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: Ивенкова Л.В.

Группа: М80 – 208Б-19

Преподаватель: Миронов Е. С.

Дата: \_\_\_\_\_\_\_\_\_\_\_

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**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 В. Отправка строк должна производится построчно. Программа В печатает в стандартый вывод, полученную строку от программы A. После получения программа В отправляет программе А сообщение о том, что строка получена. До тех пор, пока программа А не примет «сообщение о получение строки» от программы В, она не может отправялять следующую строку программе В.

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

**Описание программы**

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

Для взаимодействия программ необходимо по отдельности запустить программы A, B и C. Порядок запуска не имеет значения. 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 "Good".

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

В принимает сообщение от A (размер сообщения, которое А передало C) и выводит на экран и отправляет «Good». И так же принимает сообщение от C (размер сообщения, которое C получило от А), печатает на экран и отправляет «Good» обратно. Если получил «end», 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).

**Набор тестов:**

**Программа А**

[parsifal@DESKTOP-3G70RV4:~/OS/CP$ ./A

CTYCbv1c!

Tuu

mrya

**Программа В**

parsifal@DESKTOP-3G70RV4:~/OS/CP$ ./B

A: 9

C: 9

A: 3

C: 3

A: 6

C: 6

**Программа С**

parsifal@DESKTOP-3G70RV4:~/OS/CP$ ./C

CTYCbv1c!

Tuu

Mrya

**Листинг A.cpp**

#include <iostream>

#include "myzmq.h"

#include <string>

#define ADDRESS\_C "tcp://127.0.0.1:5555"

#define ADDRESS\_B "tcp://127.0.0.1:5556"

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(ADDRESS\_B);

C.connect(ADDRESS\_C);

std::string message, answer;

while(std::getline(std::cin, str)){

message = str;

send\_message(C, message);

std::string size = std::to\_string(message.size());

answer = receive\_message(C);

if(answer != "String received"){

break;

}

send\_message(B, size);

answer = receive\_message(B);

if(answer != "Good"){

break;

}

}

send\_message(C, "close$");

send\_message(B, "end");

C.disconnect(ADDRESS\_C);

B.disconnect(ADDRESS\_B);

C.close();

B.close();

return 0;

}

**Листинг B.cpp**

#include <iostream>

#include "myzmq.h"

#include <string>

#define ADDRESS\_A "tcp://127.0.0.1:5556"

#define ADDRESS\_C "tcp://127.0.0.1:5557"

int main(){

zmq::context\_t context;

std::string str;

zmq::socket\_t A(context,ZMQ\_REP);

zmq::socket\_t C(context,ZMQ\_REP);

A.bind(ADDRESS\_A);

C.bind(ADDRESS\_C);

std::string message;

while(1){

message = receive\_message(A);

if(message == "end"){

break;

}

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

send\_message(A, "Good");

message = receive\_message(C);

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

send\_message(C, "Good");

std::cout << std::endl;

}

C.unbind(ADDRESS\_C);

A.unbind(ADDRESS\_A);

A.close();

C.close();

return 0;

}

**Листинг С.cpp**

#include <iostream>

#include "myzmq.h"

#include <string>

#define ADDRESS\_A "tcp://127.0.0.1:5555"

#define ADDRESS\_B "tcp://127.0.0.1:5557"

int main(){

zmq::context\_t context;

std::string str;

zmq::socket\_t B(context,ZMQ\_REQ);

zmq::socket\_t A(context,ZMQ\_REP);

B.connect(ADDRESS\_B);

A.bind(ADDRESS\_A);

std::string message, size, answer;

while(1){

message = receive\_message(A);

if(message == "close$"){

break;

}

std::cout << message << std::endl;

send\_message(A, "String received");

size = std::to\_string(message.size());

send\_message(B, size);

answer = receive\_message(B);

if(answer != "Good"){

break;

}

}

B.disconnect(ADDRESS\_B);

A.unbind(ADDRESS\_A);

A.close();

B.close();

return 0;

}

**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.