# Лабораторная работа №3

## по курсу "Операционные системы"

**Выполнил: Юнусов Рустам М8О-310Б-22. 5 Вариант**

**Преподаватель: Миронов Евгений Сергеевич**

---

### *Задание:*

Пользователь вводит команды вида: «число<endline>». Далее это число передается от родительского процесса в дочерний. Дочерний процесс производит проверку на простоту. Если число составное, то в это число записывается в файл. Если число отрицательное или простое, то тогда дочерний и родительский процессы завершаются.

Использовать memory-mapped files

## Листинг программы

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

#define SHARED_MEM_FILE "shared_mem_file"
#define SEM_NAME "/mysemaphore"

struct shared_data {
    int number;
};

int is_prime(int n) {
    if (n <= 1) return 0;
    if (n <= 3) return 1;

    if (n % 2 == 0 || n % 3 == 0) return 0;

    for (int i = 5; i * i <= n; i += 6) {
        if (n % i == 0 || n % (i + 2) == 0)
            return 0;
    }
```

```
    return 1;
}

int main() {
    int fd;
    struct shared_data *shared;

    fd = open(SHARED_MEM_FILE, O_CREAT | O_RDWR, 0666);
    if (fd == -1) {
        fprintf(stderr, "Ошибка при открытии файла: %s\n",
strerror(errno));
        exit(EXIT_FAILURE);
    }

    if (ftruncate(fd, sizeof(struct shared_data)) == -1) {
        fprintf(stderr, "Ошибка при установке размера файла: %s\n",
strerror(errno));
        close(fd);
        exit(EXIT_FAILURE);
    }

    shared = mmap(NULL, sizeof(struct shared_data), PROT_READ |
PROT_WRITE, MAP_SHARED, fd, 0);
    if (shared == MAP_FAILED) {
        fprintf(stderr, "Ошибка при отображении файла: %s\n",
strerror(errno));
        close(fd);
        exit(EXIT_FAILURE);
    }

    sem_t *sem = sem_open(SEM_NAME, O_CREAT | O_EXCL, 0666, 0);
    if (sem == SEM_FAILED) {
        if (errno == EEXIST) {
            sem_unlink(SEM_NAME);
            sem = sem_open(SEM_NAME, O_CREAT, 0666, 0);
            if (sem == SEM_FAILED) {
                fprintf(stderr, "Ошибка при создании семафора: %s\n",
strerror(errno));
                munmap(shared, sizeof(struct shared_data));
                close(fd);
                exit(EXIT_FAILURE);
            }
        } else {
            fprintf(stderr, "Ошибка при создании семафора: %s\n",
strerror(errno));
            munmap(shared, sizeof(struct shared_data));
            close(fd);
            exit(EXIT_FAILURE);
        }
    }

    pid_t pid = fork();
```

```
    if (pid == -1) {
        fprintf(stderr, "Ошибка при вызове fork: %s\n",
strerror(errno));
        sem_close(sem);
        sem_unlink(SEM_NAME);
        munmap(shared, sizeof(struct shared_data));
        close(fd);
        exit(EXIT_FAILURE);
    }

    if (pid > 0) {
        // Родительский процесс
        while (1) {
            char input[256];
            printf("Введите число:\n");
            if (fgets(input, sizeof(input), stdin) == NULL) {
                fprintf(stderr, "Ошибка при чтении ввода\n");
                break;
            }

            int number = atoi(input);
            shared->number = number;

            if (sem_post(sem) == -1) {
                fprintf(stderr, "Ошибка при отправке сигнала семафора:
%s\n", strerror(errno));
                break;
            }

            int status;
            pid_t result = waitpid(pid, &status, WNOHANG);
            if (result == pid) {
                break;
            }
        }
        sem_close(sem);
        sem_unlink(SEM_NAME);
        munmap(shared, sizeof(struct shared_data));
        close(fd);
        unlink(SHARED_MEM_FILE);

        printf("Родительский процесс завершен.\n");
        exit(EXIT_SUCCESS);
    } else {
        // Дочерний процесс
        sem_t *sem_child = sem_open(SEM_NAME, 0);
        if (sem_child == SEM_FAILED) {
            fprintf(stderr, "Дочерний процесс: ошибка при открытии
семафора: %s\n", strerror(errno));
            munmap(shared, sizeof(struct shared_data));
            close(fd);
            exit(EXIT_FAILURE);
        }
```

```
        while (1) {
            if (sem_wait(sem_child) == -1) {
                fprintf(stderr, "Дочерний процесс: ошибка при ожидании
семафора: %s\n", strerror(errno));
                break;
            }

            int number = shared->number;

            if (number < 0 || is_prime(number)) {
                break;
            } else {
                FILE *fp = fopen("composites.txt", "a");
                if (fp == NULL) {
                    fprintf(stderr, "Ошибка при открытии файла для
записи: %s\n", strerror(errno));
                    break;
                }
                fprintf(fp, "%d\n", number);
                fclose(fp);
            }
        }

        sem_close(sem_child);
        munmap(shared, sizeof(struct shared_data));
        close(fd);

        printf("Дочерний процесс завершен.\n");
        exit(EXIT_SUCCESS);
    }
}
```

## Вывод

Программа реализует передачу данных между процессами через memory-mapped files с использованием семафоров для синхронизации. Родительский процесс принимает ввод числа от пользователя, записывает его в разделяемую память и сигнализирует дочернему процессу. Дочерний процесс проверяет число, если оно составное -- записывает его в файл, а при простом или отрицательном числе завершает работу, как и в первой лабораторной работе. Такой метод позволяет эффективно организовать межпроцессное взаимодействие не копируя лишние данные.