# Московский авиационный институт (национальный исследовательский университет)

Институт №8 «Информационные технологии и прикладная математика» Кафедра 806 «Вычислительная математика и программирование» Дисциплина «Операционные системы»

Лабораторные работы №5-7

Тема: Управление серверами сообщений, применение отложенный вычислений, интеграция программных систем друг с другом

Студент: Недосекин А. А. Группа: М8О-209Б-22

Преподаватель: Гапонов Н.А.

Дата:

Оценка:

Москва, 2023

# Постановка задачи

Реализовать распределенную систему по асинхронной обработке запросов. В данной распределенной системе должно существовать два вида узлов:

«управляющий» и «вычислительный». Необходимо объединить данные узлы в соответствии с той топологией, которая определена вариантом. Связь между узлами необходимо осуществить при помощи технологии очередей сообщений. Также в данной системе необходимо предусмотреть проверку доступности узлов в соответствии с вариантом. При убийстве («kill -9») любого вычислительного узла система должна пытаться максимально сохранять свою работоспособность, а именно все дочерние узлы убитого узла могут стать недоступными, но родительские узлы должны сохранить свою работоспособность.

Управляющий узел отвечает за ввод команд от пользователя и отправку этих команд на вычислительные узлы. Список основных поддерживаемых команд:

* Создание нового вычислительного узла;
* Удаление существующего вычислительного узла;
* Исполнение команды на вычислительном узле;
* Проверка доступности вычислительного узла.

*Вариант 31.* Все вычислительные узлы находятся в списке. Есть только один управляющий узел.

Исполнение команды — поиск подстроки в строке.

Команда проверки — проверка доступности конкретного узла.

# Описание программы

Связь между вычислительными узлами будем поддерживать с помощью *ZMQ\_PAIR*. При инициализации установить время ожидания *ZMQ\_SNDTIMEO* и *ZMQ\_RECVTIMEO*, чтобы предусмотреть случай, когда дочерний процесс был убит. Для обмена информацией будем использовать специальную структуру *node\_token\_t*, в которой есть перечислимое поле *actions*. Вычислительные узлы обрабатывают каждое сообщение: если идентификатор сообщения не совпадает с идентификатором узла, то он отправляет сообщение дальше и ждёт ответа снизу. Каждый вычислительный узел имеет отдельный поток для вычислений и свою очередь вычислений. Чтобы получить результат вычислений обратно, нужно запросить их от вычислительного узла. Такой подход необходим, потому что неизвестно, сколько нужно ждать результат от узла. Для поиска подстроки в строке я использовал алгоритм Кнута-Морриса-Пратта с препроцессингом через Z-функцию строки.

# Набор тестов

Программа обрабатывает команды пользователя до окончания ввода.

*Тест 1*

create 1 -1

ping 1 exec 1 a b

exec 1 a aaa

exec 1 abra abracadabra back 1

ping 1

back 1

ping 1

back 1

ping 1

remove 1

*Тест 2*

create 1 -1

create 2 -1

create 3 -1

create 4 -1

create 5 -1

exec 4 aba abacababacaba exec 1 aba abacababacaba exec 5 aba abacababacaba exec 1 aba abacababacaba exec 3 aba abacababacaba ping 1

ping 2

ping 3

ping 4

ping 5

back 1

back 1

back 3

back 4

back 5

remove 4

remove 1

remove 3

remove 2

remove 5

*Тест 3*

create 1 -1

create 2 1

create 5 -1

create 3 2

create 6 5

create 7 5

create 4 3

create 8 5

create 9 8

create 0 1

remove 8

remove 4

remove 2

remove 6

remove 0

ping 1

ping 3

ping 5

ping 7

ping 9

remove 5

remove 7

remove 1

remove 9

remove 3

# Результат выполнения тестов

Программа выводит на экран результат обработки каждой команды.

*Тест 1*

OK: 17493

OK: 1 OK OK OK

OK: 17493 : -1

OK: 1

OK: 17493 : 0, 1, 2

OK: 1

OK: 17493 : 0, 7

OK: 1 OK

*Тест 2*

OK: 17711

OK: 17717

OK: 17723

OK: 17729

OK: 17735 OK

OK OK OK OK OK: 1

OK: 1

OK: 1

OK: 1

OK: 1

OK: 17711 : 0, 4, 6, 10

OK: 17717 : 0, 4, 6, 10

OK: 17723 : 0, 4, 6, 10

OK: 17729 : 0, 4, 6, 10

OK: 17735 : 0, 4, 6, 10 OK

OK OK OK OK

*Тест 3*

OK: 17032

OK: 17038

OK: 17044

OK: 17050

OK: 17056

OK: 17062

OK: 17070

OK: 17076

OK: 17084

OK: 17092 OK

OK OK OK OK OK: 1

OK: 1

OK: 1

OK: 1

OK: 1 OK OK OK OK OK

# Листинг программы

Для удобства функции инициализации сокета, получения и отправки сообщения вынесены в отдельный файл zmq\_std.hpp, топология в topology.hpp. В файле control.cpp расположен код для управляющего узла, а в calculation\_node.cpp для вычислительного узла. Функции для поиска подстроки находятся в файлах search.hpp и search.cpp

zmq\_std.hpp

#ifndef ZMQ\_STD\_HPP #define ZMQ\_STD\_HPP #include <assert.h> #include <errno.h> #include <string.h> #include <string> #include <zmq.h>

const char\* NODE\_EXECUTABLE\_NAME = "calculation"; const char SENTINEL = '$';

const int PORT\_BASE = 8000; const int WAIT\_TIME = 1000;

enum actions\_t {

fail = 0,

success = 1,

create = 2,

destroy = 3,

bind = 4,

ping = 5,

exec = 6,

info = 7,

back = 8

};

struct node\_token\_t { actions\_t action;

long long parent\_id, id;

};

namespace zmq\_std {

void init\_pair\_socket(void\* & context, void\* & socket) { int rc;

context = zmq\_ctx\_new();

socket = zmq\_socket(context, ZMQ\_PAIR);

rc = zmq\_setsockopt(socket, ZMQ\_RCVTIMEO, &WAIT\_TIME, sizeof(int));

assert(rc == 0);

rc = zmq\_setsockopt(socket, ZMQ\_SNDTIMEO, &WAIT\_TIME, sizeof(int));

assert(rc == 0);

}

template<class T>

void recieve\_msg(T & reply\_data, void\* socket) { int rc = 0;

zmq\_msg\_t reply; zmq\_msg\_init(&reply);

rc = zmq\_msg\_recv(&reply, socket, 0); assert(rc == sizeof(T));

reply\_data = \*(T\*)zmq\_msg\_data(&reply); rc = zmq\_msg\_close(&reply);

assert(rc == 0);

}

template<class T>

void send\_msg(T\* token, void\* socket) { int rc = 0;

zmq\_msg\_t message; zmq\_msg\_init(&message);

rc = zmq\_msg\_init\_size(&message, sizeof(T)); assert(rc == 0);

rc = zmq\_msg\_init\_data(&message, token, sizeof(T), NULL,

NULL);

