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-216БВ-24

Студент: Попов К.А.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 18.11.25

Москва, 2025

Постановка задачи

Вариант 21.

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 записывает их в разделяемую память (в памяти в каждый момент времени находится только одна строка), затем процессы child1 и child2 производят работу над строками согласно правилу фильтрации. Для синхронизации доступа к общим ресурсам используются семафоры. Процессы пишут результаты своей работы в стандартный вывод.

Правило фильтрации: нечетные строки обрабатываются одним дочерним процессом, четные - другим. Дочерние процессы инвертируют строки.

Общий метод и алгоритм решения

Ключевые функции (POSIX):

Разделяемая память:

- int shm_open(const char *name, int oflag, mode_t mode). Открывает сегмент разделяемой памяти. NAME - уникальное имя объекта разделяемой памяти в системе. OFLAG - флаги (O_RDWR - открыть сегмент для чтения и записи. O_CREAT - если объект разделяемой памяти с указанным именем не существует, создать его. O_TRUNC - размер объекта будет усечен до нуля).
- int ftruncate(int fd, off_t length). Устанавливает размер LENGTH объекта файловой системы с дескриптором FD.
- void *mmap(void *addr, size_t len, int prot, int flags, int fd, off_t offset). Отображает файл (файловый дескриптор FD) в адресное пространство процесса. ADDR - адрес для отображения (по желанию), LEN - размер области памяти для отображения, PROT - права доступа (чтение/запись/исполнение), FLAGS - тип отображения (MAP_SHARED - изменения будут видны другим процессам, MAP_PRIVATE - copy-on-write отображение). OFFSET - смещение относительно начала отображаемого объекта, должен быть кратен размеру страницы памяти. В случае успеха возвращает указатель на начало отображенной области памяти в адресном пространстве процесса.
- int munmap(void *addr, size_t len). Освобождает отображение памяти, начиная с ADDR и заканчивая LEN байтами. Возвращает 0 в случае успеха, -1 в случае ошибки (и устанавливает значение errno).
- int close(int fd) - закрывает файловый дескриптор FD.
- void *memcpy(void *restrict dest, const void *restrict src, size_t n). Copy N bytes of SRC to DEST. Копирует N байт из SRC в DST.

Семафоры:

- sem_t *sem_open(const char *name, int oflag, ...). Открывает или создает (если указан флаг O_CREAT) именованный семафор NAME с флагами открытия OFLAG.
- int sem_wait(sem_t *sem). Если значение семафора SEM положительное, оно уменьшается, и поток, вызвавший sem_wait(), продолжает выполнение. Если

значение семафора равно нулю, поток блокируется до тех пор, пока значение семафора не станет больше нуля.

- int sem_post(sem_t *sem). Увеличивает значение семафора.
- int sem_close(sem_t *sem). Закрывает объект именованного семафора по указателю SEM. Уменьшает счетчик ссылок на именованный семафор.
- int sem_unlink(const char *name). Удаляет именованный семафор NAME: когда счетчик ссылок на семафор достигнет нуля, семафор будет уничтожен операционной системой.

Для безопасного освобождения ресурсов:

- int atexit(void (*func)(void)). Регистрирует функцию FUNC, которая будет вызвана тогда, когда будет вызван exit().
- sighandler_t signal(int sig, sighandler_t handler). Устанавливает обработчик HANDLER для сигнала SIG. В случае успеха возвращает предыдущий обработчик сигнала SIG (можно восстановить предыдущее состояние обработки).

Работа со случайными числами:

- time_t time(time_t *timer). Возвращает текущее время и устанавливает его в TIMER, если TIMER не NULL.
- void srand(unsigned int seed). Инициализирует числом SEED генератор псевдослучайных чисел. Следует вызывать один раз в начале программы.

Использованные системные вызовы:

- ssize_t write(int fd, const void *buf, size_t n) - пишет N байт из BUF в дескриптор FD. Возвращает количество записанных байт или -1 в случае ошибки.
- ssize_t read(int fd, void *buf, size_t nbytes) - читает NBYTES байт, записывая в BUF из дескриптора FD. Возвращает число прочитанных байт, -1 в случае ошибки и 0 для EOF.
- pid_t fork(void) клонирует вызывающий процесс, создав его точную копию. Возвращает -1 в случае ошибки, 0 для нового процесса и pid нового процесса для старого процесса.
- int execv(const char *path, char *const *argv) - выполнить PATH с аргументами ARGV. В случае успеха происходит замена образа памяти процесса, иначе возвращает -1.
- pid_t getpid(void) - возвращает pid вызывающего процесса.
- pid_t waitpid(pid_t pid, int *stat_loc, int options) - ожидает завершения дочернего процесса с соответствующим pid.
- void exit(int status) - завершает выполнение процесса с возвращением соответствующего статуса.
- int dup2(int fd, int fd2) - дублирует FD в FD2, закрыв FD2 и открыв FD2 в том же файле, что и FD.
- int open(const char *pathname, int flags, mode_t mode) - открывает файл PATHNAME в режиме, определенном FLAGS и MODE. создание файла Возвращает файловый дескриптор.

Описание работы программы.

В рамках лабораторной работы были написаны две программы: client.c и server.c. client.c отвечает за логику работы родительского процесса и создание дочерних процессов, а server.c - за работу дочерних процессов. В файле config.h определяются данные, общие для клиента и сервера.

Программа (client) определяет свою директорию, принимает от пользователя пути к файлам относительно своей директории. В первый файл будут записаны результаты обработки нечетных строк, во второй - результаты обработки четных строк. Программа открывает данные файлы, именованный сегмент разделяемой памяти для обмена строками с дочерними процессами и три именованных семафора: «от клиента серверу №1», «от клиента серверу №2» и «от сервера клиенту». При генерации имен разделяемой памяти и семафоров используются случайные числа и текущее время, чтобы исключить совпадение с именами ресурсов других процессов. Затем с помощью fork создаются два дочерних процесса. Дочерние процессы получают копии всех дескрипторов родительского процесса. Стандартные потоки вывода дочерних процессов перенаправляются в соответствующие файлы. Все неиспользуемые дескрипторы закрываются. Управление в дочерних процессах передается коду в server.c с помощью execv. При этом имена общих ресурсов передаются дочерним процессам через аргументы командной строки.

