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

Студент: Ларченко А.О.

Преподаватель: Миронов Е.С.

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

Дата: 08.01.24

Москва, 2024

***Цель работы:***

Целью является приобретение практических навыков в:

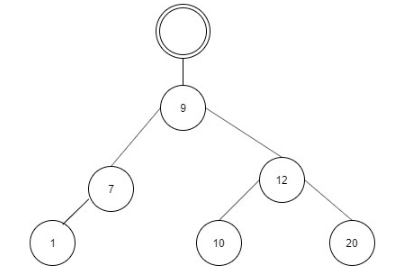
Управлении серверами сообщений (№5)

Применение отложенных вычислений (№6)

Интеграция программных систем друг с другом (№7)

***Задание***

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.

**Вариант 10.**

*Топология 4.* Все вычислительные узлы хранятся в идеально сбалансированном бинарном дереве. Каждый следующий узел должен добавляться в самое наименьшее поддерево.

*Тип 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

*Тип проверки доступности узлов 3.* 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.

**Общий метод и алгоритм решения**

Для реализации связи между управляющим узлом и исполняющими я буду использовать очередь сообщений из библиотеки *ZeroMQ.*

Очередь сообщений предоставляет гарантии, что сообщение будет доставлено независимо от того, что происходит. Очередь сообщений позволяет асинхронно взаимодействовать между слабо связанными компонентами, а также обеспечивает строгую последовательность очереди.

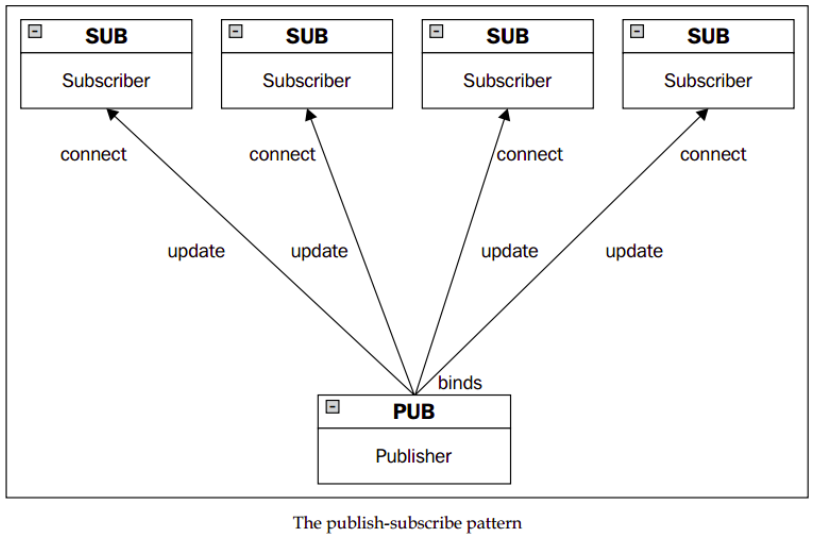
*ZeroMQ* - это не системой очередей сообщений типа WebSphereMQ, или RabbitMQ, это библиотека, которая дает нам инструменты для создания собственной системы очередей сообщений. Её еще называют сокетами на стероидах.

Для своей программы я выбрал паттерн PUB-SUB, но не обычный, управляющий узел у меня является публикатором для управляющих узлов, но при этом также является подписчиком на вычислительные узлы, чтобы получать от них обратную связь. А вычислительные узлы также являются и подписчиками и публикаторами.

PUB-сокет для управляющего узла привязан к порту, который является SUB- сокетом у вычислительных узлов.

А SUB-сокет управляющего узла привязан к другому порту, который является PUB-сокетом у вычислительных узлов.

К слову, сокет — это виртуальная конструкция из IP-адреса и номера порта, предназначенная для связи приложений или компьютеров между собой. Еще это именованный канал - FIFO, именованный pipe.



Использованные системные вызовы из библиотеки *ZeroMQ:*

socket.bind("tcp://\*IP\*:\*port\*") - установление связи socket по IP адресу и указанному порту port

* socket.set(zmq::sockopt::, "filter prefix") - настройка подписки, принимает только те сообщения, которые начинаются с префикса: "filter prefix"
* socket.connect("tcp://\*IP\*:\*port\*") - соединение с указанным портом
* socket.disconnect("tcp://\*IP\*:\*port\*") - отсоединение от порта
* socket.recv(reply, zmq::recv\_flags) - принятие сообщения
* socket.send(request1, zmq::send\_flags) - отправка сообщения

Использование адреса 127.0.0.1 позволяет устанавливать соединение и передавать информацию для программ-серверов, работающих на том же компьютере, что и программа-клиент, независимо от конфигурации аппаратных сетевых средств компьютера

**Код программы**

***msg.h:***

#pragma once

#include <iostream>

#include <string>

#include <vector>

using namespace std;

enum Subcommands{

timer,

start,

stop

};

enum Message\_type{

create,

create\_asw,

exec,

exec\_asw,

heartbit,

heartbit\_answ,

error,

die

};

struct Message{

Message\_type type;

vector<int> data;

};

***manage\_node.h:***

#pragma once

#include <iostream>

#include <unistd.h>

#include <signal.h>

#include <string>

#include <sys/wait.h>

#include <zmq.hpp>

#include "msg.h"

#include "awlTree.h"

using namespace std;

class Manage\_node{

public:

zmq::context\_t context;

zmq::socket\_t publisher;

zmq::socket\_t sub;

bool send\_msg(Message msg);

void receive\_msg(Message\_type msg\_type, AWL\_tree &tree);

Manage\_node();

~Manage\_node();

};

***manage\_node.cpp:***

#include "manage\_node.h"

Manage\_node::Manage\_node(): publisher(context, zmq::socket\_type::pub), sub(context,

zmq::socket\_type::sub){

publisher.bind("tcp://127.0.0.1:5555");

sub.bind("tcp://127.0.0.1:5556");

//фильтр для сообщений, подписываемся на все сообщения

sub.set(zmq::sockopt::subscribe, "");

}

Manage\_node::~Manage\_node(){

sub.disconnect("tcp://127.0.0.1:5556");

publisher.disconnect("tcp://127.0.0.1:5555");

}

void Manage\_node::receive\_msg(Message\_type msg\_type, AWL\_tree &tree){

cout<<"Receiving...\n";

switch(msg\_type){

case Message\_type::heartbit\_answ :{

zmq::message\_t reply;

zmq::recv\_result\_t res = sub.recv(reply, zmq::recv\_flags::none);

string id\_str=reply.to\_string();

int id=stoi(id\_str);

tree.change\_availability(id, true);

break;

}

}

}

bool Manage\_node::send\_msg(Message msg){

cout<<"Sending...\n";

string type\_str=to\_string(msg.type);

switch(msg.type){

case Message\_type::create :{

string parent\_id\_str=to\_string(msg.data[0]);

string new\_id\_str=to\_string(msg.data[1]);

zmq::message\_t request1(parent\_id\_str);

publisher.send(request1, zmq::send\_flags::sndmore);

zmq::message\_t request2(type\_str);

publisher.send(request2, zmq::send\_flags::sndmore);

zmq::message\_t request3(new\_id\_str);

publisher.send(request3, zmq::send\_flags::none);

return true;

}

case Message\_type::exec :{

string id\_str=to\_string(msg.data[0]);

string subcmd\_str=to\_string(msg.data[1]);

zmq::message\_t request1(id\_str);

publisher.send(request1, zmq::send\_flags::sndmore);

zmq::message\_t request2(type\_str);

publisher.send(request2, zmq::send\_flags::sndmore);

zmq::message\_t request3(subcmd\_str);

publisher.send(request3, zmq::send\_flags::none);

return true;

}

case Message\_type::heartbit :{

string id\_str=to\_string(msg.data[0]);

string period\_str=to\_string(msg.data[1]);

zmq::message\_t request1(id\_str);

publisher.send(request1, zmq::send\_flags::sndmore);

zmq::message\_t request2(type\_str);

publisher.send(request2, zmq::send\_flags::sndmore);

zmq::message\_t request3(period\_str);

publisher.send(request3, zmq::send\_flags::none);

return true;

