Московский авиационный институт (национальный исследовательский университет)

Институт №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование» Дисциплина «Операционные системы»

Лабораторная работа №6-8

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

Студент: Будникова В.П.

Группа: М8О-207Б-19

Преподаватель: Миронов Е. С.

Дата:

Оценка:

Москва, 2020

Цель работы: Целью является приобретение практических навыков в управлении серверами сообщений, применение отложенных вычислений интеграция программных систем друг с другом.

Задача: Реализовать распределенную систему по асинхронной обработке запросов. В данной распределенной 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 технологии очередей сообщений. Также в данной системе необходимо предусмотреть проверку доступности узлов в соответствии с вариантом. При убийстве («kill -9») любого вычислительного узла система должна пытаться максимально сохранять свою работоспособность, а именно все дочерние узлы убитого узла могут стать недоступными, но родительские узлы должны сохранить свою работоспособность.

Вариант(26):

Топология:

Все вычислительные узлы хранятся в бинарном дереве поиска. [parent] — является необязательным параметром.

Тип команд для вычислительного узла:

Формат команды: exec id n k_1 ... k_n
id – целочисленный идентификатор вычислительного узла, на который отправляется команда n – количество складываемых чисел (от 1 до 10^8)
k_1 ... k_n – складываемые числа

Тип проверки доступности узла:

Формат команды: ping id

Команда проверяет доступность конкретного узла. Если узла нет, то необходимо выводить ошибку: «Error: Not found»

Листинг программы

server.c

```
#include <iostream>
#include <string>
#include <zmq.h>
#include <unistd.h>
#include <assert.h>
#include <errno.h>
#include <time.h>
#include <cstdlib>
#include <thread>

enum Instruction{
    CREATE,
    EXEC,
    PING,
    REMOVE
};

struct Message {
    int id;
    int com;
    bool kill;
    int num;
    int size;
};

void Push(Message * mes, void * sock_push) {
    zmq_msg_t zmqMessage;
    assert(zmq_msg_init_size(&zmqMessage, sizeof(Message)) != -1);
    assert(memcpy(zmq_msg_data(&zmqMessage), mes, sizeof(Message)) != NULL);
    if (zmq_msg_send(&zmqMessage, sock_push, 0) == -1) {
        return;
    }
    assert(zmq_msg_close(&zmqMessage) != -1);
}

int main() {
    int id;
    int count_mes = 0;
    bool fl = false;
    void * context_push = zmq_ctx_new();
    assert (context_push != NULL);
    void * sock_push = zmq_socket(context_push, ZMQ_PUSH);
    assert (sock_push != NULL);
    void *context_pull = zmq_ctx_new();
    assert (context_pull != NULL);
    void * sock_pull = zmq_socket(context_pull, ZMQ_PULL);
    assert (sock_pull != NULL);
    std::thread thread([&fl, &count_mes](void * sock_pull) {
```

```
    while (!fl || count_mes > 0) {
        if (count_mes > 0) {
            zmq_msg_t message_from_client;
            assert(zmq_msg_init(&message_from_client) != -1);
            if (zmq_msg_recv(&message_from_client, sock_pull, 0) != -1) {
                char * mes = (char *)zmq_msg_data(&message_from_client);
                std::cout << "Message from client: " << mes << std::endl;
                --count_mes;
                assert(zmq_msg_close(&message_from_client) != -1);
            }
        }
    }
}, sock_pull);
int r_push;
int r_pull;
bool parent = false;
        std::cout << "Starting server..." << std::endl;
std::string message;
while (std::cin >> message && message != "break") {
    Message m;
    if (message == "create" && !parent) {
        ++count_mes;
        parent = true;
        m.com = CREATE;
        std::cin >> m.id;
        id = m.id;
        m.kill = false;
        std::string port = "tcp://*:" + std::to_string(4000 + m.id);
        assert(zmq_bind(sock_push, port.c_str()) != -1);
        assert(zmq_bind(sock_pull, "tcp://*:3999") != -1);
        int t = 1000;
        assert(zmq_setsockopt(sock_push, ZMQ_SNDTIMEO, &t, sizeof(t)) != -1);
        int id_fork = fork();
        if (id_fork == -1) {
            std::cout << "error fork" << m.id << std::endl;
            exit(EXIT_FAILURE);
        } else if (id_fork == 0) {
            execl("client", std::to_string(m.id).c_str(), NULL);
        } else {
            Push(&m, sock_push);
        }
    } else
    if (message == "create" && parent) {
        ++count_mes;
        m.com = CREATE;
        std::cin >> m.id;
        m.kill = false;
        Push(&m, sock_push);
    } else
    if (message == "exec") {
        ++count_mes;
        m.com = EXEC;
```

```
            std::cin >> m.id >> m.size;
            m.kill = false;
            for (int i = 0; i < m.size; ++i) {
                std::cin >> m.num;
                Push(&m, sock_push);
            }
        } else
        if (message == "ping") {
            ++count_mes;
            m.com = PING;
            std::cin >> m.id;
            m.kill = false;
            Push(&m, sock_push);
        } else
        if (message == "remove") {
            ++count_mes;
            m.com = REMOVE;
            std::cin >> m.id;
            m.kill = false;
            Push(&m, sock_push);
        } else {
            std::cout << "Incorrect command entered" << std::endl;
        }
    }


    Message m;
    m.kill = true;
    zmq_msg_t zmqMessage;
    assert(zmq_msg_init_size(&zmqMessage, sizeof(Message)) != -1);
    assert(memcpy(zmq_msg_data(&zmqMessage), &m, sizeof(Message)) != NULL);
    assert(zmq_msg_send(&zmqMessage, sock_push, 0) != -1);
    assert(zmq_msg_close(&zmqMessage) != -1);

    fl = true;
    thread.join();

    assert(zmq_close(sock_push) != -1);
    assert(zmq_ctx_destroy(context_push) != -1);
    assert(zmq_close(sock_pull) != -1);
    assert(zmq_ctx_destroy(context_pull) != -1);

    return 0;
}
```

