Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №6 по курсу

«Операционные системы»

Группа: М80-207Б-20

Студент: Мерц С.П.

Вариант: 15

Преподаватель: Миронов Е.С.

Оценка: ____________________

Дата: 27.12.21

Москва, 2021.

Содержание

1 Постановка задачи.
2 Общие сведения о программе.
3 Общий метод и алгоритм решения.
4 Код программы.
5 Демонстрация работы программы.
6 Вывод.

Постановка задачи

Реализовать распределенную систему по обработке запросов. В данной 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 сервера сообщений zmq. Также в данной системе необходимо предусмотреть проверку доступности узлов в соответствии с вариантом.

Вариант задания: 15. Топология — список. Тип вычислительной команды — локальный целочисленный словарь. Тип проверки узлов на доступность —heartbeat.

Общие сведения о программе

Программа состоит из двух файлов, которые компилируются в исполнительные файлы(которые представляют управляющие и вычислительные узлы). Общение между процессами происходит с помощью библиотеки zmq.

Общий метод и алгоритм решения

· Управляющий узел принимает команды, обрабатывает их и пересылает дочерним узлам(или выводит сообщение об ошибке).

· Дочерние узлы проверяют, может ли быть команда выполнена в данном узле, если нет, то команда пересылается в дочерний узел, из которого возвращается некоторое сообщение(об успехе или об ошибке), которое потом пересылается обратно по дереву.

· Для корректной проверки на доступность узлов, используется список, имитирующий поведение узлов в данной топологии.

· Если узел недоступен, то по истечении таймаута будет сгенерировано сообщение о недоступности узла и оно будет пере уничтожаются.