Алгоритм синхронизации доступа к разделяемой памяти.

Разделяемая память делится на два участка: память под длину записанной в буфер строки (ssize_t) и сам буфер (массив char). Указатели на отображенную память приводятся к типу указателей на структуру shared_data_t.

При открытии всех семафоров («от клиента серверу №1», «от клиента серверу №2», «от сервера клиенту») их значения устанавливаются в ноль.

Сервер №1 ожидает семафор «от клиента серверу №1», а сервер №2 ожидает семафор «от клиента серверу №2». В течение этого клиент принимает строку от пользователя и записывает ее вместе с ее длиной в разделяемую память. Затем клиент увеличивает семафор «от клиента серверу №1» или «от клиента серверу №2» в зависимости от очереди, которую клиент хранит в своей локальной переменной, и ожидает семафор «от сервера клиенту». Тогда соответствующий сервер захватывает свой семафор, обрабатывает данные в разделяемой памяти и увеличивает семафор «от сервера клиенту». Клиент записывает новую строку в память, а оба сервера ждут, так как семафоры «от клиента серверу №№» захвачены. Таким образом соблюдается очередность между клиентом и серверами и очередность между сервером №1 и сервером №2.

При ошибке ввода или при вводе пустой строки (окончание ввода) клиент записывает ноль в качестве длины строки и освобождает семафоры. Дочерние процессы, обнаружив нулевую длину, освобождают семафоры и завершаются. Затем завершается и родительский процесс.

Для корректного освобождения ресурсов, в том числе при ошибке, соответствующая функция регистрируется в atexit(). Так как привязываемая функция не может принимать параметров, то информация о ресурсах (разделяемая память и семафоры) хранится в глобальной структуре RESOURCES. Функция очистки также привязывается к сигналу SIGINT, означающего прерывание процесса (например, с помощью Ctrl+C).

Код программы

config.h:

```
#ifndef CONFIG_H
#define CONFIG_H

#define USER_INPUT_BUFFERSIZE 4096 - sizeof(ssize_t)
#define NAME_BUFFERSIZE 128

typedef struct {
    ssize_t length;
    char text[USER_INPUT_BUFFERSIZE];
} shared_data_t;

#endif
```

