Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №6 по курсу «Операционные системы»

Студент: Чекменев В.А.

Группа: М80-207Б-20

Преподаватель: Миронов Е.С.

Оценка: ____________________

Дата:

Москва, 2022

Содержание

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: 27. Топология — связный список. Тип вычислительной команды — поиск подстроки в строке. Тип проверки узлов на доступность — пинг всех узлов (заменено на пинг определенного узла).

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).
- Дочерние узлы проверяют, может ли быть команда выполнена в данном узле, если нет, то команда пересылается в следующий узел, из которого возвращается некоторое сообщение(об успехе или об ошибке), которое потом пересылается обратно по списку.
- Для корректной проверки на доступность узлов, используется список, эмулирующий поведение узлов в данной топологии.

Код программы

client.cpp:

```
#include <iostream>
#include <zmq.h>
#include <unistd.h>
#include <fcntl.h>
#include <zmq.hpp>
#include <unistd.h>
#include <string>
#include "server.hpp"

using namespace std;

int TopVertex = 0;

void create(int port, zmq::socket_t& socket)
{
    int NodeId;
    cin >> NodeId;
    if (TopVertex == 0) {
        int NodePid = fork();
        if (NodePid == -1) {
            cout << "Error: fail fork()\n";
            return;
        } else if (NodePid == 0) { // in child, executing the server program
            execl("./server", "server", to_string(NodeId).c_str(),
to_string(port).c_str(), NULL);
        } else { // in parent, print OK message
            cout << "Ok: " << NodePid << "\n";
        }
        TopVertex = 1;
    } else {
        sendMessage(socket, "create " + to_string(NodeId));
        string Ans = receiveMessage(socket);
        cout << Ans;
    }
}

void exec(zmq::socket_t& socket)
{
    int NodeId;
    string text, pattern;
    cin >> NodeId;
    if (TopVertex == 0) { // we need to create this node
        cout << "Error: " + std::to_string(NodeId) + ": Not found\n";
        return;
    }
    cin >> text >> pattern;
    sendMessage(socket, "exec " + to_string(NodeId) + " " + text + " " +
pattern);
    string Ans = receiveMessage(socket);
    cout << Ans;
}

void pingid(zmq::socket_t& socket)
{
    int NodeId;
    cin >> NodeId;
    if (TopVertex == 0) {
        cout << "There is no any node here yet\n";
        return;
    }
    sendMessage(socket, "pingid " + to_string(NodeId));
    string ans = receiveMessage(socket);
    cout << ans;
}

int main()
{
    zmq::context_t context(1);
    zmq::socket_t Rule_socket(context, ZMQ_REQ);
    int port = bindSocket(Rule_socket);
    string s, text, pattern;
    cout << "> ";
    while (true) {
        cin >> s;
        if (s == "create") {
            create(port, Rule_socket);
        } else if (s == "exec") {
            exec(Rule_socket);
        } else if (s == "pingid") {
            pingid(Rule_socket);
        } else if (s == "exit") {
            return 0;
        } else {
            cout << "Error Command\n";
        }
        cout << "> ";
    }
    return 0;
}
```

server.cpp:

```
#include <iostream>
#include <string>
#include <unistd.h>
#include <fcntl.h>
#include <ctime>
#include "server.hpp"

int main(int argc, char *argv[])
{
    int CurNodeId = atoi(argv[1]);
    int ParentPort = atoi(argv[2]);

    zmq::context_t context(2);
    zmq::socket_t ChildSocket(context, ZMQ_REP);
    ChildSocket.connect(getPortName(ParentPort));

    zmq::socket_t req_socket(context, ZMQ_REQ);

    int ChildPort = bindSocket(req_socket);

    int ChildTopVertex = 0;

    std::string cmd;
    while(true) {
        std::string request = receiveMessage(ChildSocket); //
receive message from parent (client)

        std::istringstream cmdStream(request); // thread to read
data from accepted string

        cmdStream >> cmd;

        if (cmd == "create") {

            int NodeId;

            cmdStream >> NodeId;

            if (CurNodeId == NodeId) { // if node has already been
created

                sendMessage(ChildSocket, "Error: Already exists\
n");

            } else {

                if (ChildTopVertex == 0) {

                    int NodePid = fork();

                    if (NodePid == -1) {

                        sendMessage(ChildSocket, "Error: fork()
fail\n");

                    } else if (NodePid == 0) {

                        execl("./server", "server",
std::to_string(NodeId).c_str(), std::to_string(ChildPort).c_str(),
NULL);

                    } else {

                        sendMessage(ChildSocket, "Ok: " +
std::to_string(NodePid) + "\n");

                    }

                    ChildTopVertex = 1;

                } else {
                    sendMessage(req_socket, "create " +
std::to_string(NodeId));

                    std::string AnsFromChild =
receiveMessage(req_socket);

                    sendMessage(ChildSocket, AnsFromChild);

                }



            }

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

            int NodeId;

            std::string Text, Pattern;

            cmdStream >> NodeId;

            cmdStream >> Text >> Pattern;

            if (NodeId == CurNodeId) {

                int pos = 0;

                std::vector<size_t> entries = KMP(Pattern, Text);

                std::string Ans;

                if (entries.size() != 0) {

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

                        if (i == 0) Ans +=
std::to_string(entries[i]);

                        else  Ans += " " +
std::to_string(entries[i]);

                    }

                } else {

                    Ans += "-1";
                }

                sendMessage(ChildSocket, "Ok: " +
std::to_string(NodeId) + ": [" + Ans + "]\n"); // send result up

            } else {

                if (ChildTopVertex == 0) { // there is no node
with this nodeid

                    sendMessage(ChildSocket, "Error: " +
std::to_string(NodeId) + ": Not found\n");

                } else { // send message to next node

                    sendMessage(req_socket, "exec " +
std::to_string(NodeId) + " " + Text + " " + Pattern);

                    std::string Ans =
receiveMessage(req_socket); // receive and send message to next
node

                    sendMessage(ChildSocket, Ans);

                }

            }

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

            int NodeId;

            cmdStream >> NodeId;

            if (NodeId == CurNodeId) {

                sendMessage(ChildSocket, "Ok: 1\n");

            } else {

                if (ChildTopVertex == 0) {

                    sendMessage(ChildSocket, "Not found\n");

                } else {

                    sendMessage(req_socket, "pingid " +
std::to_string(NodeId));

                    std::string ans = receiveMessage(req_socket);

                    if (ans == "Ok: 1\n") {

                        sendMessage(ChildSocket, ans);

                    } else {

                        sendMessage(ChildSocket, "Ok: 0\n");

                    }

                }

            }

        }

    }

    return 0;

}
```

server.hpp:

```
#pragma once

#include <cstdlib>
#include <string>
#include <vector>
#include <unistd.h>
#include <zmq.hpp>

std::vector<size_t> prefix_function(std::string s)
{
    size_t n = s.length();
    // в i-м элементе (его индекс i-1) количество
    // совпавших символов в начале и конце для подстроки длины i.
    std::vector<size_t> pi(n);
                // p[0]=0 всегда, p[1]=1, если начинается с двух
одинаковых
```

```
    for (size_t i=1; i<n; ++i)
    {
       // ищем, какой префикс-суффикс можно расширить
        size_t j = pi[i-1]; // длина предыдущего префикса-
суффикса, возможно нулевая
        while ((j > 0) && (s[i] != s[j])) // этот нельзя
расширить,
            j = pi[j-1];   // берем длину меньшего префикса-
суффикса

        if (s[i] == s[j])
            ++j;  // расширяем найденный (возможно пустой)
префикс-суффикс
        pi[i] = j;
     }
     return pi;
}

std::vector<size_t> KMP(std::string pattern, std::string text)
{
    int n = text.length();
    int m = pattern.length();

    std::vector<size_t> Lps = prefix_function(pattern); //
применяем префиекс функцию
    std::vector<size_t> out; // вектор с индексами вхождений

    int i = 0, j = 0;
    while (i < n) {
        if (pattern[j] == text[i]) {
            i++; j++;
        } // если совпало, продолжаем
        if (j == m) { // если j==m это подтверждает то, что мы
нашли образец в тексте
            out.push_back(i - m); // добавляем этот индекс минус
длина образца в вектор out
            j = Lps[j - 1]; // обновляем j как префикс последнего
совпавшего символа
        } else if (i < n && pattern[j] != text[i]) {  // если не
совпало
            if (j == 0) {
                i++; // если j становится равным нулю, делаем
инкремент индекса i
            } else {
                j = Lps[j - 1]; // обновляем j как префикс
последнего совпавшего символа
            }
        }
    }
    return out;
}

// send message to the particular socket
bool sendMessage(zmq::socket_t &socket, const std::string
&message_string)
{
    // message size init
    zmq::message_t message(message_string.size());
    // message content init
    memcpy(message.data(), message_string.c_str(),
message_string.size());
    return socket.send(message);
}

std::string receiveMessage(zmq::socket_t &socket)
{
    zmq::message_t message;
    int recResult;
    // receiving message from socket
    try {
        recResult = (int)socket.recv(&message);
        if (recResult < 0) {
            perror("socket.recv()");
            exit(1);
        }
    }
    catch (...) {
        recResult = 0;
    }
    // transform to string
    std::string recieved_message((char *)message.data(),
message.size());
    if (recieved_message.empty() || !recResult) {
        return "Error: Node is not available";
    }
    return recieved_message;
}

std::string getPortName(int port)
{
    return "tcp://127.0.0.1:" + std::to_string(port);
}

int bindSocket(zmq::socket_t &socket)
{
    int port = 8080;
    // create endpoint and bind it to the socket
    while (true) {
        try {
            socket.bind(getPortName(port)); // создаёт сокет
            break;
        }
        catch (...) { // в случае неудачи используем другой сокет
            port++;
        }
    }
    return port;
}
```

