Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математикиКафедра вычислительной математики и программирования

Лабораторная работа №6-8 по курсу

«Операционные системы»

Группа: М80 – 201Б-19
Студент: Цыкин И.А.
Преподаватель: Миронов Е.С
Оценка:

_____

_____

Дата:

_____

Москва,
2020.

Содержание

Постановка задачи

Целью является приобретение практических навыков в:

- Управлении серверами сообщений (No6)
- Применение отложенных вычислений (No7)
- Интеграция программных систем друг с другом (No8)

Вариант 45

1. Топология — отсортированное бинарное дерево (Но у меня не вышло реализовать это).
2. Тип вычислительной команды — локальный таймер.
3. Тип проверки узлов на доступность — heartbeat time.

Общие сведения о программе

Программа состоит из двух основных файлов и библиотеки, реализующей взаимодействия с узлами. Помимо этого используется библиотека zmq, которая реализует очередь сообщений.
1) main.cpp — программа управляющего узлов
3) command.hpp - реализация библиотеки для взаимодействия между узлами
5) child.cpp – файл дочернего узла

Очередь сообщений - компонент, используемый для межпроцессного или межпотокового взаимодействия внутри одного процесса. Для обмена сообщениями используется очередь. Очереди сообщений предоставляют асинхронный протокол передачи данных, означая, что отправитель и получатель сообщения не обязаны взаимодействовать с очередью сообщений одновременно. Размещённые в очереди сообщения хранятся до тех пор, пока получатель не получит их.

ZMQ - библиотека асинхронных сообщений, предназначенная для использования в распределенных или параллельных приложениях. Он обеспечивает очередь сообщений, но в отличие от промежуточного программного обеспечения, ориентированного на сообщения, система ZMQ может работать без выделенного посредника сообщений.

Сокеты - название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой сетью. Сокет — абстрактный объект, представляющий конечную точку соединения.

Общий метод и алгоритм решения

1. Управляющий узел принимает команды, обрабатывает их и пересылает дочерниму узлу.
2. Дочерние узлы проверяют, может ли быть команда выполнена в данном узле, если нет, то команда пересылается в дочерний узлел из которого возвращается некоторое сообщение (об успехе или об ошибке), которое потом пересылается обратно к главному.
3. Если узел недоступен, то по истечении таймаута будет сгенерировано сообщение о недоступности узла и оно будет передано вверх по дереву, к управляющему узлу. При удалении узла, все его потомки уничтожаются.

