Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Курсовая работа по курсу

«Операционные системы»

Тема работы

Использование знаний и навыков, полученных в течение курса

Студент: Ханнанов Руслан Маратович
Группа: М8О-208Б-20
Вариант: 20
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

Содержание

Репозиторий

https://github.com/Naksen/OS

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 стандартный 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 отправлять 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.

Общие сведения о программе

Предварительно реализованы три программы:

A.cpp

B.cpp

C.cpp

Между которыми будет происходить взаимодействие и обмен сообщениями в соответствии с заданием.

Общий метод и алгоритм решения

Makefile:

```
all:
    g++ A.cpp -lzmq -o A -Wall -pedantic
    g++ B.cpp -lzmq -o B -Wall -pedantic
    g++ C.cpp -lzmq -o C -Wall -pedantic
clean:
    rm -rf A B C
```

Общение между процессами я решил реализовать с помощью ZeroMQ. В программе используется тип соединения Request-Response. Программа A принимает сообщения из стандартного потока ввода и посылает их программе C. Программа C принимает сообщение, печатает его и пересылает в программу B, после чего присылает буфферное сообщение назад в программу A о том, что она получила новую строчку. После этого программа A отсылает программе B размер сообщения (количество символов), которое было отправлено C. Программа B выводит размер сообщения, полученного от C и размер сообщения отосланного от A. Если всё работает корректно, то эти размеры должны быть одинаковыми.

Исходный код

A.cpp:

```
#include <unistd.h>
#include <sstream>
#include <set>
#include <string>
#include <iostream>

#include <zmq.hpp>

using namespace std;

int main(){
    const string endpoint = "tcp://localhost:5555";

    zmq::context_t context;

    zmq::socket_type type = zmq::socket_type::req;
    zmq::socket_t socket (context, type);
    cout << "Please, enter the string" << endl;
```

```
socket.connect(endpoint);

//TO B
const string to_b = "tcp://localhost:5554";
zmq::socket_type b_type = zmq::socket_type::req;
zmq::socket_t b_socket (context, b_type);
b_socket.connect(to_b);

pid_t C = fork();
if (C == -1) {
    perror("fork");
    return -1;
}
if (C == 0) {
    pid_t B = fork();
    if (B == -1) {
        perror("fork");
        return -1;
    }
    if (B == 0){
        execl("./B", "./B",NULL);
    } else {
        execl("./C", "./C",NULL);
    }
}

string s;
while(cin >> s) {

    // Sending to C
    zmq::message_t reply;
    zmq::message_t message(s.size());
    memcpy(message.data(), s.c_str(), s.size());
    socket.send(message);

    if (s == "exit") {
        socket.disconnect(endpoint);
        b_socket.disconnect(to_b);
        return 0;
    }
```

```
        // Receive from C
        socket.recv(reply);
        std::string
received_msg(static_cast<char*>(reply.data()), reply.size());

        //Sending to B
        string c = "TO B";
        int cnt = s.size();
        string cnt_s = to_string(cnt);
        zmq::message_t message_to_b(cnt_s.size());
        memcpy(message_to_b.data(), cnt_s.c_str(),
cnt_s.size());
        b_socket.send(message_to_b);

        //Get from B
        zmq::message_t rep;
        b_socket.recv(rep);
    }
}
```

B.cpp:

```
#include <unistd.h>
#include <sstream>
#include <string>
#include <zmq.hpp>
#include <iostream>

using namespace std;

int main(){
    const string to_c = "tcp://*:5554";

    zmq::context_t context;
    zmq::socket_type type = zmq::socket_type::rep;
    zmq::socket_t socket (context, type);
    socket.bind(to_c);
```

```
    while (1) {

        // Receive the message from C
        zmq::message_t message;
        socket.recv(message);

        std::string
received_msg(static_cast<char*>(message.data()),
message.size());
        if (received_msg == "exit") {
            socket.unbind(to_c);
            exit(0);
        }
        cout << "B: size of message from C:" <<
received_msg.size() << endl;

        // Send to C
        sleep(1);
        string answer = "b: Get new string";
        zmq::message_t ans(answer.size());
        memcpy(ans.data(), answer.c_str(), answer.size());
        socket.send(ans);

        // Get message from A
        zmq::message_t message_from_a;
        socket.recv(message_from_a);
        std::string
cnt(static_cast<char*>(message_from_a.data()),
message_from_a.size());
        cout << "B: size of message from A:" << cnt << endl;

        // Send message to A
        string z = "get cnt from a";
        zmq::message_t buf(z.size());
        memcpy(buf.data(), z.c_str(), z.size());
        socket.send(buf);

    }
    return 0;
}
```

C.cpp:

```
#include <unistd.h>
#include <sstream>
#include <string>
#include <zmq.hpp>
#include <iostream>

using namespace std;

int main(int argc, char* argv[]) {

    const string endpoint = "tcp://*:5555";

    // FROM A
    zmq::context_t context;
    zmq::socket_type type = zmq::socket_type::rep;
    zmq::socket_t socket (context, type);
    socket.bind(endpoint);

    // TO B
    const string to_b = "tcp://localhost:5554";
    zmq::socket_type b_type = zmq::socket_type::req;
    zmq::socket_t b_socket (context, b_type);
    b_socket.connect(to_b);

    std::string message;
    while ( 1) {
        // Get message from A

        zmq::message_t message;
        socket.recv(message);
        std::string
received_msg(static_cast<char*>(message.data()),
message.size());
        if (received_msg == "exit") {
            b_socket.send(message);
            socket.unbind(endpoint);
            socket.disconnect(to_b);
            exit(0);
```

```
            }
            cout << "C: " << received_msg << endl;

            //Sending message to A
            string answer = "Get new string";
            zmq::message_t ans(answer.size());
            memcpy(ans.data(), answer.c_str(), answer.size());
            //cout << "C: Sending to A..." << endl;
            socket.send(ans);

            //Sending message to B
            b_socket.send(message);

            zmq::message_t ans_from_b;

            //Receiving message from B
            b_socket.recv(ans_from_b);

        }
}
```

Демонстрация работы программы

```
naksan@LAPTOP-TL9L61MA:~/cprog/cp_os$ ./A
Please, enter the string
ddd
C: ddd
B: size of message from C:3
B: size of message from A:3
ccccccccc
C: ccccccccc
B: size of message from C:10
B: size of message from A:10
l
C: l
B: size of message from C:1
B: size of message from A:1
exit
```

Выводы

Данная лабораторная работа понравилась мне тем, что передо мной стояла задача, а способ её выполнения можно было выбрать уже самому. В процессе работы я долго не мог решить, на чём остановиться, но в итоге выбрал

ZeroMQ, так как появилось желание получше в нём разобраться и использовать что-то более сложное, чем pipe или memory mmap. В итоге я улучшил своё понимания работы с ZeroMQ и успешно реализовал поставленную задачу.