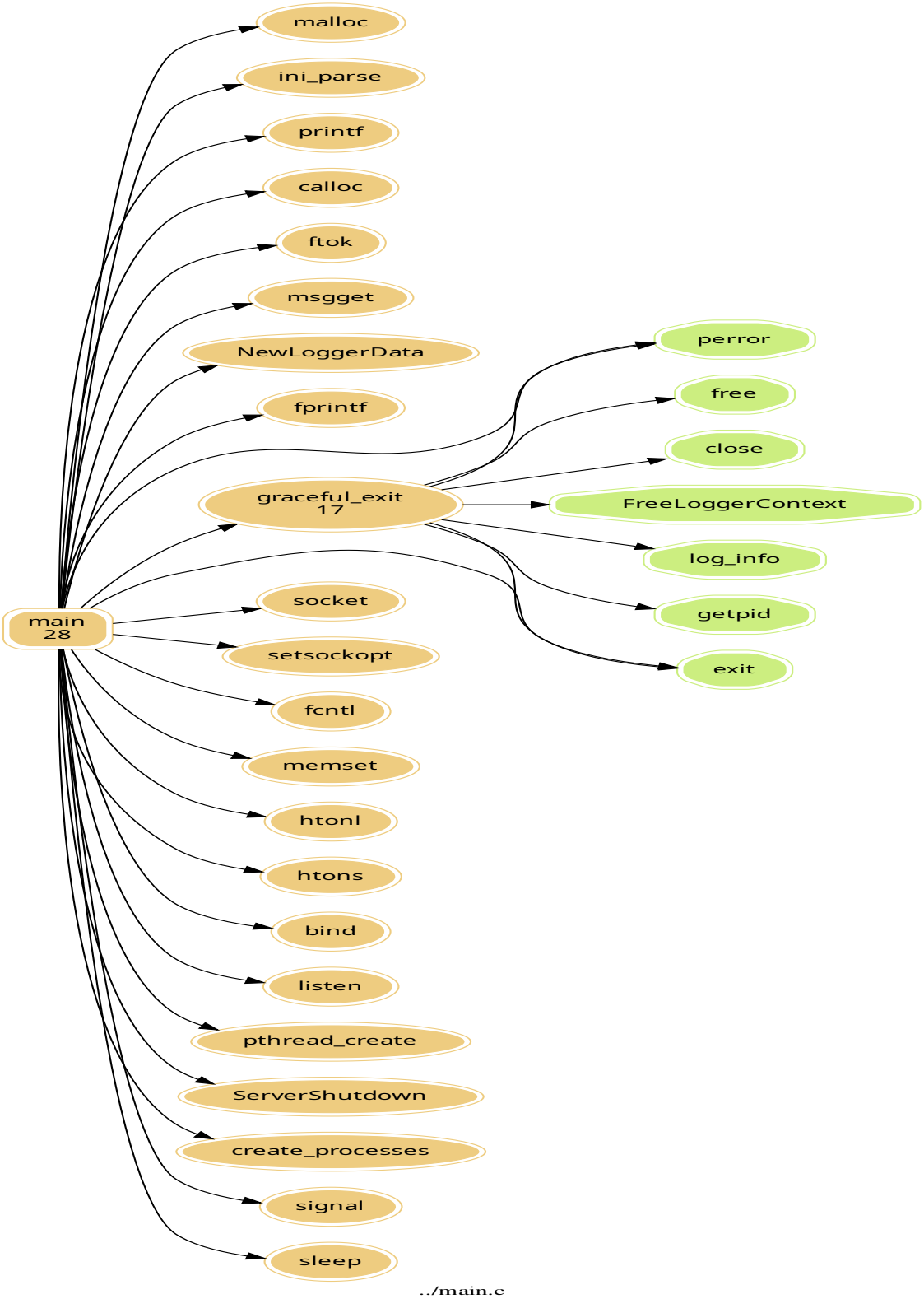

Рис. 3.1: Граф вызова функций main.c

Рис. 3.2: Граф вызова функций socket.c

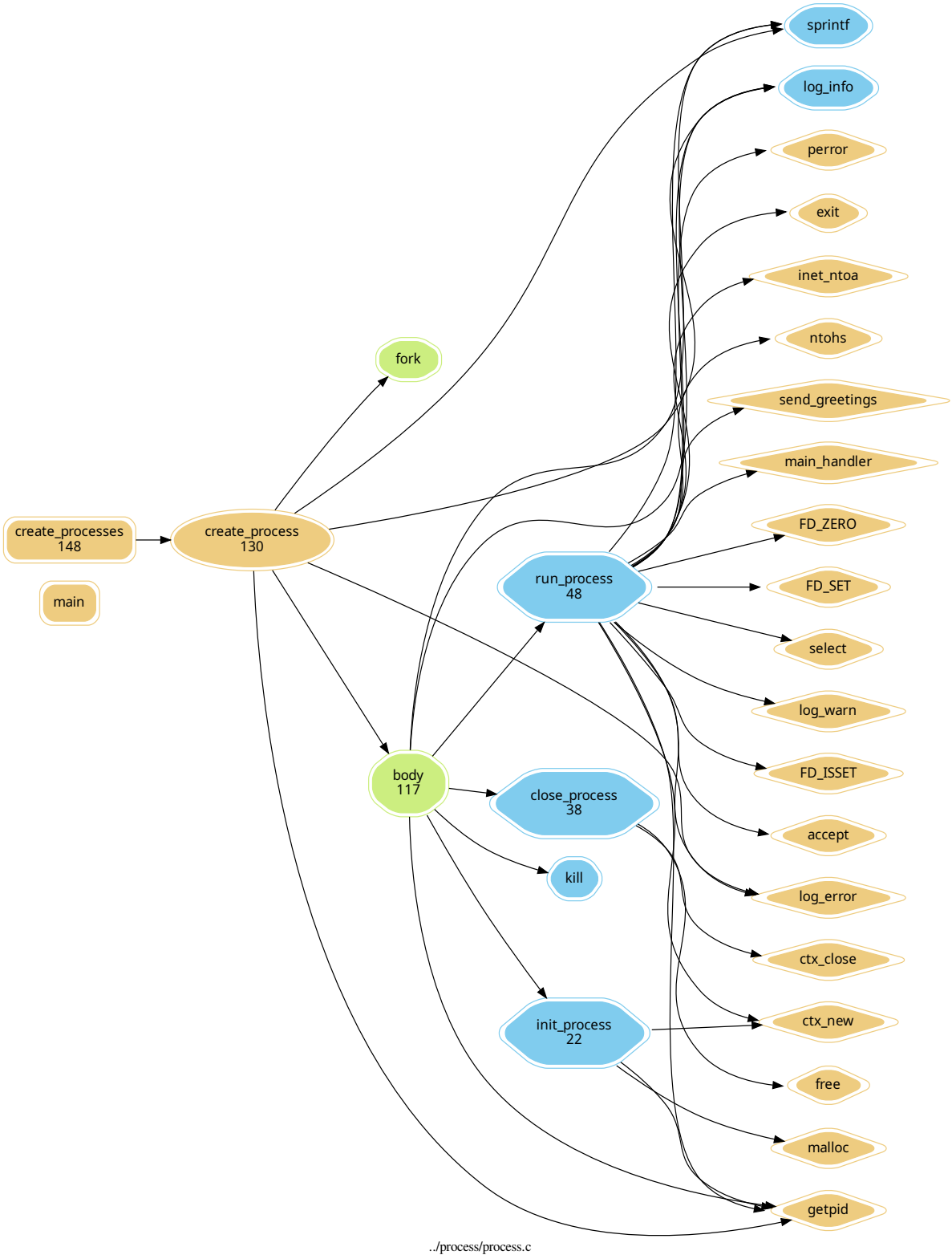

Рис. 3.3: Граф вызова функций process.c

Заключение

В ходе курсовой работы по ПВС быыл разработан сервер для принятия почты, работающий по протоколу SMTP. В аналитическом разделе были описаны: использование мультипроцессинга и сущности предметной области. В конструкторской части были описаны конечный автомат состояний сервера, используемые структуры данных и алгоритм обработки соединений. В технологическом разделе были описаны основные функции программы, граф вызова функции и организация тестирования.