

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

Студентка: Коломытцева Е. А.

Преподаватель: Миронов Е.С.

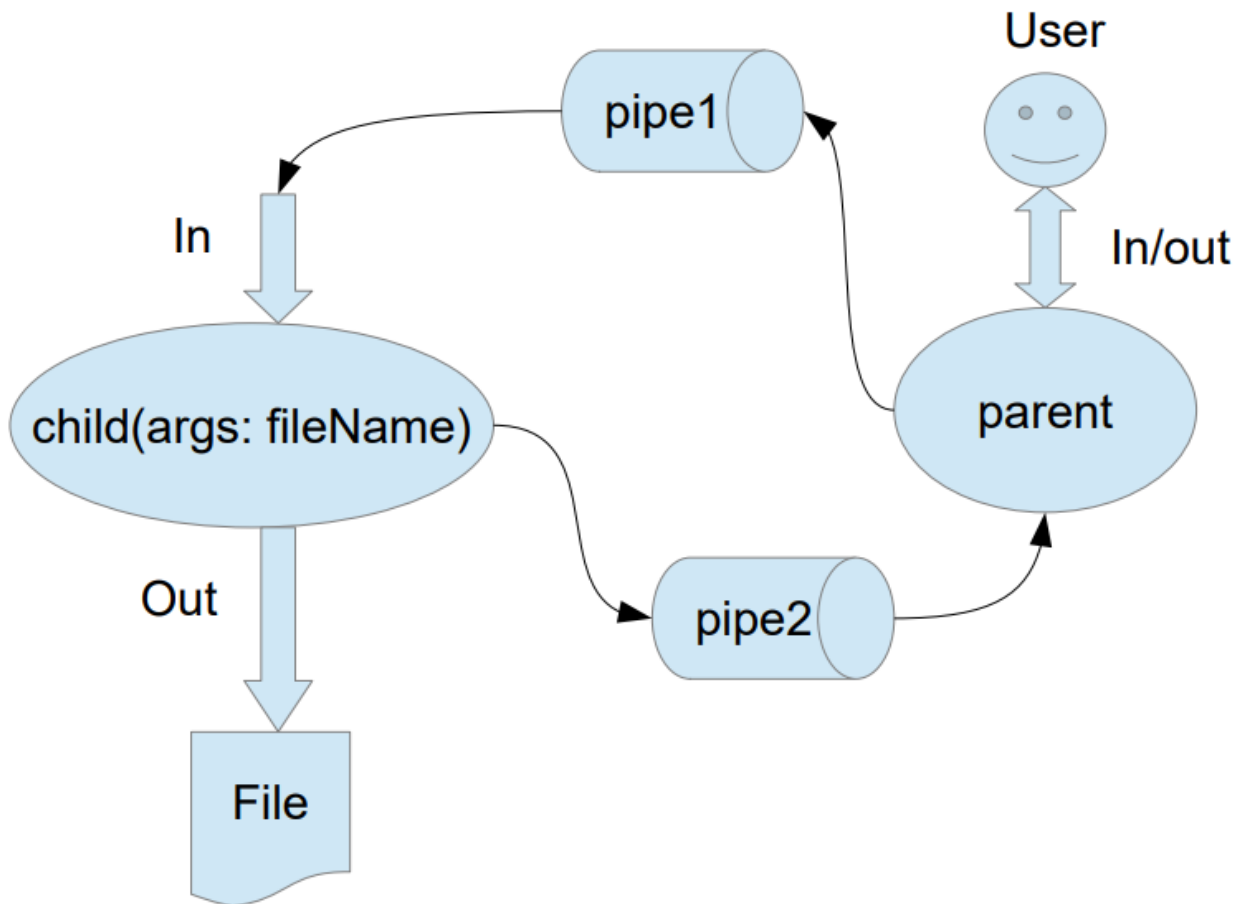
Оценка: \_\_\_\_\_

Дата: 09.10.23

Москва, 2023

# Постановка задачи

## Вариант 1.



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.

Задание: Пользователь вводит команды вида: «число число число<newline>». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс считает их сумму и выводит её в файл. Числа имеют тип int. Количество чисел может быть произвольным.

