Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

Лабораторная работа №6. Использование сигналов и средств хронометража в ОС GNU/Linux
Тема

Преподаватель _____ А. С. Кузнецов
подпись, дата — инициалы, фамилия

Студент КИ19-17/1б 031939175 _____ А. Д. Непомнящий
номер группы, зачетной книжки — подпись, дата — инициалы, фамилия

Красноярск 2021

1 Цель работы

Цель работы состоит в изучении использования сигналов и средств хронометража в ОС GNU/Linux.

2 Задачи

Выполнение работы сводится к следующим задачам.

1. Ознакомление с краткими теоретическими сведениями по использованию сигналов и средств работы с системными часами в ОС GNU/Linux.
2. Модификация результатов выполнения лабораторной работы №5 использованием программных средств для работы с сигналами и средствами хронометража.
3. Написание настоящего отчета защита его с исходными текстами и исполняемым модулем программы. Исходные тексты программ должны содержать комментарии в стиле системы doxygen, настоящий отчет должен включать содержимое файла configure.ac.

Вариант 14. Клиент принимает от пользователя беззнаковое целое число N – основание системы счисления (диапазон (1..20]) и последовательность цифр в соответствии с заданной системой счисления, отсылает серверу. Сервер принимает основание системы счисления и число в этой системе, выводит число на экран, переводит его в десятичную систему, выводит на экран, осуществляет его реверс (меняет порядок следования знаков на обратный), выводит на экран значение измененной последовательности, переводит ее в десятичную систему и выводит его на экран.

3 Исходные тексты программы

Далее приведено содержимое файлов с исходным ходом программы.

Листинг 1 – Код в файле task14.h

```
/*! \file   task14.h
 *  \brief  Header file of functions with numeral systems
 *          essential for task 14
 */

#include <stdbool.h>
#include "input.h"
#include <inttypes.h>

#ifndef LAB1_TASK14_H
#define LAB1_TASK14_H


/*! \struct taskData
 *  \brief Struct for PerformTask() function
 *
 *  \details Keeps data that is used as PerformTask() argument
 */
typedef struct
{
    /*!
     * Number in required numeric system
     */
    char number[INPUT_SIZE];
    /*!
     * Radix of numeric system
     */
    int8_t radix;
} taskData;

/*! \brief Performs task14 with required output
 *
 *  \param data argument for task 14
 */
void PerformTask(taskData* data);

/*! \brief Converts number in any (2-20) numeral system to decimal
 *
 *  \param number number to convert.
 *  \param radix radix of numeral system.
 *  \return Integer conversion result.
 */
int AnyNumeralSystemToDecimal(char* number, int radix);

/*! \brief Checks if number only contains digits, allowed for this numeral
 *  system
 *
 *  \param numberToCheck number to check.
 *  \param radix radix of numeral system.
 *  \return true if number only contains digits, allowed for this numeral
*  system, false - otherwise.
*/
```

Окончание листинга 1

```
bool CheckRadixMatch(char* numberToCheck, int radix);

/*! \brief Checks if number is not too big to be written to int after
 *  conversion
 *
 *  \param numberToCheck number to check.
 *  \param radix radix of numeral system.
 *  \return true if number is not too big to be written to int after
 *  conversion, false - otherwise
 */
bool CheckIntOverflow(char* numberToCheck, int radix);

/*! \brief Checks if number can be numeral system radix for task 14
 *
 *  \param intToCheck number to check.
 *  \return true if number can be numeral system radix for task 14
 *  false - otherwise
 */
bool RadixInputCheck(int intToCheck);


#endif //LAB1_TASK14_H
```

Листинг 2 – Код в файле task14.c

```
/*! \file   input.c
 *  \brief  Implements functions of task14.h
 */

#include "task14.h"
#include <math.h>
#include <string.h>
#include <stdbool.h>
#include <stdio.h>

/*! \enum
 *  \brief  Essential constants for task 14
 */
enum NumeralSystemsConstants
{
    VIGESIMAL_A = 'A', /** Digit next to 9 */
    MIN_RADIX = 2,     /** Minimal numeral system radix */
    MAX_RADIX = 20     /** Maximal numeral system radix for task */
};

void PerformTask(taskData* data)
{
    char reversedNumber[INPUT_SIZE];
    char* number = data->number;
    int8_t radix = data->radix;

    for (int i = (int) strlen(number) - 1; i >= 0; i--)
    {
        reversedNumber[strlen(number) - (i + 1)] = number[i];
    }
    reversedNumber[strlen(number)] = '\0';

    while (reversedNumber[strlen(reversedNumber) - 1] == '0')
    {
```

Продолжение листинга 2

```
        reversedNumber[strlen(reversedNumber) - 1] = '\0';
    }

    printf("Original: %s\n", number);
    printf("To decimal: %d\n",
           AnyNumeralSystemToDecimal(number, radix));
    printf("Reversed: %s\n", reversedNumber);
    if (CheckIntOverflow(reversedNumber, radix))
    {
        printf("Reversed to decimal: %d\n",
               AnyNumeralSystemToDecimal(reversedNumber, radix));
    }
    else
    {
        printf("Reversed number is too big");
    }
}

int AnyNumeralSystemToDecimal(char* number, int radix)
{
    int result = 0;
    int multiplier = 1;
    int currentDigit;
    for (int i = (int) strlen(number) - 1; i >= 0; i--)
    {
        if (number[i] >= VIGESIMAL_A)
        {
            currentDigit = 10 + number[i] - VIGESIMAL_A;
        }
        else
        {
            currentDigit = number[i] - '0';
        }
        result += currentDigit * multiplier;
        multiplier *= radix;
    }
    return result;
}

bool CheckRadixMatch(char* numberToCheck, int radix)
{
    int currentDigit;
    for (int i = 0; i < strlen(numberToCheck); i++)
    {
        if (numberToCheck[i] >= VIGESIMAL_A)
        {
            currentDigit = 10 + numberToCheck[i] - VIGESIMAL_A;
        }
        else
        {
            currentDigit = numberToCheck[i] - '0';
        }
        if (currentDigit >= radix || currentDigit < 0)
        {
            return false;
        }
    }
    return true;
```

