Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №6-8 по курсу

«Операционные системы»

Студентка:

Варламова Анна Борисовна

Группа: М80-207Б-20

Преподаватель: Миронов Е.С.

Оценка: ____________________

Дата: 27.12.2021

Задание

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: 7. Топология —3- бинарное дерево. Тип вычислительной команды — 4- поиск подстроки в строке. Тип проверки узлов на доступность —2- пинг узлов по ID.

Общие сведения о программе

Программа состоит из двух файлов, которые компилируются в исполнительные файлы(которые представляют управляющий 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) сокеты на REQ - REP.
- Дочерние узлы проверяют, может ли быть команда выполнена в данном узле, если нет, то команда пересылается в один из дочерних узлов, из которого возвращается некоторое сообщение(об успехе или об ошибке), которое потом пересылается обратно по дереву.
- Для корректной проверки на доступность узлов используется дерево, эмулирующее поведение узлов в данной топологии(например, при удалении узла, удаляются все его потомки).
- При удалении узла все его потомки рекурсивно уничтожаются.

Код программы

parent.cpp

```
#include "zmq.hpp"
#include <sstream>
#include <string>
#include <iostream>
#include <zconf.h>
#include <vector>
#include <signal.h>
#include <sstream>
#include <set>
#include <algorithm>

// g++ parent.cpp -lzmq -o main -w


int main(){
    zmq::context_t context(1); // служебная структура контекст
    zmq::socket_t main_socket(context, ZMQ_REP); // поднятие сокета в контексте ZMQ_REP
для отправки запросов и получения ответов
    std::string adr = "tcp://127.0.0.1:300";
    std::string command;
    int child_id = 0;
    while(1){
        std::cout << "command:";
        std::cin >> command;
        if(command == "create") {
            if(child_id == 0){
                int id;
                std::cin >> id;
                int id_tmp = id - 1;
                main_socket.bind(adr + std::to_string(++id_tmp));
                std::string new_adr = adr + std::to_string(id_tmp);
                char* adr_ = new char[new_adr.size() + 1];
                memcpy(adr_, new_adr.c_str(), new_adr.size() + 1);
                char* id_ = new char[std::to_string(id).size() + 1];
                memcpy(id_, std::to_string(id).c_str(), std::to_string(id).size() + 1);
                char* args[] = {"./child", adr_, id_, NULL};
                int id2 = fork();
                if (id2 == -1) {
                    std::cout << "Unable to create first worker node" << std::endl;
                    id = 0;
                    exit(1);
                } else if(id2 == 0){
```

```
                execv("./child", args);
            } else {
                child_id = id;
            }
            zmq::message_t message;
            main_socket.recv(&message);
            std::string recieved_message(static_cast<char*>(message.data()), message.size());
            std::cout << recieved_message << std::endl;
            delete [] adr_;
            delete [] id_;
        } else {
            int id;
            std::cin >> id;
            std::string message_string = command + " " + std::to_string(id);
            zmq::message_t message(message_string.size());
            memcpy(message.data(), message_string.c_str(), message_string.size());
            main_socket.send(message);

            // catch message from new node
            main_socket.recv(&message);
            std::string recieved_message(static_cast<char*>(message.data()), message.size());
            std::cout << recieved_message << std::endl;
        }
    } else if (command == "exec") {
        int id;
        std::string big, small;
        std::cin >> id;
        std::cin >> big >> small;
        std::string message_string = command + " " + std::to_string(id) + " " + big + " " + small;
        zmq::message_t message(message_string.size());
        memcpy(message.data(), message_string.c_str(), message_string.size());
        main_socket.send(message);
        // return value from map
        main_socket.recv(&message);
        std::string recieved_message(static_cast<char*>(message.data()), message.size());
        std::cout << recieved_message << std::endl;
    } else if (command == "ping") {
        int id;
        std::cin >> id;
        std::string message_string = command + " " + std::to_string(id);
        zmq::message_t message(message_string.size());
        memcpy(message.data(), message_string.c_str(), message_string.size());
        main_socket.send(message);
        // receive answer from child
        main_socket.recv(&message);
        std::string recieved_message(static_cast<char*>(message.data()), message.size());
        std::cout << recieved_message << std::endl;
        } else if(command == "kill"){
            int id;
            std::cin >> id;
            if(child_id == 0){
                std::cout << "Error: there isn't nodes" << std::endl;
            } else if(child_id == id){
                std::string kill_message = command + " " + std::to_string(id);
                zmq::message_t message(kill_message.size());
                memcpy(message.data(), kill_message.c_str(), kill_message.size());
                main_socket.send(message);
                main_socket.recv(&message);
                std::string received_message(static_cast<char*>(message.data()), message.size());
                std::cout << received_message << std::endl;
                std::cout << "Tree deleted successfully" << std::endl;
                return 0;
            } else {
                std::string kill_message = command + " " + std::to_string(id);
                zmq::message_t message(kill_message.size());
                memcpy(message.data(), kill_message.c_str(), kill_message.size());
                main_socket.send(message);
                main_socket.recv(&message);
                std::string received_message(static_cast<char*>(message.data()), message.size());
                std::cout << received_message << std::endl;
            }
        } else if(command == "exit"){
            if(child_id){
                std::string kill_message = "DIE";
                zmq::message_t message(kill_message.size());
                memcpy(message.data(), kill_message.c_str(), kill_message.size());
                main_socket.send(message);
                std::cout << "Tree was deleted" << std::endl;
            }
            main_socket.close();
            context.close();
            break;
        } else {
            std::cout << "Error: incorrect command\n";
        }
    }
}
```

