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-213БВ-24

Студент: Месропян А.Э.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 31.10.25

Москва, 2025

Постановка задачи

Вариант 17.

Родительский процесс создает два дочерних процесса. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись для child1. Аналогично для второй строки и процесса child2. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1 или в pipe2 в зависимости от правила фильтрации. Процесс child1 и child2 производят работу над строками. Процессы пишут результаты своей работы в стандартный вывод. Вариант 17) Правило фильтрации: строки длины больше 10 символов отправляются в pipe2, иначе в pipe1. Дочерние процессы удаляют все гласные из строк.

Общий метод и алгоритм решения

Общий метод: Разделение задач (фильтрация и обработка строк) между родительским и двумя дочерними процессами, используя **Разделяемую память (Shared Memory, SHM)** для передачи данных и **Именованные POSIX-семафоры** для синхронизации доступа и уведомления дочерних процессов о наличии новых данных (реализация паттерна **Производитель-Потребитель**).

Использованные системные вызовы:

1. pid_t fork(): Создание дочернего процесса.
2. int execlp(...): Замена образа памяти дочернего процесса на программу ./child_run.
3. pid_t waitpid(...): Ожидание завершения дочернего процесса.
4. int shm_open(...): Создание/открытие сегмента разделяемой памяти.
5. int ftruncate(...): Установка размера сегмента SHM.
6. void* mmap(...): Проецирование (отображение) SHM в адресное пространство процесса.
7. int munmap(...): Удаление проецирования SHM.
8. sem_t* sem_open(...): Создание/открытие именованного семафора.
9. int sem_wait(...): Ожидание сигнала от родительского процесса (блокирующая операция).
10. int sem_post(...): Отправка сигнала дочернему процессу.
11. int shm_unlink(...): Удаление именованного сегмента SHM.
12. int sem_unlink(...): Удаление именованного семафора.

Алгоритм работы программы:

Инициализация (Родительский процесс)

- Родительский процесс запрашивает и считывает имена двух выходных файлов (file1_name, file2_name).
- Вызывается init_ipc_channel() для каждого канала:
    - Генерируются уникальные имена для двух сегментов SHM и двух именованных семафоров.
    - Создаются два сегмента SHM (shm_open), их размер устанавливается (ftruncate), и они отображаются в память родительского процесса (mmap).
    - Создаются два именованных семафора (sem_open) с начальным значением 0.

Создание процессов

- Родительский процесс создает два дочерних процесса (pid_1 и pid_2) через fork().
- Дочерний процесс 1 и 2:
    - Запускают программу ./child_run через execlp(), передавая в качестве аргументов имя своего выходного файла, а также уникальные имена сегмента SHM и семафора.

Обработка данных (Дочерний процесс ./child_run)

- Дочерний процесс:
    - Открывает и отображает в свое адресное пространство выделенный ему сегмент SHM и семафор.
    - Входит в бесконечный цикл while (1):
        - **Ждет сигнала от родителя** с помощью sem_wait().
        - После получения сигнала, проверяет содержимое SHM: если там "QUIT" или пустая строка, выходит из цикла.
        - Копирует строку из SHM в локальный буфер.
        - Вызывает функцию remove_vowels(str), которая удаляет все латинские гласные.
        - Записывает обработанную строку в соответствующий выходной файл (fputs()).

Распределение данных (Фильтрация, Родительский процесс)

- Родительский процесс читает строки из своего стандартного ввода (stdin) с помощью fgets() до тех пор, пока не введена строка "QUIT".
- Применяется правило фильтрации по длине (FILTER_LENGTH):
    - Если длина строки **превышает** FILTER_LENGTH (т.е., len - 1 > FILTER_LENGTH), она отправляется в SHM **Child 2**.
    - В противном случае, она отправляется в SHM **Child 1**.
- Родительский процесс:
    - Записывает строку в выделенный сегмент SHM (strncpy).
    - **Разблокирует дочерний процесс** с помощью sem_post(), сигнализируя, что данные готовы к обработке.
    - Печатает в stdout информацию о том, какому дочернему процессу была отправлена строка и её длину.

Завершение работы

- При вводе строки "QUIT" родительский процесс:
    - Записывает строку "QUIT" в оба сегмента SHM.
    - Вызывает sem_post() для обоих семафоров, чтобы гарантировать, что оба дочерних процесса проснутся и корректно завершатся.
    - Ожидает завершения обоих дочерних процессов (waitpid()).
- После завершения дочерних процессов, родительский процесс вызывает cleanup_ipc(): удаляет проецирование SHM (munmap), отсоединяет сегменты SHM (shm_unlink) и удаляет именованные семафоры (sem_close, sem_unlink).
- Программа завершается, возвращая STATUS_OK.