}

assert(rc == 0);

rc = zmq\_msg\_send(&message, socket, 0); assert(rc == sizeof(T));

template<class T>

bool send\_msg\_dontwait(T\* token, void\* socket) { int rc;

zmq\_msg\_t message; zmq\_msg\_init(&message);

rc = zmq\_msg\_init\_size(&message, sizeof(T)); assert(rc == 0);

rc = zmq\_msg\_init\_data(&message, token, sizeof(T), NULL,

NULL);

assert(rc == 0);

rc = zmq\_msg\_send(&message, socket, ZMQ\_DONTWAIT); if (rc == -1) {

zmq\_msg\_close(&message); return false;

}

assert(rc == sizeof(T)); return true;

}

template<class T>

bool recieve\_msg\_wait(T & reply\_data, void\* socket) { int rc = 0;

zmq\_msg\_t reply; zmq\_msg\_init(&reply);

rc = zmq\_msg\_recv(&reply, socket, 0);

if (rc == -1) {

zmq\_msg\_close(&reply); return false;

}

assert(rc == sizeof(T));

reply\_data = \*(T\*)zmq\_msg\_data(&reply); rc = zmq\_msg\_close(&reply);

assert(rc == 0); return true;

}

/\* Returns true if T was successfully queued on the socket \*/ template<class T>

bool send\_msg\_wait(T\* token, void\* socket) { int rc;

zmq\_msg\_t message; zmq\_msg\_init(&message);

rc = zmq\_msg\_init\_size(&message, sizeof(T)); assert(rc == 0);

rc = zmq\_msg\_init\_data(&message, token, sizeof(T), NULL,

NULL);

assert(rc == 0);

rc = zmq\_msg\_send(&message, socket, 0); if (rc == -1) {

zmq\_msg\_close(&message); return false;

}

assert(rc == sizeof(T)); return true;

}

/\*

* Returns true if socket successfully queued
* message and recieved reply

\*/ template<class T>

bool send\_recieve\_wait(T\* token\_send, T & token\_reply, void\* socket) {

if (send\_msg\_wait(token\_send, socket)) {

if (recieve\_msg\_wait(token\_reply, socket)) { return true;

} else {

return false;

}

} else {

return false;

}

}

}

#endif /\* ZMQ\_STD\_HPP \*/

topology.hpp

#ifndef TOPOLOGY\_HPP #define TOPOLOGY\_HPP

#include <iostream> #include <list>

template<class T> class topology\_t { private:

using list\_type = std::list< std::list<T> >; using iterator = typename std::list<T>::iterator;

using list\_iterator = typename list\_type::iterator;

list\_type container; size\_t container\_size;

public:

explicit topology\_t() noexcept : container(), container\_size(0) {}

~topology\_t() {}

bool erase(const T & elem) {

for (list\_iterator it1 = container.begin(); it1 != container.end(); ++it1) {

for (iterator it2 = it1->begin(); it2 != it1->end();

++it2) {

if (\*it2 == elem) {

if (it1->size() > 1) { it1->erase(it2);

} else {

container.erase(it1);

}

--container\_size; return true;

}

}

}

return false;

}

long long find(const T & elem) { long long ind = 0;

for (list\_iterator it1 = container.begin(); it1 != container.end(); ++it1) {

for (iterator it2 = it1->begin(); it2 != it1->end();

++it2) {

if (\*it2 == elem) { return ind;

}

}

++ind;

}

return -1;

}

bool insert(const T & parent, const T & elem) {

for (list\_iterator it1 = container.begin(); it1 != container.end(); ++it1) {

for (iterator it2 = it1->begin(); it2 != it1->end();

++it2) {

if (\*it2 == parent) {

it1->insert(++it2, elem);

++container\_size; return true;

}

}

}

return false;

}

void insert(const T & elem) { std::list<T> new\_list; new\_list.push\_back(elem);

++container\_size; container.push\_back(new\_list);

}

size\_t size() {

return container\_size;

}

template<class U>

friend std::ostream & operator << (std::ostream & of, const topology\_t<U> & top) {

for (auto it1 = top.container.begin(); it1 != top.container.end(); ++it1) {

of << "{";

for (auto it2 = it1->begin(); it2 != it1->end(); +

+it2) {

of << \*it2 << " ";

}

of << "}" << std::endl;

}

return of;

}

};

#endif /\* TOPOLOGY\_HPP \*/

control.cpp

#include <unistd.h> #include <vector>

#include "topology.hpp" #include "zmq\_std.hpp"

using node\_id\_type = long long; int main() {

int rc;

topology\_t<node\_id\_type> control\_node; std::vector< std::pair<void\*, void\*> > childs;

std::string s; node\_id\_type id;

while (std::cin >> s >> id) { if (s == "create") {

node\_id\_type parent\_id; std::cin >> parent\_id; if (parent\_id == -1) {

void\* new\_context = NULL; void\* new\_socket = NULL;

zmq\_std::init\_pair\_socket(new\_context,

new\_socket);

rc = zmq\_bind(new\_socket, ("tcp://\*:" +

std::to\_string(PORT\_BASE + id)).c\_str());

assert(rc == 0);

int fork\_id = fork(); if (fork\_id == 0) {

rc = execl(NODE\_EXECUTABLE\_NAME, NODE\_EXECUTABLE\_NAME, std::to\_string(id).c\_str(), NULL);

assert(rc != -1);

return 0;

} else {

bool ok = true;

node\_token\_t reply\_info({fail, id, id}); ok = zmq\_std::recieve\_msg\_wait(reply\_info,

new\_socket);

node\_token\_t\* token = new node\_token\_t({ping, id, id});

node\_token\_t reply({fail, id, id});

ok = zmq\_std::send\_recieve\_wait(token,

reply, new\_socket);

if (ok and reply.action == success) {

childs.push\_back(std::make\_pair(new\_context, new\_socket));

control\_node.insert(id);

<< std::endl;

std::cout << "OK: " << reply\_info.id

} else {

rc = zmq\_close(new\_socket); assert(rc == 0);

rc = zmq\_ctx\_term(new\_context); assert(rc == 0);

}

}

std::endl;

} else if (control\_node.find(parent\_id) == -1) { std::cout << "Error: Not found" << std::endl;

} else {

if (control\_node.find(id) != -1) {

std::cout << "Error: Already exists" <<

} else {

int ind = control\_node.find(parent\_id); node\_token\_t\* token = new

node\_token\_t({create, parent\_id, id});

node\_token\_t reply({fail, id, id});

if (zmq\_std::send\_recieve\_wait(token, reply, childs[ind].second) and reply.action == success) {

std::cout << "OK: " << reply.id <<

std::endl;

control\_node.insert(parent\_id, id);

} else {

std::cout << "Error: Parent is

unavailable" << std::endl;

}

}

}

} else if (s == "remove") {

int ind = control\_node.find(id); if (ind != -1) {

node\_token\_t\* token = new node\_token\_t({destroy, id, id});

node\_token\_t reply({fail, id, id});

bool ok = zmq\_std::send\_recieve\_wait(token, reply, childs[ind].second);

if (reply.action == destroy and reply.parent\_id

== id) {

rc = zmq\_close(childs[ind].second); assert(rc == 0);

rc = zmq\_ctx\_term(childs[ind].first); assert(rc == 0);

std::vector< std::pair<void\*, void\*>

>::iterator it = childs.begin();

while (ind--) {

++it;

