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-216Б-24

Студент: Котляр Д.А.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 20.11.25

Москва, 2025

Постановка задачи

Вариант 15.

Родительский процесс создает дочерний процесс. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child проверяет строки на валидность правилу. Если строка соответствует правилу, то она выводится в стандартный поток вывода дочернего процесса, иначе в pipe2 выводится информация об ошибке. Родительский процесс полученные от child ошибки выводит в стандартный поток вывода. Нужно взять свою первую лабу и переделать её с использованием shared memory и memory mapping.

Вариант 15) Правило проверки: строка должна начинаться с заглавной буквы

Общий метод и алгоритм решения

Использованные системные вызовы:

- int shm_open(const char *name, int oflag, mode_t mode) – создание/открытие объекта разделяемой памяти
- int ftruncate(int fd, off_t length) – установка размера объекта разделяемой памяти
- void *mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset) – отображение разделяемой памяти в адресное пространство процесса
- sem_t *sem_open(const char *name, int oflag, mode_t mode, unsigned int value) – создание/открытие именованного семафора
- int sem_wait(sem_t *sem) – ожидание семафора (уменьшение значения)
- int sem_post(sem_t *sem) – освобождение семафора (увеличение значения)
- int munmap(void *addr, size_t length) – удаление отображения разделяемой памяти
- int shm_unlink(const char *name) – удаление объекта разделяемой памяти
- int sem_close(sem_t *sem) – закрытие семафора
- int sem_unlink(const char *name) – удаление именованного семафора
- pid_t wait(int *status) – ожидание завершения дочернего процесса
- int open(const char *pathname, int flags, mode_t mode) – открытие файла
- int close(int fd) – закрытие файлового дескриптора
- ssize_t read(int fd, void *buf, size_t count) – чтение из файлового дескриптора
- ssize_t write(int fd, const void *buf, size_t count) – запись в файловый дескриптора
- void exit(int status) – завершение процесса

В рамках лабораторной работы была разработана Разработана многопроцессная система для обработки числовых данных из файла с использованием разделяемой памяти и семафоров. Система состоит из серверного и клиентского процессов, взаимодействующих через именованную разделяемую память.

Серверный процесс запрашивает имя файла, открывает его и создает объекты разделяемой памяти и семафоры для синхронизации. Затем он порождает клиентский процесс, который подключается к этим же объектам. Сервер последовательно читает числа из файла и записывает их в разделяемую память, а клиентский процесс считывает эти числа, проверяет их на простоту и выводит только непростые положительные числа. Встреча простого или отрицательного числа приводит к завершению клиента.

Взаимодействие процессов синхронизируется с помощью двух семафоров, обеспечивая корректный обмен данными. После завершения работы все ресурсы (разделяемая память и семафоры) корректно освобождаются серверным процессом.

Код программы

parent.c

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

#define BUFFER_SIZE 1024

typedef struct {
    size_t size;
    char buffer[BUFFER_SIZE];
} shm_data_t;

int main() {
    pid_t pid = getpid();

    char shm_name[64];
    char sem_parent_name[64];
    char sem_child_name[64];

    snprintf(shm_name, sizeof(shm_name), "/shm_%d", pid);
    snprintf(sem_parent_name, sizeof(sem_parent_name), "/sem_p_%d", pid);
    snprintf(sem_child_name, sizeof(sem_child_name), "/sem_c_%d", pid);

    int shm_fd = shm_open(shm_name, O_CREAT | O_RDWR, 0600);
    if (shm_fd == -1) {
        perror("shm_open");
```

