Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №1 по курсу

«Операционные системы»

Группа: М8О-214Б-24

Студент: Дылдин С.В.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 09.10.25

Москва, 2025

Постановка задачи

Вариант 9.

9 вариант) В файле записаны команды вида:«число число число<endline>». 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.
- int pipe(int *fd); - создает неименованный канал для обмена данными, simplex for Linux
- int read(int fd, void *buf, size_t count); - читает данные из fd файлового дескриптора
- int write(int fd, const void *buf, size_t count); - пишет данные в fd из буфера
- int close(int fd); - закрывает файловый дескриптор
- void exit(int status); завершает процесс с кодом выхода
- pid_t wait(int *status); - ожидает щавершения дочернего процесса, возвращает его статус код

Описание:

Взаимодействие двух процессов.

- Родитель читает команды из файла и передает ребенку.
- Ребенок обрабатывает команды и выполняет действия
- При ошибке деления на 0 - процессы завершаются

Код программы

parent.c

```c
#include <stddef.h>

#include <stdint.h>

#include <stdbool.h>

#include <unistd.h>

#include <sys/wait.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>
```

```c
#include <fcntl.h>


static char SERVER_PROGRAM_NAME[] = "div-server";


int main( int argc, char **argv){

    if (argc == 1){

        char msg[1024];

        uint32_t len = snprintf(msg, sizeof(msg), "usage: %s filename\n",
argv[0]);

        write(STDERR_FILENO, msg, len);

        exit(EXIT_SUCCESS);

    }


    char progpath[1024];


    {

        ssize_t len = readlink("/proc/self/exe", progpath, sizeof(progpath) - 1);

        if (len == -1){

            const char msg[] = "error: failed to read full path of programm\n";

            write(STDERR_FILENO, msg, sizeof(msg));

            exit(EXIT_FAILURE);

        }

        progpath[len] = '\0';


        ssize_t i = len - 1;

        while (i >= 0 && progpath[i] != '/') --i;

        if (i >= 0) progpath[i] = '\0';
```

```
        else progpath[0] = '\0';
    }


    int parentToChild[2];
    if (pipe(parentToChild) == -1){
        const char msg[] = "error: failed to create pipe parent to child\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        exit(EXIT_FAILURE);
    }


    int childToParent[2];
    if (pipe(childToParent) == -1){
        const char msg[] = "error: failed to create pipe child to parent\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        exit(EXIT_FAILURE);
    }


    const pid_t child = fork();
    switch (child){
        case (-1): {
            const char msg[] = "error failed to create new process\n";
            write(STDERR_FILENO, msg, sizeof(msg));
            exit(EXIT_FAILURE);
        }
        case(0):{
            {
```

```
                pid_t pid = getpid();
                char msg[64];
                const int32_t len = snprintf(msg, sizeof(msg), "%d: child process
created\n", pid);
                write(STDOUT_FILENO, msg, len);
            }
            close(parentToChild[1]);
            close(childToParent[0]);

            dup2(parentToChild[0], STDIN_FILENO);
            close(parentToChild[0]);

            dup2(childToParent[1], STDOUT_FILENO);
            close(childToParent[1]);
            {
                char path[2048];
                snprintf(path, sizeof(path), "%s/%s", progpath,
SERVER_PROGRAM_NAME);
                char *const args[] = {SERVER_PROGRAM_NAME, NULL};
                int32_t status = execv(path, args);
                if (status == -1){
                    const char msg[] = "error: failed to exec into new image \n";
                    write(STDERR_FILENO, msg, sizeof(msg));
                    exit(EXIT_FAILURE);
                }
            }
        } break;
```

```
        default: {
            {
                pid_t pid = getpid();
                char msg[64];
                const int32_t len = snprintf(msg, sizeof(msg), "%d: parent
process, child PID %d\n", pid, child);
                write(STDOUT_FILENO, msg, len);
            }
            close(parentToChild[0]);
            close(childToParent[1]);

            int32_t file = open(argv[1], O_RDONLY);
            if (file == -1){
                const char msg[] = "error: failed to open input file\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                close(parentToChild[1]);
                close(childToParent[0]);
                exit(EXIT_FAILURE);
            }
            char buf[4096];
            ssize_t bytes;
            bool last_was_newline = true;

            while ((bytes = read(file, buf, sizeof(buf))) > 0){
                if (bytes > 0) {
                    ssize_t w = write(parentToChild[1], buf, (size_t)bytes);
                    if (w == -1) {
```

```
                        const char msg[] = "error: failed to write to child
pipe\n";
                        write(STDERR_FILENO, msg, sizeof(msg));
                        break;
                    }
                    last_was_newline = (buf[bytes-1] == '\n');
                }
                ssize_t responseBytes = read(childToParent[0], buf, sizeof(buf));
                if (responseBytes > 0){
                    write(STDOUT_FILENO, buf, responseBytes);
                } else if (responseBytes == 0){
                    const char msg[] = "parent: child process terminated (div by
0)\n";
                    write(STDERR_FILENO, msg, sizeof(msg));
                    break;
                }
            }

            if (bytes == -1) {
                const char msg[] = "error: failed to read input file\n";
                write(STDERR_FILENO, msg, sizeof(msg));
            }

            if (!last_was_newline) {
                const char nl = '\n';
                write(parentToChild[1], &nl, 1);
            }
```

```
            close(file);
            close(parentToChild[1]);
            close(childToParent[0]);

            int status;
            wait(&status);
            if (WIFEXITED(status)){
                int exitCode = WEXITSTATUS(status);
                if (exitCode != 0){
                    const char msg[] = "parent: terminating due to child error\n";
                    write(STDERR_FILENO, msg, sizeof(msg));
                    exit(EXIT_FAILURE);
                }
            }
            const char msg[] = "parent: programm completed successfully\n";
            write(STDOUT_FILENO, msg, sizeof(msg));
        }break;
    }
    return 0;
}
```