Окончание листинга 2

```
}

bool CheckIntOverflow(char* numberToCheck, int radix)
{
    return (double) strlen(numberToCheck) <
           (log((double) __INT_MAX__) / log((double) radix) - 1);
}

bool RadixInputCheck(int intToCheck)
{
    if (intToCheck < MIN_RADIX || intToCheck > MAX_RADIX)
    {
        return false;
    }
    return true;
}
```

Листинг 3 – Код в файле server.c

```
/*! \file   server.c
 *  \brief  Code of server executable and server's task
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <signal.h>
#include "task14.h"
#include "logOutput.h"
#include "timer.h"


char g_logPath[INPUT_SIZE] = "log";
int g_idleTime = 10;

/*! \brief Signal handler for server
 */
void ServerSignalHandler(int signum)
{
    if (signum == SIGINT)
    {
        WriteLogEntry(g_logPath, "Server terminated by Ctrl+C\n");
        exit(0);
    }
    else if (signum == SIGALRM)
    {
        WriteLogEntry(g_logPath, "Server terminated by timer\n");
        exit(0);
    }
    else if (signum == SIGTERM)
    {
        WriteLogEntry(g_logPath, "Server terminated by kill signal\n");
        exit(0);
    }
```

Продолжение листинга 3

```
}

/*! \brief Reads data from socket and calls PerformTask()
 *
 *  \param serverSocket descriptor of socket to listen
 */
int ServerTask(int serverSocket)
{
    struct sockaddr_in clientName;
    socklen_t clientNameLength = sizeof(clientName);
    taskData* data;
    data = (taskData*) malloc(sizeof(taskData));

    struct sigaction sa = CreateSAHandler(ServerSignalHandler);
    sigaction(SIGINT, &sa, NULL);
    sigaction(SIGTERM, &sa, NULL);
    sigaction(SIGALRM, &sa, NULL);

    struct itimerval timer = InitTimer(g_idleTime, 0);
    setitimer(ITIMER_REAL, &timer, NULL);


    while (true)
    {
        int recvResult = (int) recvfrom(serverSocket, data, sizeof(taskData),
                                        0,
                                        (struct sockaddr*) &clientName,
                                        &clientNameLength);
        if (-1 == recvResult)
        {
            perror("recvfrom");
        }
        if (recvResult > 0)
        {
            WriteLogEntry(g_logPath, "Got new task\n");
            PerformTask(data);
            RollbackTimer(&timer, g_idleTime, 0);
        }
    }
    free(data);
    return 0;
}

/*! \brief main function of server
 */
int main(int argc, char* const argv[])
{
    if (argc != 4)
    {
        fprintf(stderr, "Expected arguments:\nPort number\n"
                        "Log file name\nIdle timeout\n");
        return EXIT_FAILURE;
    }
    int socketFileDescriptor = -1;
    int portNumber = atoi(argv[1]);
    strcpy(g_logPath, argv[2]);
    g_idleTime = atoi(argv[3]);
    struct sockaddr_in name;
```

Окончание листинга 3

```
    socketFileDescriptor = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    int i = 1;
    setsockopt(socketFileDescriptor, SOL_SOCKET, SO_REUSEADDR,
               (const char*) &i, sizeof(i)
    );
    bzero((char*) &name, sizeof(name));
    name.sin_family = AF_INET;
    name.sin_port = htons((u_short) portNumber);
    name.sin_addr.s_addr = INADDR_ANY;
    if (-1 == bind(socketFileDescriptor, (const struct sockaddr*) &name,
                   sizeof(name)))
    {
        perror("bind ");
        close(socketFileDescriptor);
        exit(1);
    }
    ServerTask(socketFileDescriptor);
    close(socketFileDescriptor);
}
```

Листинг 4 – Код в файле client.c

```
/*! \file   client.c
 *  \brief  Code of client executable
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include "signal.h"
#include "task14.h"
#include "logOutput.h"
#include "timer.h"


char g_logPath[INPUT_SIZE] = "log";
int g_idleTime = 10;

/*! \brief Signal handler for server
 */
void ClientSignalHandler(int signum)
{
    if (signum == SIGINT)
    {
        WriteLogEntry(g_logPath, "Client terminated by Ctrl+C\n");
        exit(0);
    }
    else if (signum == SIGALRM)
    {
        WriteLogEntry(g_logPath, "Client terminated by timer\n");
        exit(0);
    }
    else if (signum == SIGTERM)
    {
```

Продолжение листинга 4

```
        WriteLogEntry(g_logPath, "Client terminated by kill signal\n");
        exit(0);
    }
}

/*! \brief Catches ctrl+C signal, closes socket and terminates server
 *  \details Parses CL arguments, checks them and sends to the server
 */
int main(int argc, const char* argv[])
{
    if (argc != 5)
    {
        fprintf(stderr, "Expected arguments:\nServer address\nPort"
                        "number\nLog file name\nIdle timeout\n");
        return EXIT_FAILURE;
    }
    strcpy(g_logPath, argv[3]);
    g_idleTime = atoi(argv[4]);

    struct sigaction sa = CreateSAHandler(ClientSignalHandler);
    sigaction(SIGINT, &sa, NULL);
    sigaction(SIGTERM, &sa, NULL);
    sigaction(SIGALRM, &sa, NULL);

    struct itimerval timer = InitTimer(g_idleTime, 0);
    setitimer(ITIMER_REAL, &timer, NULL);

    int8_t radix;
    char number[INPUT_SIZE];

    int socketFileDescriptor;
    int portNumber = atoi(argv[2]);
    struct sockaddr_in name;
    memset((char*) &name, 0, sizeof(name));
    name.sin_family = AF_INET;
    name.sin_addr.s_addr = inet_addr(argv[1]);
    if (INADDR_NONE == name.sin_addr.s_addr)
    {
        perror("inet_addr");
        exit(1);
    }
    name.sin_port = htons((u_short) portNumber);
    socketFileDescriptor = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    if (socketFileDescriptor < 0)
    {
        perror("socket");
        exit(1);
    }

    while (true)
    {
        printf("Enter base of numeral system (2 - 20)\n");
        radix = (int8_t) CheckedInputInt(RadixInputCheck);
        printf("Enter number in chosen system. Use \'A\' - \'J\' as"
               "digits for >10-based systems\n");

        while (true)
        {
```

