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-209БВ-24

Студент: Касаева Я.М.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 08.10.25

Москва, 2025

Постановка задачи

Вариант 5.

Родительский процесс создает дочерний процесс. Пользователь вводит имя файла, которое передается дочернему процессу. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или через отображаемые файлы (memory-mapped files). Пользователь вводит команды вида: «число<endline>». Дочерний процесс производит проверку на простоту. Если число составное, то это число записывается в файл. Если число отрицательное или простое, то тогда дочерний и родительский процессы завершаются.

Общий метод и алгоритм решения

Использованные системные вызовы:

pid_t fork(void) - создает дочерний процесс путем копирования текущего процесса

int execl(const char *path, const char *arg, ...) - заменяет образ текущего процесса новым исполняемым файлом

pid_t waitpid(pid_t pid, int *status, int options) - ожидает завершения указанного дочернего процесса

int wait(int *status) - ожидает завершения любого дочернего процесса

int shm_open(const char *name, int oflag, mode_t mode) - создает или открывает объект разделяемой памяти

void *mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset) - отображает файл или разделяемую память в адресное пространство процесса

int munmap(void *addr, size_t length) - удаляет отображение памяти

int ftruncate(int fd, off_t length) - устанавливает размер файла разделяемой памяти

int shm_unlink(const char *name) - удаляет объект разделяемой памяти

sem_t *sem_open(const char *name, int oflag, ...) - открывает или создает именованный семафор

int sem_wait(sem_t *sem) - уменьшает значение семафора на 1 (блокируется если значение 0)

int sem_post(sem_t *sem) - увеличивает значение семафора на 1

int sem_close(sem_t *sem) - закрывает семафор

int sem_unlink(const char *name) - удаляет именованный семафор из системы

FILE *fopen(const char *filename, const char *mode) - открывает файл для записи результатов

int ftruncate(int fd, off_t length) - устанавливает размер файла

void *memset(void *s, int c, size_t n) - заполняет область памяти указанным значением

size_t strlen(const char *s) - возвращает длину строки

Алгоритм работы программы:

Родительский процесс:

Создает семафоры sem_parent и sem_child с начальным значением 0

Создает разделяемую память /input_shm и /output_shm

Получает имя файла от пользователя

Создает дочерний процесс с помощью fork()

Принимает число от пользователя

Записывает число в input_shm

Сигналит ребенку через sem_post(sem_parent)

Ждет ответ через sem_wait(sem_child)

Читает ответ из output_shm

Если "EXIT" - завершает работу

Очищает ресурсы

Дочерний процесс:

Открывает семафоры и разделяемую память

Открывает файл для записи

Ждет сигнал от родителя через sem_wait(sem_parent)

Читает число из input_shm

Проверяет число

Сигналит родителю через sem_post(sem_child)

Код программы

parent.c

```c
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <ctype.h>
#include <limits.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <semaphore.h>
```

```c
int is_integer(const char *str) {
    if (str == NULL || *str == '\0') return 0;
    int i = 0;
    if (str[0] == '-') {
        if (str[1] == '\0') return 0;
        i = 1;
    }
    for (; str[i] != '\0'; i++) {
        if (!isdigit(str[i])) return 0;
    }
    long num = atol(str);
    if (num < -2147483648 || num > 2147483647) {
        return 0;
    }
    return 1;
}

int main() {
    char filename[100];
    char number[100];
    pid_t pid;

    sem_unlink("/sem_parent");
    sem_unlink("/sem_child");
    sem_t *sem_parent = sem_open("/sem_parent", O_CREAT, 0666, 0);
    sem_t *sem_child = sem_open("/sem_child", O_CREAT, 0666, 0);

    shm_unlink("/input_shm");
    shm_unlink("/output_shm");

    int input_fd = shm_open("/input_shm", O_CREAT | O_RDWR, 0666);
    int output_fd = shm_open("/output_shm", O_CREAT | O_RDWR, 0666);

    ftruncate(input_fd, 100);
    ftruncate(output_fd, 100);

    char *input_data = mmap(0, 100, PROT_READ | PROT_WRITE, MAP_SHARED, input_fd, 0);
    char *output_data = mmap(0, 100, PROT_READ | PROT_WRITE, MAP_SHARED, output_fd, 0);

    memset(input_data, 0, 100);
    memset(output_data, 0, 100);

    printf("Введите имя файла: ");
    scanf("%s", filename);

    pid = fork();
    if (pid == 0) {
        close(input_fd);
        close(output_fd);
        sem_close(sem_parent);
        sem_close(sem_child);
        execl("./child", "child", filename, NULL);
        exit(1);
    } else {
        sleep(1);

        while (1) {
            printf("Введите число: ");
            scanf("%s", number);

            if (!is_integer(number)) {
                printf("Введенные данные не типа int, попробуйте еще раз:\n");
                continue;
            }

            if (waitpid(pid, NULL, WNOHANG) == pid) {
```

```
                printf("Дочерний процесс завершен\n");
                break;
            }

            strcpy(input_data, number);
            sem_post(sem_parent);

            sem_wait(sem_child);

            char response[10];
            strcpy(response, output_data);

            if (strstr(response, "EXIT")) {
                printf("Получен сигнал завершения\n");
                break;
            }

            memset(output_data, 0, 100);
        }

        wait(NULL);

        munmap(input_data, 100);
        munmap(output_data, 100);
        close(input_fd);
        close(output_fd);
        sem_close(sem_parent);
        sem_close(sem_child);
        sem_unlink("/sem_parent");
        sem_unlink("/sem_child");
        shm_unlink("/input_shm");
        shm_unlink("/output_shm");

        printf("Родительский процесс завершен\n");
    }
    return 0;
}
```

