Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

**Лабораторная работа №3 по курсу**

**«Операционные системы»**

Группа: М8О-215Б-23

Студент: Самарский Я.В

Преподаватель: Миронов Е.С.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 14.11.24

Москва, 2024

**Постановка задачи**

**Вариант 9.**

Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия файла с таким именем на чтение. Стандартный поток ввода дочернего процесса переопределяется открытым файлом. Дочерний процесс читает команды из стандартного потока ввода. Дочерний процесс пишет в memory mapped file. Родительский процесс читает из memory mapped file и прочитанное выводит в свой стандартный поток вывода. Родительский и дочерний процесс должны быть представлены разными программами.

В файле записаны команды вида: «число число число». Дочерний процесс производит деление первого числа команда, на последующие числа в команде, а результат выводит в стандартный поток вывода. Если происходит деление на 0, то тогда дочерний и родительский процесс завершают свою работу. Проверка деления на 0 должна осуществляться на стороне дочернего процесса. Числа имеют тип float. Количество чисел может быть произвольным.

**Общий метод и алгоритм решения**

Использованные системные вызовы:

* pid\_t fork(void); – создает дочерний процесс.
* execl(const char \*path, const char \*arg, ...) – замена памяти процесса
* pid\_t waitpid(pid\_t pid, int \*stat\_loc, int options) - ожидание завершения дочернего процесса
* int dup2(int oldfd, int newfd) - переназначение файлового дескриптора
* int open(const char \*pathname, int flags, mode\_t mode) - открытие\создание файла
* int close(int fd) - закрыть файл
* void \*mmap(void \*addr, size\_t len, int prot, flags, int fd, \_\_off\_t offset) - отражает файлы или устройства в памяти
* int munmap(void \*addr, size\_t len) – снимает отражение файла или устройства в памяти
* int msync(void \*addr, size\_t len, int flags) – синхронизирует файл с отражением в памяти

В родительском процессе считываем имя файла. Далее этот файл открываем на чтение. Создаём memory-mapped file. Создаём дочерний процесс. Стандартный поток ввода дочернего процесса переопределяется открытым файлом. Используем execl для замены памяти процесса на программу, написанную для дочернего процесса. случае неудачи продолжит выполняться старый код, будет выведено сообщение об ошибке, и программа завершится. Дочерний процесс пишет информацию в memory-mapped file. В

В родительском процессе закрываем дескриптор файла. Ожидаем завершение дочернего процесса. Считываем из memory-mapped файла символы и выводим их пользователю.

В программе дочернего процесса считываем из входного потока числа и выполняем деление. В случае деления на 0 программа завершается с кодом 1.

**Код программы**

**main/main.cpp**

#include <iostream>

#include <unistd.h>

#include <fcntl.h>

#include <sys/wait.h>

#include "common.h"

#include <cstdlib>

#include <sys/mman.h>

int createMappedFile(const char\* filename) {

// Создаём файл (перезаписываем)

int fd = open(filename, O\_RDWR | O\_CREAT | O\_TRUNC, 0666);

if (fd == -1) {

perror("Error creating mapped file");

exit(1);

}

// Устанавливаем размер файла

if (ftruncate(fd, sizeof(SharedData)) == -1) {

perror("Error setting file size");

close(fd);

exit(1);

}

return fd;

}

int main() {

int fd1 = createMappedFile(MAPPED\_FILE1);

SharedData\* shared1 = (SharedData\*)mmap(nullptr, sizeof(SharedData),

PROT\_READ | PROT\_WRITE,

MAP\_SHARED, fd1, 0);

if (shared1 == MAP\_FAILED) {

std::cerr << "Error mapping files" << std::endl;

exit(1);

}

shared1->size = 0;

shared1->done = false;

// Первой строчкой пользователь в консоль родительского процесса вводит имя файла,

// которое будет использовано для открытия файла с таким именем на чтение

std::string name;

std::getline(std::cin, name);

const int file = open(name.c\_str(), O\_RDONLY);

// Родительский процесс создает дочерний процесс.

const pid\_t pid = fork();

if (pid == -1) {

std::cerr << "ERROR: Error while creating child process" << std::endl;

munmap(shared1, sizeof(SharedData));

close(fd1);

return 1;

}

// Дочерний процесс

if (pid == 0) {

// Стандартный поток ввода дочернего процесса переопределяется открытым файлом.

dup2(file, STDIN\_FILENO);

// Родительский и дочерний процесс должны быть представлены разными программами.

execl("./child", "./child", nullptr);

std::cerr << "ERROR: Can't execute child process" << std::endl;

return 1;

}

close(file);

// Родительский процесс ждёт завершение дочернего и выводит содержимое общей памяти в свой стандартный поток вывода.

