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

Лабораторная работа №5-7 по курсу
«Операционные системы»

Студент: Тутаев Владимир Владимирович
Группа: М8О-201Б-23
Вариант: 29

Преподаватель: Миронов Евгений Сергеевич

Оценка: _____
Дата: _____
Подпись: _____

Москва, 2024

Репозиторий

https://github.com/Volan4ik/MAI_OS.git

Постановка задачи

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 узла
- Удаление существующего узла
- Выполнение функции
- Проверка доступности узлов

Общие сведения о программе

Все вычислительные узлы хранятся в бинарном дереве поиска. [parent] — является необязательным параметром.

Набор команд (подсчет суммы n чисел):

Формат команды: exec id n k1 ... kn
id – целочисленный идентификатор вычислительного узла, на который отправляется команда

n – количество складываемых чисел (от 1 до 108)
k1 ... kn – складываемые числа

Формат команды: heartbit time

Каждый узел начинает сообщать раз в time миллисекунд о том, что он работоспособен. Если от узла нет сигнала в течении 4*time миллисекунд, то должна выводится пользователю строка: «Heartbit: node id is unavailable now», где id – идентификатор недоступного вычислительного узла.

Исходный код в Приложении 1

Strace в Приложении 2

Выводы

При работе с данной лабораторной работой я познакомился с библиотекой ZeroMQ, которая является мощным элементом для взаимодействия между различными частями приложения, для использования очереди сообщений.

Приложение 1

message.h

```
#ifndef _WRAP_ZMQ_H
#define _WRAP_ZMQ_H

#include <tuple>
#include <vector>
#include <atomic>
#include <string>
#include "/opt/homebrew/include/zmq.h"

using namespace std;

#define UNIVERSAL_MESSAGE (-1)
#define SERVER_ID (-2)
#define PARENT_SIGNAL (-3)

enum struct SocketType {
    PUBLISHER,
    SUBSCRIBER,
};

enum struct CommandType {
    ERROR,
    RETURN,
    CREATE_CHILD,
    REMOVE_CHILD,
    EXEC_CHILD,
};

enum struct AddressType {
    CHILD_PUB_LEFT,
    CHILD_PUB_RIGHT,
    PARENT_PUB,
};

#define MAX_CAP 1000

class Message {
protected:
    static std::atomic<int> counter;
public:
    CommandType command = CommandType::ERROR;
    int toIndex;
    int createIndex;
    int uniqueIndex;
    bool withoutProcessing;
    int size = 0;
    double value[MAX_CAP] = {0};

    Message();
```

```
    Message(CommandType command, int toIndex, int size, const double *value, int createIndex);

    Message(CommandType new_command, int new_to_id, int new_id);

    Message(string str){
        if (str == "error"){
            createIndex = -10;
        }
    }

    friend bool operator==(const Message &lhs, const Message &rhs);

    int &getCreateIndex();

    int &getToIndex();

};

void *createContext();

void destroyContext(void *context);

int getSocketType(SocketType type);

void *createSocket(void *context, SocketType type);

void closeSocket(void *socket);

string createAddress(AddressType type, pid_t id);

void bindSocket(void *socket, const string& address);

void unbindSocket(void *socket, const string& address);

void connectSocket(void *socket, const string& address);

void disconnectSocket(void *socket, const string& address);

void createMessage(zmq_msg_t *zmq_msg, Message &msg);

void sendMessage(void *socket, Message &msg);

Message getMessage(void *socket);

#endif
```

socket.h

```
#ifndef _SOCKET_H
#define _SOCKET_H

#include <string>
#include "message.h"
```

```
using namespace std;

class Socket {
public:
    Socket(void *context, SocketType socketType, const string& address) :
            socketType(socketType), address(address) {
        socket = createSocket(context, socketType);
        switch (socketType) {
            case SocketType::PUBLISHER:
                bindSocket(socket, address);
                break;
            case SocketType::SUBSCRIBER:
                connectSocket(socket, address);
                break;
            default:
                throw logic_error("undefined connection type");
        }
    }

    ~Socket() {
        try {
            switch (socketType) {
                case SocketType::PUBLISHER:
                    unbindSocket(socket, address);
                    break;
                case SocketType::SUBSCRIBER:
                    disconnectSocket(socket, address);
                    break;
            }
            closeSocket(socket);
        } catch (exception& ex){
            cout << "Socket wasn't closed: " << ex.what() << endl;
        }
    }

    void send(Message message) {
        if (socketType == SocketType::PUBLISHER){
            sendMessage(socket, message);
        } else {
            throw logic_error("SUBSCRIBER can't send messages");
        }
    }

    Message receive() {
        if (socketType == SocketType::SUBSCRIBER){
            return getMessage(socket);
        } else {
            throw logic_error("PUBLISHER can't receive messages");
        }
    }

    string getAddress() const {
```

```
        return address;
    }

    void *&getSocket() {
        return socket;
    }

private:
    void *socket;
    SocketType socketType;
    string address;
};



#endif
```