```
    exit(EXIT_FAILURE);
}

ftruncate(shm_fd, sizeof(shm_data_t));

shm_data_t *shm = mmap(NULL, sizeof(shm_data_t),
                       PROT_READ | PROT_WRITE,
                       MAP_SHARED, shm_fd, 0);

if (shm == MAP_FAILED) {
    perror("mmap");
    exit(EXIT_FAILURE);
}

sem_t *sem_parent = sem_open(sem_parent_name, O_CREAT, 0600, 0);
sem_t *sem_child  = sem_open(sem_child_name,  O_CREAT, 0600, 0);

if (sem_parent == SEM_FAILED || sem_child == SEM_FAILED) {
    perror("sem_open");
    exit(EXIT_FAILURE);
}

char filename[BUFFER_SIZE];
printf("Введите имя файла для записи: ");
fgets(filename, BUFFER_SIZE, stdin);
filename[strcspn(filename, "\n")] = 0;

int file_fd = open(filename, O_WRONLY | O_CREAT | O_TRUNC, 0644);
if (file_fd == -1) {
    perror("open");
    exit(EXIT_FAILURE);
}

pid_t child = fork();
if (child == 0) {
    execl("./child", "child",
          shm_name, sem_parent_name, sem_child_name, NULL);
    perror("execl");
    exit(EXIT_FAILURE);
}

char input[BUFFER_SIZE];
printf("Вводите строки:\n");

while (1) {
    printf("> ");
    fflush(stdout);

    if (!fgets(input, BUFFER_SIZE, stdin))
        break;

    if (strcmp(input, "exit\n") == 0)
```

```
            break;

        strncpy(shm->buffer, input, BUFFER_SIZE);
        shm->size = strlen(input);

        sem_post(sem_child);
        sem_wait(sem_parent);

        if (strncmp(shm->buffer, "ERROR", 5) == 0) {
            printf("Ошибка от child: %s", shm->buffer);
        } else {
            write(file_fd, shm->buffer, shm->size);
        }
    }

    close(file_fd);
    wait(NULL);

    munmap(shm, sizeof(shm_data_t));
    shm_unlink(shm_name);

    sem_close(sem_parent);
    sem_close(sem_child);
    sem_unlink(sem_parent_name);
    sem_unlink(sem_child_name);

    return 0;
}
```

child.c

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

#define BUFFER_SIZE 1024

typedef struct {
    size_t size;
    char buffer[BUFFER_SIZE];
} shm_data_t;

int main(int argc, char *argv[]) {
    if (argc != 4) {
        fprintf(stderr, "Usage: child <shm> <sem_parent> <sem_child>\n");
        exit(EXIT_FAILURE);
    }
```

```
    const char *shm_name = argv[1];
    const char *sem_parent_name = argv[2];
    const char *sem_child_name = argv[3];

    int shm_fd = shm_open(shm_name, O_RDWR, 0600);
    if (shm_fd == -1) {
        perror("shm_open");
        exit(EXIT_FAILURE);
    }

    shm_data_t *shm = mmap(NULL, sizeof(shm_data_t),
                           PROT_READ | PROT_WRITE,
                           MAP_SHARED, shm_fd, 0);

    if (shm == MAP_FAILED) {
        perror("mmap");
        exit(EXIT_FAILURE);
    }

    sem_t *sem_parent = sem_open(sem_parent_name, 0);
    sem_t *sem_child  = sem_open(sem_child_name,  0);

    if (sem_parent == SEM_FAILED || sem_child == SEM_FAILED) {
        perror("sem_open");
        exit(EXIT_FAILURE);
    }

    while (1) {
        sem_wait(sem_child);

        if (isupper((unsigned char)shm->buffer[0])) {

        } else {
            char truncated[200];
            strncpy(truncated, shm->buffer, 199);
            truncated[199] = '\0';

            snprintf(shm->buffer, BUFFER_SIZE,
                    "ERROR: Строка не начинается с заглавной буквы: %s\n",
                    truncated);
            shm->size = strlen(shm->buffer);
        }

        sem_post(sem_parent);
    }

    munmap(shm, sizeof(shm_data_t));
    sem_close(sem_parent);
    sem_close(sem_child);

    return 0;
}
```

