

Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной
математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу «Операционные системы»

Студент: П. Ф. Гришин
Преподаватель: Е. С. Миронов
Группа: М8О-201Б-21
Дата:
Оценка:
Подпись:

Москва, 2023

1 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

2 Сведения о программе

Программы написаны на языке C++ для Unix подобной операционной системы на базе ядра Linux. Для связи между программ используют сокет с помощью библиотеки ZeroMQ.

Для работы запустите все три программы. Программа А считывает строки у пользователя пока пользователь не завершит ввод, после чего передает строки программе С. Та в свою очередь выводит их на экран. Обе программы отправляют длины отправленных и полученных строк программе В. Она выводит их на экран, по этим данным можно проверить не потерялись ли данные при отправке.

3 Общий метод и алгоритм решения

Программа А подключается к программе В и С. После чего она считывает строки от пользователя и записывает их в вектор. При нажатии CTRL+D пользователь сигнализирует о конце ввода. Далее мы ждем подключение от программы С и начинаем передавать строки. В цикле передаем программе С строку и ждем сообщение об удачной передаче строки. После чего мы передаем программе В длину строки и также ждем ответа об удачной передаче. При завершении работы программа А передает сообщения программам В и С, чтобы они также завершили работу.

Программа С отправляет сообщение об удачном подключении к программе А, после чего попадает в бесконечный цикл. Там мы считываем сообщение от программы А, отправляем сообщение об успешном принятии сообщения и выводим сообщение на экран. Далее мы также отправляем программе сообщение с длиной принятой строки

и ждем от нее ответа об удачной передаче и повторяем цикл. Если мы приняли сообщение о завершении работы программы А, то мы входим из цикла и сами завершаем работу.

Программа В принимает длины строк от программы А и С и выводит их на экран, после чего повторяет цикл. При получении сообщения о завершении работы программы А, входит из цикла и сами завершает работу.

4 Листинг программы

A.cpp

```
1  #include <iostream>
2  #include <string>
3  #include <vector>
4  #include <zmq.hpp>
5
6  #define ADDRESS_C "tcp://127.0.0.1:5555"
7  #define ADDRESS_B "tcp://127.0.0.1:5556"
8
9  int main(){
10     zmq::context_t context(1);
11     zmq::socket_t to_c(context,ZMQ_REP);
12     zmq::socket_t to_b(context,ZMQ_REQ);
13
14     to_c.bind(ADDRESS_C);
15     to_b.connect(ADDRESS_B);
16
17     std::string str;
18     std::vector<std::string> all_strings;
19     while(std::getline(std::cin,str)){
20         all_strings.push_back(str);
21     }
22
23     std::string ans;
24     zmq::message_t message;
25     to_c.recv(message);
26     ans = std::string(static_cast<char*>(message.data()), message.size());
27     if(ans != "Connect")
28         return 1;
29
30     for(auto& string: all_strings){
31         message = zmq::message_t(string.size());
32         memcpy(message.data(), string.c_str(), string.size());
33         to_c.send(message);
34         to_c.recv(message);
35         ans = std::string(static_cast<char*>(message.data()), message.size());
36         if(ans != "Success")
```

```

37         break;
38         std::string size = std::to_string(string.size());
39         message = zmq::message_t(size.size());
40         memcpy(message.data(), size.c_str(), size.size());
41         to_b.send(message);
42         to_b.recv(message);
43         ans = std::string(static_cast<char*>(message.data()), message.size());
44         if(ans != "OK")
45             break;
46     }
47     ans = "$$$";
48     message = zmq::message_t(ans.size());
49     memcpy(message.data(), ans.c_str(), ans.size());
50     to_c.send(message);
51
52     ans = "-1";
53     message = zmq::message_t(ans.size());
54     memcpy(message.data(), ans.c_str(), ans.size());
55     to_b.send(message);
56
57     to_c.unbind(ADDRESS_C);
58     to_b.disconnect(ADDRESS_B);
59
60     return 0;
61 }