tree.h

```
#ifndef _TREE_H
#define _TREE_H

#include <vector>

using namespace std;

class treeNode {
private:
    treeNode *left;
    treeNode *right;
    int value;

public:
    explicit treeNode(int value) : left(nullptr), right(nullptr), value(value) {}

    int &getValue() {
        return value;
    }

    treeNode *&getLeft() {
        return left;
    }

    treeNode *&getRight() {
        return right;
    }

    bool operator<(treeNode &rhs) const {
        return this->value < rhs.getValue();
    }

    bool operator==(treeNode &rhs) const {
        return this->value == rhs.getValue();
    }
```

```
};

class Tree {
private:
    treeNode *root;

    void deleteTree(treeNode *&current) {
        if (!current) { return; }
        deleteTree(current->getLeft());
        deleteTree(current->getRight());
        delete current;
        current = nullptr;
    }

    static void print(treeNode *&current, int h = 0) {
        if (!current) { return; }
        print(current->getRight(), h + 2);
        for (int i = 0; i < h; i++) {
            cout << "-";
        }
        cout << current->getValue() << endl;
        print(current->getLeft(), h + 2);
    }

    bool find(treeNode *current, int value) {
        if (!current) { return false; }
        if (value < current->getValue()) {
            return find(current->getLeft(), value);
        } else if (value > current->getValue()) {
            return find(current->getRight(), value);
        } else {
            return true;
        }
    }

    void insert(treeNode *&current, int value) {
        if (!current) {
            current = new treeNode(value);
        } else if (value < current->getValue()) {
            insert(current->getLeft(), value);
        } else {
            insert(current->getRight(), value);
        }
    }

    void remove(treeNode *&current, int value) {
        if (!current) { return; }
        if (value < current->getValue()) {
            remove(current->getLeft(), value);
        } else if (value > current->getValue()) {
            remove(current->getRight(), value);
        } else {
            deleteTree(current);
```

```
        }
    }

    int getParent(treeNode *&current, int value) {
        if (value < current->getValue()) {
            if (!current->getLeft()) {
                return current->getValue();
            }
            return getParent(current->getLeft(), value);
        } else if (value > current->getValue()) {
            if (!current->getRight()) {
                return current->getValue();
            }
            return getParent(current->getRight(), value);
        }
        return -1;
    }

    void getAll(treeNode *current, vector<int> &tmp) {
        if (!current) { return; }
        getAll(current->getLeft(), tmp);
        tmp.push_back(current->getValue());
        getAll(current->getRight(), tmp);
    }

public:
    Tree() : root(nullptr) {};

    void insert(int value) {
        insert(root, value);
    }

    bool find(int value) {
        return find(root, value);
    }

    void print() {
        print(root);
    }

    void remove(int value) {
        remove(root, value);
    }

    int getPlace(int value) {
        return getParent(root, value);
    }

    vector<int> getElements(){
        vector<int> tmp;
        getAll(root, tmp);
        return tmp;
    }
```

```
    ~Tree() {
        deleteTree(root);
    }
};

#endif
```

