Федеральное государственное автономное

образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| институт |
| Информатика |
| кафедра |

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №5**

|  |
| --- |
| Взаимодействие процессов в OC GNU/Linux |
| Тема |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Преподаватель | |  |  |  | А. Н. Пупков |
|  | |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ21-16/1б, 032155832 |  |  |  | Н. А. Терентьев |
|  | номер группы, зачётной книжки |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2023

**СОДЕРЖАНИЕ**

1 Цели 3

2 Задачи 3

3 Описание варианта задания 3

4 Ход выполнения 4

4.1 Клиент 4

4.2 Сервер 6

4.3 Doxygen документация 6

5 Выводы 7

Список использованных источников 8

Приложение А 9

**1 Цель**

Изучение особенностей межпроцессного взаимодействия в OC GNU/Linux.

**2 Задачи**

Для выполнения практической работы необходимо выполнить  
следующие задачи:

* ознакомиться с краткими теоретическими сведениями по организации межпроцессного взаимодействия в ОС GNU/Linux;
* получить у преподавателя собственный вариант задания, который предусматривает разработку серверной и клиентской частей приложения, взаимодействующих посредством механизма Internet-сокетов и сетевых протоколов. Использование высокоуровневых средств является ошибочным.При выполнении заданий с нечетным вариантом должны использоваться потоковые сокеты, с четным вариантом — дейтаграммные. Обеспечить сборку обеих частей программы, как отдельно, так и полностью, с использованием инструментального набора GNU Autotools.;
* используя изученные механизмы, разработать и отладить серверную и клиентскую часть;
* составить общее описание результатов, инструкции по сборке и использованию программы, а также инструкцию по получению документации, сформировать архив формата tar.gz и представить на проверку с исходными текстами программы. **Внимание**: исходные тексты программ должны соответствовать принятому стандарту кодирования, а также содержать комментарии в стиле системы Doxygen.

**3 Описание варианта задания**

Работа выполнена в соответствии с вариантом 20.

Клиент принимает от пользователя коэффициенты квадратного  
уравнения (ax2+bx+c=0) или кубического уравнения (ax3+bx2+cx+d=0), каждый  
из которых не равен нулю или единице, и отсылает их серверу.  
 Сервер принимает коэффициенты и выводит на экран разложение  
кубического или квадратного уравнения на множители.

**4 Ход выполнения**

**4.1 Клиент**

В файле client.c был написан основной код клиента, отвечающий за межпроцессное взаимодействие. Код клиента представлен в листинге 1.

Листинг 1 – Содержимое файла client.c

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include "client.h"

#include "interface.h"

#define PORT 5555

#define MAXDATASIZE 1024

int main(int argc, char\* argv[])

{

int sockfd; // дескриптор сокета

char buffer[MAXDATASIZE]; // буфер для приема и отправки данных

struct sockaddr\_in servAddr; // структура адреса сервера

// Объявляем переменные для коэффициентов уравнения

double a = 0;

double b = 0;

double c = 0;

double d = 0; // Добавляем переменную для четвертого коэффициента

// Вызываем функцию для обработки коэффициентов уравнения из командной строки

int result = ProcessCoefficients(argc, argv, &a, &b, &c, &d);

// Проверяем результат функции

if (result != 0)

{

fprintf(stderr, "Ошибка при обработке коэффициентов.\n");

exit(1);

}

// Создаем сокет с протоколом UDP

sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);

servAddr.sin\_family = AF\_INET; // семейство адресов IPv4

servAddr.sin\_addr.s\_addr = inet\_addr(

"127.0.0.1"); // адрес сервера (локальный)

servAddr.sin\_port = htons(PORT); // порт

// Формируем строку с коэффициентами уравнения в зависимости от количества аргументов

if (argc == 7) {

sprintf(buffer, "%lf %lf %lf", a, b, c);

} else {

sprintf(buffer, "%lf %lf %lf %lf", a, b, c, d);

}

// Отправляем данные серверу с помощью функции sendto

sendto(sockfd, buffer, strlen(buffer), 0,

(struct sockaddr \*) &servAddr, sizeof(servAddr));

printf("Отправлен запрос: %s\n",

buffer); // выводим отправленные данные

// Закрываем сокет

close(sockfd);

return 0;

}

**4.2 Сервер**

В файле server.c содержится основной код сервера. Работа с клиентом и сервером представлена на рисунке 1.

