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»**

Студент: Чапалда Мария Олеговна

Группа: М8О-201Б-22

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2023

**Постановка задачи**

**Группа вариантов 2.**

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. Стандартный поток вывода дочернего процесса перенаправляется в pipe1. Родительский процесс читает из pipe1 и прочитанное выводит в свой стандартный поток вывода. Родительский и дочерний процесс должны быть представлены разными программами.

**Вариант 9.**

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, у которого первое поле отвечает за чтение, а второе - за запись.
* int execl(const char \*\_\_path, char \*const \*\_\_argv, ...); - предоставляет новой программе список аргументов в виде массива указателей на строки, заканчивающиеся (char \*)0.
* int dup2(int, int); - создает копию файлового дескриптора oldfd *(1 поле)*, используя для нового дескриптора newfd *(2 поле)* файловый дескриптор (они становятся взаимозаменяемыми).
* \_exit(int status); – выходит из процесса с заданным статусом.
* pid\_t wait(int \*status); – приостаналивает выполнение текущего процесса до тех пор, пока дочерний процесс не завершится.
* int read(int fd, void \*buffer, int nbyte); – читает nbyte байтов из файлового дескриптора fd в буффер buffer.

Пользователь вводит имя файла в командной строке, которое будет использоваться для чтения. Если введенная строка корректна и файл с таким именем существует, то создается дочерний процесс. В дочернем процессе стандартный ввод переопределяется для чтения из указанного файла, а стандартный вывод переопределяется в канал pipe. Родительский процесс читает результат работы дочернего процесса из канала и выводит его на экран, если дочерний процесс успешно выполнил проверку деления на ноль. В случае неудачи, дочерний процесс вернет значение -1, на экран будет выведено сообщение «Attempt to divide by zero», и работа завершится.

**Код программы**

**parent.c**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <fcntl.h>

#include <string.h>

#include <stdbool.h>

void check\_error(bool expression, char\* message) { *//функция для проверки ошибок*

    if (expression) { *//если выражение истинно*

        write(STDOUT\_FILENO, message, strlen(message) \* sizeof(char)); *//выод об ошибке*

        write(STDOUT\_FILENO, "\n", 1);

        exit(-1); *//завершить программу*

    }

}

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

    pid\_t pid; *//объявление переменной для хранения идентификатора процесса*

    int pipe\_1[2]; *//объявление массива для хранения файловых дескрипторов для канала*

    if (pipe(pipe\_1) == -1) { *//создание канала и проверка на ошибки*

        perror("pipe"); *//вывод сообщения об ошибке*

        \_exit(EXIT\_FAILURE); *//завершение программы*

    }

    if (argc != 2) { *//проверка наличия имени файла в аргументах командной строки*

        write(1, "Error: no filename\n", 20); *//вывод сообщения об ошибке*

        exit(EXIT\_FAILURE); *//завершение программы*

    }

    int fd = open(argv[1], O\_RDONLY); *//открытие файла для чтения*

    check\_error(fd == -1, "Can't open file"); *//проверка на ошибки при открытии файла*

    pid = fork(); *//создание дочернего процесса*

    if (pid == -1) { *//проверка на ошибку при создании дочернего процесса*

        perror("fork"); *//вывод сообщения об ошибке*

        return -1; *//возврат отрицательного статуса*

    }

    else if (pid == 0) { *//дочерний процесс*

        close (pipe\_1[0]); *//закрыть конец канала для чтения*

        check\_error(dup2(fd, STDIN\_FILENO) < 0, "Error dub"); *//перенаправить ввод на чтение из файла*

        dup2(pipe\_1[1], STDOUT\_FILENO); *//перенаправить вывод на запись в канал*

        execl("./child", "/.child", NULL); *//выполнить программу-ребенка*

        perror("execl"); *//вывод сообщения об ошибке, если execl завершился неудачно*

        return 1;

    }

    else { *//родительский процесс*

        check\_error((pid == -1), "Process error"); *//проверка на ошибки при создании дочернего процесса*

        close(pipe\_1[1]); *//закрыть конец канала для записи*

        wait(0); *//ожидание завершения дочернего процесса*

        float result; *//переменная для хранения результата деления*

        char answer[50]; *//массив для хранения результата в виде строки*

        while ((read(pipe\_1[0], &result, sizeof(float))) > 0) { *//чтение из канала в переменную result*

            if (result == -1) { *//если обнаружена попытка деления на ноль*

                write(STDOUT\_FILENO, "Attempt to divide by zero\n", 27); *//вывод сообщения об ошибке*

                exit(EXIT\_FAILURE); *//завершение программы с отрицательным статусом*

            }

            sprintf(answer, "%f\n", result); *//форматирование результата как строки*

            check\_error(write(STDOUT\_FILENO, answer, strlen(answer)) == -1, "Write error\n"); *//вывод результата*

            check\_error(write(STDOUT\_FILENO, "\n", 1) == -1, "Write error\n"); *//вывод новой строки*