client.cpp

```
#include <cstring>
#include <iostream>
#include <unistd.h>
#include <utility>
#include <vector>
#include <algorithm>
#include <csignal>
#include "headers/message.h"
#include "headers/socket.h"

using namespace std;

class Client {
private:
    int id;
    void *context;
    bool terminated;

public:
    Socket *childPublisherLeft;
    Socket *childPublisherRight;
    Socket *parentPublisher;
    Socket *parentSubscriber;
    Socket *leftSubscriber;
    Socket *rightSubscriber;

    Client(int id, const string& parentAddress) : id(id) {
        context = createContext();
        string address = createAddress(AddressType::CHILD_PUB_LEFT, getpid());
        childPublisherLeft = new Socket(context, SocketType::PUBLISHER, address);
        address = createAddress(AddressType::CHILD_PUB_RIGHT, getpid());
        childPublisherRight = new Socket(context, SocketType::PUBLISHER, address);
        address = createAddress(AddressType::PARENT_PUB, getpid());
        parentPublisher = new Socket(context, SocketType::PUBLISHER, address);
        parentSubscriber = new Socket(context, SocketType::SUBSCRIBER, parentAddress);
        leftSubscriber = nullptr;
        rightSubscriber = nullptr;
        terminated = false;
    }

    ~Client() {
        if (terminated) return;
```

```
    terminated = true;
    try {
        delete childPublisherLeft;
        delete childPublisherRight;
        delete parentPublisher;
        delete parentSubscriber;
        delete leftSubscriber;
        delete rightSubscriber;
        destroyContext(context);
    } catch (runtime_error &err) {
        cout << "Server wasn't stopped " << err.what() << endl;
    }
}

void messageProcessing(Message msg) {
    switch (msg.command) {
        case CommandType::ERROR:
            throw runtime_error("error message received");
        case CommandType::RETURN: {
            msg.getToIndex() = SERVER_ID;
            sendUp(msg);
            break;
        }
        case CommandType::CREATE_CHILD: {
            msg.getCreateIndex() = addChild(msg.getCreateIndex());
            msg.getToIndex() = SERVER_ID;
            sendUp(msg);
            break;
        }
        case CommandType::REMOVE_CHILD: {
            if (msg.withoutProcessing) {
                sendUp(msg);
                break;
            }
            if (msg.toIndex != getId() && msg.toIndex != UNIVERSAL_MESSAGE) {
                sendDown(msg);
                break;
            }
            msg.getToIndex() = UNIVERSAL_MESSAGE;
            sendDown(msg);
            this->~Client();
            throw invalid_argument("Exiting child...");
        }
        case CommandType::EXEC_CHILD: {
            double res = 0.0;
            for (int i = 0; i < msg.size; ++i) {
                res += msg.value[i];
            }
            msg.getToIndex() = SERVER_ID;
            msg.getCreateIndex() = getId();
            msg.value[0] = res;
            sendUp(msg);
            break;
```

```
            }
            default:
                throw runtime_error("undefined command");
        }
    }

    void sendUp(Message msg) const {
        msg.withoutProcessing = true;
        parentPublisher->send(msg);
    }

    void sendDown(Message msg) const {
        msg.withoutProcessing = false;
        childPublisherLeft->send(msg);
        childPublisherRight->send(msg);
    }

    int getId() const {
        return id;
    }

    int addChild(int childId) {
        pid_t pid = fork();
        if (pid == -1) throw runtime_error("fork error");
        if (!pid) {
            string address;
            if (childId < id) {
                address = childPublisherLeft->getAddress();
            } else {
                address = childPublisherRight->getAddress();
            }
            execl("client", "client", to_string(childId).data(), address.data(), nullptr);
            throw runtime_error("execl error");
        }
        string address = createAddress(AddressType::PARENT_PUB, pid);
        size_t timeout = 10000;
        if (id > childId) {
            leftSubscriber = new Socket(context, SocketType::SUBSCRIBER, address);
            //zmq_setsockopt(leftSubscriber->getSocket(), ZMQ_RCVTIMEO, &timeout, sizeof(timeout));
        } else {
            rightSubscriber = new Socket(context, SocketType::SUBSCRIBER, address);
            //zmq_setsockopt(rightSubscriber->getSocket(), ZMQ_RCVTIMEO, &timeout, sizeof(timeout));
        }
        return pid;
    }

};

Client *clientPointer = nullptr;

void terminate(int) {
    if (clientPointer) {
        clientPointer->~Client();
```

```
    }
    cout << to_string(getpid()) + " successfully terminated" << endl;
    exit(0);
}

int main(int argc, char const *argv[]) {
    if (argc != 3) {
        cout << "-1" << endl;
        return -1;
    }
    try {

        // Ctrl + C
        if (signal(SIGINT, terminate) == SIG_ERR) {
            throw runtime_error("Can not set SIGINT signal");
        }

        // Segmentation fault
        if (signal(SIGSEGV, terminate) == SIG_ERR) {
            throw runtime_error("Can not set SIGSEGV signal");
        }

        // kill
        if (signal(SIGTERM, terminate) == SIG_ERR) {
            throw runtime_error("Can not set SIGTERM signal");
        }

        Client client(stoi(argv[1]), string(argv[2]));
        clientPointer = &client;
        cout << getpid() << ": client " << client.getId() << " successfully started" << endl;
        while(true) {
            Message msg = client.parentSubscriber->receive();
            if (msg.toIndex != client.getId() && msg.toIndex != UNIVERSAL_MESSAGE) {
                if (msg.withoutProcessing) {
                    client.sendUp(msg);
                } else {
                    try {
                        if (client.getId() < msg.toIndex) {
                            msg.withoutProcessing = false;
                            client.childPublisherRight->send(msg);
                            msg = client.rightSubscriber->receive();
                        } else {
                            msg.withoutProcessing = false;
                            client.childPublisherLeft->send(msg);
                            msg = client.leftSubscriber->receive();
                        }
                        if (msg.command == CommandType::REMOVE_CHILD && msg.toIndex == PARENT_SIGNAL) {
                            msg.toIndex = SERVER_ID;
                            if (client.getId() < msg.getCreateIndex()) {
                                delete client.rightSubscriber;
                                client.rightSubscriber = nullptr;
                            } else {
                                delete client.leftSubscriber;
```

```
                        client.leftSubscriber = nullptr;
                    }
                }
                client.sendUp(msg);
            } catch (...) {
                client.sendUp(Message());
            }
        }
    } else {
        clientPointer->messageProcessing(msg);
    }
  }
  } catch (runtime_error &err) {
    cout << getpid() << ": " << err.what() << '\n';
  } catch (invalid_argument &inv) {
    cout << getpid() << ": " << inv.what() << '\n';
  }
  return 0;
}
```