child.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <semaphore.h>

int is_prime(int n) {
    if (n < 1) return 0;
    if (n == 1) return 1;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) return 0;
    }
    return 1;
}

int main(int argc, char *argv[]) {
    if (argc < 2) {
        return 1;
    }

    FILE *file = fopen(argv[1], "w");
    if (file == NULL) {
        return 1;
    }
```

```
    sem_t *sem_parent = sem_open("/sem_parent", 0);
    sem_t *sem_child = sem_open("/sem_child", 0);

    int input_fd = shm_open("/input_shm", O_RDWR, 0666);
    int output_fd = shm_open("/output_shm", O_RDWR, 0666);

    if (input_fd == -1 || output_fd == -1) {
        return 1;
    }

    char *input_data = mmap(0, 100, PROT_READ | PROT_WRITE, MAP_SHARED, input_fd, 0);
    char *output_data = mmap(0, 100, PROT_READ | PROT_WRITE, MAP_SHARED, output_fd, 0);

    while (1) {
        sem_wait(sem_parent);

        if (strlen(input_data) > 0) {
            char line[100];
            strcpy(line, input_data);
            memset(input_data, 0, 100);

            int num = atoi(line);

            if (num < 0 || is_prime(num)) {
                fprintf(file, "EXIT: %d\n", num);
                fclose(file);
                strcpy(output_data, "EXIT");
                sem_post(sem_child);
                break;
            } else {
                fprintf(file, "%d\n", num);
                fflush(file);
                strcpy(output_data, "OK");
                sem_post(sem_child);
            }
        }
    }

    munmap(input_data, 100);
    munmap(output_data, 100);
    close(input_fd);
    close(output_fd);
    sem_close(sem_parent);
    sem_close(sem_child);

    return 0;
}
```

Протокол работы программы

Тестирование:

