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 A сообщение о том, что строка получена. До тех пор, пока программа A 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 A 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 A подключается к узлам B и C, после получает на вход строки, каждую полученную строку она сразу же отправляет размер отправляемой строки в C и ждет ответа «String received», дальше отправляет сообщение с длиной строки в B и ждет ответ "OK".

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

B принимает сообщение от A (размер сообщения, которое A передало C) и выводит на экран, и отправляет «OK». И так же принимает сообщение от C (размер сообщения, которое C получило от A), 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 C.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
```

nooooooorm/////\\\

nooooooorm/////\\\

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.