Окончание листинга 4

```
            scanf("%s", number);
            if (CheckIntOverflow(number, radix) &&
                CheckRadixMatch(number, radix))
            {
                break;
            }
            printf("Wrong format or too big number!\n");
        }

        RollbackTimer(&timer, g_idleTime, 0);

        taskData* data;
        data = (taskData*) malloc(sizeof(taskData));
        strcpy(data->number, number);
        data->radix = radix;

        int resSend;
        resSend = (int) sendto(socketFileDescriptor, data, sizeof(taskData),
                               0,
                               (struct sockaddr*) &name, sizeof(name));
        if (0 > resSend)
        {
            perror("sendto");
            free(data);
            exit(1);
        }
        WriteLogEntry(g_logPath, "Sent task\n");
        free(data);
    }
    close(socketFileDescriptor);
    return 0;
}
```

Листинг 5 – код в файле input.h

```
/*! \file   input.h
 *  \brief  Header containing essential input constants
 */

#ifndef LAB1_INPUT_H
#define LAB1_INPUT_H

#include <stdbool.h>


/*! \enum
 *  \brief  Size of string for input
 */
enum Sizes
{
    INPUT_SIZE = 200
};

/*! \brief Reads int
 *
 *  \details Reads int with additional check. Continues reading until
 *  correct value is read.
 *
```

Окончание листинга 5

```
 *  \param additionalCheck Pointer to the function that checks additional
condition.
 *  \return Integer read correct integer.
 */
int CheckedInputInt(bool(* additionalCheck)(int));


#endif //LAB1_INPUT_H
```

Листинг 6 – Код в файле input.c

```
/*! \file   input.c
 *  \brief  Implements functions of input.h
 */

#include "input.h"
#include "stdio.h"


int CheckedInputInt(bool(* additionalCheck)(int))
{
    int result;
    char inputString[INPUT_SIZE];

    while (true)
    {
        scanf("%s", inputString);
        int flag = sscanf(inputString, "%d", &result);
        if (flag == 0 || flag == EOF)
        {
            printf("Input error!\n");
            continue;
        }
        if (!additionalCheck(result))
        {
            printf("Input error!\n");
            continue;
        }
        return result;
    }
}
```

Листинг 7 – Код в файле logOutput.h

```
/*! \file   logOutput.h
 *  \brief  Header for function to make an entry to the log
 */

#ifndef LOGOUTPUT_H
#define LOGOUTPUT_H


/*! \brief Writes an entry to log by path
 *
 *  \param logPath  Path for log file
 *  \param info     String that will be written to log
 */
void WriteLogEntry(char* logPath, char* info);
```

Окончание листинга 7

```
#endif //LOGOUTPUT_H
```

Листинг 8 – Код в файле logOutput.c

```
/*! \file   logOutput.c
 *  \brief  Implements functions declared in logOutput.h and defines
 *  essential functions for it.
 */

#include "logOutput.h"
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include "input.h"
#include "timer.h"


/*! \brief Writes info from buffer by pointer into file
 *
 *  \param fd    File descriptor
 *  \param info  Variable with information that will be loaded into file
 *  \param size  The number of bytes that will be loaded into file
 *
 *  \return 0 on success, -1 otherwise
 */
int WriteInfo(int fd, void* info, size_t size)
{
    int writeReturn = (int) write(fd, info, size);
    if (writeReturn == -1)
    {
        perror("Write into file went wrong.");
        return -1;
    }
    if (writeReturn == 0)
    {
        perror("Nothing was written.");
        return -1;
    }
    return 0;
}

/*! \brief Opens file
 *
 *  \details Opens file, changes file descriptor by pointer and
 *  reports errors.
 *
 *  \param fd        File descriptor pointer
 *  \param filename  Name of file to open
 *
 *  \return 0 on success, -1 otherwise
 */
int OpenFile(int* fd, char* filename)
{
    *fd = open(filename, O_WRONLY | O_CREAT | O_APPEND);
    if (*fd < 0)
    {
```

Окончание листинга 8

```
        return -1;
    }
    return 0;
}

void WriteLogEntry(char* logPath, char* info)
{
    int fd;
    if (OpenFile(&fd, logPath) != 0)
    {
        perror("log file");
        return;
    }
    char* timeString = GetTimeString();
    WriteInfo(fd, timeString, strlen(timeString));
    WriteInfo(fd, "\n", sizeof(char));
    WriteInfo(fd, info, INPUT_SIZE);
    WriteInfo(fd, "\n", sizeof(char));
    close(fd);
}
```

Листинг 9 – Код в файле timer.h

```
/*! \file   timer.h
 *  \brief  Header for functions for working with system time
 *  essential functions for it.
 */

#ifndef LR6_TIMER_UTIL_H
#define LR6_TIMER_UTIL_H

#include <stdio.h>
#include <signal.h>
#include <string.h>
#include <unistd.h>
#include <time.h>
#include <stdlib.h>
#include <sys/time.h>


/*!
 * \brief  Returns string representation of current time
 * \return String with current time
*/
char* GetTimeString();

/*!
 * \brief Initiates timer
 * \param sec   Seconds
 * \param usec  Microseconds
 * \return timer itimerval
*/
struct itimerval InitTimer(int sec, int usec);

/*!
 * \brief Rolls timer back to value
 * \param timer   timer itimerval
 * \param sec     Seconds
```

Окончание листинга 9

```
 * \param usec    Microseconds
*/
void RollbackTimer(struct itimerval* timer, int sec, int usec);

/*!
 * \brief Creates signal handler
 * \param sa_handler function
 * \return struct sigaction
*/
struct sigaction CreateSAHandler(void* TimerHandler);


#endif
```

Листинг 10 – Код в файле timer.c

```
/*! \file   timer.c
 *  \brief  Implements functions from timer.h
 */

#include <time.h>
#include "timer.h"
#include "input.h"


char* GetTimeString()
{
    struct tm* localTime;
    const time_t timer = time(NULL);
    localTime = localtime(&timer);
    char* result;
    result = (char*) malloc(INPUT_SIZE);
    strftime(result, INPUT_SIZE, "%Y-%m-%d %H:%M:%S", localTime);
    return (result);
}


struct itimerval InitTimer(int sec, int usec)
{
    struct itimerval timer;
    timer.it_value.tv_sec = sec;
    timer.it_value.tv_usec = usec;
    timer.it_interval.tv_sec = 0;
    timer.it_interval.tv_usec = 0;
    return timer;
}

void RollbackTimer(struct itimerval* timer, int sec, int usec)
{
    timer->it_value.tv_sec = sec;
    timer->it_value.tv_usec = usec;
    setitimer(ITIMER_REAL, timer, NULL);
}

struct sigaction CreateSAHandler(void* TimerHandler)
{
    struct sigaction sa;
    memset(&sa, 0, sizeof(sa));
```

Окончание листинга 10

```
    sa.sa_handler = TimerHandler;
    return sa;
}
```

4 Содержимое файла configure.ac

На следующем листинге приведено содержимое скрипта configure.

