Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

**Лабораторная работа №3 по курсу**

**«Операционные системы»**

Группа: М80-206Б-22

Студент: Филатов А. К.

Преподаватель: Миронов Е.С.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 06.12.23

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 6.**

В файле записаны команды вида: «число число число». Дочерний процесс считает их сумму и выводит результат в стандартный поток вывода. Числа имеют тип int. Количество чисел может быть произвольным.

**Общий метод и алгоритм решения**

Использованные системные вызовы:

* pid\_t fork(void); – создает дочерний процесс, возвращает PID дочернего процесса, а процессу потомку возвращается 0, а в случае ошибки -1.
* int execl(const char \*\_\_path, char \*const \*\_\_argv, ...); - предоставляет новой программе список аргументов в виде массива указателей на строки, заканчивающиеся (char \*)0.
* pid\_t wait(int \*status); – приостаналивает выполнение текущего процесса до тех пор, пока дочерний процесс не завершится.
* void\* mmap(void \*, size\_t, int, int, int, off\_t) - выделяет память или отображает файлы (или устройства) в памяти.
* int munmap(void\*, size\_t) - удаляет сопоставление с выделенной памятью.
* int ftruncate(int, off\_t) - приводит файл к заданному размеру.
* int shm\_open(const char \*, int, ...) - инициализирует область памяти
* int shm\_unlink(const char\* name) — разрывает связь между областью памяти и заданным ей именем

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. А также в дочернем процессе инициализируется область памяти с определенным именем и там обрабатываются данные, находящиеся до этого в файле. Родительский процесс, после завершения работы дочернего, «подключается» к этой области памяти и считывает оттуда полученные результаты. Ну и выводит их на экран.

**Код программы**

**parent.c**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/mman.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <sys/stat.h>

#include "stddef.h"

#include <fcntl.h>

#include <string.h>

#include <stdbool.h>

#define MEMORY\_NAME "LR3FILATOV"

#define DATA\_SIZE 256    // Больше значений и не надо

#define MEMORY\_SIZE 8192    // С лихвой хватит памяти. Вы что там считать собрались?

void check\_error(bool expression, char\* message) {      // Покажи, чего там снова не получается

    if (expression) {

        write(STDOUT\_FILENO, message, strlen(message) \* sizeof(char));

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

        exit(-1);

    }

}

typedef struct {    // Нам и размер нужен, и массив под результат!

    size\_t size;

    int data[DATA\_SIZE];

} res;

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

    pid\_t pid;      // Создай-ка процесс

    FILE \*fp = NULL;

    if (argc != 2) {    // Ну ты чего, надо же написать что-то после вызова программы!

        write(1, "Wrong arguments\n", 17);

        exit(EXIT\_FAILURE);

    }

    pid = fork();       // Дочернего привести сюда не-ме-дле-нно!

    if (pid == -1) {    // Не привели?? Что случилось, кто посмеял?

        perror("fork");

        return -1;

    }

    else if (pid == 0) {    // Ну наконец-то. Что тут у нас?

        fp = freopen(argv[1], "r", stdin);  // Откройте букву "ФАЙЛ"

        check\_error(fp == NULL, "Can't open file"); // Нет такой буквы!

        execl("./child", "/.child", NULL);  // Запустить другую программу из текущей

        perror("execl");

        return 1;

    }

    else {  // Родитель? К тебе есть пару вопросов...

        wait(0);    // Ну погоди, погоди, щас отработает Детишко

        int fd = shm\_open(MEMORY\_NAME, O\_RDONLY, S\_IRUSR | S\_IWUSR);    // Разделяем память!!!

        check\_error(fd == -1, "Can't open shared memory file"); // Что-то пошло не так???

        res \*addr = mmap(NULL, MEMORY\_SIZE, PROT\_READ, MAP\_SHARED, fd, 0);  // А теперь работаем с файликом, как с памятью!!!

        check\_error(addr == (void\*) -1, "Mmap error");  // НУ СКОЛЬКО МОЖНО!

        for (int i = 0; i < addr->size; i++) {  // Покажи, чего там получилось у Детишко?