Листинг программы

main.cpp

```
#include "command.hpp"

#include <csignal>
#include <vector>
#include <map>

using namespace std;

int main(){
      vector<int> vec;
      map<int, int> works;
      zmq::context_t context(1);
      zmq::socket_t main_socket(context, ZMQ_REQ);
      int port = bind_socket(main_socket);
      string cmd;
      string msg;
      string sub_cmd;
      string result;
      int input_id;
      int n = 5;
      int child_pid = 0;
```

```
int child_id = 0;

auto begin = chrono::steady_clock::now();
auto end = chrono::steady_clock::now();
auto elapsed_ms = 0;

for(;;){
      cin >> cmd;
      if(cmd == "create") {
            cin >> input_id;
            if (child_pid == 0) {
                  child_pid = fork();
                  if (child_pid == 0) {
                        create_node(input_id, port);
                  } else{
                        child_id = input_id;
                        msg = "pid";
                        message_send(main_socket, msg);
                        result = message_recieve(main_socket);
                        if(result.substr(0,2) == "OK"){
                              vec.push_back(input_id);
                        }
                  }
            }else {
                  ostringstream msg_stream;
                  msg_stream << "create " << input_id;
                  message_send(main_socket, msg_stream.str());
                  result = message_recieve(main_socket);
                  if(result.substr(0,2) == "OK"){
                        vec.push_back(input_id);
                  }
            }
            cout << result << endl;
      }else if(cmd == "remove") {
            if (child_pid == 0) {
                  cout << "Error: Not found" << endl;
                  continue;
            }
            cin >> input_id;
            if (input_id == child_id) {
                  msg = "kill_child";
                  message_send(main_socket, msg);
                  result = message_recieve(main_socket);
                  if(result == "OK"){
                        kill(child_pid, SIGTERM);
                        kill(child_pid, SIGKILL);
                        child_id = 0;
                        child_pid = 0;
                        cout << result << endl;
                        vec.clear();
                  }else{
```

```
                        cout << "Error: exit" << endl;
                  }
                  continue;
            }
            ostringstream msg_stream;
            msg_stream << "remove " << input_id;
            message_send(main_socket, msg_stream.str());
            result = message_recieve(main_socket);
            cout << result << endl;
            if(result.substr(0,2) == "OK"){
                  for(int i = vec.size() - 1; i >= 0; --i){
                        if(vec[i] != input_id){
                        vec.pop_back();
                  }else{
                        vec.pop_back();
                        break;
                  }
            }
      }
}else if(cmd == "all"){
      if (child_pid == 0) {
            cout << "Error: Not found" << endl;
            continue;
      }
      for(int i = 0; i < vec.size(); i++){
            cout << vec[i] << " ";
      }
      cout << endl;
}else if(cmd == "exec"){
      if (child_pid == 0) {
            cout << "Error: Not found" << endl;
            continue;
      }
      cin >> input_id;
      cin >> sub_cmd;
      ostringstream msg_stream;
      msg_stream << "exec " << input_id << " " << sub_cmd;
      message_send(main_socket, msg_stream.str());
      result = message_recieve(main_socket);
      cout << result << endl;
}else if(cmd == "heartbeat"){
      if (child_pid == 0) {
            cout << "Error: Not found" << endl;
            continue;
      }
      works.clear();
      int time;
      cin >> time;
      cmd = cmd + " " + to_string(time);
      sleep(time/1000);
      message_send(main_socket, cmd);
```

```
            result = message_recieve(main_socket);
            istringstream is = istringstream(result);
            while(is){
                  is >> input_id;
                  works.insert(make_pair(input_id, 1));
            }
            cout << "OK" << endl;
      }else if (cmd == "ping") {
            if(works.size() == 0){
                  continue;
            }
            cin >> input_id;
            if(works[input_id] == 1){
                  cout << "OK: 1" << endl;
            }else{
                  cout << "OK: -1" << endl;
            }
      }else if(cmd == "exit"){
            if(child_pid == 0){
                  cout << "OK\n";
                  return 0;
            }
            msg = "kill_child";
            message_send(main_socket, msg);
            result = message_recieve(main_socket);
            if(result == "OK"){
                  kill(child_pid, SIGTERM);
                  kill(child_pid, SIGKILL);
                  child_id = 0;
                  child_pid = 0;
                  cout << result << endl;
            }else{
                  cout << "Error: exit" << endl;
            }
                  return 0;
            }else{
                  cout << "Error: bad command" << endl;
            }
      }
}
```

child.cpp

```
#include <csignal>
#include <chrono>

using namespace std;

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

```
if(argc != 2) {
	cout << "Error: child's parametrs" << endl;
	return -1;
}

int id = stoi(argv[0]);
int port = stoi(argv[1]);
zmq::context_t context(2);
zmq::socket_t parent_socket(context, ZMQ_REP);
zmq::socket_t child_socket(context, ZMQ_REQ);

parent_socket.connect(get_port(port));
int child_port = bind_socket(child_socket);

string request;
string cmd;
string sub_cmd;
string msg;
string result;
int input_id;
int child_pid = 0;
int child_id = 0;
int send_child = 0;
int last_heartbit_time = -1;

auto begin = chrono::steady_clock::now();
auto end = chrono::steady_clock::now();
auto elapsed_ms = 0;

for(;;){
	request = message_recieve(parent_socket);
	istringstream cmd_stream(request);
	cmd_stream >> cmd;
		if(cmd == "pid") {
			msg = "OK: " + to_string(getpid());
			message_send(parent_socket, msg);
		} else if (cmd == "kill_child") {
			if (child_pid == 0) {
				msg = "OK";
				message_send(parent_socket, msg);
			} else {
				msg = "kill_child";
				message_send(child_socket, msg);
				result = message_recieve(child_socket);
				if(result == "OK"){
					message_send(parent_socket, result);
				}else{
					cout << "Error: kill" << endl;
				}
				kill(child_pid, SIGTERM);
				kill(child_pid, SIGKILL);
```