message.cpp

```
#include <tuple>
#include <cstring>
#include "headers/message.h"
#include <unistd.h>
#include <iostream>
#include "/opt/homebrew/include/zmq.h"

using namespace std;

atomic<int> Message::counter;

Message::Message() {
  command = CommandType::ERROR;
  uniqueIndex = counter++;
  withoutProcessing = false;
}

Message::Message(CommandType command, int toIndex, int size, const double *value, int createIndex)
    : command(command), toIndex(toIndex), size(size), uniqueIndex(counter++), withoutProcessing(false),
      createIndex(createIndex) {
  for (int i = 0; i < size; ++i) {
    this->value[i] = value[i];
  }
}

Message::Message(CommandType command, int toIndex, int createIndex)
    : command(command), toIndex(toIndex), uniqueIndex(counter++), withoutProcessing(false),
      createIndex(createIndex) {}

bool operator==(const Message &lhs, const Message &rhs) {
  return tie(lhs.command, lhs.toIndex, lhs.createIndex, lhs.uniqueIndex) ==
```

```
        tie(rhs.command, rhs.toIndex, rhs.createIndex, rhs.uniqueIndex);
}

int &Message::getCreateIndex() {
    return createIndex;
}

int &Message::getToIndex() {
    return toIndex;
}

void *createContext() {
    void *context = zmq_ctx_new();
    if (!context) {
        throw runtime_error("unable to create new context");
    }
    return context;
}

void destroyContext(void *context) {
    sleep(1);
    if (zmq_ctx_destroy(context)) {
        throw runtime_error("unable to destroy context");
    }
}

int getSocketType(SocketType type) {
    switch (type) {
        case SocketType::PUBLISHER:
            return ZMQ_PUB;
        case SocketType::SUBSCRIBER:
            return ZMQ_SUB;
        default:
            throw runtime_error("undefined socket type");
    }
}

void *createSocket(void *context, SocketType type) {
    int zmq_type = getSocketType(type);
    void *socket = zmq_socket(context, zmq_type);
    if (!socket) {
        throw runtime_error("unable to create socket");
    }
    return socket;
}

void closeSocket(void *socket) {
    sleep(1);
    if (zmq_close(socket)) {
        throw runtime_error("unable to close socket");
    }
}
```

```
string createAddress(AddressType type, pid_t id) {
    switch (type) {
        case AddressType::PARENT_PUB:
            return "ipc://parent_publisher_" + to_string(id);
        case AddressType::CHILD_PUB_LEFT:
            return "ipc://child_publisher_left_" + to_string(id);
        case AddressType::CHILD_PUB_RIGHT:
            return "ipc://child_publisher_right" + to_string(id);
        default:
            throw runtime_error("wrong address type");
    }
}

void bindSocket(void *socket, const string& address) {
    if (zmq_bind(socket, address.data())) {
        throw runtime_error("unable to bind socket");
    }
}

void unbindSocket(void *socket, const string& address) {
    sleep(1);
    if (zmq_unbind(socket, address.data())) {
        throw runtime_error("unable to unbind socket");
    }
}

void connectSocket(void *socket, const string& address) {
    if (zmq_connect(socket, address.data())) {
        throw runtime_error("unable to connect socket");
    }
    zmq_setsockopt(socket, ZMQ_SUBSCRIBE, nullptr, 0);
}

void disconnectSocket(void *socket, const string& address) {
    if (zmq_disconnect(socket, address.data())) {
        throw runtime_error("unable to disconnect socket.");
    }
}

void createMessage(zmq_msg_t *zmq_msg, Message &msg) {
    zmq_msg_init_size(zmq_msg, sizeof(msg));
    memcpy(zmq_msg_data(zmq_msg), &msg, sizeof(msg));
}

void sendMessage(void *socket, Message &msg) {
    zmq_msg_t zmq_msg;
    createMessage(&zmq_msg, msg);
    if (!zmq_msg_send(&zmq_msg, socket, 0)) {
        throw runtime_error("unable to send message");
    }
    zmq_msg_close(&zmq_msg);
}
```

```
Message getMessage(void *socket) {
    zmq_msg_t zmq_msg;
    zmq_msg_init(&zmq_msg);
    if (zmq_msg_recv(&zmq_msg, socket, 0) == -1) {
        return {"error"};
    }
    Message msg;
    memcpy(&msg, zmq_msg_data(&zmq_msg), sizeof(msg));
    zmq_msg_close(&zmq_msg);
    return msg;
}
```