Использование утилиты strace

```
[suraba04@asusx512fl third]$ strace ./client
execve("./client", ["./client"], 0x7ffc04e97fc0 /* 72 vars */) = 0
brk(NULL)                               = 0x55723e2c0000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc536b3470) = -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
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=230043, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 230043, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f741cb87000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/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 \200\1\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=743320, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7f741cb85000
mmap(NULL, 745560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f741cace000
mmap(0x7f741cae6000, 471040, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x18000) = 0x7f741cae6000
mmap(0x7f741cb59000, 143360, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x8b000) = 0x7f741cb59000
mmap(0x7f741cb7c000, 36864, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0xad000) = 0x7f741cb7c000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\210\255N\377\201\240\fhJ\277\340\370c\350t4"...,
36, 800) = 36
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=18844016, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 2250816, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f741c8a8000
mprotect(0x7f741c941000, 1556480, PROT_NONE) = 0
mmap(0x7f741c941000, 1085440, PROT_READ|PROT_EXEC, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x99000) = 0x7f741c941000
mmap(0x7f741ca4a000, 466944, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1a2000) = 0x7f741ca4a000
mmap(0x7f741cabd000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x214000) = 0x7f741cabd000
mmap(0x7f741cacb000, 10304, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f741cacb000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/libm.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=1061880, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 946752, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f741c7c0000
mmap(0x7f741c7d0000, 499712, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x10000) = 0x7f741c7d0000
mmap(0x7f741c84a000, 376832, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x8a000) = 0x7f741c84a000
mmap(0x7f741c8a6000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0xe5000) = 0x7f741c8a6000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=478272, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 107240, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f741c7a5000
mprotect(0x7f741c7a8000, 90112, PROT_NONE) = 0
mmap(0x7f741c7a8000, 73728, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x3000) = 0x7f741c7a8000
mmap(0x7f741c7ba000, 12288, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x15000) = 0x7f741c7ba000
mmap(0x7f741c7be000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x18000) = 0x7f741c7be000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/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\320\324\2\0\0\0\0\0"..., 832) = 832
```

```
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0@\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 80, 848) = 80
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\205vn\235\204X\261n\234|\346\340|q,\2"..., 68,
928) = 68
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2463384, ...}, AT_EMPTY_PATH) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
mmap(NULL, 2136752, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f741c59b000
mprotect(0x7f741c5c7000, 1880064, PROT_NONE) = 0
mmap(0x7f741c5c7000, 1531904, PROT_READ|PROT_EXEC, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x2c000) = 0x7f741c5c7000
mmap(0x7f741c73d000, 344064, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1a2000) = 0x7f741c73d000
mmap(0x7f741c792000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1f6000) = 0x7f741c792000
mmap(0x7f741c798000, 51888, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f741c798000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/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 \320\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=362968, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 365576, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f741c541000
mmap(0x7f741c54e000, 233472, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0xd000) = 0x7f741c54e000
mmap(0x7f741c587000, 73728, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x46000) = 0x7f741c587000
mmap(0x7f741c599000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x57000) = 0x7f741c599000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/libpgm-5.3.so.0", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0 @\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=340376, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7f741c53f000
mmap(NULL, 328976, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f741c4ee000
mprotect(0x7f741c4f2000, 290816, PROT_NONE) = 0
mmap(0x7f741c4f2000, 167936, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x4000) = 0x7f741c4f2000
mmap(0x7f741c51b000, 118784, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x2d000) = 0x7f741c51b000
mmap(0x7f741c539000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x4a000) = 0x7f741c539000
mmap(0x7f741c53b000, 13584, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f741c53b000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/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\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=16200, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 16400, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f741c4e9000
mmap(0x7f741c4ea000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1000) = 0x7f741c4ea000
mmap(0x7f741c4eb000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x2000) = 0x7f741c4eb000
mmap(0x7f741c4ec000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x2000) = 0x7f741c4ec000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/libpthread.so.0", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=16488, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 16400, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f741c4e4000
mmap(0x7f741c4e5000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1000) = 0x7f741c4e5000
mmap(0x7f741c4e6000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x2000) = 0x7f741c4e6000
mmap(0x7f741c4e7000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x2000) = 0x7f741c4e7000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7f741c4e2000