## Общий метод и алгоритм решения

### Используемые системные вызовы:

- `pid_t fork(void);` – создает дочерний процесс.
- `int pipe(int *fd);` – создает канал, который используется для связи дочернего и родительского процессов
- `ssize_t write(int fd, const void buf[count], size_t count)` - записывает `size_t count` байт в указанный файловый дескриптор `fd`

- `ssize_t read(int fd, void buf[.count], size_t count)` - считывает `size_t count` байт в указанный файловый дескриптор `fd`
- `int close(int fd)` - закрывает файловый дескриптор `fd`
- `int dup2(int oldfd, int newfd)` - дублирует файловый дескриптор `newfd` на место дескриптора `oldfd`
- `int execl(const char *pathname, const char *arg, .../*, (char *) NULL */)` - исполняет указанные файлы

#### Программа состоит из 5 файлов:

`main.cpp` – родительский процесс

`child.cpp` – дочерний процесс

`func.cpp` – функция вычисления суммы чисел из передаваемой строки

`func.h` – заголовочный файл для `func.cpp`

`Makefile` – файл, который собирает программу для выполнения задания

#### Команды для запуска программы:

`make`

`./main`

#### Пример ввода и вывода:

Enter filename:

`sum.txt`

Enter numbers:

`12 3 4`

Сумма чисел: 19

#### Описание программы и алгоритм выполнения работы:

Для лучшего понимания работы с процессами изначально я сделала задание в одном файле `main_and_child.cpp`. Далее с помощью системных вызовов `dup2` и `execl` я разделила процессы на два файла, где родительский процесс считывает с ввода имя файла и числа и передает их по каналу `pipe1`. Дочерний процесс получает название файла и числа из `STDIN_FILENO` и с помощью функции `sum_numbers` вычисляет сумму передаваемых чисел, создает файл с переданным именем и записывает сумму в файл.

Пройдемся по коду более подробно:

**В файле `main.cpp`** создаются два канала: `pipe1[2]` для передачи названия файла и чисел, `pipe2[2]` для передачи суммы. Вместе с этим происходит обработка на ошибку создания каналов. Далее с помощью `fork` создается дочерний процесс и обрабатывается ошибка создания процесса.

Далее мы проверяем, если процесс = 0, то есть является дочерним, программа закрывает записывающий конец канала `pipe1` и читающий конец канала `pipe2`, чтобы избежать проблем с утечкой ресурсов и некорректным поведением программы. Затем с помощью `dup2` программа выполняет перенаправление стандартного потока ввода (`STDIN_FILENO`) на читающий конец канала `pipe1[0]`. Если перенаправление завершается с ошибкой (возвращает -1), то условие `if` срабатывает, и в блоке `if` выводится сообщение об ошибке с использованием функции  `perror`. Далее программа выполняет перенаправление стандартного потока вывода (`STDOUT_FILENO`) на записывающий конец канала `pipe2[1]`. После программа выполняет вызов исполняемого файла с именем `"/child"` в дочернем процессе. Функция `execl` заменяет текущий образ процесса новым

образом, указанным в аргументах. Далее программа закрывает читающий конец канала `pipe1` и записывающий конец канала `pipe2`.

Иначе, то есть процесс является родительским, закрываем читающий конец канала `pipe1` и записывающий конец канала `pipe2`. Создается три массива: `filename_nums[200]` – строка для хранения названия файла и чисел, разделенных с помощью символа “\n” (Пример строки: “sum.txt\n1 2 3 4 5”); `filename[100]` – строка для хранения названия файла; `input_nums[100]` – строка для хранения цифр, записанных через пробел. Далее считывается со ввода название файла, устанавливается нулевой символ (“\0”) в строке `filename` на позиции, где найден символ новой строки (“\n”) для устранения ошибок с дополнительным последним символом. Далее со ввода считывается строка с цифрами в виде “<число> <число> ... <число>\n”. С помощью функций `strcpy` и `strcat` строки `filename` и `input_nums` сливаются в одну `filename_nums`. Далее записываем полученную строку в записывающий конец канала `pipe1`. С читающего конца канала `pipe2` читается сумма, переданная из дочернего процесса. Далее закрываются записывающий и читающий концы каналов `pipe1` и `pipe2` соответственно.

**В файле `child.cpp`** также создается 3 массива для получения с помощью `read` строки с именем файла и числами и последующего их разделения на две строки. Далее создается файл с переданным именем и с помощью функции `sum_numbers`, которая преобразует передаваемую строку в массив чисел и вычисляет сумму. Затем сумма записывается в файл с помощью функции `fprintf` и через `write` дочерний процесс передает родительскому значение суммы.