child.cpp

```
\#include "zmq.hpp"
#include <sstream>
#include <string>
#include <iostream>
```

```
#include <zconf.h>
#include <vector>
#include <signal.h>
#include <fstream>
#include <algorithm>
#include <map>

// g++ child.cpp -lzmq -o child -w


void send_message(std::string message_string, zmq::socket_t& socket){
    zmq::message_t message_back(message_string.size());
    memcpy(message_back.data(), message_string.c_str(), message_string.size());
    if(!socket.send(message_back)){
        std::cout << "Error: can't send message from node with pid " << getpid() << std::endl;
    }
}


int main(int argc, char * argv[]){
    std::string adr = argv[1];
    zmq::context_t context(1);
    zmq::socket_t main_socket(context, ZMQ_REQ);
    main_socket.connect(argv[1]);
    send_message("OK: " + std::to_string(getpid()), main_socket);
    int id = std::stoi(argv[2]); // id of this node
    std::map<std::string, int> m;
    int left_id = 0;
    int right_id = 0;
    zmq::context_t context_l(1);
    zmq::context_t context_r(1);
    zmq::socket_t left_socket(context_l, ZMQ_REP);
    std::string adr_left = "tcp://127.0.0.1:300";
    zmq::socket_t right_socket(context_r, ZMQ_REP);
    std::string adr_right = "tcp://127.0.0.1:300";
    while(1){
        zmq::message_t message_main;
        main_socket.recv(&message_main);
        std::string recieved_message(static_cast<char*>(message_main.data()), mes-
sage_main.size());
        std::string command;
        for(int i = 0; i < recieved_message.size(); ++i){
            if(recieved_message[i] != ' '){
                command += recieved_message[i];
            } else {
                break;
            }
```

```
}
if(command == "exec"){
    int id_proc; // id of node for adding
    std::string id_proc_;
    std::string big, small, for_return;
    int flag = 0;
    std::vector<int> answers;
    for(int i = 5; i < recieved_message.size(); ++i){
        if(recieved_message[i] != ' '){
            id_proc_ += recieved_message[i];
        } else {
            break;
        }
    }
    id_proc = std::stoi(id_proc_);
    if(id_proc == id) { // id == proc_id
        for(int i = 6 + id_proc_.size(); i < recieved_message.size(); ++i){
            if (recieved_message[i] == ' ')
                ++flag;
            if ((recieved_message[i] != ' ') && (flag == 0)) {
                big += recieved_message[i];
            } else if ((recieved_message[i] != ' ') && (flag == 1)){
                small += recieved_message[i];
            }
        }
        if (big.size() >= small.size()) {
            int start = 0;
            while(big.find(small, start) != -1){
                start = big.find(small, start);
                answers.push_back(start);
                ++start;
            }
        }
        if(answers.size() == 0){
            for_return = "-1";
        }else{
            for_return = std::to_string(answers[0]);
            for(int i = 1; i < answers.size();++i) {
                for_return = for_return + ";" + std::to_string(answers[i]);
            }
        }
        for_return = "OK:" + id_proc_ + ":" + for_return;
        send_message(for_return, main_socket);
    } else {
        if(id > id_proc){
            if(left_id == 0){ // if node not exists
                std::string message_string = "Error:id: Not found";
                send_message("Error:id: Not found", main_socket);
            } else {
                zmq::message_t message(recieved_message.size());
                memcpy(message.data(), recieved_message.c_str(), recieved_message.size());
                if(!left_socket.send(message)){
                    std::cout << "Error: can't send message to left node from node with pid: " <<
getpid() << std::endl;
                }
                // catch and send to parent
                if(!left_socket.recv(&message)){
                    std::cout << "Error: can't receive message from left node in node with pid: " <<
getpid() << std::endl;
                }
                if(!main_socket.send(message)){
                    std::cout << "Error: can't send message to main node from node with pid: " <<
getpid() << std::endl;
                }
            }
        } else {
            if(right_id == 0){ // if node not exists
                std::string message_string = "Error:id: Not found";
                zmq::message_t message(message_string.size());
                memcpy(message.data(), message_string.c_str(), message_string.size());
                if(!main_socket.send(message)){
                    std::cout << "Error: can't send message to main node from node with pid: " <<
getpid() << std::endl;
                }
            } else {
                zmq::message_t message(recieved_message.size());
                memcpy(message.data(), recieved_message.c_str(), recieved_message.size());
                if(!right_socket.send(message)){
                    std::cout << "Error: can't send message to right node from node with pid: " <<
getpid() << std::endl;
                }
                // catch and send to parent
                if(!right_socket.recv(&message)){
                    std::cout << "Error: can't receive message from left node in node with pid: " <<
getpid() << std::endl;
                }
                if(!main_socket.send(message)){
                    std::cout << "Error: can't send message to main node from node with pid: " <<
getpid() << std::endl;
                }
            }
        }
    }
} else if(command == "create"){
    int id_proc; // id of node for creating
    std::string id_proc_;
    for(int i = 7; i < recieved_message.size(); ++i){
        if(recieved_message[i] != ' '){
            id_proc_ += recieved_message[i];
        } else {
            break;
        }
    }
    id_proc = std::stoi(id_proc_);
    if(id_proc == id){
        send_message("Error: Already exists", main_socket);
    } else if(id_proc > id){
        if(right_id == 0){ // there is not right node
            right_id = id_proc;
            int right_id_tmp = right_id - 1;
            right_socket.bind(adr_right + std::to_string(++right_id_tmp));
            adr_right += std::to_string(right_id_tmp);
            char* adr_right_ = new char[adr_right.size() + 1];
            memcpy(adr_right_, adr_right.c_str(), adr_right.size() + 1);
            char* right_id_ = new char[std::to_string(right_id).size() + 1];
            memcpy(right_id_, std::to_string(right_id).c_str(), std::to_string(right_id).size() + 1);
            char* args[] = {"./child", adr_right_, right_id_, NULL};
            int f = fork();
            if(f == 0){
                execv("./child", args);
            } else if (f == -1){
                std::cout << "Error in forking in node with pid: " << getpid() << std::endl;
            } else {
                // catch message from new node
                zmq::message_t message_from_node;
                if(!right_socket.recv(&message_from_node)){
                    std::cout << "Error: can't receive message from right node in node with pid:" <<
getpid() << std::endl;
                }
                std::string
recieved_message_from_node(static_cast<char*>(message_from_node.data()), mes-
sage_from_node.size());
                // send message to main node
                if(!main_socket.send(message_from_node)){
                    std::cout << "Error: can't send message to main node from node with pid:" <<
getpid() << std::endl;
                }
            }
            delete [] adr_right_;
            delete [] right_id_;
        } else { // send task to right node
```

```
                    send_message(recieved_message, right_socket);
                    // catch and send to parent
                    zmq::message_t message;
                    if(!right_socket.recv(&message)){
                        std::cout << "Error: can't receive message from left node in node with pid: " <<
getpid() << std::endl;
                    }
                    if(!main_socket.send(message)){
                        std::cout << "Error: can't send message to main node from node with pid: " <<
getpid() << std::endl;
                    }
                }
            } else {
                if(left_id == 0){ // there is not left node
                    left_id = id_proc;
                    int left_id_tmp = left_id - 1;
                    left_socket.bind(adr_left + std::to_string(++left_id_tmp));
                    adr_left += std::to_string(left_id_tmp);
                    char* adr_left_ = new char[adr_left.size() + 1];
                    memcpy(adr_left_, adr_left.c_str(), adr_left.size() + 1);
                    char* left_id_ = new char[std::to_string(left_id).size() + 1];
                    memcpy(left_id_, std::to_string(left_id).c_str(), std::to_string(left_id).size() + 1);
                    char* args[] = {"./child", adr_left_, left_id_, NULL};
                    int f = fork();
                    if(f == 0){
                        execv("./child", args);
                    } else if(f == -1){
                        std::cout << "Error in forking in node with pid: " << getpid() << std::endl;
                    } else {
                        // catch message from new node
                        zmq::message_t message_from_node;
                        if(!left_socket.recv(&message_from_node)){
                            std::cout << "Error: can't receive message from left node in node with pid:" <<
getpid() << std::endl;
                        }
                        std::string
recieved_message_from_node(static_cast<char*>(message_from_node.data()), mes-
sage_from_node.size());
                        // send message to main node
                        if(!main_socket.send(message_from_node)){
                            std::cout << "Error: can't send message to main node from node with pid:" <<
getpid() << std::endl;
                        }
                    }
                    delete [] adr_left_;
                    delete [] left_id_;
                } else { // send task to left node
                    send_message(recieved_message, left_socket);
                    // catch and send to parent
                    zmq::message_t message;
                    if(!left_socket.recv(&message)){
                        std::cout << "Error: can't receive message from left node in node with pid: " <<
getpid() << std::endl;
                    }
                    if(!main_socket.send(message)){
                        std::cout << "Error: can't send message to main node from node with pid: " <<
getpid() << std::endl;
                    }
                }
            }
        } else if(command == "ping") {
            int id_proc; // id of node for creating
            std::string id_proc_;
            for(int i = 5; i < recieved_message.size(); ++i){
                if(recieved_message[i] != ' '){
                    id_proc_ += recieved_message[i];
                } else {
                    break;
                }
            }
            id_proc = std::stoi(id_proc_);
            if(id_proc == id){
                send_message("OK: 1", main_socket);
            } else if(id_proc < id) {
                if(left_id == 0){
                    send_message("OK: 0", main_socket);
                } else {
                    left_socket.send(message_main);
                    zmq::message_t answ;
                    left_socket.recv(&answ);
                    main_socket.send(answ);
                }
            } else if(id_proc > id) {
                if(right_id == 0){
                    send_message("OK: 0", main_socket);
                } else {
                    right_socket.send(message_main);
                    zmq::message_t answ;
                    right_socket.recv(&answ);
                    main_socket.send(answ);
                }
            }
        } else if(command == "kill") {
            int id_proc; // id of node for killing
            std::string id_proc_;
            for(int i = 5; i < recieved_message.size(); ++i){
                if(recieved_message[i] != ' '){
                    id_proc_ += recieved_message[i];
                } else {
                    break;
                }
            }
            id_proc = std::stoi(id_proc_);
            if(id_proc > id){
                if(right_id == 0){
                    send_message("Error: there isn`t node with this id", main_socket);
                } else {
                    if(right_id == id_proc){
                        send_message("Ok: " + std::to_string(right_id), main_socket);
                        send_message("DIE", right_socket);
                        right_socket.unbind(adr_right);
                        adr_right = "tcp://127.0.0.1:300";
                        right_id = 0;
                    } else {
                        right_socket.send(message_main);
                        zmq::message_t message;
                        right_socket.recv(&message);
                        main_socket.send(message);
                    }
                }
            } else if(id_proc < id){
                if(left_id == 0){
                    send_message("Error: there isn`t node with this id", main_socket);
                } else {
                    if(left_id == id_proc){
                        send_message("Ok: " + std::to_string(left_id), main_socket);
                        send_message("DIE", left_socket);
                        left_socket.unbind(adr_left);
                        adr_left = "tcp://127.0.0.1:300";
                        left_id = 0;
                    } else {
                        left_socket.send(message_main);
                        zmq::message_t message;
                        left_socket.recv(&message);
                        main_socket.send(message);
                    }
                }
            }
        } else if (command == "DIE") {
            if (left_id){
                send_message("DIE", left_socket);
                left_socket.unbind(adr_left);
                adr_left = "tcp://127.0.0.1:300";
                left_id = 0;
            }
            if (right_id){
                send_message("DIE", right_socket);
                right_socket.unbind(adr_right);
                adr_right = "tcp://127.0.0.1:300";
                right_id = 0;
            }
            main_socket.unbind(adr);
            return 0;
        }
      }
    }
```