}

default:

return false;

}

return false;

}

***manage\_main.cpp:***

#include <iostream>

#include <string>

#include <zmq.hpp>

#include <unistd.h>

#include <vector>

// #include "awlTree.h"

#include "manage\_node.h"

#include "timer.h"

using namespace std;

int process\_creation(){

int pid =fork();

if(pid==-1){

perror("Call fork was ended with erorr: ");

exit(-1);

}

return pid;

}

int what\_subcmd(string cmd){

if (cmd=="time"){

return Subcommands::timer;

} else if(cmd=="start"){

return Subcommands::start;

} else if(cmd=="stop"){

return Subcommands::stop;

}

return -1;

}

int main(){

AWL\_tree tree;

Manage\_node node;

cout<<"Welcome in our programm! This is what command i can do:\n";

cout<<" - create 'id' \n"<<" - exec 'id' 'command(start/stop/time)' \n";

cout<<" - heartbit 'time' (in ms) \n"<<" - draw\n";

cout<<"Or enter q or ^D to exit\n"<<"Enter you command: \n";

while (true){

string cmd;

cout<<" ->";

cin>>cmd;

if(cmd=="create"){

int id;

cin>>id;

if(id<0){

cout<<"Error: You can only use positive number\n";

} else if(tree.is\_in\_tree(id)){

cout<<"Error: Alredy exists\n";

} else{

tree.insert(id);

// tree.change\_availability(id, true);

int parent\_id=tree.parent\_id(id);

if(parent\_id==-1){

int pid=process\_creation();

if(pid==0){

string id\_str=to\_string(id);

execl("./node", "./node", id\_str.c\_str(), NULL);

} else{

cout<<"Ok: "<<pid<<"\n";

}

} else{

Message msg;

msg.type=Message\_type::create;

msg.data.push\_back(parent\_id);

msg.data.push\_back(id);

if(!node.send\_msg(msg)){

cout<<"Error...\n";

}

sleep(1);

}

}

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

int id;

string subcmd;

cin>>id;

cin>>subcmd;

if(!tree.is\_in\_tree(id)){

cout<<"Error: uncorrect id\n";

} else{

int subcmd\_int=what\_subcmd(subcmd);

if(subcmd\_int!=-1){

Message msg;

msg.type=Message\_type::exec;

msg.data.push\_back(id);

msg.data.push\_back(subcmd\_int);

if(!node.send\_msg(msg)){

cout<<"Error...\n";

}

sleep(1);

} else{

cout<<"Error: uncorrect subcommands\n";

}

}

} else if(cmd=="heartbit"){

int period;

cin>>period;

if(period<=0){

cout<<"Error: uncorrect input\n";

} else{

Message msg;

msg.type=Message\_type::heartbit;

msg.data.push\_back(-100); // for all users

msg.data.push\_back(period);

if(!node.send\_msg(msg)){

cout<<"Error...\n";

}

Timer tm;

tm.start();

sleep(0.1);

tree.bypass\_reset(tree.get\_root());

while(true){

if(tm.times()>=4\*period-150){

vector<int> unavailable;

tree.bypass(tree.get\_root(), unavailable);

for(int i=0; i< unavailable.size(); ++i){

cout << "\033[1;31m";

cout<<"Heartbit: node "<<unavailable[i]<<" is unavailable now"<<"\033[0m\n";

}

if (unavailable.size()==0){

cout << "\033[1;32m";

cout<<"Heartbit: all nodes are available"<<"\033[0m\n";

}

// tm.start();

tm.stop();

break;

}

node.receive\_msg(Message\_type::heartbit\_answ, tree);

}

}

} else if(cmd=="draw"){

tree.draw\_tree();

cout<<'\n';

} else if(cmd=="q" or cin.eof()){

cout<<"Break;\n";

break;

} else{

cout<<"Uncorrect input. Try again:\n";

}

}

}

***node.cpp:***

#include <iostream>

#include <string>

#include <zmq.hpp>

#include <unistd.h>

#include <signal.h>

#include <sys/prctl.h>

#include "msg.h"

#include "timer.h"

using namespace std;

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

string my\_id\_str=argv[1];

zmq::context\_t context;

zmq::socket\_t publisher(context, zmq::socket\_type::pub);

publisher.connect("tcp://127.0.0.1:5556");

zmq::socket\_t sub(context, zmq::socket\_type::sub);

sub.connect("tcp://127.0.0.1:5555");

sub.set(zmq::sockopt::subscribe, my\_id\_str);

sub.set(zmq::sockopt::subscribe, "-100");

sleep(1);

int exit\_status=prctl(PR\_SET\_PDEATHSIG, SIGKILL);

//Timer settings

Timer tm;

while(true){

// cout<<"Child: Receiving...\n";

zmq::message\_t reply;

zmq::recv\_result\_t res = sub.recv(reply, zmq::recv\_flags::none);

// if(res==-1){

// cout<<"FAIL\n";

// } else{

string req\_id=reply.to\_string();

zmq::message\_t cmd;

res = sub.recv(cmd, zmq::recv\_flags::none);

string cmd\_str=cmd.to\_string();

if(cmd\_str==to\_string(Message\_type::create)){

zmq::message\_t new\_id;

res = sub.recv(new\_id, zmq::recv\_flags::none);

string new\_id\_str=new\_id.to\_string();

int pid=fork();

if(pid==0){

execl("./node", "./node", new\_id\_str.c\_str(), NULL);

} else{

// cout<<"Child: Sending...\n";

string pid\_str=to\_string(pid);

// zmq::message\_t request(pid\_str);

// publisher.send(request, zmq::send\_flags::none);

cout<<"Ok:"<<my\_id\_str<<": "<<pid\_str<<'\n';

}

} else if(cmd\_str==to\_string(Message\_type::exec)){

zmq::message\_t subcmd;

res = sub.recv(subcmd, zmq::recv\_flags::none);

string subcmd\_str=subcmd.to\_string();

cout<<"Ok:"<<my\_id\_str;

if(subcmd\_str==to\_string(Subcommands::timer)){

cout<<": ";

cout<<tm.times();

} else if(subcmd\_str==to\_string(Subcommands::start)){

tm.start();

} else{

tm.stop();

}

cout<<'\n';

} else if(cmd\_str==to\_string(Message\_type::heartbit)){

zmq::message\_t period\_m;

res = sub.recv(period\_m, zmq::recv\_flags::none);

string period\_str=period\_m.to\_string();

// cout<<"Ok:"<<my\_id\_str<<"Period: "<<period\_str<<'\n';

int period=stoi(period\_str);

Timer hb\_timer;

hb\_timer.start();

Timer delay;

delay.start();

while(true){

if(delay.times()>=period\*4+200){

hb\_timer.stop();

delay.stop();

break;

}

sleep(0.1);

if(hb\_timer.times()>=period){

zmq::message\_t request1(my\_id\_str);

publisher.send(request1, zmq::send\_flags::none);

hb\_timer.start();

}

}

}

}

}

***awlTree.h:***

#pragma once

#include <iostream>

#include <vector>

using namespace std;

struct node{

int ID;

node\* left\_son;

node\* right\_son;

int hight;

bool available;

};

class AWL\_tree{

public:

AWL\_tree();

bool is\_in\_tree(int ID);

bool is\_available(int ID);

void change\_availability(int ID, bool status);

void bypass(node\* cur\_node, vector<int> &unavailable); //Bypassing tree and add to vector unavailable nodes

void bypass\_reset(node\* cur\_node);//make all nodes unavailable

node\* get(int ID);

node\* get\_root();

int parent\_id(int child\_id);

int cnt();

void draw\_tree();

node\* balancing(node\* cur\_node);

bool remove(int ID);

bool insert(int ID);

int check\_depth();