int status;

if (waitpid(pid, &status, 0) == -1) {

std::cerr << "Can't get status of child process" << std::endl;

return 1;

}

std::string\_view output(shared1->data);

std::cout << output << std::endl;

if (WIFEXITED(status) && WEXITSTATUS(status) == 1) {

std::cout << "ERROR: Division by zero" << std::endl;

munmap(shared1, sizeof(SharedData));

close(fd1);

return 1;

}

munmap(shared1, sizeof(SharedData));

close(fd1);

return 0;

}

**child/main.cpp**

#include <iostream>

#include <sys/mman.h>

#include <fcntl.h>

#include <sstream>

#include <cstring>

#include "../main/common.h"

void syncWithMemory(std::ostringstream &oss, SharedData\* shared1) {

auto length = oss.view().size();

if (length > SHARED\_STR\_SIZE)

{

perror("Too long output");

exit(1);

}

auto charPtr = oss.view().data();

strcpy(shared1->data, charPtr);

shared1->size = length;

shared1->done = true;

msync(shared1, sizeof(SharedData), MS\_SYNC);

}

int main() {

int fd1 = open(MAPPED\_FILE1, O\_RDWR);

SharedData\* shared1 = (SharedData\*)mmap(nullptr, sizeof(SharedData),

PROT\_READ | PROT\_WRITE,

MAP\_SHARED, fd1, 0);

std::ostringstream oss;

while (true) {

float result;

if (!(std::cin >> result))

break;

while (std::cin.peek() != '\n') {

float divider;

std::cin >> divider;

if (divider == 0) {

syncWithMemory(oss, shared1);

return 1;

}

result /= divider;

}

oss << result << "\n";

}

syncWithMemory(oss, shared1);

return 0;

}

**main/common.h**

#pragma once

#include <cstdlib>

#define MAX\_LINE 40

#define SHARED\_STR\_SIZE (MAX\_LINE \* 100) // Размер для mapped memory

#define MAPPED\_FILE\_PATH "/tmp/mai\_os\_mapped\_file"

struct SharedData {

char data[SHARED\_STR\_SIZE];

size\_t size;

bool done;

};

**Протокол работы программы**

**Тест 1:**

$ cat ../input.txt

9 2

8 2 2

6 3

19 3

$ ./mai\_os

../input.txt

4.5

2

2

6.33333

**Тест 2:**

$ cat ../input\_0.txt

9 2

8 2 2

6 3 0

19 3

$ ./mai\_os

../input\_0.txt

4.5

2

ERROR: Division by zero

**Strace (Тест 1):**

$ echo -e "../input.txt\n" | strace -f ./mai\_os

execve("./mai\_os", ["./mai\_os"], 0x7ffe77728a48 /\* 32 vars \*/) = 0

brk(NULL) = 0x55fa4ebfc000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffe1748a4c0) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f72f2863000

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

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=18463, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 18463, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f72f285e000

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

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=2260296, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 2275520, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f72f2632000

mprotect(0x7f72f26cc000, 1576960, PROT\_NONE) = 0

mmap(0x7f72f26cc000, 1118208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x9a000) = 0x7f72f26cc000

mmap(0x7f72f27dd000, 454656, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ab000) = 0x7f72f27dd000

mmap(0x7f72f284d000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x21a000) = 0x7f72f284d000

mmap(0x7f72f285b000, 10432, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f72f285b000

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

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=125488, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 127720, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f72f2612000

mmap(0x7f72f2615000, 94208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7f72f2615000

mmap(0x7f72f262c000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1a000) = 0x7f72f262c000

mmap(0x7f72f2630000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1d000) = 0x7f72f2630000

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\0P\237\2\0\0\0\0\0"..., 832) = 832

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

pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 68, 896) = 68

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2220400, ...}, AT\_EMPTY\_PATH) = 0

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

mmap(NULL, 2264656, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f72f23e9000

mprotect(0x7f72f2411000, 2023424, PROT\_NONE) = 0

mmap(0x7f72f2411000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7f72f2411000

mmap(0x7f72f25a6000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7f72f25a6000

mmap(0x7f72f25ff000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x215000) = 0x7f72f25ff000

mmap(0x7f72f2605000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f72f2605000