Использование утилиты strace

```
ann@ann:~/os/lab6$ strace ./main
execve("./main", ["./main"], 0x7ffdeaa21980 /* 63 vars */) = 0
brk(NULL)                               = 0x556b105e4000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
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=80719, ...}) = 0
mmap(NULL, 80719, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fd9d73f5000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/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\0P?\1\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=630464, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7fd9d73f3000
mmap(NULL, 2725560, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7fd9d6f46000
mprotect(0x7fd9d6fd9000, 2097152, PROT_NONE) = 0
mmap(0x7fd9d71d9000, 28672, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x93000) = 0x7fd9d71d9000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/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\220\304\10\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=1594864, ...}) = 0
mmap(NULL, 3702848, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7fd9d6bbd000
mprotect(0x7fd9d6d36000, 2097152, PROT_NONE) = 0
```

```
mmap(0x7fd9d6f36000, 49152, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x179000) = 0x7fd9d6f36000
mmap(0x7fd9d6f42000, 12352, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fd9d6f42000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\300*\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=96616, ...}) = 0
mmap(NULL, 2192432, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7fd9d69a5000
mprotect(0x7fd9d69bc000, 2093056, PROT_NONE) = 0
mmap(0x7fd9d6bbb000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x16000) = 0x7fd9d6bbb000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20\35\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030928, ...}) = 0
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7fd9d65b4000
mprotect(0x7fd9d679b000, 2097152, PROT_NONE) = 0
mmap(0x7fd9d699b000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7fd9d699b000
mmap(0x7fd9d69a1000, 15072, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fd9d69a1000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/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\340\251\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=330440, ...}) = 0
```

```
mmap(NULL, 2425864, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7fd9d6363000
mprotect(0x7fd9d63b3000, 2093056, PROT_NONE) = 0
mmap(0x7fd9d65b2000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4f000) = 0x7fd9d65b2000
close(3)                          = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/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\0000;\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=293784, ...}) = 0
mmap(NULL, 2406448, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7fd9d6117000
mprotect(0x7fd9d615e000, 2093056, PROT_NONE) = 0
mmap(0x7fd9d635d000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x46000) = 0x7fd9d635d000
mmap(0x7fd9d635f000, 14384, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fd9d635f000
close(3)                          = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/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\0000\374\1\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=522248, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7fd9d73f1000
mmap(NULL, 3340624, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7fd9d5de7000
mprotect(0x7fd9d5e64000, 2097152, PROT_NONE) = 0
mmap(0x7fd9d6064000, 12288, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7d000) = 0x7fd9d6064000
mmap(0x7fd9d6067000, 719184, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fd9d6067000
```

```
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/librt.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\"\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=31680, ...}) = 0
mmap(NULL, 2128864, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7fd9d5bdf000
mprotect(0x7fd9d5be6000, 2093056, PROT_NONE) = 0
mmap(0x7fd9d5de5000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x6000) = 0x7fd9d5de5000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
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\0000b\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=144976, ...}) = 0
mmap(NULL, 2221184, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7fd9d59c0000
mprotect(0x7fd9d59da000, 2093056, PROT_NONE) = 0
mmap(0x7fd9d5bd9000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19000) = 0x7fd9d5bd9000
mmap(0x7fd9d5bdb000, 13440, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fd9d5bdb000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
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\200\272\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=1700792, ...}) = 0
mmap(NULL, 3789144, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7fd9d5622000
mprotect(0x7fd9d57bf000, 2093056, PROT_NONE) = 0
mmap(0x7fd9d59be000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19c000) = 0x7fd9d59be000
```

```
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7fd9d73ef000
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7fd9d73ec000
arch_prctl(ARCH_SET_FS, 0x7fd9d73ecb80) = 0
mprotect(0x7fd9d699b000, 16384, PROT_READ) = 0