Код программы

control_node.cpp:

```
 #include <unistd.h>

#include <vector>
#include <thread>
#include <chrono>
#include <algorithm>
#include <zmq.hpp>
#include "my_zmq.h"
```

```
#include "topology.h"

std::vector<long long> ping_storage(0);
topology_t<long long> control_node;
std::vector<std::pair<void*, void*>> children;// [context, socket]

void pinger(int wait) {
    while (true) {
        for (size_t i = 0; i < ping_storage.size();i++) {
            int value = ping_storage[i];
            int ind = control_node.find(value);
            auto* token = new node_token_t({ ping, value, value });
            node_token_t reply({ fail, value, value });
            if (ind != -1 and my_zmq::send_receive_wait(token, reply, children[ind].second) and reply.action == success) {
                std::cout << "OK" << std::endl;
                continue;
            }
            else {
                std::cout << "Heartbit: node " << value << " is unavailable now" << std::endl;
                auto iterator = std::find(ping_storage.begin(), ping_storage.end(), value);
                if (iterator != ping_storage.end()) {
                    ping_storage.erase(iterator);
                }
                break;
            }
        }
        std::this_thread::sleep_for(std::chrono::milliseconds(wait));
    }
}

void delete_control_node(long long id) {
    int ind = control_node.find(id);
    int rc;
    bool ok;
    if (ind != -1) {
        auto* token = new node_token_t({ destroy, id, id });
        node_token_t reply({ fail, id, id });
        ok = my_zmq::send_receive_wait(token, reply, children[ind].second);
        if (reply.action == destroy and reply.parent_id == id) {
            rc = zmq_close(children[ind].second);
            assert(rc == 0);
            rc = zmq_ctx_destroy(children[ind].first);
            assert(rc == 0);
            auto it = children.begin();
            while (ind--) {
                ++it;
            }
            children.erase(it);
        }
        else if (reply.action == bind and reply.parent_id == id) {
            rc = zmq_close(children[ind].second);
            assert(rc == 0);
            rc = zmq_ctx_term(children[ind].first);
            assert(rc == 0);
            my_zmq::init_pair_socket(children[ind].first, children[ind].second);
            rc = zmq_bind(children[ind].second, ("tcp://*:" + std::to_string(PORT_BASE + id)).c_str());
            assert(rc == 0);
        }
        if (ok) {
            control_node.erase(id);
            std::cout << "OK: " << id << std::endl;
        }
        else {
            std::cout << "Error: Node " << id << " is unavailable" << std::endl;
        }
    }
    else {
        std::cout << "Error: Not found" << std::endl;
    }
}

int main() {
    int rc;
    bool ok;
    std::string s;
    std::thread new_thread;
    long long id;
    std::cout << "Create id parent: create calculation node (use parent = -1 if parent is control node)" << std::endl;
    std::cout << "Heartbeat milliseconds: ping calculation node with id $id" << std::endl;
    std::cout << "Remove id: delete calculation node with id " << std::endl;
    std::cout << "Exec id key val: add [key, val] add local dictionary" << std::endl;
    std::cout << "Exec id key: check local dictionary" << std::endl;
    while (std::cin >> s >> id) {
        if (s == "create") {
            long long parent_id;
            std::cin >> parent_id;
            int ind;
            if (parent_id == -1) {
                void* new_context = nullptr;
                void* new_socket = nullptr;
                my_zmq::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(), nullptr);
                    assert(rc != -1);
                    return 0;
                }
                else {
                    auto* token = new node_token_t({ ping, id, id });
                    node_token_t reply({ fail, id, id });
                    if (my_zmq::send_receive_wait(token, reply, new_socket) and reply.action == success) {//проверка создания нового сокета(выч
ноды)
                        children.emplace_back(std::make_pair(new_context, new_socket));//добавляем в вектор новый сокет ребёнка тип н дентей у
контрол ноды
                        control_node.insert(id);//вставляем ид в топологию
                    }
                    else {
                        rc = zmq_close(new_socket);
                        assert(rc == 0);
                        rc = zmq_ctx_destroy(new_context);
                        assert(rc == 0);
                    }
                }
                ping_storage.push_back(id);
            }
            else if ((ind = control_node.find(parent_id)) == -1) {
                std::cout << "Error: Not found" << std::endl;
                continue;
            }
            else {
                if (control_node.find(id) != -1) {
                    std::cout << "Error: Already exists" << std::endl;
                    continue;
                }
                auto* token = new node_token_t({ create, parent_id, id });
                node_token_t reply({ fail, id, id });
                if (my_zmq::send_receive_wait(token, reply, children[ind].second) and reply.action == success) {
                    control_node.insert(parent_id, id);
                    ping_storage.push_back(id);
                }
                else {
                    std::cout << "Error: Parent is unavailable" << std::endl;
                    continue;
                }
            }
        }


        else if (s == "remove") {
            delete_control_node(id);
        }


        else if (s == "heartbeat") {
            if (ping_storage.empty()) {
                std::cout << "Error: there are no calculation nodes at all" << std::endl;
                continue;
            }
            new_thread = std::thread(pinger, id);
        }


        else if (s == "exec") {
            ok = true;
            std::string key;
            char c;
            int val = -1;
            bool add = false;
            std::cin >> key;
            if ((c = getchar()) == ' ') {
                add = true;
                std::cin >> val;
            }
            int ind = control_node.find(id);
            if (ind == -1) {
                std::cout << "Error: Not found" << std::endl;
                continue;
            }
            key += SENTINEL;
            if (add) {
                for (auto i : key) {
                    auto* token = new node_token_t({ exec_add, i, id });
                    node_token_t reply({ fail, id, id });
                    if (!my_zmq::send_receive_wait(token, reply, children[ind].second) or reply.action != success) {
                        std::cout << "Fail: " << i << std::endl;
                        ok = false;
                        break;
                    }
                }
                auto* token = new node_token_t({ exec_add, val, id });
                node_token_t reply({ fail, id, id });
                if (!my_zmq::send_receive_wait(token, reply, children[ind].second) or reply.action != success) {
                    std::cout << "Fail: " << val << std::endl;
                    ok = false;
                }
            }
            else {
                for (auto i : key) {
                    auto* token = new node_token_t({ exec_check, i, id });
                    node_token_t reply({ fail, i, id });
                    if (!my_zmq::send_receive_wait(token, reply, children[ind].second) or reply.action != success) {
                        ok = false;
                        std::cout << "Fail: " << i << std::endl;
                        break;
                    }
                }
            }
            if (!ok) {
                std::cout << "Error: Node is unavailable" << std::endl;
            }
        }
      }
      new_thread.detach();
      return 0;
}
```