client.c

```
#include <iostream>
#include <string>
#include <zmq.h>
#include <unistd.h>
```

```
#include <assert.h>
#include <cstdlib>
#include <sys/types.h>

enum Instruction{
    CREATE,
    EXEC,
    PING,
    REMOVE
};

struct Message {
    int id;
    int com;
    bool kill;
    int num;
    int size;
};

void * sock_push;

void Error(std::string str, void * sock_push) {
    std::string mes_to_server = "Error: " + str + "\0";
    zmq_msg_t zmqMessage;
    assert(zmq_msg_init_size(&zmqMessage, sizeof(char) * mes_to_server.size() + 1) != -1);
    assert(memcpy(zmq_msg_data(&zmqMessage), mes_to_server.c_str() , sizeof(char) *
mes_to_server.size() + 1) != NULL);
    assert(zmq_msg_send(&zmqMessage, sock_push, 0) != -1);
    assert(zmq_msg_close(&zmqMessage) != -1);
}
void Ok(int id, void * sock_push) {
    std::string mes_to_server = "Ok: " + std::to_string(id) + "\0";
    zmq_msg_t zmqMessage;
    assert(zmq_msg_init_size(&zmqMessage, sizeof(char) * mes_to_server.size() + 1) != -1);
    assert(memcpy(zmq_msg_data(&zmqMessage), mes_to_server.c_str() , sizeof(char) *
mes_to_server.size() + 1) != NULL);
    assert(zmq_msg_send(&zmqMessage, sock_push, 0) != -1);
    assert(zmq_msg_close(&zmqMessage) != -1);
}
void Ok(int id, int rez, void * sock_push) {
    std::string mes_to_server = "Ok: " + std::to_string(id) + " : " + std::to_string(rez) + "\0";
    zmq_msg_t zmqMessage;
    assert(zmq_msg_init_size(&zmqMessage, sizeof(char) * mes_to_server.size() + 1) != -1);
    assert(memcpy(zmq_msg_data(&zmqMessage), mes_to_server.c_str() , sizeof(char) *
mes_to_server.size() + 1) != NULL);
    assert(zmq_msg_send(&zmqMessage, sock_push, 0) != -1);
    assert(zmq_msg_close(&zmqMessage) != -1);
}
void Ok(void * sock_push) {
    std::string mes_to_server = "Ok\0";
    zmq_msg_t zmqMessage;
    assert(zmq_msg_init_size(&zmqMessage, sizeof(char) * mes_to_server.size() + 1) != -1);
    assert(memcpy(zmq_msg_data(&zmqMessage), mes_to_server.c_str() , sizeof(char) *
mes_to_server.size() + 1) != NULL);
    assert(zmq_msg_send(&zmqMessage, sock_push, 0) != -1);
    assert(zmq_msg_close(&zmqMessage) != -1);
}
void Ok_un(void * sock_push) {
    std::string mes_to_server = "Error: Node is unavailable\0";
    zmq_msg_t zmqMessage;
    assert(zmq_msg_init_size(&zmqMessage, sizeof(char) * mes_to_server.size() + 1) != -1);
    assert(memcpy(zmq_msg_data(&zmqMessage), mes_to_server.c_str() , sizeof(char) *
mes_to_server.size() + 1) != NULL);
    assert(zmq_msg_send(&zmqMessage, sock_push, 0) != -1);
    assert(zmq_msg_close(&zmqMessage) != -1);
}
void Push(Message * mes, void * sock_push1, void * context_push1, bool &r) {
    zmq_msg_t zmqMessage;
    assert(zmq_msg_init_size(&zmqMessage, sizeof(Message)) != -1);
    assert(memcpy(zmq_msg_data(&zmqMessage), mes, sizeof(Message)) != NULL);
    int t = 1000;
    assert(zmq_setsockopt(sock_push1, ZMQ_SNDTIMEO, &t, sizeof(t)) != -1);
    if(zmq_msg_send(&zmqMessage, sock_push1, 0) == -1) {
        assert(zmq_close(sock_push1) != -1);
        assert(zmq_ctx_destroy(context_push1) != -1);
        r = false;
        Ok_un(sock_push);
    }
    assert(zmq_msg_close(&zmqMessage) != -1);
}

int main(int argc, char * argv[]) {
    int id;
    bool par = false;
    bool left = false;
    bool right = false;
    int id_left;
    int id_right;
    if (argc < 1) {
        std::cout << "ERROR";
        return 1;
    } else {
        id = atoi(argv[0]);
    }
    void * context_pull = zmq_ctx_new();
    assert (context_pull != NULL);
    void * sock_pull = zmq_socket(context_pull, ZMQ_PULL);
    assert (sock_pull != NULL);
    std::string port = "tcp://127.0.0.1:" + std::to_string(4000 + id);
    assert(zmq_connect(sock_pull, port.c_str()) != -1);
    void * context_push = zmq_ctx_new();
    assert (context_push != NULL);
    sock_push = zmq_socket(context_push, ZMQ_PUSH);
    assert (sock_push != NULL);
```

```
assert(zmq_connect(sock_push, "tcp://127.0.0.1:3999") != -1);

void * context_push_right;
void * sock_push_right;

void * context_push_left;
void * sock_push_left;

while(!false) {
    zmq_msg_t message;
    assert(zmq_msg_init(&message) != -1);
    assert(zmq_msg_recv(&message, sock_pull, 0) != -1);
    Message * mes = (Message *)zmq_msg_data(&message);
    assert(zmq_msg_close(&message) != -1);
    if (mes->kill) {
        if (left) {
            Push(mes, sock_push_left, context_push_left, right);
            assert(zmq_close(sock_push_left) != -1);
            assert(zmq_ctx_destroy(context_push_left) != -1);
        }
        if (right) {
            Push(mes, sock_push_right, context_push_right, right);
            assert(zmq_close(sock_push_right) != -1);
            assert(zmq_ctx_destroy(context_push_right) != -1);
        }
        break;
    }
    if (mes->id == id) {
