

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 А, В, С. Программа А принимает из стандартного потока ввода строки, а далее их отправляет программе С. Отправка строк должна производиться построчно. Программа С печатает в стандартный вывод, полученную строку от программы А. После получения программа С отправляет программе А сообщение о том, что строка получена. До тех пор, пока программа А не примет «сообщение о получении строки» от программы С, она не может отправлять следующую строку программе С. Программа В пишет в стандартный вывод количество отправленных символов программой А и количество принятых символов программой С. Данную информацию программа В получает от программ А и С соответственно. Способ организация межпроцессорного взаимодействия выбирает студент.

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

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

Исходный код:

А.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

dalldkak

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.