Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

**Лабораторная работа №5-7 по курсу**

**«Операционные системы»**

Группа: М80-206Б-22

Студентка: Варнак Е. В.

Преподаватель: Миронов Е.С.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 06.01.24

Москва, 2023

**Постановка задачи**

Вариант 21.

Реализовать распределенную систему по асинхронной обработке запросов. В данной распределенной системе должно существовать 2 вида узлов: «управляющий» и «вычислительный». Необходимо объединить данные узлы в соответствии с той топологией, которая определена вариантом. Связь между узлами необходимо осуществить при помощи технологии очередей сообщений. Также в данной системе необходимо предусмотреть проверку доступности узлов в соответствии с вариантом. При убийстве («kill -9») любого вычислительного узла система должна пытаться максимально сохранять свою работоспособность, а именно все дочерние узлы убитого узла могут стать недоступными, но родительские узлы должны сохранить свою работоспособность. Управляющий узел отвечает за ввод команд от пользователя и отправку этих команд на вычислительные узлы.

Список основных поддерживаемых команд:

Создание нового вычислительного узла

Формат команды: create id [parent] id – целочисленный идентификатор нового вычислительного узла parent – целочисленный идентификатор родительского узла. Если топологией не предусмотрено введение данного параметра, то его необходимо игнорировать (если его ввели)

Формат вывода:

«Ok: pid», где pid – идентификатор процесса для созданного вычислительного узла

«Error: Already exists» - вычислительный узел с таким идентификатором уже существует

«Error: Parent not found» - нет такого родительского узла с таким идентификатором

Примечания: создание нового управляющего узла осуществляется пользователем программы при помощи запуска исполняемого файла. Id и pid — это разные идентификаторы. Исполнение команды на вычислительном узле

Формат команды:

exec id [params] id – целочисленный идентификатор вычислительного узла, на который отправляется команда

Формат вывода:

«Ok:id: [result]», где result – результат выполненной команды

«Error:id: Not found» - вычислительный узел с таким идентификатором не найден

Примечание: выполнение команд должно быть асинхронным. Т.е. пока выполняется команда на одном из вычислительных узлов, то можно отправить следующую команду на другой вычислительный узел.

Все вычислительные узлы хранятся в бинарном дереве поиска. [parent] — является необязательным параметром. Узлы находятся в идеально сбалансированном бинарном дереве. Каждый следующий узел должен добавляться в самое наименьшее поддерево.

Набора команд 3 (локальный таймер):

Формат команды сохранения значения:

exec id subcommand (subcommand – одна из трех команд: start, stop, time.) start – запустить таймер stop – остановить таймер time – показать время локального таймера в миллисекундах

Пример:

$> exec 10 time

Ok:10: 0

$>exec 10 start

Ok:10

$>exec 10 start

Ok:10 \*прошло 10 секунд\*

$ > exec 10 time

Ok:10: 10000 \*прошло 2 секунды\*

$>exec 10 stop

Ok:10 \*прошло 2 секунды\*

$ >exec 10 time

Ok:10: 12000

Команда проверки 2:

Формат команды:

ping id

Команда проверяет доступность конкретного узла. Если узла нет, то необходимо выводить ошибку: «Error: Not found»

Пример:

$> ping 10

Ok: 1 // узел 10 доступен

$> ping 17

Ok: 0 // узел 17 недоступен

**Общий метод и алгоритм решения**

Решение основано на взаимодействии “сервера” и “клиента”, путем передачи сообщений с помощью очереди сообщений ZeroMQ и вычислительных узлах, которые хранятся в дереве, реализация находится в файле tree.hpp .

Сборка проекта осуществляется с помощью CMakeLists.txt, запуск исполняемого файла сервера.

server.cpp

#include <iostream>

#include <string>

#include "calcNode.hpp"

#include "zmq\_functions.hpp"

#include "tree.hpp"

*void* instruction(){

    cout << "Введите команду:" << endl;;

    cout << "1) create <id>" << endl;

    // таймер

    cout << "2) ping <id>" << endl;

    cout << "3) exec <id> <start/stop/time>" << endl;

    cout << "4) kill <id>" << endl;

    cout << "5) exit - выход" << endl;

}

*int* main(){

    instruction();

    string command, answer, str;

    CalculationNode node(-1, -1, -1);

    BalancedTree tree;

*int* child;

    while (cout << "$> " && cin >> command){

        if (command == "create"){

            cout << "Начало работы сервера create" << endl;

            cin >> child;

            if (tree.Exist(child)) {

                cout << "error: child c таким id уже существует" << endl;

            } else {

                while(1){

*int* idParent = tree.FindID();

                    if (idParent == node.id){ //если равен идентификатор, то дочерний узел будет потомком этого узла

                        answer = node.create(child);

                        tree.AddInTree(child, idParent); //добавить узел в дерево

                        break;

                    } else {

                        string message = "create " +  to\_string(child); // если он не равен, то формируем сообщение

                        answer = node.sendstring(message, idParent); //(вызывается у текущей ноды) отправка на родительский узел (поиск) сообщения о создании нового узла

                        if (answer == "Error: Parent not found") {

                            tree.AvailabilityCheck(idParent); // тогда корректируем доступность дочернего узла и продолжим цикл

                        }

                        else {

                            tree.AddInTree(child, idParent); // иначе добавим в дерево узел

                            break;

                        }

                    }

                }

                cout << answer << endl;

            }

            cout << "Конец работы сервера create" << endl;

        } else if (command == "ping"){

            cin >> child;

            try{

            if (tree.Exist(child)) {

                string message = "ping " +  to\_string(child);

                answer = node.sendstring(message, child); //отправляем сообщение на узел child, если наш id не является левым или правым потомком

                cout << answer <<  endl; //потом уже получаем ответ

            } else {

                cout << "ok: 0" << endl; // узел недоступен

            }

            } catch(*int*){}

        } else if (command == "exec"){

            cin >> child;

            getline(cin, str);

            if (tree.Exist(child)){

                string message = "exec" + str;

                answer = node.sendstring(message, child); //формирование сообщения, отправка на вычислительный узел

                cout << answer <<  endl; // результат выводится на экран

            } else {

                cout << "error: child c таким id не существует" << endl;

            }

        } else if (command == "kill"){

            cin >> child;

            if (tree.Exist(child)){

               answer = node.sendstring("kill", child); // отправляет сообщение "kill" на указанный узел child

               if (answer != "Error: Parent not found"){

                    tree.RemoveFromRoot(child); // удаление узла из дерева

                    if (child == node.left\_id){

                        node.left\_id = -2; // отсутствие узла

                        unbind(node.left, node.left\_port); // разрыв связи с портом, занятым соответсвующим узлом, чтобы освободить его

                        answer = "ok";

                    }

                    else if (child == node.right\_id) {

                        node.right\_id = -2;

                        unbind(node.right, node.right\_port);

                        answer = "ok";

                    }

               }

               cout << answer <<  endl;

            } else {

                cout << "error: child c таким id не существует" << endl;

            }

        } else if (command == "exit"){

            cout << "Выходим из программы :3" << endl;

            break;

        } else {

            instruction();

        }

    }

    node.kill(); //для корректного завершения работы сервера

    return 0;

}

client.cpp

#include <iostream>

#include <string>

#include "calcNode.hpp"

#include "zmq\_functions.hpp"

#include "tree.hpp"

using *namespace* std;

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

    CalculationNode node(atoi(*argv*[1]), atoi(*argv*[2]), atoi(*argv*[3])); //int

    std::cout << "Работает Client.cpp" << std::endl;

    string message, command, answer, str;

*int* child;

    while (1){

        message = receive\_message(node.parent); //получаем сообщение от родительского узла

        istringstream request(message); //поток ввода, как cin, только для строки message

        request >> command; //из потока извлекается первое слово

        if (command == "create"){

            cout << "Creating..." << endl;

            request >> child;

            answer = node.create(child); //вернет child\_pid созданного процесса

            send\_message(node.parent, answer);

            cout << "Done" << endl;

        } else if (command == "ping"){

            request >> child; //считываем после команды число - идентификатор процесса

            answer = node.ping(child); //пингуем дочерний процесс с эти идентификатором

            send\_message(node.parent, answer);

        } else if (command == "exec"){

            getline(request, str); // считываем всю оставшуюся строку после exec до '\n'

            answer = node.exec(str); // вернет id

            send\_message(node.parent, answer);

        } else if (command == "kill"){

            answer = node.kill(); // возвращает parent\_id

            send\_message(node.parent, answer);

            disconnect(node.parent, node.parent\_port); // отключение от родительского узла

            node.parent.close(); // закрытие соединения

            break;

        } else if (command == "pid"){

            answer = to\_string(getpid()); //просто получаем идентификатор текущего процесса

            send\_message(node.parent, answer); //отправляем ответ родительскому узлу

        }

    }

    return 0;

}

tree.hpp

#pragma once

#include <iostream>

#include <string>

*class* Tree {

*class* TreeNode {

*public:*

*int* id; //идентификатор узла

                TreeNode\* left; //указатель на левый дочерний узел

                TreeNode\* right;

*int* height; //высота узла в дереве

*bool* available; //флаг доступности узла

                TreeNode (*int* *id*) { //init узла

                    this->id = *id*;

                    available = true;

                    left = NULL;

                    right = NULL;

                }

*void* CheckAvailability (*int* *id*) {

                    if (this->id == *id*){

                        available = false; // если id совпадает с узлом, на котором проходит проверка, то это узел недоступен

                    }

                    else {

                        if (left != NULL) {

                            left->CheckAvailability(*id*);

                        }

                        if (right != NULL) {

                            right->CheckAvailability(*id*);

                        }

                    }

                }

*void* Remove (*int* *id*, set<*int*> &*ids*) { // узла; множество идентификаторов всех узлов

                    if (left != NULL && left->id == *id*) {

                        left->RecursionRemove(*ids*); //удаление всех потомков левого потомка

*ids*.erase(left->id); // удаляется идентификатор левого потомка

                        delete left; //освобождается память, выделенная для узла левого потомка

                        left = NULL;

                    }

                    else if (right != NULL && right->id == *id*) {

                        right->RecursionRemove(*ids*);

*ids*.erase(right->id);

                        delete right;

                        right = NULL;

                    }

                    else { // если не совпали ранее

                        if (left != NULL) {

                            left->Remove(*id*, *ids*);

                        }

                        if (right != NULL) {

                            right->Remove(*id*, *ids*);

                        }

                    }

                }

*void* RecursionRemove (set<*int*> &*ids*) {

                    if (left != NULL) {

                        left->RecursionRemove(*ids*); // удаляем всех левых

*ids*.erase(left->id);

                        delete left;

                        left = NULL;

                    }

                    if (right != NULL) {

                        right->RecursionRemove(*ids*); // удаляем всех правых

*ids*.erase(right->id);

                        delete right;

                        right = NULL;

                    }

                }

*void* AddInNode (*int* *id*, *int* *parent\_id*, set<*int*> &*ids*) { // id - кого добавляем, parent\_id - куда добавляем, set - все узлы

                    if (this->id == *parent\_id*) {

                        if (left == NULL){

                            left = new TreeNode(*id*);

                        }

                        else {

                            right = new TreeNode(*id*);

                        }

*ids*.insert(*id*); // добавляем в id множество всех

                    }

                    else {

                        if (left != NULL) {

                            left->AddInNode(*id*, *parent\_id*, *ids*); // ведем поиск parent\_id

                        }

                        if (right != nullptr) {

                            right->AddInNode(*id*, *parent\_id*, *ids*);

                        }

                    }

                }

*int* MinimalHeight() {

                    // Проверяется, существуют ли левый и правый потомки.

                    // Если хотя бы один из них отсутствует (равен NULL), то это значит, что текущий узел является листом и его высота равна 0.