```
                message_send(parent_socket, result);
        }
    }else if(cmd == "ping"){
        if(child_pid == 0){
                msg = "OK: ";
                message_send(parent_socket, msg);
        }else{
                message_send(child_socket, cmd);
                string str = message_recieve(child_socket);
                result = str + to_string(child_id) + " ";
                message_send(parent_socket, result);
        }
    }else if(cmd == "create") {
        cmd_stream >> input_id;
        if (input_id == id) {
                msg = "Error: Already exists";
                message_send(parent_socket, msg);
        } else if (child_pid == 0) {
                child_pid = fork();
                        if (child_pid == 0) {
                        create_node(input_id, child_port);
                } else {
                        child_id = input_id;
                        msg = "pid";
                        message_send(child_socket, msg);
                        result = message_recieve(child_socket);
                        message_send(parent_socket, result);
                }
        } else {
                message_send(child_socket, request);
                result = message_recieve(child_socket);
                message_send(parent_socket, result);
        }
    }else if(cmd == "remove"){
        cmd_stream >> input_id;
        if(child_pid == 0){
                msg = "Error: Not found";
                message_send(parent_socket, msg);
        }else if(child_id == input_id){
                msg = "kill_child";
                message_send(child_socket, msg);
                result = message_recieve(child_socket);
        if(result == "OK"){
                message_send(parent_socket, result);
        }else{
                cout << "Error: kill" << endl;
        }
        kill(child_pid, SIGTERM);
        kill(child_pid, SIGKILL);
        child_pid = 0;
        child_id = 0;
```

```
                        message_send(parent_socket, result);
                        } else{
                              message_send(child_socket, request);
                              result = message_recieve(child_socket);
                              message_send(parent_socket, result);
                        }
                  }else if(cmd == "exec"){
                        cmd_stream >> input_id;
                        if(id == input_id){
                              cmd_stream >> sub_cmd;
                              if(sub_cmd == "start"){
                                    begin = std::chrono::steady_clock::now();
                                    result = "OK: start";
                              }else if(sub_cmd == "stop"){
                                    end = std::chrono::steady_clock::now();
                                    elapsed_ms =
chrono::duration_cast<std::chrono::milliseconds>(end - begin).count();
                              result = "OK: stop";
                              }else if(sub_cmd == "time"){
                                    result = "OK: " + to_string(elapsed_ms) +
" ms";
                                    elapsed_ms = 0;
                              }else{
                                    result = "Error: bad subcommand";
                              }
                              message_send(parent_socket, result);
                        }else{
                              if(child_pid == 0){
                              msg = "Error: Not found";
                              message_send(child_socket, msg);
                        }else{
                              message_send(child_socket, request);
                              result = message_recieve(child_socket);
                              message_send(parent_socket, result);
                        }
                  }
            }else if(cmd == "heartbeat"){
                  int time;
                  cmd_stream >> time;
                  if(child_pid == 0){
                        msg = to_string(id);
                  }else{
                        auto t1 = std::chrono::steady_clock::now();
                        message_send(child_socket, request);
                        result = message_recieve(child_socket);
                        auto t2 = std::chrono::steady_clock::now();
                        auto T =
chrono::duration_cast<std::chrono::milliseconds>(t2 - t1).count();
                        if(T > 4*time){
                              msg = to_string(id);
                        }else{
```

```
                              msg = result + " " + to_string(id);
                        }
                  }
                  message_send(parent_socket, msg);
            }
      }
}
```

command.cpp

```
#include <iostream>

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

using namespace std;

void create_node(int& id, int& port) {
      char* arg_id = strdup((to_string(id)).c_str());
      char* arg_port = strdup((to_string(port)).c_str());
      char* args[] = {arg_id, arg_port, NULL};
      execv("./child", args);
}

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

int bind_socket(zmq::socket_t& socket) {
      int port = 3000;
      while (true) {
            try {
                  socket.bind(get_port(port));
                  break;
            } catch(zmq::error_t &e) {
                  ++port;
            }
      }
      return port;
}

bool message_send(zmq::socket_t& socket, const string& msg) {
      int msg_size = msg.size();
      zmq::message_t message(msg_size);
      memcpy(message.data(), msg.c_str(), msg_size);
      try {
            socket.send(message, zmq::send_flags::none);
            return true;
      } catch(...) {
```