mprotect(0x7fd9d59be000, 4096, PROT_READ) = 0
mprotect(0x7fd9d5bd9000, 4096, PROT_READ) = 0
mprotect(0x7fd9d5de5000, 4096, PROT_READ) = 0
mprotect(0x7fd9d6bbb000, 4096, PROT_READ) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7fd9d73ea000
mprotect(0x7fd9d6f36000, 40960, PROT_READ) = 0
mprotect(0x7fd9d6064000, 8192, PROT_READ) = 0
mprotect(0x7fd9d635d000, 4096, PROT_READ) = 0
mprotect(0x7fd9d65b2000, 4096, PROT_READ) = 0
mprotect(0x7fd9d71d9000, 24576, PROT_READ) = 0
mprotect(0x556b0fcf2000, 4096, PROT_READ) = 0
mprotect(0x7fd9d7409000, 4096, PROT_READ) = 0
munmap(0x7fd9d73f5000, 80719)           = 0
set_tid_address(0x7fd9d73ece50)         = 8955
set_robust_list(0x7fd9d73ece60, 24)     = 0
rt_sigaction(SIGRTMIN, {sa_handler=0x7fd9d59c5cb0, sa_mask=[],
sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7fd9d59d2980}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x7fd9d59c5d50, sa_mask=[],
sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7fd9d59d2980}, NULL,
8) = 0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY})
= 0
brk(NULL)                               = 0x556b105e4000
brk(0x556b10605000)                     = 0x556b10605000
futex(0x7fd9d6f4309c, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7fd9d6f430a8, FUTEX_WAKE_PRIVATE, 2147483647) = 0
openat(AT_FDCWD, "/sys/devices/system/cpu/online", O_RDONLY|O_CLOEXEC) = 3
read(3, "0-3\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
getdents(3, /* 22 entries */, 32768)    = 656
getdents(3, /* 0 entries */, 32768)     = 0
close(3)                                = 0
getpid()                                = 8955
sched_getaffinity(8955, 128, [0, 1, 2, 3]) = 8
openat(AT_FDCWD, "/etc/nsswitch.conf", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=556, ...}) = 0
read(3, "# /etc/nsswitch.conf\n#\n# Example"..., 4096) = 556
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=80719, ...}) = 0
mmap(NULL, 80719, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fd9d73f5000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40) = -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=16384, ...}) = 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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40) = -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=69632, ...}) = 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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40)        = -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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40)    = -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", 0x7ffcf5f73d40)     = -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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40)    = -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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40) = -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", 0x7ffcf5f73d40) = -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(0x7fd9d73f5000, 80719)           = 0
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=80719, ...}) = 0
mmap(NULL, 80719, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fd9d73f5000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
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\0P#\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=47568, ...}) = 0
mmap(NULL, 2168632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7fd9d5410000
mprotect(0x7fd9d541b000, 2093056, PROT_NONE) = 0
mmap(0x7fd9d561a000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xa000) = 0x7fd9d561a000
mmap(0x7fd9d561c000, 22328, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fd9d561c000
close(3)                                = 0
mprotect(0x7fd9d561a000, 4096, PROT_READ) = 0
munmap(0x7fd9d73f5000, 80719)           = 0
openat(AT_FDCWD, "/etc/protocols", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=2932, ...}) = 0
read(3, "# Internet (IP) protocols\n#\n# Up"..., 4096) = 2932
read(3, "", 4096)                       = 0
close(3)                                = 0
eventfd2(0, EFD_CLOEXEC)                = 3
fcntl(3, F_GETFL)                       = 0x2 (flags O_RDWR)
fcntl(3, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
fcntl(3, F_GETFL)                       = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(3, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
getrandom("\x70\x88\x38\x74\x18\x25\xd0\x04\xe8\x53\xed\x35\xed\x36\x6f\x82", 16, 0) = 16
getrandom("\xbc\x84\xbc\x70\x2f\x6b\xb0\x65\x3b\x03\x79\x5e\x6d\x5e\xf4\x30", 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
epoll_create1(EPOLL_CLOEXEC)         = 5
epoll_ctl(5, EPOLL_CTL_ADD, 4, {0, {u32=274688032, u64=93918324549664}}) = 0
epoll_ctl(5, EPOLL_CTL_MOD, 4, {EPOLLIN, {u32=274688032, u64=93918324549664}}) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -
1, 0) = 0x7fd9d4c0f000
mprotect(0x7fd9d4c10000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7fd9d540eb70,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLON
E_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
parent_tidptr=0x7fd9d540f9d0, tls=0x7fd9d540f700, child_tidptr=0x7fd9d540f9d0) = 8956
openat(AT_FDCWD, "/proc/self/task/8956/comm", O_RDWR) = 6
write(6, "ZMQbg/0", 7)               = 7
close(6)                             = 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