            printf("Сумма цифр в %d строке: %d\n", i + 1, addr->data[i]);

        }

        printf("Рассчет окончен!\n");

        munmap(addr, MEMORY\_SIZE);  // Все, молодец, закрывай всё подряд

        shm\_unlink(MEMORY\_NAME);

        close(fd);

    }

    return 0;

}

**child.c**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/mman.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <sys/stat.h>

#include "stddef.h"

#include <fcntl.h>

#include <string.h>

#include <stdbool.h>

#include <math.h>

#define MEMORY\_NAME "LR3FILATOV"

#define DATA\_SIZE 256   // Больше значений и не надо

#define MEMORY\_SIZE 8192    // С лихвой хватит памяти. Вы что там считать собрались?

void check\_error(bool expression, char\* message) {  // Покажи, чего там снова не получается

    if (expression) {

        write(STDOUT\_FILENO, message, strlen(message) \* sizeof(char));

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

        exit(-1);

    }

}

typedef struct {    // Нам и размер нужен, и массив под результат!

    size\_t size;

    int data[DATA\_SIZE];

} res;

int main() {

    int fd = shm\_open(MEMORY\_NAME, O\_EXCL | O\_CREAT | O\_RDWR, S\_IRUSR | S\_IWUSR);   // Разделяем память (подключаемся к нашему любимчику родителю)

    check\_error(fd == -1, "Can't open shared memory file"); // Что-то пошло не так???

    if (ftruncate(fd, MEMORY\_SIZE) == -1) {    // ТЫ ЧЕГО СЧИТАТЬ СОБРАЛСЯ? БУКВЫ В "ВОЙНЕ И МИРЕ"?!

        printf("File is too large");

    }

    res \*addr = mmap(NULL, MEMORY\_SIZE, PROT\_WRITE, MAP\_SHARED, fd, 0);    // А теперь работаем с файликом, как с памятью!!!

    check\_error(addr == (void\*)-1, "Mmap error");   // НУ СКОЛЬКО МОЖНО!

    addr->size = 0;    // Ну пока нету там ничего, падажжи

    char c;

    bool not\_end = true;

    int nmbr = 0;

    int result = 0;

    int count = 0;

    int numbers[100];

    do {

        if (not\_end) {

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

                nmbr = nmbr \* 10 + c - '0';

            }

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

                numbers[count] = nmbr;

                nmbr = 0;

                count++;

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

                    for (int i = 0; i < count; i++) {

                        result = result + numbers[i];

                    }

                    not\_end = false;

                    count = 0;

                }

            }

        }

        if (c == '\n' || c == EOF) {

            addr->data[addr->size++] = result;

            result = 0;

            not\_end = true;

        }

    } while((scanf("%c", &c)) > 0);

    return 0;

}

**Протокол работы программы**

**Тестирование:**

cblphblu@DESKTOP-3PDEL6G:/mnt/c/Users/user/Desktop/МАИ/2 курс/ОСИ/LR3FILATOV$ ./parent file.txt

Сумма цифр в 1 строке: 9

Сумма цифр в 2 строке: 16

Сумма цифр в 3 строке: 15

Сумма цифр в 4 строке: 17

Сумма цифр в 5 строке: 19

Сумма цифр в 6 строке: 0

Сумма цифр в 7 строке: 17

Сумма цифр в 8 строке: 55

Рассчет окончен!

**Strace**

**cblphblu@DESKTOP-3PDEL6G:/mnt/c/Users/user/Desktop/МАИ/2 курс/ОСИ/LR3FILATOV$ strace -f ./parent file.txt**

execve("./parent", ["./parent", "file.txt"], 0x7ffdcf3dbdc0 /\* 20 vars \*/) = 0

brk(NULL) = 0x55a646c6c000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffcc87154d0) = -1 EINVAL (Недопустимый аргумент)

**mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7faf03919000**

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=34303, ...}, AT\_EMPTY\_PATH) = 0