```
            return false;
      }
}

string message_recieve(zmq::socket_t& socket) {
      zmq::message_t request;
      zmq::send_result_t answer;
      try {
            answer = socket.recv(request, zmq::recv_flags::none);
      } catch(zmq::error_t &e) {
            answer = false;
      }
      string recieve_msg(static_cast<char*>(request.data()), request.size());
      if (recieve_msg.empty() || !answer)
            return "Error: Node is not available";
      else
            return recieve_msg;
}
```

Результаты работы программы

```
vaney@vaney-VirtualBox:~/OS/lab6$ g++ -o main main.cpp -lzmq
vaney@vaney-VirtualBox:~/OS/lab6$ g++ -o child child.cpp -lzmq
vaney@vaney-VirtualBox:~/OS/lab6$ ./main
create 12
OK: 6406
create 66
OK: 6410
create 88
OK: 6414
create 90
OK: 6419
all
12 66 88 90
exec 12 start
OK: start
heartbeat 100
OK
ping 12
OK: 1
ping 66
OK: 1
remove 66
OK
all
12
heartbeat 250
OK
ping 66
```

```
OK: -1
ping 12
OK: 1
exec 12 stop
OK: stop
exec 12 time
OK: 59899 ms
remove 12
OK
all
Error: Not found
heartbeat 100
Error: Not found
Error: bad command
exit
OK
vaney@vaney-VirtualBox:~/OS/lab6$
```

Strace

```
vaney@vaney-VirtualBox:~/OS/lab6$ strace ./main
execve("./main", ["./main"], 0x7fff2a1baf60 /* 58 vars */) = 0
brk(NULL)                               = 0x55580782b000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffcd2ffcd30) = -1 EINVAL (Invalid
argument)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or
directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=87309, ...}) = 0
mmap(NULL, 87309, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f2550743000
close(3)                                = 0
openat(AT_FDCWD, "/usr/local/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\210\1\0\0\0\0\0"..., 832)
= 832
fstat(3, {st_mode=S_IFREG|0755, st_size=16810584, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f2550741000
mmap(NULL, 685720, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f2550699000
mprotect(0x7f25506b0000, 557056, PROT_NONE) = 0
mmap(0x7f25506b0000, 425984, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17000) = 0x7f25506b0000
mmap(0x7f2550718000, 126976, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x7f000) = 0x7f2550718000
mmap(0x7f2550738000, 36864, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9e000) = 0x7f2550738000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.so.6", O_RDONLY|O_CLOEXEC)
= 3
```

```
read(3,
"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\240\341\t\0\0\0\0\0"..., 832)
= 832
fstat(3, {st_mode=S_IFREG|0644, st_size=1952928, ...}) = 0
mmap(NULL, 1968128, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f25504b8000
mprotect(0x7f255054e000, 1286144, PROT_NONE) = 0
mmap(0x7f255054e000, 983040, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x96000) = 0x7f255054e000
mmap(0x7f255063e000, 299008, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x186000) = 0x7f255063e000
mmap(0x7f2550688000, 57344, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1cf000) = 0x7f2550688000
mmap(0x7f2550696000, 10240, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f2550696000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) =
3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\3405\0\0\0\0\0\0"...,
832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=104984, ...}) = 0
mmap(NULL, 107592, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f255049d000
mmap(0x7f25504a0000, 73728, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7f25504a0000
mmap(0x7f25504b2000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x15000) = 0x7f25504b2000
mmap(0x7f25504b6000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x18000) = 0x7f25504b6000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360q\2\0\0\0\0\0"...,
832) = 832
pread64(3,
"\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64)
= 784
pread64(3,
"\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848)
= 32
pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X
>\263"..., 68, 880) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=2029224, ...}) = 0
pread64(3,
"\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64)
= 784
pread64(3,
"\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848)
= 32
pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X
>\263"..., 68, 880) = 68
mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f25502ab000
```