calculate_node.cpp:

```
#include "my_zmq.h"
#include <iostream>
#include <map>
#include <unistd.h>

long long node_id;

int main(int argc, char** argv) {
  std::string key;
  int val;
  std::map<std::string, int> dict;
  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 = nullptr;
  void* node_socket = nullptr;
  std::cout << "OK: " << getpid() << std::endl;

  bool has_child = false, awake = true, add = false;
  while (awake) {
    node_token_t token({ fail, 0, 0 });
    my_zmq::receive_msg(token, node_parent_socket);
```

```
auto* reply = new node_token_t({ fail, node_id, node_id });


if (token.action == bind and token.parent_id == node_id) {
  my_zmq::init_pair_socket(node_context, node_socket);
  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;
  auto* token_ping = new node_token_t({ ping, child_id, child_id });
  node_token_t reply_ping({ fail, child_id, child_id });
  if (my_zmq::send_receive_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);
    }
    my_zmq::init_pair_socket(node_context, node_socket);
    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(), nullptr);
      assert(rc != -1);
      return 0;
    }
    else {
      bool ok = true;
      if (has_child) {
        auto* token_bind = new node_token_t({ bind, token.id, child_id });
        node_token_t reply_bind({ fail, token.id, token.id });
        ok = my_zmq::send_receive_wait(token_bind, reply_bind, node_socket);
        ok = ok and (reply_bind.action == success);
      }
      if (ok) {
        auto* token_ping = new node_token_t({ ping, token.id, token.id });
        node_token_t reply_ping({ fail, token.id, token.id });
        ok = my_zmq::send_receive_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) {
        auto* token_down = new node_token_t(token);
        node_token_t reply_down(token);
        reply_down.action = fail;
        if (my_zmq::send_receive_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) {
        auto* token_down = new node_token_t(token);
        node_token_t reply_down(token);
        reply_down.action = fail;
        if (my_zmq::send_receive_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;
            auto* token_down = new node_token_t({ destroy, node_id, child_id });
            node_token_t reply_down = { fail, child_id, child_id };
            ok = my_zmq::send_receive_wait(token_down, reply_down, node_socket);
            if (reply_down.action == destroy) {
                rc = zmq_close(node_socket);
                assert(rc == 0);
                rc = zmq_ctx_destroy(node_context);
                assert(rc == 0);
                has_child = false;
                child_id = -1;
            }
            else if (reply_down.action == bind) {
                rc = zmq_close(node_socket);
                assert(rc == 0);
                rc = zmq_ctx_destroy(node_context);
                assert(rc == 0);
                my_zmq::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;
                auto* token_ping = new node_token_t({ ping, child_id, child_id });
                node_token_t reply_ping({ fail, child_id, child_id });
                ok = my_zmq::send_receive_wait(token_ping, reply_ping, node_socket) and (reply_ping.action == success);
            }
            if (ok) {
                reply->action = success;
            }
        }
        else if (token.id == node_id) {
            rc = zmq_close(node_socket);
            assert(rc == 0);
            rc = zmq_ctx_destroy(node_context);
            assert(rc == 0);
            awake = false;
            reply->action = bind;
            reply->id = child_id;
            reply->parent_id = token.parent_id;
        }
        else {
            auto* token_down = new node_token_t(token);
            node_token_t reply_down = token;
            reply_down.action = fail;
            if (my_zmq::send_receive_wait(token_down, reply_down, node_socket) and (reply_down.action == success)) {
                *reply = reply_down;
            }
        }
    }
    else if (token.id == node_id) {
        reply->action = destroy;
        awake = false;
    }
}


else if (token.action == exec_check) {
    if (token.id == node_id) {
        char c = token.parent_id;
        if (c == SENTINEL) {
            if (dict.find(key) != dict.end()) {
                std::cout << "OK:" << node_id << ":" << dict[key] << std::endl;
            }
            else {
                std::cout << "OK:" << node_id << ":'" << key << "' not found" << std::endl;
            }
            reply->action = success;
            key = "";
        }
        else {
            key += c;
            reply->action = success;
        }
    }
    else if (has_child) {
        auto* token_down = new node_token_t(token);
        node_token_t reply_down(token);
        reply_down.action = fail;
        if (my_zmq::send_receive_wait(token_down, reply_down, node_socket) and reply_down.action == success) {
            *reply = reply_down;
        }
    }
}


else if (token.action == exec_add) {
    if (token.id == node_id) {
        char c = token.parent_id;
        if (c == SENTINEL) {
            add = true;
            reply->action = success;
            }
            else if (add) {
                val = token.parent_id;
                dict[key] = val;
                std::cout << "OK:" << node_id << std::endl;
                add = false;
                key = "";
                reply->action = success;
            }
            else {
                key += c;
                reply->action = success;
            }
        }
        else if (has_child) {
            auto* token_down = new node_token_t(token);
            node_token_t reply_down(token);
            reply_down.action = fail;
            if (my_zmq::send_receive_wait(token_down, reply_down, node_socket) and reply_down.action == success) {
                *reply = reply_down;
            }
        }
    }
    my_zmq::send_msg_no_wait(reply, node_parent_socket);
  }
  rc = zmq_close(node_parent_socket);
  assert(rc == 0);
  rc = zmq_ctx_destroy(node_parent_context);
  assert(rc == 0);
}
```