epoll_create1(EPOLL_CLOEXEC)         = 7
epoll_ctl(7, EPOLL_CTL_ADD, 6, {0, {u32=274703472, u64=93918324565104}}) = 0
epoll_ctl(7, EPOLL_CTL_MOD, 6, {EPOLLIN, {u32=274703472, u64=93918324565104}}) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -
1, 0) = 0x7fd9d440e000
mprotect(0x7fd9d440f000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7fd9d4c0db70,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLON
```

```
E_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
parent_tidptr=0x7fd9d4c0e9d0, tls=0x7fd9d4c0e700, child_tidptr=0x7fd9d4c0e9d0) = 8957
openat(AT_FDCWD, "/proc/self/task/8957/comm", O_RDWR) = 8
write(8, "ZMQbg/1", 7)              = 7
close(8)                            = 0
eventfd2(0, EFD_CLOEXEC)            = 8
fcntl(8, F_GETFL)                   = 0x2 (flags O_RDWR)
fcntl(8, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
fcntl(8, F_GETFL)                   = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(8, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 1), ...}) = 0
write(1, "command:", 8command:)             = 8
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 1), ...}) = 0
read(0, create 5
"create 5\n", 1024)             = 9
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
socket(AF_NETLINK, SOCK_RAW|SOCK_CLOEXEC, NETLINK_ROUTE) = 9
bind(9, {sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000}, 12) = 0
getsockname(9, {sa_family=AF_NETLINK, nl_pid=8955, nl_groups=00000000}, [12]) = 0
sendto(9, {{len=20, type=RTM_GETLINK, flags=NLM_F_REQUEST|NLM_F_DUMP,
seq=1640791805, pid=0}, {ifi_family=AF_UNSPEC, ...}}, 20, 0, {sa_family=AF_NETLINK,
nl_pid=0, nl_groups=00000000}, 12) = 20
recvmsg(9, {msg_name={sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000},
msg_namelen=12, msg_iov=[{iov_base=[{{len=1296, type=RTM_NEWLINK,
flags=NLM_F_MULTI, seq=1640791805, pid=8955}, {ifi_family=AF_UNSPEC,
ifi_type=ARPHRD_LOOPBACK, ifi_index=if_nametoindex("lo"),
ifi_flags=IFF_UP|IFF_LOOPBACK|IFF_RUNNING|0x10000, ifi_change=0}, [{{nla_len=7,
nla_type=IFLA_IFNAME}, "lo"}, {{nla_len=8, nla_type=IFLA_TXQLEN}, 1000}, {{nla_len=5,
nla_type=IFLA_OPERSTATE}, 0}, {{nla_len=5, nla_type=IFLA_LINKMODE}, 0},
{{nla_len=8, nla_type=IFLA_MTU}, 65536}, {{nla_len=8, nla_type=IFLA_GROUP}, 0},
{{nla_len=8, nla_type=IFLA_PROMISCUITY}, 0}, {{nla_len=8,
nla_type=IFLA_NUM_TX_QUEUES}, 1}, {{nla_len=8, nla_type=IFLA_GSO_MAX_SEGS},
65535}, {{nla_len=8, nla_type=IFLA_GSO_MAX_SIZE}, 65536}, {{nla_len=8,
nla_type=IFLA_NUM_RX_QUEUES}, 1}, {{nla_len=5, nla_type=IFLA_CARRIER}, 1},
{{nla_len=12, nla_type=IFLA_QDISC}, "noqueue"}, {{nla_len=8,
nla_type=IFLA_CARRIER_CHANGES}, 0}, {{nla_len=5, nla_type=IFLA_PROTO_DOWN},
0}, {{nla_len=8, nla_type=0x2f /* IFLA_??? */}, "\x00\x00\x00\x00"}, {{nla_len=8,
nla_type=0x30 /* IFLA_??? */}, "\x00\x00\x00\x00"}, {{nla_len=36, nla_type=IFLA_MAP},
{mem_start=0, mem_end=0, base_addr=0, irq=0, dma=0, port=0}}, {{nla_len=10,
nla_type=IFLA_ADDRESS}, "\x00\x00\x00\x00\x00\x00"}, {{nla_len=10,
nla_type=IFLA_BROADCAST}, "\x00\x00\x00\x00\x00\x00"}, {{nla_len=196,
nla_type=IFLA_STATS64}, {rx_packets=13785, tx_packets=13785, rx_bytes=831821,
tx_bytes=831821, rx_errors=0, tx_errors=0, rx_dropped=0, tx_dropped=0, multicast=0,
collisions=0, rx_length_errors=0, rx_over_errors=0, rx_crc_errors=0, rx_frame_errors=0,
rx_fifo_errors=0, rx_missed_errors=0, tx_aborted_errors=0, tx_carrier_errors=0, tx_fifo_errors=0,
tx_heartbeat_errors=0, tx_window_errors=0, rx_compressed=0, tx_compressed=0,
rx_nohandler=0}}, {{nla_len=100, nla_type=IFLA_STATS}, {rx_packets=13785,
tx_packets=13785, rx_bytes=831821, tx_bytes=831821, rx_errors=0, tx_errors=0, rx_dropped=0,
tx_dropped=0, multicast=0, collisions=0, rx_length_errors=0, rx_over_errors=0, rx_crc_errors=0,
rx_frame_errors=0, rx_fifo_errors=0, rx_missed_errors=0, tx_aborted_errors=0,
tx_carrier_errors=0, tx_fifo_errors=0, tx_heartbeat_errors=0, tx_window_errors=0,
rx_compressed=0, tx_compressed=0, rx_nohandler=0}}, {{nla_len=12, nla_type=IFLA_XDP},
{{nla_len=5, nla_type=IFLA_XDP_ATTACHED}, 0}}, {{nla_len=756,
nla_type=IFLA_AF_SPEC},
"\x84\x00\x02\x00\x80\x00\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00
\x00\x00\x01\x00\x00\x00\x01\x00\x00\x00"...}]}, {{len=1304, type=RTM_NEWLINK,
flags=NLM_F_MULTI, seq=1640791805, pid=8955}, {ifi_family=AF_UNSPEC,
ifi_type=ARPHRD_ETHER, ifi_index=if_nametoindex("eno1"),
ifi_flags=IFF_UP|IFF_BROADCAST|IFF_MULTICAST, ifi_change=0}, [{{nla_len=9,
nla_type=IFLA_IFNAME}, "eno1"}, {{nla_len=8, nla_type=IFLA_TXQLEN}, 1000},
{{nla_len=5, nla_type=IFLA_OPERSTATE}, 2}, {{nla_len=5, nla_type=IFLA_LINKMODE},
0}, {{nla_len=8, nla_type=IFLA_MTU}, 1500}, {{nla_len=8, nla_type=IFLA_GROUP}, 0},
{{nla_len=8, nla_type=IFLA_PROMISCUITY}, 0}, {{nla_len=8,
nla_type=IFLA_NUM_TX_QUEUES}, 1}, {{nla_len=8, nla_type=IFLA_GSO_MAX_SEGS},
65535}, {{nla_len=8, nla_type=IFLA_GSO_MAX_SIZE}, 65536}, {{nla_len=8,
nla_type=IFLA_NUM_RX_QUEUES}, 1}, {{nla_len=5, nla_type=IFLA_CARRIER}, 0},
{{nla_len=13, nla_type=IFLA_QDISC}, "fq_codel"}, {{nla_len=8,
nla_type=IFLA_CARRIER_CHANGES}, 1}, {{nla_len=5, nla_type=IFLA_PROTO_DOWN},
0}, {{nla_len=8, nla_type=0x2f /* IFLA_??? */}, "\x00\x00\x00\x00"}, {{nla_len=8,
nla_type=0x30 /* IFLA_??? */}, "\x01\x00\x00\x00"}, {{nla_len=36, nla_type=IFLA_MAP},