        }

    }

    return 0;

}

**child.c**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/types.h>

#include <fcntl.h>

#include <string.h>

#include <stdbool.h>

#include <math.h>

библиотеку

#define buf\_size 100 *//определяем размер буфера*

int main() {

    int c; *//объявляем переменную для символа*

    bool not\_end = true; *//объявляем логическую переменную*

    float nmbr = 0; *//объявляем переменную для числа*

    float result = 0; *//объявляем переменную для результата*

    float first = 0; *//объявляем переменную для первого числа*

    int k = 0; *//объявляем переменную для степени десятки*

    float dot = 0; *//объявляем переменную для дробной части числа*

    int count = 0; *//объявляем переменную для подсчета чисел*

    float numbers[100]; *//объявляем массив для хранения чисел*

    do {

        if (not\_end) { *//если не конец ввода*

            if (c <= '9' && c >= '0') { *//если символ - цифра*

                if (nmbr != 0 && (floor(nmbr) != nmbr)) { *//если число не равно 0 и имеет дробную часть*

                    if (c == '0') { *//если цифра равна 0*

                        k += 1; *//увеличиваем степень десятки*

                        nmbr += 0; *//не изменяем число*

                    }

                    else { *//если цифра не равна 0*

                        dot = c - '0'; *//вычисляем дробную часть числа*

                        nmbr = (nmbr + 0.1) + (dot / pow(10, k + 1)); *//вычисляем новое число с учетом дробной части*

                    }

                }

                else { *//если число не имеет дробной части*

                    nmbr = nmbr \* 10 + c - '0'; *//увеличиваем число на порядок и добавляем новую цифру*

                }

            }

            if (c == '.') { *//если символ - точка*

                nmbr = nmbr - 0.1; *//уменьшаем число на порядок*

            }

            if (c == ' ' || c == '\\n' || c == EOF) { *//если символ - пробел, новая строка или конец файла*

                numbers[count] = nmbr; *//сохраняем число в массиве*

                nmbr = 0; *//обнуляем переменную для числа*

                count++; *//увеличиваем счетчик чисел*

                if (c == '\\n' || c == EOF) { *//если символ - новая строка или конец файла*

                    first = numbers[0]; *//сохраняем первое число*

                    for (int i = 1; i < count; i++) {

                        if (numbers[i] == 0) { *//если число равно 0*

                            result = -1; *//устанавливаем результат как -1*

                            write(STDOUT\_FILENO, &result, sizeof(result)); *//выводим результат*

                        }

                        result = first / numbers[i]; *//вычисляем результат деления первого числа на текущее число*

                        first = result; *//сохраняем результат как первое число для следующей итерации*

                    }

                    not\_end = false; *//устанавливаем флаг окончания ввода как ложь*

                    first = 0; *//обнуляем переменную для первого числа*

                    count = 0; *//обнуляем счетчик чисел*

                }

            }

        }

        if (c == '\\n' || c == EOF) { *//если символ новая строка или конец файла*

            write(STDOUT\_FILENO, &result, sizeof(result)); *// выводим результат*

            result = 0; *//обнуляем результат*

            k = 0; *//обнуляем степень десятки*

            dot = 0; *//обнуляем дробную часть числа*

            not\_end = true; *//устанавливаем флаг окончания ввода как истину для следующей итерации*

        }

    } while((read(STDIN\_FILENO, &c, sizeof(char))) > 0); *//считываем символ*

    return 0;

}

**Протокол работы программы**

**Тестирование:**

mariyaach@LAPTOP-HSJSPL2E:~/оси/lab1$ ./parent file.txt

900.000122

1.000000

14.273480

0.666667

90.000000

38.136364

4329.239258

2880.000000

**Strace**

mariyaach@LAPTOP-HSJSPL2E:~/оси/lab1$ strace -f ./parent file.txt

execve("./parent", ["./parent", "file.txt"], 0x7ffc1e4481e0 /\* 30 vars \*/) = 0

brk(NULL) = 0x55e76f1e8000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffff8002340) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f5e90af2000

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=16299, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 16299, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f5e90aee000

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\0 =\340\2563\265?\356\25x\261\27\313A#\350"..., 68, 896) = 68