STRACE

```
execve("./control_node", ["./control_node"], 0x7fffde231bc0 /* 20 vars */) = 0

brk(NULL)                               = 0x7fffedf96000

arch_prctl(0x3001 /* ARCH_??? */, 0x7ffff65c9a20) = -1 EINVAL (Invalid argument)

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

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

fstat(3, {st_mode=S_IFREG|0644, st_size=42585, ...}) = 0

mmap(NULL, 42585, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f02ae5d2000

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`z\1\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=675776, ...}) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f02ae5d0000
```

```
mmap(NULL, 678128, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f02ae4f0000

mmap(0x7f02ae506000, 430080, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x16000) = 0x7f02ae506000

mmap(0x7f02ae56f000, 126976, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x7f000) = 0x7f02ae56f000

mmap(0x7f02ae58e000, 32768, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9d000) = 0x7f02ae58e000

close(3)                        = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = 3

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

fstat(3, {st_mode=S_IFREG|0644, st_size=1956992, ...}) = 0

mmap(NULL, 1972224, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f02ae300000

mprotect(0x7f02ae396000, 1290240, PROT_NONE) = 0

mmap(0x7f02ae396000, 987136, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x96000) = 0x7f02ae396000

mmap(0x7f02ae487000, 299008, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x187000) = 0x7f02ae487000

mmap(0x7f02ae4d1000, 57344, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1d0000) = 0x7f02ae4d1000

mmap(0x7f02ae4df000, 10240, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f02ae4df000

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\3405\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=104984, ...}) = 0

mmap(NULL, 107592, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f02ae2e0000

mmap(0x7f02ae2e3000, 73728, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7f02ae2e3000

mmap(0x7f02ae2f5000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x15000) = 0x7f02ae2f5000

mmap(0x7f02ae2f9000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x18000) = 0x7f02ae2f9000

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\220\201\0\0\0\0\0\0"..., 832) = 832
```

```
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\345Ga\367\265T\320\374\301V)Yf]\223\337"..., 68, 824) =
68

fstat(3, {st_mode=S_IFREG|0755, st_size=157224, ...}) = 0

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\345Ga\367\265T\320\374\301V)Yf]\223\337"..., 68, 824) =
68

mmap(NULL, 140408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f02ae2bd000

mmap(0x7f02ae2c4000, 69632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x7f02ae2c4000

mmap(0x7f02ae2d5000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x18000) = 0x7f02ae2d5000

mmap(0x7f02ae2da000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c000) = 0x7f02ae2da000

mmap(0x7f02ae2dc000, 13432, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f02ae2dc000

close(3)                        = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360q\2\0\0\0\0\0"..., 832) = 832

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

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

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"...,
68, 880) = 68

fstat(3, {st_mode=S_IFREG|0755, st_size=2029224, ...}) = 0

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

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

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"...,
68, 880) = 68

mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f02ae0c0000

mprotect(0x7f02ae0e5000, 1847296, PROT_NONE) = 0

mmap(0x7f02ae0e5000, 1540096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25000) = 0x7f02ae0e5000

mmap(0x7f02ae25d000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x19d000) = 0x7f02ae25d000

mmap(0x7f02ae2a8000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f02ae2a8000

mmap(0x7f02ae2ae000, 13528, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f02ae2ae000
```

```
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\200\302\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=355016, ...}) = 0

mmap(NULL, 357384, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f02ae060000

mmap(0x7f02ae06c000, 229376, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xc000) = 0x7f02ae06c000

mmap(0x7f02ae0a4000, 73728, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x44000) = 0x7f02ae0a4000

mmap(0x7f02ae0b6000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x55000) = 0x7f02ae0b6000

close(3)                    = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libpgm-5.2.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\240L\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=302056, ...}) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f02ae050000

mmap(NULL, 321584, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f02ae000000

mmap(0x7f02ae004000, 163840, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4000) = 0x7f02ae004000