{mem_start=0, mem_end=0, base_addr=0, irq=0, dma=0, port=0}}, {{nla_len=10,
nla_type=IFLA_ADDRESS}, "\xb0\x0c\xd1\xea\x46\x8f"}, {{nla_len=10,
nla_type=IFLA_BROADCAST}, "\xff\xff\xff\xff\xff\xff"}, {{nla_len=196,
nla_type=IFLA_STATS64}, {rx_packets=0, tx_packets=0, rx_bytes=0, tx_bytes=0, rx_errors=0,
tx_errors=0, rx_dropped=0, tx_dropped=0, multicast=0, collisions=0, rx_length_errors=0,
rx_over_errors=0, rx_crc_errors=0, rx_frame_errors=0, rx_fifo_errors=0, rx_missed_errors=0,
tx_aborted_errors=0, tx_carrier_errors=0, tx_fifo_errors=0, tx_heartbeat_errors=0,
tx_window_errors=0, rx_compressed=0, tx_compressed=0, rx_nohandler=0}}, {{nla_len=100,
nla_type=IFLA_STATS}, {rx_packets=0, tx_packets=0, rx_bytes=0, tx_bytes=0, rx_errors=0,
tx_errors=0, rx_dropped=0, tx_dropped=0, multicast=0, collisions=0, rx_length_errors=0,
rx_over_errors=0, rx_crc_errors=0, rx_frame_errors=0, rx_fifo_errors=0, rx_missed_errors=0,
tx_aborted_errors=0, tx_carrier_errors=0, tx_fifo_errors=0, tx_heartbeat_errors=0,
tx_window_errors=0, rx_compressed=0, tx_compressed=0, rx_nohandler=0}}, {{nla_len=12,
nla_type=IFLA_XDP}, {{nla_len=5, nla_type=IFLA_XDP_ATTACHED}, 0}}, {{nla_len=756,
nla_type=IFLA_AF_SPEC},
"\x84\x00\x02\x00\x80\x00\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00
\x00\x00\x01\x00\x00\x00\x01\x00\x00\x00"...}]}], iov_len=4096}], msg_iovlen=1,
msg_controllen=0, msg_flags=0}, 0) = 2600
recvmsg(9, {msg_name={sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000},
msg_namelen=12, msg_iov=[{iov_base={{len=1296, type=RTM_NEWLINK,
flags=NLM_F_MULTI, seq=1640791805, pid=8955}, {ifi_family=AF_UNSPEC,
ifi_type=ARPHRD_ETHER, ifi_index=if_nametoindex("wlo1"),
ifi_flags=IFF_UP|IFF_BROADCAST|IFF_RUNNING|IFF_MULTICAST|0x10000,
ifi_change=0}, [{{nla_len=9, nla_type=IFLA_IFNAME}, "wlo1"}, {{nla_len=8,
nla_type=IFLA_TXQLEN}, 1000}, {{nla_len=5, nla_type=IFLA_OPERSTATE}, 6},
{{nla_len=5, nla_type=IFLA_LINKMODE}, 1}, {{nla_len=8, nla_type=IFLA_MTU}, 1500},
{{nla_len=8, nla_type=IFLA_GROUP}, 0}, {{nla_len=8, nla_type=IFLA_PROMISCUITY}, 0},
{{nla_len=8, nla_type=IFLA_NUM_TX_QUEUES}, 4}, {{nla_len=8,
nla_type=IFLA_GSO_MAX_SEGS}, 65535}, {{nla_len=8, nla_type=IFLA_GSO_MAX_SIZE},
65536}, {{nla_len=8, nla_type=IFLA_NUM_RX_QUEUES}, 1}, {{nla_len=5,
nla_type=IFLA_CARRIER}, 1}, {{nla_len=7, nla_type=IFLA_QDISC}, "mq"}, {{nla_len=8,
nla_type=IFLA_CARRIER_CHANGES}, 2}, {{nla_len=5, nla_type=IFLA_PROTO_DOWN},
0}, {{nla_len=8, nla_type=0x2f /* IFLA_??? */}, "\x01\x00\x00\x00"}, {{nla_len=8,
nla_type=0x30 /* IFLA_??? */}, "\x01\x00\x00\x00"}, {{nla_len=36, nla_type=IFLA_MAP},
{mem_start=0, mem_end=0, base_addr=0, irq=0, dma=0, port=0}}, {{nla_len=10,
nla_type=IFLA_ADDRESS}, "\xd4\x3b\x04\x3a\xa7\x82"}, {{nla_len=10,
nla_type=IFLA_BROADCAST}, "\xff\xff\xff\xff\xff\xff"}, {{nla_len=196,
nla_type=IFLA_STATS64}, {rx_packets=30530, tx_packets=19500, rx_bytes=25906404,
tx_bytes=4949527, rx_errors=0, tx_errors=0, rx_dropped=0, tx_dropped=0, multicast=0,
collisions=0, rx_length_errors=0, rx_over_errors=0, rx_crc_errors=0, rx_frame_errors=0,
rx_fifo_errors=0, rx_missed_errors=0, tx_aborted_errors=0, tx_carrier_errors=0, tx_fifo_errors=0,
tx_heartbeat_errors=0, tx_window_errors=0, rx_compressed=0, tx_compressed=0,
rx_nohandler=0}}, {{nla_len=100, nla_type=IFLA_STATS}, {rx_packets=30530,
tx_packets=19500, rx_bytes=25906404, tx_bytes=4949527, rx_errors=0, tx_errors=0,
rx_dropped=0, tx_dropped=0, multicast=0, collisions=0, rx_length_errors=0, rx_over_errors=0,
rx_crc_errors=0, rx_frame_errors=0, rx_fifo_errors=0, rx_missed_errors=0, tx_aborted_errors=0,
tx_carrier_errors=0, tx_fifo_errors=0, tx_heartbeat_errors=0, tx_window_errors=0,
rx_compressed=0, tx_compressed=0, rx_nohandler=0}}, {{nla_len=12, nla_type=IFLA_XDP},
{{nla_len=5, nla_type=IFLA_XDP_ATTACHED}, 0}}, {{nla_len=756,
nla_type=IFLA_AF_SPEC},
"\x84\x00\x02\x00\x80\x00\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00
\x00\x00\x01\x00\x00\x00\x01\x00\x00\x00"...}]}, iov_len=4096}], msg_iovlen=1,
msg_controllen=0, msg_flags=0}, 0) = 1296
recvmsg(9, {msg_name={sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000},
msg_namelen=12, msg_iov=[{iov_base={{len=20, type=NLMSG_DONE,
flags=NLM_F_MULTI, seq=1640791805, pid=8955}, 0}, iov_len=4096}], msg_iovlen=1,
msg_controllen=0, msg_flags=0}, 0) = 20
sendto(9, {{len=20, type=RTM_GETADDR, flags=NLM_F_REQUEST|NLM_F_DUMP,
seq=1640791806, pid=0}, {ifa_family=AF_UNSPEC, ...}}, 20, 0, {sa_family=AF_NETLINK,
nl_pid=0, nl_groups=00000000}, 12) = 20
recvmsg(9, {msg_name={sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000},
msg_namelen=12, msg_iov=[{iov_base=[{{len=76, type=RTM_NEWADDR,
flags=NLM_F_MULTI, seq=1640791806, pid=8955}, {ifa_family=AF_INET, ifa_prefixlen=8,
ifa_flags=IFA_F_PERMANENT, ifa_scope=RT_SCOPE_HOST,
ifa_index=if_nametoindex("lo")}, [{{nla_len=8, nla_type=IFA_ADDRESS}, 127.0.0.1},
{{nla_len=8, nla_type=IFA_LOCAL}, 127.0.0.1}, {{nla_len=7, nla_type=IFA_LABEL}, "lo"},
{{nla_len=8, nla_type=IFA_FLAGS}, IFA_F_PERMANENT}, {{nla_len=20,
nla_type=IFA_CACHEINFO}, {ifa_prefered=4294967295, ifa_valid=4294967295, cstamp=224,
tstamp=224}}]}, {{len=88, type=RTM_NEWADDR, flags=NLM_F_MULTI, seq=1640791806,
pid=8955}, {ifa_family=AF_INET, ifa_prefixlen=24, ifa_flags=0,
ifa_scope=RT_SCOPE_UNIVERSE, ifa_index=if_nametoindex("wlo1")}, [{{nla_len=8,
nla_type=IFA_ADDRESS}, 192.168.1.64}, {{nla_len=8, nla_type=IFA_LOCAL},