```
mprotect(0x7f25502d0000, 1847296, PROT_NONE) = 0
mmap(0x7f25502d0000, 1540096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25000) = 0x7f25502d0000
mmap(0x7f2550448000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x19d000) = 0x7f2550448000
mmap(0x7f2550493000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f2550493000
mmap(0x7f2550499000, 13528, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f2550499000
close(3)                                = 0
openat(AT_FDCWD, "/usr/local/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 \314\0\0\0\0\0\0"...,
832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=3652112, ...}) = 0
mmap(NULL, 365576, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f2550251000
mprotect(0x7f255025d000, 311296, PROT_NONE) = 0
mmap(0x7f255025d000, 233472, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xc000) = 0x7f255025d000
mmap(0x7f2550296000, 73728, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x45000) = 0x7f2550296000
mmap(0x7f25502a9000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x57000) = 0x7f25502a9000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libpthread.so.0", O_RDONLY|O_CLOEXEC)
= 3
read(3,
"\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\201\0\0\0\0\0\0"..., 832)
= 832
pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\345Ga\367\265T\320\374\301V)Yf]\223\337"...,
68, 824) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=157224, ...}) = 0
pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\345Ga\367\265T\320\374\301V)Yf]\223\337"...,
68, 824) = 68
mmap(NULL, 140408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f255022e000
mmap(0x7f2550235000, 69632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x7f2550235000
mmap(0x7f2550246000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x18000) = 0x7f2550246000
mmap(0x7f255024b000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c000) = 0x7f255024b000
mmap(0x7f255024d000, 13432, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f255024d000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3,
"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\300\363\0\0\0\0\0\0"..., 832)
= 832
fstat(3, {st_mode=S_IFREG|0644, st_size=1369352, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f255022c000
```

```
mmap(NULL, 1368336, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f25500dd000
mmap(0x7f25500ec000, 684032, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xf000) = 0x7f25500ec000
mmap(0x7f2550193000, 618496, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xb6000) = 0x7f2550193000
mmap(0x7f255022a000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x14c000) = 0x7f255022a000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f25500da000
arch_prctl(ARCH_SET_FS, 0x7f25500da980) = 0
mprotect(0x7f2550493000, 12288, PROT_READ) = 0
mprotect(0x7f255022a000, 4096, PROT_READ) = 0
mprotect(0x7f255024b000, 4096, PROT_READ) = 0
mprotect(0x7f25502a9000, 4096, PROT_READ) = 0
mprotect(0x7f25504b6000, 4096, PROT_READ) = 0
mprotect(0x7f2550688000, 45056, PROT_READ) = 0
mprotect(0x7f2550738000, 32768, PROT_READ) = 0
mprotect(0x555807298000, 4096, PROT_READ) = 0
mprotect(0x7f2550786000, 4096, PROT_READ) = 0
munmap(0x7f2550743000, 87309)           = 0
set_tid_address(0x7f25500dac50)         = 3186
set_robust_list(0x7f25500dac60, 24)     = 0
rt_sigaction(SIGRTMIN, {sa_handler=0x7f2550235bf0, sa_mask=[],
sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7f25502433c0}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x7f2550235c90, sa_mask=[],
sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f25502433c0},
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)                               = 0x55580782b000
brk(0x55580784c000)                     = 0x55580784c000
futex(0x7f25506966bc, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f25506966c8, FUTEX_WAKE_PRIVATE, 2147483647) = 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("\xa4\xe6\x4b\xb0\xd0\x75\x0a\xae\xfe\xd5\x33\x93\x56\x87\x4e\x3f",
16, 0) = 16
getrandom("\xb9\x1a\xfd\x57\xc9\xaa\xf5\x89\x9d\xb0\xb2\x72\x07\xc8\x53\x53",
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=126094528, u64=93836571577536}}) = 0
```