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -
1, 0) = 0x7f741c4df000
arch_prctl(ARCH_SET_FS, 0x7f741c4df980) = 0
set_tid_address(0x7f741c4dfc50)         = 9677
set_robust_list(0x7f741c4dfc60, 24)     = 0
rseq(0x7f741c4e0320, 0x20, 0, 0x53053053) = 0
mprotect(0x7f741c792000, 12288, PROT_READ) = 0
mprotect(0x7f741c4e7000, 4096, PROT_READ) = 0
mprotect(0x7f741c4ec000, 4096, PROT_READ) = 0
mprotect(0x7f741c8a6000, 4096, PROT_READ) = 0
mprotect(0x7f741c539000, 4096, PROT_READ) = 0
mprotect(0x7f741c599000, 4096, PROT_READ) = 0
mprotect(0x7f741c7be000, 4096, PROT_READ) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7f741c4dd000
mprotect(0x7f741cabd000, 53248, PROT_READ) = 0
mprotect(0x7f741cb7c000, 32768, PROT_READ) = 0
mprotect(0x55723dd2c000, 4096, PROT_READ) = 0
mprotect(0x7f741cbf5000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY})
= 0
munmap(0x7f741cb87000, 230043)          = 0
getrandom("\xbe\xea\xb3\xc4\x2f\x3e\x79\x2a", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x55723e2c0000
brk(0x55723e2e1000)                     = 0x55723e2e1000
futex(0x7f741cacb6bc, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f741cacb6c8, FUTEX_WAKE_PRIVATE, 2147483647) = 0
openat(AT_FDCWD, "/sys/devices/system/cpu/online", O_RDONLY|O_CLOEXEC) = 3
read(3, "0-7\n", 1024)                  = 4
close(3)                                = 0
openat(AT_FDCWD, "/sys/devices/system/cpu", O_RDONLY|O_NONBLOCK|O_CLOEXEC|
O_DIRECTORY) = 3
newfstatat(3, "", {st_mode=S_IFDIR|0755, st_size=0, ...}, AT_EMPTY_PATH) = 0
getdents64(3, 0x55723e2d1ee0 /* 26 entries */, 32768) = 752
getdents64(3, 0x55723e2d1ee0 /* 0 entries */, 32768) = 0
close(3)                                = 0
getpid()                                = 9677
sched_getaffinity(9677, 128, [0, 1, 2, 3, 4, 5, 6, 7]) = 40
newfstatat(AT_FDCWD, "/etc/nsswitch.conf", {st_mode=S_IFREG|0644, st_size=391, ...}, 0) =
0
newfstatat(AT_FDCWD, "/", {st_mode=S_IFDIR|0755, st_size=4096, ...}, 0) = 0
openat(AT_FDCWD, "/etc/nsswitch.conf", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=391, ...}, AT_EMPTY_PATH) = 0
read(3, "# Name Service Switch configurat"..., 4096) = 391
read(3, "", 4096)                       = 0
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=391, ...}, AT_EMPTY_PATH) = 0
close(3)                                = 0
openat(AT_FDCWD, "/etc/protocols", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=3171, ...}, AT_EMPTY_PATH) = 0
lseek(3, 0, SEEK_SET)                   = 0
```

```
read(3, "# Full data: /usr/share/iana-etc"..., 4096) = 3171
close(3)                                = 0
eventfd2(0, EFD_CLOEXEC)                = 3
fcntl(3, F_GETFL)                       = 0x2 (flags O_RDWR)
fcntl(3, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
fcntl(3, F_GETFL)                       = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(3, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
getpid()                                = 9677
getpid()                                = 9677
getrandom("\x8d\xad\xf5\x6d\x7d\x05\x76\xed\x86\x38\x8f\xa5\xc2\x3b\xc3\x34", 16, 0) = 16
getrandom("\x05\x2f\x49\xb5\xe8\x90\x1a\x49\xb9\xe0\xb8\x4a\xb7\x2f\x5b\x64", 16, 0) = 16
eventfd2(0, EFD_CLOEXEC)                = 4
fcntl(4, F_GETFL)                       = 0x2 (flags O_RDWR)
fcntl(4, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
fcntl(4, F_GETFL)                       = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(4, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
getpid()                                = 9677
epoll_create1(EPOLL_CLOEXEC)            = 5
epoll_ctl(5, EPOLL_CTL_ADD, 4, {events=0, data={u32=1043147360,
u64=93949157780064}}) = 0
epoll_ctl(5, EPOLL_CTL_MOD, 4, {events=EPOLLIN, data={u32=1043147360,
u64=93949157780064}}) = 0
getpid()                                = 9677
rt_sigaction(SIGRT_1, {sa_handler=0x7f741c625940, sa_mask=[], sa_flags=SA_RESTORER|
SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f741c5dd560}, NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,
-1, 0) = 0x7f741bcdc000
mprotect(0x7f741bcdd000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone(child_stack=0x7f741c4dbcb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|
CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[9678],
tls=0x7f741c4dc640, child_tidptr=0x7f741c4dc910) = 9678
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
eventfd2(0, EFD_CLOEXEC)                = 6
fcntl(6, F_GETFL)                       = 0x2 (flags O_RDWR)
fcntl(6, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
fcntl(6, F_GETFL)                       = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(6, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
getpid()                                = 9677
```

```
epoll_create1(EPOLL_CLOEXEC)           = 7
epoll_ctl(7, EPOLL_CTL_ADD, 6, {events=0, data={u32=1043145888,
u64=93949157778592}}) = 0
epoll_ctl(7, EPOLL_CTL_MOD, 6, {events=EPOLLIN, data={u32=1043145888,
u64=93949157778592}}) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,
-1, 0) = 0x7f741b4db000
mprotect(0x7f741b4dc000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)  = 0
clone(child_stack=0x7f741bcdacb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|
CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[9679],
tls=0x7f741bcdb640, child_tidptr=0x7f741bcdb910) = 9679
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
eventfd2(0, EFD_CLOEXEC)               = 8
fcntl(8, F_GETFL)                      = 0x2 (flags O_RDWR)
fcntl(8, F_SETFL, O_RDWR|O_NONBLOCK)   = 0
fcntl(8, F_GETFL)                      = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(8, F_SETFL, O_RDWR|O_NONBLOCK)   = 0
getpid()                               = 9677
getpid()                               = 9677
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=9677, nl_groups=00000000}, [12]) = 0
sendto(9, [{nlmsg_len=20, nlmsg_type=RTM_GETLINK, nlmsg_flags=NLM_F_REQUEST|
NLM_F_DUMP, nlmsg_seq=1648280464, nlmsg_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=[[{nlmsg_len=1320, nlmsg_type=RTM_NEWLINK,
nlmsg_flags=NLM_F_MULTI, nlmsg_seq=1648280464, nlmsg_pid=9677},
{ifi_family=AF_UNSPEC, ifi_type=ARPHRD_LOOPBACK, ifi_index=if_nametoindex("lo"),
ifi_flags=IFF_UP|IFF_LOOPBACK|IFF_RUNNING|IFF_LOWER_UP, ifi_change=0},
[[{nla_len=7, nla_type=IFLA_IFNAME}, "lo"], [{nla_len=8, nla_type=IFLA_TXQLEN},
1000], [{nla_len=5, nla_type=IFLA_OPERSTATE}, 0], [{nla_len=5,
nla_type=IFLA_LINKMODE}, 0], [{nla_len=8, nla_type=IFLA_MTU}, 65536], [{nla_len=8,
nla_type=IFLA_MIN_MTU}, 0], [{nla_len=8, nla_type=IFLA_MAX_MTU}, 0], [{nla_len=8,