}

childs.erase(it);

} else if (reply.action == bind and reply.parent\_id == id) {

rc = zmq\_close(childs[ind].second); assert(rc == 0);

rc = zmq\_ctx\_term(childs[ind].first); assert(rc == 0);

zmq\_std::init\_pair\_socket(childs[ind].first, childs[ind].second);

rc = zmq\_bind(childs[ind].second, ("tcp://

\*:" + std::to\_string(PORT\_BASE + reply.id)).c\_str());

assert(rc == 0);

<< std::endl;

}

if (ok) {

control\_node.erase(id); std::cout << "OK" << std::endl;

} else {

std::cout << "Error: Node is unavailable"

}

} else {

std::cout << "Error: Not found" << std::endl;

id, id});

}

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

int ind = control\_node.find(id); if (ind != -1) {

node\_token\_t\* token = new node\_token\_t({ping,

node\_token\_t reply({fail, id, id});

if (zmq\_std::send\_recieve\_wait(token, reply,

childs[ind].second) and reply.action == success) {

std::cout << "OK: 1" << std::endl;

} else {

std::cout << "OK: 0" << std::endl;

}

} else {

std::cout << "Error: Not found" << std::endl;

id, id});

}

} else if (s == "back") {

int ind = control\_node.find(id); if (ind != -1) {

node\_token\_t\* token = new node\_token\_t({back,

node\_token\_t reply({fail, id, id});

if (zmq\_std::send\_recieve\_wait(token, reply,

childs[ind].second)) {

if (reply.action == success) { node\_token\_t\* token\_back = new

node\_token\_t({back, id, id}); id});

node\_token\_t reply\_back({fail, id,

std::vector<unsigned int> calculated; while

(zmq\_std::send\_recieve\_wait(token\_back, reply\_back, childs[ind].second) and reply\_back.action == success) {

calculated.push\_back(reply\_back.id);

}

if (calculated.empty()) {

std::cout << "OK: " << reply.id

<< " : -1" << std::endl;

<< " : ";

calculated.size() - 1; ++i) {

<< ", ";

<< std::endl;

} else {

std::cout << "OK: " << reply.id for (size\_t i = 0; i < std::cout << calculated[i]

}

std::cout << calculated.back()

to back" << std::endl;

}

} else {

std::cout << "Error: No calculations

<< std::endl;

}

} else {

std::cout << "Error: Node is unavailable"

}

} else {

std::cout << "Error: Not found" << std::endl;

}

} else if (s == "exec") { std::string pattern, text; std::cin >> pattern >> text; int ind = control\_node.find(id); if (ind != -1) {

bool ok = true;

std::string text\_pattern = pattern + SENTINEL +

text + SENTINEL;

+i) {

for (size\_t i = 0; i < text\_pattern.size(); + node\_token\_t\* token = new

node\_token\_t({exec, text\_pattern[i], id});

node\_token\_t reply({fail, id, id});

if (!zmq\_std::send\_recieve\_wait(token, reply, childs[ind].second) or reply.action != success) {

ok = false; break;

}

}

<< std::endl;

if (ok) {

std::cout << "OK" << std::endl;

} else {

std::cout << "Error: Node is unavailable"

}

} else {

std::cout << "Error: Not found" << std::endl;

}

}

}

for (size\_t i = 0; i < childs.size(); ++i) { rc = zmq\_close(childs[i].second); assert(rc == 0);

rc = zmq\_ctx\_term(childs[i].first); assert(rc == 0);

}

}

calculation\_node.cpp #include <list> #include <pthread.h> #include <queue> #include <tuple> #include <unistd.h>

#include "search.hpp" #include "zmq\_std.hpp"

const std::string SENTINEL\_STR = "$"; long long node\_id;

pthread\_mutex\_t mutex; pthread\_cond\_t cond;

std::queue< std::pair<std::string, std::string> > calc\_queue; std::queue< std::list<unsigned int> > done\_queue;

void\* thread\_func(void\*) {

while (1) {

pthread\_mutex\_lock(&mutex); while (calc\_queue.empty()) {

pthread\_cond\_wait(&cond, &mutex);

}

std::pair<std::string, std::string> cur = calc\_queue.front();

calc\_queue.pop(); pthread\_mutex\_unlock(&mutex);

if (cur.first == SENTINEL\_STR and cur.second == SENTINEL\_STR) {

break;

} else {

std::vector<unsigned int> res = KMPStrong(cur.first, cur.second);

std::list<unsigned int> res\_list;

for (const unsigned int & elem : res) { res\_list.push\_back(elem);

}

pthread\_mutex\_lock(&mutex); done\_queue.push(res\_list); pthread\_mutex\_unlock(&mutex);

}

}

return NULL;

}

int main(int argc, char\*\* argv) { int rc;

assert(argc == 2);

node\_id = std::stoll(std::string(argv[1]));

void\* node\_parent\_context = zmq\_ctx\_new();

void\* node\_parent\_socket = zmq\_socket(node\_parent\_context, ZMQ\_PAIR);

rc = zmq\_connect(node\_parent\_socket, ("tcp://localhost:" + std::to\_string(PORT\_BASE + node\_id)).c\_str());

assert(rc == 0);

long long child\_id = -1; void\* node\_context = NULL; void\* node\_socket = NULL;

pthread\_t calculation\_thread;

rc = pthread\_mutex\_init(&mutex, NULL); assert(rc == 0);

rc = pthread\_cond\_init(&cond, NULL); assert(rc == 0);

rc = pthread\_create(&calculation\_thread, NULL, thread\_func, NULL);

assert(rc == 0); std::string pattern, text; bool flag\_sentinel = true;

node\_token\_t\* info\_token = new node\_token\_t({info, getpid(), getpid()});

zmq\_std::send\_msg\_dontwait(info\_token, node\_parent\_socket); std::list<unsigned int> cur\_calculated;

bool has\_child = false; bool awake = true; bool calc = true; while (awake) {

node\_token\_t token;

zmq\_std::recieve\_msg(token, node\_parent\_socket); node\_token\_t\* reply = new node\_token\_t({fail, node\_id,

node\_id});

if (token.action == back) {

if (token.id == node\_id) { if (calc) {

pthread\_mutex\_lock(&mutex); if (done\_queue.empty()) {

reply->action = exec;

} else {

cur\_calculated = done\_queue.front(); done\_queue.pop();

reply->action = success; reply->id = getpid();

}

pthread\_mutex\_unlock(&mutex); calc = false;

} else {

if (cur\_calculated.size() > 0) { reply->action = success;

reply->id = cur\_calculated.front(); cur\_calculated.pop\_front();

} else {

reply->action = exec; calc = true;

}

}

} else {

node\_token\_t\* token\_down = new node\_token\_t(token);

node\_token\_t reply\_down(token); reply\_down.action = fail;

if (zmq\_std::send\_recieve\_wait(token\_down, reply\_down, node\_socket) and reply\_down.action == success) {