## Код программы

### main.cpp

```
#include <iostream>
using namespace std;
#include "func.h"

int main() {
    int pipe1[2], pipe2[2];
    if (pipe(pipe1) == -1) {
        cerr << "Ошибка при создании канала." << endl;
        return 1;
    }
    if (pipe(pipe2) == -1) {
        cerr << "Ошибка при создании канала." << endl;
        return 1;
    }

    pid_t child_pid = fork();

    if (child_pid == -1) {
        cerr << "Ошибка при создании дочернего процесса." << endl;
        return 1;
    }
    ///////////////////////////////////////////////////////////////////
    if (child_pid == 0) { // Дочерний процесс
        close(pipe1[1]); // Закрываем записывающий конец канала
        close(pipe2[0]);
        if (dup2(pipe1[0], STDIN_FILENO) == -1) {
            perror("Call dup2 was ended with error: ");
            exit(-1);
        }

        if (dup2(pipe2[1], STDOUT_FILENO) == -1) {
            perror("dup2 out ");
            exit(-1);
        }

        if (execl("./child", "./child", NULL) == -1) {
            perror("Call execl was ended with error: ");
            exit(-1);
        }
        close(pipe1[0]);
        close(pipe2[1]);
    }
    ///////////////////////////////////////////////////////////////////
    } else {
        // Родительский процесс
        close(pipe1[0]); // Закрываем читающий конец 1 канала
```

```

close(pipe2[1]);

char filename_nums[200];
char filename[100];
char input_nums[100];
printf("Enter filename:\n");
fgets(filename, sizeof(filename), stdin);
filename[strcspn(filename, "\n")] = '\0';

int suma = 0;
printf("Enter numbers:\n");
fgets(input_nums, sizeof(input_nums), stdin);
strcpy(filename_nums, filename);
strcat(filename_nums, "\n");
strcat(filename_nums, input_nums);

write(pipe1[1], filename_nums,
      strlen(filename_nums) + 1); // Записываем имя файла в канал

read(pipe2[0], &suma, sizeof(int));
printf("Сумма чисел: %d\n", suma);

close(pipe1[1]);
close(pipe2[0]); // Закрываем записывающий конец канала

wait(NULL);
}
}

```

### **child.cpp**

```

#include <iostream>

#include "func.h"
using namespace std;

int main() {
    char fname_nums[200];
    char fname[100];
    char nums[100];
    read(STDIN_FILENO, fname_nums, sizeof(fname_nums));
    char* token = strtok(fname_nums, "\n");
    if (token != NULL) {
        strncpy(fname, token, sizeof(fname));
        fname[sizeof(fname) - 1] =
            '\0'; // Убедимся, что строка города завершена нулевым символом
    } else {
        printf("Ошибка разбора строки\n");
        return 1;
    }
}

```

```

token = strtok(NULL, "\n");
if (token != NULL) {
    strncpy(nums, token, sizeof(nums));
    nums[sizeof(nums) - 1] =
        '\0'; // Убедимся, что строка улицы завершена нулевым символом
} else {
    printf("Ошибка разбора строки\n");
    return 1;
}
// Читаем имя файла из канала
FILE* file = fopen(fname, "w");
if (file == NULL) {
    fprintf(stderr, "Failed to open the file\n");
    return 1;
}

int suma = sum_numbers(nums);
fprintf(file, "%d", suma);
write(STDOUT_FILENO, &suma, sizeof(int)); // Записываем сумму в канал
}

```

### **func.cpp**

```
#include "func.h"
```

```

int sum_numbers(char input_nums[100]) {
    int sum = 0;
    int numbers[100];
    int count = 0;
    char* token = strtok(input_nums, " ");
    while (token != NULL && count < 100) {
        numbers[count] = atoi(token); // Convert string to integer
        count++;
        token = strtok(NULL, " ");
    }
    for (int num = 0; num < count; num++) {
        sum += numbers[num];
    }
    return sum;
}

