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: М80-214Б-23
Студент: Камышников И.С.
Преподаватель: Бахарев В.Д.
Оценка:_____
Дата: 17.02.25

Москва, 2025

Постановка задачи

Разработать программу на языке C, реализующую взаимодействие между родительским и дочерним процессами с использованием общей памяти и семафоров.
Родительский процесс должен принимать строки от пользователя и передавать их в **общую память**. Дочерний процесс должен проверять строки на соответствие правилам и записывать их в файл, если они валидны. Ошибки должны передаваться обратно родительскому процессу через **общую память ошибок**.

Общий метод и алгоритм решения

1. **Родительский процесс** считывает имя файла и передает его в **общую память**.
2. **Родительский процесс** порождает **дочерний процесс**.
3. **Родительский процесс** принимает строки от пользователя и передает их в **общую память**.
4. **Дочерний процесс** ожидает данные, считывает строки, проверяет их на **соответствие правилу**.
5. Если строка валидна, она записывается в файл. Если нет, сообщение об ошибке передается обратно в **родительский процесс**.
6. **Родительский процесс** отображает ошибки в консоли.
7. При вводе команды exit оба процесса завершают работу.

Используемые системные вызовы

- fork() — создание дочернего процесса.
- read() — чтение данных из файлового дескриптора.
- write() — запись данных в файловый дескриптор.
- open() — открытие файлового дескриптора.
- close() — закрытие файлового дескриптора.
- shm_open() — создание или открытие объекта общей памяти.
- shm_unlink() — удаление объекта общей памяти.
- mmap() — отображение файла или устройства на память.
- munmap() — удаление отображения памяти.
- sem_open() — создание или открытие семафора.
- sem_wait() — ожидание сигнала от семафора.
- sem_post() — освобождение семафора.
- sem_unlink() — удаление семафора.

Код программы

child.c

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

#define SHM_NAME "/shared_memory"
#define SHM_ERROR_NAME "/shared_error_memory"
#define SEM_PARENT_WRITE "/sem_parent_write"
#define SEM_CHILD_READ "/sem_child_read"
#define BUFFER_SIZE 1024

void report_error(const char *error_memory, const char *msg) {
    strncpy((char *)error_memory, msg, BUFFER_SIZE);
}

int ends_with_dot_or_semicolon(const char *str) {
    int len = strlen(str);
    return (len > 0 && (str[len - 1] == '.' || str[len - 1] == ';'));
}
```

```
int main(int argc, char *argv[]) {
    if (argc < 2) {
        write(STDERR_FILENO, "Не указано имя файла\n", 22);
        return -1;
    }
    int shm_fd = shm_open(SHM_NAME, O_RDWR, 0666);
    int shm_error_fd = shm_open(SHM_ERROR_NAME, O_RDWR, 0666);
    if (shm_fd == -1 || shm_error_fd == -1) {
        write(STDERR_FILENO, "Ошибка при открытии shared memory\n", 34);
        return -1;
    }
    char *shared_memory = mmap(NULL, BUFFER_SIZE, PROT_READ |
PROT_WRITE, MAP_SHARED, shm_fd, 0);
    char *error_memory = mmap(NULL, BUFFER_SIZE, PROT_READ |
PROT_WRITE, MAP_SHARED, shm_error_fd, 0);
    if (shared_memory == MAP_FAILED || error_memory == MAP_FAILED) {
        write(STDERR_FILENO, "Ошибка при отображении shared memory\n", 38);
        return -1;
    }
    sem_t *sem_parent_write = sem_open(SEM_PARENT_WRITE, 0);
    sem_t *sem_child_read = sem_open(SEM_CHILD_READ, 0);
    if (sem_parent_write == SEM_FAILED || sem_child_read == SEM_FAILED) {
        write(STDERR_FILENO, "Ошибка при открытии семафоров\n", 31);
        return -1;
    }
```