~AWL\_tree();

private:

node\* add(node\* cur\_node, node\* new\_node, int step);

void draw\_node(node\* cur\_node, int level);

node\* ll\_rot(node\* cur\_node);

node\* lr\_rot(node\* cur\_node);

node\* rr\_rot(node\* cur\_node);

node\* rl\_rot(node\* cur\_node);

int bf(node\* cur\_node);

int calHight(node\* cur\_node);

int find\_parent(node\* cur\_node, int ID);

node\* find(node\* current\_node ,int ID);

int depth;

node\* root;

int node\_cnt;

};

***awlTree.cpp:***

#include "awlTree.h"

AWL\_tree::AWL\_tree(){

// node\* new\_node= new node;

// new\_node->ID=0;

node\_cnt=0;

depth=0;

root=NULL;

}

int AWL\_tree::check\_depth(){

return depth;

}

int AWL\_tree::cnt(){

return node\_cnt;

}

int AWL\_tree::calHight(node\* cur\_node){

if(cur\_node->left\_son && cur\_node->right\_son){

if(cur\_node->left\_son->hight<cur\_node->right\_son->hight){

return cur\_node->right\_son->hight +1;

} else{

return cur\_node->left\_son->hight+1;

}

} else if(cur\_node->left\_son && cur\_node->right\_son==NULL){

return cur\_node->left\_son->hight +1;

} else if(cur\_node->right\_son && cur\_node->left\_son==NULL){

return cur\_node->right\_son->hight+1;

} else{

return 1;

}

}

int AWL\_tree::bf(node\* cur\_node){

if(cur\_node->left\_son && cur\_node->right\_son){

return cur\_node->left\_son->hight-cur\_node->right\_son->hight;

} else if(cur\_node->left\_son && cur\_node->right\_son==NULL){

return cur\_node->left\_son->hight;

} else if(cur\_node->right\_son && cur\_node->left\_son==NULL){

return -cur\_node->right\_son->hight;

}

return 0;

}

node\* AWL\_tree::ll\_rot(node\* cur\_node){

node\* tmp;

node\* answ;

tmp=cur\_node->left\_son->right\_son;

answ=cur\_node->left\_son;

cur\_node->left\_son=tmp;

answ->right\_son=cur\_node;

// cur\_node->left\_son=tmp;

return answ;

}

node\* AWL\_tree::rr\_rot(node\* cur\_node){

node\* tmp;

node\* answ;

tmp=cur\_node->right\_son->left\_son;

answ=cur\_node->right\_son;

cur\_node->right\_son=tmp;

answ->left\_son=cur\_node;

return answ;

}

node\* AWL\_tree::rl\_rot(node\* cur\_node){

cur\_node->right\_son=ll\_rot(cur\_node->right\_son);

return rr\_rot(cur\_node);

}

node\* AWL\_tree::lr\_rot(node\* cur\_node){

cur\_node->left\_son=rr\_rot(cur\_node->left\_son);

return ll\_rot(cur\_node);

}

node\* AWL\_tree::balancing(node\* cur\_node){

if(bf(cur\_node)==2 && bf(cur\_node->left\_son)==1){

return ll\_rot(cur\_node);

} else if(bf(cur\_node)==-2 && bf(cur\_node->right\_son)==-1){

return rr\_rot(cur\_node);

} else if(bf(cur\_node)==-2 && bf(cur\_node->right\_son)==1){

return rl\_rot(cur\_node);

} else if(bf(cur\_node)==2 && bf(cur\_node->left\_son)==-1){

return lr\_rot(cur\_node);

}

return cur\_node;

}

node\* AWL\_tree::find(node\* current\_node,int ID){

if(current\_node==NULL){

return NULL;

}

if(current\_node->ID==ID){

return current\_node;

} else if(ID>current\_node->ID){

return find(current\_node->right\_son, ID);

}

return find(current\_node->left\_son, ID);

}

bool AWL\_tree::is\_in\_tree(int ID){

node\* cur\_node=find(root, ID);

if(cur\_node==NULL){

return false;

}

return true;

}

node\* AWL\_tree::add(node\* cur\_node, node\* new\_node, int step){

node\* answ;

if(new\_node->ID>cur\_node->ID){

if(cur\_node->right\_son==NULL){

cur\_node->right\_son=new\_node;

// new\_node->level=step+1;

// return;

} else{

add(cur\_node->right\_son, new\_node, step+1);

}

} else {

if(cur\_node->left\_son==NULL){

cur\_node->left\_son=new\_node;

// new\_node->level=step+1;

// return;

} else{

add(cur\_node->left\_son, new\_node, step+1);

}

}

new\_node->hight=calHight(new\_node);

cur\_node->hight=calHight(cur\_node);

answ= balancing(cur\_node);

cur\_node->hight=calHight(cur\_node);

return answ;

}

bool AWL\_tree::insert(int ID){

if(is\_in\_tree(ID)){

return false;

}

node\* new\_node= new node;

new\_node->left\_son=NULL;

new\_node->right\_son=NULL;

new\_node->ID=ID;

if (node\_cnt==0){

root=new\_node;

root->hight=1;

depth=1;

} else{

new\_node->hight=1;

root=add(root, new\_node, 1);

// if(depth < new\_node->level){

// depth=new\_node->level;

// }

}

node\_cnt++;

depth=root->hight;

// balancing();

return true;

}

node\* AWL\_tree::get(int ID){

node\* find\_node=find(root, ID);

return find\_node;

}

node\* AWL\_tree::get\_root(){

return root;

}

void AWL\_tree::draw\_node(node\* cur\_node, int level){

if (cur\_node==NULL){

return;

}

draw\_node(cur\_node->right\_son, level+1);

for(int i=0; i<level; ++i){

cout<<"|===";

}

cout<<cur\_node->ID<<","<<cur\_node->hight<<'\n';

draw\_node(cur\_node->left\_son, level+1);

}

void AWL\_tree::draw\_tree(){

draw\_node(root, 0);

}

int AWL\_tree::find\_parent(node\* cur\_node, int child\_id){

if(cur\_node->ID<child\_id){

if(cur\_node->right\_son!=NULL){

if(cur\_node->right\_son->ID==child\_id){

return cur\_node->ID;

} else{

return find\_parent(cur\_node->right\_son, child\_id);

}

}

} else if(cur\_node->ID>child\_id){

if(cur\_node->left\_son!=NULL){

if(cur\_node->left\_son->ID==child\_id){

return cur\_node->ID;

} else{

return find\_parent(cur\_node->left\_son, child\_id);

}

}

}

return -1;

}

int AWL\_tree::parent\_id(int child\_id){

if(!is\_in\_tree(child\_id)){

return -1;

}

return find\_parent(root, child\_id);

}

bool AWL\_tree::is\_available(int ID){

node\* cur\_node=find(root, ID);

if(cur\_node==NULL){

return false;

}

return cur\_node->available;

}

void AWL\_tree::change\_availability(int ID, bool status){

node\* cur\_node=find(root, ID);

if(cur\_node!=NULL){

cur\_node->available=status;

}

}

void AWL\_tree::bypass(node\* cur\_node, vector<int> &unavailable){

if(!cur\_node->available){

unavailable.push\_back(cur\_node->ID);

}

if(cur\_node->left\_son!=NULL){

bypass(cur\_node->left\_son, unavailable);

}

if(cur\_node->right\_son!=NULL){

bypass(cur\_node->right\_son, unavailable);

}

}

void AWL\_tree::bypass\_reset(node\* cur\_node){

cur\_node->available=false;

if(cur\_node->left\_son!=NULL){

bypass\_reset(cur\_node->left\_son);

}

if(cur\_node->right\_son!=NULL){

bypass\_reset(cur\_node->right\_son);

}

}

AWL\_tree::~AWL\_tree(){

}

int main(){

AWL\_tree tree;

vector<int> data;

tree.insert(2);

tree.insert(7);

tree.insert(1);

tree.insert(10);

tree.insert(4);

tree.insert(11);

// tree.change\_availability(2, true);

// tree.bypass(tree.get\_root(), data);

// node\* n=tree.get(4);

// cout<<"hui\n";

tree.draw\_tree();

// cout<<'\n';