                    // В этом случае метод возвращает 0.

                    if (left == NULL || right == NULL) { //

                        return 0;

                    }

*int* left\_height = -1;

*int* right\_height = -1;

                    if (left != NULL && left->available == true) {

                        left\_height = left->MinimalHeight(); // если узел существует и доступен, то считаем высоту

                    }

                    if (right != NULL && right->available == true) {

                        right\_height = right->MinimalHeight();

                    }

                    if (right\_height == -1 && left\_height == -1) {

                        available = false; // если высота не вычислена, то узел является недоступным

                        return -1;

                    }

                    else if (right\_height  == -1) {

                        return left\_height + 1;

                    }

                    else if (left\_height == -1) {

                        return right\_height + 1;

                    }

                    else {

                        return min(left\_height, right\_height) + 1;

                    }

                }

*int* IDMinimalHeight(*int* *height*, *int* *current\_height*) { //поиск идентификатора узла с заданной высотой

                    if (*height* < *current\_height*) {

                        return -2;

                    }

                    else if (*height* > *current\_height*) {

*int* current\_id = -2;

                        if (left != NULL && left->available == true) {

                            current\_id = left->IDMinimalHeight(*height*, (*current\_height* + 1));

                        }

                        if (right != NULL && right->available == true && current\_id == -2){

                            current\_id = right->IDMinimalHeight(*height*, (*current\_height* + 1));

                        }

                        return current\_id;

                    }

                    else {

                        if (left == NULL || right == NULL){

                            return id;

                        }

                        return -2;

                    }

                }

                ~TreeNode() {}

        };

*private:*

            TreeNode\* root; // корневой узел дерева

*public:*

        set<*int*> ids;

        Tree() {

            root = new TreeNode(-1); // корневой узел становися -1, init

        }

*bool* Exist(*int* *id*) {

            if (ids.find(*id*) != ids.end()) { // поиск во множестве

                return true;

            }

            return false;

        }

*void* AvailabilityCheck(*int* *id*) { // корректировка доступности узла

            root->CheckAvailability(*id*);

        }

*int* FindID() {

*int* h = root->MinimalHeight(); // сюда будем добавлять

            return root->IDMinimalHeight(h, 0); // находит идентификатор узла с минимальной высотой в дереве

        }

*void* AddInTree(*int* *id*, *int* *parent*) { // добавляет новый узел с заданным идентификатором

            root->AddInNode(*id*, *parent*, ids);

        }

*void* RemoveFromRoot(*int* *idElem*) { //удаляет узел

            root->Remove(*idElem*, ids);

        }

        ~Tree() {

            root->RecursionRemove(ids); //деструктор, рекурсивно удаляет все

            delete root;

        }

};

zmq\_functions.hpp

#pragma once

#include <iostream>

#include <string>

#include <zmq.hpp>

using *namespace* std;

*void* send\_message(zmq::*socket\_t* &*socket*, const string &*message*){

    zmq::*message\_t* m(message.size());

    memcpy(m.data(), message.c\_str(), message.size()); // копирует данные из строки message в буфер сообщения m

    socket.send(m);

    // отправляет сообщение из объекта m через заданный сокет

}

string receive\_message(zmq::*socket\_t* &*socket*){

    zmq::*message\_t* message;

*int* chars\_read; // количество прочитанных символов из сообщения

    try {

        chars\_read = (*int*)socket.recv(&message); // прием сообщения

    }

    catch (...) {

        chars\_read = 0;

    }

    if (chars\_read == 0) {

        throw -1;

    }

    string received\_msg(static\_cast<*char*\*>(message.data()), message.size());

    // конструктора, который принимает указатель char\* на данные сообщения (message.data()) и его размер (message.size())

    return received\_msg;

}

*int* bind(zmq::*socket\_t* &*socket*, *int* *id*){

*int* port = 1024 + id;

    string address = "tcp://127.0.0.1:" + to\_string(port);

    while(1){

        try{

            socket.bind(address); // попытка привязать сокет к порту

            break;

        }

        catch(...){

            port++; // попытка привязать сокет к порту на 1 больше, чем предыдущий в цикле

        }

    }

    return port;

}

*void* unbind(zmq::*socket\_t* &*socket*, *int* *port*){

    string address = "tcp://127.0.0.1:" + to\_string(port);

    socket.unbind(address);

}

*void* connect(zmq::*socket\_t* &*socket*, *int* *port*){

    string address = "tcp://127.0.0.1:" + to\_string(port); // формирование адреса (строка)

    socket.connect(address); // установка соединения

}

*void* disconnect(zmq::*socket\_t* &*socket*, *int* *port*){

    string address = "tcp://127.0.0.1:" + to\_string(port);

    socket.disconnect(address);

}

calcNode.hpp

#include <iostream>

#include <string>

#include <set>

#include "unistd.h"

#include "zmq\_functions.hpp"

*class* CalculationNode

{

*private:*

    zmq::*context\_t* context; // контекст ZMQ для создания сокетов

*bool* timerRunning; // переменная, которая отслеживает состояние таймера, если активен, то true

    chrono::steady\_clock::time\_point startTime; //время начала таймера

    std::chrono::milliseconds stopTime; // чтобы хранить время после stop таймер

*public:*

    zmq::*socket\_t* left, right, parent;      // сокеты для связи с левым, правым и родительским узлами

*int* left\_port, right\_port, parent\_port; // порты соединения с левым, правым и родительским узлами

*int* id, left\_id = -2, right\_id = -2, parent\_id; // идентификаторы текущего узла, левого узла, правого узла и родительского узла

    CalculationNode(*int* *id*, *int* *parent\_port*, *int* *parent\_id*) : id(id),

        parent\_port(parent\_port),

        parent\_id(parent\_id),

        left(context, ZMQ\_REQ), //создание сокетов с помощью контекста context и соответствующих типов (ZMQ\_REQ и ZMQ\_REP)

        right(context, ZMQ\_REQ),

        parent(context, ZMQ\_REP)

    {

        if (id != -1) // не управляющий узел

        {

            connect(parent, parent\_port); // устанавливается соединение с родительским узлом через метод connect

        }

    }

    string create(*int* *child\_id*)

    {

*int* port;

*bool* isleft = false; // создаваемый узел левый или правый

        // проверка доступности левого узла для подключения

        if (left\_id == -2) // если -2, то отсутствует подключение

        {

            left\_port = bind(left, child\_id); // связывает сокет left с указанным идентификатором child\_id и возвращает порт для соединения с левым узлом

            left\_id = child\_id;

            port = left\_port;

            isleft = true;

        }

        else if (right\_id == -2)

        {

            right\_port = bind(right, child\_id);

            right\_id = child\_id;

            port = right\_port;

        }

        else

        {

            string fail = "Error: can not create the calculation node";

            return fail;

        }

*int* fork\_id = fork();

        if (fork\_id == 0)

        {

            if (execl("./client", "client", to\_string(child\_id).c\_str(), to\_string(port).c\_str(), to\_string(id).c\_str(), (*char* \*)NULL) == -1)

            {

                cout << "Error: can not run the execl-command" << endl;

                exit(EXIT\_FAILURE);

            }

        }

        else

        {

            string child\_pid;

            try

            {

                if (isleft)

                {

                    left.setsockopt(ZMQ\_SNDTIMEO, 3000); // отправляется запрос на получение идентификатора процесса левого дочернего узла (child\_id) через сокет left.

                    send\_message(left, "pid");

                    child\_pid = receive\_message(left);

                }

                else

                {

                    right.setsockopt(ZMQ\_SNDTIMEO, 3000);

                    send\_message(right, "pid");

                    child\_pid = receive\_message(right);

                }

                return "Ok: " + child\_pid;

            }

            catch (*int*)

            {

                string fail = "Error: can not connect to the child";

                return fail;

            }

        }

    }

    string ping(*int* *id*)

    {

        string answer = "Ok: 0";

        try{

        if (this->id == id)

        {

            answer = "Ok: 1";

            return answer;

        }

        else if (left\_id == id)

        {

            string message = "ping " + to\_string(id);

            send\_message(left, message);

            try

            {

                message = receive\_message(left);

                if (message == "Ok: 1")

                {

                    answer = message;

                }

            }

            catch (*int*)

            {

            }

        }

        else if (right\_id == id)

        {

            string message = "ping " + to\_string(id);

            send\_message(right, message);

            try

            {

                message = receive\_message(right);

                if (message == "Ok: 1")

                {

                    answer = message;

                }

            }

            catch (*int*)

            {

            }

        }

        }

        catch (*int*){

        }

        return answer;

    }

    string sendstring(string *string*, *int* *id*)

    {

        std::string answer = "Error: Parent not found";

        try{

        // Проверка, что нет ни левого, ни правого потомка

        if (left\_id == -2 && right\_id == -2)

        {

            return answer;

        }

        else if (left\_id == id)

        {

            if (ping(left\_id) == "Ok: 1")

            {

                send\_message(left, string);

                try

                {

                    answer = receive\_message(left);

                }

                catch (*int*)

                {

                }

            }

        }

        else if (right\_id == id)

        {

            if (ping(right\_id) == "Ok: 1")

            {

                send\_message(right, string);

                try

                {

                    answer = receive\_message(right);

                }

                catch (*int*)

                {

                }

            }

        }

        else

        {

            // если узел ни левый, ни правый потомок

            if (ping(left\_id) == "Ok: 1") // доступен ли левый потомок

            {

                // формирование строки message, которая содержит команду "send" и идентификатор узла id, а также само сообщение string.

                std::string message = "send " + to\_string(id) + " " + string;

                send\_message(left, message);

                try

                {

                    message = receive\_message(left);

                }

                catch (*int*)

                {

                    message = "Error: Parent not found";

                }

                if (message != "Error: Parent not found")

                {

                    answer = message;

                }

            }

            if (ping(right\_id) == "Ok: 1")

            {

                std::string message = "send " + to\_string(id) + " " + string;

                send\_message(right, message);

                try

                {

                    message = receive\_message(right);

                }

                catch (*int*)

                {

                    message = "Error: Parent not found";

                }

                if (message != "Error: Parent not found")

                {

                    answer = message;

                }

            }

        }

        }

        catch(*int*){}

        return answer;

    }

    string exec(const string &*subcommand*) // входная команда, которую нужно выполнить

    {

        if (subcommand == " start")

        {

            if (!timerRunning)

            {

                startTime = chrono::steady\_clock::now();

                timerRunning = true;

                return "Ok:" + to\_string(id);

            }

            else

            {

                return "Error:" + to\_string(id) + ":Timer already running";

            }

        }

        else if (subcommand == " stop")

        {

            if (timerRunning)

            {

*auto* endTime = chrono::steady\_clock::now();

                stopTime = chrono::duration\_cast<chrono::milliseconds>(endTime - startTime);

                timerRunning = false;

                return "Ok:" + to\_string(id);

            }

            else

            {

                return "Error:" + to\_string(id) + ":Timer not running";

            }

        }

        else if (subcommand == " time")

        {

            if (timerRunning)

            {

*auto* currentTime = chrono::steady\_clock::now();

*auto* elapsedMilliseconds = chrono::duration\_cast<chrono::milliseconds>(currentTime - startTime).count();

                return "Ok:" + to\_string(id) + ":" + to\_string(elapsedMilliseconds);

            }

            else

            {

                return "Ok:" + to\_string(id) + ":" + to\_string(stopTime.count());

            }

        }

        else

        {

            std::cout << "мда, ошибка"<< std::endl;

            return "Error:" + to\_string(id) + ":" + subcommand + ": !!! Invalid subcommand";

        }

    }

    string kill()

    {

        try{

        // Завершение работы левого потомка

        if (left\_id != -2)

        {

            if (ping(left\_id) == "Ok: 1")

            {

                std::string message = "kill";

                send\_message(left, message);

                try

                {

                    message = receive\_message(left);

                }

                catch (*int*)

                {

                }

                unbind(left, left\_port);

                left.close();

            }

        }

        if (right\_id != -2)

        {

            if (ping(right\_id) == "Ok: 1")

            {

                std::string message = "kill";

                send\_message(right, message);

                try

                {

                    message = receive\_message(right);

                }

                catch (*int*)

                {

                }

                unbind(right, right\_port);

                right.close();

            }

        }

        }catch(*int*){

        }

        return to\_string(parent\_id);

    }

    ~CalculationNode() {

        // деструктор

    }

};

**Протокол работы программы**

**Тестирование:**

kateland@DESKTOP-0GN4T39:~/OSLabs/57build$ cmake --build .

