Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8: информационные технологии и прикладная математика

Кафедра вычислительной математики и программирования

**Курсовая работа по курсу**

**«Операционные системы»**

|  |  |
| --- | --- |
| Студент: | Паленов П.С. |
| Группа: | М8О-301Б-18 |
| Преподаватель: | Миронов Е.С. |
| Оценка: |  |
| Дата: |  |

Москва 2020

**Содержание**

1. Постановка задачи
2. Общий метод и алгоритм решения
3. Файлы программы
4. Демонстрация работы программы
5. Вывод
6. **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 С. Отправка строк должна производится построчно. Программа C печатает в стандартный вывод, полученную строку от программы A. После получения программа C отправляет программе А сообщение о том, что строка получена. До тех пор, пока программа А не примет «сообщение о получение строки» от программы С, она не может отправлять следующую строку программе С.

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

1. **Общий метод и алгоритм решения**

Организация межпроцессорного взаимодействия производится с помощью средств библиотеки ZeroMQ. Все три программы попарно используют связь типа сервер-клиент (REP-REQ). Для общения между программами используется структура *Message*. Определены функции отправки и приема сообщений *send\_message* и *receive\_message* соответственно. Так же отдельно определена функция отправки и приема строки *send\_text* и *receive\_text*. Это нужно для корректной отправки *string-строки.* Чтобы закончить ввод текста в программе A, требуется ввести слово END на отдельной строке.

1. **Файлы программы**

**ABC.h**

#pragma once

#include <string.h>

#include <stdio.h>

#include <iostream>

#include "zmq.h"

using namespace std;

typedef struct {

int sym;

int task;

int status;

} Message;

void print\_menu() {

cout << "1 - enter text" << endl

<< "2 - send text" << endl

<< "0 - exit" << endl;

}

void send\_message(void\* socket, Message\* out) {

zmq\_msg\_t msg;

zmq\_msg\_init\_size(&msg, sizeof(Message));

memcpy(zmq\_msg\_data(&msg), out, sizeof(Message));

zmq\_msg\_send(&msg, socket, 0);

zmq\_msg\_close(&msg);

}

void send\_text(void\* socket, string\* text) {

zmq\_msg\_t msg;

zmq\_msg\_init\_size(&msg, text->size());

memcpy(zmq\_msg\_data(&msg), text->c\_str(), text->size());

zmq\_msg\_send(&msg, socket, 0);

zmq\_msg\_close(&msg);

}

string receive\_text(void\* socket, zmq\_msg\_t\* msg) {

string text;

zmq\_msg\_init(msg);

zmq\_msg\_recv(msg, socket, 0);

char\* ptr = static\_cast<char\*>(zmq\_msg\_data(msg));

return string(ptr, zmq\_msg\_size(msg));

}

Message\* receive\_message(void\* socket, zmq\_msg\_t\* msg) {

Message \*in;

zmq\_msg\_init(msg);

zmq\_msg\_recv(msg, socket, 0);

in = static\_cast<Message\*>(zmq\_msg\_data(msg));

return in;

}

Message init\_message(){

Message mes;

mes.sym = -1;

mes.task = -1;

mes.status = -1;

return mes;

}

**A.cpp**

#include <vector>

#include "ABC.h"

int main() {

Message AtoB, \*BtoA, AtoC, \*CtoA;

zmq\_msg\_t msg, out;

vector<string> text;

string s;

int comand;

AtoB = init\_message();

AtoC = init\_message();

void\* context = zmq\_ctx\_new();

void\* psocket = zmq\_socket(context, ZMQ\_REQ);

void\* mysocket = zmq\_socket(context, ZMQ\_REP);

zmq\_bind(mysocket, "tcp://\*:5001");

zmq\_connect(psocket, "tcp://localhost:5002");

AtoB.status = 1;

send\_message(psocket, &AtoB);

CtoA = receive\_message(mysocket, &msg);

if (CtoA->status == 1) cout << "A: C connected\n";

else {

cout << "Error: unsuccessful connection C to A";

return -1;

}

zmq\_msg\_close(&msg);

BtoA = receive\_message(psocket, &msg);

zmq\_msg\_close(&msg);

print\_menu();

cout << "Enter comand:\n";

bool loop = true;

while(loop) {

cout << "=> ";

cin >> comand;

switch (comand) {

case 1:

printf("Enter text, type END to finish:\n");

cin.ignore(1);

do{

cout << "> ";

getline(cin, s, '\n');

if (s != "END") {

text.push\_back(s);

}

} while (s != "END");

break;

case 2:

for (const auto& item : text) {

AtoC.task = 1;

send\_message(mysocket, &AtoC);

CtoA = receive\_message(mysocket, &msg);

zmq\_msg\_close(&msg);

s = item;

send\_text(mysocket, &s);

AtoB.sym = item.length();

AtoB.task = 1;

send\_message(psocket, &AtoB);

BtoA = receive\_message(psocket, &msg);

if (BtoA->status == 2) {

zmq\_msg\_close(&msg);

CtoA = receive\_message(mysocket, &msg);

if (CtoA->status != 2) {

zmq\_msg\_close(&msg);

cout << "Error while sending string" << endl;

return -1;

