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-23

Студент: Шведова Е. В.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 10.10.25

Москва, 2025

Постановка задачи

Вариант 3.

Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса пишет имя файла, которое будет передано при создании дочернего процесса. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс передает команды пользователя через pipe1, который связан с стандартным входным потоком дочернего процесса. Дочерний процесс при необходимости передает данные в родительский процесс через pipe2. Результаты своей работы дочерний процесс пишет в созданный им файл. Допускается просто открыть файл и писать туда, не перенаправляя стандартный поток вывода.

Пользователь вводит команды вида: «число число число<endline>». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс производит деление первого числа на последующие, а результат выводит в файл. 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 int. Количество чисел может быть произвольным.

Общий метод и алгоритм решения

Родительский процесс запрашивает имя файла и последовательности чисел для вычислений. В дочернем процессе перенаправляет STDIN на чтение из pipe, а STDOUT - в выходной pipe, затем запускает программу child. Родительский процесс передает данные через pipe и читает результаты из pipe, выводя их в консоль. Дочерний процесс читает данные из STDIN построчно, обрабатывает каждую строку: разбивает на числа, проверяет наличие деления на ноль, при обнаружении ошибки отправляет сигнал и завершает работу, при валидных данных - вычисляет результат последовательного деления и записывает в файл, а также возвращает результат родителю.

Использованные системные вызовы:

- pid_t fork(void); – создает дочерний процесс.
- int pipe(int fd) – создание неименованного канала
- ssize_t write(int fd, const void buf, size_t count) – запись данных
- ssize_t read(int fd, void buf, size_t count) – чтение данных
- int open(const char pathname, int flags) – открытие файла
- int close(int fd) – закрытие файлового дескриптора
- int dup2(int oldfd, int newfd) – перенаправление потоков
- int execl(const char path, const char arg, ...) – запуск программы
- pid_t wait(int status) – ожидание завершения процесса
- void exit(int status) – завершение процесса

Код программы

parent.c

```
##include <stdint.h>
#include <unistd.h>
#include <sys/wait.h>
#include <stdlib.h>
#include <stdio.h>
```

```c
#include <string.h>

int main(int argc, char **argv) {
    int parent_to_child[2];
    int child_to_parent[2];

    if (pipe(parent_to_child) == -1 || pipe(child_to_parent) == -1) {
        perror("pipe failed");
        return 1;
    }

    const pid_t child = fork();

    switch (child) {
    case -1: {
        perror("fork failed");
        return 1;
    } break;

    case 0: {
        close(parent_to_child[1]);
        close(child_to_parent[0]);

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

        dup2(child_to_parent[1], STDOUT_FILENO);
        close(child_to_parent[1]);

        execl("./child", "child", NULL);
        perror("exec failed");
        return 1;
    } break;

    default: {
        close(parent_to_child[0]);
        close(child_to_parent[1]);

        printf("Enter filename: ");
        char filename[256];
        if (fgets(filename, sizeof(filename), stdin) == NULL) {
            printf("Error reading filename\n");
            return 1;
        }
        filename[strcspn(filename, "\n")] = '\0';

        write(parent_to_child[1], filename, strlen(filename));
```

```
        write(parent_to_child[1], "\n", 1);

        printf("Enter numbers separated by spaces (e.g., '12 3 4'):\n");
        printf("Press Enter on empty line to exit\n");

        char input[1024];
        while (fgets(input, sizeof(input), stdin)) {
            if (input[0] == '\n') {
                break;
            }

            write(parent_to_child[1], input, strlen(input));

            char response[100];
            ssize_t bytes = read(child_to_parent[0], response,
sizeof(response) - 1);

            if (bytes > 0) {
                response[bytes] = '\0';

                if (strstr(response, "DIVISION_BY_ZERO") != NULL) {
                    printf("Error: Division by zero! Exiting...\n");
                    close(parent_to_child[1]);
                    close(child_to_parent[0]);
                    exit(EXIT_FAILURE);
                }

                printf("Result: %s", response);
            }
        }

        close(parent_to_child[1]);
        close(child_to_parent[0]);
        wait(NULL);
        printf("Program finished successfully.\n");
    } break;
    }

    return 0;
```