[ 50%] Built target client

[100%] Built target server

kateland@DESKTOP-0GN4T39:~/OSLabs/57build$ ./server

Введите команду:

1) create <id>

2) ping <id>

3) exec <id> <start/stop/time>

4) kill <id>

5) exit - выход

$> create 4

Начало работы сервера create

Работает Client.cpp

Ok: 17471

Конец работы сервера create

$> create 3

Начало работы сервера create

Работает Client.cpp

Ok: 17474

Конец работы сервера create

$> create 2

Начало работы сервера create

Creating...

Работает Client.cpp

Done

Ok: 17477

Конец работы сервера create

$> kill 4

ok

$> ping 4

ok: 0

$> ping 3

Ok: 1

$> ping 2

ok: 0

$> exec 2 start

error: child c таким id не существует

$> exec 3 time

Ok:3:0

$> exec 3 start

Ok:3

$> exec 3 stop

Ok:3

$> exec 3 time

Ok:3:7319

$> exec 3 time

Ok:3:7319

$> exit

Выходим из программы :3

kateland@DESKTOP-0GN4T39:~/OSLabs/57build$

Обработка ошибок:

kateland@DESKTOP-0GN4T39:~/OSLabs/57build$ ls

CMakeCache.txt CMakeFiles Makefile cmake\_install.cmake server

kateland@DESKTOP-0GN4T39:~/OSLabs/57build$ ./server

Введите команду:

1) create <id>

2) ping <id>

3) exec <id> <start/stop/time>

4) kill <id>

5) exit - выход

$> create 4

Начало работы сервера create

Error: can not run the execl-command

Error: can not connect to the child

Конец работы сервера create

**Strace**

kateland@DESKTOP-0GN4T39:~/OSLabs/57build$ strace ./server

execve("./server", ["./server"], 0x7ffd8446ad30 /\* 20 vars \*/) = 0

brk(NULL) = 0x563446d2c000

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f2abcc86000

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/usr/local/lib/tls/haswell/avx512\_1/x86\_64/libzmq.so.5", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/usr/local/lib/tls/haswell/avx512\_1/x86\_64", 0x7ffc1770ce40) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/usr/local/lib/tls/haswell/avx512\_1/libzmq.so.5", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/usr/local/lib/tls/haswell/avx512\_1", 0x7ffc1770ce40) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/usr/local/lib/tls/haswell/x86\_64/libzmq.so.5", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/usr/local/lib/tls/haswell/x86\_64", 0x7ffc1770ce40) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/usr/local/lib/tls/haswell/libzmq.so.5", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/usr/local/lib/tls/haswell", 0x7ffc1770ce40) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/usr/local/lib/tls/avx512\_1/x86\_64/libzmq.so.5", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/usr/local/lib/tls/avx512\_1/x86\_64", 0x7ffc1770ce40) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/usr/local/lib/tls/avx512\_1/libzmq.so.5", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/usr/local/lib/tls/avx512\_1", 0x7ffc1770ce40) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/usr/local/lib/tls/x86\_64/libzmq.so.5", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/usr/local/lib/tls/x86\_64", 0x7ffc1770ce40) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/usr/local/lib/tls/libzmq.so.5", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/usr/local/lib/tls", 0x7ffc1770ce40) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/usr/local/lib/haswell/avx512\_1/x86\_64/libzmq.so.5", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/usr/local/lib/haswell/avx512\_1/x86\_64", 0x7ffc1770ce40) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/usr/local/lib/haswell/avx512\_1/libzmq.so.5", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/usr/local/lib/haswell/avx512\_1", 0x7ffc1770ce40) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/usr/local/lib/haswell/x86\_64/libzmq.so.5", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/usr/local/lib/haswell/x86\_64", 0x7ffc1770ce40) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/usr/local/lib/haswell/libzmq.so.5", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/usr/local/lib/haswell", 0x7ffc1770ce40) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/usr/local/lib/avx512\_1/x86\_64/libzmq.so.5", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/usr/local/lib/avx512\_1/x86\_64", 0x7ffc1770ce40) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/usr/local/lib/avx512\_1/libzmq.so.5", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/usr/local/lib/avx512\_1", 0x7ffc1770ce40) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/usr/local/lib/x86\_64/libzmq.so.5", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

stat("/usr/local/lib/x86\_64", 0x7ffc1770ce40) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/usr/local/lib/libzmq.so.5", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0`\206\1\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=16618144, ...}) = 0

mmap(NULL, 673272, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f2abcbe1000

mmap(0x7f2abcbf8000, 421888, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x17000) = 0x7f2abcbf8000

mmap(0x7f2abcc5f000, 126976, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x7e000) = 0x7f2abcc5f000

mmap(0x7f2abcc7e000, 32768, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x9c000) = 0x7f2abcc7e000

close(3) = 0

openat(AT\_FDCWD, "/usr/local/lib/libstdc++.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=57684, ...}) = 0

mmap(NULL, 57684, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f2abcbd2000

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libstdc++.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0`\341\t\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=1956992, ...}) = 0

mmap(NULL, 1972224, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f2abc9f0000

mprotect(0x7f2abca86000, 1290240, PROT\_NONE) = 0

mmap(0x7f2abca86000, 987136, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x96000) = 0x7f2abca86000

mmap(0x7f2abcb77000, 299008, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x187000) = 0x7f2abcb77000

mmap(0x7f2abcbc1000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1d0000) = 0x7f2abcbc1000

mmap(0x7f2abcbcf000, 10240, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f2abcbcf000

close(3) = 0

openat(AT\_FDCWD, "/usr/local/lib/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\3405\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=104984, ...}) = 0

mmap(NULL, 107592, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f2abc9d5000

mmap(0x7f2abc9d8000, 73728, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7f2abc9d8000

mmap(0x7f2abc9ea000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x15000) = 0x7f2abc9ea000

mmap(0x7f2abc9ee000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x18000) = 0x7f2abc9ee000

close(3) = 0

openat(AT\_FDCWD, "/usr/local/lib/libc.so.6", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\300A\2\0\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\30x\346\264ur\f|Q\226\236i\253-'o"..., 68, 880) = 68

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2029592, ...}) = 0

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\30x\346\264ur\f|Q\226\236i\253-'o"..., 68, 880) = 68

mmap(NULL, 2037344, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f2abc7e3000

mmap(0x7f2abc805000, 1540096, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x22000) = 0x7f2abc805000

mmap(0x7f2abc97d000, 319488, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19a000) = 0x7f2abc97d000

mmap(0x7f2abc9cb000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7f2abc9cb000

mmap(0x7f2abc9d1000, 13920, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f2abc9d1000

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libpthread.so.0", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220q\0\0\0\0\0\0"..., 832) = 832

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0{E6\364\34\332\245\210\204\10\350-\0106\343="..., 68, 824) = 68

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=157224, ...}) = 0

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0{E6\364\34\332\245\210\204\10\350-\0106\343="..., 68, 824) = 68

mmap(NULL, 140408, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f2abc7c0000

mmap(0x7f2abc7c6000, 69632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x6000) = 0x7f2abc7c6000

mmap(0x7f2abc7d7000, 24576, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x17000) = 0x7f2abc7d7000

mmap(0x7f2abc7dd000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1c000) = 0x7f2abc7dd000

mmap(0x7f2abc7df000, 13432, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f2abc7df000

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libm.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\300\323\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=1369384, ...}) = 0

mmap(NULL, 1368336, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f2abc671000

mmap(0x7f2abc67e000, 684032, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xd000) = 0x7f2abc67e000

mmap(0x7f2abc725000, 626688, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xb4000) = 0x7f2abc725000

mmap(0x7f2abc7be000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x14c000) = 0x7f2abc7be000

close(3) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f2abc66f000

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f2abc66c000

arch\_prctl(ARCH\_SET\_FS, 0x7f2abc66c740) = 0

mprotect(0x7f2abc9cb000, 16384, PROT\_READ) = 0

mprotect(0x7f2abc7be000, 4096, PROT\_READ) = 0

mprotect(0x7f2abc7dd000, 4096, PROT\_READ) = 0

mprotect(0x7f2abc9ee000, 4096, PROT\_READ) = 0

mprotect(0x7f2abcbc1000, 45056, PROT\_READ) = 0

mprotect(0x7f2abcc7e000, 28672, PROT\_READ) = 0

mprotect(0x563446601000, 4096, PROT\_READ) = 0

mprotect(0x7f2abccb5000, 4096, PROT\_READ) = 0

munmap(0x7f2abcbd2000, 57684) = 0

set\_tid\_address(0x7f2abc66ca10) = 17459

set\_robust\_list(0x7f2abc66ca20, 24) = 0

rt\_sigaction(SIGRTMIN, {sa\_handler=0x7f2abc7c6bf0, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_SIGINFO, sa\_restorer=0x7f2abc7d4420}, NULL, 8) = 0

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7f2abc7c6c90, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7f2abc7d4420}, NULL, 8) = 0

rt\_sigprocmask(SIG\_UNBLOCK, [RTMIN RT\_1], NULL, 8) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

brk(NULL) = 0x563446d2c000

brk(0x563446d4d000) = 0x563446d4d000

futex(0x7f2abcbcf6bc, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

futex(0x7f2abcbcf6c8, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x3), ...}) = 0

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\272\320\276\320\274\320\260\320\275\320\264\321\203:\n", 31Введите команду:

) = 31

write(1, "1) create <id>\n", 151) create <id>

) = 15

write(1, "2) ping <id>\n", 132) ping <id>

) = 13

write(1, "3) exec <id> <start/stop/time>\n", 313) exec <id> <start/stop/time>

) = 31

write(1, "4) kill <id>\n", 134) kill <id>

) = 13

write(1, "5) exit - \320\262\321\213\321\205\320\276\320\264\n", 215) exit - выход

) = 21

eventfd2(0, EFD\_CLOEXEC) = 3

fcntl(3, F\_GETFL) = 0x2 (flags O\_RDWR)

fcntl(3, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

fcntl(3, F\_GETFL) = 0x802 (flags O\_RDWR|O\_NONBLOCK)

fcntl(3, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

getpid() = 17459

getpid() = 17459

eventfd2(0, EFD\_CLOEXEC) = 4

fcntl(4, F\_GETFL) = 0x2 (flags O\_RDWR)

fcntl(4, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

fcntl(4, F\_GETFL) = 0x802 (flags O\_RDWR|O\_NONBLOCK)

fcntl(4, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

getpid() = 17459

epoll\_create1(EPOLL\_CLOEXEC) = 5

epoll\_ctl(5, EPOLL\_CTL\_ADD, 4, {0, {u32=1188307184, u64=94782526595312}}) = 0

epoll\_ctl(5, EPOLL\_CTL\_MOD, 4, {EPOLLIN, {u32=1188307184, u64=94782526595312}}) = 0

getpid() = 17459

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f2abbe6b000

mprotect(0x7f2abbe6c000, 8388608, PROT\_READ|PROT\_WRITE) = 0

clone(child\_stack=0x7f2abc66afb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tidptr=0x7f2abc66b9d0, tls=0x7f2abc66b700, child\_tidptr=0x7f2abc66b9d0) = 17460

eventfd2(0, EFD\_CLOEXEC) = 6

fcntl(6, F\_GETFL) = 0x2 (flags O\_RDWR)

fcntl(6, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

fcntl(6, F\_GETFL) = 0x802 (flags O\_RDWR|O\_NONBLOCK)

fcntl(6, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

getpid() = 17459

epoll\_create1(EPOLL\_CLOEXEC) = 7

epoll\_ctl(7, EPOLL\_CTL\_ADD, 6, {0, {u32=1188309216, u64=94782526597344}}) = 0

epoll\_ctl(7, EPOLL\_CTL\_MOD, 6, {EPOLLIN, {u32=1188309216, u64=94782526597344}}) = 0