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

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=940560, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 942344, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f72f2302000

mmap(0x7f72f2310000, 507904, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe000) = 0x7f72f2310000

mmap(0x7f72f238c000, 372736, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x8a000) = 0x7f72f238c000

mmap(0x7f72f23e7000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe4000) = 0x7f72f23e7000

close(3) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f72f2300000

arch\_prctl(ARCH\_SET\_FS, 0x7f72f23013c0) = 0

set\_tid\_address(0x7f72f2301690) = 268968

set\_robust\_list(0x7f72f23016a0, 24) = 0

rseq(0x7f72f2301d60, 0x20, 0, 0x53053053) = 0

mprotect(0x7f72f25ff000, 16384, PROT\_READ) = 0

mprotect(0x7f72f23e7000, 4096, PROT\_READ) = 0

mprotect(0x7f72f2630000, 4096, PROT\_READ) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f72f22fe000

mprotect(0x7f72f284d000, 45056, PROT\_READ) = 0

mprotect(0x55fa4e3c3000, 4096, PROT\_READ) = 0

mprotect(0x7f72f289d000, 8192, PROT\_READ) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

munmap(0x7f72f285e000, 18463) = 0

getrandom("\xb4\x85\x04\xa3\x6f\x8b\xdd\xd3", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x55fa4ebfc000

brk(0x55fa4ec1d000) = 0x55fa4ec1d000

futex(0x7f72f285b77c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

**openat(AT\_FDCWD, "/tmp/mapped\_file1", O\_RDWR|O\_CREAT|O\_TRUNC, 0666) = 3**

ftruncate(3, 4016) = 0

**mmap(NULL, 4016, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0) = 0x7f72f289c000**

newfstatat(0, "", {st\_mode=S\_IFIFO|0600, st\_size=0, ...}, AT\_EMPTY\_PATH) = 0

read(0, "../input.txt\n\n", 4096) = 14

openat(AT\_FDCWD, "../input.txt", O\_RDONLY) = 4

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 268969 attached

, child\_tidptr=0x7f72f2301690) = 268969

[pid 268969] set\_robust\_list(0x7f72f23016a0, 24 <unfinished ...>

[pid 268968] close(4 <unfinished ...>

[pid 268969] <... set\_robust\_list resumed>) = 0

[pid 268968] <... close resumed>) = 0

[pid 268968] wait4(268969, <unfinished ...>

[pid 268969] dup2(4, 0) = 0

[pid 268969] execve("./child", ["./child"], 0x7ffe1748a698 /\* 32 vars \*/) = 0

[pid 268969] brk(NULL) = 0x561a39d7d000

[pid 268969] arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7fff4e41b920) = -1 EINVAL (Invalid argument)

[pid 268969] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fd5f449d000

[pid 268969] access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

[pid 268969] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 5

[pid 268969] newfstatat(5, "", {st\_mode=S\_IFREG|0644, st\_size=18463, ...}, AT\_EMPTY\_PATH) = 0

[pid 268969] mmap(NULL, 18463, PROT\_READ, MAP\_PRIVATE, 5, 0) = 0x7fd5f4498000

[pid 268969] close(5) = 0

[pid 268969] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libstdc++.so.6", O\_RDONLY|O\_CLOEXEC) = 5

[pid 268969] read(5, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

[pid 268969] newfstatat(5, "", {st\_mode=S\_IFREG|0644, st\_size=2260296, ...}, AT\_EMPTY\_PATH) = 0

[pid 268969] mmap(NULL, 2275520, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 5, 0) = 0x7fd5f426c000

[pid 268969] mprotect(0x7fd5f4306000, 1576960, PROT\_NONE) = 0

[pid 268969] mmap(0x7fd5f4306000, 1118208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 5, 0x9a000) = 0x7fd5f4306000

[pid 268969] mmap(0x7fd5f4417000, 454656, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 5, 0x1ab000) = 0x7fd5f4417000

[pid 268969] mmap(0x7fd5f4487000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 5, 0x21a000) = 0x7fd5f4487000

[pid 268969] mmap(0x7fd5f4495000, 10432, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fd5f4495000

[pid 268969] close(5) = 0

[pid 268969] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = 5

[pid 268969] read(5, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

[pid 268969] newfstatat(5, "", {st\_mode=S\_IFREG|0644, st\_size=125488, ...}, AT\_EMPTY\_PATH) = 0

[pid 268969] mmap(NULL, 127720, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 5, 0) = 0x7fd5f424c000

[pid 268969] mmap(0x7fd5f424f000, 94208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 5, 0x3000) = 0x7fd5f424f000

[pid 268969] mmap(0x7fd5f4266000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 5, 0x1a000) = 0x7fd5f4266000

[pid 268969] mmap(0x7fd5f426a000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 5, 0x1d000) = 0x7fd5f426a000

[pid 268969] close(5) = 0

[pid 268969] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 5

[pid 268969] read(5, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) = 832

[pid 268969] pread64(5, "\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

[pid 268969] pread64(5, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

[pid 268969] pread64(5, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 68, 896) = 68