Код программы

status_codes.h

```c
#ifndef STATUS_CODES_H
#define STATUS_CODES_H

typedef enum {
    STATUS_OK = 0,
    INVALID_INPUT = 1,
    OPEN_ERROR = 2,
    FORK_ERROR = 3,
    PIPE_ERROR = 4,
    EXEC_ERROR = 5,
    SEM_ERROR = 6,
    INIT_ERROR = 7,
    MEMORY_ERROR = 8
} StatusCode;

#endif
```

functions.h

```c
#include <stdio.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <semaphore.h>
#include <ctype.h>
#include <sys/wait.h>
#include <time.h>
#include "status_codes.h"

#define MAX_LINE_LENGTH 1024
#define MAX_FILENAME_LENGTH 256
#define FILTER_LENGTH 10
#define SHM_SIZE MAX_LINE_LENGTH

void remove_vowels(char* str);

void generate_unique_name(char *buffer, size_t buf_size, const char *prefix);
```

utils.c

```c
#include "../include/functions.h"

void remove_vowels(char* str) {
    if (!str) {
        return;
    }
    int write_idx = 0;
    for (int read_idx = 0; str[read_idx]; read_idx++) {
        char c = str[read_idx];
        char lower_c = tolower((unsigned char)c);
        if (lower_c != 'a' && lower_c != 'e' && lower_c != 'i' &&
            lower_c != 'o' && lower_c != 'u' && lower_c != 'y') {
                str[write_idx++] = lower_c;
            }
    }
    str[write_idx] = '\0';
}

void generate_unique_name(char* buffer, size_t buf_size, const char* prefix) {
    pid_t pid = getpid();
    unsigned long cur_time = (unsigned long)time(NULL);
    snprintf(buffer, buf_size, "/%s_%d_%lu", prefix, pid, cur_time);
}
```

child.c

```c
#include "../include/functions.h"

typedef struct {
    char shm_name[64];
    char sem_name[64];
    char output_file[MAX_FILENAME_LENGTH];
} ChildArgs;

int main(int argc, char* argv[]) {
    if (argc != 4) {
        return INVALID_INPUT;
    }

    const char* output_name = argv[1];
    const char* shm_name = argv[2];
    const char* sem_name = argv[3];
    FILE* output = NULL;
    int shm_fd = -1;
    char* shm_ptr = NULL;
    sem_t* sem_ptr = SEM_FAILED;
    char buffer[MAX_LINE_LENGTH];

    output = fopen(output_name, "w");
    if (output == NULL) {
        perror("Child: Ошибка fopen");
        return OPEN_ERROR;
    }

    sem_ptr = sem_open(sem_name, 0);
    if (sem_ptr == SEM_FAILED) {
        perror("Child: Ошибка sem_open");
        goto cleanup_file;
    }

    shm_fd = shm_open(shm_name, O_RDWR, 0666);
    if (shm_fd == -1) {
        perror("Child: Ошибка shm_open");
        goto cleanup_sem;
    }

    shm_ptr = mmap(0, SHM_SIZE, PROT_READ, MAP_SHARED, shm_fd, 0);
    close(shm_fd);
    if (shm_ptr == MAP_FAILED) {
        perror("Child: Ошибка mmap");
        goto cleanup_sem;
    }
```

```c
    }

    while (1) {
        if (sem_wait(sem_ptr) == -1) {
            perror("Child: Ошибка sem_wait");
            break;
        }

        if (strncmp(shm_ptr, "QUIT", 4) == 0 || shm_ptr[0] == '\0') {
            break;
        }

        strncpy(buffer, shm_ptr, MAX_LINE_LENGTH);
        buffer[MAX_LINE_LENGTH - 1] = '\0';


        remove_vowels(buffer);

        if (fputs(buffer, output) == EOF) {
            perror("Child: Ошибка fputs");
            break;
        }
    }

    munmap(shm_ptr, SHM_SIZE);
cleanup_sem:
    sem_close(sem_ptr);
cleanup_file:
    fclose(output);
    return STATUS_OK;
}
```