        if (mes->com == CREATE) {
            if (par) {
                Error("Already exists", sock_push);
            } else {
                par = true;
                Ok(getpid(), sock_push);
            }
        } else
        if (mes->com == EXEC) {
            int rez = mes->num;
            for (int i = 0; i < mes->size - 1; ++i) {
                zmq_msg_t message1;
                assert(zmq_msg_init(&message1) != -1);
                assert(zmq_msg_recv(&message1, sock_pull, 0) != -1);
                Message * mes1 = (Message *)zmq_msg_data(&message1);
                assert(zmq_msg_close(&message1) != -1);
                rez += mes1->num;
            }
            sleep(5);
            Ok(getpid(), rez, sock_push);
        } else
        if (mes->com == PING) {
            Ok(getpid(), sock_push);
        } else
        if (mes->com == REMOVE) {
            std::cout << "eror remove" << std::endl;
        }
    } else if (mes->id > id) {
        if (mes->com == CREATE && !right) {
            right = true;
            id_right = mes->id;
            assert((context_push_right = zmq_ctx_new()) != NULL);
            assert((sock_push_right = zmq_socket(context_push_right, ZMQ_PUSH)) != NULL);
            std::string port_right = "tcp://*:" + std::to_string(4000 + id_right);
            assert(zmq_bind(sock_push_right, port_right.c_str()) != -1);
            int id_fork = fork();
            if (id_fork == -1) {
                std::cout << "error fork" << id << std::endl;
                exit(EXIT_FAILURE);
            } else if (id_fork == 0) {
                execl("client", std::to_string(id_right).c_str(), NULL);
            } else {
                Push(mes, sock_push_right, context_push_right, right);
            }
        } else
        if (mes->com == EXEC && !right) {
            Error("Not found", sock_push);
        } else
        if (mes->com == PING && !right) {
            Error("Not found", sock_push);
        } else
        if (mes->com == REMOVE) {
            if (!right) {
                Error("Not found", sock_push);
            } else {
                if (id_right == mes->id) {
                    mes->kill = true;
                    right = false;
                    Push(mes, sock_push_right, context_push_right, right);
                    assert(zmq_close(sock_push_right) != -1);
                    assert(zmq_ctx_destroy(context_push_right) != -1);
                    Ok(sock_push);
                } else {
                    Push(mes, sock_push_right, context_push_right, right);
                }
            }
        } else {
            Push(mes, sock_push_right, context_push_right, right);
        }
    } else {
        if (mes->com == CREATE && !left) {
            left = true;
            id_left = mes->id;
            assert((context_push_left = zmq_ctx_new()) != NULL);
            assert((sock_push_left = zmq_socket(context_push_left, ZMQ_PUSH)) != NULL);
            std::string port_left = "tcp://*:" + std::to_string(4000 + id_left);
```

```
                assert(zmq_bind(sock_push_left, port_left.c_str()) != -1);
                int id_fork = fork();
                if (id_fork == -1) {
                    std::cout << "error fork" << id << std::endl;
                    exit(EXIT_FAILURE);
                } else if (id_fork == 0) {
                    execl("client", std::to_string(id_left).c_str(), NULL);
                } else {
                    Push(mes, sock_push_left, context_push_left, left);
                }
            } else
            if (mes->com == EXEC && !left) {
                Error("Not found", sock_push);
            } else
            if (mes->com == PING && !left) {
                Error("Not found", sock_push);
            } else
            if (mes->com == REMOVE) {
                if (!left) {
                    Error("Not found", sock_push);
                } else {
                    if (id_left == mes->id) {
                        mes->kill = true;
                        right = false;
                        Push(mes, sock_push_left, context_push_left, left);
                        assert(zmq_close(sock_push_left) != -1);
                        assert(zmq_ctx_destroy(context_push_left) != -1);
                        Ok(sock_push);
                    } else {
                        Push(mes, sock_push_left, context_push_left, left);
                    }
                }
            } else {
                Push(mes, sock_push_left, context_push_left, left);
            }
        }
    }
    assert(zmq_close(sock_push) != -1);
    assert(zmq_ctx_destroy(context_push) != -1);
    assert(zmq_close(sock_pull) != -1);
    assert(zmq_ctx_destroy(context_pull) != -1);
    return 0;
}
```