client.c:

```
#include <stdint.h>
#include <stdbool.h>
#include <unistd.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/mman.h>
#include <semaphore.h>
#include <string.h>
#include <errno.h>
#include <time.h>
#include <signal.h>
#include "../include/config.h"

#define WORKDIR_BUFFERSIZE 1024
#define FILENAME_BUFFERSIZE 256
#define MSG_BUFFERSIZE 256

static char SERVER_PROGRAM_NAME[] = "server";

typedef struct {
    char sem_client_to_server_1_name[NAME_BUFFERSIZE];
    sem_t *sem_client_to_server_1;
    char sem_client_to_server_2_name[NAME_BUFFERSIZE];
    sem_t *sem_client_to_server_2;
    char sem_server_to_client_name[NAME_BUFFERSIZE];
    sem_t *sem_server_to_client;
    char *shm_buf;
    char shm_name[NAME_BUFFERSIZE];
    int shm_fd;
    bool sem_client_to_server_1_opened;
    bool sem_client_to_server_2_opened;
    bool sem_server_to_client_opened;
    bool shm_buf_mapped;
```

```
        bool shm_opened;
    } resources_t;

    void resources_init(resources_t *resources) {
        resources->sem_client_to_server_1_opened = false;
        resources->sem_client_to_server_2_opened = false;
        resources->sem_server_to_client_opened = false;
        resources->shm_buf_mapped = false;
        resources->shm_opened = false;
    }

    resources_t RESOURCES;

    int readFilename(char *filename, const ssize_t size) {
        ssize_t bytes_read = read(STDIN_FILENO, filename, size);
        if (bytes_read < 0) {
            return 1;
        }
        if (bytes_read == 0 || filename[0] == '\n') {
            return 2;
        }
        if (bytes_read == size && filename[size - 1] != '\n') {
            return 3;
        }
        if (filename[bytes_read - 1] == '\n') {
            filename[bytes_read - 1] = '\0';
        } else {
            filename[bytes_read] = '\0';
        }
        return 0;
    }

    void generate_name(char *res, size_t size, const char *pref, const char
*suf) {
        int random_n = rand();
        long long time_n = (long long)time(NULL);
        snprintf(res, size, "%s%ld_%d%s", pref, time_n, random_n, suf);
    }

    int open_resources(char *msg, size_t msg_size, int *res_msg_size) {
        generate_name(RESOURCES.shm_name, NAME_BUFFERSIZE, "shm_", "");
        generate_name(RESOURCES.sem_client_to_server_1_name, NAME_BUFFERSIZE,
"sem_client_to_server_1_", "");
        generate_name(RESOURCES.sem_client_to_server_2_name, NAME_BUFFERSIZE,
"sem_client_to_server_2_", "");
        generate_name(RESOURCES.sem_server_to_client_name, NAME_BUFFERSIZE,
"sem_server_to_client_", "");

        RESOURCES.shm_fd = shm_open(RESOURCES.shm_name, O_RDWR | O_CREAT |
O_TRUNC, 0600);
        if (RESOURCES.shm_fd == -1) {
            *res_msg_size = snprintf(msg, msg_size, "Error: Failed to open
shared memory\n");
            return -1;
```

```
        }
        RESOURCES.shm_opened = true;

        if (ftruncate(RESOURCES.shm_fd, sizeof(shared_data_t)) == -1) {
            *res_msg_size = snprintf(msg, msg_size, "Error: Failed to resize
shared memory\n");
            return -1;
        }

        RESOURCES.shm_buf = mmap(NULL, sizeof(shared_data_t), PROT_WRITE,
MAP_SHARED, RESOURCES.shm_fd, 0);
        if (RESOURCES.shm_buf == MAP_FAILED) {
            *res_msg_size = snprintf(msg, msg_size, "Error: Failed to map
shared memory\n");
            return -1;
        }
        RESOURCES.shm_buf_mapped = true;

        RESOURCES.sem_client_to_server_1 =
sem_open(RESOURCES.sem_client_to_server_1_name, O_RDWR | O_CREAT, 0600, 0);
        if (RESOURCES.sem_client_to_server_1 == SEM_FAILED) {
            *res_msg_size = snprintf(msg, msg_size, "Error: Failed to create
semaphore client_to_server_1\n");
            return -1;
        }
        RESOURCES.sem_client_to_server_1_opened = true;

        RESOURCES.sem_client_to_server_2 =
sem_open(RESOURCES.sem_client_to_server_2_name, O_RDWR | O_CREAT, 0600, 0);
        if (RESOURCES.sem_client_to_server_2 == SEM_FAILED) {
            *res_msg_size = snprintf(msg, msg_size, "Error: Failed to create
semaphore client_to_server_2\n");
            return -1;
        }
        RESOURCES.sem_client_to_server_2_opened = true;

        RESOURCES.sem_server_to_client =
sem_open(RESOURCES.sem_server_to_client_name, O_RDWR | O_CREAT, 0600, 0);
        if (RESOURCES.sem_server_to_client == SEM_FAILED) {
            *res_msg_size = snprintf(msg, msg_size, "Error: Failed to create
semaphore server_to_client\n");
            return -1;
        }
        RESOURCES.sem_server_to_client_opened = true;

        return 0;
    }

    void close_resources() {
        if (RESOURCES.sem_client_to_server_1_opened) {
            sem_close(RESOURCES.sem_client_to_server_1);
            RESOURCES.sem_client_to_server_1_opened = false;
        }
        sem_unlink(RESOURCES.sem_client_to_server_1_name);
```

```
        if (RESOURCES.sem_client_to_server_2_opened) {
            sem_close(RESOURCES.sem_client_to_server_2);
            RESOURCES.sem_client_to_server_2_opened = false;
        }
        sem_unlink(RESOURCES.sem_client_to_server_2_name);
        if (RESOURCES.sem_server_to_client_opened) {
            sem_close(RESOURCES.sem_server_to_client);
            RESOURCES.sem_server_to_client_opened = false;
        }
        sem_unlink(RESOURCES.sem_server_to_client_name);
        if (RESOURCES.shm_buf_mapped) {
            munmap(RESOURCES.shm_buf, sizeof(shared_data_t));
            RESOURCES.shm_buf_mapped = false;
        }
        if (RESOURCES.shm_opened) {
            close(RESOURCES.shm_fd);
            RESOURCES.shm_opened = false;
        }
        shm_unlink(RESOURCES.shm_name);
    }

    void safe_exit(int sig) {
        exit(0);
    }

    int main(int argc, char **argv) {
        srand(time(NULL));
        char workdir[WORKDIR_BUFFERSIZE];
        {
            ssize_t len = readlink("/proc/self/exe", workdir,
WORKDIR_BUFFERSIZE - 1);
            if (len == -1) {
                const char msg[] = "Error: Failed to read full program
path\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                exit(EXIT_FAILURE);
            }
            while (workdir[len] != '/') {
                --len;
            }
            workdir[len] = '\0';
        }

        char filename1[FILENAME_BUFFERSIZE];
        char filename2[FILENAME_BUFFERSIZE];
        {
            const char msg[] = "Enter first filename > ";
            write(STDOUT_FILENO, msg, sizeof(msg));
        }
        int errcode = readFilename(filename1, FILENAME_BUFFERSIZE);
        if (!errcode) {
            {
                const char msg[] = "Enter second filename > ";