```

### **func.h**

```

#include <fcntl.h>
#include <string.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

```

```
#include <iostream>
```

```
int sum_numbers(char input_nums[100]);
```

## **Makefile**

```
CC = g++
CFLAGS = -Wall -Wextra

all: main child

main: main.cpp
    $(CC) $(CFLAGS) -o main main.cpp

child: child.cpp func.cpp
    $(CC) $(CFLAGS) -o child child.cpp func.cpp

clean:
    rm -f main child *.txt
```

## **Протокол работы программы**

### **Тестирование:**

```
katya@katya:~/MAI_2/OS/lab1$ make
g++ -Wall -Wextra -o main main.cpp
g++ -Wall -Wextra -o child child.cpp func.cpp
katya@katya:~/MAI_2/OS/lab1$ ./main
Enter filename:
sum.txt
Enter numbers:
23 213
Сумма чисел: 236
katya@katya:~/MAI_2/OS/lab1$ cat sum.txt
236
katya@katya:~/MAI_2/OS/lab1$ ./main
Enter filename:
file
Enter numbers:
76 5 4 2 3 4 5
Сумма чисел: 99
katya@katya:~/MAI_2/OS/lab1$ cat file
99
```

### **Strace:**

```
$ strace -f ./main

execve("./main", [ "./main" ], 0x7ffc0f24bdf8 /* 47 vars */) = 0

brk(NULL)                               = 0x55b32bf86000

arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc66835ff0) = -1 EINVAL (Недопустимый аргумент)
```



```

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fe5370ab000

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=102171, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 102171, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fe537092000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.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\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2522552, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 2535872, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fe536e00000

mmap(0x7fe536e9c000, 1249280, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x9c000) = 0x7fe536e9c000

mmap(0x7fe536fcd000, 577536, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1cd000) = 0x7fe536fcd000

mmap(0x7fe53705a000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x25a000) = 0x7fe53705a000

mmap(0x7fe537068000, 12736, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7fe537068000

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\0\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"..., 784, 64)
= 784

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2072888, ...}, 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"..., 784, 64)
= 784

mmap(NULL, 2117488, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fe536a00000

mmap(0x7fe536a22000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x22000) = 0x7fe536a22000

mmap(0x7fe536b9a000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x19a000) = 0x7fe536b9a000

mmap(0x7fe536bf2000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1f1000) = 0x7fe536bf2000

mmap(0x7fe536bf8000, 53104, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7fe536bf8000

close(3) = 0

```

```

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.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\0\0\0\0\0\0\0"... , 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=948816, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 950520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fe536d17000
mmap(0x7fe536d25000, 516096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe000) = 0x7fe536d25000
mmap(0x7fe536da3000, 372736, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x8c000) = 0x7fe536da3000
mmap(0x7fe536dfe000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe6000) = 0x7fe536dfe000
close(3) = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"... , 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=141872, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 144232, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fe53706e000
mmap(0x7fe537071000, 110592, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x3000) = 0x7fe537071000
mmap(0x7fe53708c000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e000)
= 0x7fe53708c000
mmap(0x7fe537090000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x21000) = 0x7fe537090000
close(3) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fe53706c000
arch_prctl(ARCH_SET_FS, 0x7fe53706d440) = 0
set_tid_address(0x7fe53706d710) = 48645
set_robust_list(0x7fe53706d720, 24) = 0
rseq(0x7fe53706dd60, 0x20, 0, 0x53053053) = 0
mprotect(0x7fe536bf2000, 16384, PROT_READ) = 0
mprotect(0x7fe537090000, 4096, PROT_READ) = 0
mprotect(0x7fe536dfe000, 4096, PROT_READ) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fe536d15000
mprotect(0x7fe53705a000, 45056, PROT_READ) = 0
mprotect(0x55b32b875000, 4096, PROT_READ) = 0
mprotect(0x7fe5370e0000, 8192, PROT_READ) = 0