mmap(0x7f02ae02c000, 118784, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x2c000) = 0x7f02ae02c000

mmap(0x7f02ae049000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x48000) = 0x7f02ae049000

mmap(0x7f02ae04b000, 14384, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f02ae04b000

close(3)                    = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libnorm.so.1", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\257\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=690344, ...}) = 0

mmap(NULL, 1420000, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f02adea0000

mmap(0x7f02adeaa000, 421888, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xa000) = 0x7f02adeaa000

mmap(0x7f02adf11000, 217088, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x71000) = 0x7f02adf11000
```

```
mmap(0x7f02adf46000, 16384, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xa5000) = 0x7f02adf46000

mmap(0x7f02adf4a000, 723680, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f02adf4a000

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\0P\321\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=309712, ...}) = 0

mmap(NULL, 312128, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f02ade50000

mmap(0x7f02ade5b000, 204800, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xb000) = 0x7f02ade5b000

mmap(0x7f02ade8d000, 49152, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x3d000) = 0x7f02ade8d000

mmap(0x7f02ade99000, 16384, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x48000) = 0x7f02ade99000

close(3)                    = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\300\363\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=1369352, ...}) = 0

mmap(NULL, 1368336, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f02add01000

mmap(0x7f02add10000, 684032, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xf000) = 0x7f02add10000

mmap(0x7f02addb7000, 618496, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xb6000) = 0x7f02addb7000

mmap(0x7f02ade4e000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x14c000) = 0x7f02ade4e000

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 ?\2\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=902016, ...}) = 0

mmap(NULL, 904640, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f02adc20000

mprotect(0x7f02adc42000, 700416, PROT_NONE) = 0

mmap(0x7f02adc42000, 397312, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x22000) = 0x7f02adc42000
```

```
mmap(0x7f02adca3000, 299008, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x83000) = 0x7f02adca3000

mmap(0x7f02adced000, 65536, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xcc000) = 0x7f02adced000

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\240D\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=191040, ...}) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f02adc10000

mmap(NULL, 196696, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f02adbd0000

mprotect(0x7f02adbd4000, 172032, PROT_NONE) = 0

mmap(0x7f02adbd4000, 114688, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4000) = 0x7f02adbd4000

mmap(0x7f02adbf0000, 53248, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x20000) = 0x7f02adbf0000

mmap(0x7f02adbfe000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2d000) = 0x7f02adbfe000

mmap(0x7f02adc00000, 88, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f02adc00000

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\200$\0\0\0\0\0\0"..., 832) = 832

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

mmap(NULL, 24744, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f02adbc0000

mmap(0x7f02adbc2000, 8192, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f02adbc2000

mmap(0x7f02adbc4000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x4000) = 0x7f02adbc4000

mmap(0x7f02adbc5000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4000) = 0x7f02adbc5000

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\3605\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=56096, ...}) = 0
```

```
mmap(NULL, 58344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f02adbb0000

mmap(0x7f02adbb3000, 28672, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7f02adbb3000

mmap(0x7f02adbba000, 12288, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xa000) = 0x7f02adbba000

mmap(0x7f02adbbd000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xc000) = 0x7f02adbbd000

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"..., 832) = 832

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

mmap(NULL, 24592, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f02adba0000

mmap(0x7f02adba2000, 8192, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f02adba2000

mmap(0x7f02adba4000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x4000) = 0x7f02adba4000

mmap(0x7f02adba5000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4000) = 0x7f02adba5000

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

fstat(3, {st_mode=S_IFREG|0644, st_size=101320, ...}) = 0

mmap(NULL, 113280, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f02adb80000

mprotect(0x7f02adb84000, 81920, PROT_NONE) = 0

mmap(0x7f02adb84000, 65536, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4000) = 0x7f02adb84000

