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: М8О-211Б-23

Студент: Соболин Т.С.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 13.01.25

Москва, 2025

Постановка задачи

Вариант 2.

Пользователь вводит команды вида: «число число число<endline>». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс считает их сумму и

выводит её в файл. Числа имеют тип float. Количество чисел может быть произвольным.

Общий метод и алгоритм решения

Использованные системные вызовы:

- shmget - используется для создания или получения доступа к сегменту разделяемой памяти.
- fork - порождает дочерний процесс.
- shmat - присоединяет сегмент разделяемой памяти к адресному пространству процесса.
- strlen - вычисляет длину строки.
- strcpy - копирует строку.
- atof - преобразует строку в число с плавающей точкой.
- fopen - открывает файл.
- fprintf - записывает форматированные данные в файл.
- fclose - закрывает файл.
- memset - заполняет указанную область памяти заданным значением.
- shmdt - отсоединяет сегмент разделяемой памяти от адресного пространства процесса.
- shmctl - используется для управления сегментами разделяемой памяти.

* wait - ожидает завершения дочернего процесса.

Программа создает сегмент разделяемой памяти, затем порождает дочерний процесс, который прикрепляет этот сегмент и ждет данные от родителя. Родительский процесс получает данные от пользователя и записывает их в разделяемую память. Дочерний процесс суммирует числа и сохраняет результат в файл. Обе программы завершают работу по команде "exit".

Код программы

parent.c

```
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <stdio.h>

#define SHM_SIZE 1024

void print_error(const char *msg) {
    write(2, msg, strlen(msg));
}

void write_to_file(const char *filename, const char *data) {
    int file = open(filename, O_WRONLY | O_APPEND | O_CREAT, 0666);
    if (file != -1) {
```

```
            write(file, data, strlen(data));
            close(file);
        } else {
            print_error("Ошибка при открытии файла\n");
        }
    }

    int main(int argc, char *argv[]) {
        if (argc != 2) {
            print_error("Использование: <имя_файла>\n");
            exit(EXIT_FAILURE);
        }

        char *filename = argv[1];
        int shmid;
        char *shm_ptr;
        key_t key = ftok("/tmp", 'R');

        shmid = shmget(key, SHM_SIZE, IPC_CREAT | 0666);
        if (shmid < 0) {
            print_error("Ошибка при создании сегмента общей памяти\n");
            exit(EXIT_FAILURE);
        }

        pid_t pid = fork();
        if (pid < 0) {
            print_error("Ошибка при вызове fork\n");
            exit(EXIT_FAILURE);
        }

        if (pid == 0) {
            execl("./child", "./child", filename, NULL);
            print_error("Ошибка при запуске дочернего процесса\n");
            exit(EXIT_FAILURE);
        } else {
            shm_ptr = shmat(shmid, NULL, 0);
            if (shm_ptr == (char *)(-1)) {
                print_error("Ошибка при подключении к сегменту общей памяти\n");
                exit(EXIT_FAILURE);
            }

            char input[SHM_SIZE];
            while (1) {
                const char *prompt = "Введите числа (или 'exit' для выхода): ";
                write(1, prompt, strlen(prompt));
                read(0, input, SHM_SIZE);
                input[strcspn(input, "\n")] = 0;

                strncpy(shm_ptr, input, SHM_SIZE);
                if (strcmp(input, "exit") == 0) {
                    break;
                }
                sleep(1);
            }

            shmdt(shm_ptr);
            shmctl(shmid, IPC_RMID, NULL);
            wait(NULL);
```

```
            exit(EXIT_SUCCESS);
        }
    }
```

Child.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <fcntl.h>

#define SHM_SIZE 1024

void write_to_file(const char *filename, const char *data) {
    int file = open(filename, O_WRONLY | O_APPEND | O_CREAT, 0666);
    if (file != -1) {
        write(file, data, strlen(data));
        close(file);
    } else {
        perror("Ошибка при открытии файла");
    }
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        perror("Использование: <имя_файла>");
        exit(EXIT_FAILURE);
    }

    char *filename = argv[1];
    key_t key = ftok("/tmp", 'R');
    int shmid = shmget(key, SHM_SIZE, 0666);
    if (shmid < 0) {
```