Листинг 11 – Код в файле configure.ac

```
#                                               -*- Autoconf -*-
# Process this file with autoconf to produce a configure script.

AC_PREREQ([2.69])
AC_INIT([FULL-PACKAGE-NAME], [VERSION], [BUG-REPORT-ADDRESS])
AC_CONFIG_SRCDIR([task14.c])
AC_CONFIG_HEADERS([config.h])

# Checks for programs.
AC_PROG_CC

# Checks for libraries.
# FIXME: Replace `main' with a function in `-lm':
AC_CHECK_LIB([m], [main])

# Checks for header files.
AC_CHECK_HEADERS([fcntl.h inttypes.h netinet/in.h stdlib.h string.h sys/socket.h
sys/time.h unistd.h])

# Checks for typedefs, structures, and compiler characteristics.
AC_CHECK_HEADER_STDBOOL
AC_TYPE_INT8_T
AC_TYPE_SIZE_T

# Checks for library functions.
AC_FUNC_MALLOC
AC_CHECK_FUNCS([bzero memset socket])

AC_CONFIG_FILES([Makefile])
AC_OUTPUT
```

5 Тестовые примеры работы программ

Далее на рисунках приведены тестовые примеры работы программы.

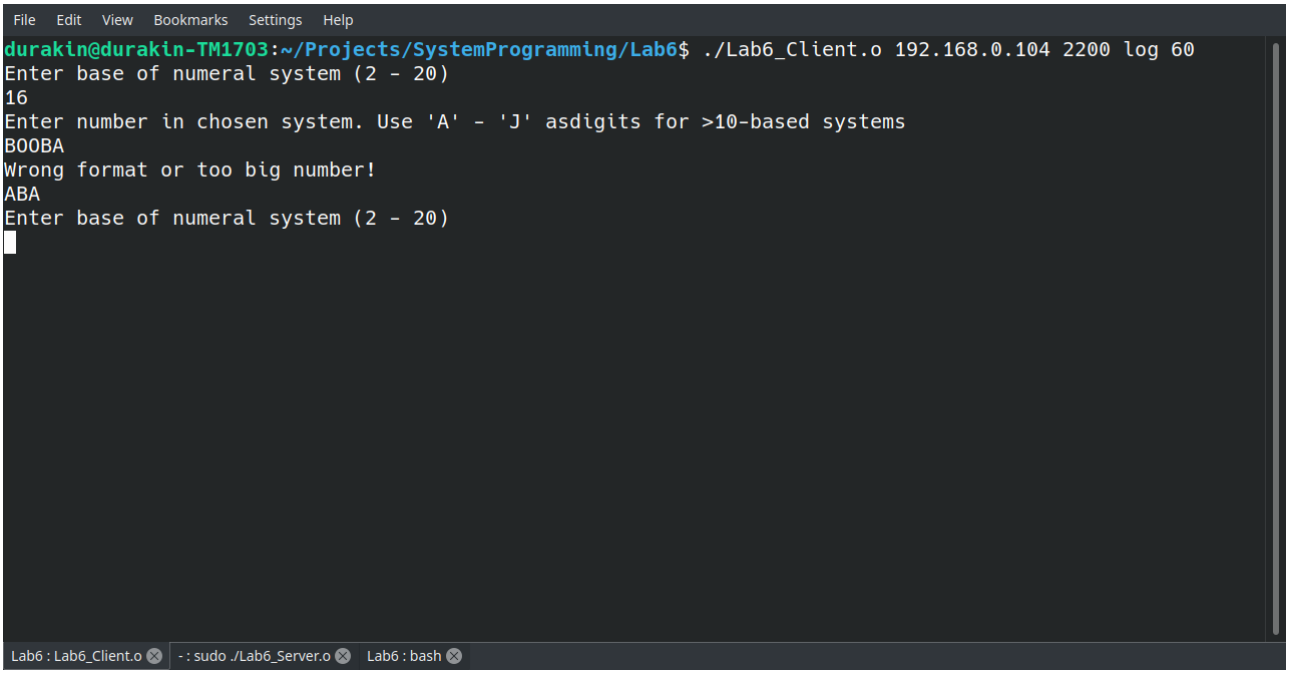

Рисунок 1 – Запуск клиента

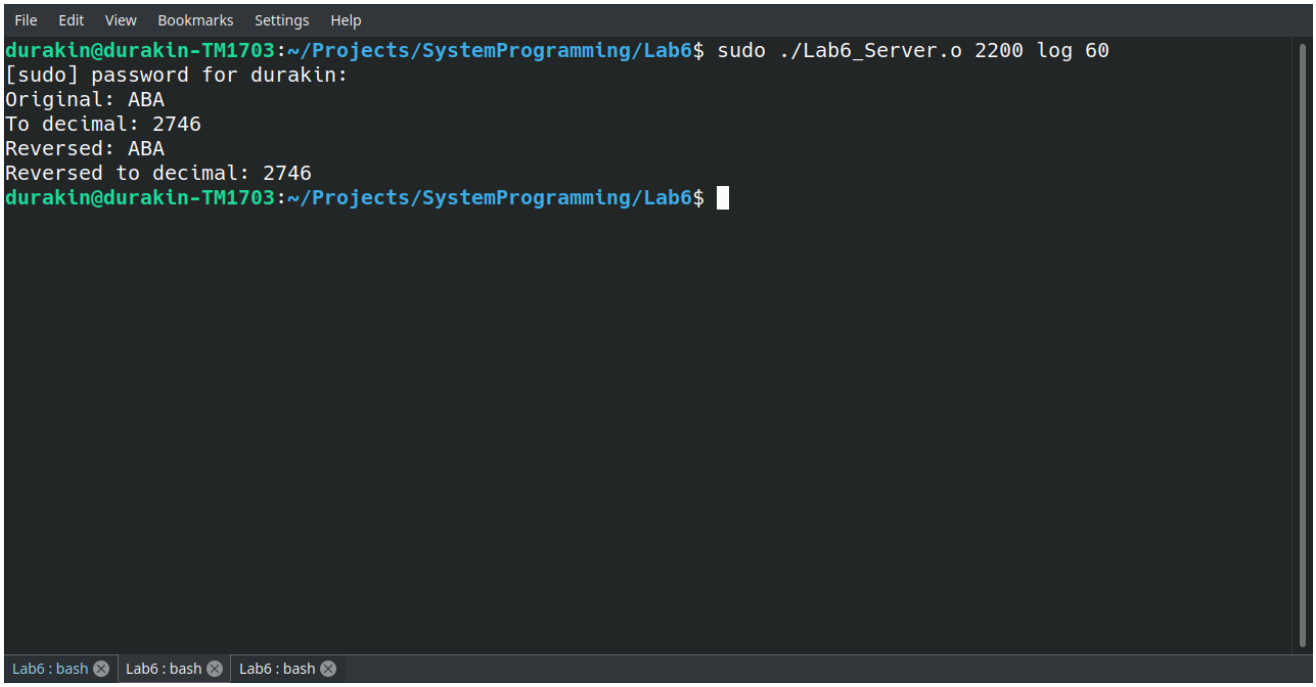

Рисунок 2 – Реакция сервера

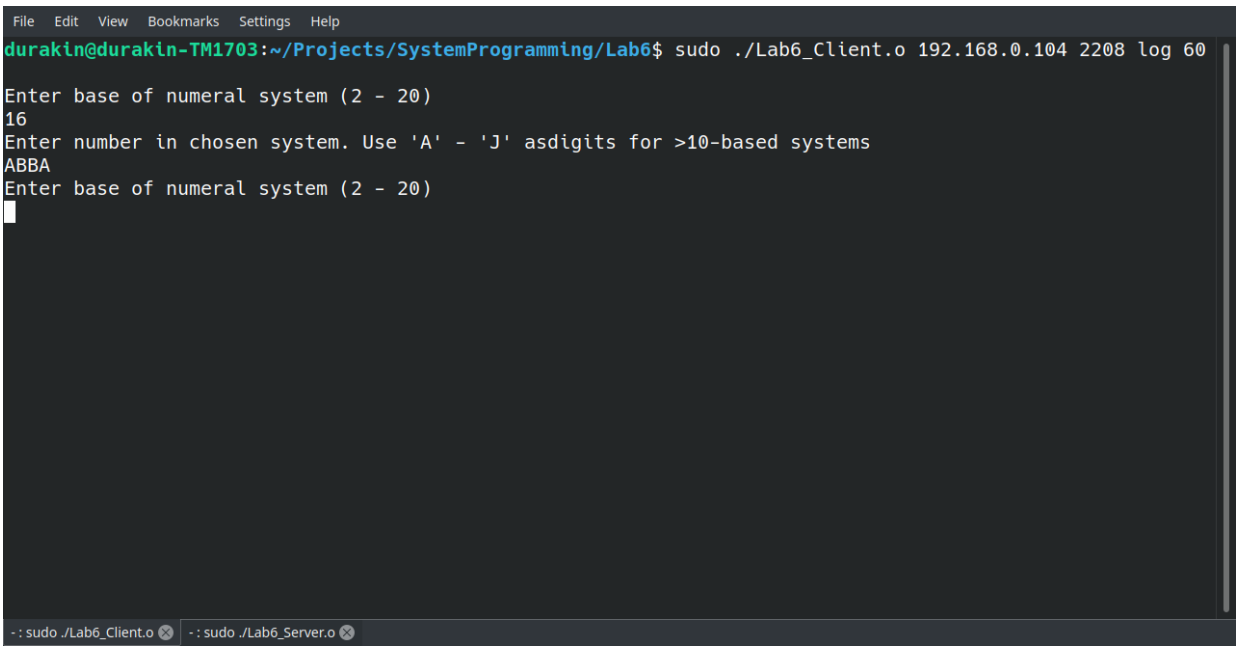

Рисунок 3 – Запуск клиента с неверным портом

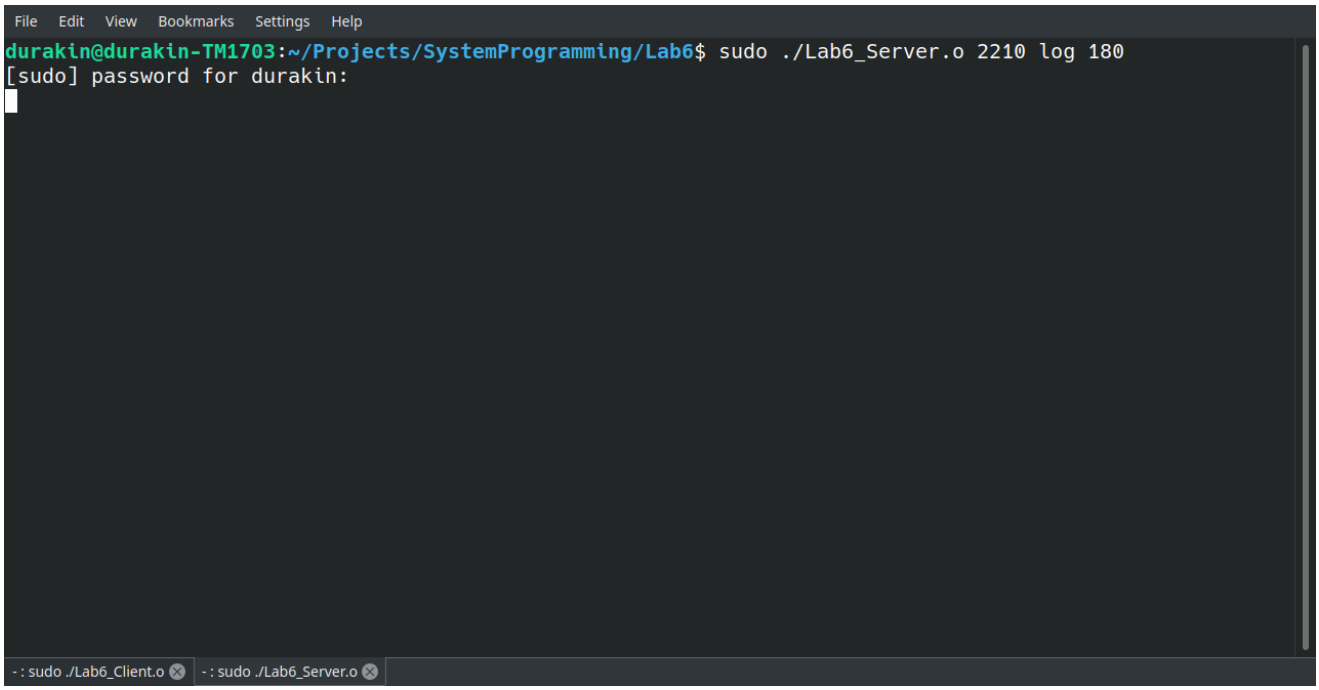

Рисунок 4 – Реакция сервера (отсутствует)