nla_type=IFLA_GROUP}, 0], [{nla_len=8, nla_type=IFLA_PROMISCUITY}, 0], [{nla_len=8,
nla_type=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=8,
nla_type=IFLA_CARRIER_UP_COUNT}, 0], [{nla_len=8,
nla_type=IFLA_CARRIER_DOWN_COUNT}, 0], [{nla_len=5,
nla_type=IFLA_PROTO_DOWN}, 0], [{nla_len=36, nla_type=IFLA_MAP}, {mem_start=0,
mem_end=0, base_addr=0, irq=0, dma=0, port=0}], [{nla_len=10, nla_type=IFLA_ADDRESS},
00:00:00:00:00:00], [{nla_len=10, nla_type=IFLA_BROADCAST}, 00:00:00:00:00:00],
[{nla_len=196, nla_type=IFLA_STATS64}, {rx_packets=3536, tx_packets=3536,
rx_bytes=232023, tx_bytes=232023, 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=3536, tx_packets=3536, rx_bytes=232023,
tx_bytes=232023, rx_errors=0, tx_errors=0, rx_dropped=0, tx_dropped=0, multicast=0,
collisions=0, rx_length_errors=0, rx_over_errors=0, rx_crc_errors=0, rx_frame_errors=0,
rx_fifo_errors=0, rx_missed_errors=0, tx_aborted_errors=0, tx_carrier_errors=0,
tx_fifo_errors=0, tx_heartbeat_errors=0, tx_window_errors=0, rx_compressed=0,
tx_compressed=0, rx_nohandler=0}], [{nla_len=12, nla_type=IFLA_XDP}, [{nla_len=5,
nla_type=IFLA_XDP_ATTACHED}, XDP_ATTACHED_NONE]], [{nla_len=764,
nla_type=IFLA_AF_SPEC}, [[{nla_len=136, nla_type=AF_INET}, [{nla_len=132,
nla_type=IFLA_INET_CONF}, [[IPV4_DEVCONF_FORWARDING-1] = 1,
[IPV4_DEVCONF_MC_FORWARDING-1] = 0, [IPV4_DEVCONF_PROXY_ARP-1] = 0,
[IPV4_DEVCONF_ACCEPT_REDIRECTS-1] = 1,
[IPV4_DEVCONF_SECURE_REDIRECTS-1] = 1, [IPV4_DEVCONF_SEND_REDIRECTS-
1] = 1, [IPV4_DEVCONF_SHARED_MEDIA-1] = 1, [IPV4_DEVCONF_RP_FILTER-1] = 2,
[IPV4_DEVCONF_ACCEPT_SOURCE_ROUTE-1] = 0,
[IPV4_DEVCONF_BOOTP_RELAY-1] = 0, [IPV4_DEVCONF_LOG_MARTIANS-1] = 0,
[IPV4_DEVCONF_TAG-1] = 0, [IPV4_DEVCONF_ARPFILTER-1] = 0,
[IPV4_DEVCONF_MEDIUM_ID-1] = 0, [IPV4_DEVCONF_NOXFRM-1] = 1,
[IPV4_DEVCONF_NOPOLICY-1] = 1, [IPV4_DEVCONF_FORCE_IGMP_VERSION-1] = 0,
[IPV4_DEVCONF_ARP_ANNOUNCE-1] = 0, [IPV4_DEVCONF_ARP_IGNORE-1] = 0,
[IPV4_DEVCONF_PROMOTE_SECONDARIES-1] = 1, [IPV4_DEVCONF_ARP_ACCEPT-
1] = 0, [IPV4_DEVCONF_ARP_NOTIFY-1] = 0, [IPV4_DEVCONF_ACCEPT_LOCAL-1] =
0, [IPV4_DEVCONF_SRC_VMARK-1] = 0, [IPV4_DEVCONF_PROXY_ARP_PVLAN-1] =
0, [IPV4_DEVCONF_ROUTE_LOCALNET-1] = 0,
[IPV4_DEVCONF_IGMPV2_UNSOLICITED_REPORT_INTERVAL-1] = 10000,
[IPV4_DEVCONF_IGMPV3_UNSOLICITED_REPORT_INTERVAL-1] = 1000,
[IPV4_DEVCONF_IGNORE_ROUTES_WITH_LINKDOWN-1] = 0,
[IPV4_DEVCONF_DROP_UNICAST_IN_L2_MULTICAST-1] = 0,
[IPV4_DEVCONF_DROP_GRATUITOUS_ARP-1] = 0,
[IPV4_DEVCONF_BC_FORWARDING-1] = 0]]], [{nla_len=624, nla_type=AF_INET6},
[[{nla_len=8, nla_type=IFLA_INET6_FLAGS}, IF_READY], [{nla_len=20,
nla_type=IFLA_INET6_CACHEINFO}, {max_reasm_len=65535, tstamp=119,
reachable_time=25594, retrans_time=1000}], [{nla_len=212, nla_type=IFLA_INET6_CONF},
[[DEVCONF_FORWARDING] = 0, [DEVCONF_HOPLIMIT] = 64, [DEVCONF_MTU6] =
65536, [DEVCONF_ACCEPT_RA] = 1, [DEVCONF_ACCEPT_REDIRECTS] = 1,
[DEVCONF_AUTOCONF] = 1, [DEVCONF_DAD_TRANSMITS] = 1,
[DEVCONF_RTR_SOLICITS] = -1, [DEVCONF_RTR_SOLICIT_INTERVAL] = 4000,
[DEVCONF_RTR_SOLICIT_DELAY] = 1000, [DEVCONF_USE_TEMPADDR] = -1,
[DEVCONF_TEMP_VALID_LFT] = 604800, [DEVCONF_TEMP_PREFERED_LFT] =
86400, [DEVCONF_REGEN_MAX_RETRY] = 3, [DEVCONF_MAX_DESYNC_FACTOR] =
600, [DEVCONF_MAX_ADDRESSES] = 16, [DEVCONF_FORCE_MLD_VERSION] = 0,
[DEVCONF_ACCEPT_RA_DEFRTR] = 1, [DEVCONF_ACCEPT_RA_PINFO] = 1,
[DEVCONF_ACCEPT_RA_RTR_PREF] = 1, [DEVCONF_RTR_PROBE_INTERVAL] =
60000, [DEVCONF_ACCEPT_RA_RT_INFO_MAX_PLEN] = 0, [DEVCONF_PROXY_NDP]
= 0, [DEVCONF_OPTIMISTIC_DAD] = 0, [DEVCONF_ACCEPT_SOURCE_ROUTE] = 0,
[DEVCONF_MC_FORWARDING] = 0, [DEVCONF_DISABLE_IPV6] = 0,
[DEVCONF_ACCEPT_DAD] = -1, [DEVCONF_FORCE_TLLAO] = 0,
[DEVCONF_NDISC_NOTIFY] = 0,
[DEVCONF_MLDV1_UNSOLICITED_REPORT_INTERVAL] = 10000,
[DEVCONF_MLDV2_UNSOLICITED_REPORT_INTERVAL] = 1000, ...]], [{nla_len=300,
nla_type=IFLA_INET6_STATS}, [[IPSTATS_MIB_NUM] = 37, [IPSTATS_MIB_INPKTS] =
173, [IPSTATS_MIB_INOCTETS] = 15413, [IPSTATS_MIB_INDELIVERS] = 173,
[IPSTATS_MIB_OUTFORWDATAGRAMS] = 0, [IPSTATS_MIB_OUTPKTS] = 173,
[IPSTATS_MIB_OUTOCTETS] = 15413, [IPSTATS_MIB_INHDRERRORS] = 0,
[IPSTATS_MIB_INTOOBIGERRORS] = 0, [IPSTATS_MIB_INNOROUTES] = 0,
[IPSTATS_MIB_INADDRERRORS] = 0, [IPSTATS_MIB_INUNKNOWNPROTOS] = 0,
[IPSTATS_MIB_INTRUNCATEDPKTS] = 0, [IPSTATS_MIB_INDISCARDS] = 0,
[IPSTATS_MIB_OUTDISCARDS] = 0, [IPSTATS_MIB_OUTNOROUTES] = 0,
[IPSTATS_MIB_REASMTIMEOUT] = 0, [IPSTATS_MIB_REASMREQDS] = 0,
[IPSTATS_MIB_REASMOKS] = 0, [IPSTATS_MIB_REASMFAILS] = 0,
[IPSTATS_MIB_FRAGOKS] = 0, [IPSTATS_MIB_FRAGFAILS] = 0,
[IPSTATS_MIB_FRAGCREATES] = 0, [IPSTATS_MIB_INMCASTPKTS] = 0,
[IPSTATS_MIB_OUTMCASTPKTS] = 0, [IPSTATS_MIB_INBCASTPKTS] = 0,
[IPSTATS_MIB_OUTBCASTPKTS] = 0, [IPSTATS_MIB_INMCASTOCTETS] = 0,
[IPSTATS_MIB_OUTMCASTOCTETS] = 0, [IPSTATS_MIB_INBCASTOCTETS] = 0,
[IPSTATS_MIB_OUTBCASTOCTETS] = 0, [IPSTATS_MIB_CSUMERRORS] = 0, ...]],
[{nla_len=52, nla_type=IFLA_INET6_ICMP6STATS}, [[ICMP6_MIB_NUM] = 6,
[ICMP6_MIB_INMSGS] = 0, [ICMP6_MIB_INERRORS] = 0, [ICMP6_MIB_OUTMSGS] = 0,
[ICMP6_MIB_OUTERRORS] = 0, [ICMP6_MIB_CSUMERRORS] = 0]], [{nla_len=20,
nla_type=IFLA_INET6_TOKEN}, inet_pton(AF_INET6, "::")], [{nla_len=5,
nla_type=IFLA_INET6_ADDR_GEN_MODE}, IN6_ADDR_GEN_MODE_EUI64]]]]]]],
[{nlmsg_len=1356, nlmsg_type=RTM_NEWLINK, nlmsg_flags=NLM_F_MULTI,
nlmsg_seq=1648280464, nlmsg_pid=9677}, {ifi_family=AF_UNSPEC,
ifi_type=ARPHRD_ETHER, ifi_index=if_nametoindex("wlo1"), ifi_flags=IFF_UP|
IFF_BROADCAST|IFF_RUNNING|IFF_MULTICAST|IFF_LOWER_UP, 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_MIN_MTU}, 256], [{nla_len=8, nla_type=IFLA_MAX_MTU}, 2304],
[{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}, 2], [{nla_len=8,
nla_type=IFLA_CARRIER_UP_COUNT}, 1], [{nla_len=8,
nla_type=IFLA_CARRIER_DOWN_COUNT}, 1], [{nla_len=5,
nla_type=IFLA_PROTO_DOWN}, 0], [{nla_len=36, nla_type=IFLA_MAP}, {mem_start=0,
mem_end=0, base_addr=0, irq=0, dma=0, port=0}], [{nla_len=10, nla_type=IFLA_ADDRESS},
5c:80:b6:b8:01:f0], [{nla_len=10, nla_type=IFLA_BROADCAST}, ff:ff:ff:ff:ff:ff],
[{nla_len=196, nla_type=IFLA_STATS64}, {rx_packets=200374, tx_packets=72363,
rx_bytes=270279625, tx_bytes=10527493, rx_errors=0, tx_errors=0, rx_dropped=100,
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=200374, tx_packets=72363, rx_bytes=270279625,
tx_bytes=10527493, rx_errors=0, tx_errors=0, rx_dropped=100, tx_dropped=0, multicast=0,
collisions=0, rx_length_errors=0, rx_over_errors=0, rx_crc_errors=0, rx_frame_errors=0,
rx_fifo_errors=0, rx_missed_errors=0, tx_aborted_errors=0, tx_carrier_errors=0,
tx_fifo_errors=0, tx_heartbeat_errors=0, tx_window_errors=0, rx_compressed=0,
tx_compressed=0, rx_nohandler=0}], [{nla_len=12, nla_type=IFLA_XDP}, [{nla_len=5,