\*reply = reply\_down;

}

}

} else if (token.action == bind and token.parent\_id == node\_id) {

node\_socket);

/\*

* Bind could be recieved when parent created node
* and this node should bind to parent's child

\*/ zmq\_std::init\_pair\_socket(node\_context,

rc = zmq\_bind(node\_socket, ("tcp://\*:" +

std::to\_string(PORT\_BASE + token.id)).c\_str());

assert(rc == 0); has\_child = true; child\_id = token.id;

node\_token\_t\* token\_ping = new node\_token\_t({ping, child\_id, child\_id});

node\_token\_t reply\_ping({fail, child\_id, child\_id}); if (zmq\_std::send\_recieve\_wait(token\_ping,

reply\_ping, node\_socket) and reply\_ping.action == success) {

reply->action = success;

}

} else if (token.action == create) { if (token.parent\_id == node\_id) {

if (has\_child) {

rc = zmq\_close(node\_socket); assert(rc == 0);

rc = zmq\_ctx\_term(node\_context); assert(rc == 0);

node\_socket);

}

zmq\_std::init\_pair\_socket(node\_context,

rc = zmq\_bind(node\_socket, ("tcp://\*:" +

std::to\_string(PORT\_BASE + token.id)).c\_str());

assert(rc == 0);

int fork\_id = fork(); if (fork\_id == 0) {

rc = execl(NODE\_EXECUTABLE\_NAME, NODE\_EXECUTABLE\_NAME, std::to\_string(token.id).c\_str(), NULL);

assert(rc != -1);

return 0;

} else {

bool ok = true;

node\_token\_t reply\_info({fail, token.id,

token.id}); node\_socket);

reply\_info.parent\_id;

ok = zmq\_std::recieve\_msg\_wait(reply\_info, if (reply\_info.action != fail) {

reply->id = reply\_info.id;

reply->parent\_id =

}

if (has\_child) {

node\_token\_t\* token\_bind = new node\_token\_t({bind, token.id, child\_id});

node\_token\_t reply\_bind({fail,

token.id, token.id});

ok =

zmq\_std::send\_recieve\_wait(token\_bind, reply\_bind, node\_socket);

ok = ok and (reply\_bind.action ==

success);

}

if (ok) {

/\* We should check if child has

connected to this node \*/

node\_token\_t\* token\_ping = new

node\_token\_t({ping, token.id, token.id});

node\_token\_t reply\_ping({fail,

token.id, token.id});

ok =

zmq\_std::send\_recieve\_wait(token\_ping, reply\_ping, node\_socket);

ok = ok and (reply\_ping.action ==

success);

if (ok) {

reply->action = success; child\_id = token.id; has\_child = true;

} else {

rc = zmq\_close(node\_socket); assert(rc == 0);

rc = zmq\_ctx\_term(node\_context); assert(rc == 0);

}

}

}

} else if (has\_child) {

node\_token\_t\* token\_down = new node\_token\_t(token);

node\_token\_t reply\_down(token); reply\_down.action = fail;

if (zmq\_std::send\_recieve\_wait(token\_down, reply\_down, node\_socket) and reply\_down.action == success) {

\*reply = reply\_down;

}

}

} else if (token.action == ping) { if (token.id == node\_id) {

reply->action = success;

} else if (has\_child) {

node\_token\_t\* token\_down = new node\_token\_t(token);

node\_token\_t reply\_down(token); reply\_down.action = fail;

if (zmq\_std::send\_recieve\_wait(token\_down, reply\_down, node\_socket) and reply\_down.action == success) {

\*reply = reply\_down;

}

}

} else if (token.action == destroy) { if (has\_child) {

if (token.id == child\_id) { bool ok = true;

node\_token\_t\* token\_down = new node\_token\_t({destroy, node\_id, child\_id});

node\_token\_t reply\_down({fail, child\_id,

child\_id});

ok =

zmq\_std::send\_recieve\_wait(token\_down, reply\_down, node\_socket);

/\* We should get special reply from child

\*/

if (reply\_down.action == destroy and

reply\_down.parent\_id == child\_id) {

rc = zmq\_close(node\_socket); assert(rc == 0);

rc = zmq\_ctx\_term(node\_context); assert(rc == 0);

has\_child = false; child\_id = -1;

} else if (reply\_down.action == bind and reply\_down.parent\_id == node\_id) {

rc = zmq\_close(node\_socket); assert(rc == 0);

rc = zmq\_ctx\_term(node\_context); assert(rc == 0);

zmq\_std::init\_pair\_socket(node\_context, node\_socket);

rc = zmq\_bind(node\_socket, ("tcp://\*:" + std::to\_string(PORT\_BASE + reply\_down.id)).c\_str());

assert(rc == 0);

child\_id = reply\_down.id; node\_token\_t\* token\_ping = new

node\_token\_t({ping, child\_id, child\_id});

node\_token\_t reply\_ping({fail,

child\_id, child\_id});

if

(zmq\_std::send\_recieve\_wait(token\_ping, reply\_ping, node\_socket) and reply\_ping.action == success) {

ok = true;

}

}

if (ok) {

reply->action = success;

}

} else if (token.id == node\_id) { rc = zmq\_close(node\_socket); assert(rc == 0);

rc = zmq\_ctx\_term(node\_context); assert(rc == 0);

has\_child = false; reply->action = bind; reply->id = child\_id;

reply->parent\_id = token.parent\_id; awake = false;

} else {

node\_token\_t\* token\_down = new

node\_token\_t(token);

node\_token\_t reply\_down(token); reply\_down.action = fail;

if (zmq\_std::send\_recieve\_wait(token\_down,

reply\_down, node\_socket) and reply\_down.action == success) {

\*reply = reply\_down;

}

}

} else if (token.id == node\_id) {

/\* Special message to parent \*/ reply->action = destroy;

reply->parent\_id = node\_id; reply->id = node\_id;

awake = false;

}

} else if (token.action == exec) { if (token.id == node\_id) {

char c = token.parent\_id; if (c == SENTINEL) {

if (flag\_sentinel) { std::swap(text, pattern);

} else {

pthread\_mutex\_lock(&mutex); if (calc\_queue.empty()) {

pthread\_cond\_signal(&cond);

}

calc\_queue.push({pattern, text}); pthread\_mutex\_unlock(&mutex); text.clear();

pattern.clear();

}

flag\_sentinel = flag\_sentinel ^ 1;

} else {

text = text + c;

}

reply->action = success;

} else if (has\_child) {

node\_token\_t\* token\_down = new node\_token\_t(token);

node\_token\_t reply\_down(token); reply\_down.action = fail;

if (zmq\_std::send\_recieve\_wait(token\_down, reply\_down, node\_socket) and reply\_down.action == success) {

\*reply = reply\_down;

}

}

}

zmq\_std::send\_msg\_dontwait(reply, node\_parent\_socket);

}

if (has\_child) {

rc = zmq\_close(node\_socket); assert(rc == 0);

rc = zmq\_ctx\_term(node\_context); assert(rc == 0);