parent.c

```c
#include "../include/functions.h"

typedef struct {
    char shm_name[64];
    char sem_name[64];
    int shm_fd;
    char* shm_ptr;
    sem_t* sem_ptr;
    pid_t pid;
} Ipc_channel;

StatusCode init_ipc_channel(Ipc_channel* channel, const char* prefix) {
    generate_unique_name(channel->shm_name, sizeof(channel->shm_name), prefix);
    generate_unique_name(channel->sem_name, sizeof(channel->sem_name), prefix);

    channel->shm_fd = shm_open(channel->shm_name, O_CREAT | O_RDWR, 0666);
    if (channel->shm_fd == -1) {
        return OPEN_ERROR;
    }

    ftruncate(channel->shm_fd, SHM_SIZE);
    channel->shm_ptr = mmap(NULL, SHM_SIZE, PROT_WRITE, MAP_SHARED, channel->shm_fd, 0);
    close(channel->shm_fd);
    if (channel->shm_ptr == MAP_FAILED) {
        return MEMORY_ERROR;
    }
    channel->shm_ptr[0] = '\0';
    channel->sem_ptr = sem_open(channel->sem_name, O_CREAT, 0666, 0);
    if (channel->sem_ptr == MAP_FAILED) {
        munmap(channel->shm_ptr, SHM_SIZE);
        shm_unlink(channel->shm_name);
        return SEM_ERROR;
    }
    return STATUS_OK;
}

void start_child_process(Ipc_channel* channel, const char* child_program, const char* output_name) {
    char child_path[256];
    snprintf(child_path, sizeof(child_path), "./lab3/%s", child_program);

    execlp(child_path, child_path, (char*)output_name,
           (char*)channel->shm_name, (char*)channel->sem_name, (char*)NULL);
    exit(EXEC_ERROR);
}
```

```c
void cleanup_ipc(Ipc_channel* channel) {
    if (channel->shm_ptr != MAP_FAILED) {
        munmap(channel->shm_ptr, SHM_SIZE);
    }
    shm_unlink(channel->shm_name);

    if (channel->sem_ptr != SEM_FAILED) {
        sem_close(channel->sem_ptr);
        sem_unlink(channel->sem_name);
    }
}

int main() {
    Ipc_channel ch1, ch2;
    char file1_name[MAX_FILENAME_LENGTH], file2_name[MAX_FILENAME_LENGTH];
    char buffer[MAX_LINE_LENGTH];
    StatusCode status = STATUS_OK;

    const char* prompt1 = "Введите имя файла для child 1: ";
    write(STDOUT_FILENO, prompt1, strlen(prompt1));
    if (fgets(file1_name, MAX_FILENAME_LENGTH, stdin) == NULL) return INVALID_INPUT;
    file1_name[strcspn(file1_name, "\n")] = '\0';

    const char* prompt2 = "Введите имя файла для child 2: ";
    write(STDOUT_FILENO, prompt2, strlen(prompt2));
    if (fgets(file2_name, MAX_FILENAME_LENGTH, stdin) == NULL) return INVALID_INPUT;
    file2_name[strcspn(file2_name, "\n")] = '\0';

    if (init_ipc_channel(&ch1, "ch1") != STATUS_OK ||
        init_ipc_channel(&ch2, "ch2") != STATUS_OK) {
        status = INIT_ERROR;
        goto cleanup_all;
    }

    if ((ch1.pid = fork()) == -1) {
        const char* error_msg = "Parent: Ошибка fork 1\n";
        write(STDERR_FILENO, error_msg, strlen(error_msg));
        status = FORK_ERROR;
        goto cleanup_all;
    }
    if (ch1.pid == 0) {
        start_child_process(&ch1, "./child_run", file1_name);
    }

    if ((ch2.pid = fork()) == -1) {
```

```
int main() {
    if ((ch2.pid = fork()) == -1) {
        const char* error_msg = "Parent: Ошибка fork 2\n";
        write(STDERR_FILENO, error_msg, strlen(error_msg));
        status = FORK_ERROR;
        goto cleanup_all;
    }
    if (ch2.pid == 0) {
        start_child_process(&ch2, "./child_run", file2_name);
    }

    const char* ready_msg = "\n--- Parent: Ready for input ---\n";
    write(STDOUT_FILENO, ready_msg, strlen(ready_msg));

    while (fgets(buffer, sizeof(buffer), stdin) != NULL) {
        int len = strlen(buffer);
        if (len > 0 && buffer[len-1] == '\n') {
        } else {
            if (len < sizeof(buffer) - 1) {
                buffer[len++] = '\n';
                buffer[len] = '\0';
            }
        }

        if (strncmp(buffer, "QUIT", 4) == 0) {
            break;
        }

        Ipc_channel* target_channel;
        if (len - 1 > FILTER_LENGTH) {
            target_channel = &ch2;
            const char* send_msg2 = "Parent: Sending to CHILD 2 (length: ";
            write(STDOUT_FILENO, send_msg2, strlen(send_msg2));
            char len_buf[32];
            snprintf(len_buf, sizeof(len_buf), "%d", len - 1);
            write(STDOUT_FILENO, len_buf, strlen(len_buf));
            write(STDOUT_FILENO, ")\n", 2);
        } else {
            target_channel = &ch1;
            const char* send_msg1 = "Parent: Sending to CHILD 1 (length: ";
            write(STDOUT_FILENO, send_msg1, strlen(send_msg1));
            char len_buf[32];
            snprintf(len_buf, sizeof(len_buf), "%d", len - 1);
            write(STDOUT_FILENO, len_buf, strlen(len_buf));
            write(STDOUT_FILENO, ")\n", 2);
```

```
        strncpy(target_channel->shm_ptr, buffer, SHM_SIZE);
        target_channel->shm_ptr[SHM_SIZE - 1] = '\0';

        if (sem_post(target_channel->sem_ptr) == -1) {
            const char* error_msg = "Parent: Ошибка sem_post\n";
            write(STDERR_FILENO, error_msg, strlen(error_msg));
            status = SEM_ERROR;
            break;
        }
    }

    strcpy(ch1.shm_ptr, "QUIT");
    sem_post(ch1.sem_ptr);

    strcpy(ch2.shm_ptr, "QUIT");
    sem_post(ch2.sem_ptr);

    waitpid(ch1.pid, NULL, 0);
    waitpid(ch2.pid, NULL, 0);
    const char* child_done_msg = "Parent: Дочерние процессы завершены.\n";
    write(STDOUT_FILENO, child_done_msg, strlen(child_done_msg));

cleanup_all:
    cleanup_ipc(&ch1);
    cleanup_ipc(&ch2);
    const char* ipc_cleanup_msg = "Parent: Все IPC-ресурсы очищены.\n";
    write(STDOUT_FILENO, ipc_cleanup_msg, strlen(ipc_cleanup_msg));
    return status;
}
```