192.168.1.64}, {{nla_len=8, nla_type=IFA_BROADCAST}, 192.168.1.255}, {{nla_len=9,
nla_type=IFA_LABEL}, "wlo1"}, {{nla_len=8, nla_type=IFA_FLAGS},
IFA_F_NOPREFIXROUTE}, {{nla_len=20, nla_type=IFA_CACHEINFO},
{ifa_prefered=79771, ifa_valid=79771, cstamp=1205, tstamp=660738}}]}], iov_len=4096}],
msg_iovlen=1, msg_controllen=0, msg_flags=0}, 0) = 164
recvmsg(9, {msg_name={sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000},
msg_namelen=12, msg_iov=[{iov_base=[{{len=72, type=RTM_NEWADDR,
flags=NLM_F_MULTI, seq=1640791806, pid=8955}, {ifa_family=AF_INET6,
ifa_prefixlen=128, ifa_flags=IFA_F_PERMANENT, ifa_scope=RT_SCOPE_HOST,
ifa_index=if_nametoindex("lo")}, [{{nla_len=20, nla_type=IFA_ADDRESS}, ::1}, {{nla_len=20,
nla_type=IFA_CACHEINFO}, {ifa_prefered=4294967295, ifa_valid=4294967295, cstamp=224,
tstamp=224}}, {{nla_len=8, nla_type=IFA_FLAGS}, IFA_F_PERMANENT}]}, {{len=72,
type=RTM_NEWADDR, flags=NLM_F_MULTI, seq=1640791806, pid=8955},
{ifa_family=AF_INET6, ifa_prefixlen=128, ifa_flags=0, ifa_scope=RT_SCOPE_UNIVERSE,
ifa_index=if_nametoindex("wlo1")}, [{{nla_len=20, nla_type=IFA_ADDRESS},
2a00:1370:8137:6796:1484:6013:4b01:bc7}, {{nla_len=20, nla_type=IFA_CACHEINFO},
{ifa_prefered=577, ifa_valid=577, cstamp=1361, tstamp=661844}}, {{nla_len=8,
nla_type=IFA_FLAGS}, IFA_F_NOPREFIXROUTE}]}, {{len=72, type=RTM_NEWADDR,
flags=NLM_F_MULTI, seq=1640791806, pid=8955}, {ifa_family=AF_INET6, ifa_prefixlen=64,
ifa_flags=IFA_F_SECONDARY, ifa_scope=RT_SCOPE_UNIVERSE,
ifa_index=if_nametoindex("wlo1")}, [{{nla_len=20, nla_type=IFA_ADDRESS},
2a00:1370:8137:6796:1dfd:c02b:dba8:950c}, {{nla_len=20, nla_type=IFA_CACHEINFO},
{ifa_prefered=311, ifa_valid=311, cstamp=1202, tstamp=661844}}, {{nla_len=8,
nla_type=IFA_FLAGS}, IFA_F_SECONDARY}]}, {{len=72, type=RTM_NEWADDR,
flags=NLM_F_MULTI, seq=1640791806, pid=8955}, {ifa_family=AF_INET6, ifa_prefixlen=64,
ifa_flags=0, ifa_scope=RT_SCOPE_UNIVERSE, ifa_index=if_nametoindex("wlo1")},
[{{nla_len=20, nla_type=IFA_ADDRESS}, 2a00:1370:8137:6796:ad39:a00d:5808:4400},
{{nla_len=20, nla_type=IFA_CACHEINFO}, {ifa_prefered=311, ifa_valid=311, cstamp=1202,
tstamp=661844}}, {{nla_len=8, nla_type=IFA_FLAGS},
IFA_F_MANAGETEMPADDR|IFA_F_NOPREFIXROUTE}]}, {{len=72,
type=RTM_NEWADDR, flags=NLM_F_MULTI, seq=1640791806, pid=8955},
{ifa_family=AF_INET6, ifa_prefixlen=64, ifa_flags=IFA_F_SECONDARY,
ifa_scope=RT_SCOPE_UNIVERSE, ifa_index=if_nametoindex("wlo1")}, [{{nla_len=20,
nla_type=IFA_ADDRESS}, fd60:e320:1618:1:1dfd:c02b:dba8:950c}, {{nla_len=20,
nla_type=IFA_CACHEINFO}, {ifa_prefered=311, ifa_valid=311, cstamp=1202,
tstamp=661844}}, {{nla_len=8, nla_type=IFA_FLAGS}, IFA_F_SECONDARY}]}, {{len=72,
type=RTM_NEWADDR, flags=NLM_F_MULTI, seq=1640791806, pid=8955},
{ifa_family=AF_INET6, ifa_prefixlen=64, ifa_flags=0, ifa_scope=RT_SCOPE_UNIVERSE,
ifa_index=if_nametoindex("wlo1")}, [{{nla_len=20, nla_type=IFA_ADDRESS},
fd60:e320:1618:1:a2d2:41d3:2efe:aa5e}, {{nla_len=20, nla_type=IFA_CACHEINFO},
{ifa_prefered=311, ifa_valid=311, cstamp=1202, tstamp=661844}}, {{nla_len=8,
nla_type=IFA_FLAGS}, IFA_F_MANAGETEMPADDR|IFA_F_NOPREFIXROUTE}]},
{{len=72, type=RTM_NEWADDR, flags=NLM_F_MULTI, seq=1640791806, pid=8955},
{ifa_family=AF_INET6, ifa_prefixlen=64, ifa_flags=IFA_F_PERMANENT,
ifa_scope=RT_SCOPE_LINK, ifa_index=if_nametoindex("wlo1")}, [{{nla_len=20,
nla_type=IFA_ADDRESS}, fe80::e44:a50:7435:c1b8}, {{nla_len=20,
nla_type=IFA_CACHEINFO}, {ifa_prefered=4294967295, ifa_valid=4294967295, cstamp=1000,
tstamp=661844}}, {{nla_len=8, nla_type=IFA_FLAGS},
IFA_F_PERMANENT|IFA_F_NOPREFIXROUTE}]}], iov_len=4096}], msg_iovlen=1,
msg_controllen=0, msg_flags=0}, 0) = 504
```

```
recvmsg(9, {msg_name={sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000},
msg_namelen=12, msg_iov=[{iov_base={{len=20, type=NLMSG_DONE,
flags=NLM_F_MULTI, seq=1640791806, pid=8955}, 0}, iov_len=4096}], msg_iovlen=1,
msg_controllen=0, msg_flags=0}, 0) = 20
close(9)                              = 0
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(3005), sin_addr=inet_addr("127.0.0.1")}, 16) = 0
listen(9, 100)                        = 0
getsockname(9, {sa_family=AF_INET, sin_port=htons(3005), sin_addr=inet_addr("127.0.0.1")},
[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=0x7fd9d73ece50) = 8998
futex(0x7fd9d69a1918, FUTEX_WAKE_PRIVATE, 1) = 1
poll([{fd=8, events=POLLIN}], 1, -1)   = 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, -1)   = 1 ([{fd=8, revents=POLLIN}])
read(8, "\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(1, "OK: 8998\n", 9OK: 8998
)             = 9
write(1, "command:", 8command:)             = 8
read(0, create 9
"create 9\n", 1024)          = 9
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, -1)    = 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, -1)    = 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(1, "OK: 9049\n", 9OK: 9049
)               = 9
write(1, "command:", 8command:)                 = 8
read(0, exec 9
"exec 9\n", 1024)               = 7
read(0, qwertyhjklkjhgfd
"qwertyhjklkjhgfd\n", 1024)     = 17
read(0, j
"j\n", 1024)                    = 2
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, -1)    = 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(1, "OK:9:7;11\n", 10OK:9:7;11
)             = 10
write(1, "command:", 8command:)                 = 8
```