}

rc = zmq\_close(node\_parent\_socket); assert(rc == 0);

rc = zmq\_ctx\_term(node\_parent\_context); assert(rc == 0); pthread\_mutex\_lock(&mutex);

if (calc\_queue.empty()) { pthread\_cond\_signal(&cond);

}

calc\_queue.push({SENTINEL\_STR, SENTINEL\_STR}); pthread\_mutex\_unlock(&mutex);

rc = pthread\_join(calculation\_thread, NULL); assert(rc == 0);

rc = pthread\_cond\_destroy(&cond); assert(rc == 0);

rc = pthread\_mutex\_destroy(&mutex); assert(rc == 0);

}

search.hpp

#ifndef SEARCH\_HPP #define SEARCH\_HPP

#include <string> #include <vector>

std::vector<unsigned int> PrefixFunction(const std::string & s); std::vector<unsigned int> KMPWeak(const std::string & pattern, const std::string & text);

std::vector<unsigned int> ZFunction(const std::string & s); std::vector<unsigned int> StrongPrefixFunction(const std::string & s);

std::vector<unsigned int> KMPStrong(const std::string & pattern, const std::string & text);

#endif /\* SEARCH\_HPP \*/

search.cpp

#include "search.hpp"

std::vector<unsigned int> PrefixFunction(const std::string & s) { unsigned int n = s.size();

std::vector<unsigned int> p(n);

for (unsigned int i = 1; i < n; ++i) { p[i] = p[i - 1];

while (p[i] > 0 and s[i] != s[p[i]]) { p[i] = p[p[i] - 1];

}

if (s[i] == s[p[i]]) {

++p[i];

}

}

return p;

}

std::vector<unsigned int> KMPWeak(const std::string & pattern, const std::string & text) {

std::vector<unsigned int> p = PrefixFunction(pattern); unsigned int m = pattern.size();

unsigned int n = text.size(); unsigned int i = 0; std::vector<unsigned int> ans; if (m > n) {

return ans;

}

while (i < n - m + 1) { unsigned int j = 0;

while (j < m and pattern[j] == text[i + j]) {

++j;

}

if (j == m) {

ans.push\_back(i);

} else {

if (j > 0 and j > p[j - 1]) {

i = i + j - p[j - 1] - 1;

}

}

++i;

}

return ans;

}

std::vector<unsigned int> ZFunction(const std::string & s) { unsigned int n = s.size();

std::vector<unsigned int> z(n); unsigned int l = 0, r = 0;

for (unsigned int i = 1; i < n; ++i) { if (i <= r) {

z[i] = std::min(z[i - l], r - i);

}

while (i + z[i] < n and s[i + z[i]] == s[z[i]]) {

++z[i];

}

if (i + z[i] > r) { l = i;

r = i + z[i];

}

}

return z;

}

std::vector<unsigned int> StrongPrefixFunction(const std::string & s) {

std::vector<unsigned int> z = ZFunction(s); unsigned int n = s.size(); std::vector<unsigned int> sp(n);

for (unsigned int i = n - 1; i > 0; --i) { sp[i + z[i] - 1] = z[i];

}

return sp;

}

std::vector<unsigned int> KMPStrong(const std::string & pattern, const std::string & text) {

std::vector<unsigned int> p = StrongPrefixFunction(pattern); unsigned int m = pattern.size();

unsigned int n = text.size(); unsigned int i = 0; std::vector<unsigned int> ans; if (m > n) {

return ans;

}

while (i < n - m + 1) { unsigned int j = 0;

while (j < m and pattern[j] == text[i + j]) {

++j;

}

if (j == m) {

ans.push\_back(i);

} else {

if (j > 0 and j > p[j - 1]) {

i = i + j - p[j - 1] - 1;

}

}

++i;

}

return ans;

}

**strace**

strace ./control 3

execve("./control", ["./control", "3"], 0x7ffc1749a378 /\* 45 vars \*/) = 0

brk(NULL) = 0x561137ebc000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7fff31a736f0) = -1 EINVAL (Недопустимый аргумент)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f4a2f24d000

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (Нет такого файла или каталога)

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

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=70735, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 70735, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f4a2f23b000

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\240\233\1\0\0\0\0\0"..., 832) = 832

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

mmap(NULL, 636784, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4a2f19f000

mmap(0x7f4a2f1b7000, 397312, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x18000) = 0x7f4a2f1b7000

mmap(0x7f4a2f218000, 106496, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x79000) = 0x7f4a2f218000

mmap(0x7f4a2f232000, 36864, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x92000) = 0x7f4a2f232000

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\0"..., 832) = 832

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

mmap(NULL, 2275520, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4a2ee00000

mprotect(0x7f4a2ee9a000, 1576960, PROT\_NONE) = 0

mmap(0x7f4a2ee9a000, 1118208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x9a000) = 0x7f4a2ee9a000

mmap(0x7f4a2efab000, 454656, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ab000) = 0x7f4a2efab000

mmap(0x7f4a2f01b000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x21a000) = 0x7f4a2f01b000

mmap(0x7f4a2f029000, 10432, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f4a2f029000

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\0"..., 832) = 832

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

mmap(NULL, 127720, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4a2f17f000

mmap(0x7f4a2f182000, 94208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7f4a2f182000

mmap(0x7f4a2f199000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1a000) = 0x7f4a2f199000

mmap(0x7f4a2f19d000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1d000) = 0x7f4a2f19d000

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\0P\237\2\0\0\0\0\0"..., 832) = 832

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

pread64(3, "\4\0\0\0 \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

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

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2216304, ...}, AT\_EMPTY\_PATH) = 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, 2260560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4a2ea00000

mmap(0x7f4a2ea28000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7f4a2ea28000

mmap(0x7f4a2ebbd000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7f4a2ebbd000

mmap(0x7f4a2ec15000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x214000) = 0x7f4a2ec15000

mmap(0x7f4a2ec1b000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f4a2ec1b000

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\0"..., 832) = 832

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

mmap(NULL, 94432, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4a2f167000

mprotect(0x7f4a2f16b000, 69632, PROT\_NONE) = 0

mmap(0x7f4a2f16b000, 53248, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000) = 0x7f4a2f16b000

mmap(0x7f4a2f178000, 12288, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x11000) = 0x7f4a2f178000

mmap(0x7f4a2f17c000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x14000) = 0x7f4a2f17c000

mmap(0x7f4a2f17e000, 224, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f4a2f17e000

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\0"..., 832) = 832

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

mmap(NULL, 357440, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4a2f10f000

mprotect(0x7f4a2f11b000, 303104, PROT\_NONE) = 0

mmap(0x7f4a2f11b000, 229376, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xc000) = 0x7f4a2f11b000

mmap(0x7f4a2f153000, 69632, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x44000) = 0x7f4a2f153000

mmap(0x7f4a2f165000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x55000) = 0x7f4a2f165000

close(3) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f4a2f10d000

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\340L\0\0\0\0\0\0"..., 832) = 832