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f2abb66a000

mprotect(0x7f2abb66b000, 8388608, PROT\_READ|PROT\_WRITE) = 0

clone(child\_stack=0x7f2abbe69fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tidptr=0x7f2abbe6a9d0, tls=0x7f2abbe6a700, child\_tidptr=0x7f2abbe6a9d0) = 17461

eventfd2(0, EFD\_CLOEXEC) = 8

fcntl(8, F\_GETFL) = 0x2 (flags O\_RDWR)

fcntl(8, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

fcntl(8, F\_GETFL) = 0x802 (flags O\_RDWR|O\_NONBLOCK)

fcntl(8, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

getpid() = 17459

eventfd2(0, EFD\_CLOEXEC) = 9

fcntl(9, F\_GETFL) = 0x2 (flags O\_RDWR)

fcntl(9, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

fcntl(9, F\_GETFL) = 0x802 (flags O\_RDWR|O\_NONBLOCK)

fcntl(9, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

getpid() = 17459

eventfd2(0, EFD\_CLOEXEC) = 10

fcntl(10, F\_GETFL) = 0x2 (flags O\_RDWR)

fcntl(10, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

fcntl(10, F\_GETFL) = 0x802 (flags O\_RDWR|O\_NONBLOCK)

fcntl(10, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

getpid() = 17459

write(1, "$> ", 3$> ) = 3

fstat(0, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x3), ...}) = 0

read(0, create 4

"create 4\n", 1024) = 9

write(1, "\320\235\320\260\321\207\320\260\320\273\320\276 \321\200\320\260\320\261\320\276\321\202\321\213 \321\201\320\265\321\200"..., 48Начало работы сервера create

) = 48

getpid() = 17459

poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

socket(AF\_NETLINK, SOCK\_RAW|SOCK\_CLOEXEC, NETLINK\_ROUTE) = 11

**bind(11, {sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, 12) = 0**

**getsockname(11, {sa\_family=AF\_NETLINK, nl\_pid=17459, nl\_groups=00000000}, [12]) = 0**

**sendto**(11, {{len=20, type=RTM\_GETLINK, flags=NLM\_F\_REQUEST|NLM\_F\_DUMP, seq=1704571335, pid=0}, {ifi\_family=AF\_UNSPEC, ...}}, 20, 0, {sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, 12) = 20