```
epoll_ctl(5, EPOLL_CTL_MOD, 4, {EPOLLIN, {u32=126094528,
u64=93836571577536}}) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7f254f8d9000
mprotect(0x7f254f8da000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7f25500d8d70,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[3187],
tls=0x7f25500d9700, child_tidptr=0x7f25500d99d0) = 3187
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=126096560, u64=93836571579568}}) = 0
epoll_ctl(7, EPOLL_CTL_MOD, 6, {EPOLLIN, {u32=126096560,
u64=93836571579568}}) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7f254f0d8000
mprotect(0x7f254f0d9000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7f254f8d7d70,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[3188],
tls=0x7f254f8d8700, child_tidptr=0x7f254f8d89d0) = 3188
eventfd2(0, EFD_CLOEXEC)                = 8
fcntl(8, F_GETFL)                       = 0x2 (flags O_RDWR)
fcntl(8, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
fcntl(8, F_GETFL)                       = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(8, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
socket(AF_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=3186, nl_groups=00000000}, [12])
= 0
sendto(9, {{len=20, type=RTM_GETLINK, flags=NLM_F_REQUEST|NLM_F_DUMP,
seq=1619027209, 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=1320, type=RTM_NEWLINK,
flags=NLM_F_MULTI, seq=1619027209, pid=3186}, {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=5, nla_type=IFLA_PROTO_DOWN},
```

0}, {{nla_len=8, nla_type=IFLA_CARRIER_UP_COUNT}, 0}, {{nla_len=8,
nla_type=IFLA_CARRIER_DOWN_COUNT}, 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}, "\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=286, tx_packets=286, rx_bytes=30545,
tx_bytes=30545, 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=286,
tx_packets=286, rx_bytes=30545, tx_bytes=30545, 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] = 0,
[IPV4_DEVCONF_MC_FORWARDING-1] = 0, [IPV4_DEVCONF_PROXY_ARP-1] = 0,
[IPV4_DEVCONF_ACCEPT_REDIRECTS-1] = 1, [IPV4_DEVCONF_SECURE_REDIRECTS-1] = 1,
[IPV4_DEVCONF_SEND_REDIRECTS-1] = 1, [IPV4_DEVCONF_SHARED_MEDIA-1] = 1,
[IPV4_DEVCONF_RP_FILTER-1] = 0, [IPV4_DEVCONF_ACCEPT_SOURCE_ROUTE-1] = 1,
[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=195,
reachable_time=42272, 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] = 6, [IPSTATS_MIB_INOCTETS] = 432,
[IPSTATS_MIB_INDELIVERS] = 6, [IPSTATS_MIB_OUTFORWDATAGRAMS] = 0,
[IPSTATS_MIB_OUTPKTS] = 6, [IPSTATS_MIB_OUTOCTETS] = 432,
[IPSTATS_MIB_INHDRERRORS] = 0, [IPSTATS_MIB_INTOOBIGERRORS] = 0,
[IPSTATS_MIB_INNOROUTES] = 0, [IPSTATS_MIB_INADDRERRORS] = 0,
[IPSTATS_MIB_INUNKNOWNPROTOS] = 0, [IPSTATS_MIB_INTRUNCATEDPKTS] = 0,
[IPSTATS_MIB_INDISCARDS] = 0, [IPSTATS_MIB_OUTDISCARDS] = 0,
[IPSTATS_MIB_OUTNOROUTES] = 0, [IPSTATS_MIB_REASMTIMEOUT] = 0,
[IPSTATS_MIB_REASMREQDS] = 0, [IPSTATS_MIB_REASMOKS] = 0,
[IPSTATS_MIB_REASMFAILS] = 0, [IPSTATS_MIB_FRAGOKS] = 0,
[IPSTATS_MIB_FRAGFAILS] = 0, [IPSTATS_MIB_FRAGCREATES] = 0,
[IPSTATS_MIB_INMCASTPKTS] = 0, [IPSTATS_MIB_OUTMCASTPKTS] = 2,
[IPSTATS_MIB_INBCASTPKTS] = 0, [IPSTATS_MIB_OUTBCASTPKTS] = 0,
[IPSTATS_MIB_INMCASTOCTETS] = 0, [IPSTATS_MIB_OUTMCASTOCTETS] = 152,
[IPSTATS_MIB_INBCASTOCTETS] = 0, [IPSTATS_MIB_OUTBCASTOCTETS] = 0,
[IPSTATS_MIB_CSUMERRORS] = 0, ...]}, {{nla_len=52,
nla_type=IFLA_INET6_ICMP6STATS}, [[ICMP6_MIB_NUM] = 6, [ICMP6_MIB_INMSGS] =
2, [ICMP6_MIB_INERRORS] = 0, [ICMP6_MIB_OUTMSGS] = 2, [ICMP6_MIB_OUTERRORS] =
0, [ICMP6_MIB_CSUMERRORS] = 0]}, {{nla_len=20, nla_type=IFLA_INET6_TOKEN},
inet_pton(AF_INET6, "::")}, {{nla_len=5, nla_type=IFLA_INET6_ADDR_GEN_MODE},
IN6_ADDR_GEN_MODE_EUI64}]}]}]}, {{len=1340, type=RTM_NEWLINK,
flags=NLM_F_MULTI, seq=1619027209, pid=3186}, {ifi_family=AF_UNSPEC,
ifi_type=ARPHRD_ETHER, ifi_index=if_nametoindex("enp0s3"),
ifi_flags=IFF_UP|IFF_BROADCAST|IFF_RUNNING|IFF_MULTICAST|IFF_LOWER_UP,
ifi_change=0}, [{{nla_len=11, nla_type=IFLA_IFNAME}, "enp0s3"}, {{nla_len=8,
nla_type=IFLA_TXQLEN}, 1000}, {{nla_len=5, nla_type=IFLA_OPERSTATE}, 6},
{{nla_len=5, nla_type=IFLA_LINKMODE}, 0}, {{nla_len=8, nla_type=IFLA_MTU},
1500}, {{nla_len=8, nla_type=IFLA_MIN_MTU}, 46}, {{nla_len=8,
nla_type=IFLA_MAX_MTU}, 16110}, {{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=13, nla_type=IFLA_QDISC}, "fq_codel"}, {{nla_len=8,
nla_type=IFLA_CARRIER_CHANGES}, 2}, {{nla_len=5, nla_type=IFLA_PROTO_DOWN},
0}, {{nla_len=8, nla_type=IFLA_CARRIER_UP_COUNT}, 1}, {{nla_len=8,
nla_type=IFLA_CARRIER_DOWN_COUNT}, 1}, {{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}, "\x08\x00\x27\xbc\xa3\x0a"}, {{nla_len=10,
nla_type=IFLA_BROADCAST}, "\xff\xff\xff\xff\xff\xff"}, {{nla_len=196,
nla_type=IFLA_STATS64}, {rx_packets=7371, tx_packets=2000, rx_bytes=10317888,
tx_bytes=228168, 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=7371,
tx_packets=2000, rx_bytes=10317888, tx_bytes=228168, rx_errors=0,
tx_errors=0, rx_dropped=0, tx_dropped=0, multicast=0, collisions=0,
rx_length_errors=0, rx_over_errors=0, rx_crc_errors=0, rx_frame_errors=0,
rx_fifo_errors=0, rx_missed_errors=0, tx_aborted_errors=0,
tx_carrier_errors=0, tx_fifo_errors=0, tx_heartbeat_errors=0,
tx_window_errors=0, rx_compressed=0, tx_compressed=0, rx_nohandler=0}},
{{nla_len=12, nla_type=IFLA_XDP}, {{nla_len=5, nla_type=IFLA_XDP_ATTACHED},
XDP_ATTACHED_NONE}}, {{nla_len=10, nla_type=0x36 /* IFLA_??? */},
"\x08\x00\x27\xbc\xa3\x0a"}, {{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] = 0, [IPV4_DEVCONF_MC_FORWARDING-1] = 0,