```
int fd = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC, 0644);
if (fd < 0) {
    report_error(error_memory, "Ошибка открытия файла\n");
    sem_post(sem_child_read);
    return -1;
}
while (1) {
    sem_wait(sem_parent_write);
    char buffer[BUFFER_SIZE];
    strncpy(buffer, shared_memory, BUFFER_SIZE);
    if (strcmp(buffer, "exit") == 0) {
        break;
    }
    if (ends_with_dot_or_semicolon(buffer)) {
        write(fd, buffer, strlen(buffer));
        write(fd, "\n", 1);
    } else {
        report_error(error_memory, "Строка не соответствует правилу!\n");
    }
    sem_post(sem_child_read);
}
close(fd);
munmap(shared_memory, BUFFER_SIZE);
munmap(error_memory, BUFFER_SIZE);
```

```
    sem_close(sem_parent_write);
    sem_close(sem_child_read);
    return 0;
}
```

main.c

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

#define SHM_NAME "/shared_memory"
#define SHM_ERROR_NAME "/shared_error_memory"
#define SEM_PARENT_WRITE "/sem_parent_write"
#define SEM_CHILD_READ "/sem_child_read"
#define BUFFER_SIZE 1024

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

```
int main() {
    pid_t pid;
    char filename[BUFFER_SIZE];
    char input[BUFFER_SIZE];
    char *shared_memory;
    char *error_memory;

    const char start_msg[] = "Введите имя файла: ";
    write(STDOUT_FILENO, start_msg, sizeof(start_msg));
    int filename_len = read(STDIN_FILENO, filename, sizeof(filename) - 1);
    if (filename_len <= 0) {
        print_error("Ошибка ввода имени файла\n");
        exit(EXIT_FAILURE);
    }
    filename[filename_len - 1] = '\0';

    int shm_fd = shm_open(SHM_NAME, O_CREAT | O_RDWR, 0666);
    int shm_error_fd = shm_open(SHM_ERROR_NAME, O_CREAT | O_RDWR,
0666);
    if (shm_fd == -1 || shm_error_fd == -1) {
        print_error("Ошибка при создании shared memory\n");
        exit(EXIT_FAILURE);
    }
```

```
    ftruncate(shm_fd, BUFFER_SIZE);

    ftruncate(shm_error_fd, BUFFER_SIZE);

    shared_memory = mmap(NULL, BUFFER_SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, shm_fd, 0);

    error_memory = mmap(NULL, BUFFER_SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, shm_error_fd, 0);

    if (shared_memory == MAP_FAILED || error_memory == MAP_FAILED) {

        print_error("Ошибка при отображении shared memory\n");

        exit(EXIT_FAILURE);

    }


    sem_t *sem_parent_write = sem_open(SEM_PARENT_WRITE, O_CREAT, 0666,
0);

    sem_t *sem_child_read = sem_open(SEM_CHILD_READ, O_CREAT, 0666, 0);

    if (sem_parent_write == SEM_FAILED || sem_child_read == SEM_FAILED) {

        print_error("Ошибка при создании семафоров\n");

        exit(EXIT_FAILURE);

    }


    pid = fork();

    if (pid < 0) {

        print_error("Ошибка при создании процесса\n");

        exit(EXIT_FAILURE);

    }
```

```
if (pid == 0) {
    execlp("./child", "./child", filename, (char *)NULL);
    print_error("Ошибка при запуске дочернего процесса\n");
    exit(EXIT_FAILURE);
}

const char msg1[] = "Введите строку (или 'exit' для выхода):\n";
write(STDOUT_FILENO, msg1, sizeof(msg1));

while (1) {
    int input_len = read(STDIN_FILENO, input, sizeof(input));
    if (input_len <= 0) {
        print_error("Ошибка ввода строки\n");
        break;
    }
    input[strcspn(input, "\n")] = 0;

    strncpy(shared_memory, input, BUFFER_SIZE);

    sem_post(sem_parent_write);

    if (strcmp(input, "exit") == 0) {
        break;
    }
```

```
        sem_wait(sem_child_read);

        if (strlen(error_memory) > 0) {
            write(STDERR_FILENO, error_memory, strlen(error_memory));
            memset(error_memory, 0, BUFFER_SIZE);
        }
    }

    wait(NULL);
    munmap(shared_memory, BUFFER_SIZE);
    munmap(error_memory, BUFFER_SIZE);
    shm_unlink(SHM_NAME);
    shm_unlink(SHM_ERROR_NAME);
    sem_close(sem_parent_write);
    sem_close(sem_child_read);
    sem_unlink(SEM_PARENT_WRITE);
    sem_unlink(SEM_CHILD_READ);