Протокол работы программы

Тестирование:

```
armani@LAPTOP-F5TL4SI7:~/OS_Labs/build$ ./lab3/parent_run
Введите имя файла для child 1: short_output.txt
Введите имя файла для child 2: long_output.txt

--- Parent: Ready for input ---
Hi its me man!
Parent: Sending to CHILD 2 (length: 14)
LOL why are you reading this?
Parent: Sending to CHILD 2 (length: 29)
i wanted to be shorter
Parent: Sending to CHILD 2 (length: 22)
ok short line
Parent: Sending to CHILD 2 (length: 13)
ok short
Parent: Sending to CHILD 1 (length: 8)
wow lol
Parent: Sending to CHILD 1 (length: 7)
omg this
Parent: Sending to CHILD 1 (length: 9)
thanks bye
Parent: Sending to CHILD 1 (length: 10)
QUIT
Parent: Дочерние процессы завершены.
Parent: Все IPC-ресурсы очищены.
armani@LAPTOP-F5TL4SI7:~/OS_Labs/build$ cat short_output.txt
k shrt
ww ll
mg ths
thnks b
armani@LAPTOP-F5TL4SI7:~/OS_Labs/build$ cat long_output.txt
h ts m mn!
ll wh r  rdng ths?
 wntd t b shrtr
k shrt ln
```

Strace:

armani@LAPTOP-F5TL4SI7:~/OS_Labs/build$ strace -f ./lab3/parent_run

execve("./lab3/parent_run", ["./lab3/parent_run"], 0x7ffc5a817118 /* 37 vars */) = 0

brk(NULL)   = 0x5f4c14ab1000

arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc0c5b48e0) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x77e9d7e62000

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

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=17368, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 17368, PROT_READ, MAP_PRIVATE, 3, 0) = 0x77e9d7e5d000

```
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\0P\237\2\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"..., 784, 64) = 784

pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0"..., 48, 848) = 48

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O{\f\225\\=\201\327\312\301P\32$\230\266\235"..., 68, 896)
= 68

newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 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"..., 784, 64) = 784

mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x77e9d7c00000

mprotect(0x77e9d7c28000, 2023424, PROT_NONE) = 0

mmap(0x77e9d7c28000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x77e9d7c28000

mmap(0x77e9d7dbd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1bd000) = 0x77e9d7dbd000

mmap(0x77e9d7e16000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x77e9d7e16000

mmap(0x77e9d7e1c000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x77e9d7e1c000

close(3)                                = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x77e9d7e5a000

arch_prctl(ARCH_SET_FS, 0x77e9d7e5a740) = 0

set_tid_address(0x77e9d7e5aa10)         = 52104

set_robust_list(0x77e9d7e5aa20, 24)     = 0

rseq(0x77e9d7e5b0e0, 0x20, 0, 0x53053053) = 0

mprotect(0x77e9d7e16000, 16384, PROT_READ) = 0

mprotect(0x5f4bdc22b000, 4096, PROT_READ) = 0

mprotect(0x77e9d7e9c000, 8192, PROT_READ) = 0

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

munmap(0x77e9d7e5d000, 17368)           = 0

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"..., 49Введите имя файла для child 1: ) = 49

newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x6), ...}, AT_EMPTY_PATH) = 0
```

```
getrandom("\x5e\x33\x75\xa0\xfd\x2a\xdb\x0b", 8, GRND_NONBLOCK) = 8

brk(NULL)                               = 0x5f4c14ab1000

brk(0x5f4c14ad2000)                     = 0x5f4c14ad2000

read(0, Hello this is a big line

"Hello this is a big line\n", 1024) = 25

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"..., 49Введите имя файла для child 2: ) = 49

read(0, oh small

"oh small\n", 1024)             = 9

getpid()                                = 52104

getpid()                                = 52104

openat(AT_FDCWD, "/dev/shm/ch1_52104_1763917285",
O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0666) = 3

ftruncate(3, 1024)                      = 0

mmap(NULL, 1024, PROT_WRITE, MAP_SHARED, 3, 0) = 0x77e9d7e9b000

close(3)                                = 0

openat(AT_FDCWD, "/dev/shm/sem.ch1_52104_1763917285", O_RDWR|O_NOFOLLOW) = -1
ENOENT (No such file or directory)

getrandom("\xd4\x7e\x93\x69\x55\xcc\xdb\x70", 8, GRND_NONBLOCK) = 8

newfstatat(AT_FDCWD, "/dev/shm/sem.kJcp6R", 0x7ffc0c5b3e40, AT_SYMLINK_NOFOLLOW) = -1
ENOENT (No such file or directory)

openat(AT_FDCWD, "/dev/shm/sem.kJcp6R", O_RDWR|O_CREAT|O_EXCL, 0666) = 3

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

mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x77e9d7e61000

link("/dev/shm/sem.kJcp6R", "/dev/shm/sem.ch1_52104_1763917285") = 0

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH) = 0

unlink("/dev/shm/sem.kJcp6R")           = 0

close(3)                                = 0

getpid()                                = 52104

getpid()                                = 52104

openat(AT_FDCWD, "/dev/shm/ch2_52104_1763917285",
O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0666) = 3

ftruncate(3, 1024)                      = 0

mmap(NULL, 1024, PROT_WRITE, MAP_SHARED, 3, 0) = 0x77e9d7e60000
```