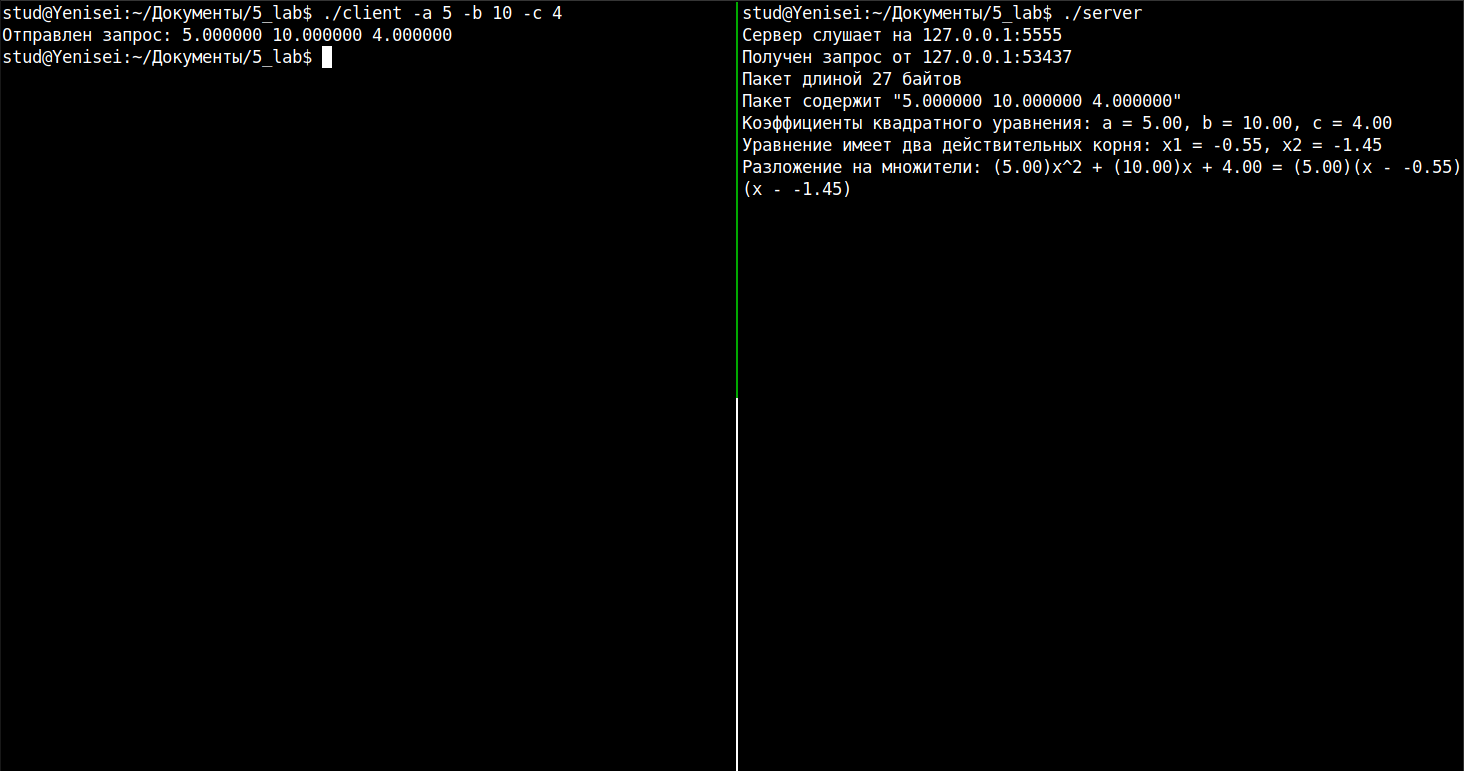


Рисунок 1 – Работа с программой

**4.3 Doxygen документация**

Все файлы содержат комментарии в стиле системы doxygen. Благодаря конфигурационному файлу была сгенерирована документация. Её внешний вид представлен на рисунке 2.