    return 0;
}
```

Тестировка

x104193ED0, 0x0, 0x1B6) = 8 0

1773/0xab31: open("a.txt\0", 0x601, 0x1A4) = 9 0

1773/0xab31: sem_wait(0x7, 0x0, 0x0) = 0 0

1773/0xab31: sem_post(0x8, 0x0, 0x0) = 0 0

1773/0xab31: sem_wait(0x7, 0x0, 0x0) = 0 0

1773/0xab31: sem_post(0x8, 0x0, 0x0) = 0 0

1771/0xaa10: read(0x0, "\n\0", 0x400) = 1 0

1771/0xaa10: sem_post(0x5, 0x0, 0x0) = 0 0

1771/0xaa10: sem_wait(0x6, 0x0, 0x0) = 0 0

1773/0xab31: sem_wait(0x7, 0x0, 0x0) = 0 0

1773/0xab31: sem_post(0x8, 0x0, 0x0) = 0 0

Ошибка: строка не соответствует правилу!

1771/0xaa10: read(0x0, "\n\a\0", 0x400) = 1 0

1771/0xaa10: sem_post(0x5, 0x0, 0x0) = 0 0

1773/0xab31: sem_wait(0x7, 0x0, 0x0) = 0 0

1771/0xaa10: sem_wait(0x6, 0x0, 0x0) = 0 0

1771/0xaa10: write(0x2, "\320\236\321\210\320\270\320\261\320\272\320\260:
\321\201\321\202\321\200\320\276\320\272\320\260 \320\275\320\265
\321\201\320\276\320\276\321\202\320\262\320\265\321\202\321\201\321\202\320\2
62\321\203\320\265\321\202
\320\277\321\200\320\260\320\262\320\270\320\273\321\203!\n\0", 0x4B) = 75 0

1773/0xab31: sem_post(0x8, 0x0, 0x0) = 0 0

1771/0xaa10: read(0x0, "\n\a\0", 0x400) = 1 0

1771/0xaa10: sem_post(0x5, 0x0, 0x0) = 0 0

1773/0xab31: sem_wait(0x7, 0x0, 0x0) = 0 0

1771/0xaa10: sem_wait(0x6, 0x0, 0x0) = 0 0

1773/0xab31: sem_post(0x8, 0x0, 0x0) = 0 0

Ошибка: строка не соответствует правилу!

1771/0xaa10: read(0x0, "\n\a\0", 0x400) = 1 0

1771/0xaa10: sem_post(0x5, 0x0, 0x0) = 0 0

1773/0xab31: sem_wait(0x7, 0x0, 0x0) = 0 0

1771/0xaa10: sem_wait(0x6, 0x0, 0x0) = 0 0

1771/0xaa10: write(0x2, "\320\236\321\210\320\270\320\261\320\272\320\260:
\321\201\321\202\321\200\320\276\320\272\320\260 \320\275\320\265
\321\201\320\276\320\276\321\202\320\262\320\265\321\202\321\201\321\202\320\2
62\321\203\320\265\321\202
\320\277\321\200\320\260\320\262\320\270\320\273\321\203!\n\0", 0x4B) = 75 0

1773/0xab31: sem_post(0x8, 0x0, 0x0) = 0 0

1771/0xaa10: read(0x0, "\n\0", 0x400) = 1 0

1771/0xaa10: sem_post(0x5, 0x0, 0x0) = 0 0

1773/0xab31: sem_wait(0x7, 0x0, 0x0) = 0 0

1771/0xaa10: sem_wait(0x6, 0x0, 0x0) = 0 0

1773/0xab31: sem_post(0x8, 0x0, 0x0) = 0 0

Ошибка: строка не соответствует правилу!

1771/0xaa10: read(0x0, "\n\0", 0x400) = 1 0

1771/0xaa10: sem_post(0x5, 0x0, 0x0) = 0 0

1773/0xab31: sem_wait(0x7, 0x0, 0x0) = 0 0

1771/0xaa10: sem_wait(0x6, 0x0, 0x0) = 0 0

1773/0xab31: sem_post(0x8, 0x0, 0x0) = 0 0

1771/0xaa10: write(0x2, "\320\236\321\210\320\270\320\261\320\272\320\260:
\321\201\321\202\321\200\320\276\320\272\320\260 \320\275\320\265