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2216304, ...}, 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, 2260560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f5e908c6000

mmap(0x7f5e908ee000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7f5e908ee000

mmap(0x7f5e90a83000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7f5e90a83000

mmap(0x7f5e90adb000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x214000) = 0x7f5e90adb000

mmap(0x7f5e90ae1000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f5e90ae1000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f5e908c3000

arch\_prctl(ARCH\_SET\_FS, 0x7f5e908c3740) = 0

set\_tid\_address(0x7f5e908c3a10) = 3986

set\_robust\_list(0x7f5e908c3a20, 24) = 0

rseq(0x7f5e908c40e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7f5e90adb000, 16384, PROT\_READ) = 0

mprotect(0x55e76e0be000, 4096, PROT\_READ) = 0

mprotect(0x7f5e90b2c000, 8192, PROT\_READ) = 0

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

munmap(0x7f5e90aee000, 16299) = 0

pipe2([3, 4], 0) = 0

openat(AT\_FDCWD, "file.txt", O\_RDONLY) = 5

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 3987 attached

, child\_tidptr=0x7f5e908c3a10) = 3987

[pid 3987] set\_robust\_list(0x7f5e908c3a20, 24 <unfinished ...>

[pid 3986] close(4 <unfinished ...>

[pid 3987] <... set\_robust\_list resumed>) = 0

[pid 3986] <... close resumed>) = 0

[pid 3987] close(3 <unfinished ...>

[pid 3986] wait4(-1, <unfinished ...>

[pid 3987] <... close resumed>) = 0

[pid 3987] dup2(5, 0) = 0

[pid 3987] dup2(4, 1) = 1

[pid 3987] execve("./child", ["/.child"], 0x7ffff8002520 /\* 30 vars \*/) = 0

[pid 3987] brk(NULL) = 0x5650cba18000

[pid 3987] arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffddf031f80) = -1 EINVAL (Invalid argument)

[pid 3987] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ffb51137000

[pid 3987] access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

[pid 3987] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

[pid 3987] newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=16299, ...}, AT\_EMPTY\_PATH) = 0

[pid 3987] mmap(NULL, 16299, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7ffb51133000

[pid 3987] close(3) = 0

[pid 3987] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libm.so.6", O\_RDONLY|O\_CLOEXEC) = 3

[pid 3987] 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

[pid 3987] newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=940560, ...}, AT\_EMPTY\_PATH) = 0

[pid 3987] mmap(NULL, 942344, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7ffb5104c000

[pid 3987] mmap(0x7ffb5105a000, 507904, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe000) = 0x7ffb5105a000

[pid 3987] mmap(0x7ffb510d6000, 372736, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x8a000) = 0x7ffb510d6000

[pid 3987] mmap(0x7ffb51131000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe4000) = 0x7ffb51131000

[pid 3987] close(3) = 0

[pid 3987] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

[pid 3987] 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

[pid 3987] 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

[pid 3987] 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

[pid 3987] pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0 =\340\2563\265?\356\25x\261\27\313A#\350"..., 68, 896) = 68

[pid 3987] newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2216304, ...}, AT\_EMPTY\_PATH) = 0

[pid 3987] 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

[pid 3987] mmap(NULL, 2260560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7ffb50e24000

[pid 3987] mmap(0x7ffb50e4c000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7ffb50e4c000

[pid 3987] mmap(0x7ffb50fe1000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7ffb50fe1000

[pid 3987] mmap(0x7ffb51039000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x214000) = 0x7ffb51039000

[pid 3987] mmap(0x7ffb5103f000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7ffb5103f000