```
(base) yanakasaeva@MacBook-Air--YanaK src % gcc -o parent parent.c

gcc -o child child.c

./parent

Введите имя файла: res.txt

Введите число: 35

Введите число: 9
```

```
Введите число: 0

Введите число: -20

Получен сигнал завершения

Родительский процесс завершен

(base) yanakasaeva@MacBook-Air--YanaK src % cat res.txt

35

9

0

EXIT: -20
```

Strace:

1102 execve("./parent", ["./parent"], 0xffffdb09d888 /* 8 vars */) = 0

1102 brk(NULL) = 0xaaaae34fd000

1102 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffff9cb18000

1102 faccessat(AT_FDCWD, "/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

1102 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

1102 fstat(3, {st_mode=S_IFREG|0644, st_size=8467, ...}) = 0

1102 mmap(NULL, 8467, PROT_READ, MAP_PRIVATE, 3, 0) = 0xffff9cb15000

1102 close(3) = 0

1102 openat(AT_FDCWD, "/lib/aarch64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

1102 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0\360\206\2\0\0\0\0\0"..., 832) = 832

1102 fstat(3, {st_mode=S_IFREG|0755, st_size=1722920, ...}) = 0

1102 mmap(NULL, 1892240, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_DENYWRITE, -1, 0) = 0xffff9c911000

1102 mmap(0xffff9c920000, 1826704, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffff9c920000

1102 munmap(0xffff9c911000, 61440) = 0

1102 munmap(0xffff9cade000, 3984) = 0

1102 mprotect(0xffff9cab9000, 81920, PROT_NONE) = 0

1102 mmap(0xffff9cacd000, 20480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) = 0xffff9cacd000

1102 mmap(0xffff9cad2000, 49040, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xffff9cad2000

1102 close(3) = 0

1102 set_tid_address(0xffff9cb18fb0) = 1102

1102 set_robust_list(0xffff9cb18fc0, 24) = 0

1102 rseq(0xffff9cb19600, 0x20, 0, 0xd428bc00) = 0

1102 mprotect(0xffff9cacd000, 12288, PROT_READ) = 0

1102 mprotect(0xaaaab502f000, 4096, PROT_READ) = 0

1102 mprotect(0xffff9cb1d000, 8192, PROT_READ) = 0

1102 prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

1102 munmap(0xffff9cb15000, 8467) = 0

1102 unlinkat(AT_FDCWD, "/dev/shm/sem.sem_parent", 0) = -1 ENOENT (No such file or directory)

1102 unlinkat(AT_FDCWD, "/dev/shm/sem.sem_child", 0) = -1 ENOENT (No such file or directory)

1102 openat(AT_FDCWD, "/dev/shm/sem.sem_parent", O_RDWR|O_NOFOLLOW|O_CLOEXEC) = -1 ENOENT (No such file or directory)

1102 getrandom("\x23\x62\xfe\x6a\x30\x9e\x4a\x8c", 8, GRND_NONBLOCK) = 8

1102 newfstatat(AT_FDCWD, "/dev/shm/sem.DnG8on", 0xffffec86c2b8, AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)

1102 openat(AT_FDCWD, "/dev/shm/sem.DnG8on", O_RDWR|O_CREAT|O_EXCL|O_NOFOLLOW|O_CLOEXEC, 0666) = 3

1102 write(3, "\0\0\0\0\0\0\0\200\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32) = 32

1102 mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0xffff9cb17000

1102 linkat(AT_FDCWD, "/dev/shm/sem.DnG8on", AT_FDCWD, "/dev/shm/sem.sem_parent", 0) = 0

1102 fstat(3, {st_mode=S_IFREG|0644, st_size=32, ...}) = 0

1102 getrandom("\x9b\xe0\x2b\x37\x09\xb1\xf2\xc7", 8, GRND_NONBLOCK) = 8

1102 brk(NULL) = 0xaaaae34fd000

1102 brk(0xaaaae351e000) = 0xaaaae351e000

1102 unlinkat(AT_FDCWD, "/dev/shm/sem.DnG8on", 0) = 0

1102 close(3) = 0

1102 openat(AT_FDCWD, "/dev/shm/sem.sem_child", O_RDWR|O_NOFOLLOW|O_CLOEXEC) = -1 ENOENT (No such file or directory)

1102 getrandom("\x3c\x29\x44\x86\x52\xea\xc1\x33", 8, GRND_NONBLOCK) = 8

1102 newfstatat(AT_FDCWD, "/dev/shm/sem.O6mjGh", 0xffffec86c2b8, AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)

1102 openat(AT_FDCWD, "/dev/shm/sem.O6mjGh", O_RDWR|O_CREAT|O_EXCL|O_NOFOLLOW|O_CLOEXEC, 0666) = 3

1102 write(3, "\0\0\0\0\0\0\0\200\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32) = 32

1102 mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0xffff9cb16000

1102 linkat(AT_FDCWD, "/dev/shm/sem.O6mjGh", AT_FDCWD, "/dev/shm/sem.sem_child", 0) = 0

1102 fstat(3, {st_mode=S_IFREG|0644, st_size=32, ...}) = 0

