

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: М80-206Б-22

Студент: Свиридов С. Ю.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 20.10.23

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<newline>». 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);` – приостанавливает 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`.

Сначала пользователь в качестве аргумента командной строки пишет имя файла, которое будет использоваться для открытия файла с таким же именем на чтение. Если строка введена корректно, и файл с таким именем существует, то создается дочерний процесс, и происходит переопределение стандартного ввода для дочернего процесса: стандартным вводом теперь является открытый файл, имя которого пользователь указал, и стандартный вывод дочернего процесса переопределяется каналом pipe. Родительский процесс считывает из pipe результат работы дочернего процесса и выводит его на стандартный ввод, если дочерний процесс успешно выполнит проверку деления на ноль. В противном случае 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) {
    if (expression) {
        write(STDOUT_FILENO, message, strlen(message) *
sizeof(char));
        write(STDOUT_FILENO, "\n", 1);
        exit(-1);
    }
}

int main (int argc, char* argv[]) {
    pid_t pid;
    int pipe_1[2];
    if (pipe(pipe_1) == -1) {
        perror("pipe");
        _exit(EXIT_FAILURE);
    }
    if (argc != 2) {
        write(1, "Error: no filename\n", 20);
        exit(EXIT_FAILURE);
    }
}
```

```

int fd = open(argv[1], O_RDONLY);
check_error(fd == -1, "Can't open file");
pid = fork();
if (pid == -1) {
    perror("fork");
    return -1;
}
else if (pid == 0) {
    close(pipe_1[0]);
    check_error(dup2(fd, STDIN_FILENO) < 0, "Error dub");
    dup2(pipe_1[1], STDOUT_FILENO);
    execl("./child", "./.child", NULL);
    perror("execl");
    return 1;
}
else {
    check_error((pid == -1), "Process error");
    close(pipe_1[1]);
    wait(0);
    float result;
    char answer[50];
    while ((read(pipe_1[0], &result, sizeof(float))) > 0) {
        if (result == -1) {
            write(STDOUT_FILENO, "Attempt to divide by zero\n", 27);
            exit(EXIT_FAILURE);
        }
        sprintf(answer, "%f\n", result);
        check_error(write(STDOUT_FILENO, answer, strlen(answer))
== -1, "Write error\n");
        check_error(write(STDOUT_FILENO, "\n", 1) == -1, "Write
error\n");
    }
}
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>

#define buf_size 100


int main() {

    int c;

    bool not_end = true;

    float nmbr = 0;

    float result = 0;

    float first = 0;

    int k = 0;

    float dot = 0;

    int count = 0;

    float numbers[100];


    do {

        if (not_end) {

            if (c <= '9' && c >= '0') {

                if (nmbr != 0 && (floor(nmbr) != nmbr)) {

                    if (c == '0') {

                        k += 1;


```

```

        nmbr += 0;
    }
    else {
        dot = c - '0';
        nmbr = (nmbr + 0.1) + (dot / pow(10, k + 1));
    }
}
else {
    nmbr = nmbr * 10 + c - '0';
}
}
if (c == '.') {
    nmbr = nmbr - 0.1;
}
if (c == ' ' || c == '\n' || c == EOF) {
    numbers[count] = nmbr;
    nmbr = 0;
    count++;
    if (c == '\n' || c == EOF) {
        first = numbers[0];
        for (int i = 1; i < count; i++) {
            if (numbers[i] == 0) {
                result = -1;
                write(STDOUT_FILENO, &result,
sizeof(result));
            }
            result = first / numbers[i];
            first = result;
        }
    }
}

```

```
        not_end = false;

        first = 0;

        count = 0;

    }

}

}

if (c == '\n' || c == EOF) {

    write(STDOUT_FILENO, &result, sizeof(result));

    result = 0;

    k = 0;

    dot = 0;

    not_end = true;

}

} while((read(STDIN_FILENO, &c, sizeof(char))) > 0);

return 0;