[pid 3987] close(3) = 0

[pid 3987] mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ffb50e21000

[pid 3987] arch\_prctl(ARCH\_SET\_FS, 0x7ffb50e21740) = 0

[pid 3987] set\_tid\_address(0x7ffb50e21a10) = 3987

[pid 3987] set\_robust\_list(0x7ffb50e21a20, 24) = 0

[pid 3987] rseq(0x7ffb50e220e0, 0x20, 0, 0x53053053) = 0

[pid 3987] mprotect(0x7ffb51039000, 16384, PROT\_READ) = 0

[pid 3987] mprotect(0x7ffb51131000, 4096, PROT\_READ) = 0

[pid 3987] mprotect(0x5650cb733000, 4096, PROT\_READ) = 0

[pid 3987] mprotect(0x7ffb51171000, 8192, PROT\_READ) = 0

[pid 3987] prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

[pid 3987] munmap(0x7ffb51133000, 16299) = 0

[pid 3987] read(0, "9", 1) = 1

[pid 3987] read(0, " ", 1) = 1

[pid 3987] read(0, "0", 1) = 1

[pid 3987] read(0, "0", 1) = 1

[pid 3987] read(0, ".", 1) = 1

[pid 3987] read(0, "0", 1) = 1

[pid 3987] read(0, "1", 1) = 1

[pid 3987] read(0, " ", 1) = 1

[pid 3987] read(0, "1", 1) = 1

[pid 3987] read(0, "\n", 1) = 1

[pid 3987] read(0, "1", 1) = 1

[pid 3987] read(0, "0", 1) = 1

[pid 3987] read(0, " ", 1) = 1

[pid 3987] read(0, "2", 1) = 1

[pid 3987] read(0, ".", 1) = 1

[pid 3987] read(0, "5", 1) = 1

[pid 3987] read(0, " ", 1) = 1

[pid 3987] read(0, "4", 1) = 1

[pid 3987] read(0, "\n", 1) = 1

[pid 3987] read(0, "1", 1) = 1

[pid 3987] read(0, "0", 1) = 1

[pid 3987] read(0, "0", 1) = 1

[pid 3987] read(0, "0", 1) = 1

[pid 3987] read(0, " ", 1) = 1

[pid 3987] read(0, "7", 1) = 1

[pid 3987] read(0, ".", 1) = 1

[pid 3987] read(0, "0", 1) = 1

[pid 3987] read(0, "0", 1) = 1

[pid 3987] read(0, "6", 1) = 1

[pid 3987] read(0, " ", 1) = 1

[pid 3987] read(0, "2", 1) = 1

[pid 3987] read(0, " ", 1) = 1

[pid 3987] read(0, "5", 1) = 1

[pid 3987] read(0, "\n", 1) = 1

[pid 3987] read(0, "2", 1) = 1

[pid 3987] read(0, " ", 1) = 1

[pid 3987] read(0, "1", 1) = 1

[pid 3987] read(0, " ", 1) = 1

[pid 3987] read(0, "3", 1) = 1

[pid 3987] read(0, "\n", 1) = 1

[pid 3987] read(0, "9", 1) = 1

[pid 3987] read(0, " ", 1) = 1

[pid 3987] read(0, "0", 1) = 1

[pid 3987] read(0, ".", 1) = 1

[pid 3987] read(0, "1", 1) = 1

[pid 3987] read(0, "\n", 1) = 1

[pid 3987] read(0, "8", 1) = 1

[pid 3987] read(0, "3", 1) = 1

[pid 3987] read(0, "9", 1) = 1

[pid 3987] read(0, " ", 1) = 1

[pid 3987] read(0, "2", 1) = 1

[pid 3987] read(0, "2", 1) = 1

[pid 3987] read(0, "\n", 1) = 1

[pid 3987] read(0, "3", 1) = 1

[pid 3987] read(0, "9", 1) = 1

[pid 3987] read(0, "8", 1) = 1

[pid 3987] read(0, "2", 1) = 1

[pid 3987] read(0, "9", 1) = 1

[pid 3987] read(0, " ", 1) = 1

[pid 3987] read(0, "9", 1) = 1

[pid 3987] read(0, ".", 1) = 1

[pid 3987] read(0, "2", 1) = 1

[pid 3987] read(0, "\n", 1) = 1

[pid 3987] read(0, "2", 1) = 1

[pid 3987] read(0, "8", 1) = 1

[pid 3987] read(0, "8", 1) = 1

[pid 3987] read(0, " ", 1) = 1

[pid 3987] read(0, "0", 1) = 1

[pid 3987] read(0, "0", 1) = 1

[pid 3987] read(0, "0", 1) = 1

[pid 3987] read(0, "0", 1) = 1

[pid 3987] read(0, "0", 1) = 1

[pid 3987] read(0, "0", 1) = 1

[pid 3987] read(0, "0", 1) = 1

[pid 3987] read(0, ".", 1) = 1

[pid 3987] read(0, "1", 1) = 1

[pid 3987] read(0, "\n", 1) = 1

[pid 3987] read(0, "2", 1) = 1

[pid 3987] read(0, "9", 1) = 1

[pid 3987] read(0, "4", 1) = 1

[pid 3987] read(0, "8", 1) = 1

[pid 3987] read(0, "2", 1) = 1

[pid 3987] read(0, " ", 1) = 1

[pid 3987] read(0, "0", 1) = 1

[pid 3987] read(0, "", 1) = 0

[pid 3987] exit\_group(0) = ?

[pid 3987] +++ exited with 0 +++

<... wait4 resumed>NULL, 0, NULL) = 3987

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=3987, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=3} ---

read(3, "", 4) = 0

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

Выполняя эту работу, я освоила принципы работы с каналами для взаимодействия между процессами. Я познакомилась с процессом перенаправления ввода-вывода процесса, изучила понятие файлового дескриптора и осознала важность их своевременного закрытия. В процессе выполнения лабораторной работы возникли две основные сложности: сначала мне было трудно написать рабочий парсер, но по совету одногруппника мне удалось справиться с этой задачей. Позже у меня возникли вопросы относительно функции write, и я столкнулась с трудностями в понимании того, как она обрабатывает тип float. В целом, я усвоила много новой информации, которая будет полезна мне при написании более сложных программ в будущем.