child.c

```c
#include <stdint.h>

#include <stdbool.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

#include <stdio.h>

#include <string.h>



#define MAX_NUMS 100







int main(int argc, char **argv){

    char buf[4096];

    ssize_t bytes;

    pid_t pid = getpid();



    char str[4096];

    uint32_t strInd = 0;

    while((bytes = read(STDIN_FILENO, buf, sizeof(buf))) > 0){

        if(bytes < 0){

            const char msg[] = "error occured while reading from stdin\n";

            write(STDERR_FILENO, msg, sizeof(msg));

            exit(EXIT_FAILURE);

        }
```

```
for(ssize_t i = 0; i < bytes; i++){
    if (buf[i] == '\n') {
        if (strInd == 0){
            continue;
        }
        str[strInd] = '\0';

        float nums[MAX_NUMS];
        uint32_t cnt = 0;

        char *ptr = str;
        char *endptr = NULL;

        while(*ptr != '\0' && cnt < MAX_NUMS) {
            while(*ptr == ' ' || *ptr == '\t'){
                ptr++;
            }
            if (*ptr == '\0'){
                break;
            }
            nums[cnt] = strtof(ptr, &endptr);
            if (endptr == ptr){
                break;
            }
            cnt++;
```

```
                    ptr = endptr;
                }
                if (cnt < 2){
                    const char msg[] = "error occured, need at least 2 nums\n";
                    write(STDERR_FILENO, msg, sizeof(msg));
                    strInd = 0;
                    continue;
                }

                float res = nums[0];
                bool divByZero = false;
                char output[4096];
                int outLen = snprintf(output, sizeof(output), "processing %.2f",
nums[0]);
                for (uint32_t j = 1; j < cnt; j++){
                    if(nums[j] == 0.0f){
                        char msg[256];
                        int msgLen = snprintf(msg, sizeof(msg), "div by zero
occeured, child PID: %d is terminatied\n", pid);
                        write(STDERR_FILENO, msg, msgLen);
                        divByZero = true;
                        break;
                    }
                    outLen += snprintf(output + outLen, sizeof(output) - outLen, "
/ %.2f", nums[j]);
                    res /= nums[j];
                }
```

```
                if (divByZero) {
                    close(STDOUT_FILENO);
                    exit(EXIT_FAILURE);
                }
                outLen += snprintf(output + outLen, sizeof(output) - outLen, " =
%.6f\n", res);

                int32_t writing = write(STDOUT_FILENO, output, outLen);
                if(writing != outLen){
                    const char msg[] = "error occured while writing the result
into the file\n";
                    write(STDERR_FILENO, msg, sizeof(msg));
                    exit(EXIT_FAILURE);
                }

                char log[256];
                int logLen = snprintf(log, sizeof(log), "PID: %d with result :
%.6f\n", pid, res);
                write(STDERR_FILENO, log, logLen);
                strInd = 0;

                } else {
                    if(strInd < sizeof(str) - 1){
                        str[strInd++] = buf[i];
                }
            }
```

```
        }
    }
    if (bytes == 0){
        char msg[128];
        int len = snprintf(msg, sizeof(msg), "PID %d successfully terminated\n",
pid);
        write(STDERR_FILENO, msg, len);
    }
    return 0;
}
```

