Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Курсовая по курсу

«Операционные системы»

Управление серверами сообщений, применение отложенных вычислений, интеграция программных систем друг с другом.

Студент: Ивенкова Л.В.
Группа: М80 – 208Б-19
Преподаватель: Миронов Е. С.
Дата: _____
Оценка: _____
Подпись: _____

Москва, 2021

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 B. Отправка строк должна производится построчно. Программа B печатает в стандартый вывод, полученную строку от программы A. После получения программа B отправляет программе A сообщение о том, что строка получена. До тех пор, пока программа A не примет «сообщение о получение строки» от программы B, она не может отправялять следующую строку программе B.
Программа C пишет в стандартный вывод количество отправленных символов программой A и количество принятых символов программой B. Данную информацию программа C получает от программ A и B соответственно.

Описание программы

Программа была реализована с помощью библиотеки ZeroMQ. Данная библиотека предлагает разработчику более высокий уровень абстракции при работе с сокетами/соединениями/очередями.

Для взаимодействия программ необходимо по отдельности запустить программы A, B и C. Порядок запуска не имеет значения. Программа A подключается к узлам B и C, после получает на вход строки, каждую полученную строку она сразу же отправляет размер отправляемой строки в C и ждет ответа «String received», дальше отправляет сообщение с длиной строки в B и ждет ответ "Good".

При получении сообщения от A, C проверяет, что это не «close$» и выводит в свой терминал сообщение, переданное от A, и отвечает обратным сообщением «String received», а в B передает размер сообщения. Если C получил «close$», то узел перестает принимать сообщения и выключается.

B принимает сообщение от A (размер сообщения, которое A передало C) и выводит на экран и отправляет «Good». И так же принимает сообщение от C (размер сообщения, которое C получило от A), печатает на экран и отправляет «Good» обратно. Если получил «end», то узел перестает принимать сообщения и выключается.

Чтобы прекратить ввод в A, необходимо нажать Ctrl+D (на linux-системах данная команда посылает EOF во входной поток).

Набор тестов:

Программа A

```
[parsifal@DESKTOP-3G70RV4:~/OS/CP$ ./A
CTYCbv1c!
Tuu
  mrya
```

Программа B

```
parsifal@DESKTOP-3G70RV4:~/OS/CP$ ./B
A: 9
C: 9

A: 3
C: 3

A: 6
C: 6
```

Программа C

```
parsifal@DESKTOP-3G70RV4:~/OS/CP$ ./C
CTYCbv1c!
Tuu
  Mrya
```

Листинг A.cpp

```
#include <iostream>
#include "myzmq.h"
#include <string>

#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;
}
```

Листинг 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;
}
```

Листинг 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;
}
```

Вывод

Данная курсовая работа основывается на знаниях, полученных в ходе изучения курса. По итогу мы получили несколько программ, которые взаимодействуют друг с другом с помощью сокетов.

Задача курсового проекта не сложна в реализации, но ее реализация обобщает и закрепляет полученные в курсе знания.