}

} else {

cout << "Error: missing message from B" << endl;

return -1;

}

}

text.clear();

break;

case 0:

AtoC.task = 2;

AtoB.task = 2;

send\_message(mysocket, &AtoC);

send\_message(psocket, &AtoB);

loop = false;

break;

}

}

zmq\_close(psocket);

zmq\_close(mysocket);

zmq\_ctx\_destroy(context);

return 0;

}

**B.cpp**

#include "ABC.h"

string adr = "tcp://localhost:";

int main(){

Message BtoC, \*CtoB, BtoA, \*AtoB;

zmq\_msg\_t msg;

BtoA = init\_message();

BtoC = init\_message();

void\* context = zmq\_ctx\_new();

void\* psocket = zmq\_socket(context, ZMQ\_REQ);

void\* mysocket = zmq\_socket(context, ZMQ\_REP);

zmq\_bind(mysocket, "tcp://\*:5002");

zmq\_connect(psocket, "tcp://localhost:5003");

BtoC.status = 1;

send\_message(psocket, &BtoC);

AtoB = receive\_message(mysocket, &msg);

if (AtoB->status == 1) cout << "B: A connected\n";

else {

cout << "Error: unsuccessful connection A to B";

return -1;

}

zmq\_msg\_close(&msg);

BtoA.status = 1;

send\_message(mysocket, &BtoA);

bool loop = true;

while(loop){

AtoB = receive\_message(mysocket, &msg);

switch(AtoB->task){

case 1:

cout << "---------------" << endl;

cout << "A sent " << AtoB->sym << " symbols" << endl;

zmq\_msg\_close(&msg);

BtoA.status = 2;

send\_message(mysocket, &BtoA);

CtoB = receive\_message(psocket, &msg);

if(CtoB->task == 1){

cout << "C recieved " << CtoB->sym << " symbols" << endl;

cout << "---------------" << endl << endl;

zmq\_msg\_close(&msg);

BtoC.status = 2;

send\_message(psocket, &BtoC);

} else {

cout << "Error: missing message from C";

return -1;

}

break;

case 2:

loop = false;

break;

}

}

zmq\_close(psocket);

zmq\_close(mysocket);

zmq\_ctx\_destroy(context);

return 0;

}

**C.cpp**

#include "ABC.h"

string adr = "tcp://localhost:";

int main() {

Message CtoA, \*AtoC, CtoB, \*BtoC;

string text;

zmq\_msg\_t msg;

CtoA = init\_message();

CtoB = init\_message();

void\* context = zmq\_ctx\_new();

void\* psocket = zmq\_socket(context, ZMQ\_REQ);

void\* mysocket = zmq\_socket(context, ZMQ\_REP);

zmq\_bind(mysocket, "tcp://\*:5003");

zmq\_connect(psocket, "tcp://localhost:5001");

CtoA.status = 1;

send\_message(psocket, &CtoA);

BtoC = receive\_message(mysocket, &msg);

if (BtoC->status == 1) cout << "C: B connected\n";

else {

cout << "Error: unsuccessful connection B to C";

return -1;

}

zmq\_msg\_close(&msg);

bool loop = true;

while(loop){

AtoC = receive\_message(psocket, &msg);

switch(AtoC->task){

case 1:

{

zmq\_msg\_close(&msg);

send\_message(psocket, &CtoA);

string recvd = receive\_text(psocket, &msg);

CtoB.sym = recvd.length();

CtoB.task = 1;

zmq\_msg\_close(&msg);

send\_message(mysocket, &CtoB);

BtoC = receive\_message(mysocket, &msg);

cout << "Recieved string: " << recvd << endl;

if(BtoC->status == 2){

zmq\_msg\_close(&msg);

CtoA.status = 2;

send\_message(psocket, &CtoA);

}

break;

}

case 2:

loop = false;

break;

}

}

zmq\_close(psocket);

zmq\_close(mysocket);

zmq\_ctx\_destroy(context);

return 0;

}

1. **Демонстрация работы программы**

**A.out**

justp@Pavel:~$ ./A.out

A: C connected

1 - enter text

2 - send text

0 - exit

Enter comand:

=> 1

Enter text, type END to finish:

> first line 111111111

> second line 222

> 33333 enil driht

> END

=> 2

=>

**B.out**

justp@Pavel:~$ ./B.out

B: A connected

---------------

A sent 20 symbols

C recieved 20 symbols

---------------

---------------

A sent 15 symbols

C recieved 15 symbols

---------------

---------------

A sent 16 symbols

C recieved 16 symbols

---------------

**C.out**

justp@Pavel:~$ ./C.out

C: B connected

Recieved string: first line 111111111

Recieved string: second line 222

Recieved string: 33333 enil driht

1. **Вывод**

С помощью библиотеки ZeroMQ удобно организовывать межпроцессорные связи. ZMQ-сокеты позволяют легко производить обмен сообщениями. Единственное неудобство, хотя и не особо большое, использованной мной связи REP-REQ – это необходимость чередования запроса и ответа как на стороне сервера, так и клиента. Это в некоторых случаях вынуждает делать лишние отправки, что влияет на быстродействие программы.