nla_type=IFLA_XDP_ATTACHED}, XDP_ATTACHED_NONE]], [{nla_len=10,
nla_type=IFLA_PERM_ADDRESS}, 5c:80:b6:b8:01:f0], [{nla_len=764,
nla_type=IFLA_AF_SPEC}, [[{nla_len=136, nla_type=AF_INET}, [{nla_len=132,
nla_type=IFLA_INET_CONF}, [[IPV4_DEVCONF_FORWARDING-1] = 1,
[IPV4_DEVCONF_MC_FORWARDING-1] = 0, [IPV4_DEVCONF_PROXY_ARP-1] = 0,
[IPV4_DEVCONF_ACCEPT_REDIRECTS-1] = 1,
[IPV4_DEVCONF_SECURE_REDIRECTS-1] = 1, [IPV4_DEVCONF_SEND_REDIRECTS-
1] = 1, [IPV4_DEVCONF_SHARED_MEDIA-1] = 1, [IPV4_DEVCONF_RP_FILTER-1] = 2,
[IPV4_DEVCONF_ACCEPT_SOURCE_ROUTE-1] = 0,
[IPV4_DEVCONF_BOOTP_RELAY-1] = 0, [IPV4_DEVCONF_LOG_MARTIANS-1] = 0,
[IPV4_DEVCONF_TAG-1] = 0, [IPV4_DEVCONF_ARPFILTER-1] = 0,
[IPV4_DEVCONF_MEDIUM_ID-1] = 0, [IPV4_DEVCONF_NOXFRM-1] = 0,
[IPV4_DEVCONF_NOPOLICY-1] = 0, [IPV4_DEVCONF_FORCE_IGMP_VERSION-1] = 0,
[IPV4_DEVCONF_ARP_ANNOUNCE-1] = 0, [IPV4_DEVCONF_ARP_IGNORE-1] = 0,
[IPV4_DEVCONF_PROMOTE_SECONDARIES-1] = 1, [IPV4_DEVCONF_ARP_ACCEPT-
1] = 0, [IPV4_DEVCONF_ARP_NOTIFY-1] = 0, [IPV4_DEVCONF_ACCEPT_LOCAL-1] =
0, [IPV4_DEVCONF_SRC_VMARK-1] = 0, [IPV4_DEVCONF_PROXY_ARP_PVLAN-1] =
0, [IPV4_DEVCONF_ROUTE_LOCALNET-1]
 = 0, [IPV4_DEVCONF_IGMPV2_UNSOLICITED_REPORT_INTERVAL-1] = 10000,
[IPV4_DEVCONF_IGMPV3_UNSOLICITED_REPORT_INTERVAL-1] = 1000,
[IPV4_DEVCONF_IGNORE_ROUTES_WITH_LINKDOWN-1] = 0,
[IPV4_DEVCONF_DROP_UNICAST_IN_L2_MULTICAST-1] = 0,
[IPV4_DEVCONF_DROP_GRATUITOUS_ARP-1] = 0,
[IPV4_DEVCONF_BC_FORWARDING-1] = 0]]], [{nla_len=624, nla_type=AF_INET6},
[[{nla_len=8, nla_type=IFLA_INET6_FLAGS}, IF_READY], [{nla_len=20,
nla_type=IFLA_INET6_CACHEINFO}, {max_reasm_len=65535, tstamp=724,
reachable_time=29730, retrans_time=1000}], [{nla_len=212, nla_type=IFLA_INET6_CONF},
[[DEVCONF_FORWARDING] = 0, [DEVCONF_HOPLIMIT] = 64, [DEVCONF_MTU6] =
1500, [DEVCONF_ACCEPT_RA] = 0, [DEVCONF_ACCEPT_REDIRECTS] = 1,
[DEVCONF_AUTOCONF] = 1, [DEVCONF_DAD_TRANSMITS] = 1,
[DEVCONF_RTR_SOLICITS] = -1, [DEVCONF_RTR_SOLICIT_INTERVAL] = 4000,
[DEVCONF_RTR_SOLICIT_DELAY] = 1000, [DEVCONF_USE_TEMPADDR] = 0,
[DEVCONF_TEMP_VALID_LFT] = 604800, [DEVCONF_TEMP_PREFERED_LFT] =
86400, [DEVCONF_REGEN_MAX_RETRY] = 3, [DEVCONF_MAX_DESYNC_FACTOR] =
600, [DEVCONF_MAX_ADDRESSES] = 16, [DEVCONF_FORCE_MLD_VERSION] = 0,
[DEVCONF_ACCEPT_RA_DEFRTR] = 1, [DEVCONF_ACCEPT_RA_PINFO] = 1,
[DEVCONF_ACCEPT_RA_RTR_PREF] = 1, [DEVCONF_RTR_PROBE_INTERVAL] =
60000, [DEVCONF_ACCEPT_RA_RT_INFO_MAX_PLEN] = 0, [DEVCONF_PROXY_NDP]
= 0, [DEVCONF_OPTIMISTIC_DAD] = 0, [DEVCONF_ACCEPT_SOURCE_ROUTE] = 0,
[DEVCONF_MC_FORWARDING] = 0, [DEVCONF_DISABLE_IPV6] = 0,
[DEVCONF_ACCEPT_DAD] = 1, [DEVCONF_FORCE_TLLAO] = 0,
[DEVCONF_NDISC_NOTIFY] = 0,
[DEVCONF_MLDV1_UNSOLICITED_REPORT_INTERVAL] = 10000,
[DEVCONF_MLDV2_UNSOLICITED_REPORT_INTERVAL] = 1000, ...]], [{nla_len=300,
nla_type=IFLA_INET6_STATS}, [[IPSTATS_MIB_NUM] = 37, [IPSTATS_MIB_INPKTS] =
13, [IPSTATS_MIB_INOCTETS] = 1048, [IPSTATS_MIB_INDELIVERS] = 0,
[IPSTATS_MIB_OUTFORWDATAGRAMS] = 0, [IPSTATS_MIB_OUTPKTS] = 23,
[IPSTATS_MIB_OUTOCTETS] = 1520, [IPSTATS_MIB_INHDRERRORS] = 0,
[IPSTATS_MIB_INTOOBIGERRORS] = 0, [IPSTATS_MIB_INNOROUTES] = 0,
[IPSTATS_MIB_INADDRERRORS] = 0, [IPSTATS_MIB_INUNKNOWNPROTOS] = 0,
[IPSTATS_MIB_INTRUNCATEDPKTS] = 0, [IPSTATS_MIB_INDISCARDS] = 0,
[IPSTATS_MIB_OUTDISCARDS] = 0, [IPSTATS_MIB_OUTNOROUTES] = 0,
[IPSTATS_MIB_REASMTIMEOUT] = 0, [IPSTATS_MIB_REASMREQDS] = 0,
[IPSTATS_MIB_REASMOKS] = 0, [IPSTATS_MIB_REASMFAILS] = 0,
[IPSTATS_MIB_FRAGOKS] = 0, [IPSTATS_MIB_FRAGFAILS] = 0,
[IPSTATS_MIB_FRAGCREATES] = 0, [IPSTATS_MIB_INMCASTPKTS] = 13,
[IPSTATS_MIB_OUTMCASTPKTS] = 23, [IPSTATS_MIB_INBCASTPKTS] = 0,
[IPSTATS_MIB_OUTBCASTPKTS] = 0, [IPSTATS_MIB_INMCASTOCTETS] = 1048,
[IPSTATS_MIB_OUTMCASTOCTETS] = 1520, [IPSTATS_MIB_INBCASTOCTETS] = 0,
[IPSTATS_MIB_OUTBCASTOCTETS] = 0, [IPSTATS_MIB_CSUMERRORS] = 0, ...]],
[{nla_len=52, nla_type=IFLA_INET6_ICMP6STATS}, [[ICMP6_MIB_NUM] = 6,
[ICMP6_MIB_INMSGS] = 0, [ICMP6_MIB_INERRORS] = 0, [ICMP6_MIB_OUTMSGS] =
19, [ICMP6_MIB_OUTERRORS] = 0, [ICMP6_MIB_CSUMERRORS] = 0]], [{nla_len=20,
nla_type=IFLA_INET6_TOKEN}, inet_pton(AF_INET6, "::")], [{nla_len=5,
nla_type=IFLA_INET6_ADDR_GEN_MODE}, IN6_ADDR_GEN_MODE_NONE]]]]],
[{nla_len=20, nla_type=NLA_F_NESTED|IFLA_PROP_LIST}, [{nla_len=14,
nla_type=IFLA_ALT_IFNAME}, "wlp0s20f3"]]]]], iov_len=4096}], msg_iovlen=1,
msg_controllen=0, msg_flags=0}, 0) = 2676
recvmsg(9, {msg_name={sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000},
msg_namelen=12, msg_iov=[{iov_base=[{nlmsg_len=1752, nlmsg_type=RTM_NEWLINK,
nlmsg_flags=NLM_F_MULTI, nlmsg_seq=1648280464, nlmsg_pid=9677},
{ifi_family=AF_UNSPEC, ifi_type=ARPHRD_ETHER, ifi_index=if_nametoindex("docker0"),
ifi_flags=IFF_UP|IFF_BROADCAST|IFF_MULTICAST, ifi_change=0}, [[{nla_len=12,
nla_type=IFLA_IFNAME}, "docker0"], [{nla_len=8, nla_type=IFLA_TXQLEN}, 0],
[{nla_len=5, nla_type=IFLA_OPERSTATE}, 2], [{nla_len=5, nla_type=IFLA_LINKMODE},
0], [{nla_len=8, nla_type=IFLA_MTU}, 1500], [{nla_len=8, nla_type=IFLA_MIN_MTU}, 68],
[{nla_len=8, nla_type=IFLA_MAX_MTU}, 65535], [{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=12, nla_type=IFLA_QDISC}, "noqueue"], [{nla_len=8,
nla_type=IFLA_CARRIER_CHANGES}, 1], [{nla_len=8,
nla_type=IFLA_CARRIER_UP_COUNT}, 0], [{nla_len=8,
nla_type=IFLA_CARRIER_DOWN_COUNT}, 1], [{nla_len=5,
nla_type=IFLA_PROTO_DOWN}, 0], [{nla_len=36, nla_type=IFLA_MAP}, {mem_start=0,
mem_end=0, base_addr=0, irq=0, dma=0, port=0}], [{nla_len=10, nla_type=IFLA_ADDRESS},
02:42:54:1b:0a:7f], [{nla_len=10, nla_type=IFLA_BROADCAST}, ff:ff:ff:ff:ff:ff],
[{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}, XDP_ATTACHED_NONE]], [{nla_len=428,
nla_type=IFLA_LINKINFO}, [[{nla_len=11, nla_type=IFLA_INFO_KIND}, "bridge"],
[{nla_len=412, nla_type=IFLA_INFO_DATA}, [[{nla_len=12,
nla_type=IFLA_BR_HELLO_TIMER}, 0], [{nla_len=12, nla_type=IFLA_BR_TCN_TIMER},
0], [{nla_len=12, nla_type=IFLA_BR_TOPOLOGY_CHANGE_TIMER}, 0], [{nla_len=12,
nla_type=IFLA_BR_GC_TIMER}, 28937 /* 289.37 s */], [{nla_len=8,
nla_type=IFLA_BR_FORWARD_DELAY}, 1499 /* 14.99 s */], [{nla_len=8,
nla_type=IFLA_BR_HELLO_TIME}, 199 /* 1.99 s */], [{nla_len=8,
nla_type=IFLA_BR_MAX_AGE}, 1999 /* 19.99 s */], [{nla_len=8,
nla_type=IFLA_BR_AGEING_TIME}, 29999 /* 299.99 s */], [{nla_len=8,
nla_type=IFLA_BR_STP_STATE}, 0], [{nla_len=6, nla_type=IFLA_BR_PRIORITY}, 32768],
[{nla_len=5, nla_type=IFLA_BR_VLAN_FILTERING}, 0], [{nla_len=6,
nla_type=IFLA_BR_GROUP_FWD_MASK}, 0], [{nla_len=12,
nla_type=IFLA_BR_BRIDGE_ID}, {prio=[128, 0], addr=02:42:54:1b:0a:7f}], [{nla_len=12,
nla_type=IFLA_BR_ROOT_ID}, {prio=[128, 0], addr=02:42:54:1b:0a:7f}], [{nla_len=6,
nla_type=IFLA_BR_ROOT_PORT}, 0], [{nla_len=8,
nla_type=IFLA_BR_ROOT_PATH_COST}, 0], [{nla_len=5,