close(3)   = 0

openat(AT_FDCWD, "/dev/shm/sem.ch2_52104_1763917285", O_RDWR|O_NOFOLLOW) = -1 ENOENT (No such file or directory)

getrandom("\xba\xb2\x4d\x01\x11\x82\xd0\x86", 8, GRND_NONBLOCK) = 8

newfstatat(AT_FDCWD, "/dev/shm/sem.eDY5Nc", 0x7ffc0c5b3e40, AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/dev/shm/sem.eDY5Nc", O_RDWR|O_CREAT|O_EXCL, 0666) = 3

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

mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x77e9d7e5f000

link("/dev/shm/sem.eDY5Nc", "/dev/shm/sem.ch2_52104_1763917285") = 0

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH) = 0

unlink("/dev/shm/sem.eDY5Nc")   = 0

close(3)   = 0

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 52177 attached

, child_tidptr=0x77e9d7e5aa10) = 52177

[pid 52177] set_robust_list(0x77e9d7e5aa20, 24 <unfinished ...>

[pid 52104] clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD <unfinished ...>

[pid 52177] <... set_robust_list resumed>) = 0

[pid 52177] execve("./lab3/./child_run", ["./lab3/./child_run", "Hello this is a big line", "/ch1_52104_1763917285", "/ch1_52104_1763917285"], 0x7ffc0c5b4ab8 /* 37 vars */strace: Process 52178 attached

 <unfinished ...>

[pid 52104] <... clone resumed>, child_tidptr=0x77e9d7e5aa10) = 52178

[pid 52178] set_robust_list(0x77e9d7e5aa20, 24 <unfinished ...>

[pid 52104] write(1, "\n--- Parent: Ready for input ---"..., 33 <unfinished ...>

--- Parent: Ready for input ---

[pid 52178] <... set_robust_list resumed>) = 0

[pid 52104] <... write resumed>)   = 33

[pid 52104] read(0, <unfinished ...>

[pid 52178] execve("./lab3/./child_run", ["./lab3/./child_run", "oh small", "/ch2_52104_1763917285", "/ch2_52104_1763917285"], 0x7ffc0c5b4ab8 /* 37 vars */ <unfinished ...>

```
[pid 52177] <... execve resumed>)       = 0

[pid 52177] brk(NULL)                   = 0x5b9a0e1ba000

[pid 52177] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffddd3ea380 <unfinished ...>

[pid 52178] <... execve resumed>)       = 0

[pid 52177] <... arch_prctl resumed>)   = -1 EINVAL (Invalid argument)

[pid 52178] brk(NULL <unfinished ...>

[pid 52177] mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 52178] <... brk resumed>)          = 0x584b19b23000

[pid 52177] <... mmap resumed>)         = 0x7e3bf2b94000

[pid 52178] arch_prctl(0x3001 /* ARCH_??? */, 0x7fffdf414b50 <unfinished ...>

[pid 52177] access("/etc/ld.so.preload", R_OK <unfinished ...>

[pid 52178] <... arch_prctl resumed>)   = -1 EINVAL (Invalid argument)

[pid 52177] <... access resumed>)       = -1 ENOENT (No such file or directory)

[pid 52178] mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 52177] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>

[pid 52178] <... mmap resumed>)         = 0x79566af3a000

[pid 52177] <... openat resumed>)       = 3

[pid 52178] access("/etc/ld.so.preload", R_OK <unfinished ...>

[pid 52177] newfstatat(3, "",  <unfinished ...>

[pid 52178] <... access resumed>)       = -1 ENOENT (No such file or directory)

[pid 52177] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=17368, ...},
AT_EMPTY_PATH) = 0

[pid 52178] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>

[pid 52177] mmap(NULL, 17368, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>

[pid 52178] <... openat resumed>)       = 3

[pid 52178] newfstatat(3, "",  <unfinished ...>

[pid 52177] <... mmap resumed>)         = 0x7e3bf2b8f000

[pid 52178] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=17368, ...},
AT_EMPTY_PATH) = 0

[pid 52177] close(3 <unfinished ...>

[pid 52178] mmap(NULL, 17368, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>
```