**recvmsg**(11, {msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, msg\_namelen=12, msg\_iov=[{iov\_base=[{{len=1320, type=RTM\_NEWLINK, flags=NLM\_F\_MULTI, seq=1704571335, pid=17459}, {ifi\_family=AF\_UNSPEC, ifi\_type=ARPHRD\_LOOPBACK, ifi\_index=if\_nametoindex("lo"), ifi\_flags=IFF\_UP|IFF\_LOOPBACK|IFF\_RUNNING|IFF\_LOWER\_UP, ifi\_change=0}, [{{nla\_len=7, nla\_type=IFLA\_IFNAME}, "lo"}, {{nla\_len=8, nla\_type=IFLA\_TXQLEN}, 1000}, {{nla\_len=5, nla\_type=IFLA\_OPERSTATE}, 0}, {{nla\_len=5, nla\_type=IFLA\_LINKMODE}, 0}, {{nla\_len=8, nla\_type=IFLA\_MTU}, 65536}, {{nla\_len=8, nla\_type=IFLA\_MIN\_MTU}, 0}, {{nla\_len=8, nla\_type=IFLA\_MAX\_MTU}, 0}, {{nla\_len=8, nla\_type=IFLA\_GROUP}, 0}, {{nla\_len=8, nla\_type=IFLA\_PROMISCUITY}, 0}, {{nla\_len=8, nla\_type=IFLA\_NUM\_TX\_QUEUES}, 1}, {{nla\_len=8, nla\_type=IFLA\_GSO\_MAX\_SEGS}, 65535}, {{nla\_len=8, nla\_type=IFLA\_GSO\_MAX\_SIZE}, 65536}, {{nla\_len=8, nla\_type=IFLA\_NUM\_RX\_QUEUES}, 1}, {{nla\_len=5, nla\_type=IFLA\_CARRIER}, 1}, {{nla\_len=12, nla\_type=IFLA\_QDISC}, "noqueue"}, {{nla\_len=8, nla\_type=IFLA\_CARRIER\_CHANGES}, 0}, {{nla\_len=8, nla\_type=IFLA\_CARRIER\_UP\_COUNT}, 0}, {{nla\_len=8, nla\_type=IFLA\_CARRIER\_DOWN\_COUNT}, 0}, {{nla\_len=5, nla\_type=IFLA\_PROTO\_DOWN}, 0}, {{nla\_len=36, nla\_type=IFLA\_MAP}, {mem\_start=0, mem\_end=0, base\_addr=0, irq=0, dma=0, port=0}}, {{nla\_len=10, nla\_type=IFLA\_ADDRESS}, "\x00\x00\x00\x00\x00\x00"}, {{nla\_len=10, nla\_type=IFLA\_BROADCAST}, "\x00\x00\x00\x00\x00\x00"}, {{nla\_len=196, nla\_type=IFLA\_STATS64}, {rx\_packets=28803, tx\_packets=28803, rx\_bytes=1483474, tx\_bytes=1483474, rx\_errors=0, tx\_errors=0, rx\_dropped=0, tx\_dropped=0, multicast=0, collisions=0, rx\_length\_errors=0, rx\_over\_errors=0, rx\_crc\_errors=0, rx\_frame\_errors=0, rx\_fifo\_errors=0, rx\_missed\_errors=0, tx\_aborted\_errors=0, tx\_carrier\_errors=0, tx\_fifo\_errors=0, tx\_heartbeat\_errors=0, tx\_window\_errors=0, rx\_compressed=0, tx\_compressed=0, rx\_nohandler=0}}, {{nla\_len=100, nla\_type=IFLA\_STATS}, {rx\_packets=28803, tx\_packets=28803, rx\_bytes=1483474, tx\_bytes=1483474, rx\_errors=0, tx\_errors=0, rx\_dropped=0, tx\_dropped=0, multicast=0, collisions=0, rx\_length\_errors=0, rx\_over\_errors=0, rx\_crc\_errors=0, rx\_frame\_errors=0, rx\_fifo\_errors=0, rx\_missed\_errors=0, tx\_aborted\_errors=0, tx\_carrier\_errors=0, tx\_fifo\_errors=0, tx\_heartbeat\_errors=0, tx\_window\_errors=0, rx\_compressed=0, tx\_compressed=0, rx\_nohandler=0}}, {{nla\_len=12, nla\_type=IFLA\_XDP}, {{nla\_len=5, nla\_type=IFLA\_XDP\_ATTACHED}, XDP\_ATTACHED\_NONE}}, {{nla\_len=764, nla\_type=IFLA\_AF\_SPEC}, [{{nla\_len=136, nla\_type=AF\_INET}, {{nla\_len=132, nla\_type=IFLA\_INET\_CONF}, [[IPV4\_DEVCONF\_FORWARDING-1] = 1, [IPV4\_DEVCONF\_MC\_FORWARDING-1] = 0, [IPV4\_DEVCONF\_PROXY\_ARP-1] = 0, [IPV4\_DEVCONF\_ACCEPT\_REDIRECTS-1] = 1, [IPV4\_DEVCONF\_SECURE\_REDIRECTS-1] = 1, [IPV4\_DEVCONF\_SEND\_REDIRECTS-1] = 1, [IPV4\_DEVCONF\_SHARED\_MEDIA-1] = 1, [IPV4\_DEVCONF\_RP\_FILTER-1] = 0, [IPV4\_DEVCONF\_ACCEPT\_SOURCE\_ROUTE-1] = 1, [IPV4\_DEVCONF\_BOOTP\_RELAY-1] = 0, [IPV4\_DEVCONF\_LOG\_MARTIANS-1] = 0, [IPV4\_DEVCONF\_TAG-1] = 0, [IPV4\_DEVCONF\_ARPFILTER-1] = 0, [IPV4\_DEVCONF\_MEDIUM\_ID-1] = 0, [IPV4\_DEVCONF\_NOXFRM-1] = 1, [IPV4\_DEVCONF\_NOPOLICY-1] = 1, [IPV4\_DEVCONF\_FORCE\_IGMP\_VERSION-1] = 0, [IPV4\_DEVCONF\_ARP\_ANNOUNCE-1] = 0, [IPV4\_DEVCONF\_ARP\_IGNORE-1] = 0, [IPV4\_DEVCONF\_PROMOTE\_SECONDARIES-1] = 0, [IPV4\_DEVCONF\_ARP\_ACCEPT-1] = 0, [IPV4\_DEVCONF\_ARP\_NOTIFY-1] = 0, [IPV4\_DEVCONF\_ACCEPT\_LOCAL-1] = 0, [IPV4\_DEVCONF\_SRC\_VMARK-1] = 0, [IPV4\_DEVCONF\_PROXY\_ARP\_PVLAN-1] = 0, [IPV4\_DEVCONF\_ROUTE\_LOCALNET-1] = 0, [IPV4\_DEVCONF\_IGMPV2\_UNSOLICITED\_REPORT\_INTERVAL-1] = 10000, [IPV4\_DEVCONF\_IGMPV3\_UNSOLICITED\_REPORT\_INTERVAL-1] = 1000, [IPV4\_DEVCONF\_IGNORE\_ROUTES\_WITH\_LINKDOWN-1] = 0, [IPV4\_DEVCONF\_DROP\_UNICAST\_IN\_L2\_MULTICAST-1] = 0, [IPV4\_DEVCONF\_DROP\_GRATUITOUS\_ARP-1] = 0, [IPV4\_DEVCONF\_BC\_FORWARDING-1] = 0]}}, {{nla\_len=624, nla\_type=AF\_INET6}, [{{nla\_len=8, nla\_type=IFLA\_INET6\_FLAGS}, IF\_READY}, {{nla\_len=20, nla\_type=IFLA\_INET6\_CACHEINFO}, {max\_reasm\_len=65535, tstamp=356, reachable\_time=24780, retrans\_time=1000}}, {{nla\_len=212, nla\_type=IFLA\_INET6\_CONF}, [[DEVCONF\_FORWARDING] = 0, [DEVCONF\_HOPLIMIT] = 64, [DEVCONF\_MTU6] = 65536, [DEVCONF\_ACCEPT\_RA] = 1, [DEVCONF\_ACCEPT\_REDIRECTS] = 1, [DEVCONF\_AUTOCONF] = 1, [DEVCONF\_DAD\_TRANSMITS] = 1, [DEVCONF\_RTR\_SOLICITS] = -1, [DEVCONF\_RTR\_SOLICIT\_INTERVAL] = 4000, [DEVCONF\_RTR\_SOLICIT\_DELAY] = 1000, [DEVCONF\_USE\_TEMPADDR] = -1, [DEVCONF\_TEMP\_VALID\_LFT] = 604800, [DEVCONF\_TEMP\_PREFERED\_LFT] = 86400, [DEVCONF\_REGEN\_MAX\_RETRY] = 3, [DEVCONF\_MAX\_DESYNC\_FACTOR] = 600, [DEVCONF\_MAX\_ADDRESSES] = 16, [DEVCONF\_FORCE\_MLD\_VERSION] = 0, [DEVCONF\_ACCEPT\_RA\_DEFRTR] = 1, [DEVCONF\_ACCEPT\_RA\_PINFO] = 1, [DEVCONF\_ACCEPT\_RA\_RTR\_PREF] = 0, [DEVCONF\_RTR\_PROBE\_INTERVAL] = 0, [DEVCONF\_ACCEPT\_RA\_RT\_INFO\_MAX\_PLEN] = 0, [DEVCONF\_PROXY\_NDP] = 0, [DEVCONF\_OPTIMISTIC\_DAD] = 0, [DEVCONF\_ACCEPT\_SOURCE\_ROUTE] = 0, [DEVCONF\_MC\_FORWARDING] = 0, [DEVCONF\_DISABLE\_IPV6] = 0, [DEVCONF\_ACCEPT\_DAD] = -1, [DEVCONF\_FORCE\_TLLAO] = 0, [DEVCONF\_NDISC\_NOTIFY] = 0, [DEVCONF\_MLDV1\_UNSOLICITED\_REPORT\_INTERVAL] = 10000, [DEVCONF\_MLDV2\_UNSOLICITED\_REPORT\_INTERVAL] = 1000, ...]}, {{nla\_len=300, nla\_type=IFLA\_INET6\_STATS}, [[IPSTATS\_MIB\_NUM] = 37, [IPSTATS\_MIB\_INPKTS] = 0, [IPSTATS\_MIB\_INOCTETS] = 0, [IPSTATS\_MIB\_INDELIVERS] = 0, [IPSTATS\_MIB\_OUTFORWDATAGRAMS] = 0, [IPSTATS\_MIB\_OUTPKTS] = 0, [IPSTATS\_MIB\_OUTOCTETS] = 0, [IPSTATS\_MIB\_INHDRERRORS] = 0, [IPSTATS\_MIB\_INTOOBIGERRORS] = 0, [IPSTATS\_MIB\_INNOROUTES] = 0, [IPSTATS\_MIB\_INADDRERRORS] = 0, [IPSTATS\_MIB\_INUNKNOWNPROTOS] = 0, [IPSTATS\_MIB\_INTRUNCATEDPKTS] = 0, [IPSTATS\_MIB\_INDISCARDS] = 0, [IPSTATS\_MIB\_OUTDISCARDS] = 0, [IPSTATS\_MIB\_OUTNOROUTES] = 0, [IPSTATS\_MIB\_REASMTIMEOUT] = 0, [IPSTATS\_MIB\_REASMREQDS] = 0, [IPSTATS\_MIB\_REASMOKS] = 0, [IPSTATS\_MIB\_REASMFAILS] = 0, [IPSTATS\_MIB\_FRAGOKS] = 0, [IPSTATS\_MIB\_FRAGFAILS] = 0, [IPSTATS\_MIB\_FRAGCREATES] = 0, [IPSTATS\_MIB\_INMCASTPKTS] = 0, [IPSTATS\_MIB\_OUTMCASTPKTS] = 0, [IPSTATS\_MIB\_INBCASTPKTS] = 0, [IPSTATS\_MIB\_OUTBCASTPKTS] = 0, [IPSTATS\_MIB\_INMCASTOCTETS] = 0, [IPSTATS\_MIB\_OUTMCASTOCTETS] = 0, [IPSTATS\_MIB\_INBCASTOCTETS] = 0, [IPSTATS\_MIB\_OUTBCASTOCTETS] = 0, [IPSTATS\_MIB\_CSUMERRORS] = 0, ...]}, {{nla\_len=52, nla\_type=IFLA\_INET6\_ICMP6STATS}, [[ICMP6\_MIB\_NUM] = 6, [ICMP6\_MIB\_INMSGS] = 0, [ICMP6\_MIB\_INERRORS] = 0, [ICMP6\_MIB\_OUTMSGS] = 0, [ICMP6\_MIB\_OUTERRORS] = 0, [ICMP6\_MIB\_CSUMERRORS] = 0]}, {{nla\_len=20, nla\_type=IFLA\_INET6\_TOKEN}, inet\_pton(AF\_INET6, "::")}, {{nla\_len=5, nla\_type=IFLA\_INET6\_ADDR\_GEN\_MODE}, IN6\_ADDR\_GEN\_MODE\_EUI64}]}]}]}, {{len=1512, type=RTM\_NEWLINK, flags=NLM\_F\_MULTI, seq=1704571335, pid=17459}, {ifi\_family=AF\_UNSPEC, ifi\_type=ARPHRD\_ETHER, ifi\_index=if\_nametoindex("bond0"), ifi\_flags=IFF\_BROADCAST|IFF\_MASTER|IFF\_MULTICAST, ifi\_change=0}, [{{nla\_len=10, nla\_type=IFLA\_IFNAME}, "bond0"}, {{nla\_len=8, nla\_type=IFLA\_TXQLEN}, 1000}, {{nla\_len=5, nla\_type=IFLA\_OPERSTATE}, 2}, {{nla\_len=5, nla\_type=IFLA\_LINKMODE}, 0}, {{nla\_len=8, nla\_type=IFLA\_MTU}, 1500}, {{nla\_len=8, nla\_type=IFLA\_MIN\_MTU}, 68}, {{nla\_len=8, nla\_type=IFLA\_MAX\_MTU}, 65535}, {{nla\_len=8, nla\_type=IFLA\_GROUP}, 0}, {{nla\_len=8, nla\_type=IFLA\_PROMISCUITY}, 0}, {{nla\_len=8, nla\_type=IFLA\_NUM\_TX\_QUEUES}, 16}, {{nla\_len=8, nla\_type=IFLA\_GSO\_MAX\_SEGS}, 65535}, {{nla\_len=8, nla\_type=IFLA\_GSO\_MAX\_SIZE}, 65536}, {{nla\_len=8, nla\_type=IFLA\_NUM\_RX\_QUEUES}, 16}, {{nla\_len=5, nla\_type=IFLA\_CARRIER}, 0}, {{nla\_len=9, nla\_type=IFLA\_QDISC}, "noop"}, {{nla\_len=8, nla\_type=IFLA\_CARRIER\_CHANGES}, 1}, {{nla\_len=8, nla\_type=IFLA\_CARRIER\_UP\_COUNT}, 0}, {{nla\_len=8, nla\_type=IFLA\_CARRIER\_DOWN\_COUNT}, 1}, {{nla\_len=5, nla\_type=IFLA\_PROTO\_DOWN}, 0}, {{nla\_len=36, nla\_type=IFLA\_MAP}, {mem\_start=0, mem\_end=0, base\_addr=0, irq=0, dma=0, port=0}}, {{nla\_len=10, nla\_type=IFLA\_ADDRESS}, "\x62\xc1\x92\xea\xf8\x53"}, {{nla\_len=10, nla\_type=IFLA\_BROADCAST}, "\xff\xff\xff\xff\xff\xff"}, {{nla\_len=196, nla\_type=IFLA\_STATS64}, {rx\_packets=0, tx\_packets=0, rx\_bytes=0, tx\_bytes=0, rx\_errors=0, tx\_errors=0, rx\_dropped=0, tx\_dropped=0, multicast=0, collisions=0, rx\_length\_errors=0, rx\_over\_errors=0, rx\_crc\_errors=0, rx\_frame\_errors=0, rx\_fifo\_errors=0, rx\_missed\_errors=0, tx\_aborted\_errors=0, tx\_carrier\_errors=0, tx\_fifo\_errors=0, tx\_heartbeat\_errors=0, tx\_window\_errors=0, rx\_compressed=0, tx\_compressed=0, rx\_nohandler=0}}, {{nla\_len=100, nla\_type=IFLA\_STATS}, {rx\_packets=0, tx\_packets=0, rx\_bytes=0, tx\_bytes=0, rx\_errors=0, tx\_errors=0, rx\_dropped=0, tx\_dropped=0, multicast=0, collisions=0, rx\_length\_errors=0, rx\_over\_errors=0, rx\_crc\_errors=0, rx\_frame\_errors=0, rx\_fifo\_errors=0, rx\_missed\_errors=0, tx\_aborted\_errors=0, tx\_carrier\_errors=0, tx\_fifo\_errors=0, tx\_heartbeat\_errors=0, tx\_window\_errors=0, rx\_compressed=0, tx\_compressed=0, rx\_nohandler=0}}, {{nla\_len=12, nla\_type=IFLA\_XDP}, {{nla\_len=5, nla\_type=IFLA\_XDP\_ATTACHED}, XDP\_ATTACHED\_NONE}}, {{nla\_len=188, nla\_type=IFLA\_LINKINFO}, [{{nla\_len=9, nla\_type=IFLA\_INFO\_KIND}, "bond"}, {{nla\_len=172, nla\_type=IFLA\_INFO\_DATA}, "\x05\x00\x01\x00\x00\x00\x00\x00\x08\x00\x03\x00\x00\x00\x00\x00\x08\x00\x04\x00\x00\x00\x00\x00\x08\x00\x05\x00\x00\x00\x00\x00"...}]}, {{nla\_len=764, nla\_type=IFLA\_AF\_SPEC}, [{{nla\_len=136, nla\_type=AF\_INET}, {{nla\_len=132, nla\_type=IFLA\_INET\_CONF}, [[IPV4\_DEVCONF\_FORWARDING-1] = 1, [IPV4\_DEVCONF\_MC\_FORWARDING-1] = 0, [IPV4\_DEVCONF\_PROXY\_ARP-1] = 0, [IPV4\_DEVCONF\_ACCEPT\_REDIRECTS-1] = 1, [IPV4\_DEVCONF\_SECURE\_REDIRECTS-1] = 1, [IPV4\_DEVCONF\_SEND\_REDIRECTS-1] = 1, [IPV4\_DEVCONF\_SHARED\_MEDIA-1] = 1, [IPV4\_DEVCONF\_RP\_FILTER-1] = 0, [IPV4\_DEVCONF\_ACCEPT\_SOURCE\_ROUTE-1] = 1, [IPV4\_DEVCONF\_BOOTP\_RELAY-1] = 0, [IPV4\_DEVCONF\_LOG\_MARTIANS-1] = 0, [IPV4\_DEVCONF\_TAG-1] = 0, [IPV4\_DEVCONF\_ARPFILTER-1] = 0, [IPV4\_DEVCONF\_MEDIUM\_ID-1] = 0, [IPV4\_DEVCONF\_NOXFRM-1] = 0, [IPV4\_DEVCONF\_NOPOLICY-1] = 0, [IPV4\_DEVCONF\_FORCE\_IGMP\_VERSION-1] = 0, [IPV4\_DEVCONF\_ARP\_ANNOUNCE-1] = 0, [IPV4\_DEVCONF\_ARP\_IGNORE-1] = 0, [IPV4\_DEVCONF\_PROMOTE\_SECONDARIES-1] = 0, [IPV4\_DEVCONF\_ARP\_ACCEPT-1] = 0, [IPV4\_DEVCONF\_ARP\_NOTIFY-1] = 0, [IPV4\_DEVCONF\_ACCEPT\_LOCAL-1] = 0, [IPV4\_DEVCONF\_SRC\_VMARK-1] = 0, [IPV4\_DEVCONF\_PROXY\_ARP\_PVLAN-1] = 0, [IPV4\_DEVCONF\_ROUTE\_LOCALNET-1] = 0, [IPV4\_DEVCONF\_IGMPV2\_UNSOLICITED\_REPORT\_INTERVAL-1] = 10000, [IPV4\_DEVCONF\_IGMPV3\_UNSOLICITED\_REPORT\_INTERVAL-1] = 1000, [IPV4\_DEVCONF\_IGNORE\_ROUTES\_WITH\_LINKDOWN-1] = 0, [IPV4\_DEVCONF\_DROP\_UNICAST\_IN\_L2\_MULTICAST-1] = 0, [IPV4\_DEVCONF\_DROP\_GRATUITOUS\_ARP-1] = 0, [IPV4\_DEVCONF\_BC\_FORWARDING-1] = 0]}}, {{nla\_len=624, nla\_type=AF\_INET6}, [{{nla\_len=8, nla\_type=IFLA\_INET6\_FLAGS}, 0}, {{nla\_len=20, nla\_type=IFLA\_INET6\_CACHEINFO}, {max\_reasm\_len=65535, tstamp=345, reachable\_time=36380, retrans\_time=1000}}, {{nla\_len=212, nla\_type=IFLA\_INET6\_CONF}, [[DEVCONF\_FORWARDING] = 0, [DEVCONF\_HOPLIMIT] = 64, [DEVCONF\_MTU6] = 1500, [DEVCONF\_ACCEPT\_RA] = 1, [DEVCONF\_ACCEPT\_REDIRECTS] = 1, [DEVCONF\_AUTOCONF] = 1, [DEVCONF\_DAD\_TRANSMITS] = 1, [DEVCONF\_RTR\_SOLICITS] = -1, [DEVCONF\_RTR\_SOLICIT\_INTERVAL] = 4000, [DEVCONF\_RTR\_SOLICIT\_DELAY] = 1000, [DEVCONF\_USE\_TEMPADDR] = 0, [DEVCONF\_TEMP\_VALID\_LFT] = 604800, [DEVCONF\_TEMP\_PREFERED\_LFT] = 86400, [DEVCONF\_REGEN\_MAX\_RETRY] = 3, [DEVCONF\_MAX\_DESYNC\_FACTOR] = 600, [DEVCONF\_MAX\_ADDRESSES] = 16, [DEVCONF\_FORCE\_MLD\_VERSION] = 0, [DEVCONF\_ACCEPT\_RA\_DEFRTR] = 1, [DEVCONF\_ACCEPT\_RA\_PINFO] = 1, [DEVCONF\_ACCEPT\_RA\_RTR\_PREF] = 0, [DEVCONF\_RTR\_PROBE\_INTERVAL] = 0, [DEVCONF\_ACCEPT\_RA\_RT\_INFO\_MAX\_PLEN] = 0, [DEVCONF\_PROXY\_NDP] = 0, [DEVCONF\_OPTIMISTIC\_DAD] = 0, [DEVCONF\_ACCEPT\_SOURCE\_ROUTE] = 0, [DEVCONF\_MC\_FORWARDING] = 0, [DEVCONF\_DISABLE\_IPV6] = 0, [DEVCONF\_ACCEPT\_DAD] = 1, [DEVCONF\_FORCE\_TLLAO] = 0, [DEVCONF\_NDISC\_NOTIFY] = 0, [DEVCONF\_MLDV1\_UNSOLICITED\_REPORT\_INTERVAL] = 10000, [DEVCONF\_MLDV2\_UNSOLICITED\_REPORT\_INTERVAL] = 1000, ...]}, {{nla\_len=300, nla\_type=IFLA\_INET6\_STATS}, [[IPSTATS\_MIB\_NUM] = 37, [IPSTATS\_MIB\_INPKTS] = 0, [IPSTATS\_MIB\_INOCTETS] = 0, [IPSTATS\_MIB\_INDELIVERS] = 0, [IPSTATS\_MIB\_OUTFORWDATAGRAMS] = 0, [IPSTATS\_MIB\_OUTPKTS] = 0, [IPSTATS\_MIB\_OUTOCTETS] = 0, [IPSTATS\_MIB\_INHDRERRORS] = 0, [IPSTATS\_MIB\_INTOOBIGERRORS] = 0, [IPSTATS\_MIB\_INNOROUTES] = 0, [IPSTATS\_MIB\_INADDRERRORS] = 0, [IPSTATS\_MIB\_INUNKNOWNPROTOS] = 0, [IPSTATS\_MIB\_INTRUNCATEDPKTS] = 0, [IPSTATS\_MIB\_INDISCARDS] = 0, [IPSTATS\_MIB\_OUTDISCARDS] = 0, [IPSTATS\_MIB\_OUTNOROUTES] = 0, [IPSTATS\_MIB\_REASMTIMEOUT] = 0, [IPSTATS\_MIB\_REASMREQDS] = 0, [IPSTATS\_MIB\_REASMOKS] = 0, [IPSTATS\_MIB\_REASMFAILS] = 0, [IPSTATS\_MIB\_FRAGOKS] = 0, [IPSTATS\_MIB\_FRAGFAILS] = 0, [IPSTATS\_MIB\_FRAGCREATES] = 0, [IPSTATS\_MIB\_INMCASTPKTS] = 0, [IPSTATS\_MIB\_OUTMCASTPKTS] = 0, [IPSTATS\_MIB\_INBCASTPKTS] = 0, [IPSTATS\_MIB\_OUTBCASTPKTS] = 0, [IPSTATS\_MIB\_INMCASTOCTETS] = 0, [IPSTATS\_MIB\_OUTMCASTOCTETS] = 0, [IPSTATS\_MIB\_INBCASTOCTETS] = 0, [IPSTATS\_MIB\_OUTBCASTOCTETS] = 0, [IPSTATS\_MIB\_CSUMERRORS] = 0, ...]}, {{nla\_len=52, nla\_type=IFLA\_INET6\_ICMP6STATS}, [[ICMP6\_MIB\_NUM] = 6, [ICMP6\_MIB\_INMSGS] = 0, [ICMP6\_MIB\_INERRORS] = 0, [ICMP6\_MIB\_OUTMSGS] = 0, [ICMP6\_MIB\_OUTERRORS] = 0, [ICMP6\_MIB\_CSUMERRORS] = 0]}, {{nla\_len=20, nla\_type=IFLA\_INET6\_TOKEN}, inet\_pton(AF\_INET6, "::")}, {{nla\_len=5, nla\_type=IFLA\_INET6\_ADDR\_GEN\_MODE}, IN6\_ADDR\_GEN\_MODE\_EUI64}]}]}]}], iov\_len=4096}], msg\_iovlen=1, msg\_controllen=0, msg\_flags=0}, 0) = 2832