mmap(0x7f02adb94000, 12288, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x14000) = 0x7f02adb94000

mmap(0x7f02adb98000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17000) = 0x7f02adb98000

mmap(0x7f02adb9a000, 6784, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f02adb9a000

close(3)                    = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libdl.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 \22\0\0\0\0\0\0"..., 832) = 832
```

```
fstat(3, {st_mode=S_IFREG|0644, st_size=18816, ...}) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f02adb70000

mmap(NULL, 20752, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f02adb60000

mmap(0x7f02adb61000, 8192, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7f02adb61000

mmap(0x7f02adb63000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x3000) = 0x7f02adb63000

mmap(0x7f02adb64000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7f02adb64000

close(3)                              = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f02adb50000

arch_prctl(ARCH_SET_FS, 0x7f02adb51180) = 0

mprotect(0x7f02ae2a8000, 12288, PROT_READ) = 0

mprotect(0x7f02adb64000, 4096, PROT_READ) = 0

mprotect(0x7f02adb98000, 4096, PROT_READ) = 0

mprotect(0x7f02adba5000, 4096, PROT_READ) = 0

mprotect(0x7f02adbbd000, 4096, PROT_READ) = 0

mprotect(0x7f02ae2da000, 4096, PROT_READ) = 0

mprotect(0x7f02adbc5000, 4096, PROT_READ) = 0

mprotect(0x7f02adbfe000, 4096, PROT_READ) = 0

mprotect(0x7f02adced000, 57344, PROT_READ) = 0

mprotect(0x7f02ade4e000, 4096, PROT_READ) = 0

mprotect(0x7f02ade99000, 8192, PROT_READ) = 0

mprotect(0x7f02ae2f9000, 4096, PROT_READ) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f02adb40000

mprotect(0x7f02ae4d1000, 45056, PROT_READ) = 0

mprotect(0x7f02adf46000, 12288, PROT_READ) = 0

mprotect(0x7f02ae049000, 4096, PROT_READ) = 0

mprotect(0x7f02ae0b6000, 4096, PROT_READ) = 0

mprotect(0x7f02ae58e000, 28672, PROT_READ) = 0
mprotect(0x7f02ae5ef000, 4096, PROT_READ) = 0

mprotect(0x7f02ae5cd000, 4096, PROT_READ) = 0

munmap(0x7f02ae5d2000, 42585)          = 0

set_tid_address(0x7f02adb51450)        = 87

set_robust_list(0x7f02adb51460, 24)    = 0

rt_sigaction(SIGRTMIN, {sa_handler=0x7f02ae2c4bf0, sa_mask=[],
sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7f02ae2d23c0}, NULL, 8) = 0

rt_sigaction(SIGRT_1, {sa_handler=0x7f02ae2c4c90, sa_mask=[],
sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f02ae2d23c0}, NULL, 8) = 0

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

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

brk(NULL)                              = 0x7fffedf96000

brk(0x7fffedfb7000)                    = 0x7fffedfb7000

gettimeofday({tv_sec=1640746528, tv_usec=202230}, {tz_minuteswest=0, tz_dsttime=0}) = 0

futex(0x7f02ae4df6bc, FUTEX_WAKE_PRIVATE, 2147483647) = 0

futex(0x7f02ae4df6c8, FUTEX_WAKE_PRIVATE, 2147483647) = 0

fstat(1, {st_mode=S_IFCHR|0660, st_rdev=makedev(0x4, 0x1), ...}) = 0

ioctl(1, TCGETS, {B38400 opost isig icanon echo ...}) = 0

write(1, "Create id parent: create calcula"..., 86Create id parent: create calculation node (use parent = -1 if
parent is control node)

) = 86

write(1, "Heartbeat milliseconds: ping cal"..., 58Heartbeat milliseconds: ping calculation node with id $id

) = 58

write(1, "Remove id: delete calculation no"..., 44Remove id: delete calculation node with id

) = 44

write(1, "Exec id key val: add [key, val] "..., 53Exec id key val: add [key, val] add local dictionary

) = 53

write(1, "Exec id key: check local diction"..., 36Exec id key: check local dictionary

) = 36

fstat(0, {st_mode=S_IFCHR|0660, st_rdev=makedev(0x4, 0x1), ...}) = 0

ioctl(0, TCGETS, {B38400 opost isig icanon echo ...}) = 0

read(0, create 10 -1
"create 10 -1\n", 4096)        = 13

clock_gettime(CLOCK_REALTIME_COARSE, {tv_sec=1640746540, tv_nsec=395348300}) = 0

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

read(3, "0-7\n", 8192)              = 4

close(3)                        = 0

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

fstat(3, {st_mode=S_IFDIR|0755, st_size=0, ...}) = 0

getdents64(3, /* 13 entries */, 32768)  = 336

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

close(3)                        = 0

getpid()                        = 87

sched_getaffinity(87, 128, [0, 1, 2, 3, 4, 5, 6, 7]) = 64

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

fstat(3, {st_mode=S_IFREG|0644, st_size=510, ...}) = 0

read(3, "# /etc/nsswitch.conf\n#\n# Example"..., 4096) = 510

read(3, "", 4096)                   = 0

close(3)                        = 0

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

fstat(3, {st_mode=S_IFREG|0644, st_size=42585, ...}) = 0

mmap(NULL, 42585, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f02ae5d2000

close(3)                        = 0

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

stat("/lib/x86_64-linux-gnu/tls/haswell/x86_64", 0x7ffff65c6c80) = -1 ENOENT (No such file or
directory)

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

stat("/lib/x86_64-linux-gnu/tls/haswell", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

stat("/lib/x86_64-linux-gnu/tls/x86_64", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)
```