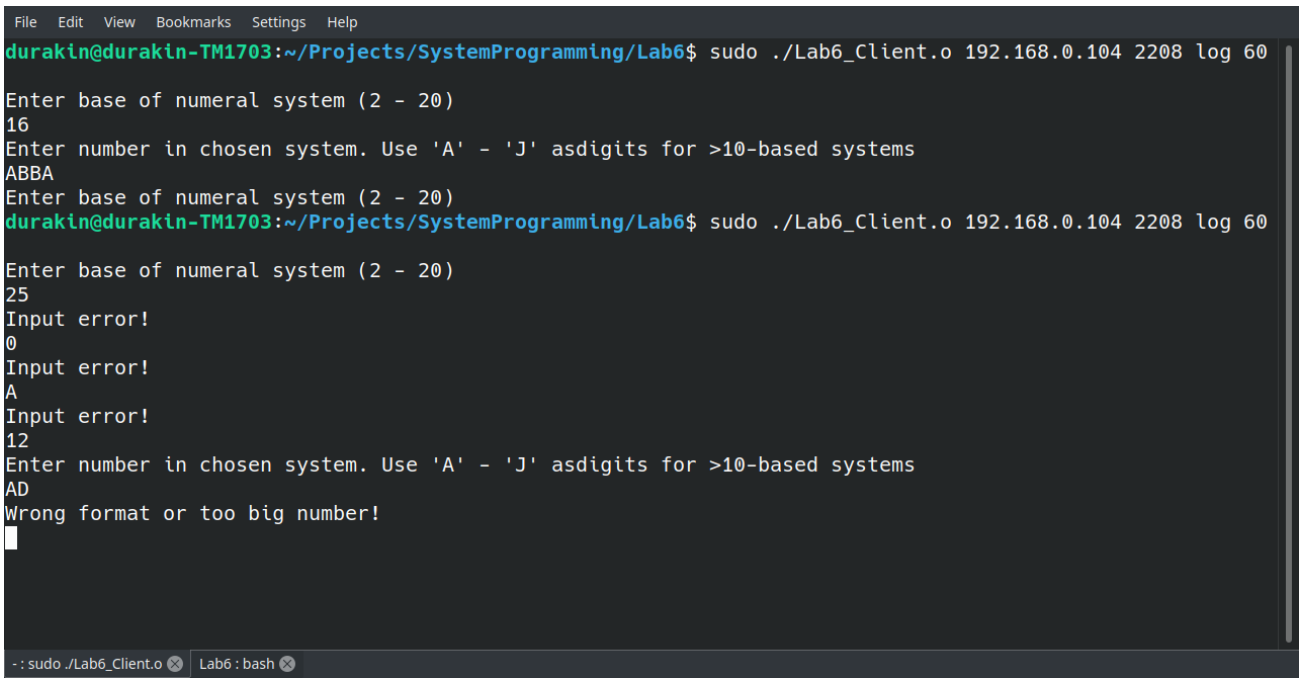

Рисунок 5 – Попытки ввода неверных данных

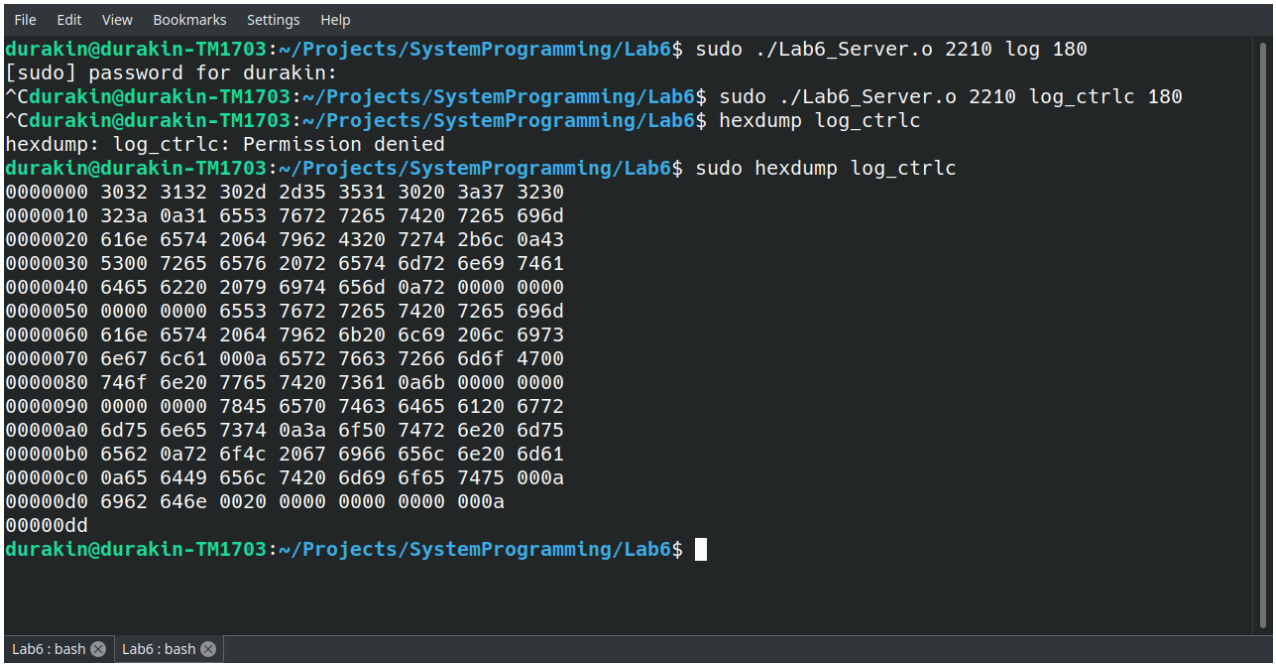

Рисунок 6 – Реакция сервера на ввод Ctrl + C

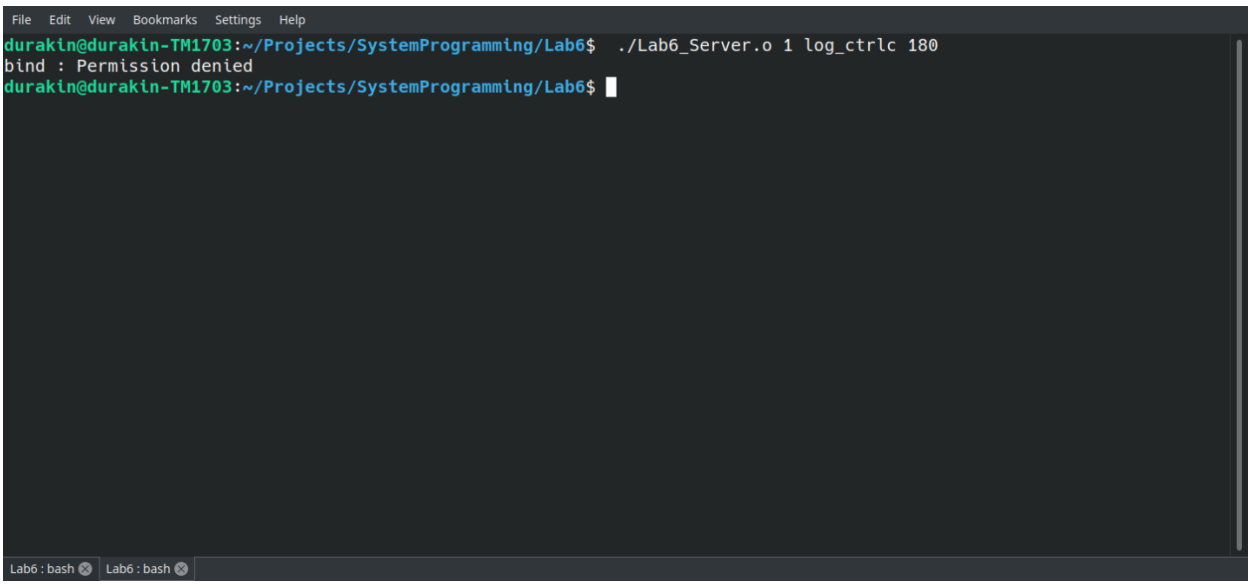