// cout<<tree.parent\_id(2);

// cout<<"\n"<<tree.check\_depth();

// cout<<"hui hui\n";

// cout<<"Size = "<<data.size()<<" Tree cnt = "<<tree.cnt();

}

***timer.h:***

#pragma once

#include <iostream>

#include <chrono>

using namespace std;

using t\_t = chrono::time\_point<std::chrono::system\_clock>;

using Clock = chrono::high\_resolution\_clock;

using ms= chrono::milliseconds; // can use int64\_t

class Timer{

public:

Timer();

int64\_t times();

void start();

void stop();

~Timer();

private:

bool t\_work;

t\_t begin, end;

ms oper\_time{0};

};

***timer.cpp:***

#include "timer.h"

// #include<unistd.h>

Timer::Timer() {

t\_work=false;

}

Timer::~Timer(){

}

int64\_t Timer::times(){

if(t\_work){

return chrono::duration\_cast<chrono::milliseconds>(Clock::now()-begin).count();

} else{

return oper\_time.count();

}

}

void Timer::start(){

t\_work=true;

begin=Clock::now();

}

void Timer::stop(){

if(t\_work){

t\_work=false;

oper\_time=chrono::duration\_cast<chrono::milliseconds>(Clock::now()-begin);

}

}

// int main(){

// Timer t;

// cout<<"Time: "<<t.times()<<'\n';

// t.start();

// sleep(2);

// cout<<"Time: "<<t.times()<<'\n';

// t.stop();

// sleep(2);

// cout<<"Time: "<<t.times()<<'\n';

// }

**Протокол работы программы**

**Тестирование:**

arsenii@PC-Larcha14:~/Documents/VS\_code\_prog/OSI/laba\_5\_7\_clear$ strace -f

-owrite-simple.log ./manage\_main

Welcome in our programm! This is what command i can do:

- create 'id'

- exec 'id' 'command(start/stop/time)'

- heartbit 'time' (in ms)

- draw

Or enter q or ^D to exit

Enter you command:

->create 8

Ok: 17873

->create 9

Sending...

Ok:8: 17889

->create 2

Sending...

Ok:8: 17893

->exec 2

time

Sending...

Ok:2: 0

->exec 2 start

Sending...

Ok:2

->exec 2 time

Sending...

Ok:2: 5656

->exec 2 stop

Sending...

Ok:2

->exec 2 time

Sending...

Ok:2: 11183

->exec 2 time

Sending...

Ok:2: 11183

->draw

|===9,1

8,2

|===2,1

->heartbit 3000

Sending...

Receiving...

Receiving...

Receiving...

Receiving...

Receiving...

Receiving...

Receiving...

Receiving...

Receiving...

Receiving...

Heartbit: all nodes are available

->heartbit 3000

Sending...

Receiving...

Receiving...

Receiving...

Receiving...

Receiving...

Receiving...

Receiving...

Receiving...

Receiving...

Heartbit: all nodes are available

->heartbit 3000

Sending...

Receiving...

Receiving...

Receiving...

Receiving...

Receiving...

Receiving...

Receiving...

Receiving...

Heartbit: node 2 is unavailable now

->q

Break;

arsenii@PC-Larcha14:~/Documents/VS\_code\_prog/OSI/laba\_5\_7\_clear$

==========================================================================================

**Strace:**

arsenii@PC-Larcha14:~/Documents/VS\_code\_prog/OSI/laba\_5\_7\_clear$ strace -f -e trace=\!brk,clock\_nanosleep,mmap,mprotect,munmap -owrite-simple3.log ./manage\_main

Welcome in our programm! This is what command i can do:

- create 'id'

- exec 'id' 'command(start/stop/time)'

- heartbit 'time' (in ms)

- draw

Or enter q or ^D to exit

Enter you command:

->create 2

Ok: 20397

->create 3

Sending...

Ok:2: 20402

->exec 3 time

Sending...

Ok:3: 0

->exec 3 start

Sending...

Ok:3

ex ->exec 3 stop

Uncorrect input. Try again:

->Uncorrect input. Try again:

->Uncorrect input. Try again:

->exec 3 stop

Sending...

Ok:3

->exec 3 time

Sending...

Ok:3: 13275

->q

Break;

***write-simple3.log:***

*(отключил вывод системного вызова clock\_nanosleep, т.к. занимает уж ооочень много*

*места, прошлый лог с ним был на 200 мб и 2,5 млн строчек…)*

***Пояснения:***

sendto - отправление сообщения на сокет

recvmsg - получение сообщения с сокета

socket - создать конечную точку для связи

setsockopt() - set the socket options

bind() - bind a name to a socket

listen() - network listener daemon

getsockname() - get socket name

epoll\_ctl - интерфейс управления описателями epoll (очень полезная штука, которая позволяет отложить реакцию на событие и продолжить ждать остальные события)

poll - input/output multiplexing (мультиплекси́рование — уплотнение канала, то есть передача нескольких потоков данных с меньшей скоростью по одному каналу)

20387 execve("./manage\_main", ["./manage\_main"], 0x7ffc80cae7e0 /\* 56 vars \*/) = 0

20387 arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffe4ad486d0) = -1 EINVAL (Invalid argument)

20387 access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

20387 openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=81715, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libzmq.so.5", O\_RDONLY|O\_CLOEXEC) = 3

20387 read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\240\233\1\0\0\0\0\0"..., 832) = 832

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=634936, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libstdc++.so.6", O\_RDONLY|O\_CLOEXEC) = 3

20387 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\0"..., 832) = 832

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=2260296, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = 3

20387 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\0"..., 832) = 832

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=125488, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

20387 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) = 832

20387 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

20387 pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

20387 pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0

=\340\2563\265?\356\25x\261\27\313A#\350"..., 68, 896) = 68

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2216304, ...}, AT\_EMPTY\_PATH) = 0

20387 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

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libbsd.so.0", O\_RDONLY|O\_CLOEXEC) = 3

20387 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\0"..., 832) = 832

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=89096, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libsodium.so.23",

O\_RDONLY|O\_CLOEXEC) = 3

20387 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\0"..., 832) = 832

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=355040, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libpgm-5.3.so.0",

O\_RDONLY|O\_CLOEXEC) = 3

20387 read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\340L\0\0\0\0\0\0"..., 832) = 832

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=310264, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libnorm.so.1", O\_RDONLY|O\_CLOEXEC) = 3

20387 read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0 \255\0\0\0\0\0\0"..., 832) = 832

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=497824, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libgssapi\_krb5.so.2",

O\_RDONLY|O\_CLOEXEC) = 3

20387 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\0"..., 832) = 832

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=338648, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libm.so.6", O\_RDONLY|O\_CLOEXEC) = 3

20387 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\0"..., 832) = 832

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=940560, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libmd.so.0", O\_RDONLY|O\_CLOEXEC) = 3

20387 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\0"..., 832) = 832

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=47472, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libpthread.so.0", O\_RDONLY|O\_CLOEXEC) = 3

20387 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\0"..., 832) = 832

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=21448, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libkrb5.so.3", O\_RDONLY|O\_CLOEXEC) = 3

20387 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\0"..., 832) = 832

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=827936, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libk5crypto.so.3",

O\_RDONLY|O\_CLOEXEC) = 3

20387 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\0"..., 832) = 832

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=182864, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libcom\_err.so.2",

O\_RDONLY|O\_CLOEXEC) = 3

20387 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\0"..., 832) = 832

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=18504, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libkrb5support.so.0",

O\_RDONLY|O\_CLOEXEC) = 3

20387 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\0"..., 832) = 832

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=52016, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libkeyutils.so.1", O\_RDONLY|O\_CLOEXEC) = 3

20387 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\0"..., 832) = 832

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=22600, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libresolv.so.2", O\_RDONLY|O\_CLOEXEC) = 3

20387 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\0"..., 832) = 832

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=68552, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 arch\_prctl(ARCH\_SET\_FS, 0x7f57c38339c0) = 0