nla_type=IFLA_BR_TOPOLOGY_CHANGE}, 0], [{nla_len=5,
nla_type=IFLA_BR_TOPOLOGY_CHANGE_DETECTED}, 0], [{nla_len=10,
nla_type=IFLA_BR_GROUP_ADDR}, 01:80:c2:00:00:00], [{nla_len=12,
nla_type=IFLA_BR_MULTI_BOOLOPT}, {optval=0,
optmask=1<<BR_BOOLOPT_NO_LL_LEARN}], [{nla_len=6,
nla_type=IFLA_BR_VLAN_PROTOCOL}, htons(ETH_P_8021Q)], [{nla_len=6,
nla_type=IFLA_BR_VLAN_DEFAULT_PVID}, 1], [{nla_len=5,
nla_type=IFLA_BR_VLAN_STATS_ENABLED}, 0], [{nla_len=5,
nla_type=IFLA_BR_VLAN_STATS_PER_PORT}, 0], [{nla_len=5,
nla_type=IFLA_BR_MCAST_ROUTER}, 1], [{nla_len=5,
nla_type=IFLA_BR_MCAST_SNOOPING}, 1], [{nla_len=5,
nla_type=IFLA_BR_MCAST_QUERY_USE_IFADDR}, 0], [{nla_len=5,
nla_type=IFLA_BR_MCAST_QUERIER}, 0], [{nla_len=5,
nla_type=IFLA_BR_MCAST_STATS_ENABLED}, 0], [{nla_len=8,
nla_type=IFLA_BR_MCAST_HASH_ELASTICITY}, 16], [{nla_len=8,
nla_type=IFLA_BR_MCAST_HASH_MAX}, 4096], [{nla_len=8,
nla_type=IFLA_BR_MCAST_LAST_MEMBER_CNT}, 2], ...]]]], [{nla_len=764,
nla_type=IFLA_AF_SPEC}, [[{nla_len=136, nla_type=AF_INET}, [{nla_len=132,
nla_type=IFLA_INET_CONF}, [[IPV4_DEVCONF_FORWARDING-1] = 1,
[IPV4_DEVCONF_MC_FORWARDING-1] = 0, [IPV4_DEVCONF_PROXY_ARP-1] = 0,
[IPV4_DEVCONF_ACCEPT_REDIRECTS-1] = 1,
[IPV4_DEVCONF_SECURE_REDIRECTS-1] = 1, [IPV4_DEVCONF_SEND_REDIRECTS-
1] = 1, [IPV4_DEVCONF_SHARED_MEDIA-1] = 1, [IPV4_DEVCONF_RP_FILTER-1] = 2,
[IPV4_DEVCONF_ACCEPT_SOURCE_ROUTE-1] = 0,
[IPV4_DEVCONF_BOOTP_RELAY-1] = 0, [IPV4_DEVCONF_LOG_MARTIANS-1] = 0,
[IPV4_DEVCONF_TAG-1] = 0, [IPV4_DEVCONF_ARPFILTER-1] = 0,
[IPV4_DEVCONF_MEDIUM_ID-1] = 0, [IPV4_DEVCONF_NOXFRM-1] = 0,
[IPV4_DEVCONF_NOPOLICY-1] = 0, [IPV4_DEVCONF_FORCE_IGMP_VERSION-1] = 0,
[IPV4_DEVCONF_ARP_ANNOUNCE-1] = 0, [IPV4_DEVCONF_ARP_IGNORE-1] = 0,
[IPV4_DEVCONF_PROMOTE_SECONDARIES-1] = 1, [IPV4_DEVCONF_ARP_ACCEPT-
1] = 0, [IPV4_DEVCONF_ARP_NOTIFY-1] = 0, [IPV4_DEVCONF_ACCEPT_LOCAL-1] =
0, [IPV4_DEVCONF_SRC_VMARK-1] = 0, [IPV4_DEVCONF_PROXY_ARP_PVLAN-1] =
0, [IPV4_DEVCONF_ROUTE_LOCALNET-1] = 0,
[IPV4_DEVCONF_IGMPV2_UNSOLICITED_REPORT_INTERVAL-1] = 10000,
[IPV4_DEVCONF_IGMPV3_UNSOLICITED_REPORT_INTERVAL-1] = 1000,
[IPV4_DEVCONF_IGNORE_ROUTES_WITH_LINKDOWN-1] = 0,
[IPV4_DEVCONF_DROP_UNICAST_IN_L2_MULTICAST-1] = 0,
[IPV4_DEVCONF_DROP_GRATUITOUS_ARP-1] = 0,
[IPV4_DEVCONF_BC_FORWARDING-1] = 0]]], [{nla_len=624, nla_type=AF_INET6},
[[{nla_len=8, nla_type=IFLA_INET6_FLAGS}, 0], [{nla_len=20,
nla_type=IFLA_INET6_CACHEINFO}, {max_reasm_len=65535, tstamp=811,
reachable_time=35664, retrans_time=1000}], [{nla_len=212, nla_type=IFLA_INET6_CONF},
[[DEVCONF_FORWARDING] = 0, [DEVCONF_HOPLIMIT] = 64, [DEVCONF_MTU6] =
1500, [DEVCONF_ACCEPT_RA] = 0, [DEVCONF_ACCEPT_REDIRECTS] = 1,
[DEVCONF_AUTOCONF] = 1, [DEVCONF_DAD_TRANSMITS] = 1,
[DEVCONF_RTR_SOLICITS] = -1, [DEVCONF_RTR_SOLICIT_INTERVAL] = 4000,
[DEVCONF_RTR_SOLICIT_DELAY] = 1000, [DEVCONF_USE_TEMPADDR] = 0,
[DEVCONF_TEMP_VALID_LFT] = 604800, [DEVCONF_TEMP_PREFERED_LFT] =
86400, [DEVCONF_REGEN_MAX_RETRY] = 3, [DEVCONF_MAX_DESYNC_FACTOR] =
600, [DEVCONF_MAX_ADDRESSES] = 16, [DEVCONF_FORCE_MLD_VERSION] = 0,
[DEVCONF_ACCEPT_RA_DEFRTR] = 1, [DEVCONF_ACCEPT_RA_PINFO] = 1,
[DEVCONF_ACCEPT_RA_RTR_PREF] = 1, [DEVCONF_RTR_PROBE_INTERVAL] =
60000, [DEVCONF_ACCEPT_RA_RT_INFO_MAX_PLEN] = 0, [DEVCONF_PROXY_NDP]
= 0, [DEVCONF_OPTIMISTIC_DAD] = 0, [DEVCONF_ACCEPT_SOURCE_ROUTE] = 0,
[DEVCONF_MC_FORWARDING] = 0, [DEVCONF_DISABLE_IPV6] = 0,
[DEVCONF_ACCEPT_DAD] = 1, [DEVCONF_FORCE_TLLAO] = 0,
[DEVCONF_NDISC_NOTIFY] = 0,
[DEVCONF_MLDV1_UNSOLICITED_REPORT_INTERVAL] = 10000,
[DEVCONF_MLDV2_UNSOLICITED_REPORT_INTERVAL] = 1000, ...]], [{nla_len=300,
nla_type=IFLA_INET6_STATS}, [[IPSTATS_MIB_NUM] = 37, [IPSTATS_MIB_INPKTS] =
0, [IPSTATS_MIB_INOCTETS] = 0, [IPSTATS_MIB_INDELIVERS] = 0,
[IPSTATS_MIB_OUTFORWDATAGRAMS] = 0, [IPSTATS_MIB_OUTPKTS] = 0,
[IPSTATS_MIB_OUTOCTETS] = 0, [IPSTATS_MIB_INHDRERRORS] = 0,
[IPSTATS_MIB_INTOOBIGERRORS] = 0, [IPSTATS_MIB_INNOROUTES] = 0,
[IPSTATS_MIB_INADDRERRORS] = 0, [IPSTATS_MIB_INUNKNOWNPROTOS] = 0,
[IPSTATS_MIB_INTRUNCATEDPKTS] = 0, [IPSTATS_MIB_INDISCARDS] = 0,
[IPSTATS_MIB_OUTDISCARDS] = 0, [IPSTATS_MIB_OUTNOROUTES] = 0,
[IPSTATS_MIB_REASMTIMEOUT] = 0, [IPSTATS_MIB_REASMREQDS] = 0,
[IPSTATS_MIB_REASMOKS] = 0, [IPSTATS_MIB_REASMFAILS] = 0,
[IPSTATS_MIB_FRAGOKS] = 0, [IPSTATS_MIB_FRAGFAILS] = 0,
[IPSTATS_MIB_FRAGCREATES] = 0, [IPSTATS_MIB_INMCASTPKTS] = 0,
[IPSTATS_MIB_OUTMCASTPKTS] = 0, [IPSTATS_MIB_INBCASTPKTS] = 0,
[IPSTATS_MIB_OUTBCASTPKTS] = 0, [IPSTATS_MIB_INMCASTOCTETS] = 0,
[IPSTATS_MIB_OUTMCASTOCTETS] = 0, [IPSTATS_MIB_INBCASTOCTETS] = 0,
[IPSTATS_MIB_OUTBCASTOCTETS] = 0, [IPSTATS_MIB_CSUMERRORS] = 0, ...]],
[{nla_len=52, nla_type=IFLA_INET6_ICMP6STATS}, [[ICMP6_MIB_NUM] = 6,
[ICMP6_MIB_INMSGS] = 0, [ICMP6_MIB_INERRORS] = 0, [ICMP6_MIB_OUTMSGS] = 0,
[ICMP6_MIB_OUTERRORS] = 0, [ICMP6_MIB_CSUMERRORS] = 0]], [{nla_len=20,
nla_type=IFLA_INET6_TOKEN}, inet_pton(AF_INET6, "::")], [{nla_len=5,
nla_type=IFLA_INET6_ADDR_GEN_MODE}, IN6_ADDR_GEN_MODE_EUI64]]]]]]],
iov_len=4096}], msg_iovlen=1, msg_controllen=0, msg_flags=0}, 0) = 1752
recvmsg(9, {msg_name={sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000},
msg_namelen=12, msg_iov=[{iov_base=[{nlmsg_len=20, nlmsg_type=NLMSG_DONE,
nlmsg_flags=NLM_F_MULTI, nlmsg_seq=1648280464, nlmsg_pid=9677}, 0], iov_len=4096}],
msg_iovlen=1, msg_controllen=0, msg_flags=0}, 0) = 20
sendto(9, [{nlmsg_len=20, nlmsg_type=RTM_GETADDR, nlmsg_flags=NLM_F_REQUEST|
NLM_F_DUMP, nlmsg_seq=1648280465, nlmsg_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=[[{nlmsg_len=76, nlmsg_type=RTM_NEWADDR,
nlmsg_flags=NLM_F_MULTI, nlmsg_seq=1648280465, nlmsg_pid=9677},
{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}, inet_addr("127.0.0.1")], [{nla_len=8, nla_type=IFA_LOCAL},
inet_addr("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=119, tstamp=119}]]],
[{nlmsg_len=88, nlmsg_type=RTM_NEWADDR, nlmsg_flags=NLM_F_MULTI,
nlmsg_seq=1648280465, nlmsg_pid=9677}, {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}, inet_addr("192.168.1.137")], [{nla_len=8,
nla_type=IFA_LOCAL}, inet_addr("192.168.1.137")], [{nla_len=8,
nla_type=IFA_BROADCAST}, inet_addr("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=16774, ifa_valid=16774, cstamp=726, tstamp=739}]]], [{nlmsg_len=88,
nlmsg_type=RTM_NEWADDR, nlmsg_flags=NLM_F_MULTI, nlmsg_seq=1648280465,
nlmsg_pid=9677}, {ifa_family=AF_INET, ifa_prefixlen=16, ifa_flags=IFA_F_PERMANENT,
ifa_scope=RT_SCOPE_UNIVERSE, ifa_index=if_nametoindex("docker0")}, [[{nla_len=8,
nla_type=IFA_ADDRESS}, inet_addr("172.17.0.1")], [{nla_len=8, nla_type=IFA_LOCAL},
inet_addr("172.17.0.1")], [{nla_len=8, nla_type=IFA_BROADCAST},
inet_addr("172.17.255.255")], [{nla_len=12, nla_type=IFA_LABEL}, "docker0"], [{nla_len=8,