                write(STDOUT_FILENO, msg, sizeof(msg));
```

```
            }
            errcode = readFilename(filename2, FILENAME_BUFFERSIZE);
        }
        switch (errcode) {
        case 1: {
            const char msg[] = "Error: Failed to read filename from stdin\n";
            write(STDERR_FILENO, msg, sizeof(msg));
        } exit(EXIT_FAILURE);
        case 2: {
            const char msg[] = "Error: No filename\n";
            write(STDERR_FILENO, msg, sizeof(msg));
        } exit(EXIT_FAILURE);
        case 3: {
            const char msg[] = "Error: Filename too long\n";
            write(STDERR_FILENO, msg, sizeof(msg));
        } exit(EXIT_FAILURE);
        }

        char filepath1[WORKDIR_BUFFERSIZE + FILENAME_BUFFERSIZE];
        snprintf(filepath1, WORKDIR_BUFFERSIZE + FILENAME_BUFFERSIZE,
"%s/%s", workdir, filename1);
        int32_t file1d = open(filepath1, O_WRONLY | O_CREAT | O_TRUNC, 0600);
        if (file1d == -1) {
            const char msg[] = "Error: Cannot open file_1\n";
            write(STDERR_FILENO, msg, sizeof(msg));
            exit(EXIT_FAILURE);
        }

        char filepath2[WORKDIR_BUFFERSIZE + FILENAME_BUFFERSIZE];
        snprintf(filepath2, WORKDIR_BUFFERSIZE + FILENAME_BUFFERSIZE,
"%s/%s", workdir, filename2);
        int32_t file2d = open(filepath2, O_WRONLY | O_CREAT | O_TRUNC, 0600);
        if (file2d == -1) {
            const char msg[] = "Error: Cannot open file_2\n";
            write(STDERR_FILENO, msg, sizeof(msg));
            exit(EXIT_FAILURE);
        }

        atexit(close_resources);
        signal(SIGINT, safe_exit);

        char msg[MSG_BUFFERSIZE];
        int msg_size;
        resources_init(&RESOURCES);
        if (open_resources(msg, MSG_BUFFERSIZE, &msg_size)) {
            write(STDERR_FILENO, msg, msg_size);
            exit(EXIT_FAILURE);
        }

        const pid_t server1_pid = fork();
        switch (server1_pid) {
        case -1: {
            const char msg[] = "Error: Failed to spawn new process
server_1\n";
```

```
            write(STDERR_FILENO, msg, sizeof(msg));
        } exit(EXIT_FAILURE);
        case 0: {
            {
                const pid_t pid = getpid();
                char msg[64];
                const int length = snprintf(msg, sizeof(msg), "PID %d: I'm a
server_1\n", pid);
                write(STDOUT_FILENO, msg, length);
            }
            dup2(file1d, STDOUT_FILENO);
            close(file1d);
            close(file2d);

            char serverpath[WORKDIR_BUFFERSIZE + FILENAME_BUFFERSIZE];
            snprintf(serverpath, sizeof(serverpath), "%s/%s", workdir,
SERVER_PROGRAM_NAME);
            char *const args[] = {
                SERVER_PROGRAM_NAME,
                RESOURCES.shm_name,
                RESOURCES.sem_client_to_server_1_name,
                RESOURCES.sem_server_to_client_name,
                NULL
            };
            int32_t status = execv(serverpath, args);
            if (status == -1) {
                const char msg[] = "Error: Failed to exec into new
exectuable image\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                exit(EXIT_FAILURE);
            }
        }
        }

        const pid_t server2_pid = fork();
        switch (server2_pid) {
        case -1: {
            const char msg[] = "Error: Failed to spawn new process
server_2\n";
            write(STDERR_FILENO, msg, sizeof(msg));
        } exit(EXIT_FAILURE);
        case 0: {
            {
                const pid_t pid = getpid();
                char msg[64];
                const int length = snprintf(msg, sizeof(msg), "PID %d: I'm a
server_2\n", pid);
                write(STDOUT_FILENO, msg, length);
            }
            dup2(file2d, STDOUT_FILENO);
            close(file2d);
            close(file1d);

            char serverpath[WORKDIR_BUFFERSIZE + FILENAME_BUFFERSIZE];
```

```
            snprintf(serverpath, sizeof(serverpath), "%s/%s", workdir,
SERVER_PROGRAM_NAME);
            char *const args[] = {
                SERVER_PROGRAM_NAME,
                RESOURCES.shm_name,
                RESOURCES.sem_client_to_server_2_name,
                RESOURCES.sem_server_to_client_name,
                NULL
            };
            int32_t status = execv(serverpath, args);
            if (status == -1) {
                const char msg[] = "Error: Failed to exec into new
exectuable image\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                exit(EXIT_FAILURE);
            }
        }
        }
        {
            const pid_t pid = getpid();
            char msg[128];
            const int length = snprintf(msg, sizeof(msg),
                "PID %d: I'm a parent, my children has PID %d and %d\n", pid,
server1_pid, server2_pid);
            write(STDOUT_FILENO, msg, length);
        }
        close(file1d);
        close(file2d);

        shared_data_t *data = (shared_data_t *)RESOURCES.shm_buf;
        char buf[USER_INPUT_BUFFERSIZE];
        ssize_t bytes_read;
        bool oddLine = true;
        while (bytes_read = read(STDIN_FILENO, buf, USER_INPUT_BUFFERSIZE))
{
            bool finish_input = buf[0] == '\n';
            bool failed_read = bytes_read < 0;
            bool overflow_input = bytes_read == USER_INPUT_BUFFERSIZE &&
buf[bytes_read - 1] != '\n';
            if (finish_input || failed_read || overflow_input) {
                data->length = 0;
                sem_post(RESOURCES.sem_client_to_server_1);
                sem_post(RESOURCES.sem_client_to_server_2);
                if (failed_read || overflow_input) {
                    char msg[MSG_BUFFERSIZE];
                    int msg_size;
                    if (failed_read) {
                        msg_size = snprintf(msg, MSG_BUFFERSIZE, "Error:
Failed to read from stdin (parent)\n");
                    } else if (overflow_input) {
                        msg_size = snprintf(msg, MSG_BUFFERSIZE, "Error:
Received line too long\n");
                    }
                    write(STDERR_FILENO, msg, msg_size);
```

```
                exit(EXIT_FAILURE);
            }
            break;
        }
        data->length = bytes_read;
        memcpy(data->text, buf, bytes_read);
        if (oddLine) {
            sem_post(RESOURCES.sem_client_to_server_1);
        } else {
            sem_post(RESOURCES.sem_client_to_server_2);
        }
        sem_wait(RESOURCES.sem_server_to_client);
        oddLine = !oddLine;
    }
    waitpid(server1_pid, NULL, 0);
    waitpid(server2_pid, NULL, 0);
    {
        const char msg[] = "Parent exit successfully\n";
        write(STDOUT_FILENO, msg, sizeof(msg));
    }
    return 0;
}
```