```
[IPV4_DEVCONF_PROXY_ARP-1] = 0, [IPV4_DEVCONF_ACCEPT_REDIRECTS-1] = 1,
[IPV4_DEVCONF_SECURE_REDIRECTS-1] = 1, [IPV4_DEVCONF_SEND_REDIRECTS-1] = 1,
[IPV4_DEVCONF_SHARED_MEDIA-1] = 1, [IPV4_DEVCONF_RP_FILTER-1] = 2,
[IPV4_DEVCONF_ACCEPT_SOURCE_ROUTE-1] = 1, [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=1129,
reachable_time=42756, 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] = 2, [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] = 14, [IPSTATS_MIB_INOCTETS] = 1376,
[IPSTATS_MIB_INDELIVERS] = 14, [IPSTATS_MIB_OUTFORWDATAGRAMS] = 0,
[IPSTATS_MIB_OUTPKTS] = 24, [IPSTATS_MIB_OUTOCTETS] = 2168,
[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] = 14, [IPSTATS_MIB_OUTMCASTPKTS] = 24,
[IPSTATS_MIB_INBCASTPKTS] = 0, [IPSTATS_MIB_OUTBCASTPKTS] = 0,
[IPSTATS_MIB_INMCASTOCTETS] = 1376, [IPSTATS_MIB_OUTMCASTOCTETS] = 2168,
[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] = 10, [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}]}]}]}], iov_len=4096}], msg_iovlen=1,
msg_controllen=0, msg_flags=0}, 0) = 2660
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=1619027209, pid=3186}, 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=1619027210, 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=1619027210, pid=3186}, {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=195, tstamp=195}}]}, {{len=88, type=RTM_NEWADDR, flags=NLM_F_MULTI,
seq=1619027210, pid=3186}, {ifa_family=AF_INET, ifa_prefixlen=24,
ifa_flags=0, ifa_scope=RT_SCOPE_UNIVERSE,
ifa_index=if_nametoindex("enp0s3")}, [{{nla_len=8, nla_type=IFA_ADDRESS},
inet_addr("10.0.2.15")}, {{nla_len=8, nla_type=IFA_LOCAL},
inet_addr("10.0.2.15")}, {{nla_len=8, nla_type=IFA_BROADCAST},
inet_addr("10.0.2.255")}, {{nla_len=11, nla_type=IFA_LABEL}, "enp0s3"},
{{nla_len=8, nla_type=IFA_FLAGS}, IFA_F_NOPREFIXROUTE}, {{nla_len=20,
nla_type=IFA_CACHEINFO}, {ifa_prefered=86202, ifa_valid=86202, cstamp=1132,
tstamp=1226}}]}], 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=1619027210, pid=3186}, {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=195, tstamp=195}},
{{nla_len=8, nla_type=IFA_FLAGS}, IFA_F_PERMANENT}]}, {{len=72,
type=RTM_NEWADDR, flags=NLM_F_MULTI, seq=1619027210, pid=3186},
{ifa_family=AF_INET6, ifa_prefixlen=64, ifa_flags=IFA_F_PERMANENT,
ifa_scope=RT_SCOPE_LINK, ifa_index=if_nametoindex("enp0s3")}, [{{nla_len=20,
nla_type=IFA_ADDRESS}, inet_pton(AF_INET6, "fe80::36ad:1a3a:8ee6:f329")},
{{nla_len=20, nla_type=IFA_CACHEINFO}, {ifa_prefered=4294967295,
ifa_valid=4294967295, cstamp=1131, tstamp=1345}}, {{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={{len=20, type=NLMSG_DONE,
flags=NLM_F_MULTI, seq=1619027210, pid=3186}, 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(3000),
sin_addr=inet_addr("127.0.0.1")}, 16) = 0
listen(9, 100)                          = 0
getsockname(9, {sa_family=AF_INET, sin_port=htons(3000),
sin_addr=inet_addr("127.0.0.1")}, [128->16]) = 0
```