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

mmap(NULL, 329808, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4a2f0bc000

mmap(0x7f4a2f0c0000, 172032, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000) = 0x7f4a2f0c0000

mmap(0x7f4a2f0ea000, 118784, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2e000) = 0x7f4a2f0ea000

mmap(0x7f4a2f107000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4a000) = 0x7f4a2f107000

mmap(0x7f4a2f109000, 14416, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f4a2f109000

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 \255\0\0\0\0\0\0"..., 832) = 832

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

mmap(NULL, 1223168, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4a2ecd5000

mprotect(0x7f4a2ecdf000, 446464, PROT\_NONE) = 0

mmap(0x7f4a2ecdf000, 286720, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xa000) = 0x7f4a2ecdf000

mmap(0x7f4a2ed25000, 155648, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x50000) = 0x7f4a2ed25000

mmap(0x7f4a2ed4c000, 16384, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x76000) = 0x7f4a2ed4c000

mmap(0x7f4a2ed50000, 719360, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f4a2ed50000

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\0"..., 832) = 832

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

mmap(NULL, 340960, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4a2f068000

mprotect(0x7f4a2f073000, 282624, PROT\_NONE) = 0

mmap(0x7f4a2f073000, 229376, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xb000) = 0x7f4a2f073000

mmap(0x7f4a2f0ab000, 49152, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x43000) = 0x7f4a2f0ab000

mmap(0x7f4a2f0b8000, 16384, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4f000) = 0x7f4a2f0b8000

close(3) = 0

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

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

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

mmap(NULL, 942344, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4a2e919000

mmap(0x7f4a2e927000, 507904, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe000) = 0x7f4a2e927000

mmap(0x7f4a2e9a3000, 372736, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x8a000) = 0x7f4a2e9a3000

mmap(0x7f4a2e9fe000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe4000) = 0x7f4a2e9fe000

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

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

mmap(NULL, 49384, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4a2f05b000

mmap(0x7f4a2f05d000, 28672, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2000) = 0x7f4a2f05d000

mmap(0x7f4a2f064000, 8192, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x9000) = 0x7f4a2f064000

mmap(0x7f4a2f066000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xa000) = 0x7f4a2f066000

close(3) = 0

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

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

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

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f4a2f059000

mmap(NULL, 16424, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4a2f054000

mmap(0x7f4a2f055000, 4096, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1000) = 0x7f4a2f055000

mmap(0x7f4a2f056000, 4096, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2000) = 0x7f4a2f056000

mmap(0x7f4a2f057000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2000) = 0x7f4a2f057000

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

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

mmap(NULL, 830576, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4a2e84e000

mprotect(0x7f4a2e86f000, 634880, PROT\_NONE) = 0

mmap(0x7f4a2e86f000, 380928, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x21000) = 0x7f4a2e86f000

mmap(0x7f4a2e8cc000, 249856, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x7e000) = 0x7f4a2e8cc000

mmap(0x7f4a2e90a000, 61440, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xbb000) = 0x7f4a2e90a000

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\0"..., 832) = 832

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

mmap(NULL, 188472, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4a2eca6000

mprotect(0x7f4a2ecaa000, 163840, PROT\_NONE) = 0

mmap(0x7f4a2ecaa000, 110592, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000) = 0x7f4a2ecaa000

mmap(0x7f4a2ecc5000, 49152, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1f000) = 0x7f4a2ecc5000

mmap(0x7f4a2ecd2000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2b000) = 0x7f4a2ecd2000

mmap(0x7f4a2ecd4000, 56, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f4a2ecd4000

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\0"..., 832) = 832

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

mmap(NULL, 20552, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4a2f04e000

mmap(0x7f4a2f050000, 4096, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2000) = 0x7f4a2f050000

mmap(0x7f4a2f051000, 4096, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7f4a2f051000

mmap(0x7f4a2f052000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7f4a2f052000

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\0"..., 832) = 832

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

mmap(NULL, 54224, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4a2f040000

mprotect(0x7f4a2f043000, 36864, PROT\_NONE) = 0

mmap(0x7f4a2f043000, 24576, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7f4a2f043000

mmap(0x7f4a2f049000, 8192, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x9000) = 0x7f4a2f049000

mmap(0x7f4a2f04c000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xb000) = 0x7f4a2f04c000

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\0"..., 832) = 832

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

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f4a2f03e000

mmap(NULL, 24592, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4a2f037000

mmap(0x7f4a2f039000, 8192, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2000) = 0x7f4a2f039000

mmap(0x7f4a2f03b000, 4096, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000) = 0x7f4a2f03b000

mmap(0x7f4a2f03c000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000) = 0x7f4a2f03c000

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\0"..., 832) = 832

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

mmap(NULL, 80456, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4a2ec92000

mmap(0x7f4a2ec95000, 40960, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7f4a2ec95000

mmap(0x7f4a2ec9f000, 12288, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xd000) = 0x7f4a2ec9f000

mmap(0x7f4a2eca2000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xf000) = 0x7f4a2eca2000

mmap(0x7f4a2eca4000, 6728, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f4a2eca4000

close(3) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f4a2f035000

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f4a2f032000

arch\_prctl(ARCH\_SET\_FS, 0x7f4a2f0329c0) = 0

set\_tid\_address(0x7f4a2f032c90) = 6262

set\_robust\_list(0x7f4a2f032ca0, 24) = 0

rseq(0x7f4a2f033360, 0x20, 0, 0x53053053) = 0

mprotect(0x7f4a2ec15000, 16384, PROT\_READ) = 0

mprotect(0x7f4a2eca2000, 4096, PROT\_READ) = 0

mprotect(0x7f4a2f03c000, 4096, PROT\_READ) = 0

mprotect(0x7f4a2f04c000, 4096, PROT\_READ) = 0

mprotect(0x7f4a2f052000, 4096, PROT\_READ) = 0

mprotect(0x7f4a2ecd2000, 4096, PROT\_READ) = 0

mprotect(0x7f4a2e90a000, 53248, PROT\_READ) = 0

mprotect(0x7f4a2f057000, 4096, PROT\_READ) = 0

mprotect(0x7f4a2f066000, 4096, PROT\_READ) = 0

mprotect(0x7f4a2e9fe000, 4096, PROT\_READ) = 0

mprotect(0x7f4a2f0b8000, 8192, PROT\_READ) = 0

mprotect(0x7f4a2f19d000, 4096, PROT\_READ) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f4a2f030000

mprotect(0x7f4a2f01b000, 45056, PROT\_READ) = 0

mprotect(0x7f4a2ed4c000, 12288, PROT\_READ) = 0

mprotect(0x7f4a2f107000, 4096, PROT\_READ) = 0

mprotect(0x7f4a2f165000, 4096, PROT\_READ) = 0

mprotect(0x7f4a2f17c000, 4096, PROT\_READ) = 0

mprotect(0x7f4a2f232000, 32768, PROT\_READ) = 0

mprotect(0x561136520000, 4096, PROT\_READ) = 0

mprotect(0x7f4a2f287000, 8192, PROT\_READ) = 0

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

munmap(0x7f4a2f23b000, 70735) = 0

getrandom("\xcc\x27\xa9\xb7\x3a\xf4\x2c\xb2", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x561137ebc000

brk(0x561137edd000) = 0x561137edd000

gettimeofday({tv\_sec=1703532414, tv\_usec=823545}, {tz\_minuteswest=0, tz\_dsttime=0}) = 0

futex(0x7f4a2f02977c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

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

read(0, create 1 -1

create 2 1

create 3 2

create 4 3

create 5 4

ping 1

ping 2

ping 3

ping 4

ping 5

remove 5

remove 4

remove 3

remove 2

remove 1

"create 1 -1\n", 1024) = 12

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

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

close(3) = 0

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

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

getdents64(3, 0x561137ece2f0 /\* 29 entries \*/, 32768) = 832

getdents64(3, 0x561137ece2f0 /\* 0 entries \*/, 32768) = 0

close(3) = 0

getpid() = 6262

sched\_getaffinity(6262, 128, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]) = 8

