

Московский Авиационный Институт  
(Национальный Исследовательский Университет)  
Факультет информационных технологий и прикладной математики  
Кафедра вычислительной математики и программирования

**Курсовая по курсу  
«Операционные системы»**

**Управление серверами сообщений, применение отложенных  
вычислений, интеграция программных систем друг с другом.**

Студент: Яруллин А.Р.  
Группа: М8О-201Б-21  
Вариант: (на  
удовлетворительно)  
Преподаватель: Миронов Евгений Сергеевич  
Оценка: \_\_\_\_\_  
Дата: \_\_\_\_\_  
Подпись: \_\_\_\_\_

Москва, 2022

## **Постановка задачи**

Необходимо написать 3-и программы. Далее будем обозначать эти программы А, В, С. Программа А принимает из стандартного потока ввода строки, а далее их отправляет программе С. Отправка строк должна производиться построчно. Программа С печатает в стандартный вывод, полученную строку от программы А. После получения программа С отправляет программе А сообщение о том, что строка получена. До тех пор, пока программа А не примет «сообщение о получении строки» от программы С, она не может отправлять следующую строку программе С.

Программа В пишет в стандартный вывод количество отправленных символов программой А и количество принятых символов программой С. Данную информацию программа В получает от программ А и С соответственно.

## **Описание программы**

Программа была реализована с помощью библиотеки ZeroMQ. Данная библиотека предлагает разработчику более высокий уровень абстракции при работе с сокетами/соединениями/очередями.

Программа А подключается к узлам В и С, после получает на вход строки, каждую полученную строку она сразу же отправляет размер отправляемой строки в С и ждет ответа «String received», дальше отправляет сообщение с длиной строки в В и ждет ответ "OK".

При получении сообщения от А, С проверяет, что это не «exit» и выводит в свой терминал сообщение, переданное от А, и отвечает обратным сообщением «String received», а в В передает размер сообщения. Если С получил «exit», то узел перестает принимать сообщения и выключается.

В принимает сообщение от А (размер сообщения, которое А передало С) и выводит на экран, и отправляет «OK». И так же принимает сообщение от С (размер сообщения, которое С получило от А), печатает на экран и отправляет «OK» обратно. Если получил «exit», то узел перестает принимать сообщения и выключается.

Чтобы прекратить ввод в А, необходимо нажать Ctrl+D (на linux-системах данная команда посылает EOF во входной поток).

## Листинг А.cpp

```
#include <iostream>
#include <string>
#include <unistd.h>
#include "zmq_functions.h"

const std::string АДРЕСС_C = "tcp://127.0.0.1:4040";
const std::string АДРЕСС_B = "tcp://127.0.0.1:4041";

int main(){
    zmq::context_t context;
    std::string str;
    zmq::socket_t B(context, ZMQ_REQ);
    zmq::socket_t C(context, ZMQ_REQ);

    B.connect(АДРЕСС_B);
    C.connect(АДРЕСС_C);

    std::string answer;

    pid_t pid1 = fork();
    pid_t pid2 = 1;

    if (pid1 > 0){
        pid2 = fork();
    }

    if (pid1 < 0 || pid2 < 0)
    {
        perror("process error ");
        exit(EXIT_FAILURE);
    }
    if (pid1 == 0)
    {
        execl("./B", "./B", NULL);
    }
    else if (pid2 == 0)
    {
        execl("./C", "./C", NULL);
    }

    while(std::getline(std::cin, str))
    {
        int size = str.size();
        send_message(C, str);
        answer = receive_message(C);

        while(answer != "String recieved"){
            std::cout << "Error: string not recieved" << std::endl;
```

```

    }

    send_message(B, std::to_string(size));

    answer = receive_message(B);

    if(answer != "OK"){
        std::cout << "Error: string not recieved" << std::endl;
    }
}

send_message(C, "exit");

if (receive_message(C) == "OK")
{
    C.disconnect(ADRESS_C);
    C.close();

    send_message(B, "exit");

    if (receive_message(B) == "OK")
    {
        B.disconnect(ADRESS_B);
        B.close();
    }
}
return 0;
}

```

## Листинг В.cpp

```

#include <iostream>
#include <string>
#include "zmq_functions.h"

const std::string ADRESS_A = "tcp://127.0.0.1:4041";
const std::string ADRESS_C = "tcp://127.0.0.1:4042";

int main(){
    zmq::context_t context;
    zmq::socket_t A(context, ZMQ_REP);
    zmq::socket_t C(context, ZMQ_REP);

    A.bind(ADRESS_A);
    C.bind(ADRESS_C);

    std::string answerA, answerC;
    while(1)
    {
        answerA = receive_message(A);
        send_message(A, "OK");
    }
}

```

```

        if (answerA == "exit"){
            break;
        }

        std::cout << "A: " << answerA << std::endl;

        answerC = receive_message(C);
        send_message(C, "OK");

        std::cout << "C: " << answerC << std::endl;

    }
    A.unbind(ADDRESS_A);
    C.unbind(ADDRESS_C);
    A.close();
    C.close();

    return 0;
}

```

**Листинг C.cpp**

```

#include <iostream>

#include <string>

#include "zmq_functions.h"

const std::string ADDRESS_A = "tcp://127.0.0.1:4040";
const std::string ADDRESS_B = "tcp://127.0.0.1:4042";

int main(){

    zmq::context_t context;

    zmq::socket_t A(context, ZMQ_REP);
    zmq::socket_t B(context, ZMQ_REQ);

    A.bind(ADDRESS_A);
    B.connect(ADDRESS_B);

    std::string answer;

    while(1)
    {
        answer = receive_message(A);
    }
}

```

```

        if (answer == "exit"){
            send_message(A, "OK");
            break;
        }

        std::cout << answer << std::endl;

        send_message(A, "String recieved");

        int size = answer.size();
        send_message(B, std::to_string(size));
        answer = receive_message(B);

        if (answer != "OK"){
            std::cout << "Error: string not recieved" << std::endl;
        }

    }

    A.unbind(ADRESS_A);
    B.disconnect(ADRESS_B);

    A.close();
    B.close();

    return 0;
}

```

## Набор тестов:

yarullin@yarullin-OMEN-by-HP-Laptop-15-dc1xxx:~/OS-Labs/KP/src\$ ./A

Hello

Hello

A: 5

C: 5

kak dela?

kak dela?

A: 9

C: 9

noooooorm////\\

noooooorm////\\

A: 19

C: 19

:((

:((

A: 4

C: 4

## Вывод

Данная курсовая работа основывается на знаниях, полученных в ходе изучения курса. По итогу мы получили несколько программ, которые взаимодействуют друг с другом с помощью сокетов.

Задача курсового проекта не сложна в реализации, но ее реализация обобщает и закрепляет полученные в курсе знания.