```

```

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7fe537092000, 102171) = 0
futex(0x7fe5370687fc, FUTEX_WAKE_PRIVATE, 2147483647) = 0
getrandom("\x27\x98\xa4\xa7\x44\x2c\xe7\x1d", 8, GRND_NONBLOCK) = 8
brk(NULL) = 0x55b32bf86000
brk(0x55b32bfa7000) = 0x55b32bfa7000
pipe2([3, 4], 0) = 0
pipe2([5, 6], 0) = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7fe53706d710) = 48646
strace: Process 48646 attached
[pid 48645] close(3 <unfinished ...>
[pid 48646] set_robust_list(0x7fe53706d720, 24 <unfinished ...>
[pid 48645] <... close resumed>) = 0
[pid 48646] <... set_robust_list resumed>) = 0
[pid 48645] close(6) = 0
[pid 48646] close(4 <unfinished ...>
[pid 48645] newfstatat(1, "", <unfinished ...>
[pid 48646] <... close resumed>) = 0
[pid 48645] <... newfstatat resumed>{st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0),
...}, AT_EMPTY_PATH) = 0
[pid 48646] close(5) = 0
[pid 48645] write(1, "Enter filename:\n", 16 <unfinished ...>
[pid 48646] dup2(3, 0Enter filename:
<unfinished ...>
[pid 48645] <... write resumed>) = 16
[pid 48646] <... dup2 resumed>) = 0
[pid 48645] newfstatat(0, "", <unfinished ...>
[pid 48646] dup2(6, 1 <unfinished ...>
[pid 48645] <... newfstatat resumed>{st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0),
...}, AT_EMPTY_PATH) = 0
[pid 48646] <... dup2 resumed>) = 1
[pid 48645] read(0, <unfinished ...>
[pid 48646] execve("./child", ["./child"], 0x7ffc66836158 /* 47 vars */) = 0

```

```

[pid 48646] brk(NULL) = 0x56366dd34000

[pid 48646] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffcb5e76240) = -1 EINVAL
(Недопустимый аргумент)

[pid 48646] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f0e895b2000

[pid 48646] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Нет такого файла или
каталога)

[pid 48646] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 4

[pid 48646] newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=102171, ...},
AT_EMPTY_PATH) = 0

[pid 48646] mmap(NULL, 102171, PROT_READ, MAP_PRIVATE, 4, 0) = 0x7f0e89599000

[pid 48646] close(4) = 0

[pid 48646] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.so.6",
O_RDONLY|O_CLOEXEC) = 4

[pid 48646] read(4,
"\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0"..., 832) = 832

[pid 48646] newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=2522552, ...},
AT_EMPTY_PATH) = 0

[pid 48646] mmap(NULL, 2535872, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 4, 0) =
0x7f0e89200000

[pid 48646] mmap(0x7f0e8929c000, 1249280, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x9c000) = 0x7f0e8929c000

[pid 48646] mmap(0x7f0e893cd000, 577536, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x1cd000) = 0x7f0e893cd000

[pid 48646] mmap(0x7f0e8945a000, 57344, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x25a000) = 0x7f0e8945a000

[pid 48646] mmap(0x7f0e89468000, 12736, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f0e89468000

[pid 48646] close(4) = 0

[pid 48646] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 4

[pid 48646] read(4, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P<\2\0\0\0\0"...,
832) = 832

[pid 48646] pread64(4,
"\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"..., 784, 64) = 784

[pid 48646] newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=2072888, ...},
AT_EMPTY_PATH) = 0

[pid 48646] pread64(4,
"\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"..., 784, 64) = 784

[pid 48646] mmap(NULL, 2117488, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 4, 0) =
0x7f0e88e00000

```

```

[pid 48646] mmap(0x7f0e88e22000, 1540096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x22000) = 0x7f0e88e22000

[pid 48646] mmap(0x7f0e88f9a000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x19a000) = 0x7f0e88f9a000

[pid 48646] mmap(0x7f0e88ff2000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x1f1000) = 0x7f0e88ff2000

[pid 48646] mmap(0x7f0e88ff8000, 53104, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f0e88ff8000

[pid 48646] close(4) = 0

[pid 48646] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 4

[pid 48646] read(4,
"\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0"... , 832) = 832

[pid 48646] newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=948816, ...},
AT_EMPTY_PATH) = 0

[pid 48646] mmap(NULL, 950520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 4, 0) =
0x7f0e894b0000

[pid 48646] mmap(0x7f0e894be000, 516096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0xe000) = 0x7f0e894be000

[pid 48646] mmap(0x7f0e8953c000, 372736, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x8c000) = 0x7f0e8953c000

[pid 48646] mmap(0x7f0e89597000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0xe6000) = 0x7f0e89597000

[pid 48646] close(4) = 0

[pid 48646] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC)
= 4

[pid 48646] read(4,
"\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0"... , 832) = 832

[pid 48646] newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=141872, ...},
AT_EMPTY_PATH) = 0

[pid 48646] mmap(NULL, 144232, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 4, 0) =
0x7f0e8948c000

[pid 48646] mmap(0x7f0e8948f000, 110592, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x3000) = 0x7f0e8948f000

[pid 48646] mmap(0x7f0e894aa000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
4, 0x1e000) = 0x7f0e894aa000

[pid 48646] mmap(0x7f0e894ae000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x21000) = 0x7f0e894ae000

[pid 48646] close(4) = 0

[pid 48646] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f0e8948a000