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

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

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

read(3, "", 4096) = 0

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

close(3) = 0

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

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=70735, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 70735, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f4a2f23b000

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/glibc-hwcaps/x86-64-v3/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/glibc-hwcaps/x86-64-v3", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/glibc-hwcaps/x86-64-v2/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/glibc-hwcaps/x86-64-v2", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/tls/x86\_64/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/tls/x86\_64/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/tls/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/tls/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/tls/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/tls/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/tls/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/tls", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/x86\_64/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/x86\_64/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/lib/x86\_64-linux-gnu", {st\_mode=S\_IFDIR|0755, st\_size=102400, ...}, 0) = 0

openat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/glibc-hwcaps/x86-64-v3/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/glibc-hwcaps/x86-64-v3", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/glibc-hwcaps/x86-64-v2/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/glibc-hwcaps/x86-64-v2", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/tls/x86\_64/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/tls/x86\_64/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/tls/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/tls/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/tls/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/tls/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/tls/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/tls", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/x86\_64/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/x86\_64/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu", {st\_mode=S\_IFDIR|0755, st\_size=102400, ...}, 0) = 0

openat(AT\_FDCWD, "/lib/glibc-hwcaps/x86-64-v3/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/lib/glibc-hwcaps/x86-64-v3", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/lib/glibc-hwcaps/x86-64-v2/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/lib/glibc-hwcaps/x86-64-v2", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/lib/tls/x86\_64/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/lib/tls/x86\_64/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/lib/tls/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/lib/tls/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/lib/tls/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/lib/tls/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/lib/tls/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/lib/tls", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/lib/x86\_64/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/lib/x86\_64/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/lib/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/lib/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/lib/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/lib/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/lib/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

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

openat(AT\_FDCWD, "/usr/lib/glibc-hwcaps/x86-64-v3/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/usr/lib/glibc-hwcaps/x86-64-v3", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/glibc-hwcaps/x86-64-v2/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/usr/lib/glibc-hwcaps/x86-64-v2", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/tls/x86\_64/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/usr/lib/tls/x86\_64/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/tls/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/usr/lib/tls/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/tls/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/usr/lib/tls/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/tls/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/usr/lib/tls", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/x86\_64/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/usr/lib/x86\_64/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/usr/lib/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/x86\_64/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

newfstatat(AT\_FDCWD, "/usr/lib/x86\_64", 0x7fff31a70680, 0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/libnss\_db.so.2", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

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

munmap(0x7f4a2f23b000, 70735) = 0

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

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=70735, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 70735, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f4a2f23b000

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libnss\_db-2.35.so", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/libnss\_db-2.35.so", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/lib/libnss\_db-2.35.so", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/usr/lib/libnss\_db-2.35.so", O\_RDONLY|O\_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

munmap(0x7f4a2f23b000, 70735) = 0

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

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

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

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

read(3, "", 4096) = 0

close(3) = 0

gettimeofday({tv\_sec=1703532416, tv\_usec=589599}, NULL) = 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() = 6262

getpid() = 6262

getrandom("\x5c\x11\x2a\x63\x6c\x6a\x27\xeb\xcc\x29\xad\x14\xfa\x12\x0a\xb4", 16, 0) = 16

getrandom("\xfb\x63\x09\x3d\x92\x61\x59\x07\x5b\x83\xcc\xdc\x2f\x80\x8c\xc0", 16, 0) = 16

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() = 6262

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=562575616}) = 0

epoll\_create1(EPOLL\_CLOEXEC) = 5

epoll\_ctl(5, EPOLL\_CTL\_ADD, 4, {events=0, data={u32=938274592, u64=94631952707360}}) = 0

epoll\_ctl(5, EPOLL\_CTL\_MOD, 4, {events=EPOLLIN, data={u32=938274592, u64=94631952707360}}) = 0

getpid() = 6262

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7f4a2ea91870, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_ONSTACK|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7f4a2ea42520}, 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) = 0x7f4a2e04d000

mprotect(0x7f4a2e04e000, 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=0x7f4a2e84d910, parent\_tid=0x7f4a2e84d910, exit\_signal=0, stack=0x7f4a2e04d000, stack\_size=0x7ffc80, tls=0x7f4a2e84d640} => {parent\_tid=[6263]}, 88) = 6263

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() = 6262

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=564621700}) = 0

epoll\_create1(EPOLL\_CLOEXEC) = 7

epoll\_ctl(7, EPOLL\_CTL\_ADD, 6, {events=0, data={u32=938273120, u64=94631952705888}}) = 0

epoll\_ctl(7, EPOLL\_CTL\_MOD, 6, {events=EPOLLIN, data={u32=938273120, u64=94631952705888}}) = 0

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f4a2d84c000

mprotect(0x7f4a2d84d000, 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=0x7f4a2e04c910, parent\_tid=0x7f4a2e04c910, exit\_signal=0, stack=0x7f4a2d84c000, stack\_size=0x7ffc80, tls=0x7f4a2e04c640} => {parent\_tid=[6264]}, 88) = 6264

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

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=565475626}) = 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() = 6262

getpid() = 6262

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

socket(AF\_INET, SOCK\_STREAM|SOCK\_CLOEXEC, IPPROTO\_TCP) = 9

setsockopt(9, SOL\_SOCKET, SO\_REUSEADDR, [1], 4) = 0

bind(9, {sa\_family=AF\_INET, sin\_port=htons(8001), sin\_addr=inet\_addr("0.0.0.0")}, 16) = 0

listen(9, 100) = 0

getsockname(9, {sa\_family=AF\_INET, sin\_port=htons(8001), sin\_addr=inet\_addr("0.0.0.0")}, [128 => 16]) = 0

getsockname(9, {sa\_family=AF\_INET, sin\_port=htons(8001), sin\_addr=inet\_addr("0.0.0.0")}, [128 => 16]) = 0

getpid() = 6262

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

getpid() = 6262

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

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f4a2f032c90) = 6265

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=568396378}) = 0

getpid() = 6262

poll([{fd=8, events=POLLIN}], 1, 1000) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 6262

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

getpid() = 6262

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

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=569046234}) = 0

getpid() = 6262

poll([{fd=8, events=POLLIN}], 1, 999) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 6262