server.c:

```
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdint.h>
#include <stdbool.h>
#include <sys/mman.h>
#include <semaphore.h>
#include <signal.h>

#include "../include/config.h"

#define MSG_BUFFERSIZE 256

enum process_data_error {
    NO_DATA_ERROR = 1,
    WRITE_ERROR,
};

typedef struct {
    char sem_client_to_server_name[NAME_BUFFERSIZE];
    sem_t *sem_client_to_server;
    char sem_server_to_client_name[NAME_BUFFERSIZE];
    sem_t *sem_server_to_client;
    char *shm_buf;
    char shm_name[NAME_BUFFERSIZE];
    int shm_fd;
    bool shm_opened;
    bool shm_buf_mapped;
    bool sem_client_to_server_opened;
```

```
        bool sem_server_to_client_opened;
    } resources_t;

    resources_t RESOURCES;

    void resources_init(resources_t *resources) {
        resources->shm_opened = false;
        resources->shm_buf_mapped = false;
        resources->sem_client_to_server_opened = false;
        resources->sem_server_to_client_opened = false;
    }

    int open_resources(char *msg, size_t msg_size, int *res_msg_size) {
        RESOURCES.shm_fd = shm_open(RESOURCES.shm_name, O_RDWR, 0);
        if (RESOURCES.shm_fd == -1) {
            *res_msg_size = snprintf(msg, msg_size, "Error: Failed to open
SHM\n");
            return -1;
        }
        RESOURCES.shm_opened = true;

        RESOURCES.shm_buf = mmap(NULL, sizeof(shared_data_t),
            PROT_READ | PROT_WRITE, MAP_SHARED, RESOURCES.shm_fd, 0);
        if (RESOURCES.shm_buf == MAP_FAILED) {
            *res_msg_size = snprintf(msg, msg_size, "Error: Failed to map
SHM\n");
            return -1;
        }
        RESOURCES.shm_buf_mapped = true;

        RESOURCES.sem_client_to_server =
sem_open(RESOURCES.sem_client_to_server_name, O_RDWR);
        if (RESOURCES.sem_client_to_server == SEM_FAILED) {
            *res_msg_size = snprintf(msg, msg_size, "Error: Failed to open
semaphore\n");
            return -1;
        }
        RESOURCES.sem_client_to_server_opened = true;

        RESOURCES.sem_server_to_client =
sem_open(RESOURCES.sem_server_to_client_name, O_RDWR);
        if (RESOURCES.sem_server_to_client == SEM_FAILED) {
            *res_msg_size = snprintf(msg, msg_size, "Error: Failed to open
semaphore\n");
            return -1;
        }
        RESOURCES.sem_server_to_client_opened = true;

        return 0;
    }

    void close_resources() {
        if (RESOURCES.sem_server_to_client_opened) {
            sem_close(RESOURCES.sem_server_to_client);
```

```
            RESOURCES.sem_server_to_client_opened = false;
        }
        if (RESOURCES.sem_client_to_server_opened) {
            sem_close(RESOURCES.sem_client_to_server);
            RESOURCES.sem_client_to_server_opened = false;
        }
        if (RESOURCES.shm_buf_mapped) {
            munmap(RESOURCES.shm_buf, sizeof(shared_data_t));
            RESOURCES.shm_buf_mapped = false;
        }
        if (RESOURCES.shm_opened) {
            close(RESOURCES.shm_fd);
            RESOURCES.shm_opened = false;
        }
    }

    void safe_exit(int sig) {
        exit(0);
    }

    int process_data(shared_data_t *data) {
        if (data->length == 0) {
            return NO_DATA_ERROR;
        }
        ssize_t j = data->text[data->length - 1] == '\n' ? data->length - 2 :
data->length - 1;
        for (ssize_t i = 0; i < j; ++i, --j) {
            char tmp = data->text[i];
            data->text[i] = data->text[j];
            data->text[j] = tmp;
        }
        ssize_t bytes_written = write(STDOUT_FILENO, data->text, data-
>length);
        if (bytes_written != data->length) {
            return WRITE_ERROR;
        }
        return 0;
    }

    int main(int argc, char **argv) {
        const pid_t pid = getpid();

        snprintf(RESOURCES.shm_name, NAME_BUFFERSIZE, "%s", argv[1]);
        snprintf(RESOURCES.sem_client_to_server_name, NAME_BUFFERSIZE, "%s",
argv[2]);
        snprintf(RESOURCES.sem_server_to_client_name, NAME_BUFFERSIZE, "%s",
argv[3]);

        atexit(close_resources);
        signal(SIGINT, safe_exit);

        char msg[MSG_BUFFERSIZE];
        int msg_size;
        resources_init(&RESOURCES);
```

```
    if (open_resources(msg, MSG_BUFFERSIZE, &msg_size)) {
        write(STDERR_FILENO, msg, MSG_BUFFERSIZE);
        exit(EXIT_FAILURE);
    }

    shared_data_t *data = (shared_data_t *)RESOURCES.shm_buf;
    int error = 0;
    bool running = true;
    while (running) {
        sem_wait(RESOURCES.sem_client_to_server);
        error = process_data(data);
        if (error) {
            running = false;
        }
        sem_post(RESOURCES.sem_server_to_client);
    }
    if (error == WRITE_ERROR) {
        char msg[MSG_BUFFERSIZE];
        const int msg_size = snprintf(msg, MSG_BUFFERSIZE,
            "Error: Failed to write to stdout (PID: %d)\n", pid);
        write(STDERR_FILENO, msg, msg_size);
        exit(EXIT_FAILURE);
    }
    {
        char msg[MSG_BUFFERSIZE];
        const int msg_size = snprintf(msg, MSG_BUFFERSIZE,
            "Child PID %d exit successfully\n", pid);
        write(STDERR_FILENO, msg, msg_size);
    }
    return 0;
}
```

