Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовая работа по курсу

«Операционные системы»

Вариант на «удовлетворительно»

Студент: [ Нелюбин В.С ]

Группа: М8О–206Б–19

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2021.

Постановка задачи

**Цель работы**

* Приобретение практических навыков в использовании знаний, полученных в течении курса
* Проведение исследования в выбранной предметной области

## Задание

Необходимо спроектировать и реализовать программный прототип в соответствии с выбранным вариантом. Произвести анализ и сделать вывод на основании данных, полученных при работе программно прототипа. 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 стандартый 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 отправялять 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. Способ организация межпроцессорного взаимодействия выбирает студент. Выбран ZeroMQ и TCP.

**Общие сведения о программе**

Три программы. a.cpp, b.cpp c.cpp. Работа начинается с запуска В. В открывает разъем приёма данных о передаваемой информации, и запускает процесс А. А открывает разъем обмена данными с процессом С, запускает процесс С, и начинает считывать информацию со стандартного ввода. С подключается к открытым TCP разъемам. Получив строку, А отправляет её процессу С, и отправляет В количество считанных байтов. Получив информацию от А, С выводит её на стандартный вывод, и так же отвравляет количество полученных байтов В. Число байтов получается из функции fgets() в А, а в С – из параметра size() класса zmq::message\_t

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Повторить работу с ZMQ из 6-8 лабораторных.

**Основные файлы программы**

**a.cpp:**

#include <iostream>

#include <chrono>

#include <thread>

#include <string>

#include <map>

#include <vector>

#include <cstring>

#ifndef \_WIN32

#include <unistd.h>

#else

#include <windows.h>

#define sleep(n) Sleep(n)

#endif

#include <zmq.hpp>

const size\_t TCPE=32;

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

char\*NodeExec=new char(10);

for(int i=0;i<10;i++){

NodeExec[i]="./c.out\0"[i];

}

int num=255,i,j;

zmq::context\_t context (1);

zmq::socket\_t toC (context,ZMQ\_REQ);

zmq::socket\_t toB (context,ZMQ\_PUSH);

toB.connect(argv[1]);

toC.bind("tcp://\*:\*");//\*/

char\*adresC=new char[TCPE];

size\_t tmpsizeN=TCPE\*sizeof(char);

if(zmq\_getsockopt(toC,ZMQ\_LAST\_ENDPOINT,adresC,&tmpsizeN)){

toB.close();

toC.close();

delete[] adresC;

printf("socket T ID failure1\n");

return -1;

}

pid\_t pidN;

pidN=fork();

if(pidN==0){

char\*args[]={NodeExec,adresC,argv[1],NULL};

execvp(args[0],args);

\_exit (EXIT\_FAILURE);

}

else if(pidN<0){

printf("something went wrong with fork()\n");

toC.close();

delete[] adresC;

return -1;

}

char\*inpt=new char[num];

while(NULL!=fgets (inpt, num, stdin)){

zmq::message\_t request(num);

memcpy(request.data (), inpt, num);

int timp;

for(timp=0;((char\*)request.data())[timp]!='\0';timp++);

std::string temp="A sent "+std::to\_string(timp)+" symbols\0";//reply.size())+"\0";

zmq::message\_t request2 (temp.size()+1);

memcpy (request2.data (), temp.c\_str(), temp.size()+1);

toB.send(request2,zmq::send\_flags::none);

((char\*)request.data())[request.size()]='\0';

toC.send(request,zmq::send\_flags::none);

zmq::message\_t reply;

toC.recv(reply,zmq::recv\_flags::none);

((char\*)reply.data())[reply.size()]='\0';

printf("A: %s\n",(char\*)reply.data());

}

zmq::message\_t fnmsg(4);

memcpy (fnmsg.data (), "\nN\0", 3);

((char\*)fnmsg.data())[fnmsg.size()]='\0';

toB.send(fnmsg,zmq::send\_flags::none);

zmq::message\_t fnmsg2(4);

memcpy (fnmsg2.data (), "\nN\0", 3);