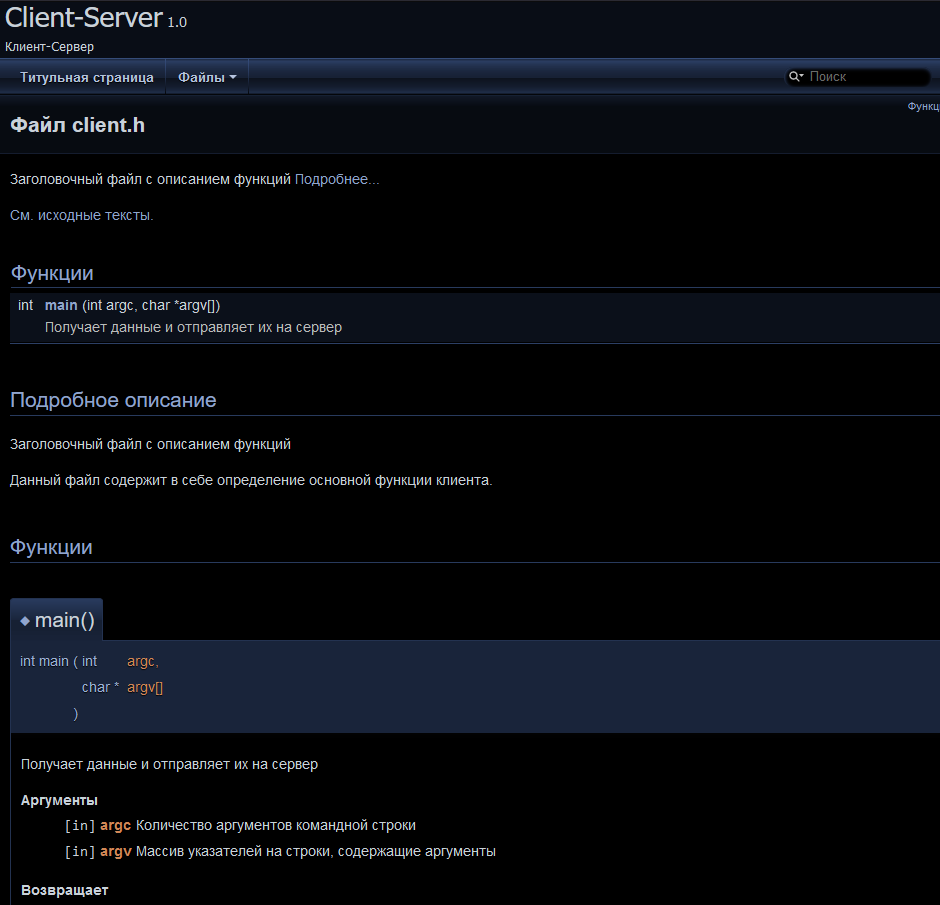


Рисунок 2 – Сгенерированная документация

**5** **Выводы**

В ходе работы были изучены особенности межпроцессного взаимодействия в OC GNU/Linux

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 СТО 4.2-07-2021 Система менеджмента качества. Общие требования  
к построению, изложении и оформлению документов учебной деятельности.  
Дата введения – 30.12.2-13.

**ПРИЛОЖЕНИЕ А**

**Исходный код файла server.c**

/\*! Функция сервера \*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

#include "server.h"

#include "logic.h"

#define PORT 5555

#define MAXBUF 1024

// Главная функция сервера

int main()

{

int numbytes; // количество байтов, полученных или отправленных

int sockfd; // дескриптор сокета

struct sockaddr\_in servaddr, cliaddr; // структуры адресов сервера и клиента

char buffer[MAXBUF]; // буфер для приема и отправки данных

socklen\_t len; // длина адреса клиента

// Создаем сокет с протоколом UDP

if ((sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) == -1)

{

perror("socket");

exit(1);

}

// Заполняем структуру адреса сервера

servaddr.sin\_family = AF\_INET; // семейство адресов IPv4

servaddr.sin\_addr.s\_addr = inet\_addr(

"127.0.0.1"); // автоматический выбор IP-адреса сервера

servaddr.sin\_port = htons(PORT); // порт сервера в сетевом порядке байтов

memset(servaddr.sin\_zero, '\0',

sizeof servaddr.sin\_zero); // обнуляем оставшуюся часть структуры

// связываем сокет с адресом сервера

if (bind(sockfd, (struct sockaddr \*) &servaddr,

sizeof servaddr) == -1)

{

perror("bind");

exit(1);

}

printf("Сервер слушает на %s:%d\n", inet\_ntoa(servaddr.sin\_addr),

ntohs(servaddr.sin\_port));

// бесконечный цикл обработки запросов клиентов

while (1)

{

// Принимаем данные от клиента

len = sizeof(cliaddr); // длина адреса клиента

// получаем сообщение от клиента в буфер buffer и запоминаем его адрес в cliaddr

if ((numbytes = recvfrom(sockfd, buffer, MAXBUF, 0,

(struct sockaddr \*) &cliaddr,

&len)) == -1)

{

perror("recvfrom");

exit(1);

}

buffer[numbytes] = '\0'; // добавляем нулевой символ в конец сообщения

printf("Получен запрос от %s:%d\n",

inet\_ntoa(cliaddr.sin\_addr),

ntohs(cliaddr.sin\_port)); // выводим адрес клиента

printf("Пакет длиной %d байтов\n", numbytes);

printf("Пакет содержит \"%s\"\n", buffer);

double a, b, c, d; // коэффициенты уравнения

int n; // количество коэффициентов уравнения

n = sscanf(buffer, "%lf %lf %lf %lf", &a, &b, &c,

&d); // читаем коэффициенты из сообщения

if (n == 3)

{

// квадратное уравнение

SolveQuadratic(a, b,

c); // решаем квадратное уравнение и выводим разложение на множители

}

else if (n == 4)

{

// кубическое уравнение

SolveCubic(a, b, c,

d); // решаем кубическое уравнение и выводим разложение на множители

}

else

{

printf("Неверный формат запроса.\n"); // некорректное количество коэффициентов

}

}

close(sockfd); // закрываем сокет

return 0;

}