**sendto**(11, {{len=20, type=RTM\_GETADDR, flags=NLM\_F\_REQUEST|NLM\_F\_DUMP, seq=1704571336, pid=0}, {ifa\_family=AF\_UNSPEC, ...}}, 20, 0, {sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, 12) = 20

recvmsg(11, {msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, msg\_namelen=12, msg\_iov=[{iov\_base=[{{len=76, type=RTM\_NEWADDR, flags=NLM\_F\_MULTI, seq=1704571336, pid=17459}, {ifa\_family=AF\_INET, ifa\_prefixlen=8, ifa\_flags=IFA\_F\_PERMANENT, ifa\_scope=RT\_SCOPE\_HOST, ifa\_index=if\_nametoindex("lo")}, [{{nla\_len=8, nla\_type=IFA\_ADDRESS}, inet\_addr("127.0.0.1")}, {{nla\_len=8, nla\_type=IFA\_LOCAL}, inet\_addr("127.0.0.1")}, {{nla\_len=7, nla\_type=IFA\_LABEL}, "lo"}, {{nla\_len=8, nla\_type=IFA\_FLAGS}, IFA\_F\_PERMANENT}, {{nla\_len=20, nla\_type=IFA\_CACHEINFO}, {ifa\_prefered=4294967295, ifa\_valid=4294967295, cstamp=356, tstamp=356}}]}, {{len=88, type=RTM\_NEWADDR, flags=NLM\_F\_MULTI, seq=1704571336, pid=17459}, {ifa\_family=AF\_INET, ifa\_prefixlen=20, ifa\_flags=IFA\_F\_PERMANENT, ifa\_scope=RT\_SCOPE\_UNIVERSE, ifa\_index=if\_nametoindex("eth0")}, [{{nla\_len=8, nla\_type=IFA\_ADDRESS}, inet\_addr("172.19.183.9")}, {{nla\_len=8, nla\_type=IFA\_LOCAL}, inet\_addr("172.19.183.9")}, {{nla\_len=8, nla\_type=IFA\_BROADCAST}, inet\_addr("172.19.191.255")}, {{nla\_len=9, nla\_type=IFA\_LABEL}, "eth0"}, {{nla\_len=8, nla\_type=IFA\_FLAGS}, IFA\_F\_PERMANENT}, {{nla\_len=20, nla\_type=IFA\_CACHEINFO}, {ifa\_prefered=4294967295, ifa\_valid=4294967295, cstamp=458, tstamp=458}}]}], iov\_len=4096}], msg\_iovlen=1, msg\_controllen=0, msg\_flags=0}, 0) = 164

recvmsg(11, {msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, msg\_namelen=12, msg\_iov=[{iov\_base=[{{len=72, type=RTM\_NEWADDR, flags=NLM\_F\_MULTI, seq=1704571336, pid=17459}, {ifa\_family=AF\_INET6, ifa\_prefixlen=128, ifa\_flags=IFA\_F\_PERMANENT, ifa\_scope=RT\_SCOPE\_HOST, ifa\_index=if\_nametoindex("lo")}, [{{nla\_len=20, nla\_type=IFA\_ADDRESS}, inet\_pton(AF\_INET6, "::1")}, {{nla\_len=20, nla\_type=IFA\_CACHEINFO}, {ifa\_prefered=4294967295, ifa\_valid=4294967295, cstamp=356, tstamp=356}}, {{nla\_len=8, nla\_type=IFA\_FLAGS}, IFA\_F\_PERMANENT}]}, {{len=72, type=RTM\_NEWADDR, flags=NLM\_F\_MULTI, seq=1704571336, pid=17459}, {ifa\_family=AF\_INET6, ifa\_prefixlen=64, ifa\_flags=IFA\_F\_PERMANENT, ifa\_scope=RT\_SCOPE\_LINK, ifa\_index=if\_nametoindex("eth0")}, [{{nla\_len=20, nla\_type=IFA\_ADDRESS}, inet\_pton(AF\_INET6, "fe80::215:5dff:fe9b:12b8")}, {{nla\_len=20, nla\_type=IFA\_CACHEINFO}, {ifa\_prefered=4294967295, ifa\_valid=4294967295, cstamp=459, tstamp=459}}, {{nla\_len=8, nla\_type=IFA\_FLAGS}, IFA\_F\_PERMANENT}]}], iov\_len=4096}], msg\_iovlen=1, msg\_controllen=0, msg\_flags=0}, 0) = 144

recvmsg(11, {msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, msg\_namelen=12, msg\_iov=[{iov\_base={{len=20, type=NLMSG\_DONE, flags=NLM\_F\_MULTI, seq=1704571336, pid=17459}, 0}, iov\_len=4096}], msg\_iovlen=1, msg\_controllen=0, msg\_flags=0}, 0) = 20

close(11) = 0

**socket(AF\_INET, SOCK\_STREAM|SOCK\_CLOEXEC, IPPROTO\_TCP) = 11**

setsockopt(11, SOL\_SOCKET, SO\_REUSEADDR, [1], 4) = 0

**bind(11, {sa\_family=AF\_INET, sin\_port=htons(1028), sin\_addr=inet\_addr("127.0.0.1")}, 16) = 0**

listen(11, 100) = 0

getsockname(11, {sa\_family=AF\_INET, sin\_port=htons(1028), sin\_addr=inet\_addr("127.0.0.1")}, [128->16]) = 0

getsockname(11, {sa\_family=AF\_INET, sin\_port=htons(1028), sin\_addr=inet\_addr("127.0.0.1")}, [128->16]) = 0

getpid() = 17459

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17459

write(8, "\1\0\0\0\0\0\0\0", 8) = 8

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f2abc66ca10) = 17462

getpid() = 17459

poll([{fd=8, events=POLLIN}], 1, 0) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 17459

read(8, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17459

poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17459

poll([{fd=8, events=POLLIN}], 1, 3000Работает Client.cpp

) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 17459

read(8, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17459

poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17459

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17459

poll([{fd=8, events=POLLIN}], 1, -1) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 17459

read(8, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17459

poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17459

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

write(1, "Ok: 17462\n", 10Ok: 17462

) = 10

write(1, "\320\232\320\276\320\275\320\265\321\206 \321\200\320\260\320\261\320\276\321\202\321\213 \321\201\320\265\321\200\320\262"..., 46Конец работы сервера create