Демонстрация работы программы

```
ann@ann:~/os/lab6$ ./main
command:create 686
OK: 6035
command:create 54
OK: 6063
command:ping 2
OK: 0
command:create 8
```

```
OK: 6184
command:create 4585
OK: 6192
command:create 455
OK: 6253
command:ping 54
OK: 1
command:kill 54
Ok: 54
command:ping 54
OK: 0
command:ping 2
OK: 0
command:ping 8
OK: 0
command:create 45
OK: 6562
command:exec 45
aaadorayy
dora
OK:45:3
command:exec 45
aaaaaaaaaaaaaaaaaaaaaaaaa
aa
OK:45:0;1;2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;20;21;22;23
command:exec 45
yhtjrktrjkhjfk
b
OK:45:-1
command:exec 45
yt
hgjfkd
OK:45:-1
command:exit
Tree was deleted
```

Вывод

Лабораторная работа была сложной (самая сложная лабораторная работа за все время обучения в МАИ), но очень интересной. Ведь в ней сразу можно применить знания, полученные в ходе выполнения предыдущих лабораторных работ, так как здесь и многопоточность, и межпроцессорное взаимодействие, основанное на очередях сообщений, и синхронизация потоков. А помимо всего этого также необходимо разобраться с дополнительной библиотекой (zmq), которую пришлось дополнительно устанавливать. Единственное, что мне не очень понравилось – наличие небольшого количества документации по этой библиотеки и более сложная отладка программ. Заодно я вспомнила функции работы со строками, а именно поиск подстроки.