```
        perror("Ошибка при подключении к сегменту общей памяти");
        exit(EXIT_FAILURE);
    }

    char *shm_ptr = shmat(shmid, NULL, 0);
    if (shm_ptr == (char *)(-1)) {
        perror("Ошибка при подключении к общей памяти");
        exit(EXIT_FAILURE);
    }

    while (1) {
        if (strlen(shm_ptr) > 0) {
            if (strcmp(shm_ptr, "exit") == 0) {
                break;
            }

            float sum = 0.0;
            char *token = strtok(shm_ptr, " ");
            while (token != NULL) {
                sum += atof(token);
                token = strtok(NULL, " ");
            }

            char output[50];
            snprintf(output, sizeof(output), "Сумма: %.2f\n", sum);
            write_to_file(filename, output);
            memset(shm_ptr, 0, SHM_SIZE);
        }
        sleep(1);
    }

    shmdt(shm_ptr);
    exit(EXIT_SUCCESS);
}
```

Протокол работы программы

Тестирование:

kotlasboy@kotlasboy-Modern-15-B12M:~/Programming/Projects/OS/lab_3$ strace -f ./a.out output.txt

execve("./a.out", ["./a.out", "output.txt"], 0x7fff7b41c2b0 /* 59 vars */) = 0

brk(NULL) = 0x56b4fc64e000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x73a1eb98a000

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=68847, ...}) = 0

mmap(NULL, 68847, PROT_READ, MAP_PRIVATE, 3, 0) = 0x73a1eb979000

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) = 0x73a1eb600000

mmap(0x73a1eb628000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x73a1eb628000

mmap(0x73a1eb7b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x73a1eb7b0000

mmap(0x73a1eb7ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x73a1eb7ff000

mmap(0x73a1eb805000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x73a1eb805000

close(3) = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x73a1eb976000

arch_prctl(ARCH_SET_FS, 0x73a1eb976740) = 0

set_tid_address(0x73a1eb976a10) = 35011

set_robust_list(0x73a1eb976a20, 24)   = 0

rseq(0x73a1eb977060, 0x20, 0, 0x53053053) = 0

mprotect(0x73a1eb7ff000, 16384, PROT_READ) = 0

mprotect(0x56b4de650000, 4096, PROT_READ) = 0

mprotect(0x73a1eb9c2000, 8192, PROT_READ) = 0

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

munmap(0x73a1eb979000, 68847)   = 0

newfstatat(AT_FDCWD, "/tmp", {st_mode=S_IFDIR|S_ISVTX|0777, st_size=4096, ...}, 0) = 0

shmget(0x52050001, 1024, IPC_CREAT|0666) = 98327

clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x73a1eb976a10) = 35012

strace: Process 35012 attached

[pid 35011] shmat(98327, NULL, 0)   = 0x73a1eb989000

[pid 35011] write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\321\207\320\270\321\201\320\273\320\260 (\320\270\320\273\320"..., 63Введите числа (или 'exit'
для выхода):) = 63

[pid 35012] set_robust_list(0x73a1eb976a20, 24 <unfinished ...>

[pid 35011] read(0, <unfinished ...>

[pid 35012] <... set_robust_list resumed>) = 0

[pid 35012] execve("./child", ["./child", "output.txt"], 0x7ffedc25b480 /* 59 vars */) = 0

[pid 35012] brk(NULL)   = 0x5ef2ae706000

[pid 35012] mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x735d1d31f000

[pid 35012] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

[pid 35012] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

[pid 35012] fstat(3, {st_mode=S_IFREG|0644, st_size=68847, ...}) = 0

[pid 35012] mmap(NULL, 68847, PROT_READ, MAP_PRIVATE, 3, 0) = 0x735d1d30e000

[pid 35012] close(3)   = 0

[pid 35012] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) =
3

[pid 35012] 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 35012] 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 35012] fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0

[pid 35012] 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 35012] mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x735d1d000000

[pid 35012] mmap(0x735d1d028000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x735d1d028000

[pid 35012] mmap(0x735d1d1b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x735d1d1b0000

[pid 35012] mmap(0x735d1d1ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x735d1d1ff000

[pid 35012] mmap(0x735d1d205000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x735d1d205000

[pid 35012] close(3)   = 0