1102 unlinkat(AT_FDCWD, "/dev/shm/sem.O6mjGh", 0) = 0

1102 close(3) = 0

1102 unlinkat(AT_FDCWD, "/dev/shm/input_shm", 0) = -1 ENOENT (No such file or directory)

1102 unlinkat(AT_FDCWD, "/dev/shm/output_shm", 0) = -1 ENOENT (No such file or directory)

1102 openat(AT_FDCWD, "/dev/shm/input_shm", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0666) = 3

1102 openat(AT_FDCWD, "/dev/shm/output_shm", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0666) = 4

1102 ftruncate(3, 100) = 0

1102 ftruncate(4, 100) = 0

1102 mmap(NULL, 100, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0xffff9cb15000

1102 mmap(NULL, 100, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0xffff9cb14000

1102 fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0

1102 fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0

1102 write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\270\320\274\321\217 \321\204\320\260\320\271\320\273\320\260"..., 34) = 34

1102 read(0, "res.txt\n", 1024) = 8

1102 clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0xffff9cb18fb0) = 1103

1102 clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0}, <unfinished ...>

1103 set_robust_list(0xffff9cb18fc0, 24) = 0

1103 close(3) = 0

1103 close(4) = 0

1103 munmap(0xffff9cb17000, 32) = 0

1103 munmap(0xffff9cb16000, 32) = 0

1103 execve("./child", ["child", "res.txt"], 0xffffec86c8e8 /* 8 vars */) = 0

1103 brk(NULL) = 0xaaab03689000

1103 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffff801f9000

1103 faccessat(AT_FDCWD, "/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

1103 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

1103 fstat(3, {st_mode=S_IFREG|0644, st_size=8467, ...}) = 0

1103 mmap(NULL, 8467, PROT_READ, MAP_PRIVATE, 3, 0) = 0xffff801f6000

1103 close(3) = 0

1103 openat(AT_FDCWD, "/lib/aarch64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

1103 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0\360\206\2\0\0\0\0"..., 832) = 832

1103 fstat(3, {st_mode=S_IFREG|0755, st_size=1722920, ...}) = 0

1103 mmap(NULL, 1892240, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_DENYWRITE, -1, 0) = 0xffff7fff2000

1103 mmap(0xffff80000000, 1826704, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffff80000000

1103 munmap(0xffff7fff2000, 57344) = 0

1103 munmap(0xffff801be000, 8080) = 0

1103 mprotect(0xffff80199000, 81920, PROT_NONE) = 0

1103 mmap(0xffff801ad000, 20480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) = 0xffff801ad000

1103 mmap(0xffff801b2000, 49040, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xffff801b2000

1103 close(3) = 0

1103 set_tid_address(0xffff801f9fb0) = 1103

1103 set_robust_list(0xffff801f9fc0, 24) = 0

1103 rseq(0xffff801fa600, 0x20, 0, 0xd428bc00) = 0

1103 mprotect(0xffff801ad000, 12288, PROT_READ) = 0

1103 mprotect(0xaaaaca01f000, 4096, PROT_READ) = 0

1103 mprotect(0xffff801fe000, 8192, PROT_READ) = 0

1103 prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

1103 munmap(0xffff801f6000, 8467) = 0

1103 getrandom("\x0a\xbe\xe4\xe2\x9e\xc6\xf0\x15", 8, GRND_NONBLOCK) = 8

1103 brk(NULL) = 0xaaab03689000

1103 brk(0xaaab036aa000) = 0xaaab036aa000

1103 openat(AT_FDCWD, "res.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3

1103 openat(AT_FDCWD, "/dev/shm/sem.sem_parent", O_RDWR|O_NOFOLLOW|O_CLOEXEC) = 4

1103 fstat(4, {st_mode=S_IFREG|0644, st_size=32, ...}) = 0

1103 mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0xffff801f8000

1103 close(4) = 0

1103 openat(AT_FDCWD, "/dev/shm/sem.sem_child", O_RDWR|O_NOFOLLOW|O_CLOEXEC) = 4

1103 fstat(4, {st_mode=S_IFREG|0644, st_size=32, ...}) = 0

1103 mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0xffff801f7000

1103 close(4) = 0

1103 openat(AT_FDCWD, "/dev/shm/input_shm", O_RDWR|O_NOFOLLOW|O_CLOEXEC) = 4

1103 openat(AT_FDCWD, "/dev/shm/output_shm", O_RDWR|O_NOFOLLOW|O_CLOEXEC) = 5

1103 mmap(NULL, 100, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0xffff801f6000