Тесты и протокол исполнения

```
./server
Starting server...
create 10
Message from client: Ok: 1731
create 20
Message from client: Ok: 1732
```

```
create 30
Message from client: Ok: 1734
remove 30
Message from client: Ok
ping 30
Message from client: Error: Not found
create 25
Message from client: Ok: 1735
create 50
Message from client: Ok: 1737
exec 50 5 1 2 3 4 5
ping 25
Message from client: Ok: 1735
Message from client: Ok: 1737 : 15
ping 30
Message from client: Error: Not found
remove 25
Message from client: Ok
ping 25
Message from client: Error: Not found
ping 50
Message from client: Error: Not found
break
Lera:lab6-8 valeriabudnikova$ ps
  PID TTY           TIME CMD
  488 ttys004    0:00.27 -bash
```

Вывод strace

```
strace ./server < test2.txt > stracelab6.txt
execve("./server", ["./server"], 0x7fffadc9e0f0 /* 67 vars */) = 0
brk(NULL)                               = 0x556220e4f000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd5fdd20b0) = -1 EINVAL (Недопустимый аргумент)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=118993, ...}) = 0
mmap(NULL, 118993, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f7da6916000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/libzmq.so.5", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0`z\1\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=675776, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f7da6914000
mmap(NULL, 678128, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da686e000
mmap(0x7f7da6884000, 430080, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x16000) = 0x7f7da6884000
mmap(0x7f7da68ed000, 126976, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x7f000) = 0x7f7da68ed000
mmap(0x7f7da690c000, 32768, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x9d000) = 0x7f7da690c000
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\0O\305\3743\364B\2216\244\224\306@\261\23\327o"..., 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\0O\305\3743\364B\2216\244\224\306@\261\23\327o"..., 68,
824) = 68
mmap(NULL, 140408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da684b000
mmap(0x7f7da6852000, 69632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x7000) = 0x7f7da6852000
mmap(0x7f7da6863000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x18000) = 0x7f7da6863000
mmap(0x7f7da6868000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1c000) = 0x7f7da6868000
mmap(0x7f7da686a000, 13432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f7da686a000
close(3)                          = 0
openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\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) = 0x7f7da666a000
mprotect(0x7f7da6700000, 1286144, PROT_NONE) = 0
mmap(0x7f7da6700000, 983040, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x96000) = 0x7f7da6700000
mmap(0x7f7da67f0000, 299008, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x186000) = 0x7f7da67f0000
mmap(0x7f7da683a000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1cf000) = 0x7f7da683a000
mmap(0x7f7da6848000, 10240, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f7da6848000
close(3)                          = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\3405\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=104984, ...}) = 0
mmap(NULL, 107592, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da664f000
mmap(0x7f7da6652000, 73728, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x3000) = 0x7f7da6652000
mmap(0x7f7da6664000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x15000) = 0x7f7da6664000
mmap(0x7f7da6668000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x18000) = 0x7f7da6668000
close(3)                          = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360q\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\363\377?\332\200\270\27\304d\245n\355Y\377\t\334"..., 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\363\377?\332\200\270\27\304d\245n\355Y\377\t\334"..., 68,
880) = 68
mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da645d000
mprotect(0x7f7da6482000, 1847296, PROT_NONE) = 0
mmap(0x7f7da6482000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x25000) = 0x7f7da6482000
mmap(0x7f7da65fa000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x19d000) = 0x7f7da65fa000
mmap(0x7f7da6645000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1e7000) = 0x7f7da6645000
mmap(0x7f7da664b000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f7da664b000
close(3)                          = 0
openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/libsodium.so.23", O_RDONLY|O_CLOEXEC) = 3
```

```
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\200\302\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=355016, ...}) = 0
mmap(NULL, 357384, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da6405000
mmap(0x7f7da6411000, 229376, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0xc000) = 0x7f7da6411000
mmap(0x7f7da6449000, 73728, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x44000) = 0x7f7da6449000
mmap(0x7f7da645b000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x55000) = 0x7f7da645b000
close(3)                        = 0
openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/libpgm-5.2.so.0", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\240L\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=302056, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f7da6403000
mmap(NULL, 321584, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da63b4000
mmap(0x7f7da63b8000, 163840, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x4000) = 0x7f7da63b8000
mmap(0x7f7da63e0000, 118784, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x2c000) = 0x7f7da63e0000
mmap(0x7f7da63fd000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x48000) = 0x7f7da63fd000