```

```

[pid 48646] arch_prctl(ARCH_SET_FS, 0x7f0e8948b440) = 0
[pid 48646] set_tid_address(0x7f0e8948b710) = 48646
[pid 48646] set_robust_list(0x7f0e8948b720, 24) = 0
[pid 48646] rseq(0x7f0e8948bd60, 0x20, 0, 0x53053053) = 0
[pid 48646] mprotect(0x7f0e88ff2000, 16384, PROT_READ) = 0
[pid 48646] mprotect(0x7f0e894ae000, 4096, PROT_READ) = 0
[pid 48646] mprotect(0x7f0e89597000, 4096, PROT_READ) = 0
[pid 48646] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f0e89488000
[pid 48646] mprotect(0x7f0e8945a000, 45056, PROT_READ) = 0
[pid 48646] mprotect(0x56366d9ee000, 4096, PROT_READ) = 0
[pid 48646] mprotect(0x7f0e895e7000, 8192, PROT_READ) = 0
[pid 48646] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
[pid 48646] munmap(0x7f0e89599000, 102171) = 0
[pid 48646] futex(0x7f0e894687fc, FUTEX_WAKE_PRIVATE, 2147483647) = 0
[pid 48646] getrandom("\xa0\xb3\x40\x40\x83\x7b\x4c\x81", 8, GRND_NONBLOCK) = 8
[pid 48646] brk(NULL) = 0x56366dd34000
[pid 48646] brk(0x56366dd55000) = 0x56366dd55000
[pid 48646] read(0,
<unfinished ...>
[pid 48645] <... read resumed>"\n", 1024) = 1
[pid 48645] write(1, "Enter numbers:\n", 15Enter numbers:
) = 15
[pid 48645] read(0,
"\n", 1024) = 1
[pid 48645] write(4, "\n\n\0", 3) = 3
[pid 48646] <... read resumed>"\n\n\0", 200) = 3
[pid 48645] read(5, <unfinished ...>
[pid 48646] newfstatat(1, "", {st_mode=S_IFIFO|0600, st_size=0, ...}, AT_EMPTY_PATH) =
0
[pid 48646] write(1, "\320\236\321\210\320\270\320\261\320\272\320\260
\321\200\320\260\320\267\320\261\320\276\321\200\320\260 \321\201\321\202"... , 41) = 41
[pid 48645] <... read resumed>"\320\236\321\210", 4) = 4
[pid 48646] exit_group(1) = ?

```

```

[pid 48645] write(1, "\320\241\321\203\320\274\320\274\320\260
\321\207\320\270\321\201\320\265\320\273: -19995282"... , 35Сумма чисел: -1999528240

) = 35

[pid 48645] close(4) = 0

[pid 48645] close(5) = 0

[pid 48645] wait4(-1, <unfinished ...>

[pid 48646] +++ exited with 1 +++

<... wait4 resumed>NULL, 0, NULL) = 48646

--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=48646, si_uid=1000,
si_status=1, si_utime=0, si_stime=0} ---

exit_group(0) = ?

+++ exited with 0 +++

```

## Вывод

В этой лабораторной работе я научилась работать с процессами, поняла как передавать данные между ними и в целом познакомилась с абсолютно новыми для меня утилитами, как `execl`, `dup2`, `fork`, `pipe`, `read` и `write`.

Изначально лабораторная вызвала некоторые трудности, так как примеры, предлагаемые в папке с условием лабораторной не дали полное понимание как все работает, но с помощью проб, ошибок, немножко потраченных нервов и самостоятельного изучения темы я смогла сделать требуемое задание и даже получила удовольствие, но лишь после того как она заработала:) Полученный опыт считаю полезным для себя, так как это расширяет мои знания в программировании.