20387 set\_tid\_address(0x7f57c3833c90) = 20387

20387 set\_robust\_list(0x7f57c3833ca0, 24) = 0

20387 rseq(0x7f57c3834360, 0x20, 0, 0x53053053) = 0

20387 prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024,

rlim\_max=RLIM64\_INFINITY}) = 0

20387 getrandom("\x2c\x01\xb6\x7f\xd4\xc3\xed\xa3", 8, GRND\_NONBLOCK) = 8

20387 futex(0x7f57c382977c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

20387 openat(AT\_FDCWD, "/sys/devices/system/cpu/online", O\_RDONLY|O\_CLOEXEC) = 3

20387 read(3, "0-11\n", 1024) = 5

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/sys/devices/system/cpu",

O\_RDONLY|O\_NONBLOCK|O\_CLOEXEC|O\_DIRECTORY) = 3

20387 newfstatat(3, "", {st\_mode=S\_IFDIR|0755, st\_size=0, ...}, AT\_EMPTY\_PATH) = 0

20387 getdents64(3, 0x55f459f1bee0 /\* 30 entries \*/, 32768) = 864

20387 getdents64(3, 0x55f459f1bee0 /\* 0 entries \*/, 32768) = 0

20387 close(3) = 0

20387 getpid() = 20387

20387 sched\_getaffinity(20387, 128, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]) = 8

20387 newfstatat(AT\_FDCWD, "/etc/nsswitch.conf", {st\_mode=S\_IFREG|0644, st\_size=542, ...}, 0) = 0

20387 newfstatat(AT\_FDCWD, "/", {st\_mode=S\_IFDIR|0755, st\_size=4096, ...}, 0) = 0

20387 openat(AT\_FDCWD, "/etc/nsswitch.conf", O\_RDONLY|O\_CLOEXEC) = 3

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=542, ...}, AT\_EMPTY\_PATH) = 0

20387 read(3, "# /etc/nsswitch.conf\n#\n# Example"..., 4096) = 542

20387 read(3, "", 4096) = 0

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=542, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=81715, ...}, AT\_EMPTY\_PATH) = 0

20387 close(3) = 0

20387 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)

20387 newfstatat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/glibc-hwcaps/x86-64-v4",

0x7ffe4ad456c0, 0) = -1 ENOENT (No such file or directory)

...openat+newfstatat…

20387 openat(AT\_FDCWD, "/etc/protocols", O\_RDONLY|O\_CLOEXEC) = 3

20387 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=2932, ...}, AT\_EMPTY\_PATH) = 0

20387 lseek(3, 0, SEEK\_SET) = 0

20387 read(3, "# Internet (IP) protocols\n#\n# Up"..., 4096) = 2932

20387 read(3, "", 4096) = 0

20387 close(3) = 0

20387 eventfd2(0, EFD\_CLOEXEC) = 3

20387 fcntl(3, F\_GETFL) = 0x2 (flags O\_RDWR)

20387 fcntl(3, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

20387 fcntl(3, F\_GETFL) = 0x802 (flags O\_RDWR|O\_NONBLOCK)

20387 fcntl(3, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

20387 getpid() = 20387

20387 getpid() = 20387

20387 getrandom("\x44\x22\x00\x61\x70\xa2\x3a\xe2\xe4\x3b\x22\x87\xa3\x11\xb3\xff", 16, 0) = 16

20387 getrandom("\x0d\x1d\xd1\x8b\xc8\x9a\x35\xc7\x37\x4b\x70\x6c\x57\xdd\xe8\xad", 16, 0) = 16

20387 eventfd2(0, EFD\_CLOEXEC) = 4

20387 fcntl(4, F\_GETFL) = 0x2 (flags O\_RDWR)

20387 fcntl(4, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

20387 fcntl(4, F\_GETFL) = 0x802 (flags O\_RDWR|O\_NONBLOCK)

20387 fcntl(4, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

20387 getpid() = 20387

20387 epoll\_create1(EPOLL\_CLOEXEC) = 5

20387 epoll\_ctl(5, EPOLL\_CTL\_ADD, 4, {events=0, data={u32=1509016160,

u64=94507969397344}}) = 0

20387 epoll\_ctl(5, EPOLL\_CTL\_MOD, 4, {events=EPOLLIN, data={u32=1509016160,

u64=94507969397344}}) = 0

20387 getpid() = 20387

20387 rt\_sigaction(SIGRT\_1, {sa\_handler=0x7f57c3291870, sa\_mask=[],

sa\_flags=SA\_RESTORER|SA\_ONSTACK|SA\_RESTART|SA\_SIGINFO,

sa\_restorer=0x7f57c3242520}, NULL, 8) = 0

20387 rt\_sigprocmask(SIG\_UNBLOCK, [RTMIN RT\_1], NULL, 8) = 0

20387 rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

20387clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_T\

HREAD|CL

ONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID,

child\_tid=0x7f57c2fb6910, parent\_tid=0x7f57c2fb6910, exit\_signal=0, stack=0x7f57c27b6000,

stack\_size=0x7ffc80, tls=0x7f57c2fb6640} => {parent\_tid=[20389]}, 88) = 20389

20389 rseq(0x7f57c2fb6fe0, 0x20, 0, 0x53053053 <unfinished ...>

20387 rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

20389 <... rseq resumed>) = 0

20387 <... rt\_sigprocmask resumed>NULL, 8) = 0

20389 set\_robust\_list(0x7f57c2fb6920, 24 <unfinished ...>

20387 eventfd2(0, EFD\_CLOEXEC <unfinished ...>

20389 <... set\_robust\_list resumed>) = 0

20387 <... eventfd2 resumed>) = 6

20389 rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

20387 fcntl(6, F\_GETFL <unfinished ...>

20389 <... rt\_sigprocmask resumed>NULL, 8) = 0

20387 <... fcntl resumed>) = 0x2 (flags O\_RDWR)

20387 fcntl(6, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

20389 rt\_sigprocmask(SIG\_BLOCK, ~[RTMIN RT\_1], <unfinished ...>

20387 fcntl(6, F\_GETFL <unfinished ...>

20389 <... rt\_sigprocmask resumed>NULL, 8) = 0

20387 <... fcntl resumed>) = 0x802 (flags O\_RDWR|O\_NONBLOCK)

20387 fcntl(6, F\_SETFL, O\_RDWR|O\_NONBLOCK <unfinished ...>

20389 sched\_getparam(20389, <unfinished ...>

20387 <... fcntl resumed>) = 0

20389 <... sched\_getparam resumed>[0]) = 0

20387 getpid( <unfinished ...>

20389 sched\_getscheduler(20389 <unfinished ...>

20387 <... getpid resumed>) = 20387

20389 <... sched\_getscheduler resumed>) = 0 (SCHED\_OTHER)

20387 epoll\_create1(EPOLL\_CLOEXEC <unfinished ...>

20389 sched\_setscheduler(20389, SCHED\_OTHER, [0] <unfinished ...>

20387 <... epoll\_create1 resumed>) = 7

20387 epoll\_ctl(7, EPOLL\_CTL\_ADD, 6, {events=0, data={u32=1509037216,

u64=94507969418400}} <unfinished ...>

20389 <... sched\_setscheduler resumed>) = 0

20387 <... epoll\_ctl resumed>) = 0

20387 epoll\_ctl(7, EPOLL\_CTL\_MOD, 6, {events=EPOLLIN, data={u32=1509037216,

u64=94507969418400}} <unfinished ...>

20389 prctl(PR\_SET\_NAME, "ZMQbg/Reaper" <unfinished ...>

20387 <... epoll\_ctl resumed>) = 0

20389 <... prctl resumed>) = 0

20389 epoll\_wait(5, <unfinished ...>

20387 rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

20387

clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CL

ONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID,

child\_tid=0x7f57c27b5910, parent\_tid=0x7f57c27b5910, exit\_signal=0, stack=0x7f57c1fb5000,

stack\_size=0x7ffc80, tls=0x7f57c27b5640} => {parent\_tid=[20390]}, 88) = 20390

20387 rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

20387 eventfd2(0, EFD\_CLOEXEC) = 8

20387 fcntl(8, F\_GETFL) = 0x2 (flags O\_RDWR)

