Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу

«Операционные системы»

Тема работы

Приобретение практических навыков в использовании знаний, полученных в течении курса.

Студент: Полонский Кирилл Андреевич
Группа: М8О-208Б-20
Вариант: __
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

Содержание

Репозиторий

https://github.com/kirillpolonskii/OS/tree/master/os_cp

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

Общие сведения о программе

Файл zmq_helper.hpp содержит вспомогательные данные, файлы prog_a.cpp, и prog_b.cpp и prog_c.cpp содержат реализацию программ A, B, C, сборка осуществляется с помощью утилиты cmake.

Общий метод и алгоритм решения

Межпроцессорное взаимодействие осуществляется с помощью очереди сообщений, программы A и C используют паттерн REQ-REP, а программы A и B, C и B — PUSH-PULL.

Исходный код

zmq_helper.hpp

```
#ifndef OS_CP_CLION_ZMQ_HELPER_HPP
#define OS_CP_CLION_ZMQ_HELPER_HPP

#include <zmq.hpp>

std::string addressAB = "tcp://127.0.0.1:5558";
std::string addressAC = "tcp://127.0.0.1:5559";
std::string addressBC = "tcp://127.0.0.1:5560";

#endif //OS_CP_CLION_ZMQ_HELPER_HPP
```

prog_a.cpp

```
#include <iostream>
#include "zmq_helper.hpp"

zmq::context_t context;

int main() {
    zmq::socket_t sockAB(context, zmq::socket_type::push);
    zmq::socket_t sockAC(context, zmq::socket_type::req);
    sockAB.bind(addressAB);
    sockAC.connect(addressAC);

    std::string curStr;
    while(std::cin >> curStr){
        if(curStr == "exit"){
            zmq::message_t zMesToC(curStr);
            sockAC.send(zMesToC, zmq::send_flags::none);
            zmq::message_t zMesToB(curStr);
            sockAB.send(zMesToB, zmq::send_flags::none);
            break;
        }
        zmq::message_t zMesToC(curStr);
        sockAC.send(zMesToC, zmq::send_flags::none);

        std::string mesToB = std::to_string(curStr.size());
        zmq::message_t zMesToB(mesToB);
        sockAB.send(zMesToB, zmq::send_flags::none);

        zmq::message_t zMesFromC;
```

```
        if(sockAC.recv(zMesFromC) == -1){
            return 1;
        }
        if(zMesFromC.to_string() == "String was received"){
            continue;
        }
    }


    sockAB.close();
    sockAC.close();
    return 0;
}
```

prog_b.cpp

```
#include <iostream>
#include "zmq_helper.hpp"

zmq::context_t context;

int main() {
    zmq::socket_t sockAB(context, zmq::socket_type::pull);
    zmq::socket_t sockBC(context, zmq::socket_type::pull);
    sockAB.connect(addressAB);
    sockBC.connect(addressBC);

    while(true){
        std::cout << "in while\n";
        zmq::message_t zMesFromA;
        if(sockAB.recv(zMesFromA) == -1){
            return 1;
        }
        std::string mesFromA = zMesFromA.to_string();
        if(mesFromA == "exit"){
            break;
        }
        else{
            std::cout << mesFromA << std::endl;
        }
        //sleep(1);
        zmq::message_t zMesFromC;
        if(sockBC.recv(zMesFromC) == -1){
            return 1;
        }
        std::string mesFromC = zMesFromC.to_string();
        std::cout << mesFromC << std::endl;
    }


    sockAB.close();
```

```
    sockBC.close();
    return 0;
}
```

prog_c.cpp

```
#include <iostream>
#include "zmq_helper.hpp"

zmq::context_t context;

int main() {
    zmq::socket_t sockAC(context, zmq::socket_type::rep);
    zmq::socket_t sockBC(context, zmq::socket_type::push);
    sockAC.bind(addressAC);
    sockBC.bind(addressBC);

    std::string curStr;
    while(true){
        zmq::message_t zMesFromA;
        if(sockAC.recv(zMesFromA) == -1){
            return 1;
        }
        std::string mesFromA = zMesFromA.to_string();
        if(mesFromA == "exit"){
            break;
        }
        else{
            std::cout << mesFromA << std::endl;
        }

        std::string mesToA("String was received");
        zmq::message_t zMesToA(mesToA);
        sockAC.send(zMesToA, zmq::send_flags::none);

        std::string mesToB = std::to_string(mesFromA.size());
        zmq::message_t zMesToB(mesToB);
        sockBC.send(zMesToB, zmq::send_flags::none);
    }


    sockAC.close();
    sockBC.close();
    return 0;
}
```

Демонстрация работы программы

progA	progB	progC

kirill@kirill-acpire:~/labs MAI/sem3/os_cp_clion/cmake-build-debug$ ./progC hello salut bye	kirill@kirill-acpire:~/labs MAI/sem3/os_cp_clion/cmake-build-debug$ ./progB 5 5 5 5 3 3	kirill@kirill-acpire:~/labs MAI/sem3/os_cp_clion/cmake-build-debug$ ./progC hello salut bye

Выводы

В ходе выполнения курсового проекта я закрепил навыки работы с очередью сообщений.