child.c

```
}#include <stdint.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>
#include <string.h>
```

```
int main(int argc, char **argv) {
    char buf[4096];
    ssize_t bytes;

    char filename[256];
    if (read(STDIN_FILENO, filename, sizeof(filename)-1) <= 0) {
        const char msg[] = "Error: failed to read filename\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        exit(EXIT_FAILURE);
    }
    filename[strcspn(filename, "\n")] = '\0';

    int file = open(filename, O_WRONLY | O_CREAT | O_TRUNC, 0600);
    if (file == -1) {
        const char msg[] = "Error: failed to open file\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        exit(EXIT_FAILURE);
    }

    while ((bytes = read(STDIN_FILENO, buf, sizeof(buf)-1)) > 0) {
        buf[bytes] = '\0';

        int numbers[100];
        int count = 0;
        char *token = strtok(buf, " \t\n");

        while (token != NULL && count < 100) {
            numbers[count++] = atoi(token);
            token = strtok(NULL, " \t\n");
        }

        if (count < 2) {
            const char error_msg[] = "Error: need at least 2 numbers\n";
            write(STDOUT_FILENO, error_msg, strlen(error_msg));
            write(file, error_msg, strlen(error_msg));
            continue;
        }

        int division_by_zero = 0;
        for (int i = 1; i < count; i++) {
            if (numbers[i] == 0) {
                division_by_zero = 1;
                break;
            }
        }
```

```
        if (division_by_zero) {
            const char error[] = "Error: division by zero\n";
            write(file, error, strlen(error));

            const char signal[] = "DIVISION_BY_ZERO\n";
            write(STDOUT_FILENO, signal, strlen(signal));

            close(file);
            exit(EXIT_FAILURE);
        }

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

        dprintf(file, "%d", numbers[0]);
        for (int i = 1; i < count; i++) {
            dprintf(file, " / %d", numbers[i]);
        }
        dprintf(file, " = %.2f\n", result);

        char response[50];
        snprintf(response, sizeof(response), "%.2f\n", result);
        write(STDOUT_FILENO, response, strlen(response));
    }

    close(file);
    return 0;
}
```

Протокол работы программы

crane@Assus:~/op$ ./parent

Enter filename: results.txt

Enter numbers separated by spaces (e.g., '12 3 4'):

