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: Знай Артемий Олегович

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 A, B, C. Программа A принимает из стандартного потока ввода строки, а далее их отправляет программе C. Отправка строк должна производится построчно. Программа C печатает в стандартый вывод, полученную строку от программы A. После получения программа C отправляет программе А сообщение о том, что строка получена. До тех пор, пока программа А не примет «сообщение о получение строки» от программы C, она не может отправялять следующую строку программе C.

Программа B пишет в стандартный вывод количество отправленных символов программой А и количество принятых символов программой C. Данную информацию программа B получает от программ A и C 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 B 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 B и ждет ответ "OK".

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

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

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

**Листинг A.cpp**

#include <iostream>

#include <string>

#include <unistd.h>

#include "zmq\_functions.h"

const std::string ADRESS\_C = "tcp://127.0.0.1:4040";

const std::string ADRESS\_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(ADRESS\_B);

C.connect(ADRESS\_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 B.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(ADRESS\_A);

C.unbind(ADRESS\_C);

A.close();

C.close();

return 0;

}

**Листинг С.cpp**

#include <iostream>

#include <string>

#include "zmq\_functions.h"

const std::string ADRESS\_A = "tcp://127.0.0.1:4040";

const std::string ADRESS\_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(ADRESS\_A);

B.connect(ADRESS\_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:**

znako@znako-VirtualBox:~/UtoW/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.