nla_type=IFA_FLAGS}, IFA_F_PERMANENT], [{nla_len=20, nla_type=IFA_CACHEINFO},
{ifa_prefered=4294967295, ifa_valid=4294967295, cstamp=811, tstamp=811}]]]]],
iov_len=4096}], msg_iovlen=1, msg_controllen=0, msg_flags=0}, 0) = 252
recvmsg(9, {msg_name={sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000},
msg_namelen=12, msg_iov=[{iov_base=[[{nlmsg_len=72, nlmsg_type=RTM_NEWADDR,
nlmsg_flags=NLM_F_MULTI, nlmsg_seq=1648280465, nlmsg_pid=9677},
{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}, inet_pton(AF_INET6, "::1")], [{nla_len=20,
nla_type=IFA_CACHEINFO}, {ifa_prefered=4294967295, ifa_valid=4294967295, cstamp=119,
tstamp=119}], [{nla_len=8, nla_type=IFA_FLAGS}, IFA_F_PERMANENT]]],
[{nlmsg_len=72, nlmsg_type=RTM_NEWADDR, nlmsg_flags=NLM_F_MULTI,
nlmsg_seq=1648280465, nlmsg_pid=9677}, {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},
inet_pton(AF_INET6, "fe80::37d0:fc9b:b6d4:8b78")], [{nla_len=20,
nla_type=IFA_CACHEINFO}, {ifa_prefered=4294967295, ifa_valid=4294967295, cstamp=724,
tstamp=826}], [{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) = 144
recvmsg(9, {msg_name={sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000},
msg_namelen=12, msg_iov=[{iov_base=[{nlmsg_len=20, nlmsg_type=NLMSG_DONE,
nlmsg_flags=NLM_F_MULTI, nlmsg_seq=1648280465, nlmsg_pid=9677}, 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(8080), sin_addr=inet_addr("127.0.0.1")}, 16) = 0
listen(9, 100)                          = 0
getsockname(9, {sa_family=AF_INET, sin_port=htons(8080), sin_addr=inet_addr("127.0.0.1")},
[128 => 16]) = 0
getsockname(9, {sa_family=AF_INET, sin_port=htons(8080), sin_addr=inet_addr("127.0.0.1")},
[128 => 16]) = 0
getpid()                                = 9677
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                                = 9677
write(8, "\1\0\0\0\0\0\0\0", 8)         = 8
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x5), ...},
AT_EMPTY_PATH) = 0
write(1, "> ", 2> )                     = 2
newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x5), ...},
AT_EMPTY_PATH) = 0
read(0, create 10
"create 10\n", 1024)            = 10
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|
SIGCHLD, child_tidptr=0x7f741c4dfc50) = 9698
write(1, "Ok: 9698\n", 9Ok: 9698
)               = 9
write(1, "> ", 2> )                     = 2
read(0, create 20
"create 20\n", 1024)            = 10
getpid()                                = 9677
poll([{fd=8, events=POLLIN}], 1, 0)     = 1 ([{fd=8, revents=POLLIN}])
getpid()                                = 9677
read(8, "\1\0\0\0\0\0\0\0", 8)          = 8
getpid()                                = 9677
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
getpid()                                = 9677
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                                = 9677
poll([{fd=8, events=POLLIN}], 1, -1)    = 1 ([{fd=8, revents=POLLIN}])
getpid()                                = 9677
read(8, "\1\0\0\0\0\0\0\0", 8)          = 8
getpid()                                = 9677
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
getpid()                                = 9677
poll([{fd=8, events=POLLIN}], 1, -1)    = 1 ([{fd=8, revents=POLLIN}])
getpid()                                = 9677
```

```
read(8, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                               = 9677
poll([{fd=8, events=POLLIN}], 1, 0)    = 0 (Timeout)
getpid()                               = 9677
write(6, "\1\0\0\0\0\0\0\0", 8)        = 8
write(1, "Ok: 9724\n", 9Ok: 9724
)             = 9
write(1, "> ", 2> )                    = 2
read(0, exec 10 abraasabra abra
"exec 10 abraasabra abra\n", 1024) = 24
getpid()                               = 9677
poll([{fd=8, events=POLLIN}], 1, 0)    = 0 (Timeout)
getpid()                               = 9677
write(6, "\1\0\0\0\0\0\0\0", 8)        = 8
getpid()                               = 9677
poll([{fd=8, events=POLLIN}], 1, -1)   = 1 ([{fd=8, revents=POLLIN}])
getpid()                               = 9677
read(8, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                               = 9677
poll([{fd=8, events=POLLIN}], 1, 0)    = 0 (Timeout)
write(1, "Ok: 10: [0 6]\n", 14Ok: 10: [0 6]
)        = 14
write(1, "> ", 2> )                    = 2
read(0, pingid 10
"pingid 10\n", 1024)         = 10
getpid()                               = 9677
poll([{fd=8, events=POLLIN}], 1, 0)    = 0 (Timeout)
getpid()                               = 9677
write(6, "\1\0\0\0\0\0\0\0", 8)        = 8
getpid()                               = 9677
poll([{fd=8, events=POLLIN}], 1, -1)   = 1 ([{fd=8, revents=POLLIN}])
getpid()                               = 9677
read(8, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                               = 9677
poll([{fd=8, events=POLLIN}], 1, 0)    = 0 (Timeout)
write(1, "Ok: 1\n", 6Ok: 1
)               = 6
write(1, "> ", 2> )                    = 2
read(0, pingid 12
"pingid 12\n", 1024)         = 10
getpid()                               = 9677
```

```
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
getpid()                                = 9677
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                                = 9677
poll([{fd=8, events=POLLIN}], 1, -1)    = 1 ([{fd=8, revents=POLLIN}])
getpid()                                = 9677
read(8, "\1\0\0\0\0\0\0\0", 8)          = 8
getpid()                                = 9677
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
write(1, "Ok: 0\n", 6Ok: 0
)                  = 6
write(1, "> ", 2> )                     = 2
read(0, exit
"exit\n", 1024)                 = 5
getpid()                                = 9677
write(4, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                                = 9677
getpid()                                = 9677
poll([{fd=3, events=POLLIN}], 1, -1)    = 1 ([{fd=3, revents=POLLIN}])
getpid()                                = 9677
read(3, "\1\0\0\0\0\0\0\0", 8)          = 8
getpid()                                = 9677
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
futex(0x7f741bcdb910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 9679, NULL,
FUTEX_BITSET_MATCH_ANY) = -1 EAGAIN (Resource temporarily unavailable)
close(7)                                = 0
close(6)                                = 0
close(5)                                = 0
close(4)                                = 0
close(3)                                = 0
lseek(0, -1, SEEK_CUR)                  = -1 ESPIPE (Illegal seek)
exit_group(0)                           = ?
+++ exited with 0 +++
[suraba04@asusx512fl third]$
```