```
getsockname(9, {sa_family=AF_INET, sin_port=htons(3000),
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
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
read(0, create 10
"create 10\n", 1024)            = 10
clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f25500dac50) = 3191
poll([{fd=8, events=POLLIN}], 1, 0)     = 1 ([{fd=8, revents=POLLIN}])
read(8, "\1\0\0\0\0\0\0\0", 8)          = 8
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
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(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(6, "\1\0\0\0\0\0\0\0", 8)         = 8
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
write(1, "OK: 3191\n", 9OK: 3191
)               = 9
read(0, exec 10 start
"exec 10 start\n", 1024)        = 14
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: start\n", 10OK: start
)             = 10
read(0, exec 10 stop
"exec 10 stop\n", 1024)         = 13
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: stop\n", 9OK: stop
)               = 9
read(0, exec 10 time
"exec 10 time\n", 1024)         = 13
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: 4577 ms\n", 12OK: 4577 ms
)           = 12
read(0, exit
"exit\n", 1024)                 = 5
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)
kill(3191, SIGTERM)                     = 0
kill(3191, SIGKILL)                     = 0
write(1, "OK\n", 3OK
)                     = 3
write(4, "\1\0\0\0\0\0\0\0", 8)         = 8
write(4, "\1\0\0\0\0\0\0\0", 8)         = 8
poll([{fd=3, events=POLLIN}], 1, -1)    = 1 ([{fd=3, revents=POLLIN}])
read(3, "\1\0\0\0\0\0\0\0", 8)          = 8
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
close(7)                                = 0
close(6)                                = 0
close(5)                                = 0
close(4)                                = 0
close(3)                                = 0
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_KILLED, si_pid=3191, si_uid=1000,
si_status=SIGTERM, si_utime=0, si_stime=0} ---
lseek(0, -1, SEEK_CUR)                  = -1 ESPIPE (Illegal seek)
exit_group(0)                           = ?
+++ exited with 0 +++
```

Выводы

В результате данной лабораторной работы я научился работать с технологией очереди сообщений, создающие и связывающие процессы в определенные топологии, понимать клиент-серверную архитектуру, читать документацию и осваивать новые библиотеки (zmq) .