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-209Б-23

Студент: Осипов М.Н.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 12.10.25

Москва, 2025

Постановка задачи

Вариант 9.

В файле записаны строки с числами в формате: «число число число<конец строки>». Дочерний процесс выполняет деление первого числа в строке на все последующие числа и выводит результат в стандартный поток вывода. Если происходит деление на ноль, то дочерний и родительский процессы завершают свою работу. Проверка деления на ноль должна осуществляться в дочернем процессе. Все числа имеют тип float. Количество чисел в строке может быть произвольным.

Общий метод и алгоритм решения

Использованные системные вызовы:

В parent.c:

- fork(); – создает дочерний процесс;
- pipe() - создание канала для межпроцессорного подключения;
- open() - открытие файла;
- close() - закрытие файловых дескрипторов;
- read() - чтение данных из pipe;
- waitpid() - ожидание завершения дочернего процесса;
- execl() - запуск исполняемого файла (child);
- access() - проверка существования файла.

В child.c:

- fgets() - чтение строк из стандартного ввода;
- malloc()/realloc()/free() - управление динамической памятью;
- exit() - завершение процесса с кодом возврата;
- fflush() - сброс буферов вывода.

Далее родитель - parent.c, дочерний процесс - child.c:

1. Родитель создает pipe и дочерний процесс
2. Дочерний процесс перенаправляет stdin на файл, stdout на pipe
3. Дочерний процесс читает строки, вычисляет результат и пишет в pipe
4. Родитель читает результаты из pipe и выводит на экран
5. При делении на ноль дочерний процесс завершается с кодом 2, родитель обрабатывает это.

Код программы

parent.c

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>
```

```c
#include <sys/wait.h>
#include <string.h>
#include <fcntl.h>
#include <ctype.h>


int main() {
    char filename[256];

    printf("Введите имя файла: ");
    if (fgets(filename, sizeof(filename), stdin) == NULL) {
        perror("Ошибка чтения имени файла");
        exit(1);
    }

    filename[strcspn(filename, "\n")] = 0;

    if (access(filename, F_OK) != 0) {
        printf("Ошибка: файл '%s' не существует\n", filename);
        exit(1);
    }

    FILE *test_file = fopen(filename, "r");
    if (test_file == NULL) {
        perror("Ошибка открытия файла");
        exit(1);
    }

    int is_empty = 1;
    int ch;
    while ((ch = fgetc(test_file)) != EOF) {
        if (!isspace(ch)) {
```

```
            is_empty = 0;
            break;
        }
    }
    fclose(test_file);

    if (is_empty) {
        printf("Ошибка: файл '%s' пустой\n", filename);
        exit(1);
    }

    int pipefd[2];
    if (pipe(pipefd) == -1) {
        perror("Ошибка создания pipe");
        exit(1);
    }

    pid_t pid = fork();

    if (pid == -1) {
        perror("Ошибка fork");
        exit(1);
    }

    if (pid == 0) {
        close(pipefd[0]);

        if (dup2(pipefd[1], STDOUT_FILENO) == -1) {
            perror("Ошибка dup2 в дочернем процессе");
            exit(1);
        }
        close(pipefd[1]);
```

```
    int fd = open(filename, O_RDONLY);
    if (fd == -1) {
        perror("Ошибка открытия файла");
        exit(1);
    }

    if (dup2(fd, STDIN_FILENO) == -1) {
        perror("Ошибка dup2 для файла");
        close(fd);
        exit(1);
    }
    close(fd);

    execl("./child", "child", NULL);
    perror("Ошибка execl");
    exit(1);
} else {
    close(pipefd[1]);

    char buffer[1024];
    ssize_t bytes_read;

    while ((bytes_read = read(pipefd[0], buffer, sizeof(buffer) - 1)) > 0) {
        buffer[bytes_read] = '\0';
        printf("%s", buffer);
    }

    close(pipefd[0]);

    int status;
    waitpid(pid, &status, 0);
```

```
            if (WIFEXITED(status)) {
                int exit_status = WEXITSTATUS(status);
                if (exit_status == 2) {
                    printf("Программа завершена из-за ошибки деления на ноль или в строке присутствуют другие символы, не являющиеся числами.\n");
                    exit(1);
                } else if (exit_status != 0) {
                    exit(1);
                }
            }