Протокол работы программы

Strace

```
4320  execve("./parent", ["./parent"], 0x7ffd1499a7c8 /* 69 vars */) = 0
4320  brk(NULL)                         = 0x5a0defb5c000
4320  mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x70d39bdad000
4320  access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
4320  openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
4320  fstat(3, {st_mode=S_IFREG|0644, st_size=75827, ...}) = 0
4320  mmap(NULL, 75827, PROT_READ, MAP_PRIVATE, 3, 0) = 0x70d39bd9a000
4320  close(3)                          = 0
4320  openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
4320  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
4320  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
4320  fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
4320  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
4320  mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x70d39ba00000
4320  mmap(0x70d39ba28000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x70d39ba28000
4320  mmap(0x70d39bbb0000, 323584, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x70d39bbb0000
4320  mmap(0x70d39bbff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x70d39bbff000
4320  mmap(0x70d39bc05000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x70d39bc05000
4320  close(3)                          = 0
4320  mmap(NULL, 12288, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x70d39bd97000
4320  arch_prctl(ARCH_SET_FS, 0x70d39bd97740) = 0
4320  set_tid_address(0x70d39bd97a10)   = 4320
4320  set_robust_list(0x70d39bd97a20, 24) = 0
4320  rseq(0x70d39bd98060, 0x20, 0, 0x53053053) = 0
4320  mprotect(0x70d39bbff000, 16384, PROT_READ) = 0
4320  mprotect(0x5a0ddf7b5000, 4096, PROT_READ) = 0
4320  mprotect(0x70d39bdeb000, 8192, PROT_READ) = 0
4320  prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
4320  munmap(0x70d39bd9a000, 75827)     = 0
4320  getpid()                          = 4320

4320  openat(AT_FDCWD, "/dev/shm/shm_4320",
O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0600) = 3
4320  ftruncate(3, 1032)                = 0
4320  mmap(NULL, 1032, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) =
0x70d39bdac000

4320  openat(AT_FDCWD, "/dev/shm/sem.sem_p_4320",
O_RDWR|O_NOFOLLOW|O_CLOEXEC) = -1 ENOENT (No such file or directory)
```

4320 getrandom("\x98\x60\x95\x14\xe9\x9a\x64\x69", 8, GRND_NONBLOCK) = 8
4320 newfstatat(AT_FDCWD, "/dev/shm/sem.M3e4pf", 0x7ffc293efa90,
AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)

**4320 openat(AT_FDCWD, "/dev/shm/sem.M3e4pf",
O_RDWR|O_CREAT|O_EXCL|O_NOFOLLOW|O_CLOEXEC, 0600) = 4**
4320 write(4, "\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
**4320 mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) =
0x70d39bdab000**
4320 link("/dev/shm/sem.M3e4pf", "/dev/shm/sem.sem_p_4320") = 0

4320 fstat(4, {st_mode=S_IFREG|0600, st_size=32, ...}) = 0
4320 getrandom("\xb9\xd0\x69\xb4\xd0\x32\xb0\x20", 8, GRND_NONBLOCK) = 8
4320 brk(NULL) = 0x5a0defb5c000
4320 brk(0x5a0defb7d000) = 0x5a0defb7d000
4320 unlink("/dev/shm/sem.M3e4pf") = 0
4320 close(4) = 0
4320 openat(AT_FDCWD, "/dev/shm/sem.sem_c_4320",
O_RDWR|O_NOFOLLOW|O_CLOEXEC) = -1 ENOENT (No such file or directory)
4320 getrandom("\xe8\x3b\x8b\x87\x13\x41\xe7\xde", 8, GRND_NONBLOCK) = 8
4320 newfstatat(AT_FDCWD, "/dev/shm/sem.AXyYIa", 0x7ffc293efa90,
AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)

**4320 openat(AT_FDCWD, "/dev/shm/sem.AXyYIa",
O_RDWR|O_CREAT|O_EXCL|O_NOFOLLOW|O_CLOEXEC, 0600) = 4**
4320 write(4, "\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
**4320 mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) =
0x70d39bdaa000**
4320 link("/dev/shm/sem.AXyYIa", "/dev/shm/sem.sem_c_4320") = 0