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

stat("/lib/x86_64-linux-gnu/tls", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

stat("/lib/x86_64-linux-gnu/haswell/x86_64", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

stat("/lib/x86_64-linux-gnu/haswell", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

stat("/lib/x86_64-linux-gnu/x86_64", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

stat("/lib/x86_64-linux-gnu", {st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0

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

stat("/usr/lib/x86_64-linux-gnu/tls/haswell/x86_64", 0x7ffff65c6c80) = -1 ENOENT (No such file or
directory)

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

stat("/usr/lib/x86_64-linux-gnu/tls/haswell", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

stat("/usr/lib/x86_64-linux-gnu/tls/x86_64", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

stat("/usr/lib/x86_64-linux-gnu/tls", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

stat("/usr/lib/x86_64-linux-gnu/haswell/x86_64", 0x7ffff65c6c80) = -1 ENOENT (No such file or
directory)

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

stat("/usr/lib/x86_64-linux-gnu/haswell", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

```
stat("/usr/lib/x86_64-linux-gnu/x86_64", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

stat("/usr/lib/x86_64-linux-gnu", {st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0

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

stat("/lib/tls/haswell/x86_64", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

stat("/lib/tls/haswell", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

stat("/lib/tls/x86_64", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

stat("/lib/tls", 0x7ffff65c6c80)        = -1 ENOENT (No such file or directory)

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

stat("/lib/haswell/x86_64", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

stat("/lib/haswell", 0x7ffff65c6c80)    = -1 ENOENT (No such file or directory)

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

stat("/lib/x86_64", 0x7ffff65c6c80)     = -1 ENOENT (No such file or directory)

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

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

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

stat("/usr/lib/tls/haswell/x86_64", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

stat("/usr/lib/tls/haswell", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

```
stat("/usr/lib/tls/x86_64", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

stat("/usr/lib/tls", 0x7ffff65c6c80)    = -1 ENOENT (No such file or directory)

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

stat("/usr/lib/haswell/x86_64", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

stat("/usr/lib/haswell", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

stat("/usr/lib/x86_64", 0x7ffff65c6c80) = -1 ENOENT (No such file or directory)

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

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

munmap(0x7f02ae5d2000, 42585)          = 0

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

fstat(3, {st_mode=S_IFREG|0644, st_size=42585, ...}) = 0

mmap(NULL, 42585, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f02ae5d2000

close(3)                                = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libnss_files.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\3005\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=51832, ...}) = 0

mmap(NULL, 79672, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f02adb20000

mmap(0x7f02adb23000, 28672, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7f02adb23000

mmap(0x7f02adb2a000, 8192, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xa000) = 0x7f02adb2a000

mmap(0x7f02adb2c000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xb000) = 0x7f02adb2c000

mmap(0x7f02adb2e000, 22328, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f02adb2e000

close(3)                                = 0
```

```
mprotect(0x7f02adb2c000, 4096, PROT_READ) = 0

munmap(0x7f02ae5d2000, 42585)          = 0

openat(AT_FDCWD, "/etc/protocols", O_RDONLY|O_CLOEXEC) = 3

lseek(3, 0, SEEK_CUR)                  = 0

fstat(3, {st_mode=S_IFREG|0644, st_size=2932, ...}) = 0

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

lseek(3, 0, SEEK_CUR)                  = 2932

read(3, "", 4096)                      = 0

close(3)                               = 0

gettimeofday({tv_sec=1640746540, tv_usec=447459}, 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