```

C.cpp

```

1  #include <iostream>
2  #include <string>
3  #include <zmq.hpp>
4
5  #define ADDRESS_A "tcp://127.0.0.1:5556"
6  #define ADDRESS_C "tcp://127.0.0.1:5557"
7
8  int main(){
9      zmq::context_t context(1);
10     zmq::socket_t to_a(context,ZMQ_REP);
11     zmq::socket_t to_c(context,ZMQ_REP);
12     std::string my_ans = "OK";
13     to_a.bind(ADDRESS_A);
14     to_c.bind(ADDRESS_C);
15     while(true){
16         zmq::message_t message_a;
17         to_a.recv(message_a);
18         std::string ans_a = std::string(static_cast<char*>(message_a.data()), message_a
            .size());
19         int size_str_a = std::stoi(ans_a);
20         if(size_str_a == -1)
21             break;

```

```

22     message_a = zmq::message_t(my_ans.size());
23     memcpy(message_a.data(), my_ans.c_str(), my_ans.size());
24     to_a.send(message_a);
25
26     zmq::message_t message_c;
27     to_c.recv(message_a);
28     std::string ans_c = std::string(static_cast<char*>(message_a.data()), message_a
29         .size());
30     int size_str_c = std::stoi(ans_c);
31     message_c = zmq::message_t(my_ans.size());
32     memcpy(message_c.data(), my_ans.c_str(), my_ans.size());
33     to_c.send(message_c);
34
35     std::cout << "Size str send A: " << size_str_a << "\tSize str send C: " <<
36         size_str_c << std::endl;
37 }
38 to_a.unbind(ADDRESS_A);
39 to_c.unbind(ADDRESS_C);
40 return 0;
41 }

```

B.cpp

```

1  #include <iostream>
2  #include <string>
3  #include <zmq.hpp>
4
5  #define ADDRESS_A "tcp://127.0.0.1:5556"
6  #define ADDRESS_C "tcp://127.0.0.1:5557"
7
8  int main(){
9      zmq::context_t context(1);
10     zmq::socket_t to_a(context,ZMQ_REP);
11     zmq::socket_t to_c(context,ZMQ_REP);
12     std::string my_ans = "OK";
13     to_a.bind(ADDRESS_A);
14     to_c.bind(ADDRESS_C);
15     while(true){
16         zmq::message_t message_a;
17         to_a.recv(message_a);
18         std::string ans_a = std::string(static_cast<char*>(message_a.data()), message_a
19             .size());
20         int size_str_a = std::stoi(ans_a);
21         if(size_str_a == -1)
22             break;
23         message_a = zmq::message_t(my_ans.size());
24         memcpy(message_a.data(), my_ans.c_str(), my_ans.size());
25         to_a.send(message_a);

```

```

26     zmq::message_t message_c;
27     to_c.recv(message_a);
28     std::string ans_c = std::string(static_cast<char*>(message_a.data()), message_a
    .size());
29     int size_str_c = std::stoi(ans_c);
30     message_c = zmq::message_t(my_ans.size());
31     memcpy(message_c.data(), my_ans.c_str(), my_ans.size());
32     to_c.send(message_c);
33
34     std::cout << "Size str send A: " << size_str_a << "\tSize str send C: " <<
    size_str_c << std::endl;
35
36 }
37 to_a.unbind(ADDRESS_A);
38 to_c.unbind(ADDRESS_C);
39 return 0;
40 }

```

5 Демонстрация работы программ

Программа A

```

gpavel@gpavel-HP-Pavilion-Gaming-Laptop-17-cd1xxx:~/Desktop/OS/KP$ ./A
hello world
check
test
I'm working

```

Программа C

```

gpavel@gpavel-HP-Pavilion-Gaming-Laptop-17-cd1xxx:~/Desktop/OS/KP$ ./C
hello world
check
test
I'm working

```

Программа B

```

gpavel@gpavel-HP-Pavilion-Gaming-Laptop-17-cd1xxx:~/Desktop/OS/KP$ ./B
Size str send A: 11      Size str send C: 11
Size str send A: 5      Size str send C: 5
Size str send A: 4      Size str send C: 4
Size str send A: 11     Size str send C: 11

```

6 Вывод

Данная курсовая работа основывается на знаниях полученных в ходе изучения курса. По итогу мы получили нейколько 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.