[pid 268969] newfstatat(5, "", {st\_mode=S\_IFREG|0755, st\_size=2220400, ...}, AT\_EMPTY\_PATH) = 0

[pid 268969] pread64(5, "\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

[pid 268969] mmap(NULL, 2264656, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 5, 0) = 0x7fd5f4023000

[pid 268969] mprotect(0x7fd5f404b000, 2023424, PROT\_NONE) = 0

[pid 268969] mmap(0x7fd5f404b000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 5, 0x28000) = 0x7fd5f404b000

[pid 268969] mmap(0x7fd5f41e0000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 5, 0x1bd000) = 0x7fd5f41e0000

[pid 268969] mmap(0x7fd5f4239000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 5, 0x215000) = 0x7fd5f4239000

[pid 268969] mmap(0x7fd5f423f000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fd5f423f000

[pid 268969] close(5) = 0

[pid 268969] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libm.so.6", O\_RDONLY|O\_CLOEXEC) = 5

[pid 268969] read(5, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

[pid 268969] newfstatat(5, "", {st\_mode=S\_IFREG|0644, st\_size=940560, ...}, AT\_EMPTY\_PATH) = 0

[pid 268969] mmap(NULL, 942344, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 5, 0) = 0x7fd5f3f3c000

[pid 268969] mmap(0x7fd5f3f4a000, 507904, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 5, 0xe000) = 0x7fd5f3f4a000

[pid 268969] mmap(0x7fd5f3fc6000, 372736, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 5, 0x8a000) = 0x7fd5f3fc6000

[pid 268969] mmap(0x7fd5f4021000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 5, 0xe4000) = 0x7fd5f4021000

[pid 268969] close(5) = 0

[pid 268969] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fd5f3f3a000

[pid 268969] arch\_prctl(ARCH\_SET\_FS, 0x7fd5f3f3b3c0) = 0

[pid 268969] set\_tid\_address(0x7fd5f3f3b690) = 268969

[pid 268969] set\_robust\_list(0x7fd5f3f3b6a0, 24) = 0

[pid 268969] rseq(0x7fd5f3f3bd60, 0x20, 0, 0x53053053) = 0

[pid 268969] mprotect(0x7fd5f4239000, 16384, PROT\_READ) = 0

[pid 268969] mprotect(0x7fd5f4021000, 4096, PROT\_READ) = 0

[pid 268969] mprotect(0x7fd5f426a000, 4096, PROT\_READ) = 0

[pid 268969] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fd5f3f38000

[pid 268969] mprotect(0x7fd5f4487000, 45056, PROT\_READ) = 0

[pid 268969] mprotect(0x561a3849e000, 4096, PROT\_READ) = 0

[pid 268969] mprotect(0x7fd5f44d7000, 8192, PROT\_READ) = 0

[pid 268969] prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

[pid 268969] munmap(0x7fd5f4498000, 18463) = 0

[pid 268969] getrandom("\x36\xe5\x1b\xae\xbf\x11\x8a\x23", 8, GRND\_NONBLOCK) = 8

[pid 268969] brk(NULL) = 0x561a39d7d000

[pid 268969] brk(0x561a39d9e000) = 0x561a39d9e000

[pid 268969] futex(0x7fd5f449577c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

**[pid 268969] openat(AT\_FDCWD, "/tmp/mapped\_file1", O\_RDWR) = 5**

**[pid 268969] mmap(NULL, 4016, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 5, 0) = 0x7fd5f44d6000**

[pid 268969] newfstatat(0, "", {st\_mode=S\_IFREG|0644, st\_size=19, ...}, AT\_EMPTY\_PATH) = 0

[pid 268969] read(0, "9 2\n8 2 2\n6 3\n19 3\n", 4096) = 19

[pid 268969] read(0, "", 4096) = 0

**[pid 268969] msync(0x7fd5f44d6000, 4016, MS\_SYNC) = 0**

[pid 268969] exit\_group(0) = ?

[pid 268969] +++ exited with 0 +++

<... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 268969

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=268969, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

newfstatat(1, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x3), ...}, AT\_EMPTY\_PATH) = 0

write(1, "4.5\n", 44.5

) = 4

write(1, "2\n", 22

) = 2

write(1, "2\n", 22

) = 2

write(1, "6.33333\n", 86.33333

) = 8

write(1, "\n", 1

) = 1

**munmap(0x7f72f289c000, 4016) = 0**

close(3) = 0

lseek(0, -1, SEEK\_CUR) = -1 ESPIPE (Illegal seek)

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

При выполнении работы познакомился с memory-mapped файлами в Linux. Возникли проблемы с чтением данных из дочернего процесса. Потребовалась внимательность не запутаться в дескрипторах**.**