            printf("Все вычисления завершены успешно.\n");
        }

        return 0;
    }
```

child.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>


int main() {
    char line[1024];
    int empty_file = 1;

    while (fgets(line, sizeof(line), stdin)) {
        line[strcspn(line, "\n")] = 0;
```

```
        if (strlen(line) == 0) continue;

        empty_file = 0;

        float *numbers = NULL;
        int capacity = 10;
        int count = 0;

        numbers = (float*)malloc(capacity * sizeof(float));
        if (numbers == NULL) {
            fprintf(stdout, "Ошибка выделения памяти\n");
            exit(1);
        }

        char *token = strtok(line, " ");
        while (token != NULL) {
            if (count >= capacity) {
                capacity *= 2;
                float *temp = (float*)realloc(numbers, capacity * sizeof(float));
                if (temp == NULL) {
                    fprintf(stdout, "Ошибка перевыделения памяти\n");
                    free(numbers);
                    exit(1);
                }
                numbers = temp;
            }

            numbers[count++] = atof(token);
            token = strtok(NULL, " ");
        }

        if (count < 2) {
```

```
            fprintf(stdout, "Ошибка: в строке должно быть хотя бы 2 числа\n");
            fflush(stdout);
            free(numbers);
            continue;
        }

        for (int i = 1; i < count; i++) {
            if (numbers[i] == 0.0f) {
                free(numbers);
                exit(2);
            }
        }

        float result = numbers[0];
        for (int i = 1; i < count; i++) {
            result /= numbers[i];
        }

        printf("%f\n", result);
        fflush(stdout);

        free(numbers);
    }

    if (empty_file) {
        fprintf(stdout, "Ошибка: файл пустой\n");
        exit(1);
    }

    return 0;
}
```

Протокол работы программы

$ ./parent

Введите имя файла: input.txt

2.000000

Ошибка: в строке должно быть хотя бы 2 числа

0.500000

Программа завершена из-за ошибки деления на ноль или в строке присутствуют другие символы, не являющиеся числами.

$ strace -f ./parent

execve("./parent", ["./parent"], 0x7ffe6bf59de8 /* 29 vars */) = 0

brk(NULL) = 0x62a0d8d28000

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

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

fstat(3, {st_mode=S_IFREG|0644, st_size=21784, ...}) = 0

mmap(NULL, 21784, PROT_READ, MAP_PRIVATE, 3, 0) = 0x708ec1446000

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\260>\2\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0755, st_size=1901536, ...}) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x708ec1444000

mmap(NULL, 1914496, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x708ec1270000

mmap(0x708ec1292000, 1413120, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x22000) = 0x708ec1292000

mmap(0x708ec13eb000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17b000) = 0x708ec13eb000

mmap(0x708ec143a000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c9000) = 0x708ec143a000

mmap(0x708ec1440000, 13952, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x708ec1440000

close(3) = 0

arch_prctl(ARCH_SET_FS, 0x708ec1445540) = 0

mprotect(0x708ec143a000, 16384, PROT_READ) = 0

mprotect(0x62a0beb2d000, 4096, PROT_READ) = 0

mprotect(0x708ec1476000, 4096, PROT_READ) = 0

munmap(0x708ec1446000, 21784) = 0

fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0

brk(NULL) = 0x62a0d8d28000

brk(0x62a0d8d49000) = 0x62a0d8d49000

fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\270\320\274\321\217 \321\204\320\260\320\271\320\273\320\260"..., 34Введите имя файла:) = 34

read(0,

"\n", 1024) = **1**

access("", F_OK) = **-1 ENOENT (No such file or directory)**

write(1, "\320\236\321\210\320\270\320\261\320\272\320\260: \321\204\320\260\320\271\320\273 " \320\275\320\265 \321"..., 52Ошибка: файл '' не существует

) = 52

exit_group(1) = ?

+++ exited with 1 +++

Вывод

Я составил и отладил программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними. В результате работы программа (основной процесс) создаёт один дочерний процесс. Взаимодействие между процессами осуществляется через системные сигналы/события и/или каналы (pipe).

Основная проблемма оказалась в рассмотрении всех вариантах развития событии и обработке всех случаев и различных данных.