[pid 52177] <... close resumed>) = 0

[pid 52178] <... mmap resumed>) = 0x79566af35000

[pid 52177] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC <unfinished ...>

[pid 52178] close(3 <unfinished ...>

[pid 52177] <... openat resumed>) = 3

[pid 52178] <... close resumed>) = 0

[pid 52177] read(3, <unfinished ...>

[pid 52178] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC <unfinished ...>

[pid 52177] <... read resumed>"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0"..., 832) = 832

[pid 52178] <... openat resumed>) = 3

[pid 52177] pread64(3, <unfinished ...>

[pid 52178] read(3, <unfinished ...>

[pid 52177] <... pread64 resumed>"\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 52178] <... read resumed>"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0"..., 832) = 832

[pid 52177] pread64(3, <unfinished ...>

[pid 52178] pread64(3, <unfinished ...>

[pid 52177] <... pread64 resumed>"\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

[pid 52178] <... pread64 resumed>"\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 52177] pread64(3, <unfinished ...>

[pid 52178] pread64(3, <unfinished ...>

[pid 52177] <... pread64 resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O{\f\225\\=\201\327\312\301P\32$\230\266\235"..., 68, 896) = 68

[pid 52178] <... pread64 resumed>"\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

[pid 52177] newfstatat(3, "", <unfinished ...>

[pid 52178] pread64(3, <unfinished ...>

[pid 52177] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0

[pid 52178] <... pread64 resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O{\f\225\\=\201\327\312\301P\32$\230\266\235"..., 68, 896) = 68

[pid 52177] pread64(3, <unfinished ...>

[pid 52178] newfstatat(3, "", <unfinished ...>

[pid 52177] <... pread64 resumed>"\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 52178] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0

[pid 52177] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished ...>

[pid 52178] pread64(3, <unfinished ...>

[pid 52177] <... mmap resumed>)   = 0x7e3bf2800000

[pid 52178] <... pread64 resumed>"\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 52177] mprotect(0x7e3bf2828000, 2023424, PROT_NONE <unfinished ...>

[pid 52178] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished ...>

[pid 52177] <... mprotect resumed>)   = 0

[pid 52178] <... mmap resumed>)   = 0x79566ac00000

[pid 52177] mmap(0x7e3bf2828000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000 <unfinished ...>

[pid 52178] mprotect(0x79566ac28000, 2023424, PROT_NONE <unfinished ...>

[pid 52177] <... mmap resumed>)   = 0x7e3bf2828000

[pid 52178] <... mprotect resumed>)   = 0

[pid 52177] mmap(0x7e3bf29bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000 <unfinished ...>

[pid 52178] mmap(0x79566ac28000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000 <unfinished ...>

[pid 52177] <... mmap resumed>)   = 0x7e3bf29bd000

[pid 52178] <... mmap resumed>)   = 0x79566ac28000

[pid 52177] mmap(0x7e3bf2a16000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000 <unfinished ...>

[pid 52178] mmap(0x79566adbd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000 <unfinished ...>

[pid 52177] <... mmap resumed>)   = 0x7e3bf2a16000

[pid 52178] <... mmap resumed>)   = 0x79566adbd000

[pid 52177] mmap(0x7e3bf2a1c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 52178] mmap(0x79566ae16000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000 <unfinished ...>

[pid 52177] <... mmap resumed>) = 0x7e3bf2a1c000

[pid 52178] <... mmap resumed>) = 0x79566ae16000

[pid 52177] close(3 <unfinished ...>

[pid 52178] mmap(0x79566ae1c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 52177] <... close resumed>) = 0

[pid 52178] <... mmap resumed>) = 0x79566ae1c000

[pid 52177] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 52178] close(3 <unfinished ...>

[pid 52177] <... mmap resumed>) = 0x7e3bf2b8c000

[pid 52178] <... close resumed>) = 0

[pid 52177] arch_prctl(ARCH_SET_FS, 0x7e3bf2b8c740 <unfinished ...>

[pid 52178] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 52177] <... arch_prctl resumed>) = 0

[pid 52178] <... mmap resumed>) = 0x79566af32000

[pid 52177] set_tid_address(0x7e3bf2b8ca10 <unfinished ...>

[pid 52178] arch_prctl(ARCH_SET_FS, 0x79566af32740 <unfinished ...>

[pid 52177] <... set_tid_address resumed>) = 52177

[pid 52178] <... arch_prctl resumed>) = 0

[pid 52177] set_robust_list(0x7e3bf2b8ca20, 24 <unfinished ...>

[pid 52178] set_tid_address(0x79566af32a10 <unfinished ...>

[pid 52177] <... set_robust_list resumed>) = 0

[pid 52178] <... set_tid_address resumed>) = 52178

[pid 52177] rseq(0x7e3bf2b8d0e0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 52178] set_robust_list(0x79566af32a20, 24 <unfinished ...>