Press Enter on empty line to exit

12 3 4

Result: 1.00

10 4 2

Result: 1.25

10 0 5

Error: Division by zero! Exiting...

```
crane@Assus:~/op$ ./parent
Enter filename: results.txt
Enter numbers separated by spaces (e.g., '12 3 4'):
Press Enter on empty line to exit
1000000 1000 10
Result: 100.00
100 -5 2
Result: -10.00
5
Result: Error: need at least 2 numbers

Program finished successfully.
crane@Assus:~/op$ ./parent
Enter filename: invalid/result.txt
Enter numbers separated by spaces (e.g., '12 3 4'):
Press Enter on empty line to exit
Error: failed to open file

Program finished successfully.

crane@Assus:~/op$ strace -f ./parent
execve("./parent", ["./parent"], 0x7ffefb2bc0b8 /* 77 vars */) = 0
brk(NULL)                               = 0x5de06693f000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7f2ddf31d000
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=75563, ...}) = 0
mmap(NULL, 75563, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f2ddf3ea000
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\220\243\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
fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 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, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f2ddf000000
mmap(0x7f2ddf028000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x28000) = 0x7f2ddf028000
```

```
mmap(0x7f2ddf1b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1b0000) = 0x7f2ddf1b0000
mmap(0x7f2ddf1ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1fe000) = 0x7f2ddf1ff000
mmap(0x7f2ddf205000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f2ddf205000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7f2ddf3e7000
arch_prctl(ARCH_SET_FS, 0x7f2ddf3e7740) = 0
set_tid_address(0x7f2ddf3e7a10)         = 43318
set_robust_list(0x7f2ddf3e7a20, 24)     = 0
rseq(0x7f2ddf3e8060, 0x20, 0, 0x53053053) = 0
mprotect(0x7f2ddf1ff000, 16384, PROT_READ) = 0
mprotect(0x5de0344a5000, 4096, PROT_READ) = 0
mprotect(0x7f2ddf435000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY})
= 0
munmap(0x7f2ddf3ea000, 75563)           = 0
pipe2([3, 4], 0)                        = 0
pipe2([5, 6], 0)                        = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|
SIGCHLDstrace: Process 43319 attached
, child_tidptr=0x7f2ddf3e7a10) = 43319
[pid 43319] set_robust_list(0x7f2ddf3e7a20, 24 <unfinished ...>
[pid 43318] close(3 <unfinished ...>
[pid 43319] <... set_robust_list resumed>) = 0
[pid 43318] <... close resumed>)        = 0
[pid 43318] close(6 <unfinished ...>
[pid 43319] close(4 <unfinished ...>
[pid 43318] <... close resumed>)        = 0
[pid 43319] <... close resumed>)        = 0
[pid 43318] fstat(1,  <unfinished ...>
[pid 43319] close(5 <unfinished ...>
[pid 43318] <... fstat resumed>{st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
[pid 43319] <... close resumed>)        = 0
[pid 43318] getrandom( <unfinished ...>
[pid 43319] dup2(3, 0 <unfinished ...>
[pid 43318] <... getrandom resumed>"\x82\x08\x26\xf9\x98\xf2\x96\x00", 8,
GRND_NONBLOCK) = 8
[pid 43319] <... dup2 resumed>)         = 0
[pid 43318] brk(NULL <unfinished ...>
[pid 43319] close(3 <unfinished ...>
[pid 43318] <... brk resumed>)          = 0x5de06693f000
[pid 43319] <... close resumed>)        = 0
[pid 43318] brk(0x5de066960000 <unfinished ...>
[pid 43319] dup2(6, 1 <unfinished ...>
[pid 43318] <... brk resumed>)          = 0x5de066960000
[pid 43319] <... dup2 resumed>)         = 1
[pid 43318] fstat(0,  <unfinished ...>
```