server.cpp

```
#include <iostream>
#include <vector>
#include <unistd.h>
#include <csignal>
#include "headers/message.h"
#include "headers/socket.h"
#include "headers/tree.h"
#include "/opt/homebrew/include/zmq.h"

#define SECOND 1'000'000

void *receiveFunction(void *server);
void *heartbeatFunction(void *server);

class Server {
public:
    void commandProcessing(const string &cmd) {
        if (cmd == "create") {
            int id;
            cin >> id;
            createChild(id);
        } else if (cmd == "exec") {
            int id, n;
            cin >> id >> n;
            vector<double> nums(n);
            for (int i = 0; i < n; ++i) {
                cin >> nums[i];
            }
            execChild(id, n, nums.data());
        } else if (cmd == "exit") {
            throw invalid_argument("Exiting...");
        } else if (cmd == "heartbit") {
            int time;
            cin >> time;
            heartbeat(time);
        } else if (cmd == "status") {
            int id;
            cin >> id;
            if (!getTree().find(id)) {
```

```
            throw runtime_error("Error: node " + to_string(id) + " doesn't exist");
        }
        if (check(id)) {
            cout << "OK" << endl;
        } else {
            cout << "Node " + to_string(id) + " is unavailable" << endl;
        }
    } else {
        cout << "invalid command\n";
    }
}

Server() {
    context = createContext();
    pid = getpid();
    string address = createAddress(AddressType::CHILD_PUB_LEFT, pid);
    publisher = new Socket(context, SocketType::PUBLISHER, address);
    if (pthread_create(&receiveMessage, nullptr, receiveFunction, this) != 0) {
        throw runtime_error("thread create error");
    }
    working = true;
}

~Server() {
    if (!working) return;
    working = false;
    send(Message(CommandType::REMOVE_CHILD, 0, 0));
    try {
        delete publisher;
        delete subscriber;
        publisher = nullptr;
        subscriber = nullptr;
        destroyContext(context);
        usleep(7.5 * SECOND);
    } catch (runtime_error &err) {
        cout << "Server wasn't stopped " << err.what() << endl;
    }
}

void send(Message msg) {
    msg.withoutProcessing = false;
    publisher->send(msg);
}

void createChild(int id) {
    if (t.find(id)) {
        throw runtime_error("Error: node " + to_string(id) + " already exists");
    }
    if (t.getPlace(id) && !check(t.getPlace(id))) {
        throw runtime_error("Error: parent node " + to_string(t.getPlace(id)) + " is unavailable");
    }
    send(Message(CommandType::CREATE_CHILD, t.getPlace(id), id));
    t.insert(id);
```

```
}

void execChild(int id, int n, const double *nums) {
    if (!t.find(id)) {
        throw runtime_error("Error: node " + to_string(id) + " doesn't exist");
    }
    if (!check(id)) {
        throw runtime_error("Error: node " + to_string(id) + " is unavailable");
    }
    send(Message(CommandType::EXEC_CHILD, id, n, nums, 0));
}

bool check(int id) {
    Message msg(CommandType::RETURN, id, 0);
    send(msg);
    usleep(SECOND);
    msg.getToIndex() = SERVER_ID;
    return lastMessage == msg;
}

bool check(int id, int time) {
    int timeout = 4 * time;
    zmq_setsockopt(getSubscriber()->getSocket(), ZMQ_RCVTIMEO, &timeout, sizeof(timeout));
    Message msg(CommandType::RETURN, id, 0);
    send(msg);
    usleep(timeout * 1000);
    return true;
}

Socket *&getPublisher() {
    return publisher;
}

Socket *&getSubscriber() {
    return subscriber;
}

void *getContext() {
    return context;
}

Tree &getTree() {
    return t;
}

Message lastMessage;

pthread_t heartbeatThread;
int heartbeatTime;
bool isHeartbeat;

void heartbeat(int time) {
    if (!isHeartbeat) {
```

```
            heartbeatTime = time;
            isHeartbeat = true;
            if (pthread_create(&heartbeatThread, nullptr, heartbeatFunction, this) != 0) {
                throw runtime_error("thread create error");
            }
        } else {
            isHeartbeat = false;
            if (pthread_join(heartbeatThread, nullptr) != 0) {
                throw runtime_error("thread join error");
            }
        }
    }

private:
    pid_t pid;
    Tree t;
    void *context;
    Socket *publisher;
    Socket *subscriber;
    bool working;
    pthread_t receiveMessage;
};

void *receiveFunction(void *server) {
    auto *serverPointer = (Server *) server;
    try {
        pid_t child_pid = fork();
        if (child_pid == -1) throw runtime_error("Can not fork.");
        if (child_pid == 0) {
            execl("client", "client", "0", serverPointer->getPublisher()->getAddress().data(), nullptr);
            throw runtime_error("Can not execl");
        }
        string address = createAddress(AddressType::PARENT_PUB, child_pid);
        serverPointer->getSubscriber() = new Socket(serverPointer->getContext(), SocketType::SUBSCRIBER,
address);
        serverPointer->getTree().insert(0);
        while (true) {
            Message msg = serverPointer->getSubscriber()->receive();
            if (msg.command == CommandType::ERROR) {
                continue;
            }
            serverPointer->lastMessage = msg;
            switch (msg.command) {
                case CommandType::CREATE_CHILD:
                    cout << "OK: " << msg.getCreateIndex() << endl;
                    break;
                case CommandType::RETURN:
                    break;
                case CommandType::EXEC_CHILD:
                    cout << "OK: response from node " << msg.getCreateIndex() << " is " << msg.value[0] << endl;
                    break;
                default:
                    break;
```

```
            }
        }
    } catch (runtime_error &err) {
        cout << "Server wasn't started " << err.what() << endl;
    }
    return nullptr;
}

void *heartbeatFunction(void *server) {
    auto *serverPointer = (Server *) server;
    while (serverPointer->isHeartbeat) {
        vector<int> tmp = serverPointer->getTree().getElements();
        bool answer = true;
        for (int &e: tmp) {
            if (!(serverPointer->check(e, serverPointer->heartbeatTime))) {
                answer = false;
                cout << "Heartbit: node " << e << " is unavailable now\n";
            }
        }
        if (answer) {
            cout << "OK\n";
        }
        usleep(serverPointer->heartbeatTime * 1000);  // Пауза на указанное время
    }
    return nullptr;
}

Server *serverPointer = nullptr;

void terminate(int) {
    if (serverPointer) {
        serverPointer->~Server();
    }
    cout << to_string(getpid()) + " successfully terminated" << endl;
    exit(0);
}

int main() {
    try {
        // Ctrl + C
        if (signal(SIGINT, terminate) == SIG_ERR) {
            throw runtime_error("Can not set SIGINT signal");
        }

        // Segmentation fault
        if (signal(SIGSEGV, terminate) == SIG_ERR) {
            throw runtime_error("Can not set SIGSEGV signal");
        }

        // kill
        if (signal(SIGTERM, terminate) == SIG_ERR) {
            throw runtime_error("Can not set SIGTERM signal");
        }
```

```
        Server server = Server();
        serverPointer = &server;
        cout << getpid() << " server started correctly!\n";
        while (true) {
            try {
                string cmd;
                while (cin >> cmd) {
                    server.commandProcessing(cmd);
                }
            } catch (const runtime_error &arg) {
                cout << arg.what() << endl;
            }
        }
    } catch (const runtime_error &arg) {
        cout << arg.what() << endl;
    } catch (...) {}
    return 0;
}
```

Приложение 2

execve("./server", ["./server"], 0x7fff0b371940 /* 74 vars */) = 0

brk(NULL) = 0x645aec98a000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75eef31dd000

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

openat(AT_FDCWD, "/opt/homebrew/lib/glibc-hwcaps/x86-64-v4/libzmq.so.5", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/opt/homebrew/lib/glibc-hwcaps/x86-64-v4/", 0x7fff187aece0, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/opt/homebrew/lib/glibc-hwcaps/x86-64-v3/libzmq.so.5", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/opt/homebrew/lib/glibc-hwcaps/x86-64-v3/", 0x7fff187aece0, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/opt/homebrew/lib/glibc-hwcaps/x86-64-v2/libzmq.so.5", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/opt/homebrew/lib/glibc-hwcaps/x86-64-v2/", 0x7fff187aece0, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/opt/homebrew/lib/libzmq.so.5", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/opt/homebrew/lib/", 0x7fff187aece0, 0) = -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=87915, ...}) = 0

mmap(NULL, 87915, PROT_READ, MAP_PRIVATE, 3, 0) = 0x75eef31c7000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/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\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=663584, ...}) = 0

mmap(NULL, 661336, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75eef3125000

mmap(0x75eef313e000, 425984, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19000) = 0x75eef313e000

mmap(0x75eef31a6000, 98304, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x81000) = 0x75eef31a6000

mmap(0x75eef31be000, 36864, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x99000) = 0x75eef31be000