[pid 52177] <... rseq resumed>) = 0

[pid 52178] <... set_robust_list resumed>) = 0

[pid 52178] rseq(0x79566af330e0, 0x20, 0, 0x53053053 <unfinished ...>

```
[pid 52177] mprotect(0x7e3bf2a16000, 16384, PROT_READ <unfinished ...>
[pid 52178] <... rseq resumed>)         = 0
[pid 52177] <... mprotect resumed>)     = 0
[pid 52178] mprotect(0x79566ae16000, 16384, PROT_READ <unfinished ...>
[pid 52177] mprotect(0x5b99f6a94000, 4096, PROT_READ <unfinished ...>
[pid 52178] <... mprotect resumed>)     = 0
[pid 52177] <... mprotect resumed>)     = 0
[pid 52178] mprotect(0x584aed3fb000, 4096, PROT_READ <unfinished ...>
[pid 52177] mprotect(0x7e3bf2bce000, 8192, PROT_READ <unfinished ...>
[pid 52178] <... mprotect resumed>)     = 0
[pid 52177] <... mprotect resumed>)     = 0
[pid 52178] mprotect(0x79566af74000, 8192, PROT_READ <unfinished ...>
[pid 52177] prlimit64(0, RLIMIT_STACK, NULL,  <unfinished ...>
[pid 52178] <... mprotect resumed>)     = 0
[pid 52177] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
[pid 52178] prlimit64(0, RLIMIT_STACK, NULL,  <unfinished ...>
[pid 52177] munmap(0x7e3bf2b8f000, 17368 <unfinished ...>
[pid 52178] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
[pid 52177] <... munmap resumed>)       = 0
[pid 52178] munmap(0x79566af35000, 17368 <unfinished ...>
[pid 52177] getrandom( <unfinished ...>
[pid 52178] <... munmap resumed>)       = 0
[pid 52177] <... getrandom resumed>"\x8b\x34\xfb\x21\xb8\x66\xeb\xb3", 8, GRND_NONBLOCK) = 8
[pid 52178] getrandom( <unfinished ...>
[pid 52177] brk(NULL <unfinished ...>
[pid 52178] <... getrandom resumed>"\x3f\x18\x20\xf7\x10\x60\x1e\xa7", 8, GRND_NONBLOCK) = 8
[pid 52177] <... brk resumed>)          = 0x5b9a0e1ba000
[pid 52178] brk(NULL <unfinished ...>
[pid 52177] brk(0x5b9a0e1db000 <unfinished ...>
[pid 52178] <... brk resumed>)          = 0x584b19b23000
[pid 52177] <... brk resumed>)          = 0x5b9a0e1db000
```

[pid 52178] brk(0x584b19b44000) = 0x584b19b44000

[pid 52177] openat(AT_FDCWD, "Hello this is a big line", O_WRONLY|O_CREAT|O_TRUNC, 0666 <unfinished ...>

[pid 52178] openat(AT_FDCWD, "oh small", O_WRONLY|O_CREAT|O_TRUNC, 0666 <unfinished ...>

[pid 52177] <... openat resumed>) = 3

[pid 52178] <... openat resumed>) = 3

[pid 52177] openat(AT_FDCWD, "/dev/shm/sem.ch1_52104_1763917285", O_RDWR|O_NOFOLLOW <unfinished ...>

[pid 52178] openat(AT_FDCWD, "/dev/shm/sem.ch2_52104_1763917285", O_RDWR|O_NOFOLLOW <unfinished ...>

[pid 52177] <... openat resumed>) = 4

[pid 52178] <... openat resumed>) = 4

[pid 52177] newfstatat(4, "", <unfinished ...>

[pid 52178] newfstatat(4, "", <unfinished ...>

[pid 52177] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH) = 0

[pid 52178] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH) = 0

[pid 52177] mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0 <unfinished ...>

[pid 52178] mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0 <unfinished ...>

[pid 52177] <... mmap resumed>) = 0x7e3bf2bcd000

[pid 52178] <... mmap resumed>) = 0x79566af73000

[pid 52177] close(4 <unfinished ...>

[pid 52178] close(4 <unfinished ...>

[pid 52177] <... close resumed>) = 0

[pid 52178] <... close resumed>) = 0

[pid 52177] openat(AT_FDCWD, "/dev/shm/ch1_52104_1763917285", O_RDWR|O_NOFOLLOW|O_CLOEXEC <unfinished ...>

[pid 52178] openat(AT_FDCWD, "/dev/shm/ch2_52104_1763917285", O_RDWR|O_NOFOLLOW|O_CLOEXEC <unfinished ...>

[pid 52177] <... openat resumed>) = 4

[pid 52178] <... openat resumed>) = 4

[pid 52177] mmap(NULL, 1024, PROT_READ, MAP_SHARED, 4, 0 <unfinished ...>

[pid 52178] mmap(NULL, 1024, PROT_READ, MAP_SHARED, 4, 0 <unfinished ...>

[pid 52177] <... mmap resumed>) = 0x7e3bf2b93000

[pid 52178] <... mmap resumed>) = 0x79566af39000

[pid 52177] close(4 <unfinished ...>

[pid 52178] close(4 <unfinished ...>

[pid 52177] <... close resumed>) = 0

[pid 52178] <... close resumed>) = 0

[pid 52177] futex(0x7e3bf2bcd000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