Протокол работы программы

Тестирование:

```
root@4bcd85e1faed:/workspaces/OS_labs/lab3/build# ./client
Enter first filename > ../tests/f1.txt
Enter second filename > ../tests/f2.txt
PID 53463: I'm a server_1
PID 53406: I'm a parent, my children has PID 53463 and 53464
PID 53464: I'm a server_2
string1
string2
string with spaces
qwerty
123456789

Child PID 53463 exit successfully
Child PID 53464 exit successfully
Parent exit successfully

root@4bcd85e1faed:/workspaces/OS_labs/lab3/build# cat < ../tests/f1.txt
1gnirts
secaps htiw gnirts
```

```
987654321

root@4bcd85e1faed:/workspaces/OS_labs/lab3/build# cat < ../tests/f2.txt
2gnirts
ytrewq
```

Strace:

```
root@4bcd85e1faed:/workspaces/OS_labs/lab3/build# strace -
o ../tests/strace_out.txt -f ./client
```

strace_out.txt:

```
4409  execve("./client", ["./client"], 0x7ffeb80a6c38 /* 27 vars */) = 0
4409  brk(NULL)                         = 0x18036000
4409  mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
-1, 0) = 0x73029f25d000
4409  access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or
directory)
4409  openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
4409  newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=33091, ...},
AT_EMPTY_PATH) = 0
4409  mmap(NULL, 33091, PROT_READ, MAP_PRIVATE, 3, 0) = 0x73029f254000
4409  close(3)                          = 0
4409  openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6",
O_RDONLY|O_CLOEXEC) = 3
4409  read(3,
"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20t\2\0\0\0\0\0"..., 832) =
832
4409  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
4409  newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=1926232, ...},
AT_EMPTY_PATH) = 0
4409  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
4409  mmap(NULL, 1974096, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x73029f072000
4409  mmap(0x73029f098000, 1400832, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x26000) = 0x73029f098000
4409  mmap(0x73029f1ee000, 339968, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17c000) = 0x73029f1ee000
4409  mmap(0x73029f241000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1cf000) = 0x73029f241000
4409  mmap(0x73029f247000, 53072, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x73029f247000
4409  close(3)                          = 0
4409  mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
-1, 0) = 0x73029f06f000
4409  arch_prctl(ARCH_SET_FS, 0x73029f06f740) = 0
4409  set_tid_address(0x73029f06fa10)   = 4409
4409  set_robust_list(0x73029f06fa20, 24) = 0
4409  rseq(0x73029f070060, 0x20, 0, 0x53053053) = 0
4409  mprotect(0x73029f241000, 16384, PROT_READ) = 0
```

```
    4409  mprotect(0x404000, 4096, PROT_READ) = 0
    4409  mprotect(0x73029f290000, 8192, PROT_READ) = 0
    4409  prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
    4409  munmap(0x73029f254000, 33091)     = 0
    4409  readlink("/proc/self/exe", "/workspaces/OS_labs/lab3/build/c"...,
1023) = 37
    4409  write(1, "Enter first filename > \0", 24) = 24
    4409  read(0, "../tests/f1.txt\n", 256) = 16
    4409  write(1, "Enter second filename > \0", 25) = 25
    4409  read(0, "../tests/f2.txt\n", 256) = 16
    4409  openat(AT_FDCWD, "/workspaces/OS_labs/lab3/build/../tests/f1.txt",
O_WRONLY|O_CREAT|O_TRUNC, 0600) = 3
    4409  openat(AT_FDCWD, "/workspaces/OS_labs/lab3/build/../tests/f2.txt",
O_WRONLY|O_CREAT|O_TRUNC, 0600) = 4
    4409  rt_sigaction(SIGINT, {sa_handler=0x4017c4, sa_mask=[INT],
sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x73029f0ae050},
{sa_handler=SIG_DFL, sa_mask=[], sa_flags=0}, 8) = 0
    4409  openat(AT_FDCWD, "/dev/shm/shm_1763446038_1983967848",
O_RDWR|O_CREAT|O_TRUNC|O_NOFOLLOW|O_CLOEXEC, 0600) = 5
    4409  ftruncate(5, 4096)                = 0
    4409  mmap(NULL, 4096, PROT_WRITE, MAP_SHARED, 5, 0) = 0x73029f25c000
    4409  openat(AT_FDCWD,
"/dev/shm/sem.sem_client_to_server_1_1763446038_807639635",
O_RDWR|O_NOFOLLOW) = -1 ENOENT (No such file or directory)
    4409  getrandom("\x3f\x73\x01\x1d\x1a\x10\x96\xbe", 8, GRND_NONBLOCK) =
8
    4409  newfstatat(AT_FDCWD, "/dev/shm/sem.Xh93co", 0x7ffc6943c440,
AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)
    4409  openat(AT_FDCWD, "/dev/shm/sem.Xh93co", O_RDWR|O_CREAT|O_EXCL,
0600) = 6
    4409  write(6,
"\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
    4409  mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 6, 0) =
0x73029f25b000
    4409  link("/dev/shm/sem.Xh93co",
"/dev/shm/sem.sem_client_to_server_1_1763446038_807639635") = 0
    4409  newfstatat(6, "", {st_mode=S_IFREG|0600, st_size=32, ...},
AT_EMPTY_PATH) = 0
    4409  getrandom("\xe9\xf7\x1c\xf9\x40\xcf\xc2\x06", 8, GRND_NONBLOCK) =
8
    4409  brk(NULL)                         = 0x18036000
    4409  brk(0x18057000)                   = 0x18057000
    4409  unlink("/dev/shm/sem.Xh93co")     = 0
    4409  close(6)                          = 0
    4409  openat(AT_FDCWD,
"/dev/shm/sem.sem_client_to_server_2_1763446038_1802162044",
O_RDWR|O_NOFOLLOW) = -1 ENOENT (No such file or directory)
    4409  getrandom("\x09\xd3\xdb\x0a\xb8\x74\x55\x6b", 8, GRND_NONBLOCK) =
8
    4409  newfstatat(AT_FDCWD, "/dev/shm/sem.l1oGJM", 0x7ffc6943c440,
AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)
```

```
    4409  openat(AT_FDCWD, "/dev/shm/sem.l1oGJM", O_RDWR|O_CREAT|O_EXCL,
0600) = 6
    4409  write(6,
"\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
    4409  mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 6, 0) =
0x73029f25a000
    4409  link("/dev/shm/sem.l1oGJM",
"/dev/shm/sem.sem_client_to_server_2_1763446038_1802162044") = 0
    4409  newfstatat(6, "", {st_mode=S_IFREG|0600, st_size=32, ...},
AT_EMPTY_PATH) = 0
    4409  unlink("/dev/shm/sem.l1oGJM")     = 0
    4409  close(6)                          = 0
    4409  openat(AT_FDCWD,
"/dev/shm/sem.sem_server_to_client_1763446038_54883658", O_RDWR|O_NOFOLLOW)
= -1 ENOENT (No such file or directory)
    4409  getrandom("\x82\x93\x6f\xa1\x2b\x06\x76\xce", 8, GRND_NONBLOCK) =
8
    4409  newfstatat(AT_FDCWD, "/dev/shm/sem.6Q5vgY", 0x7ffc6943c440,
AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)
    4409  openat(AT_FDCWD, "/dev/shm/sem.6Q5vgY", O_RDWR|O_CREAT|O_EXCL,
0600) = 6
    4409  write(6,
"\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
    4409  mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 6, 0) =