20387 fcntl(8, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

20387 fcntl(8, F\_GETFL <unfinished ...>

20390 rseq(0x7f57c27b5fe0, 0x20, 0, 0x53053053 <unfinished ...>

20387 <... fcntl resumed>) = 0x802 (flags O\_RDWR|O\_NONBLOCK)

20387 fcntl(8, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

20387 getpid() = 20387

20387 eventfd2(0, EFD\_CLOEXEC) = 9

20387 fcntl(9, F\_GETFL) = 0x2 (flags O\_RDWR)

20390 <... rseq resumed>) = 0

20387 fcntl(9, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

20390 set\_robust\_list(0x7f57c27b5920, 24 <unfinished ...>

20387 fcntl(9, F\_GETFL) = 0x802 (flags O\_RDWR|O\_NONBLOCK)

20390 <... set\_robust\_list resumed>) = 0

20387 fcntl(9, F\_SETFL, O\_RDWR|O\_NONBLOCK) = 0

20390 rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

20387 getpid() = 20387

20390 <... rt\_sigprocmask resumed>NULL, 8) = 0

20387 getpid( <unfinished ...>

20390 rt\_sigprocmask(SIG\_BLOCK, ~[RTMIN RT\_1], <unfinished ...>

20387 <... getpid resumed>) = 20387

20390 <... rt\_sigprocmask resumed>NULL, 8) = 0

20387 poll([{fd=8, events=POLLIN}], 1, 0 <unfinished ...>

20390 sched\_getparam(20390, <unfinished ...>

20387 <... poll resumed>) = 0 (Timeout)

20390 <... sched\_getparam resumed>[0]) = 0

20390 sched\_getscheduler(20390 <unfinished ...>

20387 socket(AF\_NETLINK, SOCK\_RAW|SOCK\_CLOEXEC, NETLINK\_ROUTE <unfinished ...>

20390 <... sched\_getscheduler resumed>) = 0 (SCHED\_OTHER)

20387 <... socket resumed>) = 10

20390 sched\_setscheduler(20390, SCHED\_OTHER, [0] <unfinished ...>

20387 bind(10, {sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, 12 <unfinished ...>

20390 <... sched\_setscheduler resumed>) = 0

20387 <... bind resumed>) = 0

20390 prctl(PR\_SET\_NAME, "ZMQbg/IO/0" <unfinished ...>

20387 getsockname(10, {sa\_family=AF\_NETLINK, nl\_pid=20387, nl\_groups=00000000}, [12]) = 0

20390 <... prctl resumed>) = 0

20387 sendto(10, [{nlmsg\_len=20, nlmsg\_type=RTM\_GETLINK, nlmsg\_flags=NLM\_F\_REQUEST|NLM\_F\_DUMP, nlmsg\_seq=1704658042, nlmsg\_pid=0}, {ifi\_family=AF\_UNSPEC, ...}], 20, 0, {sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, 12 <unfinished ...>

20390 epoll\_wait(7, <unfinished ...>

20387 <... sendto resumed>) = 20

20387 recvmsg(10, {msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, msg\_namelen=12, msg\_iov=[{iov\_base=[[{nlmsg\_len=1404, nlmsg\_type=RTM\_NEWLINK, nlmsg\_flags=NLM\_F\_MULTI, nlmsg\_seq=1704658042, nlmsg\_pid=20387}, {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=0x3d /\* IFLA\_??? \*/}, "\x00\x00\x00\x00"], [{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=0x3a /\* IFLA\_??? \*/}, "\x00\x00\x01\x00"], [{nla\_len=8, nla\_type=0x3b /\* IFLA\_??? \*/}, "\xf8\xff\x07\x00"], [{nla\_len=8, nla\_type=0x3c /\* IFLA\_??? \*/}, "\xff\xff\x00\x00"], [{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}, 00:00:00:00:00:00], [{nla\_len=10, nla\_type=IFLA\_BROADCAST}, 00:00:00:00:00:00], [{nla\_len=204, nla\_type=IFLA\_STATS64}, {rx\_packets=18620, tx\_packets=18620, rx\_bytes=1990342, tx\_bytes=1990342, 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=18620, tx\_packets=18620, rx\_bytes=1990342, tx\_bytes=1990342, 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=804, nla\_type=IFLA\_AF\_SPEC}, [[{nla\_len=12, nla\_type=AF\_MCTP}, [{nla\_len=8, nla\_type=IFLA\_MCTP\_NET}, 1]], [{nla\_len=140, nla\_type=AF\_INET}, [{nla\_len=136, nla\_type=IFLA\_INET\_CONF}, [[IPV4\_DEVCONF\_FORWARDING-1] = 0, [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] = 2, [IPV4\_DEVCONF\_ACCEPT\_SOURCE\_ROUTE-1] = 0, [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] = 1, [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=648, 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=170, reachable\_time=21796, retrans\_time=1000}], [{nla\_len=236, 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] = 1, [DEVCONF\_RTR\_PROBE\_INTERVAL] = 60000, [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] = 6, [IPSTATS\_MIB\_INOCTETS] = 432, [IPSTATS\_MIB\_INDELIVERS] = 6, [IPSTATS\_MIB\_OUTFORWDATAGRAMS] = 0, [IPSTATS\_MIB\_OUTPKTS] = 6, [IPSTATS\_MIB\_OUTOCTETS] = 432, [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] = 2, [IPSTATS\_MIB\_INBCASTPKTS] = 0, [IPSTATS\_MIB\_OUTBCASTPKTS] = 0, [IPSTATS\_MIB\_INMCASTOCTETS] = 0, [IPSTATS\_MIB\_OUTMCASTOCTETS] = 152, [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] = 2, [ICMP6\_MIB\_INERRORS] = 0, [ICMP6\_MIB\_OUTMSGS] = 2, [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]]]]], {nla\_len=4, nla\_type=NLA\_F\_NESTED|0x3e /\* IFLA\_??? \*/}]], [{nlmsg\_len=1440, nlmsg\_type=RTM\_NEWLINK, nlmsg\_flags=NLM\_F\_MULTI, nlmsg\_seq=1704658042, nlmsg\_pid=20387}, {ifi\_family=AF\_UNSPEC, ifi\_type=ARPHRD\_ETHER, ifi\_index=if\_nametoindex("enp3s0"), ifi\_flags=IFF\_UP|IFF\_BROADCAST|IFF\_MULTICAST, ifi\_change=0}, [[{nla\_len=11, nla\_type=IFLA\_IFNAME}, "enp3s0"], [{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}, 9194], [{nla\_len=8, nla\_type=IFLA\_GROUP}, 0], [{nla\_len=8, nla\_type=IFLA\_PROMISCUITY}, 0], [{nla\_len=8, nla\_type=0x3d /\* IFLA\_??? \*/}, "\x00\x00\x00\x00"], [{nla\_len=8, nla\_type=IFLA\_NUM\_TX\_QUEUES}, 1], [{nla\_len=8, nla\_type=IFLA\_GSO\_MAX\_SEGS}, 64], [{nla\_len=8, nla\_type=IFLA\_GSO\_MAX\_SIZE}, 64000], [{nla\_len=8, nla\_type=0x3a /\* IFLA\_??? \*/}, "\x00\x00\x01\x00"], [{nla\_len=8, nla\_type=0x3b /\* IFLA\_??? \*/}, "\x00\xfa\x00\x00"], [{nla\_len=8, nla\_type=0x3c /\* IFLA\_??? \*/}, "\x40\x00\x00\x00"], [{nla\_len=8, nla\_type=IFLA\_NUM\_RX\_QUEUES}, 1], [{nla\_len=5, nla\_type=IFLA\_CARRIER}, 0], [{nla\_len=13, nla\_type=IFLA\_QDISC}, "fq\_codel"], [{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}, 04:7c:16:31:22:f5], [{nla\_len=10, nla\_type=IFLA\_BROADCAST}, ff:ff:ff:ff:ff:ff], [{nla\_len=204, 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=10, nla\_type=IFLA\_PERM\_ADDRESS}, 04:7c:16:31:22:f5], [{nla\_len=792, nla\_type=IFLA\_AF\_SPEC}, [[{nla\_len=140, nla\_type=AF\_INET}, [{nla\_len=136, nla\_type=IFLA\_INET\_CONF}, [[IPV4\_DEVCONF\_FORWARDING-1] = 0, [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] = 2, [IPV4\_DEVCONF\_ACCEPT\_SOURCE\_ROUTE-1] = 0, [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] = 1, [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=648, 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=1058776, reachable\_time=17116, retrans\_time=1000}], [{nla\_len=236, nla\_type=IFLA\_INET6\_CONF}, [[DEVCONF\_FORWARDING] = 0, [DEVCONF\_HOPLIMIT] = 64, [DEVCONF\_MTU6] = 1500, [DEVCONF\_ACCEPT\_RA] = 0, [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] = 1, [DEVCONF\_RTR\_PROBE\_INTERVAL] = 60000, [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\_NONE]]]]], [{nla\_len=17, nla\_type=IFLA\_PARENT\_DEV\_NAME}, "0000:03:00.0"], ...]]], iov\_len=4096}], msg\_iovlen=1, msg\_controllen=0, msg\_flags=0}, 0) = 2844