4320 fstat(4, {st_mode=S_IFREG|0600, st_size=32, ...}) = 0
4320 unlink("/dev/shm/sem.AXyYIa") = 0
4320 close(4) = 0
4320 fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
4320 fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
4320 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"..., 54) = 54
4320 read(0, "res.txt\n", 1024) = 8
4320 openat(AT_FDCWD, "res.txt", O_WRONLY|O_CREAT|O_TRUNC, 0644) = 4
4320 clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x70d39bd97a10) = 4485
4485 set_robust_list(0x70d39bd97a20, 24) = 0
4320 write(1, "\320\222\320\262\320\276\320\264\320\270\321\202\320\265
\321\201\321\202\321\200\320\276\320\272\320\270:\n", 29) = 29
4485 execve("./child", ["child", "/shm_4320", "/sem_p_4320", "/sem_c_4320"],
0x7ffc293f0848 /* 69 vars */ <unfinished ...>
4320 write(1, "> ", 2) = 2
4320 read(0, <unfinished ...>
4485 <... execve resumed>) = 0
4485 brk(NULL) = 0x644ee8202000
4485 mmap(NULL, 8192, PROT_READ|PROT_WRITE,

```
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75bb8a90f000
4485  access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
4485  openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
4485  fstat(3, {st_mode=S_IFREG|0644, st_size=75827, ...}) = 0
4485  mmap(NULL, 75827, PROT_READ, MAP_PRIVATE, 3, 0) = 0x75bb8a8fc000
4485  close(3)                          = 0
4485  openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
4485  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
4485  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
4485  fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
4485  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
4485  mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x75bb8a600000
4485  mmap(0x75bb8a628000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x75bb8a628000
4485  mmap(0x75bb8a7b0000, 323584, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x75bb8a7b0000
4485  mmap(0x75bb8a7ff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x75bb8a7ff000
4485  mmap(0x75bb8a805000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x75bb8a805000
4485  close(3)                          = 0
4485  mmap(NULL, 12288, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75bb8a8f9000
4485  arch_prctl(ARCH_SET_FS, 0x75bb8a8f9740) = 0
4485  set_tid_address(0x75bb8a8f9a10)   = 4485
4485  set_robust_list(0x75bb8a8f9a20, 24) = 0
4485  rseq(0x75bb8a8fa060, 0x20, 0, 0x53053053) = 0
4485  mprotect(0x75bb8a7ff000, 16384, PROT_READ) = 0
4485  mprotect(0x644eba89c000, 4096, PROT_READ) = 0
4485  mprotect(0x75bb8a94d000, 8192, PROT_READ) = 0
4485  prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
4485  munmap(0x75bb8a8fc000, 75827)     = 0
4485  openat(AT_FDCWD, "/dev/shm/shm_4320", O_RDWR|O_NOFOLLOW|O_CLOEXEC)
= 3
4485  mmap(NULL, 1032, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) =
0x75bb8a90e000
4485  openat(AT_FDCWD, "/dev/shm/sem.sem_p_4320",
O_RDWR|O_NOFOLLOW|O_CLOEXEC) = 5
4485  fstat(5, {st_mode=S_IFREG|0600, st_size=32, ...}) = 0
4485  getrandom("\xc0\xd2\x16\x65\x8f\xc0\xbb\x94", 8, GRND_NONBLOCK) = 8
4485  brk(NULL)                         = 0x644ee8202000
4485  brk(0x644ee8223000)               = 0x644ee8223000
4485  mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 5, 0) =
0x75bb8a90d000
4485  close(5)                          = 0
4485  openat(AT_FDCWD, "/dev/shm/sem.sem_c_4320",
O_RDWR|O_NOFOLLOW|O_CLOEXEC) = 5
```

4485 fstat(5, {st_mode=S_IFREG|0600, st_size=32, ...}) = 0
4485 mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 5, 0) =
0x75bb8a90c000
4485 close(5) = 0