\321\201\320\276\320\276\321\202\320\262\320\265\321\202\321\201\321\202\320\2
62\321\203\320\265\321\202
\320\277\321\200\320\260\320\262\320\270\320\273\321\203!\n\0", 0x4B) = 75 0

1771/0xaa10: read(0x0, "\n\0", 0x400) = 1 0

1771/0xaa10: sem_post(0x5, 0x0, 0x0) = 0 0

1773/0xab31: sem_wait(0x7, 0x0, 0x0) = 0 0

1771/0xaa10: sem_wait(0x6, 0x0, 0x0) = 0 0

1773/0xab31: sem_post(0x8, 0x0, 0x0) = 0 0, 0x0) = -1 Err#2

1773/0xab31: stat64("/usr/lib/system/libdispatch.dylib\0", 0x16BC6BB10, 0x0) = -1
Err#2

1773/0xab31: open("/dev/dtracehelper\0", 0x2, 0x0) = 3 0

1773/0xab31: ioctl(0x3, 0x80086804, 0x16BC6DB58) = 0 0

1773/0xab31: close(0x3) = 0 0

1773/0xab31: open("/Users/ivankamblshnikov/Documents/os/lab3/child\0", 0x0, 0x0)
= 3 0

1773/0xab31: __mac_syscall(0x18555E908, 0x2, 0x16BC6D1D0) = 0 0

1773/0xab31: map_with_linking_np(0x16BC6D080, 0x1, 0x16BC6D0B0) = 0 0

1773/0xab31: close(0x3) = 0 0

1773/0xab31: mprotect(0x104194000, 0x4000, 0x1) = 0 0

1773/0xab31: shared_region_check_np(0xFFFFFFFFFFFFFFFF, 0x0, 0x0) = 0 0

1773/0xab31: mprotect(0x1045A0000, 0x40000, 0x1) = 0 0

1773/0xab31: access("/AppleInternal/XBS/.isChrooted\0", 0x0, 0x0) = -1 Err#2

1773/0xab31: bsdthread_register(0x18584CE34, 0x18584CE28, 0x4000) = 1073742303 0

1773/0xab31: getpid(0x0, 0x0, 0x0) = 1773 0

1773/0xab31: shm_open(0x1856E6F51, 0x0, 0x45A0030) = 3 0

1773/0xab31: fstat64(0x3, 0x16BC6DFE0, 0x0) = 0 0

1773/0xab31: mmap(0x0, 0x4000, 0x1, 0x40001, 0x3, 0x0) = 0x1045EC000 0

1773/0xab31: close(0x3) = 0 0

1773/0xab31: ioctl(0x2, 0x4004667A, 0x16BC6E08C) = 0 0

1773/0xab31: mprotect(0x1045F8000, 0x4000, 0x0) = 0 0

1773/0xab31: mprotect(0x104604000, 0x4000, 0x0) = 0 0

1773/0xab31: mprotect(0x104608000, 0x4000, 0x0) = 0 0

1773/0xab31: mprotect(0x104614000, 0x4000, 0x0) = 0 0

1773/0xab31: mprotect(0x104618000, 0x4000, 0x0) = 0 0

1773/0xab31: mprotect(0x104624000, 0x4000, 0x0) = 0 0

1773/0xab31: mprotect(0x1045F0000, 0x98, 0x1) = 0 0

1773/0xab31: mprotect(0x1045F0000, 0x98, 0x3) = 0 0

1773/0xab31: mprotect(0x1045F0000, 0x98, 0x1) = 0 0

1773/0xab31: mprotect(0x104628000, 0x4000, 0x1) = 0 0

1773/0xab31: mprotect(0x10462C000, 0x98, 0x1) = 0 0

1773/0xab31: mprotect(0x10462C000, 0x98, 0x3) = 0 0

1773/0xab31: mprotect(0x10462C000, 0x98, 0x1) = 0 0

1773/0xab31: mprotect(0x1045F0000, 0x98, 0x3) = 0 0

1773/0xab31: mprotect(0x1045F0000, 0x98, 0x1) = 0 0

1773/0xab31: mprotect(0x104628000, 0x4000, 0x3) = 0 0

1773/0xab31: mprotect(0x104628000, 0x4000, 0x1) = 0 0

1773/0xab31: mprotect(0x1045A0000, 0x40000, 0x3) = 0 0

