Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовая работа

«Операционные системы»

Студент: Лукманова Аэлита
Группа: М8О–201Б–19
Вариант: «на удовлетворительно»
Преподаватель: Миронов Е. С.
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021.

Постановка задачи

Цель работы

Целью является приобретение практических навыков в:

· использовании знаний, полученных в курсе

· проведение исследований в выбранной области

Задание

Написать три программы A, B, C. A принимает из стандартного ввода строки и отправляет их в C построчно.C печатает в стандартный вывод то, что получила, отправляет в A сообщение о том, что строка получена. Только после этого сообщения A может отослать новую строку. B пишет в стандартный вывод количество отправленных символов A и количество принятых символов C. Эту информацию она получает от A и B.

Общие сведения о программе

Для работы с очередями используется ZMQ. Три файла A.c, B.c, C.c. B

программе используются следующие системные вызовы:

1. **socket.setsockopt** – устанавливает флаги для сокета.
2. **zmq::context_t –** создает ZMQ контекст.
3. **zmq::socket_t –** создает ZMQ сокет.
4. **zmq::message_t** – создает ZMQ сообщение.
5. **socket.send** – отправляет ZMQ сообщение на socket.
6. **socket.bind** – принимает соединие к сокету.

Общий метод и алгоритм решения.

Все описано в задании.

Основные файлы программы

A.c:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <errno.h>
#include <fcntl.h>

#include "zmq.h"
```

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

    char arrayOfString[50][200];
    printf("Input strings: \n");
    int arrayCount = 0;
    char line[200];
    while(fgets(line, 200, stdin)){
        //чтобы не было в конце перевода строки
        line[strlen(line) - 1] = '\0';
        strcpy(arrayOfString[arrayCount], line);
        arrayCount++;
    }

    //присоединяемся, как слушатель C
    void* context2 = zmq_ctx_new();
    void* subscriber = zmq_socket(context2, ZMQ_SUB);
    int conn = zmq_connect(subscriber, "tcp://localhost:4041");
    conn = zmq_setsockopt(subscriber, ZMQ_SUBSCRIBE, "", 0);

    //присоединяемся, как диктор для B
    void* context3 = zmq_ctx_new();
    void* publisher3 = zmq_socket(context3, ZMQ_PUB);
    zmq_bind(publisher3, "tcp://*:4042");
    printf("Устанавливаем соединение между A и B..\n");
    sleep(3);

    //присоединяемся, как диктор для C
    void* context = zmq_ctx_new();
    void* publisher = zmq_socket(context, ZMQ_PUB);
    zmq_bind(publisher, "tcp://*:4040");
    //ждем, когда соединение установится
    printf("Устанавливаем соединение между A и C...\n");
    sleep(3);

    int i = 0;
    while(i < arrayCount) {

        printf("A отсылает в C строку номер %d: %s\n", i+1, arrayOfString[i]);
        char *name = arrayOfString[i];
        zmq_msg_t message;
        printf("strlen(name): %lu\n", strlen(name));
        zmq_msg_init_size(&message, strlen(name)+1);
        memcpy(zmq_msg_data(&message), name, strlen(name)+1);
        zmq_msg_send(&message, publisher, 0);
        zmq_msg_close(&message);
        i++;
```

```
        //После отправления отсылаем в B, сколько символов отправлено
        zmq_msg_t messageB;
        zmq_msg_init_size(&messageB, strlen(name)+1);
        memcpy(zmq_msg_data(&messageB), name, strlen(name)+1);
        zmq_msg_send(&messageB, publisher3, 0);
        zmq_msg_close(&messageB);


        //После отправления ждем ответа от C, что строка получена
        zmq_msg_t reply;
        zmq_msg_init(&reply);
        zmq_msg_recv(&reply, subscriber, 0);
        int length = zmq_msg_size(&reply);
        char* value = malloc(length+1);
        memcpy(value, zmq_msg_data(&reply), length);
        zmq_msg_close(&reply);
        printf("%s\n", value);
        free(value);

        zmq_sleep(1);
    }

    zmq_close(publisher);
    zmq_ctx_destroy(context);

    zmq_close(subscriber);
    zmq_ctx_destroy(context2);

    zmq_close(publisher3);
    zmq_ctx_destroy(context3);
}
```

B.c:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <errno.h>
#include <fcntl.h>

#include "zmq.h"

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

    //присоединяемся, как слушатель A
    void* context1 = zmq_ctx_new();
    void* subscriber1 = zmq_socket(context1, ZMQ_SUB);
    int conn1 = zmq_connect(subscriber1, "tcp://localhost:4042");
    conn1 = zmq_setsockopt(subscriber1, ZMQ_SUBSCRIBE, "", 0);

    //присоединяемся, как слушатель C
    void* context2 = zmq_ctx_new();
    void* subscriber2 = zmq_socket(context2, ZMQ_SUB);
    int conn2 = zmq_connect(subscriber2, "tcp://localhost:4043");
```

```
    conn2 = zmq_setsockopt(subscriber2, ZMQ_SUBSCRIBE, "", 0);


    while(1) {
        //печать, сколько символов отправлено A
        zmq_msg_t replyA;
        zmq_msg_init(&replyA);
        zmq_msg_recv(&replyA, subscriber1, 0);
        int length1 = zmq_msg_size(&replyA);
        zmq_msg_close(&replyA);
        printf("Send from A: %d symbols\n", length1);