Протокол работы программы

Тестирование:

```
root@9c4acf821f5f:/workspaces/OSI/laba1# ./parent inp.txt
28414: parent process, child PID 28415
28415: child process created
PID: 28415 with result : 1.000000
PID: 28415 with result : 3.000000
PID: 28415 with result : 4.000000
processing 10.00 / 2.00 / 5.00 = 1.000000
processing 9.00 / 3.00 / 1.00 = 3.000000
processing 8.00 / 2.00 = 4.000000
PID 28415 successfully terminated
parent: programm completed successfully


root@9c4acf821f5f:/workspaces/OSI/laba1# ./parent inp2.txt
28426: parent process, child PID 28427
28427: child process created
div by zero occeured, child PID: 28427 is terminatied
parent: child process terminated (div by 0)
parent: terminating due to child error


root@9c4acf821f5f:/workspaces/OSI/laba1# ./parent inp3.txt
28443: parent process, child PID 28444
```

```
28444: child process created
PID: 28444 with result : 10.000000
PID: 28444 with result : 3.000000
processing 10.50 / 2.10 / 0.50 = 10.000000
processing 3.60 / 1.20 = 3.000000
PID 28444 successfully terminated
parent: programm completed successfully


root@9c4acf821f5f:/workspaces/OSI/laba1# ./parent inp4.txt
28455: parent process, child PID 28456
28456: child process created
PID: 28456 with result : 3.125000
processing 100.00 / 2.00 / 2.00 / 2.00 / 2.00 / 2.00 = 3.125000
PID 28456 successfully terminated
parent: programm completed successfully
root@9c4acf821f5f:/workspaces/OSI/laba1#
```

Strace:

root@9c4acf821f5f:/workspaces/OSI/laba1# strace ./parent inp.txt
execve("./parent", ["./parent", "inp.txt"], 0x7ffc99e3adf8 /* 31 vars */) = 0
brk(NULL) = 0x3853c000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7370c00db000
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=33723, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 33723, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7370c00d2000
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\20t\2\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
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=1926232, ...}, 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, 1974096, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7370bfef0000
mmap(0x7370bff16000, 1400832, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x26000) = 0x7370bff16000
mmap(0x7370c006c000, 339968, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17c000) = 0x7370c006c000
mmap(0x7370c00bf000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1cf000) = 0x7370c00bf000
mmap(0x7370c00c5000, 53072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7370c00c5000
close(3) = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7370bfeed000

```
arch_prctl(ARCH_SET_FS, 0x7370bfeed740) = 0
set_tid_address(0x7370bfeeda10)         = 28697
set_robust_list(0x7370bfeeda20, 24)     = 0
rseq(0x7370bfeee060, 0x20, 0, 0x53053053) = 0
mprotect(0x7370c00bf000, 16384, PROT_READ) = 0
mprotect(0x403000, 4096, PROT_READ)     = 0
mprotect(0x7370c010e000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7370c00d2000, 33723)           = 0
readlink("/proc/self/exe", "/workspaces/OSI/laba1/parent", 1023) = 28
pipe2([3, 4], 0)                        = 0
pipe2([5, 6], 0)                        = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
 child_tidptr=0x7370bfeeda10) = 28698
28698: child process created
getpid()                                = 28697
write(1, "28697: parent process, child PID"..., 3928697: parent process, child PID 28698
) = 39
close(3)                                = 0
close(6)                                = 0
openat(AT_FDCWD, "inp.txt", O_RDONLY)   = 3
read(3, "10 2 5\n9 3 1\n8 2\n", 4096)   = 17
write(4, "10 2 5\n9 3 1\n8 2\n", 17)    = 17
read(5, PID: 28698 with result : 1.000000
PID: 28698 with result : 3.000000
PID: 28698 with result : 4.000000
"processing 10.00 / 2.00 / 5.00 ="..., 4096) = 42
write(1, "processing 10.00 / 2.00 / 5.00 ="..., 42processing 10.00 / 2.00 / 5.00 = 1.000000
) = 42
read(3, "", 4096)                       = 0
close(3)                                = 0
close(4PID 28698 successfully terminated
)                                       = 0
close(5)                                = 0
wait4(-1, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 28698
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=28698, si_uid=0, si_status=0,
si_utime=0, si_stime=0} ---
write(1, "parent: programm completed succe"..., 41parent: programm completed successfully
) = 41
exit_group(0)                           = ?
+++ exited with 0 +++


root@9c4acf821f5f:/workspaces/OSI/laba1# strace ./parent inp2.txt
execve("./parent", ["./parent", "inp2.txt"], 0x7ffd74c26c98 /* 31 vars */) = 0
brk(NULL)                               = 0x2d2f5000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
```

