Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ *Робототехники и комплексной автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

по дисциплине: «Разработка программных систем»

Студент	Израелян Ева Арамовна
Группа	РК6-64Б
Тип задания	Лабораторная работа №3
Тема лабораторной работы	Сетевое программирование

Студент _____ **Израелян Е.А.**
подпись, дата *фамилия, и.о.*

Преподаватель _____ **Федорук В.Г.**
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 2022 г.

Оглавление

Задание на лабораторную работу

Разработать программу-"получатель" электрической почты по протоколу POP-3 из почтового ящика на POP3-сервере. Программа должна обеспечивать минимальную функциональность UNIX-утилиты mail (без аргументов). Описание протокола POP-3 можно найти в Internet (например, www.citforum.ru), другой путь - использование сканеров сети типа tcpdump или snoop.

Описание протокола

Протокол POP3 (Post Office Protocol) предназначен для получения сообщений электронной почты, находящихся на удалённом сервере электронной почты. POP3-сервер прослушивает TCP-порт 110. Команды POP3 состоят из ключевых слов (3-4 символа), за которыми могут следовать аргументы. Каждая команда завершается парой символов CRLF. Как ключевые слова, так и аргументы могут содержать только печатаемые ASCII-символы. В качестве разделителя используются символы пробела. Каждый аргумент может содержать до 40 символов.

Сигнал отклика в POP3 содержит индикатор состояния: положительный - "+OK" и отрицательный "-ERR" (все символы прописные). За откликом может следовать дополнительная информация. Многострочные отклики завершаются последовательностью CRLF.CRLF.

В начале работы клиент должен представиться – передать команды «USER [Логин]» и «PASS [Пароль]». Затем клиент имеет возможности отправлять команды на сервер и получать ответы. Команды, которые используются в лабораторной работе:

STAT – сервер возвращает количество сообщений в ящике и их длину.

RETR [n] – сервер возвращает письмо с номером n.

Чтобы завершить работу с POP3-сервером, клиент должен отправить команду QUIT.

Описание структуры программы

Необходимые данные для подключения к POP3 серверу хранятся в конфигурационном файле, название которого передаётся первым аргументом командной строки. Конфигурационный файл состоит из трёх строк – POP3 сервер, логин и пароль. Например,

rk6lab.bmstu.ru

rk6stud

rk6stud

В начале работы программа настраивает TCP сокет, привязывает к порту клиента, заданному произвольно, и подключается к серверу с адресом, переданным из файла, по порту 110. Затем происходит авторизация с помощью команд USER и PASS с проверкой правильности введённых данных путём анализа ответа сервера. Если первый символ ответа – «-», то авторизация неудачна, о чём программа сообщает пользователю. После успешной авторизации программа проверяет количество писем в ящике командой STAT, и начинается цикл вывода писем. Если количество писем больше, чем MAXMES (максимальное количество писем, заданное директивой препроцессора), то выводятся первые MAXMES писем. Получение писем происходит путём отправки серверу команды RETR. В цикле выполняется получение части ответа из сокета и запись его в buf, затем вывод содержимого buf пользователю, пока не будет достигнута последовательность «CRLF.CRLF». По окончании работы программы на сервер должна быть отправлена команда «QUIT», а сокет обязательно должен быть закрыт командами shutdown и close.

Основные используемые структуры данных

- Массив строк char* errors[32] хранит текст ошибок подключения к серверу согласно их номерам;
- Переменная char buf[BUFSIZE] представляет из себя буфер для чтения данных из сокетов;
- Переменные struct sockaddr_in csin, ssin хранят структуры интернет-адресов и параметров связи клиента и сервера соответственно;
- Переменная struct hostent* hp содержит структуру для разрешения доменного имени узла в его адрес;
- Структура FILE* fd используется для чтения информации из конфигурационного файла;

- Строки char servn[32], usern[32], passn[32] хранят доменное имя почтового сервера, логин и пароль;
- Переменная int sock хранит дескриптор используемого сокета.