}
```

Протокол работы программы

Тестирование:

```
stepan@stepan-ASUS:~/Рабочий стол/учеба/labs 3 sem/OSI/laba1$ ./parent  
file.txt
```

900.000122

1.000000

14.273480

0.666667

90.000000

38.136364

4329.239258

2880.000000

Attempt to divide by zero

Strace

```
stepan@stepan-ASUS:~/Рабочий стол/учеба/labs 3 sem/OSI/lab1$ strace -f ./parent file.txt
```

```
execve("./parent", [ "./parent", "file.txt" ], 0x7ffc8e906880 /* 59 vars */) = 0
```

```
brk(NULL) = 0x563978a26000
```

```
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc5bc53310) = -1 EINVAL  
(Недопустимый аргумент)
```

```
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)  
= 0x7f21e8843000
```

```
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
```

```
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=88411, ...},  
AT_EMPTY_PATH) = 0
```

```
mmap(NULL, 88411, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f21e882d000
```

```
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\3\0>\0\1\0\0\0P\237\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
```

```
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\244;\374\204(\337f#\315I\214\234\f\256\27
1\32"... , 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) =
0x7f21e8600000

mmap(0x7f21e8628000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f21e8628000

mmap(0x7f21e87bd000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7f21e87bd000

mmap(0x7f21e8815000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x214000) = 0x7f21e8815000

mmap(0x7f21e881b000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f21e881b000

close(3) = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
= 0x7f21e882a000

arch_prctl(ARCH_SET_FS, 0x7f21e882a740) = 0

set_tid_address(0x7f21e882aa10) = 5467

set_robust_list(0x7f21e882aa20, 24) = 0

rseq(0x7f21e882b0e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7f21e8815000, 16384, PROT_READ) = 0

mprotect(0x563976c6a000, 4096, PROT_READ) = 0

mprotect(0x7f21e887d000, 8192, PROT_READ) = 0

```

```
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,  
rlim_max=RLIM64_INFINITY}) = 0
```

```
munmap(0x7f21e882d000, 88411) = 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 5468  
attached  
, child_tidptr=0x7f21e882aa10) = 5468
```

```
[pid 5467] close(4 <unfinished ...>
```

```
[pid 5468] set_robust_list(0x7f21e882aa20, 24 <unfinished ...>
```

```
[pid 5467] <... close resumed> = 0
```

```
[pid 5468] <... set_robust_list resumed> = 0
```

```
[pid 5467] wait4(-1, <unfinished ...>
```

```
[pid 5468] close(3) = 0
```

```
[pid 5468] dup2(5, 0) = 0
```

```
[pid 5468] dup2(4, 1) = 1
```

```
[pid 5468] execve("./child", ["/.child"], 0x7ffc5bc534f0 /* 59 vars */) = 0
```

```
[pid 5468] brk(NULL) = 0x562c8e65a000
```

```
[pid 5468] arch_prctl(0x3001 /* ARCH_??? */, 0x7fff59b07440) = -1 EINVAL  
(Недопустимый аргумент)
```

```
[pid 5468] mmap(NULL, 8192, PROT_READ|PROT_WRITE,  
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fbe2c0cd000
```

```
[pid 5468] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Нет такого  
файла или каталога)
```

```

[pid 5468] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

[pid 5468] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=88411, ...},
AT_EMPTY_PATH) = 0

[pid 5468] mmap(NULL, 88411, PROT_READ, MAP_PRIVATE, 3, 0) =
0x7fbe2c0b7000

[pid 5468] close(3) = 0

[pid 5468] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6",
O_RDONLY|O_CLOEXEC) = 3

[pid 5468] 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"... , 832)
= 832

[pid 5468] newfstatat(3, "", {st_mode=S_IFREG|0644,
st_size=940560, ...}, AT_EMPTY_PATH) = 0

[pid 5468] mmap(NULL, 942344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7fbe2bfd0000

[pid 5468] mmap(0x7fbe2bfd0000, 507904, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe000) = 0x7fbe2bfd0000