**mmap(NULL, 34303, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7faf03910000**

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\0i8\235HZ\227\223\333\350s\360\352,\223\340."..., 68, 896) = 68

newfstatat(3, "", {st\_mode=S\_IFREG|0644, 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) = 0x7faf036e8000**

**mmap(0x7faf03710000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7faf03710000**

**mmap(0x7faf038a5000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7faf038a5000**

**mmap(0x7faf038fd000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x214000) = 0x7faf038fd000**

**mmap(0x7faf03903000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7faf03903000**

close(3) = 0

**mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7faf036e5000**

arch\_prctl(ARCH\_SET\_FS, 0x7faf036e5740) = 0

set\_tid\_address(0x7faf036e5a10) = 97

set\_robust\_list(0x7faf036e5a20, 24) = 0

rseq(0x7faf036e60e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7faf038fd000, 16384, PROT\_READ) = 0

mprotect(0x55a645235000, 4096, PROT\_READ) = 0

mprotect(0x7faf03953000, 8192, PROT\_READ) = 0

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

**munmap(0x7faf03910000, 34303) = 0**

**clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 98 attached**

**, child\_tidptr=0x7faf036e5a10) = 98**

[pid 98] set\_robust\_list(0x7faf036e5a20, 24 <unfinished ...>

[pid 97] wait4(-1, <unfinished ...>

[pid 98] <... set\_robust\_list resumed>) = 0

[pid 98] openat(AT\_FDCWD, "file.txt", O\_RDONLY) = 3

[pid 98] dup3(3, 0, 0) = 0

[pid 98] close(3) = 0

[pid 98] execve("./child", ["/.child"], 0x7ffcc87156b0 /\* 20 vars \*/) = 0

[pid 98] brk(NULL) = 0x555de0f07000

[pid 98] arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffd5eee49d0) = -1 EINVAL (Недопустимый аргумент)

[pid 98**] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff578bc1000**

[pid 98] access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (Нет такого файла или каталога)

[pid 98] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

[pid 98] newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=34303, ...}, AT\_EMPTY\_PATH) = 0

[pid 98] **mmap(NULL, 34303, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7ff578bb8000**

[pid 98] close(3) = 0

[pid 98] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

[pid 98] 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 98] 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 98] 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 98] pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0i8\235HZ\227\223\333\350s\360\352,\223\340."..., 68, 896) = 68

[pid 98] newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=2216304, ...}, AT\_EMPTY\_PATH) = 0

[pid 98] 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 98] **mmap(NULL, 2260560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7ff578990000**

**[pid 98] mmap(0x7ff5789b8000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7ff5789b8000**

**[pid 98] mmap(0x7ff578b4d000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7ff578b4d000**

**[pid 98] mmap(0x7ff578ba5000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x214000) = 0x7ff578ba5000**

**[pid 98] mmap(0x7ff578bab000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7ff578bab000**

[pid 98] close(3) = 0

[pid 98] **mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff57898d000**

[pid 98] arch\_prctl(ARCH\_SET\_FS, 0x7ff57898d740) = 0

[pid 98] set\_tid\_address(0x7ff57898da10) = 98

[pid 98] set\_robust\_list(0x7ff57898da20, 24) = 0

[pid 98] rseq(0x7ff57898e0e0, 0x20, 0, 0x53053053) = 0

[pid 98] mprotect(0x7ff578ba5000, 16384, PROT\_READ) = 0

[pid 98] mprotect(0x555de04ce000, 4096, PROT\_READ) = 0

[pid 98] mprotect(0x7ff578bfb000, 8192, PROT\_READ) = 0

[pid 98] prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

[pid 98] **munmap(0x7ff578bb8000, 34303) = 0**

[pid 98] openat(AT\_FDCWD, "/dev/shm/LR3FILATOV", O\_RDWR|O\_CREAT|O\_EXCL|O\_NOFOLLOW|O\_CLOEXEC, 0600) = 3

[pid 98] ftruncate(3, 8192) = 0

[pid 98] **mmap(NULL, 8192, PROT\_WRITE, MAP\_SHARED, 3, 0) = 0x7ff578bbf000**

[pid 98] newfstatat(0, "", {st\_mode=S\_IFREG|0777, st\_size=69, ...}, AT\_EMPTY\_PATH) = 0