Блок-схема программы

На рисунке 1 представлены блок-схемы функции main, функции инициализации почтового клиента и функции приёма сообщений.

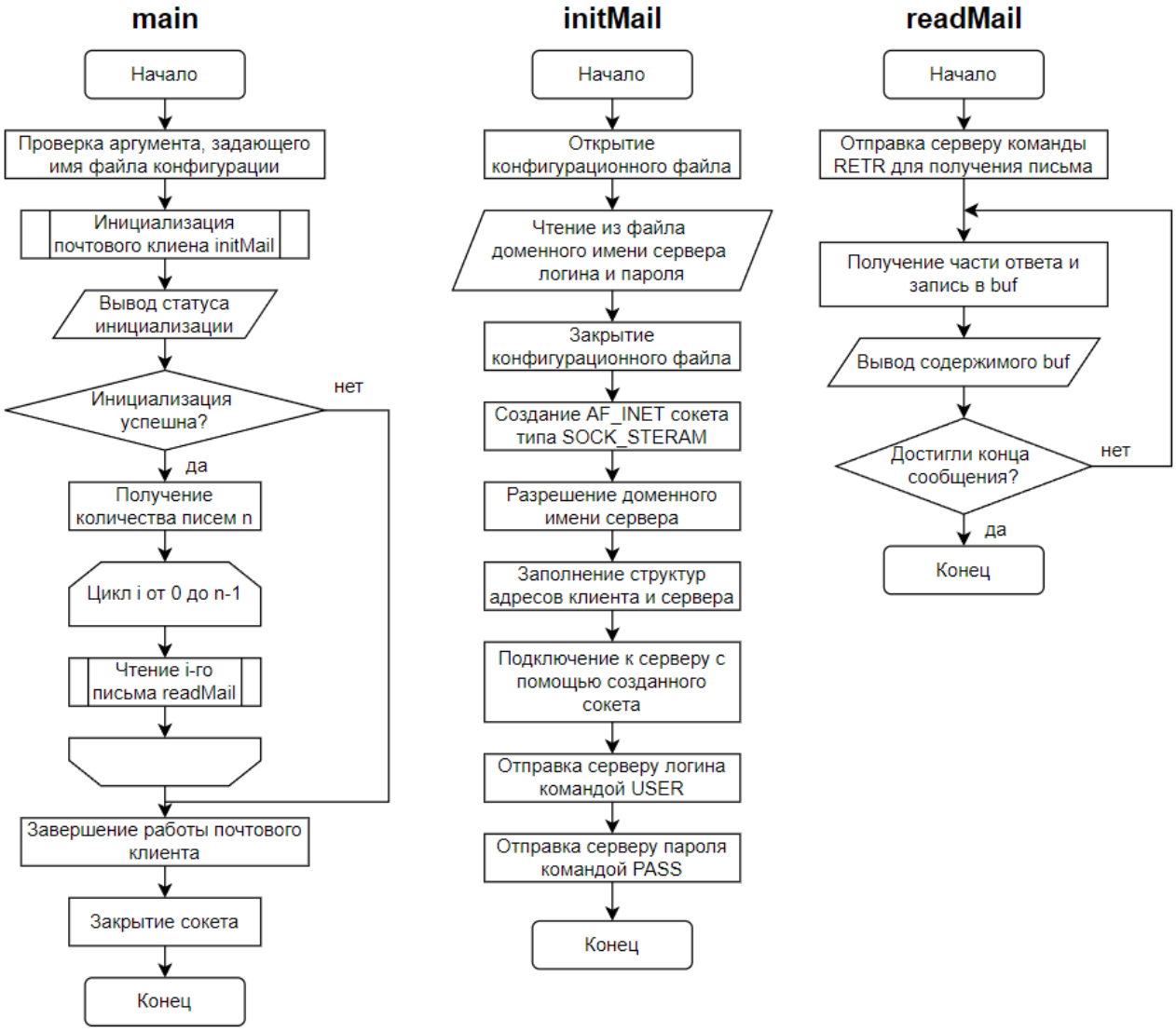

Рисунок 1. Блок-схемы главных функций программы

Пример работы программы

Результат работы программы представлен на рисунке 2.

```
eva@eva-VirtualBox:~$ ./a.out config.cfg
Connection is successful


Message №1
---------------||---------------
+OK 2188 octets
Return-Path: <israelyanea@student.bmstu.ru>
Delivered-To: evaislab3@rambler.ru
Received: from 1vm0010.prod.mail.rambler.tech ([10.99.0.12])
        by 2vm0017.prod.mail.rambler.tech with LMTP
        id gBGTCFT1Z2IAbAAARIqM+w
        (envelope-from <israelyanea@student.bmstu.ru>)
        for <evaislab3@rambler.ru>; Tue, 26 Apr 2022 16:36:20 +0300
Received: from mx9.mail.rambler.ru ([10.99.0.12])
        by 1vm0010.prod.mail.rambler.tech with LMTP
        id MJQzCFT1Z2ItFwAA2vrefQ
        (envelope-from <israelyanea@student.bmstu.ru>)
        for <evaislab3@rambler.ru>; Tue, 26 Apr 2022 16:36:20 +0300
Received: from mailhub.bmstu.ru (mailhub.bmstu.ru [195.19.32.15])
        by mx9.mail.rambler.ru (Postfix) with ESMTP id 3BE69C40334
        for <evaislab3@rambler.ru>; Tue, 26 Apr 2022 16:36:19 +0300 (MSK)
Received: from mailhub.bmstu.ru (mailhub.bmstu.ru [195.19.32.15])
        by 2vm0048.prod.mail.rambler.tech (resmtp/Rambler) with ESMTP id aJ1Cwc6o;
        Tue, 26 Apr 2022 13:36:19 +0000
Received: from localhost (localhost [127.0.0.1])
        by mailhub.bmstu.ru (Postfix) with ESMTP id 2900E8DA3D
        for <evaislab3@rambler.ru>; Tue, 26 Apr 2022 16:36:19 +0300 (MSK)