[pid 5468] mmap(0x7fbe2c05a000, 372736, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8a000) = 0x7fbe2c05a000

[pid 5468] mmap(0x7fbe2c0b5000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe4000) = 0x7fbe2c0b5000

[pid 5468] close(3) = 0

[pid 5468] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6",
O_RDONLY|O_CLOEXEC) = 3

[pid 5468] 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"... ,
832) = 832

```

```

[pid 5468] 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 5468] 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 5468] pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\244;\374\204(\337f#\315I\214\234\f\256\27
1\32"..., 68, 896) = 68

[pid 5468] newfstatat(3, "", {st_mode=S_IFREG|0755,
st_size=2216304, ...}, AT_EMPTY_PATH) = 0

[pid 5468] 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 5468] mmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7fbe2bc00000

[pid 5468] mmap(0x7fbe2bc28000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7fbe2bc28000

[pid 5468] mmap(0x7fbe2bdbd000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7fbe2bdbd000

[pid 5468] mmap(0x7fbe2be15000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x214000) = 0x7fbe2be15000

[pid 5468] mmap(0x7fbe2be1b000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fbe2be1b000

[pid 5468] close(3) = 0

[pid 5468] mmap(NULL, 12288, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fbe2bfcd000

[pid 5468] arch_prctl(ARCH_SET_FS, 0x7fbe2bfcd740) = 0

[pid 5468] set_tid_address(0x7fbe2bfcd10) = 5468

[pid 5468] set_robust_list(0x7fbe2bfcd20, 24) = 0

```

```
[pid 5468] rseq(0x7fbe2bfce0e0, 0x20, 0, 0x53053053) = 0

[pid 5468] mprotect(0x7fbe2be15000, 16384, PROT_READ) = 0

[pid 5468] mprotect(0x7fbe2c0b5000, 4096, PROT_READ) = 0

[pid 5468] mprotect(0x562c8dd73000, 4096, PROT_READ) = 0

[pid 5468] mprotect(0x7fbe2c107000, 8192, PROT_READ) = 0

[pid 5468] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0

[pid 5468] munmap(0x7fbe2c0b7000, 88411) = 0

[pid 5468] read(0, "9", 1) = 1

[pid 5468] read(0, " ", 1) = 1

[pid 5468] read(0, "0", 1) = 1

[pid 5468] read(0, "0", 1) = 1

[pid 5468] read(0, ".", 1) = 1

[pid 5468] read(0, "0", 1) = 1

[pid 5468] read(0, "1", 1) = 1

[pid 5468] read(0, " ", 1) = 1

[pid 5468] read(0, "1", 1) = 1

[pid 5468] read(0, "\n", 1) = 1

[pid 5468] read(0, "1", 1) = 1

[pid 5468] read(0, "0", 1) = 1

[pid 5468] read(0, " ", 1) = 1

[pid 5468] read(0, "2", 1) = 1

[pid 5468] read(0, ".", 1) = 1

[pid 5468] read(0, "5", 1) = 1
```

[pid 5468] read(0, " ", 1)	= 1
[pid 5468] read(0, "4", 1)	= 1
[pid 5468] read(0, "\n", 1)	= 1
[pid 5468] read(0, "1", 1)	= 1
[pid 5468] read(0, "0", 1)	= 1
[pid 5468] read(0, "0", 1)	= 1
[pid 5468] read(0, "0", 1)	= 1
[pid 5468] read(0, " ", 1)	= 1
[pid 5468] read(0, "7", 1)	= 1
[pid 5468] read(0, ".", 1)	= 1
[pid 5468] read(0, "0", 1)	= 1
[pid 5468] read(0, "0", 1)	= 1
[pid 5468] read(0, "6", 1)	= 1
[pid 5468] read(0, " ", 1)	= 1