) = 46

write(1, "$> ", 3$> ) = 3

read(0, create 3

"create 3\n", 1024) = 9

write(1, "\320\235\320\260\321\207\320\260\320\273\320\276 \321\200\320\260\320\261\320\276\321\202\321\213 \321\201\320\265\321\200"..., 48Начало работы сервера create

) = 48

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

socket(AF\_NETLINK, SOCK\_RAW|SOCK\_CLOEXEC, NETLINK\_ROUTE) = 13

bind(13, {sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, 12) = 0

getsockname(13, {sa\_family=AF\_NETLINK, nl\_pid=17435, nl\_groups=00000000}, [12]) = 0

sendto(13, {{len=20, type=RTM\_GETLINK, flags=NLM\_F\_REQUEST|NLM\_F\_DUMP, seq=1704571085, pid=0}, {ifi\_family=AF\_UNSPEC, ...}}, 20, 0, {sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, 12) = 20

recvmsg(13, {msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, msg\_namelen=12, msg\_iov=[{iov\_base=[{{len=1320, type=RTM\_NEWLINK, flags=NLM\_F\_MULTI, seq=1704571085, pid=17435}, {ifi\_family=AF\_UNSPEC, ifi\_type=ARPHRD\_LOOPBACK, ifi\_index=if\_nametoindex("lo"), ifi\_flags=IFF\_UP|IFF\_LOOPBACK|IFF\_RUNNING|IFF\_LOWER\_UP, ifi\_change=0}, [{{nla\_len=7, nla\_type=IFLA\_IFNAME}, "lo"}, {{nla\_len=8, nla\_type=IFLA\_TXQLEN}, 1000}, {{nla\_len=5, nla\_type=IFLA\_OPERSTATE}, 0}, {{nla\_len=5, nla\_type=IFLA\_LINKMODE}, 0}, {{nla\_len=8, nla\_type=IFLA\_MTU}, 65536}, {{nla\_len=8, nla\_type=IFLA\_MIN\_MTU}, 0}, {{nla\_len=8, nla\_type=IFLA\_MAX\_MTU}, 0}, {{nla\_len=8, nla\_type=IFLA\_GROUP}, 0}, {{nla\_len=8, nla\_type=IFLA\_PROMISCUITY}, 0}, {{nla\_len=8, nla\_type=IFLA\_NUM\_TX\_QUEUES}, 1}, {{nla\_len=8, nla\_type=IFLA\_GSO\_MAX\_SEGS}, 65535}, {{nla\_len=8, nla\_type=IFLA\_GSO\_MAX\_SIZE}, 65536}, {{nla\_len=8, nla\_type=IFLA\_NUM\_RX\_QUEUES}, 1}, {{nla\_len=5, nla\_type=IFLA\_CARRIER}, 1}, {{nla\_len=12, nla\_type=IFLA\_QDISC}, "noqueue"}, {{nla\_len=8, nla\_type=IFLA\_CARRIER\_CHANGES}, 0}, {{nla\_len=8, nla\_type=IFLA\_CARRIER\_UP\_COUNT}, 0}, {{nla\_len=8, nla\_type=IFLA\_CARRIER\_DOWN\_COUNT}, 0}, {{nla\_len=5, nla\_type=IFLA\_PROTO\_DOWN}, 0}, {{nla\_len=36, nla\_type=IFLA\_MAP}, {mem\_start=0, mem\_end=0, base\_addr=0, irq=0, dma=0, port=0}}, {{nla\_len=10, nla\_type=IFLA\_ADDRESS}, "\x00\x00\x00\x00\x00\x00"}, {{nla\_len=10, nla\_type=IFLA\_BROADCAST}, "\x00\x00\x00\x00\x00\x00"}, {{nla\_len=196, nla\_type=IFLA\_STATS64}, {rx\_packets=28694, tx\_packets=28694, rx\_bytes=1476999, tx\_bytes=1476999, rx\_errors=0, tx\_errors=0, rx\_dropped=0, tx\_dropped=0, multicast=0, collisions=0, rx\_length\_errors=0, rx\_over\_errors=0, rx\_crc\_errors=0, rx\_frame\_errors=0, rx\_fifo\_errors=0, rx\_missed\_errors=0, tx\_aborted\_errors=0, tx\_carrier\_errors=0, tx\_fifo\_errors=0, tx\_heartbeat\_errors=0, tx\_window\_errors=0, rx\_compressed=0, tx\_compressed=0, rx\_nohandler=0}}, {{nla\_len=100, nla\_type=IFLA\_STATS}, {rx\_packets=28694, tx\_packets=28694, rx\_bytes=1476999, tx\_bytes=1476999, rx\_errors=0, tx\_errors=0, rx\_dropped=0, tx\_dropped=0, multicast=0, collisions=0, rx\_length\_errors=0, rx\_over\_errors=0, rx\_crc\_errors=0, rx\_frame\_errors=0, rx\_fifo\_errors=0, rx\_missed\_errors=0, tx\_aborted\_errors=0, tx\_carrier\_errors=0, tx\_fifo\_errors=0, tx\_heartbeat\_errors=0, tx\_window\_errors=0, rx\_compressed=0, tx\_compressed=0, rx\_nohandler=0}}, {{nla\_len=12, nla\_type=IFLA\_XDP}, {{nla\_len=5, nla\_type=IFLA\_XDP\_ATTACHED}, XDP\_ATTACHED\_NONE}}, {{nla\_len=764, nla\_type=IFLA\_AF\_SPEC}, [{{nla\_len=136, nla\_type=AF\_INET}, {{nla\_len=132, nla\_type=IFLA\_INET\_CONF}, [[IPV4\_DEVCONF\_FORWARDING-1] = 1, [ICMP6\_MIB\_CSUMERRORS] = 0]}, {{nla\_len=20, nla\_type=IFLA\_INET6\_TOKEN}, inet\_pton(AF\_INET6, "::")}, {{nla\_len=5, nla\_type=IFLA\_INET6\_ADDR\_GEN\_MODE}, IN6\_ADDR\_GEN\_MODE\_EUI64}]}]}]}], iov\_len=4096}], msg\_iovlen=1, msg\_controllen=0, msg\_flags=0}, 0) = 2832

recvmsg(13, {msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, msg\_namelen=12, msg\_iov=[{iov\_base=[{{len=1340, type=RTM\_NEWLINK, flags=NLM\_F\_MULTI, seq=1704571085, pid=17435}, {ifi\_family=AF\_UNSPEC, ifi\_type=ARPHRD\_ETHER, ifi\_index=if\_nametoindex("dummy0"), ifi\_flags=IFF\_BROADCAST|IFF\_NOARP, ifi\_change=0}, [{{nla\_len=11, nla\_type=IFLA\_IFNAME}, "dummy0"}, {{nla\_len=8, nla\_type=IFLA\_TXQLEN}, 1000}, {{nla\_len=5, nla\_type=IFLA\_OPERSTATE}, 2}, {{nla\_len=5, nla\_type=IFLA\_LINKMODE}, 0}, {{nla\_len=8, nla\_type=IFLA\_MTU}, 1500}, {{nla\_len=8, nla\_type=IFLA\_MIN\_MTU}, 0}, {{nla\_len=8, nla\_type=IFLA\_MAX\_MTU}, 0}, {{nla\_len=8, nla\_type=IFLA\_GROUP}, 0}, {{nla\_len=8, nla\_type=IFLA\_PROMISCUITY}, 0}, {{nla\_len=8, nla\_type=IFLA\_NUM\_TX\_QUEUES}, 1}, {{nla\_len=8, nla\_type=IFLA\_GSO\_MAX\_SEGS}, 65535}, {{nla\_len=8, nla\_type=IFLA\_GSO\_MAX\_SIZE}, 65536}, {{nla\_len=8, nla\_type=IFLA\_NUM\_RX\_QUEUES}, 1}, {{nla\_len=5, nla\_type=IFLA\_CARRIER}, 1}, {{nla\_len=9, nla\_type=IFLA\_QDISC}, "noop"}, {{nla\_len=8, nla\_type=IFLA\_CARRIER\_CHANGES}, 0}, {{nla\_len=8, nla\_type=IFLA\_CARRIER\_UP\_COUNT}, 0}, {{nla\_len=8, nla\_type=IFLA\_CARRIER\_DOWN\_COUNT}, 0}, {{nla\_len=5, nla\_type=IFLA\_PROTO\_DOWN}, 0}, {{nla\_len=36, nla\_type=IFLA\_MAP}, {mem\_start=0, mem\_end=0, base\_addr=0, irq=0, dma=0, port=0}}, {{nla\_len=10, nla\_type=IFLA\_ADDRESS}, "\x6a\x9c\xd6\x6f\x41\x12"}, {{nla\_len=10, nla\_type=IFLA\_BROADCAST}, "\xff\xff\xff\xff\xff\xff"}, {{nla\_len=196, nla\_type=IFLA\_STATS64}, {rx\_packets=0, tx\_packets=0, rx\_bytes=0, tx\_bytes=0, rx\_errors=0, tx\_errors=0, rx\_dropped=0, tx\_dropped=0, multicast=0, collisions=0, rx\_length\_errors=0, rx\_over\_errors=0, rx\_crc\_errors=0, rx\_frame\_errors=0, rx\_fifo\_errors=0, rx\_missed\_errors=0, tx\_aborted\_errors=0, tx\_carrier\_errors=0, tx\_fifo\_errors=0, tx\_heartbeat\_errors=0, tx\_window\_errors=0, rx\_compressed=0, tx\_compressed=0, rx\_nohandler=0}}, {{nla\_len=100, nla\_type=IFLA\_STATS}, {rx\_packets=0, tx\_packets=0, rx\_bytes=0, tx\_bytes=0, rx\_errors=0, tx\_errors=0, rx\_dropped=0, tx\_dropped=0, multicast=0, collisions=0, rx\_length\_errors=0, rx\_over\_errors=0, rx\_crc\_errors=0, rx\_frame\_errors=0, rx\_fifo\_errors=0, rx\_missed\_errors=0, tx\_aborted\_errors=0, tx\_carrier\_errors=0, tx\_fifo\_errors=0, tx\_heartbeat\_errors=0, tx\_window\_errors=0, rx\_compressed=0, tx\_compressed=0, rx\_nohandler=0}}, {{nla\_len=12, nla\_type=IFLA\_XDP}, {{nla\_len=5, nla\_type=IFLA\_XDP\_ATTACHED}, XDP\_ATTACHED\_NONE}}, {{nla\_len=16, nla\_type=IFLA\_INET6\_ICMP6STATS}, [[ICMP6\_MIB\_NUM] = 6, [ICMP6\_MIB\_INMSGS] = 1, [ICMP6\_MIB\_INERRORS] = 0, [ICMP6\_MIB\_OUTMSGS] = 216, [ICMP6\_MIB\_OUTERRORS] = 0, [ICMP6\_MIB\_CSUMERRORS] = 0]}, {{nla\_len=20, nla\_type=IFLA\_INET6\_TOKEN}, inet\_pton(AF\_INET6, "::")}, {{nla\_len=5, nla\_type=IFLA\_INET6\_ADDR\_GEN\_MODE}, IN6\_ADDR\_GEN\_MODE\_EUI64}]}]}]}], iov\_len=4096}], msg\_iovlen=1, msg\_controllen=0, msg\_flags=0}, 0) = 2776

recvmsg(13, {msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, msg\_namelen=12, msg\_iov=[{iov\_base={{len=20, type=NLMSG\_DONE, flags=NLM\_F\_MULTI, seq=1704571085, pid=17435}, 0}, iov\_len=4096}], msg\_iovlen=1, msg\_controllen=0, msg\_flags=0}, 0) = 20