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\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=2592144, ...}) = 0

mmap(NULL, 2605376, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75eef2e00000

mmap(0x75eef2e9d000, 1310720, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9d000) = 0x75eef2e9d000

mmap(0x75eef2fdd000, 581632, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1dd000) = 0x75eef2fdd000

mmap(0x75eef306b000, 57344, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x26b000) = 0x75eef306b000

mmap(0x75eef3079000, 12608, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x75eef3079000

close(3) = 0

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\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=182944, ...}) = 0

mmap(NULL, 181160, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75eef30f8000

mmap(0x75eef30fc000, 143360, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4000) = 0x75eef30fc000

mmap(0x75eef311f000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x27000)
= 0x75eef311f000

mmap(0x75eef3123000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2b000) = 0x75eef3123000

close(3) = 0

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\220\243\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

fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 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

mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75eef2a00000

mmap(0x75eef2a28000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x75eef2a28000

mmap(0x75eef2bb0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x75eef2bb0000

mmap(0x75eef2bff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x75eef2bff000

mmap(0x75eef2c05000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x75eef2c05000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libbsd.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\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=80888, ...}) = 0

mmap(NULL, 86208, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75eef30e2000

mmap(0x75eef30e6000, 49152, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4000) = 0x75eef30e6000

mmap(0x75eef30f2000, 12288, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x10000)
= 0x75eef30f2000

mmap(0x75eef30f5000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x12000) = 0x75eef30f5000

mmap(0x75eef30f7000, 192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x75eef30f7000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libsodium.so.23", 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\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=355040, ...}) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x75eef30e0000

mmap(NULL, 353336, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75eef3089000

mmap(0x75eef3095000, 233472, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xc000) = 0x75eef3095000

mmap(0x75eef30ce000, 65536, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x45000)
= 0x75eef30ce000

mmap(0x75eef30de000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x55000) = 0x75eef30de000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libpgm-5.3.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\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=285568, ...}) = 0

mmap(NULL, 301040, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75eef2db6000

mmap(0x75eef2dba000, 159744, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4000) = 0x75eef2dba000

mmap(0x75eef2de1000, 102400, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2b000)
= 0x75eef2de1000

mmap(0x75eef2dfa000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x44000) = 0x75eef2dfa000

mmap(0x75eef2dfc000, 14320, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x75eef2dfc000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libnorm.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\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=366760, ...}) = 0

mmap(NULL, 1092032, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75eef2cab000

mmap(0x75eef2cb4000, 274432, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9000) = 0x75eef2cb4000

mmap(0x75eef2cf7000, 45056, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4c000) =
0x75eef2cf7000

mmap(0x75eef2d02000, 16384, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x56000) = 0x75eef2d02000

mmap(0x75eef2d06000, 719296, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x75eef2d06000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgssapi_krb5.so.2", 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\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=338696, ...}) = 0

mmap(NULL, 341080, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75eef2c57000

mmap(0x75eef2c63000, 237568, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xc000) = 0x75eef2c63000

mmap(0x75eef2c9d000, 40960, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x46000)
= 0x75eef2c9d000

mmap(0x75eef2ca7000, 16384, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4f000) = 0x75eef2ca7000

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\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=952616, ...}) = 0

mmap(NULL, 950296, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75eef2917000

mmap(0x75eef2927000, 520192, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x10000) = 0x75eef2927000

mmap(0x75eef29a6000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8f000)
= 0x75eef29a6000

mmap(0x75eef29fe000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe7000) = 0x75eef29fe000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libmd.so.0", O_RDONLY|O_CLOEXEC) = 3

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

fstat(3, {st_mode=S_IFREG|0644, st_size=55536, ...}) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x75eef3087000

mmap(NULL, 57448, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75eef2c48000

mmap(0x75eef2c4a000, 36864, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x75eef2c4a000

mmap(0x75eef2c53000, 8192, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xb000) =
0x75eef2c53000

mmap(0x75eef2c55000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xc000) = 0x75eef2c55000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libkrb5.so.3", O_RDONLY|O_CLOEXEC) = 3

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

fstat(3, {st_mode=S_IFREG|0644, st_size=823488, ...}) = 0

mmap(NULL, 822032, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75eef284e000

mmap(0x75eef286e000, 397312, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x20000) = 0x75eef286e000

mmap(0x75eef28cf000, 233472, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x81000) = 0x75eef28cf000

mmap(0x75eef2908000, 61440, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xba000) = 0x75eef2908000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libk5crypto.so.3", 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\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=178648, ...}) = 0

mmap(NULL, 176392, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75eef2c1c000

mmap(0x75eef2c20000, 110592, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4000) = 0x75eef2c20000

mmap(0x75eef2c3b000, 45056, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1f000) = 0x75eef2c3b000

mmap(0x75eef2c46000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2a000) = 0x75eef2c46000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libcom_err.so.2", 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\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=18504, ...}) = 0

mmap(NULL, 20552, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75eef3081000

mmap(0x75eef3083000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x75eef3083000

mmap(0x75eef3084000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x75eef3084000

mmap(0x75eef3085000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x75eef3085000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libkrb5support.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\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=47904, ...}) = 0

mmap(NULL, 50128, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75eef2841000

mmap(0x75eef2844000, 24576, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x75eef2844000

mmap(0x75eef284a000, 8192, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9000) = 0x75eef284a000

mmap(0x75eef284c000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xa000) = 0x75eef284c000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libkeyutils.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\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=22600, ...}) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75eef307f000

mmap(NULL, 24592, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75eef2c15000

mmap(0x75eef2c17000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x75eef2c17000

mmap(0x75eef2c19000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4000) = 0x75eef2c19000