mmap(0x7f7da63ff000, 14384, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f7da63ff000
close(3)                        = 0
openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/libnorm.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\257\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=690344, ...}) = 0
mmap(NULL, 1420000, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da6259000
mmap(0x7f7da6263000, 421888, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0xa000) = 0x7f7da6263000
mmap(0x7f7da62ca000, 217088, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x71000) = 0x7f7da62ca000
mmap(0x7f7da62ff000, 16384, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0xa5000) = 0x7f7da62ff000
mmap(0x7f7da6303000, 723680, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f7da6303000
close(3)                        = 0
openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/libgssapi_krb5.so.2", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\321\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=309712, ...}) = 0
mmap(NULL, 312128, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da620c000
mmap(0x7f7da6217000, 204800, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0xb000) = 0x7f7da6217000
mmap(0x7f7da6249000, 49152, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x3d000) = 0x7f7da6249000
mmap(0x7f7da6255000, 16384, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x48000) = 0x7f7da6255000
close(3)                        = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\300\363\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=1369352, ...}) = 0
mmap(NULL, 1368336, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da60bd000
mmap(0x7f7da60cc000, 684032, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0xf000) = 0x7f7da60cc000
mmap(0x7f7da6173000, 618496, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xb6000) = 0x7f7da6173000
mmap(0x7f7da620a000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x14c000) = 0x7f7da620a000
close(3)                        = 0
```

```
openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/libkrb5.so.3", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0 ?\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=902016, ...}) = 0
mmap(NULL, 904640, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da5fe0000
mprotect(0x7f7da6002000, 700416, PROT_NONE) = 0
mmap(0x7f7da6002000, 397312, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x22000) = 0x7f7da6002000
mmap(0x7f7da6063000, 299008, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x83000) = 0x7f7da6063000
mmap(0x7f7da60ad000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0xcc000) = 0x7f7da60ad000
close(3)                        = 0
openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/libk5crypto.so.3", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\240D\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=191040, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f7da5fde000
mmap(NULL, 196696, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da5fad000
mprotect(0x7f7da5fb1000, 172032, PROT_NONE) = 0
mmap(0x7f7da5fb1000, 114688, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x4000) = 0x7f7da5fb1000
mmap(0x7f7da5fcd000, 53248, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x20000) = 0x7f7da5fcd000
mmap(0x7f7da5fdb000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x2d000) = 0x7f7da5fdb000
mmap(0x7f7da5fdd000, 88, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f7da5fdd000
close(3)                        = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libcom_err.so.2", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\200$\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=22600, ...}) = 0
mmap(NULL, 24744, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da5fa6000
mmap(0x7f7da5fa8000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x2000) = 0x7f7da5fa8000
mmap(0x7f7da5faa000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x4000) = 0x7f7da5faa000
mmap(0x7f7da5fab000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x4000) = 0x7f7da5fab000
close(3)                        = 0
openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/libkrb5support.so.0", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\3605\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=56096, ...}) = 0
mmap(NULL, 58344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da5f97000
mmap(0x7f7da5f9a000, 28672, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x3000) = 0x7f7da5f9a000
mmap(0x7f7da5fa1000, 12288, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xa000) = 0x7f7da5fa1000
mmap(0x7f7da5fa4000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0xc000) = 0x7f7da5fa4000
close(3)                        = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libkeyutils.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0@\"\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=22600, ...}) = 0
mmap(NULL, 24592, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da5f90000
mmap(0x7f7da5f92000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x2000) = 0x7f7da5f92000
mmap(0x7f7da5f94000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x4000) = 0x7f7da5f94000
mmap(0x7f7da5f95000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x4000) = 0x7f7da5f95000
```

```
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libresolv.so.2", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0 G\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=101320, ...}) = 0
mmap(NULL, 113280, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da5f74000
mprotect(0x7f7da5f78000, 81920, PROT_NONE) = 0
mmap(0x7f7da5f78000, 65536, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x4000) = 0x7f7da5f78000