0x70a66cde2000
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=33723, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 33723, PROT_READ, MAP_PRIVATE, 3, 0) = 0x70a66cdd9000
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\20t\2\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
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=1926232, ...}, 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, 1974096, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x70a66cbf7000
mmap(0x70a66cc1d000, 1400832, PROT_READ|PROT_EXEC,
 MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x26000) = 0x70a66cc1d000
mmap(0x70a66cd73000, 339968, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
 0x17c000) = 0x70a66cd73000
mmap(0x70a66cdc6000, 24576, PROT_READ|PROT_WRITE,
 MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1cf000) = 0x70a66cdc6000
mmap(0x70a66cdcc000, 53072, PROT_READ|PROT_WRITE,
 MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x70a66cdcc000
close(3)   = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
 0x70a66cbf4000
arch_prctl(ARCH_SET_FS, 0x70a66cbf4740) = 0
set_tid_address(0x70a66cbf4a10)   = 28728
set_robust_list(0x70a66cbf4a20, 24)   = 0
rseq(0x70a66cbf5060, 0x20, 0, 0x53053053) = 0
mprotect(0x70a66cdc6000, 16384, PROT_READ) = 0
mprotect(0x403000, 4096, PROT_READ)   = 0
mprotect(0x70a66ce15000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x70a66cdd9000, 33723)   = 0
readlink("/proc/self/exe", "/workspaces/OSI/laba1/parent", 1023) = 28
pipe2([3, 4], 0)   = 0
pipe2([5, 6], 0)   = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
 child_tidptr=0x70a66cbf4a10) = 28729
28729: child process created
getpid()   = 28728
write(1, "28728: parent process, child PID"..., 3928728: parent process, child PID 28729
) = 39
close(3)   = 0
close(6)   = 0
openat(AT_FDCWD, "inp2.txt", O_RDONLY) = 3
read(3, "12 3 0 2\n5 1\n", 4096)   = 13
write(4, "12 3 0 2\n5 1\n", 13)   = 13
read(5, div by zero occeured, child PID: 28729 is terminatied

"", 4096)   = 0
write(2, "parent: child process terminated"..., 45parent: child process terminated (div by 0)
) = 45
close(3)   = 0
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=28729, si_uid=0, si_status=1,
 si_utime=0, si_stime=0} ---
close(4)   = 0
close(5)   = 0
wait4(-1, [{WIFEXITED(s) && WEXITSTATUS(s) == 1}], 0, NULL) = 28729
write(2, "parent: terminating due to child"..., 40parent: terminating due to child error
) = 40
exit_group(1)   = ?
+++ exited with 1 +++