Демонстрация работы программы

```
[suraba04@asusx512fl third]$ ./client
> create 1
Ok: 3965
> create 2
Ok: 3986
> create 3
Ok: 4000
> create 4
Ok: 4005
> create 5
Ok: 4009
> cl;ak
Error Command
> exec 1 hlasghljh as
Ok: 1: [2]
> exec 2 abracadabra abra
Ok: 2: [0 7]
> exec 3 ComebackpapaFranku a
Ok: 3: [5 9 11 14]
> exec 6 alsjh lhjg
Error: 6: Not found
> exec 5 lasjgh abab
Ok: 5: [-1]
> pingid 1
Ok: 1
> pingid 3
Ok: 1
> exit
```

Вывод

Данная лабораторная работа оказалась самой сложной частью курса, даже отчасти сложнее моего курсового проекта.

Однако она является и самой полезной. Ее польза состоит в том, что она соединяет все элементы курса в одном проекте. Хотя я и не использовал потоки, но их тоже можно было добавить.

Выполнение ЛР побудило меня разобраться в zmq, она имеет хорошую документации и подробный гайд, поэтому я хочу использовать данную библиотеку и в курсовом проекте.

Больше всего понравилось строить схему общения (архитектуру) между серверами и клиентом. Долго думал как и какие паттерны использовать, чтобы программа работала максимально логично, однако остановился на req-rep. Но, считаю, можно было использовать pub-sub, однако нативно у него нет свойства ответа.

Также для выполнения специального задания (поиск подстроки в строке) я решил использовать какой-нибудь интересный алгоритм, осатновился на алгоритме КМП, так как я его писал еще в первом семестре.