mmap(0x7f7da5f88000, 12288, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x14000) = 0x7f7da5f88000
mmap(0x7f7da5f8c000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x17000) = 0x7f7da5f8c000
mmap(0x7f7da5f8e000, 6784, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f7da5f8e000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libdl.so.2", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0 \22\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=18816, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f7da5f72000
mmap(NULL, 20752, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da5f6c000
mmap(0x7f7da5f6d000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1000) = 0x7f7da5f6d000
mmap(0x7f7da5f6f000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x3000) = 0x7f7da5f6f000
mmap(0x7f7da5f70000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x3000) = 0x7f7da5f70000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f7da5f6a000
arch_prctl(ARCH_SET_FS, 0x7f7da5f6b180) = 0
mprotect(0x7f7da6645000, 12288, PROT_READ) = 0
mprotect(0x7f7da5f70000, 4096, PROT_READ) = 0
mprotect(0x7f7da5f8c000, 4096, PROT_READ) = 0
mprotect(0x7f7da5f95000, 4096, PROT_READ) = 0
mprotect(0x7f7da5fa4000, 4096, PROT_READ) = 0
mprotect(0x7f7da6868000, 4096, PROT_READ) = 0
mprotect(0x7f7da5fab000, 4096, PROT_READ) = 0
mprotect(0x7f7da5fdb000, 4096, PROT_READ) = 0
mprotect(0x7f7da60ad000, 57344, PROT_READ) = 0
mprotect(0x7f7da620a000, 4096, PROT_READ) = 0
mprotect(0x7f7da6255000, 8192, PROT_READ) = 0
mprotect(0x7f7da6668000, 4096, PROT_READ) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f7da5f68000
mprotect(0x7f7da683a000, 45056, PROT_READ) = 0
mprotect(0x7f7da62ff000, 12288, PROT_READ) = 0
mprotect(0x7f7da63fd000, 4096, PROT_READ) = 0
mprotect(0x7f7da645b000, 4096, PROT_READ) = 0
mprotect(0x7f7da690c000, 28672, PROT_READ) = 0
mprotect(0x55621f289000, 4096, PROT_READ) = 0
mprotect(0x7f7da6961000, 4096, PROT_READ) = 0
munmap(0x7f7da6916000, 118993)          = 0
set_tid_address(0x7f7da5f6b450)         = 55477
set_robust_list(0x7f7da5f6b460, 24)     = 0
rt_sigaction(SIGRTMIN, {sa_handler=0x7f7da6852bf0, sa_mask=[], sa_flags=SA_RESTORER|
SA_SIGINFO, sa_restorer=0x7f7da68603c0}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x7f7da6852c90, sa_mask=[], sa_flags=SA_RESTORER|
SA_RESTART|SA_SIGINFO, sa_restorer=0x7f7da68603c0}, 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)                               = 0x556220e4f000
brk(0x556220e70000)                     = 0x556220e70000
futex(0x7f7da68486bc, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f7da68486c8, FUTEX_WAKE_PRIVATE, 2147483647) = 0
openat(AT_FDCWD, "/sys/devices/system/cpu/online", O_RDONLY|O_CLOEXEC) = 3
read(3, "0-3\n", 8192)                  = 4
close(3)                                = 0
openat(AT_FDCWD, "/sys/devices/system/cpu", O_RDONLY|O_NONBLOCK|O_CLOEXEC|
O_DIRECTORY) = 3
fstat(3, {st_mode=S_IFDIR|0755, st_size=0, ...}) = 0
getdents64(3, /* 22 entries */, 32768)  = 656
getdents64(3, /* 0 entries */, 32768)   = 0
close(3)                                = 0
getpid()                                = 55477
sched_getaffinity(55477, 128, [0, 1, 2, 3]) = 8
openat(AT_FDCWD, "/etc/nsswitch.conf", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=545, ...}) = 0
read(3, "# /etc/nsswitch.conf\n#\n# Example"..., 4096) = 545
read(3, "", 4096)                       = 0
close(3)                                = 0
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=118993, ...}) = 0
mmap(NULL, 118993, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f7da6916000
close(3)                                = 0
stat("/usr/lib", {st_mode=S_IFDIR|0755, st_size=12288, ...}) = 0
munmap(0x7f7da6916000, 118993)          = 0
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=118993, ...}) = 0
mmap(NULL, 118993, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f7da6916000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libnss_files.so.2", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\3005\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=51832, ...}) = 0
mmap(NULL, 79672, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da5f54000
mmap(0x7f7da5f57000, 28672, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x3000) = 0x7f7da5f57000
mmap(0x7f7da5f5e000, 8192, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xa000) = 0x7f7da5f5e000
mmap(0x7f7da5f60000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0xb000) = 0x7f7da5f60000
mmap(0x7f7da5f62000, 22328, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f7da5f62000
close(3)                                = 0
mprotect(0x7f7da5f60000, 4096, PROT_READ) = 0
munmap(0x7f7da6916000, 118993)          = 0
openat(AT_FDCWD, "/etc/protocols", O_RDONLY|O_CLOEXEC) = 3
lseek(3, 0, SEEK_CUR)                   = 0
fstat(3, {st_mode=S_IFREG|0644, st_size=2932, ...}) = 0
read(3, "# Internet (IP) protocols\n#\n# Up"..., 4096) = 2932
lseek(3, 0, SEEK_CUR)                   = 2932
read(3, "", 4096)                       = 0
close(3)                                = 0
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("\x6a\xbd\x80\xeb\xfa\x36\xfe\x1e\x73\xbd\x27\x54\xeb\xd9\xf5\x0e", 16, 0) = 16
getrandom("\x78\x5e\x82\x41\x3a\xd4\xda\x79\x27\xcb\x3b\xf5\x16\xb6\x6a\x3e", 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=551951712, u64=93879947107680}}) = 0
epoll_ctl(5, EPOLL_CTL_MOD, 4, {EPOLLIN, {u32=551951712, u64=93879947107680}}) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7f7da5753000
mprotect(0x7f7da5754000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7f7da5f52d30, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|
CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, parent_tid=[55478], tls=0x7f7da5f53700, child_tidptr=0x7f7da5f539d0) =
55478
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=551969376, u64=93879947125344}}) = 0
