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. А. Мохляков
Преподаватель: Е. С. Миронов
Группа: М8О-208Б-19

Дата:
Оценка:
Подпись:

Москва, 2021

1 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 принятыхсимволов программой C. Данную информацию программа B получает от программ A и C соответственно

2 Сведения о программе

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

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

3 Общий метод и алгоритм решения

Программа A подключается к программе B и C. 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 C и начинаем передавать строки. В цикле передаем программе C троку и ждем сообщение об удачной передаче строки. После чего мы передаем программе B длину строки и также ждем ответа об удачной передаче. При завершеении работы программа A передает сообщения программам B и C, чтобы они также завершили работу.

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

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

Программа B принимает длины строк от программы A и C и выводит их на экран, после чего повторяет цикл. При пололучении сообщения о завершении работы программы A, входит из цикла и сами завершает работу.

4 Листинг программы

A.cpp

```
#include <iostream>
#include <string>
#include <vector>
#include <zmq.hpp>

#define ADDRESS_C "tcp://127.0.0.1:5555"
#define ADDRESS_B "tcp://127.0.0.1:5556"

int main(){
    zmq::context_t context(1);
    zmq::socket_t to_c(context,ZMQ_REP);
    zmq::socket_t to_b(context,ZMQ_REQ);

    to_c.bind(ADDRESS_C);
    to_b.connect(ADDRESS_B);

    std::string str;
    std::vector<std::string> all_strings;
    while(std::getline(std::cin,str)){
        all_strings.push_back(str);
    }

    std::string ans;
    zmq::message_t message;
    to_c.recv(message);
    ans = std::string(static_cast<char*>(message.data()), message.size());
    if(ans != "Connect")
        return 1;

    for(auto& string: all_strings){
        message = zmq::message_t(string.size());
        memcpy(message.data(), string.c_str(), string.size());
        to_c.send(message);
        to_c.recv(message);
        ans = std::string(static_cast<char*>(message.data()), message.size());
        if(ans != "Success")
```

```
            break;
        std::string size = std::to_string(string.size());
        message = zmq::message_t(size.size());
        memcpy(message.data(), size.c_str(), size.size());
        to_b.send(message);
        to_b.recv(message);
        ans = std::string(static_cast<char*>(message.data()), message.size());
        if(ans != "OK")
            break;
    }
    ans = "$$$";
    message = zmq::message_t(ans.size());
    memcpy(message.data(), ans.c_str(), ans.size());
    to_c.send(message);

    ans = "-1";
    message = zmq::message_t(ans.size());
    memcpy(message.data(), ans.c_str(), ans.size());
    to_b.send(message);

    to_c.unbind(ADDRESS_C);
    to_b.disconnect(ADDRESS_B);

    return 0;
}
```

C.cpp

```
#include <iostream>
#include <string>
#include <zmq.hpp>

#define ADDRESS_A "tcp://127.0.0.1:5556"
#define ADDRESS_C "tcp://127.0.0.1:5557"

int main(){
    zmq::context_t context(1);
    zmq::socket_t to_a(context,ZMQ_REP);
    zmq::socket_t to_c(context,ZMQ_REP);
    std::string my_ans = "OK";
    to_a.bind(ADDRESS_A);
    to_c.bind(ADDRESS_C);
    while(true){
        zmq::message_t message_a;
        to_a.recv(message_a);
        std::string ans_a = std::string(static_cast<char*>(message_a.data()), message_a
            .size());
        int size_str_a = std::stoi(ans_a);
        if(size_str_a == -1)
            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
                .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  }
```

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
                .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
                .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

```
pavel@DESKTOP-K5KMLPV:~/Project/mai/2_course/OS/KP$ ./A
hello world
check
test
I'm working
```

Программа C

```
pavel@DESKTOP-K5KMLPV:~/Project/mai/2_course/OS/KP$ ./C
hello world
check
test
I'm working
```

Программа B

```
pavel@DESKTOP-K5KMLPV:~/Project/mai/2_course/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 крусовая 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 обощает и закрепляет полученные в курсе знания.