X-Virus-Scanned: amavisd-new at mailhub.bmstu.ru
Received: from mailhub.bmstu.ru ([127.0.0.1])
        by localhost (mailhub.bmstu.ru [127.0.0.1]) (amavisd-new, port 10024)
        with ESMTP id 5r6ZLn1zPxkm for <evaislab3@rambler.ru>;
        Tue, 26 Apr 2022 16:36:19 +0300 (MSK)
Received: from student.bmstu.ru (mailstudent.bmstu.ru [195.19.32.120])
        by mailhub.bmstu.ru (Postfix) with ESMTP id 167058D9D8
        for <evaislab3@rambler.ru>; Tue, 26 Apr 2022 16:36:19 +0300 (MSK)
Received: from [46.31.27.48] (account iea19r115@student.bmstu.ru)
  by student.bmstu.ru (CommuniGate Pro XIMSS 6.3.11d)
  with HTTPU id 5893291 for evaislab3@rambler.ru; Tue, 26 Apr 2022 16:36:17 +0300
X-Mailer: Samoware HTML5 6.3.6472776
Subject: lab
MIME-Version: 1.0
From:  =?utf-8?B?0JjRgdGA0LDQtdC70Y/QvSDQldCy0LAg0JDRgNCw0LzQvtCy0L0=?=
 =?utf-8?B?0LA=?= <israelyanea@student.bmstu.ru>
To:  evaislab3@rambler.ru
Date: Tue, 26 Apr 2022 16:36:17 +0300
Message-ID: <ximss-5893293@student.bmstu.ru>
Content-Type: text/plain; charset="utf-8"; format="flowed"
Content-Transfer-Encoding: 8bit

lab3
.
```

Рисунок 2. Пример вывода сообщения

Код программы

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <memory.h>
#include <netdb.h>
#include <string.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

#define BUFSIZE 1024 // размер буфера
#define MAXMES 4 // Максимальное количество отображаемых сообщений

char* errors[32] = // Статусы подключения
{
	"Connection is successful	\n",
	"Unable to create socket	\n",
	"Unable to connect to POP server\n",
	"Authentication failed		\n"
};