epoll_ctl(7, EPOLL_CTL_MOD, 6, {EPOLLIN, {u32=551969376, u64=93879947125344}}) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7f7da4f52000
mprotect(0x7f7da4f53000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7f7da5751d30, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|
CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, parent_tid=[55479], tls=0x7f7da5752700, child_tidptr=0x7f7da57529d0) =
55479
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
eventfd2(0, EFD_CLOEXEC)             = 9
fcntl(9, F_GETFL)                    = 0x2 (flags O_RDWR)
fcntl(9, F_SETFL, O_RDWR|O_NONBLOCK)   = 0
fcntl(9, F_GETFL)                    = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(9, F_SETFL, O_RDWR|O_NONBLOCK)   = 0
eventfd2(0, EFD_CLOEXEC)             = 10
fcntl(10, F_GETFL)                   = 0x2 (flags O_RDWR)
fcntl(10, F_SETFL, O_RDWR|O_NONBLOCK)  = 0
fcntl(10, F_GETFL)                   = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(10, F_SETFL, O_RDWR|O_NONBLOCK)  = 0
epoll_create1(EPOLL_CLOEXEC)          = 11
epoll_ctl(11, EPOLL_CTL_ADD, 10, {0, {u32=551989888, u64=93879947145856}}) = 0
epoll_ctl(11, EPOLL_CTL_MOD, 10, {EPOLLIN, {u32=551989888, u64=93879947145856}}) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7f7da4751000
mprotect(0x7f7da4752000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7f7da4f50d30, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|
CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, parent_tid=[55480], tls=0x7f7da4f51700, child_tidptr=0x7f7da4f519d0) =
55480
eventfd2(0, EFD_CLOEXEC)             = 12
fcntl(12, F_GETFL)                   = 0x2 (flags O_RDWR)
fcntl(12, F_SETFL, O_RDWR|O_NONBLOCK)  = 0
fcntl(12, F_GETFL)                   = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(12, F_SETFL, O_RDWR|O_NONBLOCK)  = 0
```

```
epoll_create1(EPOLL_CLOEXEC)          = 13
epoll_ctl(13, EPOLL_CTL_ADD, 12, {0, {u32=551991936, u64=93879947147904}}) = 0
epoll_ctl(13, EPOLL_CTL_MOD, 12, {EPOLLIN, {u32=551991936, u64=93879947147904}}) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7f7da3f50000
mprotect(0x7f7da3f51000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7f7da474fd30, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|
CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, parent_tid=[55481], tls=0x7f7da4750700, child_tidptr=0x7f7da47509d0) =
55481
eventfd2(0, EFD_CLOEXEC)              = 14
fcntl(14, F_GETFL)                    = 0x2 (flags O_RDWR)
fcntl(14, F_SETFL, O_RDWR|O_NONBLOCK)   = 0
fcntl(14, F_GETFL)                    = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(14, F_SETFL, O_RDWR|O_NONBLOCK)   = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7f7da374f000
mprotect(0x7f7da3750000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7f7da3f4ed30, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|
CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, parent_tid=[55482], tls=0x7f7da3f4f700, child_tidptr=0x7f7da3f4f9d0) =
55482
fstat(1, {st_mode=S_IFREG|0664, st_size=0, ...}) = 0
write(1, "Starting server...\n", 19)    = 19
fstat(0, {st_mode=S_IFREG|0664, st_size=78, ...}) = 0
read(0, "create 1\ncreate 2\ncreate 3\ncreat"..., 4096) = 78
//выделю только первые значимые системные вызовы
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
socket(AF_INET, SOCK_STREAM|SOCK_CLOEXEC, IPPROTO_TCP) = 15
setsockopt(15, SOL_SOCKET, SO_REUSEADDR, [1], 4) = 0
bind(15, {sa_family=AF_INET, sin_port=htons(4001), sin_addr=inet_addr("0.0.0.0")}, 16) = 0
listen(15, 100)                       = 0
getsockname(15, {sa_family=AF_INET, sin_port=htons(4001), sin_addr=inet_addr("0.0.0.0")}, [128->16])
= 0
getsockname(15, {sa_family=AF_INET, sin_port=htons(4001), sin_addr=inet_addr("0.0.0.0")}, [128->16])
= 0
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
write(8, "\1\0\0\0\0\0\0\0", 8)         = 8
poll([{fd=14, events=POLLIN}], 1, 0)    = 0 (Timeout)
socket(AF_INET, SOCK_STREAM|SOCK_CLOEXEC, IPPROTO_TCP) = 16
setsockopt(16, SOL_SOCKET, SO_REUSEADDR, [1], 4) = 0
bind(16, {sa_family=AF_INET, sin_port=htons(3999), sin_addr=inet_addr("0.0.0.0")}, 16) = 0
listen(16, 100)                       = 0
getsockname(16, {sa_family=AF_INET, sin_port=htons(3999), sin_addr=inet_addr("0.0.0.0")}, [128->16])
= 0
getsockname(16, {sa_family=AF_INET, sin_port=htons(3999), sin_addr=inet_addr("0.0.0.0")}, [128->16])
= 0
write(12, "\1\0\0\0\0\0\0\0", 8)        = 8
write(14, "\1\0\0\0\0\0\0\0", 8)        = 8
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f7da5f6b450) = 55483
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, 1000)  = 1 ([{fd=8, revents=POLLIN}])
read(8, "\1\0\0\0\0\0\0\0", 8)          = 8
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
write(6, "\1\0\0\0\0\0\0\0", 8)        = 8
poll([{fd=8, events=POLLIN}], 1, 0)    = 0 (Timeout)
write(6, "\1\0\0\0\0\0\0\0", 8)        = 8
write(6, "\1\0\0\0\0\0\0\0", 8)        = 8
write(6, "\1\0\0\0\0\0\0\0", 8)        = 8
write(6, "\1\0\0\0\0\0\0\0", 8)        = 8
poll([{fd=8, events=POLLIN}], 1, 0)    = 0 (Timeout)
write(6, "\1\0\0\0\0\0\0\0", 8)        = 8
read(0, "", 4096)                 = 0
write(6, "\1\0\0\0\0\0\0\0", 8)        = 8
futex(0x7f7da3f4f9d0, FUTEX_WAIT, 55482, NULL) = 0
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
write(10, "\1\0\0\0\0\0\0\0", 8)       = 8
poll([{fd=9, events=POLLIN}], 1, -1)    = 1 ([{fd=9, revents=POLLIN}])
read(9, "\1\0\0\0\0\0\0\0", 8)         = 8
write(12, "\1\0\0\0\0\0\0\0", 8)       = 8
close(13)                          = 0
close(12)                          = 0
munmap(0x7f7da374f000, 8392704)        = 0
close(11)                          = 0
close(10)                          = 0
close(9)                          = 0
exit_group(0)                     = ?
+++ exited with 0 +++
```