Рисунок 7 – Попытка запуска сервера с указанием “плохого” порта

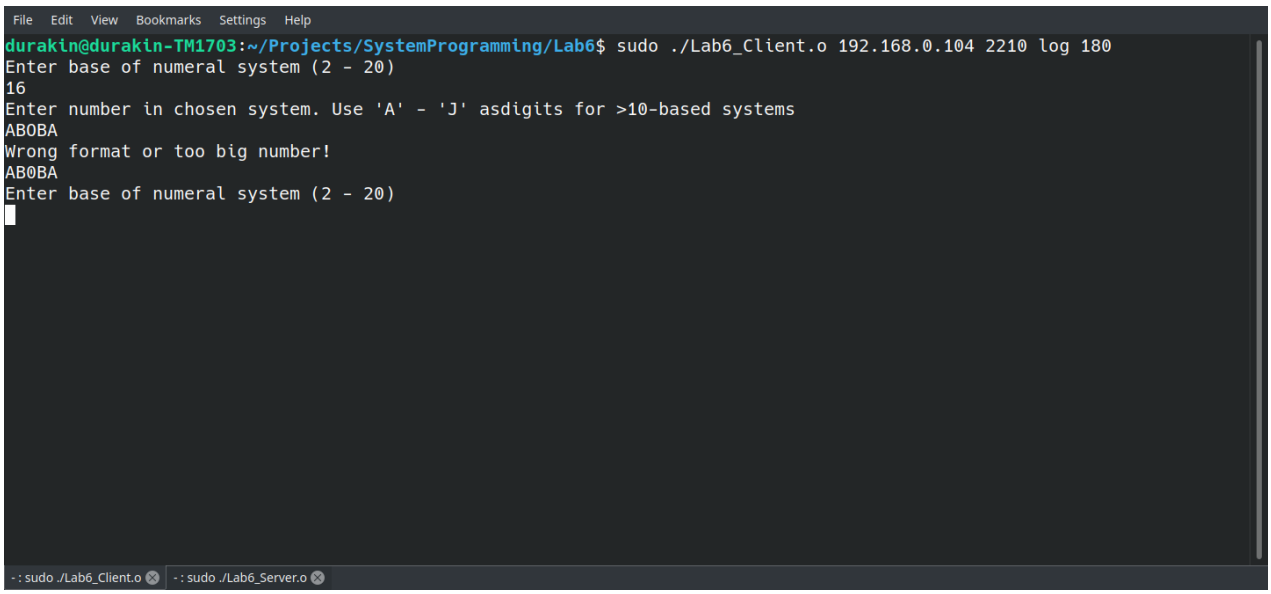

Рисунок 8 – Запуск клиента

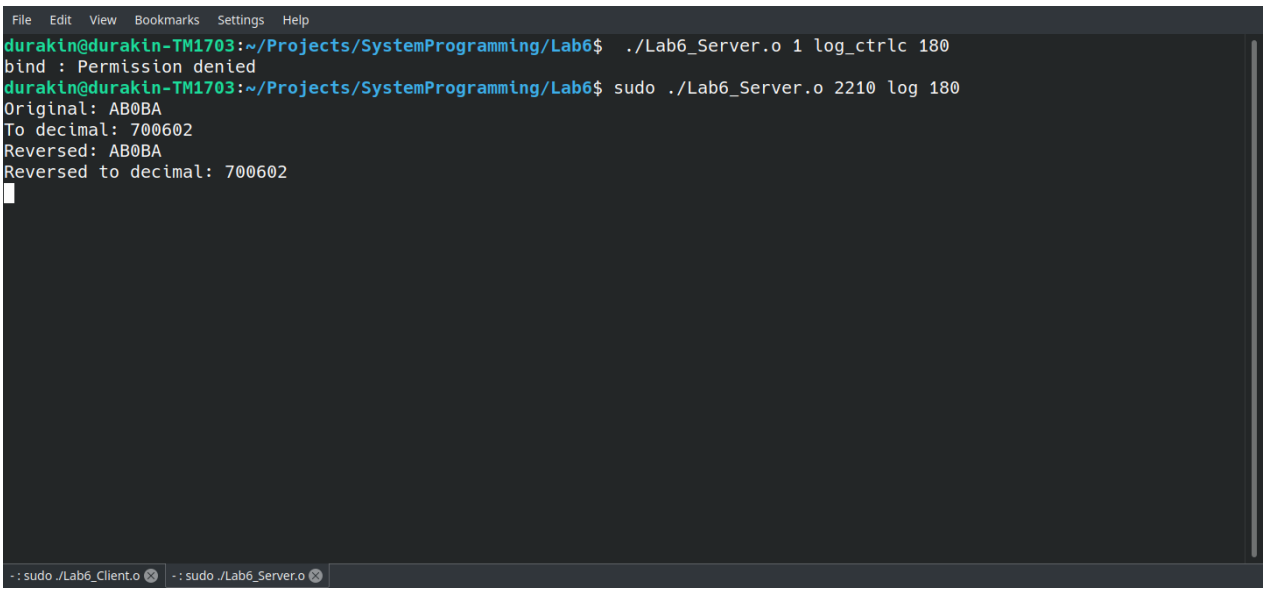

Рисунок 9 – Реакция сервера