[pid 35012] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x735d1d30b000

[pid 35012] arch_prctl(ARCH_SET_FS, 0x735d1d30b740) = 0

[pid 35012] set_tid_address(0x735d1d30ba10) = 35012

[pid 35012] set_robust_list(0x735d1d30ba20, 24) = 0

[pid 35012] rseq(0x735d1d30c060, 0x20, 0, 0x53053053) = 0

[pid 35012] mprotect(0x735d1d1ff000, 16384, PROT_READ) = 0

[pid 35012] mprotect(0x5ef29bb31000, 4096, PROT_READ) = 0

[pid 35012] mprotect(0x735d1d357000, 8192, PROT_READ) = 0

[pid 35012] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

[pid 35012] munmap(0x735d1d30e000, 68847) = 0

[pid 35012] newfstatat(AT_FDCWD, "/tmp", {st_mode=S_IFDIR|S_ISVTX|0777, st_size=4096, ...}, 0) = 0

[pid 35012] shmget(0x52050001, 1024, 0666) = 98327

[pid 35012] shmat(98327, NULL, 0)   = 0x735d1d31e000

[pid 35012] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, 0x7ffd97ee7580) = 0

[pid 35012] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, 0x7ffd97ee7580) = 0

[pid 35012] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, 100x7ffd97ee7580) = 0

[pid 35012] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, .2 10.20x7ffd97ee7580) = 0

[pid 35012] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, 0x7ffd97ee7580) = 0

[pid 35012] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, 10.20x7ffd97ee7580) = 0

[pid 35012] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},

<unfinished ...>

[pid 35011] <... read resumed>"10.2 10.210.2\n", 1024) = 14

[pid 35011] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, <unfinished ...>

[pid 35012] <... clock_nanosleep resumed>0x7ffd97ee7580) = 0

[pid 35012] openat(AT_FDCWD, "output.txt", O_WRONLY|O_CREAT|O_APPEND, 0666) = 3

[pid 35012] write(3, "\320\241\321\203\320\274\320\274\320\260: 20.41\n", 18) = 18

[pid 35012] close(3) = 0

[pid 35012] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, <unfinished ...>

[pid 35011] <... clock_nanosleep resumed>0x7ffedc25aeb0) = 0

[pid 35011] write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \321\207\320\270\321\201\320\273\320\260 (\320\270\320\273\320"..., 63Введите числа (или 'exit' для выхода):) = 63

[pid 35011] read(0, <unfinished ...>

[pid 35012] <... clock_nanosleep resumed>0x7ffd97ee7580) = 0

[pid 35012] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, 100x7ffd97ee7580) = 0

[pid 35012] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, .2

<unfinished ...>

[pid 35011] <... read resumed>"10.2\n", 1024) = 5

[pid 35011] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, <unfinished ...>

[pid 35012] <... clock_nanosleep resumed>0x7ffd97ee7580) = 0

[pid 35012] openat(AT_FDCWD, "output.txt", O_WRONLY|O_CREAT|O_APPEND, 0666) = 3

[pid 35012] write(3, "\320\241\321\203\320\274\320\274\320\260: 10.20\n", 18) = 18

[pid 35012] close(3)   = 0

[pid 35012] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, <unfinished ...>

[pid 35011] <... clock_nanosleep resumed>0x7ffedc25aeb0) = 0

[pid 35011] write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \321\207\320\270\321\201\320\273\320\260 (\320\270\320\273\320"..., 63Введите числа (или 'exit' для выхода):) = 63

[pid 35011] read(0, <unfinished ...>

[pid 35012] <... clock_nanosleep resumed>0x7ffd97ee7580) = 0

[pid 35012] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, e0x7ffd97ee7580) = 0

[pid 35012] clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, xit

<unfinished ...>

[pid 35011] <... read resumed>"exit\n", 1024) = 5

[pid 35011] shmdt(0x73a1eb989000)   = 0

[pid 35011] shmctl(98327, IPC_RMID, NULL) = 0

[pid 35011] wait4(-1, <unfinished ...>

[pid 35012] <... clock_nanosleep resumed>0x7ffd97ee7580) = 0

[pid 35012] shmdt(0x735d1d31e000)   = 0

[pid 35012] exit_group(0)   = ?

[pid 35012] +++ exited with 0 +++

<... wait4 resumed>NULL, 0, NULL)   = 35012

--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=35012, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---

exit_group(0)   = ?

+++ exited with 0 +++

Вывод

Эта программа демонстрирует использование механизма разделяемой памяти для коммуникации между родительским и дочерним процессами в Unix-подобных системах. Она

позволяет эффективно обмениваться данными и выполнять их обработку в реальном времени, предоставляя простой способ взаимодействия процессов.