getrandom("\xed\xa7\xf9\xc7\x89\x66\xac\xec\x87\x06\x0c\xce\xb5\x1e\x64\x18", 16, 0) = 16

getrandom("\x6e\x76\x25\xb2\x37\x5c\x69\x9a\xc7\xe9\x1d\xc7\x12\x66\x72\xb9", 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

clock_gettime(CLOCK_MONOTONIC, {tv_sec=228, tv_nsec=971427800}) = 0

epoll_create1(EPOLL_CLOEXEC)           = 5

epoll_ctl(5, EPOLL_CTL_ADD, 4, {0, {u32=3992627936, u64=140737186015968}}) = 0

epoll_ctl(5, EPOLL_CTL_MOD, 4, {EPOLLIN, {u32=3992627936, u64=140737186015968}}) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7f02ad310000

mprotect(0x7f02ad311000, 8388608, PROT_READ|PROT_WRITE) = 0

clone(child_stack=0x7f02adb0fd30,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYS
VSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[88],
tls=0x7f02adb10700, child_tidptr=0x7f02adb109d0) = 88

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

clock_gettime(CLOCK_MONOTONIC, {tv_sec=228, tv_nsec=974797800}) = 0

epoll_create1(EPOLL_CLOEXEC)          = 7

epoll_ctl(7, EPOLL_CTL_ADD, 6, {0, {u32=3992630064, u64=140737186018096}}) = 0

epoll_ctl(7, EPOLL_CTL_MOD, 6, {EPOLLIN, {u32=3992630064, u64=140737186018096}}) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7f02acb00000

mprotect(0x7f02acb01000, 8388608, PROT_READ|PROT_WRITE) = 0

clone(child_stack=0x7f02ad2ffd30,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYS
VSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[89],
tls=0x7f02ad300700, child_tidptr=0x7f02ad3009d0) = 89

clock_gettime(CLOCK_MONOTONIC, {tv_sec=228, tv_nsec=976958100}) = 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

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(8010), sin_addr=inet_addr("0.0.0.0")}, 16) = 0

listen(9, 100)                      = 0

getsockname(9, {sa_family=AF_INET, sin_port=htons(8010), sin_addr=inet_addr("0.0.0.0")}, [128->16])
= 0

getsockname(9, {sa_family=AF_INET, sin_port=htons(8010), sin_addr=inet_addr("0.0.0.0")}, [128->16])
= 0

write(6, "\1\0\0\0\0\0\0\0", 8)       = 8
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=0x7f02adb51450) = 90

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

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

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

clock_gettime(CLOCK_MONOTONIC, {tv_sec=228, tv_nsec=989376700}) = 0

poll([{fd=8, events=POLLIN}], 1, 1000OK: 90

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

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

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

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

clock_gettime(CLOCK_MONOTONIC, {tv_sec=229, tv_nsec=10995000}) = 0

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

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

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

read(0, exec 10 sava 18

"exec 10 sava 18\n", 4096)      = 16

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

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

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

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

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

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

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

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

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

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

OK:10

clock_gettime(CLOCK_MONOTONIC, {tv_sec=237, tv_nsec=256938300}) = 0

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

```
read(8, "\1\0\0\0\0\0\0\0", 8)          = 8
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
read(0, exec 10 sava
"exec 10 sava\n", 4096)         = 13
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
clock_gettime(CLOCK_MONOTONIC, {tv_sec=244, tv_nsec=243435800}) = 0
poll([{fd=8, events=POLLIN}], 1, 1000)  = 1 ([{fd=8, revents=POLLIN}])
read(8, "\1\0\0\0\0\0\0\0", 8)          = 8
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
OK:10:18
clock_gettime(CLOCK_MONOTONIC, {tv_sec=244, tv_nsec=246598300}) = 0
poll([{fd=8, events=POLLIN}], 1, 1000)  = 1 ([{fd=8, revents=POLLIN}])
read(8, "\1\0\0\0\0\0\0\0", 8)          = 8
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
read(0, ^Cstrace: Process 87 detached
 <detached ...>
```

Вывод

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.