root@9c4acf821f5f:/workspaces/OSI/laba1# strace ./parent inp3.txt
execve("./parent", ["./parent", "inp3.txt"], 0x7ffc93610858 /* 31 vars */) = 0
brk(NULL)   = 0x12a84000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
 0x7a2a29290000
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=33723, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 33723, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7a2a29287000
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\20t\2\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
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=1926232, ...}, 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, 1974096, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7a2a290a5000
mmap(0x7a2a290cb000, 1400832, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x26000) = 0x7a2a290cb000
mmap(0x7a2a29221000, 339968, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
 0x17c000) = 0x7a2a29221000
mmap(0x7a2a29274000, 24576, PROT_READ|PROT_WRITE,
 MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1cf000) = 0x7a2a29274000
mmap(0x7a2a2927a000, 53072, PROT_READ|PROT_WRITE,
 MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7a2a2927a000
close(3)   = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
 0x7a2a290a2000
arch_prctl(ARCH_SET_FS, 0x7a2a290a2740) = 0
set_tid_address(0x7a2a290a2a10)   = 28744
set_robust_list(0x7a2a290a2a20, 24)   = 0
rseq(0x7a2a290a3060, 0x20, 0, 0x53053053) = 0
mprotect(0x7a2a29274000, 16384, PROT_READ) = 0

```
mprotect(0x403000, 4096, PROT_READ)     = 0
mprotect(0x7a2a292c3000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7a2a29287000, 33723)           = 0
readlink("/proc/self/exe", "/workspaces/OSI/laba1/parent", 1023) = 28
pipe2([3, 4], 0)                        = 0
pipe2([5, 6], 0)                        = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
 child_tidptr=0x7a2a290a2a10) = 28745
getpid(28745: child process created
)                                       = 28744
write(1, "28744: parent process, child PID"..., 3928744: parent process, child PID 28745
) = 39
close(3)                                = 0
close(6)                                = 0
openat(AT_FDCWD, "inp3.txt", O_RDONLY)  = 3
read(3, "10.5 2.1 0.5\n3.6 1.2\n", 4096) = 21
write(4, "10.5 2.1 0.5\n3.6 1.2\n", 21) = 21
read(5, PID: 28745 with result : 10.000000
PID: 28745 with result : 3.000000
"processing 10.50 / 2.10 / 0.50 ="..., 4096) = 43
write(1, "processing 10.50 / 2.10 / 0.50 ="..., 43processing 10.50 / 2.10 / 0.50 = 10.000000
) = 43
read(3, "", 4096)                       = 0
close(3)                                = 0
close(4)                                = 0
PID 28745 successfully terminated
close(5)                                = 0
wait4(-1, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 28745
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=28745, si_uid=0, si_status=0,
 si_utime=0, si_stime=0} ---
write(1, "parent: programm completed succe"..., 41parent: programm completed successfully
) = 41
exit_group(0)                           = ?
+++ exited with 0 +++


root@9c4acf821f5f:/workspaces/OSI/laba1# strace ./parent inp4.txt
execve("./parent", ["./parent", "inp4.txt"], 0x7fff123f27e8 /* 31 vars */) = 0
brk(NULL)                               = 0x2b16000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
 0x730c22c70000
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=33723, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 33723, PROT_READ, MAP_PRIVATE, 3, 0) = 0x730c22c67000
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\20t\2\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
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=1926232, ...}, 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, 1974096, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x730c22a85000

mmap(0x730c22aab000, 1400832, PROT_READ|PROT_EXEC,
 MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x26000) = 0x730c22aab000
mmap(0x730c22c01000, 339968, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
 0x17c000) = 0x730c22c01000
mmap(0x730c22c54000, 24576, PROT_READ|PROT_WRITE,
 MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1cf000) = 0x730c22c54000
mmap(0x730c22c5a000, 53072, PROT_READ|PROT_WRITE,
 MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x730c22c5a000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
 0x730c22a82000
arch_prctl(ARCH_SET_FS, 0x730c22a82740) = 0
set_tid_address(0x730c22a82a10)         = 28759
set_robust_list(0x730c22a82a20, 24)     = 0
rseq(0x730c22a83060, 0x20, 0, 0x53053053) = 0
mprotect(0x730c22c54000, 16384, PROT_READ) = 0
mprotect(0x403000, 4096, PROT_READ)     = 0
mprotect(0x730c22ca3000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x730c22c67000, 33723)           = 0
readlink("/proc/self/exe", "/workspaces/OSI/laba1/parent", 1023) = 28
pipe2([3, 4], 0)                        = 0
pipe2([5, 6], 0)                        = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
 child_tidptr=0x730c22a82a10) = 28760
28760: child process created
getpid()                                = 28759
write(1, "28759: parent process, child PID"..., 3928759: parent process, child PID 28760
) = 39
close(3)                                = 0
close(6)                                = 0
openat(AT_FDCWD, "inp4.txt", O_RDONLY)  = 3
read(3, "100 2 2 2 2 2\n", 4096)        = 14
write(4, "100 2 2 2 2 2\n", 14)         = 14
read(5, PID: 28760 with result : 3.125000
"processing 100.00 / 2.00 / 2.00 "..., 4096) = 64
write(1, "processing 100.00 / 2.00 / 2.00 "..., 64processing 100.00 / 2.00 / 2.00 / 2.00 / 2.00 / 2.00 =
 3.125000
) = 64
read(3, "", 4096)                       = 0
```

```
close(3)                                = 0
close(4PID 28760 successfully terminated
)                                       = 0
close(5)                                = 0
wait4(-1, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 28760
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=28760, si_uid=0, si_status=0,
 si_utime=0, si_stime=0} ---
write(1, "parent: programm completed succe"..., 41parent: programm completed successfully
) = 41
exit_group(0)                           = ?
+++ exited with 0 +++
root@9c4acf821f5f:/workspaces/OSI/laba1#
```

Вывод

Научился создавать дочерние процессы, научился организовавывать взамиодействие между процессами через pipe. Разобрался в работе системных вызовов для Linux. Во время выполнения работы были проблемы с взаимодействием процессов при получении ошибки в дочернем при делении на 0.