Strace:

socket - системный возов, который создает конечную точку соединения и возвращает ее описатель, в аргументах задается протокол, который будет использоваться для соединения. Этот системный вызов используется для создания сокета.

listen - системный вызов, который задает размер очереди для сокета.

bind - системный вызов, который привязывает к сокету локальный адрес.

poll - системный вызов, который ожидает, пока один из файловых дескрипторов не начнет выполнять операции на ввод или вывод.

close - системный вызов, который закрывает файловый дескриптор, после этого дескриптор не будет ссылаться на файл и может быть(при необходимости) использован еще раз.

clone - системный вызов для создания нового процесса. Обеспечивает разделяемые адресные пространства, родительский процесс(откуда происходит вызов) не может выполняться в том же стеке, что и дочерний. родительский процесс передает указатель на пространство памяти для дочернего.

write - системный вызов, для записи в файл. Первый аргумент - файловый дескриптор. Второй аргумент - адрес, с которого начинается буфер. Третий аргумент - количество байтов, которое нужно записать в файл, на который ссылается файловый дескриптор из буфера. При удачном завершении возвращается количество байтов, которые были записаны(в случае ошибки - "-1").

Выводы

В данной лабораторной работе я научилась работать с сокетами, на основе библиотеки ZERO_MQ. При реализации я столкнулась с проблемой асинхронного вычисления, а точнее его отсутствия из за формулировки задания, так как мы не знаем какое количество чисел нам прийдется передавать, то нужно отправлять по одному число, когда число попадает в нужный вычислительный узел, оно сразу складывается с результатом, что исключает асинхронность, так как когда мы отправляем последнее число - результат вычисляется очень быстро, для моделирования асинхронности я использовала sleep(), которым задала "искусственное" время вычисления, после этого после exec проходит определенное время, за которые мы можем работать дальше с нашей программой.

Список литературы

1. Таненбаум Э., Бос Х. *Современные операционные системы. — 4-е изд.* — СПб.: Издательский дом «Питер», 2018. — С. 111 - 123

2. Поисковик Google [электронный ресурс] URL: https://google.com/ (дата обращения: 22.09.2020)