[pid 52178] futex(0x79566af73000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY ok

<unfinished ...>

[pid 52104] <... read resumed>" ok\n", 1024) = 4

[pid 52104] write(1, "Parent: Sending to CHILD 1 (leng"..., 36Parent: Sending to CHILD 1 (length:) = 36

[pid 52104] write(1, "3", 13) = 1

[pid 52104] write(1, ")\n", 2)

) = 2

[pid 52104] futex(0x77e9d7e61000, FUTEX_WAKE, 1) = 1

[pid 52177] <... futex resumed>) = 0

[pid 52104] read(0, <unfinished ...>

[pid 52177] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=0, ...}, AT_EMPTY_PATH) = 0

[pid 52177] futex(0x7e3bf2bcd000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANYQUIT

<unfinished ...>

[pid 52104] <... read resumed>"QUIT\n", 1024) = 5

[pid 52104] futex(0x77e9d7e61000, FUTEX_WAKE, 1) = 1

[pid 52177] <... futex resumed>) = 0

[pid 52104] futex(0x77e9d7e5f000, FUTEX_WAKE, 1 <unfinished ...>

[pid 52177] munmap(0x7e3bf2b93000, 1024 <unfinished ...>

[pid 52104] <... futex resumed>) = 1

[pid 52178] <... futex resumed>) = 0

[pid 52104] wait4(52177, <unfinished ...>

[pid 52177] <... munmap resumed>) = 0

[pid 52178] munmap(0x79566af39000, 1024) = 0

[pid 52177] munmap(0x7e3bf2bcd000, 32 <unfinished ...>

[pid 52178] munmap(0x79566af73000, 32 <unfinished ...>

[pid 52177] <... munmap resumed>) = 0

[pid 52178] <... munmap resumed>) = 0

[pid 52178] close(3 <unfinished ...>

[pid 52177] write(3, " k\n", 3 <unfinished ...>

[pid 52178] <... close resumed>) = 0

[pid 52178] exit_group(0 <unfinished ...>

[pid 52177] <... write resumed>) = 3

[pid 52178] <... exit_group resumed>) = ?

[pid 52177] close(3) = 0

[pid 52178] +++ exited with 0 +++

[pid 52177] exit_group(0 <unfinished ...>

[pid 52104] <... wait4 resumed>NULL, 0, NULL) = ? ERESTARTSYS (To be restarted if SA_RESTART is set)

[pid 52177] <... exit_group resumed>) = ?

[pid 52104] --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=52178, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---

[pid 52177] +++ exited with 0 +++

--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=52177, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---

wait4(52177, NULL, 0, NULL) = 52177

wait4(52178, NULL, 0, NULL) = 52178

write(1, "Parent: \320\224\320\276\321\207\320\265\321\200\320\275\320\270\320\265\320\277\321\200\320\276\321"..., 62Parent: Дочерние процессы завершены.

) = 62

munmap(0x77e9d7e9b000, 1024) = 0

unlink("/dev/shm/ch1_52104_1763917285") = 0

munmap(0x77e9d7e61000, 32) = 0

unlink("/dev/shm/sem.ch1_52104_1763917285") = 0

munmap(0x77e9d7e60000, 1024) = 0

unlink("/dev/shm/ch2_52104_1763917285") = 0

```
munmap(0x77e9d7e5f000, 32)              = 0

unlink("/dev/shm/sem.ch2_52104_1763917285") = 0

write(1, "Parent: \320\222\321\201\320\265 IPC-
\321\200\320\265\321\201\321\203\321\200\321\201\321"..., 50Parent: Все IPC-ресурсы очищены.

) = 50

exit_group(0)                           = ?

+++ exited with 0 +++
```

Вывод

В ходе выполнения лабораторной работы были успешно освоены принципы организации межпроцессного взаимодействия (IPC) с использованием разделяемой памяти (Shared Memory) и именованных POSIX-семафоров. Основные трудности возникли при корректной инициализации и очистке создаваемых ресурсов (сегментов SHM и семафоров), а также при реализации механизма синхронизации для обеспечения упорядоченной передачи данных от родителя к соответствующему дочернему процессу. В процессе работы был детально изучен и применён набор системных вызовов: fork() для создания процессов, shm_open() и mmap() для организации и проецирования разделяемой памяти, sem_open(), sem_wait() и sem_post() для синхронизации, а также execlp() для запуска внешнего кода. Это позволило получить глубокое понимание механизмов асинхронной обработки данных в операционной системе, научиться корректно управлять жизненным циклом процессов и обеспечивать их безопасное взаимодействие без "гонки данных". Полученные навыки будут полезны для выполнения последующих, более сложных задач, требующих высокоскоростного обмена данными между независимыми процессами.