sendto(13, {{len=20, type=RTM\_GETADDR, flags=NLM\_F\_REQUEST|NLM\_F\_DUMP, seq=1704571086, pid=0}, {ifa\_family=AF\_UNSPEC, ...}}, 20, 0, {sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, 12) = 20

recvmsg(13, {msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, msg\_namelen=12, msg\_iov=[{iov\_base=[{{len=76, type=RTM\_NEWADDR, flags=NLM\_F\_MULTI, seq=1704571086, pid=17435}, {ifa\_family=AF\_INET, ifa\_prefixlen=8, ifa\_flags=IFA\_F\_PERMANENT, ifa\_scope=RT\_SCOPE\_HOST, ifa\_index=if\_nametoindex("lo")}, [{{nla\_len=8, nla\_type=IFA\_ADDRESS}, inet\_addr("127.0.0.1")}, {{nla\_len=8, nla\_type=IFA\_LOCAL}, inet\_addr("127.0.0.1")}, {{nla\_len=7, nla\_type=IFA\_LABEL}, "lo"}, {{nla\_len=8, nla\_type=IFA\_FLAGS}, IFA\_F\_PERMANENT}, {{nla\_len=20, nla\_type=IFA\_CACHEINFO}, {ifa\_prefered=4294967295, ifa\_valid=4294967295, cstamp=356, tstamp=356}}]}, {{len=88, type=RTM\_NEWADDR, flags=NLM\_F\_MULTI, seq=1704571086, pid=17435}, {ifa\_family=AF\_INET, ifa\_prefixlen=20, ifa\_flags=IFA\_F\_PERMANENT, ifa\_scope=RT\_SCOPE\_UNIVERSE, ifa\_index=if\_nametoindex("eth0")}, [{{nla\_len=8, nla\_type=IFA\_ADDRESS}, inet\_addr("172.19.183.9")}, {{nla\_len=8, nla\_type=IFA\_LOCAL}, inet\_addr("172.19.183.9")}, {{nla\_len=8, nla\_type=IFA\_BROADCAST}, inet\_addr("172.19.191.255")}, {{nla\_len=9, nla\_type=IFA\_LABEL}, "eth0"}, {{nla\_len=8, nla\_type=IFA\_FLAGS}, IFA\_F\_PERMANENT}, {{nla\_len=20, nla\_type=IFA\_CACHEINFO}, {ifa\_prefered=4294967295, ifa\_valid=4294967295, cstamp=458, tstamp=458}}]}], iov\_len=4096}], msg\_iovlen=1, msg\_controllen=0, msg\_flags=0}, 0) = 164

recvmsg(13, {msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, msg\_namelen=12, msg\_iov=[{iov\_base=[{{len=72, type=RTM\_NEWADDR, flags=NLM\_F\_MULTI, seq=1704571086, pid=17435}, {ifa\_family=AF\_INET6, ifa\_prefixlen=128, ifa\_flags=IFA\_F\_PERMANENT, ifa\_scope=RT\_SCOPE\_HOST, ifa\_index=if\_nametoindex("lo")}, [{{nla\_len=20, nla\_type=IFA\_ADDRESS}, inet\_pton(AF\_INET6, "::1")}, {{nla\_len=20, nla\_type=IFA\_CACHEINFO}, {ifa\_prefered=4294967295, ifa\_valid=4294967295, cstamp=356, tstamp=356}}, {{nla\_len=8, nla\_type=IFA\_FLAGS}, IFA\_F\_PERMANENT}]}, {{len=72, type=RTM\_NEWADDR, flags=NLM\_F\_MULTI, seq=1704571086, pid=17435}, {ifa\_family=AF\_INET6, ifa\_prefixlen=64, ifa\_flags=IFA\_F\_PERMANENT, ifa\_scope=RT\_SCOPE\_LINK, ifa\_index=if\_nametoindex("eth0")}, [{{nla\_len=20, nla\_type=IFA\_ADDRESS}, inet\_pton(AF\_INET6, "fe80::215:5dff:fe9b:12b8")}, {{nla\_len=20, nla\_type=IFA\_CACHEINFO}, {ifa\_prefered=4294967295, ifa\_valid=4294967295, cstamp=459, tstamp=459}}, {{nla\_len=8, nla\_type=IFA\_FLAGS}, IFA\_F\_PERMANENT}]}], iov\_len=4096}], msg\_iovlen=1, msg\_controllen=0, msg\_flags=0}, 0) = 144

recvmsg(13, {msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, msg\_namelen=12, msg\_iov=[{iov\_base={{len=20, type=NLMSG\_DONE, flags=NLM\_F\_MULTI, seq=1704571086, pid=17435}, 0}, iov\_len=4096}], msg\_iovlen=1, msg\_controllen=0, msg\_flags=0}, 0) = 20

close(13) = 0

socket(AF\_INET, SOCK\_STREAM|SOCK\_CLOEXEC, IPPROTO\_TCP) = 13

setsockopt(13, SOL\_SOCKET, SO\_REUSEADDR, [1], 4) = 0

bind(13, {sa\_family=AF\_INET, sin\_port=htons(1027), sin\_addr=inet\_addr("127.0.0.1")}, 16) = 0

listen(13, 100) = 0

getsockname(13, {sa\_family=AF\_INET, sin\_port=htons(1027), sin\_addr=inet\_addr("127.0.0.1")}, [128->16]) = 0

getsockname(13, {sa\_family=AF\_INET, sin\_port=htons(1027), sin\_addr=inet\_addr("127.0.0.1")}, [128->16]) = 0

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

write(9, "\1\0\0\0\0\0\0\0", 8) = 8

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f3681b0ca10) = 17450

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 1 ([{fd=9, revents=POLLIN}])

getpid() = 17435

read(9, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 3000Работает Client.cpp

) = 1 ([{fd=9, revents=POLLIN}])

getpid() = 17435

read(9, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, -1) = 1 ([{fd=9, revents=POLLIN}])

getpid() = 17435

read(9, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

write(1, "Ok: 17450\n", 10Ok: 17450

) = 10

write(1, "\320\232\320\276\320\275\320\265\321\206 \321\200\320\260\320\261\320\276\321\202\321\213 \321\201\320\265\321\200\320\262"..., 46Конец работы сервера create

) = 46

write(1, "$> ", 3$> ) = 3

read(0, create 2

"create 2\n", 1024) = 9

write(1, "\320\235\320\260\321\207\320\260\320\273\320\276 \321\200\320\260\320\261\320\276\321\202\321\213 \321\201\320\265\321\200"..., 48Начало работы сервера create

) = 48

getpid() = 17435

poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=8, events=POLLIN}], 1, -1) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 17435

read(8, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

Creating...

getpid() = 17435

poll([{fd=8, events=POLLIN}], 1, -1Работает Client.cpp

Done

) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 17435

read(8, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

write(1, "Ok: 17453\n", 10Ok: 17453

) = 10

write(1, "\320\232\320\276\320\275\320\265\321\206 \321\200\320\260\320\261\320\276\321\202\321\213 \321\201\320\265\321\200\320\262"..., 46Конец работы сервера create

) = 46

write(1, "$> ", 3$> ) = 3

read(0, kill 4

"kill 4\n", 1024) = 7

getpid() = 17435

poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=8, events=POLLIN}], 1, -1) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 17435

read(8, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=8, events=POLLIN}], 1, -1) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 17435

read(8, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

poll([{fd=8, events=POLLIN}], 1, 0) = 0 (Timeout)

write(1, "ok\n", 3ok

) = 3

write(1, "$> ", 3$> ) = 3

read(0, 0x55b9b1225300, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=17447, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=6} ---

read(0, ping 4

"ping 4\n", 1024) = 7

write(1, "ok: 0\n", 6ok: 0

) = 6

write(1, "$> ", 3$> ) = 3

read(0, ping 3

"ping 3\n", 1024) = 7

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, -1) = 1 ([{fd=9, revents=POLLIN}])

getpid() = 17435

read(9, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, -1) = 1 ([{fd=9, revents=POLLIN}])

getpid() = 17435

read(9, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

write(1, "Ok: 1\n", 6Ok: 1

) = 6

write(1, "$> ", 3$> ) = 3

read(0, ping 2

"ping 2\n", 1024) = 7

write(1, "ok: 0\n", 6ok: 0

) = 6

write(1, "$> ", 3$> ) = 3

read(0, exec 3 start

"exec 3 start\n", 1024) = 13

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, -1) = 1 ([{fd=9, revents=POLLIN}])

getpid() = 17435

read(9, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, -1) = 1 ([{fd=9, revents=POLLIN}])

getpid() = 17435

read(9, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

write(1, "Ok:3\n", 5Ok:3

) = 5

write(1, "$> ", 3$> ) = 3

read(0, exec 3 stop

"exec 3 stop\n", 1024) = 12

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, -1) = 1 ([{fd=9, revents=POLLIN}])

getpid() = 17435

read(9, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, -1) = 1 ([{fd=9, revents=POLLIN}])

getpid() = 17435

read(9, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

write(1, "Ok:3\n", 5Ok:3

) = 5

write(1, "$> ", 3$> ) = 3

read(0, exec 3 time

"exec 3 time\n", 1024) = 12

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, -1) = 1 ([{fd=9, revents=POLLIN}])

getpid() = 17435

read(9, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, -1) = 1 ([{fd=9, revents=POLLIN}])

getpid() = 17435

read(9, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

write(1, "Ok:3:15069\n", 11Ok:3:15069

) = 11

write(1, "$> ", 3$> ) = 3

read(0, exec 3 time

"exec 3 time\n", 1024) = 12

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, -1) = 1 ([{fd=9, revents=POLLIN}])

getpid() = 17435

read(9, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, -1) = 1 ([{fd=9, revents=POLLIN}])

getpid() = 17435

read(9, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

write(1, "Ok:3:15069\n", 11Ok:3:15069

) = 11

write(1, "$> ", 3$> ) = 3

read(0, exit

"exit\n", 1024) = 5

write(1, "\320\222\321\213\321\205\320\276\320\264\320\270\320\274 \320\270\320\267 \320\277\321\200\320\276\320\263\321\200\320\260"..., 42Выходим из программы :3

) = 42

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, -1) = 1 ([{fd=9, revents=POLLIN}])

getpid() = 17435

read(9, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, -1) = 1 ([{fd=9, revents=POLLIN}])

getpid() = 17435

read(9, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = ? ERESTART\_RESTARTBLOCK (Interrupted by signal)

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=17450, si\_uid=1000, si\_status=0, si\_utime=2, si\_stime=2} ---

restart\_syscall(<... resuming interrupted poll ...>) = 0

getpid() = 17435

poll([{fd=9, events=POLLIN}], 1, 0) = 0 (Timeout)

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

write(4, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

write(4, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

write(4, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

getpid() = 17435

write(4, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

poll([{fd=3, events=POLLIN}], 1, -1) = 1 ([{fd=3, revents=POLLIN}])

getpid() = 17435

read(3, "\1\0\0\0\0\0\0\0", 8) = 8

getpid() = 17435

write(6, "\1\0\0\0\0\0\0\0", 8) = 8

close(7) = 0

close(6) = 0

close(5) = 0

close(4) = 0

close(3) = 0

lseek(0, -1, SEEK\_CUR) = -1 ESPIPE (Illegal seek)

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

В ходе выполнения лабораторной работы я познакомилась с очередями сообщений, а конкретно ZeroMQ, она достаточно простая в использовании и в Интернете можно найти много полезной информации об этой очереди. Я разобралась с тем, как применить деревья для хранения узлов, что было довольно трудно для меня. Эта лабораторная работа была самой сложной для меня в курсе “Операционные системы”, но я старалась приложить все силы, чтобы разобраться с работой распределенной системы по асинхронной обработке запросов. Не могу не отметить, что после первого семестра второго курса и после выполнения 19 лабораторных работ по программированию в целом, я действительно многому научилась.