20387 recvmsg(10, {msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000},

…)

20387 recvmsg(10, {msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, msg\_namelen=12, msg\_iov=[{iov\_base=[{nlmsg\_len=20, nlmsg\_type=NLMSG\_DONE, nlmsg\_flags=NLM\_F\_MULTI, nlmsg\_seq=1704658042, nlmsg\_pid=20387}, 0], iov\_len=4096}], msg\_iovlen=1, msg\_controllen=0, msg\_flags=0}, 0) = 20

20387 sendto(10, [{nlmsg\_len=20, nlmsg\_type=RTM\_GETADDR, nlmsg\_flags=NLM\_F\_REQUEST|NLM\_F\_DUMP, nlmsg\_seq=1704658043, nlmsg\_pid=0}, {ifa\_family=AF\_UNSPEC, ...}], 20, 0, {sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, 12) = 20

20387 recvmsg(10, {msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000},

…)

20387 recvmsg(10, {msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000},

…)

20387 recvmsg(10, {msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, msg\_namelen=12, msg\_iov=[{iov\_base=[{nlmsg\_len=20, nlmsg\_type=NLMSG\_DONE, nlmsg\_flags=NLM\_F\_MULTI, nlmsg\_seq=1704658043, nlmsg\_pid=20387}, 0], iov\_len=4096}], msg\_iovlen=1, msg\_controllen=0, msg\_flags=0}, 0) = 20

20387 close(10) = 0

20387 socket(AF\_INET, SOCK\_STREAM|SOCK\_CLOEXEC, IPPROTO\_TCP) = 10

20387 setsockopt(10, SOL\_SOCKET, SO\_REUSEADDR, [1], 4) = 0

20387 bind(10, {sa\_family=AF\_INET, sin\_port=htons(5555), sin\_addr=inet\_addr("127.0.0.1")}, 16) = 0

20387 listen(10, 100) = 0

20387 getsockname(10, {sa\_family=AF\_INET, sin\_port=htons(5555), sin\_addr=inet\_addr("127.0.0.1")}, [128 => 16]) = 0

20387 getsockname(10, {sa\_family=AF\_INET, sin\_port=htons(5555), sin\_addr=inet\_addr("127.0.0.1")}, [128 => 16]) = 0

20387 getpid() = 20387

20387 write(6, "\1\0\0\0\0\0\0\0", 8) = 8

20387 getpid( <unfinished ...>

20390 <... epoll\_wait resumed>[{events=EPOLLIN, data={u32=1509037216, u64=94507969418400}}], 256, -1) = 1

20387 <... getpid resumed>) = 20387

20387 write(8, "\1\0\0\0\0\0\0\0", 8 <unfinished ...>

20390 getpid( <unfinished ...>

20387 <... write resumed>) = 8

20390 <... getpid resumed>) = 20387

20387 getpid( <unfinished ...>

20390 poll([{fd=6, events=POLLIN}], 1, 0 <unfinished ...>

20387 <... getpid resumed>) = 20387

20390 <... poll resumed>) = 1 ([{fd=6, revents=POLLIN}])

20387 poll([{fd=9, events=POLLIN}], 1, 0 <unfinished ...>

20390 getpid( <unfinished ...>

20387 <... poll resumed>) = 0 (Timeout)

20390 <... getpid resumed>) = 20387

20387 socket(AF\_NETLINK, SOCK\_RAW|SOCK\_CLOEXEC, NETLINK\_ROUTE <unfinished ...>

20390 read(6, <unfinished ...>

20387 <... socket resumed>) = 11

20390 <... read resumed>"\1\0\0\0\0\0\0\0", 8) = 8

20387 bind(11, {sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, 12) = 0

20387 getsockname(11, {sa\_family=AF\_NETLINK, nl\_pid=20387, nl\_groups=00000000}, [12]) = 0

20387 sendto(11, [{nlmsg\_len=20, nlmsg\_type=RTM\_GETLINK, nlmsg\_flags=NLM\_F\_REQUEST|NLM\_F\_DUMP, nlmsg\_seq=1704658042, nlmsg\_pid=0}, {ifi\_family=AF\_UNSPEC, ...}], 20, 0, {sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, 12 <unfinished ...>

20390 epoll\_ctl(7, EPOLL\_CTL\_ADD, 10, {events=0, data={u32=3154119536, u64=140014793001840}} <unfinished ...>

20387 <... sendto resumed>) = 20

20390 <... epoll\_ctl resumed>) = 0

20387 recvmsg(11, <unfinished ...>

20390 epoll\_ctl(7, EPOLL\_CTL\_MOD, 10, {events=EPOLLIN, data={u32=3154119536, u64=140014793001840}}) = 0

20387 <... recvmsg resumed>{msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0,

…)

20390 getpid( <unfinished ...>

20387 recvmsg(11, <unfinished ...>

20390 <... getpid resumed>) = 20387

20387 <... recvmsg resumed>{msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0,

…)

20390 poll([{fd=6, events=POLLIN}], 1, 0 <unfinished ...>

20387 recvmsg(11, <unfinished ...>

20390 <... poll resumed>) = 0 (Timeout)

20387 <... recvmsg resumed>{msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, msg\_namelen=12, msg\_iov=[{iov\_base=[{nlmsg\_len=20, nlmsg\_type=NLMSG\_DONE, nlmsg\_flags=NLM\_F\_MULTI, nlmsg\_seq=1704658042, nlmsg\_pid=20387}, 0], iov\_len=4096}], msg\_iovlen=1, msg\_controllen=0, msg\_flags=0}, 0) = 20

20390 epoll\_wait(7, <unfinished ...>

20387 sendto(11, [{nlmsg\_len=20, nlmsg\_type=RTM\_GETADDR, nlmsg\_flags=NLM\_F\_REQUEST|NLM\_F\_DUMP, nlmsg\_seq=1704658043, nlmsg\_pid=0}, {ifa\_family=AF\_UNSPEC, ...}], 20, 0, {sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, 12) = 20

20387 recvmsg(11, {msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000},

…)

20387 recvmsg(11, {msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000},

…)

20387 recvmsg(11, {msg\_name={sa\_family=AF\_NETLINK, nl\_pid=0, nl\_groups=00000000}, msg\_namelen=12, msg\_iov=[{iov\_base=[{nlmsg\_len=20, nlmsg\_type=NLMSG\_DONE, nlmsg\_flags=NLM\_F\_MULTI, nlmsg\_seq=1704658043, nlmsg\_pid=20387}, 0], iov\_len=4096}], msg\_iovlen=1, msg\_controllen=0, msg\_flags=0}, 0) = 20