[pid 5468] read(0, "2", 1)	= 1
[pid 5468] read(0, " ", 1)	= 1
[pid 5468] read(0, "5", 1)	= 1
[pid 5468] read(0, "\n", 1)	= 1
[pid 5468] read(0, "2", 1)	= 1
[pid 5468] read(0, " ", 1)	= 1
[pid 5468] read(0, "1", 1)	= 1
[pid 5468] read(0, " ", 1)	= 1
[pid 5468] read(0, "3", 1)	= 1
[pid 5468] read(0, "\n", 1)	= 1

[pid 5468] read(0, "9", 1)	= 1
[pid 5468] read(0, " ", 1)	= 1
[pid 5468] read(0, "0", 1)	= 1
[pid 5468] read(0, ".", 1)	= 1
[pid 5468] read(0, "1", 1)	= 1
[pid 5468] read(0, "\n", 1)	= 1
[pid 5468] read(0, "8", 1)	= 1
[pid 5468] read(0, "3", 1)	= 1
[pid 5468] read(0, "9", 1)	= 1
[pid 5468] read(0, " ", 1)	= 1
[pid 5468] read(0, "2", 1)	= 1
[pid 5468] read(0, "2", 1)	= 1
[pid 5468] read(0, "\n", 1)	= 1
[pid 5468] read(0, "3", 1)	= 1
[pid 5468] read(0, "9", 1)	= 1
[pid 5468] read(0, "8", 1)	= 1
[pid 5468] read(0, "2", 1)	= 1
[pid 5468] read(0, "9", 1)	= 1
[pid 5468] read(0, " ", 1)	= 1
[pid 5468] read(0, "9", 1)	= 1
[pid 5468] read(0, ".", 1)	= 1
[pid 5468] read(0, "2", 1)	= 1
[pid 5468] read(0, "\n", 1)	= 1
[pid 5468] read(0, "2", 1)	= 1


```
[pid 5468] read(0, "8", 1)          = 1
[pid 5468] read(0, "8", 1)          = 1
[pid 5468] read(0, " ", 1)          = 1
[pid 5468] read(0, "0", 1)          = 1
[pid 5468] read(0, "0", 1)          = 1
[pid 5468] read(0, "0", 1)          = 1
[pid 5468] read(0, "0", 1)          = 1
[pid 5468] read(0, "0", 1)          = 1
[pid 5468] read(0, "0", 1)          = 1
[pid 5468] read(0, "0", 1)          = 1
[pid 5468] read(0, ".", 1)          = 1
[pid 5468] read(0, "1", 1)          = 1
[pid 5468] read(0, "\n", 1)         = 1
[pid 5468] read(0, "2", 1)          = 1
[pid 5468] read(0, "9", 1)          = 1
[pid 5468] read(0, "4", 1)          = 1
[pid 5468] read(0, "8", 1)          = 1
[pid 5468] read(0, "2", 1)          = 1
[pid 5468] read(0, " ", 1)          = 1
[pid 5468] read(0, "0", 1)          = 1
[pid 5468] read(0, "\n", 1)         = 1
[pid 5468] read(0, "", 1)           = 0
[pid 5468] exit_group(0)            = ?
[pid 5468] +++ exited with 0 +++
```

```
<... wait4 resumed>NULL, 0, NULL)      = 5468
```

```
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=5468,
si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
```

```
read(3, "", 4) = 0
```

```
exit_group(0) = ?
```

```
+++ exited with 0 +++
```

Вывод

Благодаря выполнению данной работы я изучил принцип работы с каналами для межпроцессорного взаимодействия. Я немного пощупал процесс перенаправления ввода и вывода процесса, узнал, что такое файловый дескриптор и понял, что важно вовремя закрывать их. Во время выполнения лабораторной были 2 основных трудности: сначала мне было трудно написать рабочий парсер, но потом, по совету одногруппника, я все таки смог это сделать. После этого у возникли вопросы по поводу работы функции write, я не мог понять принцип того, как она обрабатывает тип float. В целом, я подчеркнул много нового для себя, что поможет мне в написании будущих более сложных кодов.