// Функция чтения письма с номером i из сокета sock
void readMail(int sock, int i)
{
	int n;
	char buf[BUFSIZE];
	int f;

	sprintf(buf, "RETR %d\n", i); // Запрос к серверу дать письмо
	write(sock, buf, strlen(buf));

	printf("\n\nMessage №%d\n----------------||----------------\n", i); fflush(stdout);

	do {
		n = recv(sock, buf, BUFSIZE, 0); // Запись части сообщения в buf
		f = memcmp("\r\n.\r\n", buf + n - 5, 5); // Ожидание конца сообщения
		write(1, buf, n); // Вывод содержимого buf
	} while (f); // Приём завершается по CRLF.CRLF

	write(1, "\n----------------||----------------\n\n", 38);
}

int initMail(int* sock, char* confFile)
{
	struct sockaddr_in csin, ssin;
	struct hostent* hp;
	int n;
	char buf[BUFSIZE];
	FILE* fd;
	char servn[32], usern[32], passn[32]; // Имя сервера, логин и пароль

	fd = fopen(confFile, "r"); // Файл с именем сервера, логином и паролем
	fscanf(fd, "%s\n%s\n%s", servn, usern, passn);
	fclose(fd);

	// Инициализация TCP интернет-сокета
	if ((*sock = socket(AF_INET, SOCK_STREAM, 0)) == -1)
		return 1;

	// Заполнение структуры ssin
	memset((char*)&ssin, '\0', sizeof(ssin));
	ssin.sin_family = AF_INET;
	hp = gethostbyname(servn); // Разрешение доменного имени
	memcpy((char*)&ssin.sin_addr, hp->h_addr, hp->h_length);
	ssin.sin_port = htons(110); // 110ый номер порта преобразуется в необходимый формат
```

```
	// Заполнение структуры csin
	memset((char*)&csin, '\0', sizeof(csin));
	csin.sin_family = AF_INET;
	csin.sin_port = htons(12524);  // Произвольный порт

	// Привязка csin к сокету
	bind(*sock, (struct sockaddr*)&csin, sizeof(csin));

	// Подключение к серверу по сокету
	if (connect(*sock, (struct sockaddr*)&ssin, sizeof(ssin)) == -1)
		return 2;

	buf[0] = 0;
	n = read(*sock, buf, BUFSIZE); // Ожидание ответа от сервера

	// Отправка логина
	sprintf(buf, "USER %s\n", usern);
	write(*sock, buf, strlen(buf));
	n = read(*sock, buf, BUFSIZE);
	if (buf[0] == '-') return 3; // Неудача

	// Отправка пароля
	sprintf(buf, "PASS %s\n", passn);
	write(*sock, buf, strlen(buf));
	n = read(*sock, buf, BUFSIZE);
	if (buf[0] == '-') return 3;

	return 0;
}

// Получение количества сообщений в ящике
int getCntM(int sock)
{
	int n;
	char buf[BUFSIZE];

	write(sock, "STAT\n", 5); // Команда для получения кол-ва сообщений
	n = read(sock, buf, BUFSIZE);
	sscanf(buf + 3, "%d", &n);
	return n;
}

int main(int argc, char** argv) {
	char buf[BUFSIZE];
	int n;
	int sock; // Дескриптор сокета


	if (argc < 2) { // Если не передано имя файла в аргументах программы
		write(1, "Missing argument 'config_file_name.cfg'\n", 40);
		return -1;
	}

	n = initMail(&sock, argv[1]);
	write(1, errors[n], 32); // Вывод статуса подключения
	if (!n) { // Если было успешное подключение
		n = getCntM(sock);
		for (int i = 0; i < n && i < MAXMES; ++i) // Чтение n писем не больше MAXMES
			readMail(sock, i + 1);
	}
	write(sock, "QUIT\n", 5); // Сообщение серверу о выходе
	shutdown(sock, 2);  // Обязательное отключение и закрытие сокета
	close(sock);
}
```