        //печать, сколько символов получено C
        zmq_msg_t replyC;
        zmq_msg_init(&replyC);
        zmq_msg_recv(&replyC, subscriber2, 0);
        int length2 = zmq_msg_size(&replyC);
        zmq_msg_close(&replyC);
        printf("Recieved by C: %d symbols\n\n", length2);
    }

    zmq_close(subscriber1);
    zmq_ctx_destroy(context1);

    zmq_close(subscriber2);
    zmq_ctx_destroy(context2);

}
```

C.c:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <errno.h>
#include <fcntl.h>

#include "zmq.h"

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

    //присоединяемся, как диктор для A
    void* context2 = zmq_ctx_new();
    void* publisher = zmq_socket(context2, ZMQ_PUB);
    zmq_bind(publisher, "tcp://*:4041");

    //присоединяемся, как диктор для B
    void* context3 = zmq_ctx_new();
    void* publisher3 = zmq_socket(context3, ZMQ_PUB);
    zmq_bind(publisher3, "tcp://*:4043");
    printf("Устанавливаем соединение между C и B..\n");
    sleep(3);

    //присоединяемся, как слушатель A
    void* context = zmq_ctx_new();
```

```
    void* subscriber = zmq_socket(context, ZMQ_SUB);
    printf("Collecting stock information from the server.\n");
    int conn = zmq_connect(subscriber, "tcp://localhost:4040");
    conn = zmq_setsockopt(subscriber, ZMQ_SUBSCRIBE, "", 0);
    int i = 0;
    for(i = 0; i < 10; i++)
    {
        sleep(3);
        zmq_msg_t reply;
        zmq_msg_init(&reply);
        zmq_msg_recv(&reply, subscriber, 0);
        int length = zmq_msg_size(&reply);
        char* value = malloc(length+1);
        memcpy(value, zmq_msg_data(&reply), length);
        zmq_msg_close(&reply);
        printf("%s\n", value);
        free(value);

        //после каждого получения отправляем уведомление об этом в A
        char *name = "C получила строку от A";
        printf("C отправляет A сообщение: %s\n", name);
        zmq_msg_t message;
        zmq_msg_init_size(&message, strlen(name)+1);
        memcpy(zmq_msg_data(&message), name, strlen(name)+1);
        zmq_msg_send(&message, publisher, 0);
        zmq_msg_close(&message);


        //а также отправляем, сколько получено символов, в B
        //не нашла быстро, как отправлять число, будем отправлять строку а там уже
считать
        zmq_msg_t messageB;
        zmq_msg_init_size(&messageB, length);
        memcpy(zmq_msg_data(&messageB), value, length);
        zmq_msg_send(&messageB, publisher3, 0);
        zmq_msg_close(&messageB);


    }
    zmq_close(subscriber);
    zmq_ctx_destroy(context);

    zmq_close(publisher);
    zmq_ctx_destroy(context2);

    zmq_close(publisher3);
    zmq_ctx_destroy(context3);
}
```

Пример работы

```
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/os303725_lab_678/src$ make
g++ -std=c++11 -c server.cpp -o server.o
g++ -std=c++11 -c client.cpp -o client.o
g++ -std=c++11 -c server_func.cpp -o server_func.o
g++ -std=c++11 -c my_tree.cpp -o tree.o
g++ -std=c++11 server.o server_func.o tree.o -o server -lzmq
g++ -std=c++11 client.o server_func.o -o client -lzmq
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/os_lab_678/src$ ./server

COMMANDS
create <id>
exec <id> <n> <k0...kn>
remove <id>
heartbit
help
exit

create 5
Ok:2511
create 6
Ok:2516
create 7
Ok:2523
exec 5 2 2 3
Ok:5:5
exec 6 1 2
Ok:6:2
exec 7 3 1 0 3
Ok:7:4
remove 7
Ok
exec 5 1 2
Ok:5:2
exec 6 2 2 3
Ok:6:5
heartbit 5
Ok
exit

Программа A.c
aelitalukmanova@MacBook-Pro-Aelita os-kp % gcc -Wall A.c -over -L/usr/local/
lib -lzmq
aelitalukmanova@MacBook-Pro-Aelita os-kp % ./ver
Input strings:
first string
second string
3d string
last string
Устанавливаем соединение между A и B..
Устанавливаем соединение между A и C...
A отсылает в C строку номер 1: first string

C получила строку от A
A отсылает в C строку номер 2: second string

C получила строку от A
A отсылает в C строку номер 3: 3d string

C получила строку от A
A отсылает в C строку номер 4: last string

C получила строку от A
```

```
Программа C.c
aelitalukmanova@MacBook-Pro-Aelita os-kp % gcc -Wall C.c -over -L/usr/local/
lib -lzmq
aelitalukmanova@MacBook-Pro-Aelita os-kp % ./ver
Устанавливаем соединение между C и B..
first string
C отправляет A сообщение: C получила строку от A
second string
C отправляет A сообщение: C получила строку от A
3d string
C отправляет A сообщение: C получила строку от A
last string
C отправляет A сообщение: C получила строку от A

Программа B.c
aelitalukmanova@MacBook-Pro-Aelita os-kp % gcc -Wall B.c -over -L/usr/local/
lib -lzmq
aelitalukmanova@MacBook-Pro-Aelita os-kp % ./ver
Send from A: 13 symbols
Recieved by C: 13 symbols

Send from A: 14 symbols
Recieved by C: 14 symbols

Send from A: 10 symbols
Recieved by C: 10 symbols

Send from A: 12 symbols
Recieved by C: 12 symbols
```

Вывод

Получилось решение в лоб, но работает. Может, можно было сделать более просто, но я не знаю как. Задание похоже на подзадачу последней лабораторной, то есть нового вывода не добавилось.