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

getpid() = 6262

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

getpid() = 6262

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

getpid() = 6262

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

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=574926639}) = 0

getpid() = 6262

poll([{fd=8, events=POLLIN}], 1, 1000) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 6262

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

getpid() = 6262

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

newfstatat(1, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}, AT\_EMPTY\_PATH) = 0

write(1, "OK: 6265\n", 9OK: 6265

) = 9

read(0, "create 2 1\n", 1024) = 11

getpid() = 6262

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

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=575655273}) = 0

getpid() = 6262

poll([{fd=8, events=POLLIN}], 1, 1000) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 6262

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

getpid() = 6262

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

write(1, "OK: 6271\n", 9OK: 6271

) = 9

read(0, "create 3 2\n", 1024) = 11

getpid() = 6262

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

getpid() = 6262

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

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=582926739}) = 0

getpid() = 6262

poll([{fd=8, events=POLLIN}], 1, 1000) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 6262

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

getpid() = 6262

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

write(1, "OK: 6277\n", 9OK: 6277

) = 9

read(0, "create 4 3\n", 1024) = 11

getpid() = 6262

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

getpid() = 6262

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

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=589382900}) = 0

getpid() = 6262

poll([{fd=8, events=POLLIN}], 1, 1000) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 6262

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

getpid() = 6262

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

write(1, "OK: 6283\n", 9OK: 6283

) = 9

read(0, "create 5 4\n", 1024) = 11

getpid() = 6262

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

getpid() = 6262

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

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=595680176}) = 0

getpid() = 6262

poll([{fd=8, events=POLLIN}], 1, 1000) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 6262

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

getpid() = 6262

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

write(1, "OK: 6289\n", 9OK: 6289

) = 9

read(0, "ping 1\n", 1024) = 7

getpid() = 6262

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

getpid() = 6262

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

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=603361041}) = 0

getpid() = 6262

poll([{fd=8, events=POLLIN}], 1, 1000) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 6262

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

getpid() = 6262

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

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

) = 6

read(0, "ping 2\n", 1024) = 7

getpid() = 6262

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

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=603878830}) = 0

getpid() = 6262

poll([{fd=8, events=POLLIN}], 1, 1000) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 6262

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

getpid() = 6262

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

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

) = 6

read(0, "ping 3\n", 1024) = 7

getpid() = 6262

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

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=604621222}) = 0

getpid() = 6262

poll([{fd=8, events=POLLIN}], 1, 1000) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 6262

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

getpid() = 6262

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

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

) = 6

read(0, "ping 4\n", 1024) = 7

getpid() = 6262

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

getpid() = 6262

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

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=605492119}) = 0

getpid() = 6262

poll([{fd=8, events=POLLIN}], 1, 1000) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 6262

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

getpid() = 6262

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

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

) = 6

read(0, "ping 5\n", 1024) = 7

getpid() = 6262

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

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=606360572}) = 0

getpid() = 6262

poll([{fd=8, events=POLLIN}], 1, 1000) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 6262

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

getpid() = 6262

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

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

) = 6

read(0, "remove 5\n", 1024) = 9

getpid() = 6262

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

getpid() = 6262

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

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=607443292}) = 0

getpid() = 6262

poll([{fd=8, events=POLLIN}], 1, 1000) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 6262

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

getpid() = 6262

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

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

) = 3

read(0, "remove 4\n", 1024) = 9

getpid() = 6262

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

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=608655773}) = 0

getpid() = 6262

poll([{fd=8, events=POLLIN}], 1, 1000) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 6262

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

getpid() = 6262

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

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

) = 3

read(0, "remove 3\n", 1024) = 9

getpid() = 6262

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

getpid() = 6262

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

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=609614877}) = 0

getpid() = 6262

poll([{fd=8, events=POLLIN}], 1, 1000) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 6262

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

getpid() = 6262

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

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

) = 3

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

getpid() = 6262

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

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=610429484}) = 0

getpid() = 6262

poll([{fd=8, events=POLLIN}], 1, 1000) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 6262

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

getpid() = 6262

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

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

) = 3

read(0, "remove 1\n", 1024) = 9

getpid() = 6262

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

getpid() = 6262

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

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=2150, tv\_nsec=611286482}) = 0

getpid() = 6262

poll([{fd=8, events=POLLIN}], 1, 1000) = 1 ([{fd=8, revents=POLLIN}])

getpid() = 6262

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

getpid() = 6262

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

getpid() = 6262

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

getpid() = 6262

getpid() = 6262

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

futex(0x561137ece798, FUTEX\_WAKE\_PRIVATE, 1) = 1

getpid() = 6262

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

getpid() = 6262

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

getpid() = 6262

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

close(7) = 0

close(6) = 0

close(5) = 0

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=6265, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=1} ---

close(4) = 0

close(3) = 0

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

) = 3

read(0, "\n", 1024) = 1

# Выводы

В ходе выполнения лабораторной работы я изучил основы работы с очередями сообщений *ZeroMQ* и реализовал программу с использованием этой библиотеки. Для достижения отказоустойчивости я пробовал разные способы связи, больше всего подошёл *ZMQ\_PAIR*. Самым сложным в работе оказались удаление узла из сети и вставка узла между другими узлами. При таких операциях нужно было переподключать сокеты на вычислительных узлах.

Когда параллельных вычислений становится мало, на помощь приходят распределённые вычисления (распределение вычислений осуществляется уже не между потоками процессора, а между отдельными ЭВМ). Очереди сообщений используются для взаимодействия нескольких машин в одной большой сети. Опыт работы с *ZeroMQ* пригодится мне при настройке собственной системы распределённых вычислений.

# Список литературы

1. zmq\_socket(3) — 0MQ Api — ZeroMQ API

URL: http://api.zeromq.org/2-1:zmq-socket (дата обращения 08.12.2020)

1. zmq\_bind(3) — 0MQ Api — ZeroMQ API

URL: http://api.zeromq.org/2-1:zmq-bind (дата обращения 09.12.2020)

1. zmq\_connect(3) — 0MQ Api — ZeroMQ API

URL: http://api.zeromq.org/2-1:zmq-connect (дата обращения 09.12.2020)

1. Sockets and Patterns | ØMQ — The Guide — ZeroMQ Guide

URL: <https://zguide.zeromq.org/docs/chapter2/> (дата обращения 08.12.2020)

1. Socket API — ZeroMQ

URL: <https://zeromq.org/socket-api/> (дата обращения 08.12.2020)

1. zmq\_setsockopt(3) — 0MQ Api — ZeroMQ API

URL: http://api.zeromq.org/3-2:zmq-setsockopt (дата обращения 09.12.2020)

1. Messages — ZeroMQ

URL: <https://zeromq.org/messages/> (дата обращения 10.12.2020)

1. zmq\_msg\_send(3) — 0MQ Api — ZeroMQ API

URL: http://api.zeromq.org/3-2:zmq-msg-send (дата обращения 10.12.2020)

1. zmq\_msg\_recv(3) — 0MQ Api — ZeroMQ API

URL: http://api.zeromq.org/master:zmq-msg-recv (дата обращения 10.12.2020)