20387 close(11) = 0

20387 socket(AF\_INET, SOCK\_STREAM|SOCK\_CLOEXEC, IPPROTO\_TCP) = 11

20387 setsockopt(11, SOL\_SOCKET, SO\_REUSEADDR, [1], 4) = 0

20387 bind(11, {sa\_family=AF\_INET, sin\_port=htons(5556), sin\_addr=inet\_addr("127.0.0.1")}, 16) = 0

20387 listen(11, 100) = 0

20387 getsockname(11, {sa\_family=AF\_INET, sin\_port=htons(5556), sin\_addr=inet\_addr("127.0.0.1")}, [128 => 16]) = 0

20387 getsockname(11, {sa\_family=AF\_INET, sin\_port=htons(5556), sin\_addr=inet\_addr("127.0.0.1")}, [128 => 16]) = 0

20387 getpid() = 20387

20387 write(6, "\1\0\0\0\0\0\0\0", 8) = 8

20390 <... epoll\_wait resumed>[{events=EPOLLIN, data={u32=1509037216, u64=94507969418400}}], 256, -1) = 1

20387 getpid() = 20387

20390 getpid( <unfinished ...>

20387 write(9, "\1\0\0\0\0\0\0\0", 8 <unfinished ...>

20390 <... getpid resumed>) = 20387

20387 <... write resumed>) = 8

20390 poll([{fd=6, events=POLLIN}], 1, 0) = 1 ([{fd=6, revents=POLLIN}])

20387 newfstatat(1, "", <unfinished ...>

20390 getpid( <unfinished ...>

20387 <... newfstatat resumed>{st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x1), ...}, AT\_EMPTY\_PATH) = 0

20390 <... getpid resumed>) = 20387

20387 write(1, "Welcome in our programm! This is"..., 56 <unfinished ...>

20390 read(6, <unfinished ...>

20387 <... write resumed>) = 56

20390 <... read resumed>"\1\0\0\0\0\0\0\0", 8) = 8

20387 write(1, " - create 'id' \n", 16 <unfinished ...>

20390 epoll\_ctl(7, EPOLL\_CTL\_ADD, 11, {events=0, data={u32=3154119568, u64=140014793001872}} <unfinished ...>

20387 <... write resumed>) = 16

20390 <... epoll\_ctl resumed>) = 0

20387 write(1, " - exec 'id' 'command(start/stop"..., 41 <unfinished ...>

20390 epoll\_ctl(7, EPOLL\_CTL\_MOD, 11, {events=EPOLLIN, data={u32=3154119568, u64=140014793001872}} <unfinished ...>

20387 <... write resumed>) = 41

20390 <... epoll\_ctl resumed>) = 0

20387 write(1, " - heartbit 'time' (in ms) \n", 28 <unfinished ...>

20390 getpid( <unfinished ...>

20387 <... write resumed>) = 28

20390 <... getpid resumed>) = 20387

20387 write(1, " - draw\n", 8 <unfinished ...>

20390 poll([{fd=6, events=POLLIN}], 1, 0 <unfinished ...>

20387 <... write resumed>) = 8

20390 <... poll resumed>) = 0 (Timeout)

20387 write(1, "Or enter q or ^D to exit\n", 25 <unfinished ...>

20390 epoll\_wait(7, <unfinished ...>

20387 <... write resumed>) = 25

20387 write(1, "Enter you command: \n", 20) = 20

20387 write(1, " ->", 3) = 3

20387 newfstatat(0, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x1), ...},

AT\_EMPTY\_PATH) = 0

20387 read(0, "create 2\n", 1024) = 9

20387 clone(child\_stack=NULL,

flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD,

child\_tidptr=0x7f57c3833c90) = 20397

20397 set\_robust\_list(0x7f57c3833ca0, 24 <unfinished ...>

20387 write(1, "Ok: 20397\n", 10 <unfinished ...>

20397 <... set\_robust\_list resumed>) = 0

20387 <... write resumed>) = 10

20387 write(1, " ->", 3) = 3

20387 read(0, <unfinished ...>

20397 execve("./node", ["./node", "2"], 0x7ffe4ad488a8 /\* 56 vars \*/) = 0

20397 arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffd7f3e2270) = -1 EINVAL (Invalid argument)

20397 access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

20397 openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

20397 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=81715, ...}, AT\_EMPTY\_PATH) = 0

20397 close(3) = 0

20397 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libzmq.so.5", O\_RDONLY|O\_CLOEXEC) = 3

20397 read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\240\233\1\0\0\0\0\0"..., 832) = 832

20397 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=634936, ...}, AT\_EMPTY\_PATH) = 0

20397 close(3) = 0

*…Где фулл? А фулл лежит в логе…*

20389 poll([{fd=4, events=POLLIN}], 1, 0 <unfinished ...>

20387 write(6, "\1\0\0\0\0\0\0\0", 8 <unfinished ...>

20389 <... poll resumed>) = 0 (Timeout)

20387 <... write resumed>) = 8

20390 <... epoll\_wait resumed>[{events=EPOLLIN, data={u32=1509037216, u64=94507969418400}}], 256, -1) = 1

20387 futex(0x7f57c27b5910, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 20390, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

20389 rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

20390 getpid( <unfinished ...>

20389 <... rt\_sigprocmask resumed>NULL, 8) = 0

20390 <... getpid resumed>) = 20387

20389 madvise(0x7f57c27b6000, 8368128, MADV\_DONTNEED <unfinished ...>

20390 poll([{fd=6, events=POLLIN}], 1, 0 <unfinished ...>

20389 <... madvise resumed>) = 0

20390 <... poll resumed>) = 1 ([{fd=6, revents=POLLIN}])

20389 exit(0 <unfinished ...>

20390 getpid( <unfinished ...>

20389 <... exit resumed>) = ?

20390 <... getpid resumed>) = 20387

20390 read(6, <unfinished ...>

20389 +++ exited with 0 +++

20390 <... read resumed>"\1\0\0\0\0\0\0\0", 8) = 8

20390 epoll\_ctl(7, EPOLL\_CTL\_DEL, 6, 0x55f459f214a4) = 0

20390 getpid() = 20387

20390 poll([{fd=6, events=POLLIN}], 1, 0) = 0 (Timeout)

20390 rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], NULL, 8) = 0

20390 madvise(0x7f57c1fb5000, 8368128, MADV\_DONTNEED) = 0

20390 exit(0) = ?

20387 <... futex resumed>) = 0

20390 +++ exited with 0 +++

20387 close(7) = 0

20387 close(6) = 0

20387 close(5) = 0

20387 close(4) = 0

20387 close(3) = 0

20387 lseek(0, -1, SEEK\_CUR) = -1 ESPIPE (Illegal seek)

20387 exit\_group(0) = ?

20387 +++ exited with 0 +++

20399 <... epoll\_wait resumed> <unfinished ...>) = ?

20398 <... epoll\_wait resumed> <unfinished ...>) = ?

20397 <... poll resumed> <unfinished ...>) = ?

20399 +++ killed by SIGKILL +++

20398 +++ killed by SIGKILL +++

20404 <... epoll\_wait resumed> <unfinished ...>) = ?

20397 +++ killed by SIGKILL +++

20403 <... epoll\_wait resumed> <unfinished ...>) = ?

20404 +++ killed by SIGKILL +++

20402 <... poll resumed> <unfinished ...>) = ?

20403 +++ killed by SIGKILL +++

20402 +++ killed by SIGKILL +++

**Вывод**

Эта лабораторная работа фактически завершает наше знакомство с курсом ОС и в принципе с темой IPC. И завершает его, знакомя нас с последним и самым популярным способом передачи данных между разными процессами - сокетами, в нашем случае в обертке очередей сообщений.

Честно скажу, что эта лабораторная мне очень понравилась, т.к. было поставлено интересное ТЗ и предоставлен достаточно простой в освоении, но при этом очень мощный инструмент в видео очередей сообщений.

В итоге я научился работать с очередями сообщений, проникся этой темой и смог написать исправно работающий код, поэтому считаю, что с поставленной задачей справился успешно.