0x73029f259000
    4409  link("/dev/shm/sem.6Q5vgY",
"/dev/shm/sem.sem_server_to_client_1763446038_54883658") = 0
    4409  newfstatat(6, "", {st_mode=S_IFREG|0600, st_size=32, ...},
AT_EMPTY_PATH) = 0
    4409  unlink("/dev/shm/sem.6Q5vgY")     = 0
    4409  close(6)                          = 0
    4409  clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x73029f06fa10) = 4468
    4468  set_robust_list(0x73029f06fa20, 24 <unfinished ...>
    4409  clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD <unfinished ...>
    4468  <... set_robust_list resumed>)    = 0
    4409  <... clone resumed>, child_tidptr=0x73029f06fa10) = 4469
    4469  set_robust_list(0x73029f06fa20, 24 <unfinished ...>
    4468  getpid( <unfinished ...>
    4409  getpid( <unfinished ...>
    4469  <... set_robust_list resumed>)    = 0
    4409  <... getpid resumed>)             = 4409
    4468  <... getpid resumed>)             = 4468
    4409  write(1, "PID 4409: I'm a parent, my child"..., 58
<unfinished ...>
    4469  getpid( <unfinished ...>
    4409  <... write resumed>)              = 58
    4468  write(1, "PID 4468: I'm a server_1\n", 25 <unfinished ...>
    4409  close(3 <unfinished ...>
    4469  <... getpid resumed>)             = 4469
```

```
    4409  <... close resumed>)              = 0
    4468  <... write resumed>)              = 25
    4409  close(4 <unfinished ...>
    4469  write(1, "PID 4469: I'm a server_2\n", 25 <unfinished ...>
    4409  <... close resumed>)              = 0
    4468  dup2(3, 1 <unfinished ...>
    4409  read(0,  <unfinished ...>
    4469  <... write resumed>)              = 25
    4468  <... dup2 resumed>)               = 1
    4469  dup2(4, 1 <unfinished ...>
    4468  close(3 <unfinished ...>
    4469  <... dup2 resumed>)               = 1
    4468  <... close resumed>)              = 0
    4469  close(4 <unfinished ...>
    4468  close(4 <unfinished ...>
    4469  <... close resumed>)              = 0
    4468  <... close resumed>)              = 0
    4469  close(3 <unfinished ...>
    4468  execve("/workspaces/OS_labs/lab3/build/server", ["server",
"shm_1763446038_1983967848", "sem_client_to_server_1_176344603"...,
"sem_server_to_client_1763446038_"...], 0x7ffc6943f0c8 /* 27 vars */
<unfinished ...>
    4469  <... close resumed>)              = 0
    4469  execve("/workspaces/OS_labs/lab3/build/server", ["server",
"shm_1763446038_1983967848", "sem_client_to_server_2_176344603"...,
"sem_server_to_client_1763446038_"...], 0x7ffc6943f0c8 /* 27 vars */) = 0
    4468  <... execve resumed>)             = 0
    4469  brk(NULL <unfinished ...>
    4468  brk(NULL <unfinished ...>
    4469  <... brk resumed>)                = 0x37264000
    4468  <... brk resumed>)                = 0x1c277000
    4469  mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
-1, 0 <unfinished ...>
    4468  mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
-1, 0 <unfinished ...>
    4469  <... mmap resumed>)               = 0x7e98ea6b0000
    4468  <... mmap resumed>)               = 0x74b3bad7f000
    4469  access("/etc/ld.so.preload", R_OK <unfinished ...>
    4468  access("/etc/ld.so.preload", R_OK <unfinished ...>
    4469  <... access resumed>)             = -1 ENOENT (No such file or
directory)
    4468  <... access resumed>)             = -1 ENOENT (No such file or
directory)
    4469  openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC
<unfinished ...>
    4468  openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC
<unfinished ...>
    4469  <... openat resumed>)             = 3
    4468  <... openat resumed>)             = 3
    4469  newfstatat(3, "",  <unfinished ...>
    4468  newfstatat(3, "",  <unfinished ...>
    4469  <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=33091, ...},
AT_EMPTY_PATH) = 0
```

```
    4468  <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=33091, ...},
AT_EMPTY_PATH) = 0
    4469  mmap(NULL, 33091, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>
    4468  mmap(NULL, 33091, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>
    4469  <... mmap resumed>)               = 0x7e98ea6a7000
    4468  <... mmap resumed>)               = 0x74b3bad76000
    4469  close(3 <unfinished ...>
    4468  close(3 <unfinished ...>
    4469  <... close resumed>)              = 0
    4468  <... close resumed>)              = 0
    4469  openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6",
O_RDONLY|O_CLOEXEC <unfinished ...>
    4468  openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6",
O_RDONLY|O_CLOEXEC <unfinished ...>
    4469  <... openat resumed>)             = 3
    4468  <... openat resumed>)             = 3
    4469  read(3,  <unfinished ...>
    4468  read(3,  <unfinished ...>
    4469  <... read
resumed>"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20t\2\0\0\0\0\0"...,
832) = 832
    4468  <... read
resumed>"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20t\2\0\0\0\0\0"...,
832) = 832
    4469  pread64(3,  <unfinished ...>
    4468  pread64(3,  <unfinished ...>
    4469  <... 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
    4468  <... 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
    4469  newfstatat(3, "",  <unfinished ...>
    4468  newfstatat(3, "",  <unfinished ...>
    4469  <... newfstatat resumed>{st_mode=S_IFREG|0755,
st_size=1926232, ...}, AT_EMPTY_PATH) = 0
    4468  <... newfstatat resumed>{st_mode=S_IFREG|0755,
st_size=1926232, ...}, AT_EMPTY_PATH) = 0
    4469  pread64(3,  <unfinished ...>
    4468  pread64(3,  <unfinished ...>
    4469  <... 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
    4468  <... 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
    4469  mmap(NULL, 1974096, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0
<unfinished ...>
    4468  mmap(NULL, 1974096, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0
<unfinished ...>
    4469  <... mmap resumed>)               = 0x7e98ea4c5000
    4468  <... mmap resumed>)               = 0x74b3bab94000
    4469  mmap(0x7e98ea4eb000, 1400832, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x26000 <unfinished ...>
```

```
    4468  mmap(0x74b3babba000, 1400832, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x26000 <unfinished ...>
    4469  <... mmap resumed>)               = 0x7e98ea4eb000
