Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовой проект по курсу**

**«Операционные системы»**

Студент: Борисов Ян Артурович

Группа: М8О-208Б-20

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

<https://github.com/Yannikupy/OS>

**Постановка задачи**

Необходимо написать 3 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. Способ организация межпроцессорного взаимодействия выбирает студент.

**Общие сведения о программе:** программа состоит из четырёх файлов: A.cpp, B.cpp, C.cpp и zmq.h, в котором я реализовал функции для более удобной работы с очередью сообщений.

**Общий метод и алгоритм решения:** Сначала A считывает строку, передаёт в B количество считанных символов, а в C — количество считанных символов и саму строку посимвольно, затем B выводит количество введённых символов, C выводит строку и передаёт B количество выведенных символов, после чего B выводит количество выведенных символов и цикл начинается заново. Межпроцессорное взаимодействие основано на очереди сообщений.

**Исходный код:**

A.cpp

#include <iostream>

#include <string>

#include "zmq.h"

#define ADDRESS\_C "tcp://127.0.0.1:5555"

#define ADDRESS\_B "tcp://127.0.0.1:5556"

int main(){

zmq::context\_t context;

std::string str;

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

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

B.connect(ADDRESS\_B);

C.connect(ADDRESS\_C);

std::string message, answer;

while(std::getline(std::cin, str)){

message = str;

send\_message(C, message);

std::string size = std::to\_string(message.size());

answer = receive\_message(C);

if(answer != "String received"){

break;

}

send\_message(B, size);

answer = receive\_message(B);

if(answer != "Good"){

break;

}

}

send\_message(C, "close$");

send\_message(B, "end");

C.disconnect(ADDRESS\_C);

B.disconnect(ADDRESS\_B);

C.close();

B.close();

return 0;

}

B.cpp

#include <iostream>

#include "myzmq.h"

#include <string>

#define ADDRESS\_A "tcp://127.0.0.1:5556"

#define ADDRESS\_C "tcp://127.0.0.1:5557"

int main(){

zmq::context\_t context;

std::string str;

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

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

A.bind(ADDRESS\_A);

C.bind(ADDRESS\_C);

std::string message;

while(1){

message = receive\_message(A);

if(message == "end"){

break;

}

std::cout << "A: " << message << std::endl;

send\_message(A, "Good");

message = receive\_message(C);

std::cout << "C: " << message << std::endl;

send\_message(C, "Good");

std::cout << std::endl;

}

C.unbind(ADDRESS\_C);

A.unbind(ADDRESS\_A);

A.close();

C.close();

return 0;

}

C.cpp

#include <iostream>

#include "myzmq.h"

#include <string>

#define ADDRESS\_A "tcp://127.0.0.1:5555"

#define ADDRESS\_B "tcp://127.0.0.1:5557"

int main(){

zmq::context\_t context;

std::string str;

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

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

B.connect(ADDRESS\_B);

A.bind(ADDRESS\_A);

std::string message, size, answer;

while(1){

message = receive\_message(A);

if(message == "close$"){

break;

}

std::cout << message << std::endl;

send\_message(A, "String received");

size = std::to\_string(message.size());

send\_message(B, size);

answer = receive\_message(B);

if(answer != "Good"){

break;

}

}

B.disconnect(ADDRESS\_B);

A.unbind(ADDRESS\_A);

A.close();

B.close();

return 0;

}

Makefile

all: A B C  
  
A: A.cpp  
 g++ A.cpp -lzmq -o A  
  
B: B.cpp  
 g++ B.cpp -lzmq -o B  
  
C: C.cpp  
 g++ C.cpp -lzmq -o C  
  
clean:  
 rm A B

**Демонстрация работы программы**

yanniku@DESKTOP-US1A6DR:/mnt/c/Users/boris/CLionProjects/OCs/OS/Course\_Work$ ./A

daldkak

pfokbopgf

apwodokqwop

xoajoj

skldnvkd

iwejqi

ld

epoq

bpofokb

e;lqkeq

yanniku@DESKTOP-US1A6DR:/mnt/c/Users/boris/CLionProjects/OCs/OS/Course\_Work$ ./B

A: 7

C: 7

A: 9

C: 9

A: 11

C: 11

A: 6

C: 6

A: 8

C: 8

A: 6

C: 6

A: 3

C: 3

A: 4

C: 4

A: 7

C: 7

A: 7

C: 7

yanniku@DESKTOP-US1A6DR:/mnt/c/Users/boris/CLionProjects/OCs/OS/Course\_Work$ ./C

daldkak

pfokbopgf

apwodokqwop

xoajoj

skldnvkd

iwejqi

ld

epoq

bpofokb

e;lqkeq

**Выводы**

При написании курсового проекта я укрепил знания и навыки, полученные мной во время прохождения курса операционных систем.