mmap(0x75eef2c1a000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4000) = 0x75eef2c1a000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libresolv.so.2", 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\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=68104, ...}) = 0

mmap(NULL, 75912, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75eef282e000

mmap(0x75eef2831000, 40960, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x75eef2831000

mmap(0x75eef283b000, 8192, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xd000) = 0x75eef283b000

mmap(0x75eef283d000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xf000) = 0x75eef283d000

mmap(0x75eef283f000, 6280, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x75eef283f000

close(3) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75eef307d000

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75eef2c12000

arch_prctl(ARCH_SET_FS, 0x75eef2c129c0) = 0

set_tid_address(0x75eef2c12c90) = 923974

set_robust_list(0x75eef2c12ca0, 24) = 0

rseq(0x75eef2c132e0, 0x20, 0, 0x53053053) = 0

mprotect(0x75eef2bff000, 16384, PROT_READ) = 0

mprotect(0x75eef283d000, 4096, PROT_READ) = 0

mprotect(0x75eef2c1a000, 4096, PROT_READ) = 0

mprotect(0x75eef284c000, 4096, PROT_READ) = 0

mprotect(0x75eef3085000, 4096, PROT_READ) = 0

mprotect(0x75eef2c46000, 4096, PROT_READ) = 0

mprotect(0x75eef2908000, 53248, PROT_READ) = 0

mprotect(0x75eef2c55000, 4096, PROT_READ) = 0

mprotect(0x75eef29fe000, 4096, PROT_READ) = 0

mprotect(0x75eef2ca7000, 8192, PROT_READ) = 0

mprotect(0x75eef3123000, 4096, PROT_READ) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75eef282c000

mprotect(0x75eef306b000, 45056, PROT_READ) = 0

mprotect(0x75eef2d02000, 12288, PROT_READ) = 0

mprotect(0x75eef2dfa000, 4096, PROT_READ) = 0

mprotect(0x75eef30de000, 4096, PROT_READ) = 0

mprotect(0x75eef30f5000, 4096, PROT_READ) = 0

mprotect(0x75eef31be000, 32768, PROT_READ) = 0

mprotect(0x645aebc0e000, 4096, PROT_READ) = 0

mprotect(0x75eef3215000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

munmap(0x75eef31c7000, 87915) = 0

futex(0x75eef30797bc, FUTEX_WAKE_PRIVATE, 2147483647) = 0

getrandom("\xa4\xa6\x12\x7a\x96\x8d\xee\xdb", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0x645aec98a000

brk(0x645aec9ab000) = 0x645aec9ab000

rt_sigaction(SIGINT, {sa_handler=0x645aebc050c7, sa_mask=[INT], sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x75eef2a45320}, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=0}, 8) = 0

rt_sigaction(SIGSEGV, {sa_handler=0x645aebc050c7, sa_mask=[SEGV], sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x75eef2a45320}, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=0}, 8) = 0

rt_sigaction(SIGTERM, {sa_handler=0x645aebc050c7, sa_mask=[TERM], sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x75eef2a45320}, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=0}, 8) = 0

openat(AT_FDCWD, "/sys/devices/system/cpu/online", O_RDONLY|O_CLOEXEC) = 3

read(3, "0-15\n", 1024) = 5

close(3) = 0

openat(AT_FDCWD, "/sys/devices/system/cpu/possible", O_RDONLY|O_CLOEXEC) = 3

read(3, "0-15\n", 1024) = 5

close(3) = 0

getpid() = 923974

sched_getaffinity(923974, 128, [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15]) = 8

newfstatat(AT_FDCWD, "/etc/nsswitch.conf", {st_mode=S_IFREG|0644, st_size=569, ...}, 0) = 0

newfstatat(AT_FDCWD, "/", {st_mode=S_IFDIR|0755, st_size=4096, ...}, 0) = 0

openat(AT_FDCWD, "/etc/nsswitch.conf", O_RDONLY|O_CLOEXEC) = 3

fstat(3, {st_mode=S_IFREG|0644, st_size=569, ...}) = 0

read(3, "# /etc/nsswitch.conf\n#\n# Example"..., 4096) = 569

read(3, "", 4096) = 0

fstat(3, {st_mode=S_IFREG|0644, st_size=569, ...}) = 0

close(3) = 0

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

fstat(3, {st_mode=S_IFREG|0644, st_size=87915, ...}) = 0

mmap(NULL, 87915, PROT_READ, MAP_PRIVATE, 3, 0) = 0x75eef31c7000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/glibc-hwcaps/x86-64-v4/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/lib/x86_64-linux-gnu/glibc-hwcaps/x86-64-v4/", 0x7fff187aa9e0, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/glibc-hwcaps/x86-64-v3/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/lib/x86_64-linux-gnu/glibc-hwcaps/x86-64-v3/", 0x7fff187aa9e0, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/glibc-hwcaps/x86-64-v2/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/lib/x86_64-linux-gnu/glibc-hwcaps/x86-64-v2/", 0x7fff187aa9e0, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/lib/x86_64-linux-gnu/", {st_mode=S_IFDIR|0755, st_size=126976, ...}, 0) = 0

openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/glibc-hwcaps/x86-64-v4/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/glibc-hwcaps/x86-64-v4/", 0x7fff187aa9e0, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/glibc-hwcaps/x86-64-v3/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/glibc-hwcaps/x86-64-v3/", 0x7fff187aa9e0, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/glibc-hwcaps/x86-64-v2/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/glibc-hwcaps/x86-64-v2/", 0x7fff187aa9e0, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/", {st_mode=S_IFDIR|0755, st_size=126976, ...}, 0) = 0