((char\*)fnmsg2.data())[fnmsg2.size()]='\0';

toC.send(fnmsg2,zmq::send\_flags::none);

printf("A ended\n");

toB.close();

toC.close();

context.close();

return 0;

}

**b.cpp:**

#include <iostream>

#include <thread>

#include <time.h>

#include <string>

#include <vector>

#include <map>

#include <cstring>

#ifndef \_WIN32

#include <unistd.h>

#else

#include <windows.h>

#define sleep(n) Sleep(n)

#endif

#include <zmq.hpp>

const size\_t TCPE=32;

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

char\*NodeExec=new char(10);

for(int i=0;i<10;i++){

NodeExec[i]="./a.out\0"[i];

}

zmq::context\_t context (1);

zmq::socket\_t sci (context, ZMQ\_PULL);

sci.bind("tcp://\*:\*");//\*/

char\*adresC=new char[TCPE];

size\_t tmpsizeN=TCPE\*sizeof(char);

if(zmq\_getsockopt(sci,ZMQ\_LAST\_ENDPOINT,adresC,&tmpsizeN)){

sci.close();

delete[] adresC;

printf("socket T ID failure1\n");

return -1;

}

pid\_t pidN;

pidN=fork();

if(pidN==0){

char\*args[]={NodeExec,adresC,NULL};

execvp(args[0],args);

\_exit (EXIT\_FAILURE);

}

else if(pidN<0){

printf("something went wrong with fork()\n");

sci.close();

delete[] adresC;

return -1;

}

zmq::message\_t reply;

while(1){

sci.recv(reply,zmq::recv\_flags::none);

if(((char\*)reply.data())[0]=='\n'){

break;

}

printf("B: %s\n",(char\*)reply.data());

}

printf("B ended\n");

sci.close();

context.close();

return 0;

}

**c.cpp:**

#include <iostream>

#include <thread>

#include <time.h>

#include <string>

#include <vector>

#include <map>

#include <cstring>

#ifndef \_WIN32

#include <unistd.h>

#else

#include <windows.h>

#define sleep(n) Sleep(n)

#endif

#include <zmq.hpp>

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

zmq::context\_t context (1);

zmq::socket\_t toA (context, ZMQ\_REP);

zmq::socket\_t toB (context, ZMQ\_PUSH);

toA.connect (argv[1]);

toB.connect (argv[2]);

std::string temp;

zmq::message\_t reply;

int timp;

while(1){

toA.recv(reply,zmq::recv\_flags::none);

((char\*)reply.data())[reply.size()]='\0';

/\*if(((char\*)reply.data())[0]=='N'){

break;

}\*/

printf("C: %s\n",(char\*)reply.data());

for(timp=0;((char\*)reply.data())[timp]!='\0';timp++);

temp="C reeived "+std::to\_string(timp)+" symbols\0";//reply.size())+"\0";

zmq::message\_t request (temp.size()+1);

memcpy (request.data (), temp.c\_str(), temp.size()+1);

//printf("c send to b\n");

toB.send(request,zmq::send\_flags::none);

//printf("c sent to b\n");

zmq::message\_t request2 (4);

memcpy (request2.data (), "Yes\0", 4);

toA.send(request2,zmq::send\_flags::none);

if(((char\*)reply.data())[0]=='\n'){

break;

}

}

printf("C ended\n");

toA.close();

toB.close();

context.close();

return 0;

}

**Пример работы**

aaaaa@TRASHBox:~/Desktop/OC/KP$ ./b.out

bruh

B: A sent 5 symbols

C: bruh

B: C reeived 5 symbols

A: Yes

ba na na

B: A sent 9 symbols

C: ba na na

B: C reeived 9 symbols

A: Yes

A ended

B ended

aaaaa@TRASHBox:~/Desktop/OC/KP$ C:

N

C ended

A и С завершаются позже В, и могут вывести что-то на stdout после того, как консоль выводит «aaaaa@TRASHBox:~/Desktop/OC/KP$»

**Вывод**

Скорее всего, задание КП на 3 когда-то было 6 лабораторной, и 6-8 было курсовой. Очень странно, что после достаточно сложной 6-8 лабораторной работы, можно выбрать такой простой вариант курсовой работы.