[pid 98] getrandom("\x5c\x20\xbe\x60\x0c\xaa\xb2\xff", 8, GRND\_NONBLOCK) = 8

[pid 98] brk(NULL) = 0x555de0f07000

[pid 98] brk(0x555de0f28000) = 0x555de0f28000

[pid 98] read(0, "9\r\n10 2 4\r\n0 3 12\r\n13 1 3\r\n12 7\r"..., 4096) = 69

[pid 98] read(0, "", 4096) = 0

[pid 98] exit\_group(0) = ?

[pid 98] +++ exited with 0 +++

<... wait4 resumed>NULL, 0, NULL) = 98

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=98, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

openat(AT\_FDCWD, "/dev/shm/LR3FILATOV", O\_RDONLY|O\_NOFOLLOW|O\_CLOEXEC) = 3

**mmap(NULL, 8192, PROT\_READ, MAP\_SHARED, 3, 0) = 0x7faf03917000**

newfstatat(1, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}, AT\_EMPTY\_PATH) = 0

getrandom("\x49\x8f\x61\x6b\xed\xd6\xd7\x79", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x55a646c6c000

brk(0x55a646c8d000) = 0x55a646c8d000

write(1, "\320\241\321\203\320\274\320\274\320\260 \321\206\320\270\321\204\321\200 \320\262 1 \321\201\321\202\321\200\320"..., 41Сумма цифр в 1 строке: 9

) = 41

write(1, "\320\241\321\203\320\274\320\274\320\260 \321\206\320\270\321\204\321\200 \320\262 2 \321\201\321\202\321\200\320"..., 42Сумма цифр в 2 строке: 16

) = 42

write(1, "\320\241\321\203\320\274\320\274\320\260 \321\206\320\270\321\204\321\200 \320\262 3 \321\201\321\202\321\200\320"..., 42Сумма цифр в 3 строке: 15

) = 42

write(1, "\320\241\321\203\320\274\320\274\320\260 \321\206\320\270\321\204\321\200 \320\262 4 \321\201\321\202\321\200\320"..., 42Сумма цифр в 4 строке: 17

) = 42

write(1, "\320\241\321\203\320\274\320\274\320\260 \321\206\320\270\321\204\321\200 \320\262 5 \321\201\321\202\321\200\320"..., 42Сумма цифр в 5 строке: 19

) = 42

write(1, "\320\241\321\203\320\274\320\274\320\260 \321\206\320\270\321\204\321\200 \320\262 6 \321\201\321\202\321\200\320"..., 41Сумма цифр в 6 строке: 0

) = 41

write(1, "\320\241\321\203\320\274\320\274\320\260 \321\206\320\270\321\204\321\200 \320\262 7 \321\201\321\202\321\200\320"..., 42Сумма цифр в 7 строке: 17

) = 42

write(1, "\320\241\321\203\320\274\320\274\320\260 \321\206\320\270\321\204\321\200 \320\262 8 \321\201\321\202\321\200\320"..., 42Сумма цифр в 8 строке: 55

) = 42

**munmap(0x7faf03917000, 8192) = 0**

unlink("/dev/shm/LR3FILATOV") = 0

close(3) = 0

write(1, "\320\240\320\260\321\201\321\201\321\207\320\265\321\202 \320\276\320\272\320\276\320\275\321\207\320\265\320\275!", 30Рассчет окончен!) = 30

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

Благодаря выполнению данной работы я изучил новый принцип работы с межпроцессорным взаимодействием. Я понял, как можно «связывать» между собой процессы, чтобы они работали в одной области памяти. Во время выполнения лабораторной основная трудность была следующая: сначала мне было трудно разобраться с системными вызовами, так как они не самые популярные и на сайтах встречались еще более сложные слова и определения, однако я все же понял, что за что отвечает. Единственное, мне очень повезло с вариантом задания, сложить числа в строке я могу. В целом, я подчеркнул много нового для себя, что поможет мне в написании будущих более сложных кодов.