1773/0xab31: mprotect(0x1045A0000, 0x40000, 0x1) = 0 0

1773/0xab31: objc_bp_assist_cfg_np(0x185485400, 0x80000018001C1048, 0x0) = -1 Err#5

1773/0xab31: issetugid(0x0, 0x0, 0x0) = 0 0

1773/0xab31: mprotect(0x1045A0000, 0x40000, 0x3) = 0 0

1773/0xab31: getentropy(0x16BC6D7F8, 0x20, 0x0) = 0 0

1773/0xab31: mprotect(0x1045A0000, 0x40000, 0x1) = 0 0

1773/0xab31: mprotect(0x1045A0000, 0x40000, 0x3) = 0 0

1773/0xab31: mprotect(0x1045A0000, 0x40000, 0x1) = 0 0

1773/0xab31: getattrlist("/Users/ivankamblshnikov/Documents/os/lab3/child\0", 0x16BC6DF70, 0x16BC6DF88) = 0 0

1773/0xab31: access("/Users/ivankamblshnikov/Documents/os/lab3\0", 0x4, 0x0) = 0 0

1773/0xab31: open("/Users/ivankamblshnikov/Documents/os/lab3\0", 0x0, 0x0) = 3 0

1773/0xab31: fstat64(0x3, 0x1576044C0, 0x0) = 0 0

1773/0xab31: csrctl(0x0, 0x16BC6E19C, 0x4) = 0 0

1773/0xab31: fcntl(0x3, 0x32, 0x16BC6DE58) = 0 0

1773/0xab31: close(0x3) = 0 0

1773/0xab31: open("/Users/ivankamblshnikov/Documents/os/lab3/Info.plist\0", 0x0, 0x0) = -1 Err#2

1773/0xab31: proc_info(0x2, 0x6ED, 0xD) = 64 0

1773/0xab31: csops_audittoken(0x6ED, 0x10, 0x16BC6E1E0) = 0 0

1773/0xab31: sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16BC6E538, 0x16BC6E530, 0x188BFED3D, 0x15) = 0 0

1773/0xab31: sysctl([CTL_KERN, 138, 0, 0, 0, 0] (2), 0x16BC6E5C8, 0x16BC6E5C0, 0x0, 0x0) = 0 0

1773/0xab31: csops(0x6ED, 0x0, 0x16BC6E66C) = 0 0

1773/0xab31: mprotect(0x1045A0000, 0x40000, 0x3) = 0 0

1773/0xab31: shm_open(0x104193E2C, 0x2, 0x1B6) = 3 0

1773/0xab31: shm_open(0x104193E40, 0x2, 0x1B6) = 4 0

1773/0xab31: mmap(0x0, 0x400, 0x3, 0x40001, 0x3, 0x0) = 0x10419C000 0

1773/0xab31: mmap(0x0, 0x400, 0x3, 0x40001, 0x4, 0x0) = 0x1041A0000 0

1773/0xab31: sem_open(0x104193EBE, 0x0, 0x1B6) = 7 0

1773/0xab31: sem_open(0x104193ED0, 0x0, 0x1B6) = 8 0

1773/0xab31: open("a.txt\0", 0x601, 0x1A4) = 9 0

1773/0xab31: sem_wait(0x7, 0x0, 0x0) = 0 0

1773/0xab31: sem_post(0x8, 0x0, 0x0) = 0 0

1773/0xab31: sem_wait(0x7, 0x0, 0x0) = 0 0

1773/0xab31: sem_post(0x8, 0x0, 0x0) = 0 0

1771/0xaa10: read(0x0, "\n\0", 0x400) = 1 0

1771/0xaa10: sem_post(0x5, 0x0, 0x0) = 0 0

1771/0xaa10: sem_wait(0x6, 0x0, 0x0) = 0 0

1773/0xab31: sem_wait(0x7, 0x0, 0x0) = 0 0

1773/0xab31: sem_post(0x8, 0x0, 0x0) = 0 0

Ошибка: строка не соответствует правилу!

1771/0xaa10: read(0x0, "\n\a\0", 0x400) = 1 0

1771/0xaa10: sem_post(0x5, 0x0, 0x0) = 0 0

1773/0xab31: sem_wait(0x7, 0x0, 0x0) = 0 0

1771/0xaa10: sem_wait(0x6, 0x0, 0x0) = 0 0

1771/0xaa10: write(0x2, "\320\236\321\210\320\270\320\261\320\272\320\260:
\321\201\321\202\321\200\320\276\320\272\320\260 \320\275\320\265
\321\201\320\276\320\276\321\202\320\262\320\265\321\202\321\201\321\202\320\2
62\321\203\320\265\321\202
\320\277\321\200\320\260\320\262\320\270\320\273\321\203!\n\0", 0x4B) = 75 0

1773/0xab31: sem_post(0x8, 0x0, 0x0) = 0 0

1771/0xaa10: read(0x0, "\n\a\0", 0x400) = 1 0

1771/0xaa10: sem_post(0x5, 0x0, 0x0) = 0 0

1773/0xab31: sem_wait(0x7, 0x0, 0x0) = 0 0

1771/0xaa10: sem_wait(0x6, 0x0, 0x0) = 0 0

1773/0xab31: sem_post(0x8, 0x0, 0x0) = 0 0

Ошибка: строка не соответствует правилу!

1771/0xaa10: read(0x0, "\n\a\0", 0x400) = 1 0

1771/0xaa10: sem_post(0x5, 0x0, 0x0) = 0 0

1773/0xab31: sem_wait(0x7, 0x0, 0x0) = 0 0

1771/0xaa10: sem_wait(0x6, 0x0, 0x0) = 0 0

1771/0xaa10: write(0x2, "\320\236\321\210\320\270\320\261\320\272\320\260:
\321\201\321\202\321\200\320\276\320\272\320\260 \320\275\320\265

\321\201\320\276\320\276\321\202\320\262\320\265\321\202\321\201\321\202\320\262\321\203\320\265\321\202
\320\277\321\200\320\260\320\262\320\270\320\273\321\203!\n\0", 0x4B) = 75 0

1773/0xab31: sem_post(0x8, 0x0, 0x0) = 0 0

1771/0xaa10: read(0x0, "\n\0", 0x400) = 1 0

1771/0xaa10: sem_post(0x5, 0x0, 0x0) = 0 0

1773/0xab31: sem_wait(0x7, 0x0, 0x0) = 0 0

1771/0xaa10: sem_wait(0x6, 0x0, 0x0) = 0 0

1773/0xab31: sem_post(0x8, 0x0, 0x0) = 0 0

Ошибка: строка не соответствует правилу!

1771/0xaa10: read(0x0, "\n\0", 0x400) = 1 0

1771/0xaa10: sem_post(0x5, 0x0, 0x0) = 0 0

1773/0xab31: sem_wait(0x7, 0x0, 0x0) = 0 0

1771/0xaa10: sem_wait(0x6, 0x0, 0x0) = 0 0

1773/0xab31: sem_post(0x8, 0x0, 0x0) = 0 0

1771/0xaa10: write(0x2, "\320\236\321\210\320\270\320\261\320\272\320\260:
\321\201\321\202\321\200\320\276\320\272\320\260 \320\275\320\265
\321\201\320\276\320\276\321\202\320\262\320\265\321\202\321\201\321\202\320\262\321\203\320\265\321\202
\320\277\321\200\320\260\320\262\320\270\320\273\321\203!\n\0", 0x4B) = 75 0

1771/0xaa10: read(0x0, "\n\0", 0x400) = 1 0

1771/0xaa10: sem_post(0x5, 0x0, 0x0) = 0 0

1773/0xab31: sem_wait(0x7, 0x0, 0x0) = 0 0

1771/0xaa10: sem_wait(0x6, 0x0, 0x0) = 0 0

1773/0xab31: sem_post(0x8, 0x0, 0x0) = 0 0

Вывод

В ходе выполнения лабораторной работы была написана программа на языке C, реализующая:

- Создание дочернего процесса.
- Перенаправление стандартных потоков ввода/вывода.
- Взаимодействие между процессами с использованием общей памяти и семафоров.
- Проверку строк на соответствие заданному правилу.

Программа успешно протестирована и соответствует поставленной задаче.