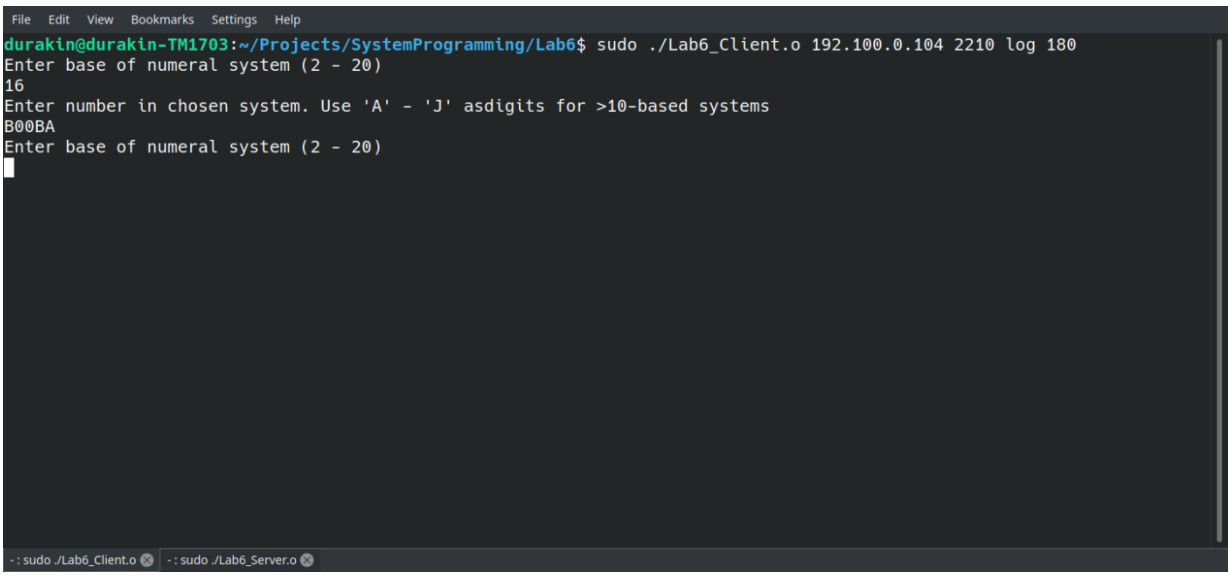

Рисунок 10 – Запуск клиента с передачей неверного IP-адреса сервера

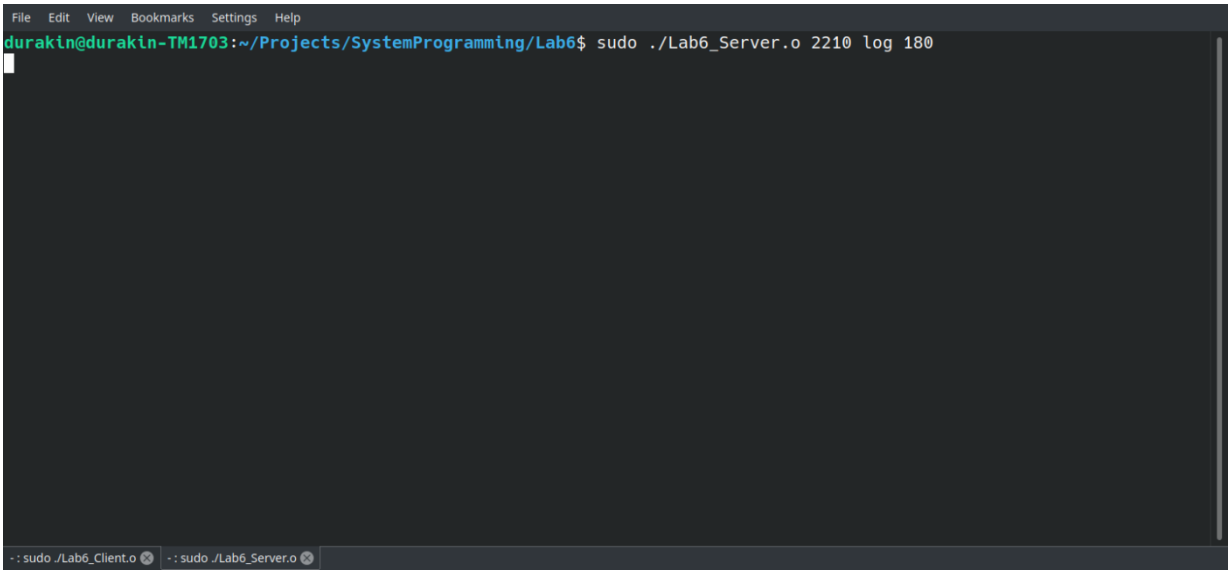

Рисунок 10 – Реакция сервера (отсутствует)