1103 mmap(NULL, 100, PROT_READ|PROT_WRITE, MAP_SHARED, 5, 0) = 0xffff801f5000

1103 futex(0xffff801f8000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

1102 <... clock_nanosleep resumed>0xffffec86c5e8) = 0

1102 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\276: ", 27) = 27

1102 read(0, "40\n", 1024) = 3

1102 wait4(1103, NULL, WNOHANG, NULL) = 0

1102 futex(0xffff9cb17000, FUTEX_WAKE, 1) = 1

1103 <... futex resumed>) = 0

1102 futex(0xffff9cb16000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

1103 fstat(3, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0

1103 write(3, "40\n", 3) = 3

1103 futex(0xffff801f7000, FUTEX_WAKE, 1 <unfinished ...>

1102 <... futex resumed>) = 0

1103 <... futex resumed>) = 1

1102 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\276: ", 27 <unfinished ...>

1103 futex(0xffff801f8000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

1102 <... write resumed>) = 27

1102 read(0, "3002\n", 1024) = 5

1102 wait4(1103, NULL, WNOHANG, NULL) = 0

1102 futex(0xffff9cb17000, FUTEX_WAKE, 1) = 1

1103 <... futex resumed>) = 0

1102 futex(0xffff9cb16000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

1103 write(3, "3002\n", 5) = 5

1103 futex(0xffff801f7000, FUTEX_WAKE, 1 <unfinished ...>

1102 <... futex resumed>) = 0

1103 <... futex resumed>) = 1

1102 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\276: ", 27 <unfinished ...>

1103 futex(0xffff801f8000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

1102 <... write resumed>) = 27

1102 read(0, "-1\n", 1024) = 3

1102 wait4(1103, NULL, WNOHANG, NULL) = 0

1102 futex(0xffff9cb17000, FUTEX_WAKE, 1) = 1

1103 <... futex resumed>) = 0

1102 futex(0xffff9cb16000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

1103 write(3, "EXIT: -1\n", 9) = 9

1103 close(3) = 0

1103 futex(0xffff801f7000, FUTEX_WAKE, 1 <unfinished ...>

1102 <... futex resumed>) = 0

1103 <... futex resumed>) = 1

1102 write(1, "\320\237\320\276\320\273\321\203\321\207\320\265\320\275 \321\201\320\270\320\263\320\275\320\260\320\273 \320\267\320\260"..., 49 <unfinished ...>

1103 munmap(0xffff801f6000, 100 <unfinished ...>

1102 <... write resumed>) = 49

1103 <... munmap resumed>) = 0

1102 wait4(-1, <unfinished ...>

1103 munmap(0xffff801f5000, 100) = 0

1103 close(4) = 0

1103 close(5) = 0

1103 munmap(0xffff801f8000, 32) = 0

1103 munmap(0xffff801f7000, 32) = 0

1103 exit_group(0) = ?

1103 +++ exited with 0 +++

1102 <... wait4 resumed>NULL, 0, NULL) = 1103

1102 --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=1103, si_uid=0, si_status=0, si_utime=0, si_stime=0} ---

1102 munmap(0xffff9cb15000, 100) = 0

1102 munmap(0xffff9cb14000, 100) = 0

1102 close(3) = 0

1102 close(4) = 0

1102 munmap(0xffff9cb17000, 32) = 0

1102 munmap(0xffff9cb16000, 32) = 0

1102 unlinkat(AT_FDCWD, "/dev/shm/sem.sem_parent", 0) = 0

1102 unlinkat(AT_FDCWD, "/dev/shm/sem.sem_child", 0) = 0

1102 unlinkat(AT_FDCWD, "/dev/shm/input_shm", 0) = 0

1102 unlinkat(AT_FDCWD, "/dev/shm/output_shm", 0) = 0

1102 write(1, "\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320\271 \320\277\321\200\320\276\321"..., 57) = 57

1102 lseek(0, -1, SEEK_CUR) = -1 ESPIPE (Illegal seek)

1102 exit_group(0) = ?

1102 +++ exited with 0 +++

Вывод

В ходе лабораторной работы были успешно реализованы механизмы межпроцессного взаимодействия через Memory-mapped files и семафоры POSIX. Программа корректно обрабатывает ввод пользователя, проверяет числа на простоту и завершает работу при выполнении условий. Особое внимание уделено созданию и управлению именованными ресурсами: семафорами и областями разделяемой памяти, их корректному освобождению при завершении работы. Работа позволила получить практические навыки управления процессами, использования системных вызовов `shm_open()`, `mmap()`, `sem_open()` и организации межпроцессной коммуникации через отображаемые файлы в операционных системах.