[pid 43319] close(6 <unfinished ...>
[pid 43318] <... fstat resumed>{st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
[pid 43319] <... close resumed>) = 0
[pid 43318] write(1, "Enter filename: ", 16 <unfinished ...>
[pid 43319] execve("./child", ["child"], 0x7ffd4f2fead8 /* 77 vars */Enter filename: <unfinished ...>
[pid 43318] <... write resumed>) = 16
[pid 43318] read(0, <unfinished ...>
[pid 43319] <... execve resumed>) = 0
[pid 43319] brk(NULL) = 0x61877df34000
[pid 43319] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7eacf30c3000
[pid 43319] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Нет такого файла или каталога)
[pid 43319] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
[pid 43319] fstat(3, {st_mode=S_IFREG|0644, st_size=75563, ...}) = 0
[pid 43319] mmap(NULL, 75563, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7eacf30b0000
[pid 43319] close(3) = 0
[pid 43319] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
[pid 43319] read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"..., 832) = 832
[pid 43319] 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 43319] fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
[pid 43319] 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 43319] mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7eacf2e00000
[pid 43319] mmap(0x7eacf2e28000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7eacf2e28000
[pid 43319] mmap(0x7eacf2fb0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x7eacf2fb0000
[pid 43319] mmap(0x7eacf2fff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7eacf2fff000
[pid 43319] mmap(0x7eacf3005000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7eacf3005000
[pid 43319] close(3) = 0
[pid 43319] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7eacf30ad000
[pid 43319] arch_prctl(ARCH_SET_FS, 0x7eacf30ad740) = 0
[pid 43319] set_tid_address(0x7eacf30ada10) = 43319
[pid 43319] set_robust_list(0x7eacf30ada20, 24) = 0
[pid 43319] rseq(0x7eacf30ae060, 0x20, 0, 0x53053053) = 0
[pid 43319] mprotect(0x7eacf2fff000, 16384, PROT_READ) = 0
[pid 43319] mprotect(0x61873f4bb000, 4096, PROT_READ) = 0
[pid 43319] mprotect(0x7eacf30fb000, 8192, PROT_READ) = 0
[pid 43319] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
[pid 43319] munmap(0x7eacf30b0000, 75563) = 0
[pid 43319] read(0, results.txt
<unfinished ...>
[pid 43318] <... read resumed>"results.txt\n", 1024) = 12
[pid 43318] write(4, "results.txt\n", 12) = 12

```
[pid 43319] <... read resumed>"results.txt\n", 255) = 12
[pid 43318] write(1, "Enter numbers separated by space"..., 52Enter numbers separated by spaces (e.g., "12 3 4"):
) = 52
[pid 43318] write(1, "Press Enter on empty line to exi"..., 34 <unfinished ...>
Press Enter on empty line to exit
[pid 43319] openat(AT_FDCWD, "results.txt", O_WRONLY|O_CREAT|O_TRUNC, 0600 <unfinished ...>
[pid 43318] <... write resumed>)       = 34
[pid 43318] read(0,  <unfinished ...>
[pid 43319] <... openat resumed>)      = 3
[pid 43319] read(0, 12 3 4
 <unfinished ...>
[pid 43318] <... read resumed>"12 3 4\n", 1024) = 7
[pid 43318] write(4, "12 3 4\n", 7)     = 7
[pid 43319] <... read resumed>"12 3 4\n", 1023) = 7
[pid 43318] read(5,  <unfinished ...>
[pid 43319] write(3, "12", 2)          = 2
[pid 43319] write(3, " / 3", 4)        = 4
[pid 43319] write(3, " / 4", 4)        = 4
[pid 43319] write(3, " = 1.00\n", 8)    = 8
[pid 43319] write(1, "1.00\n", 5 <unfinished ...>
[pid 43318] <... read resumed>"1.00\n", 99) = 5
[pid 43319] <... write resumed>)       = 5
[pid 43318] write(1, "Result: 1.00\n", 13 <unfinished ...>
Result: 1.00
[pid 43319] read(0,  <unfinished ...>
[pid 43318] <... write resumed>)       = 13
[pid 43318] read(0,
"\n", 1024)        = 1
[pid 43318] close(4)                   = 0
[pid 43319] <... read resumed>"", 1023) = 0
[pid 43319] close(3 <unfinished ...>
[pid 43318] close(5)                   = 0
[pid 43318] wait4(-1,  <unfinished ...>
[pid 43319] <... close resumed>)       = 0
[pid 43319] exit_group(0)              = ?
[pid 43319] +++ exited with 0 +++
<... wait4 resumed>NULL, 0, NULL)      = 43319
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=43319, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
write(1, "Program finished successfully.\n", 31Program finished successfully.
) = 31
exit_group(0)                          = ?
+++ exited with 0 +++
```

Вывод

В данной лабораторной работе были использованы системные вызовы операционной системы Linux для организации межпроцессного взаимодействия. Реализована архитектура с разделением на родительский и дочерний процессы, общающиеся через два неименованных канала (pipe). Родительский процесс отвечает за взаимодействие с пользователем и управление данными, а дочерний процесс выполняет математические вычисления - проверку входных данных на корректность, обнаружение деления на ноль и вычисление результата последовательного деления чисел с записью операций в файл.