openat(AT_FDCWD, "/lib/glibc-hwcaps/x86-64-v4/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/lib/glibc-hwcaps/x86-64-v4/", 0x7fff187aa9e0, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/lib/glibc-hwcaps/x86-64-v3/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/lib/glibc-hwcaps/x86-64-v3/", 0x7fff187aa9e0, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/lib/glibc-hwcaps/x86-64-v2/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/lib/glibc-hwcaps/x86-64-v2/", 0x7fff187aa9e0, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/lib/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/lib/", {st_mode=S_IFDIR|0755, st_size=4096, ...}, 0) = 0

openat(AT_FDCWD, "/usr/lib/glibc-hwcaps/x86-64-v4/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/usr/lib/glibc-hwcaps/x86-64-v4/", 0x7fff187aa9e0, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/usr/lib/glibc-hwcaps/x86-64-v3/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/usr/lib/glibc-hwcaps/x86-64-v3/", 0x7fff187aa9e0, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/usr/lib/glibc-hwcaps/x86-64-v2/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/usr/lib/glibc-hwcaps/x86-64-v2/", 0x7fff187aa9e0, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/usr/lib/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/usr/lib/", {st_mode=S_IFDIR|0755, st_size=4096, ...}, 0) = 0

munmap(0x75eef31c7000, 87915)   = 0

```
openat(AT_FDCWD, "/etc/protocols", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=3144, ...}) = 0
lseek(3, 0, SEEK_SET)                   = 0
read(3, "# Internet (IP) protocols\n#\n# Up"..., 4096) = 3144
read(3, "", 4096)                       = 0
close(3)                                = 0
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()                                = 923974
getpid()                                = 923974
getrandom("\xcc\xee\xc5\x75\xdd\x4a\x80\x4c\xd6\x20\x7c\x90\x62\x52\x00\x2c", 16, 0) = 16
getrandom("\x04\x86\xa0\xfb\xdc\x66\x25\x23\x62\x76\x35\x7f\xb1\x0b\x71\xed", 16, 0) = 16
getpid()                                = 923974
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()                                = 923974
epoll_create1(EPOLL_CLOEXEC)            = 5
epoll_ctl(5, EPOLL_CTL_ADD, 4, {events=0, data={u32=3969500656, u64=110341679334896}}) = 0
epoll_ctl(5, EPOLL_CTL_MOD, 4, {events=EPOLLIN, data={u32=3969500656, u64=110341679334896}}) = 0
getpid()                                = 923974
rt_sigaction(SIGRT_1, {sa_handler=0x75eef2a99520, sa_mask=[],
sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x75eef2a45320}, NULL,
8) = 0
```

rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x75eef2000000

mprotect(0x75eef2001000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75eef2800990, parent_tid=0x75eef2800990, exit_signal=0, stack=0x75eef2000000, stack_size=0x7ffd00, tls=0x75eef28006c0} => {parent_tid=[923976]}, 88) = 923976

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

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() = 923974

epoll_create1(EPOLL_CLOEXEC) = 7

epoll_ctl(7, EPOLL_CTL_ADD, 6, {events=0, data={u32=3969501408, u64=110341679335648}}) = 0

epoll_ctl(7, EPOLL_CTL_MOD, 6, {events=EPOLLIN, data={u32=3969501408, u64=110341679335648}}) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x75eef1600000

mprotect(0x75eef1601000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75eef1e00990, parent_tid=0x75eef1e00990, exit_signal=0, stack=0x75eef1600000, stack_size=0x7ffd00, tls=0x75eef1e006c0} => {parent_tid=[923977]}, 88) = 923977

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

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()   = 923974

getpid()   = 923974

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

unlink("child_publisher_left_923974")   = -1 ENOENT (No such file or directory)

socket(AF_UNIX, SOCK_STREAM|SOCK_CLOEXEC, 0) = 9

bind(9, {sa_family=AF_UNIX, sun_path="child_publisher_left_923974"}, 29) = 0

listen(9, 100)   = 0

getsockname(9, {sa_family=AF_UNIX, sun_path="child_publisher_left_923974"}, [128 => 30]) = 0

getpid()   = 923974

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

getpid()   = 923974

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

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x75eef0c00000

mprotect(0x75eef0c01000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75eef1400990, parent_tid=0x75eef1400990, exit_signal=0, stack=0x75eef0c00000, stack_size=0x7ffd00, tls=0x75eef14006c0} => {parent_tid=[923978]}, 88) = 923978

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

getpid()   = 923974

fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0

write(1, "923974 server started correctly!"..., 33) = 33

fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0

read(0, "exec 10 3 1 2 3\n", 1024)   = 16

futex(0x75eef3124230, FUTEX_WAKE_PRIVATE, 2147483647) = 0

write(1, "Error: node 10 doesn't exist\n", 29) = 29

read(0, "heartbit 200\n", 1024)   = 13

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x75eef0200000

mprotect(0x75eef0201000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75eef0a00990, parent_tid=0x75eef0a00990, exit_signal=0, stack=0x75eef0200000, stack_size=0x7ffd00, tls=0x75eef0a006c0} => {parent_tid=[924075]}, 88) = 924075

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

read(0, "quit\n", 1024)   = 5

write(1, "invalid command\n", 16)   = 16

read(0, 0x645aec9a45b0, 1024)   = ? ERESTARTSYS (To be restarted if SA_RESTART is set)

--- SIGINT {si_signo=SIGINT, si_code=SI_KERNEL} ---

getpid()   = 923974

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

getpid()   = 923974

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

clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, 0x7fff187aacf0) = 0

getpid()   = 923974

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

getpid()   = 923974

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

clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, 0x7fff187aacf0) = 230

+++ killed by SIGHUP +++