4485 futex(0x75bb8a90c000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

4320 <... read resumed>"Stroka1\n", 1024) = 8

4320 futex(0x70d39bdaa000, FUTEX_WAKE, 1) = 1

4485 <... futex resumed>) = 0
4320 futex(0x70d39bdab000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
4485 futex(0x75bb8a90d000, FUTEX_WAKE, 1 <unfinished ...>
4320 <... futex resumed>) = -1 EAGAIN (Resource temporarily unavailable)
4485 <... futex resumed>) = 0
4320 write(4, "Stroka1\n", 8 <unfinished ...>
4485 futex(0x75bb8a90c000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
4320 <... write resumed>) = 8
4320 write(1, "> ", 2) = 2
4320 read(0, "STRING TWOO\n", 1024) = 12
4320 futex(0x70d39bdaa000, FUTEX_WAKE, 1) = 1
4485 <... futex resumed>) = 0
4320 futex(0x70d39bdab000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
4485 futex(0x75bb8a90d000, FUTEX_WAKE, 1 <unfinished ...>
4320 <... futex resumed>) = -1 EAGAIN (Resource temporarily unavailable)
4485 <... futex resumed>) = 0
4320 write(4, "STRING TWOO\n", 12 <unfinished ...>
4485 futex(0x75bb8a90c000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
4320 <... write resumed>) = 12
4320 write(1, "> ", 2) = 2
4320 read(0, "mini stroka\n", 1024) = 12
4320 futex(0x70d39bdaa000, FUTEX_WAKE, 1) = 1
4485 <... futex resumed>) = 0
4320 futex(0x70d39bdab000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
4485 futex(0x75bb8a90d000, FUTEX_WAKE, 1 <unfinished ...>
4320 <... futex resumed>) = -1 EAGAIN (Resource temporarily unavailable)
4485 <... futex resumed>) = 0
4320 write(1, "\320\236\321\210\320\270\320\261\320\272\320\260 \320\276\321\202 child:
ERROR: "..., 118 <unfinished ...>
4485 futex(0x75bb8a90c000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
4320 <... write resumed>) = 118
4320 write(1, "> ", 2) = 2
4320 read(0, "popa\n", 1024) = 5
4320 futex(0x70d39bdaa000, FUTEX_WAKE, 1) = 1

```
4485  <... futex resumed>)              = 0
4320  futex(0x70d39bdab000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
4485  futex(0x75bb8a90d000, FUTEX_WAKE, 1 <unfinished ...>
4320  <... futex resumed>)              = -1 EAGAIN (Resource temporarily unavailable)
4485  <... futex resumed>)              = 0
4320  write(1, "\320\236\321\210\320\270\320\261\320\272\320\260 \320\276\321\202 child:
ERROR: "..., 111 <unfinished ...>
4485  futex(0x75bb8a90c000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
4320  <... write resumed>)              = 111
4320  write(1, "> ", 2)                 = 2
4320  read(0, "It's end...\n", 1024)    = 12
4320  futex(0x70d39bdaa000, FUTEX_WAKE, 1) = 1
4485  <... futex resumed>)              = 0
4320  futex(0x70d39bdab000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
4485  futex(0x75bb8a90d000, FUTEX_WAKE, 1 <unfinished ...>
4320  <... futex resumed>)              = -1 EAGAIN (Resource temporarily unavailable)
4485  <... futex resumed>)              = 0
4320  write(4, "It's end...\n", 12 <unfinished ...>
4485  futex(0x75bb8a90c000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
4320  <... write resumed>)              = 12
4320  write(1, "> ", 2)                 = 2
4320  read(0, "", 1024)                 = 0
4320  close(4)                          = 0
4320  wait4(-1, NULL, 0, NULL)          = ? ERESTARTSYS (To be restarted if SA_RESTART
is set)
4485  <... futex resumed>)              = ? ERESTARTSYS (To be restarted if SA_RESTART is
set)
4320  --- SIGINT {si_signo=SIGINT, si_code=SI_KERNEL} ---
4485  --- SIGINT {si_signo=SIGINT, si_code=SI_KERNEL} ---
4320  +++ exited with 0 +++
```

Вывод

Программа корректно создает дочерний процесс и организует передачу данных между процессами с использованием разделяемой памяти и семафоров, что подтверждается анализом системных вызовов strace.

Основные проблемы при выполнении работы возникли с пониманием принципов работы с разделяемой памятью, особенно в области правильного создания и управления именованными объектами shared memory, а также сложности вызвало обеспечение корректного доступа к одним и тем же объектам разделяемой памяти из разных процессов и правильное освобождение системных ресурсов (памяти и семафоров) при завершении работы программы.

В процессе работы были успешно реализованы уникальные имена для объектов разделяемой памяти и семафоров на основе PID процесса, что исключает конфликты при параллельном запуске нескольких экземпляров программы.