    4468  <... mmap resumed>)               = 0x74b3babba000
    4469  mmap(0x7e98ea641000, 339968, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17c000 <unfinished ...>
    4468  mmap(0x74b3bad10000, 339968, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17c000 <unfinished ...>
    4469  <... mmap resumed>)               = 0x7e98ea641000
    4468  <... mmap resumed>)               = 0x74b3bad10000
    4469  mmap(0x7e98ea694000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1cf000 <unfinished ...>
    4468  mmap(0x74b3bad63000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1cf000 <unfinished ...>
    4469  <... mmap resumed>)               = 0x7e98ea694000
    4468  <... mmap resumed>)               = 0x74b3bad63000
    4469  mmap(0x7e98ea69a000, 53072, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
    4468  mmap(0x74b3bad69000, 53072, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
    4469  <... mmap resumed>)               = 0x7e98ea69a000
    4468  <... mmap resumed>)               = 0x74b3bad69000
    4469  close(3 <unfinished ...>
    4468  close(3 <unfinished ...>
    4469  <... close resumed>)              = 0
    4468  <... close resumed>)              = 0
    4469  mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
-1, 0 <unfinished ...>
    4468  mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
-1, 0 <unfinished ...>
    4469  <... mmap resumed>)               = 0x7e98ea4c2000
    4468  <... mmap resumed>)               = 0x74b3bab91000
    4469  arch_prctl(ARCH_SET_FS, 0x7e98ea4c2740 <unfinished ...>
    4468  arch_prctl(ARCH_SET_FS, 0x74b3bab91740 <unfinished ...>
    4469  <... arch_prctl resumed>)         = 0
    4468  <... arch_prctl resumed>)         = 0
    4469  set_tid_address(0x7e98ea4c2a10 <unfinished ...>
    4468  set_tid_address(0x74b3bab91a10 <unfinished ...>
    4469  <... set_tid_address resumed>)    = 4469
    4468  <... set_tid_address resumed>)    = 4468
    4469  set_robust_list(0x7e98ea4c2a20, 24 <unfinished ...>
    4468  set_robust_list(0x74b3bab91a20, 24 <unfinished ...>
    4469  <... set_robust_list resumed>)    = 0
    4468  <... set_robust_list resumed>)    = 0
    4469  rseq(0x7e98ea4c3060, 0x20, 0, 0x53053053 <unfinished ...>
    4468  rseq(0x74b3bab92060, 0x20, 0, 0x53053053 <unfinished ...>
    4469  <... rseq resumed>)               = 0
    4468  <... rseq resumed>)               = 0
    4469  mprotect(0x7e98ea694000, 16384, PROT_READ <unfinished ...>
    4468  mprotect(0x74b3bad63000, 16384, PROT_READ <unfinished ...>
    4469  <... mprotect resumed>)           = 0
    4468  <... mprotect resumed>)           = 0
    4469  mprotect(0x403000, 4096, PROT_READ <unfinished ...>
    4468  mprotect(0x403000, 4096, PROT_READ <unfinished ...>
```

```
    4469  <... mprotect resumed>)           = 0
    4468  <... mprotect resumed>)           = 0
    4469  mprotect(0x7e98ea6e3000, 8192, PROT_READ <unfinished ...>
    4468  mprotect(0x74b3badb2000, 8192, PROT_READ <unfinished ...>
    4469  <... mprotect resumed>)           = 0
    4468  <... mprotect resumed>)           = 0
    4469  prlimit64(0, RLIMIT_STACK, NULL,  <unfinished ...>
    4468  prlimit64(0, RLIMIT_STACK, NULL,  <unfinished ...>
    4469  <... prlimit64 resumed>{rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
    4468  <... prlimit64 resumed>{rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
    4469  munmap(0x7e98ea6a7000, 33091 <unfinished ...>
    4468  munmap(0x74b3bad76000, 33091 <unfinished ...>
    4469  <... munmap resumed>)             = 0
    4468  <... munmap resumed>)             = 0
    4469  getpid( <unfinished ...>
    4468  getpid( <unfinished ...>
    4469  <... getpid resumed>)             = 4469
    4468  <... getpid resumed>)             = 4468
    4469  rt_sigaction(SIGINT, {sa_handler=0x40144f, sa_mask=[INT],
sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7e98ea501050},
<unfinished ...>
    4468  rt_sigaction(SIGINT, {sa_handler=0x40144f, sa_mask=[INT],
sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x74b3babd0050},
<unfinished ...>
    4469  <... rt_sigaction resumed>{sa_handler=SIG_DFL, sa_mask=[],
sa_flags=0}, 8) = 0
    4468  <... rt_sigaction resumed>{sa_handler=SIG_DFL, sa_mask=[],
sa_flags=0}, 8) = 0
    4469  openat(AT_FDCWD, "/dev/shm/shm_1763446038_1983967848",
O_RDWR|O_NOFOLLOW|O_CLOEXEC <unfinished ...>
    4468  openat(AT_FDCWD, "/dev/shm/shm_1763446038_1983967848",
O_RDWR|O_NOFOLLOW|O_CLOEXEC <unfinished ...>
    4469  <... openat resumed>)             = 3
    4468  <... openat resumed>)             = 3
    4469  mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0
<unfinished ...>
    4468  mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0
<unfinished ...>
    4469  <... mmap resumed>)               = 0x7e98ea6af000
    4468  <... mmap resumed>)               = 0x74b3bad7e000
    4469  openat(AT_FDCWD,
"/dev/shm/sem.sem_client_to_server_2_1763446038_1802162044",
O_RDWR|O_NOFOLLOW <unfinished ...>
    4468  openat(AT_FDCWD,
"/dev/shm/sem.sem_client_to_server_1_1763446038_807639635",
O_RDWR|O_NOFOLLOW <unfinished ...>
    4469  <... openat resumed>)             = 4
    4468  <... openat resumed>)             = 4
    4469  newfstatat(4, "",  <unfinished ...>
    4468  newfstatat(4, "",  <unfinished ...>
    4469  <... newfstatat resumed>{st_mode=S_IFREG|0600, st_size=32, ...},
AT_EMPTY_PATH) = 0
```

```
    4468  <... newfstatat resumed>{st_mode=S_IFREG|0600, st_size=32, ...},
AT_EMPTY_PATH) = 0
    4469  getrandom( <unfinished ...>
    4468  getrandom( <unfinished ...>
    4469  <... getrandom resumed>"\xb7\xe6\x40\xc0\x32\xe4\x20\x73", 8,
GRND_NONBLOCK) = 8
    4468  <... getrandom resumed>"\xee\x4a\xf2\x31\x81\x91\xf8\xdf", 8,
GRND_NONBLOCK) = 8
    4469  brk(NULL <unfinished ...>
    4468  brk(NULL <unfinished ...>
    4469  <... brk resumed>)                = 0x37264000
    4468  <... brk resumed>)                = 0x1c277000
    4469  brk(0x37285000 <unfinished ...>
    4468  brk(0x1c298000 <unfinished ...>
    4469  <... brk resumed>)                = 0x37285000
    4468  <... brk resumed>)                = 0x1c298000
    4469  mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0
<unfinished ...>
    4468  mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0
<unfinished ...>
    4469  <... mmap resumed>)               = 0x7e98ea6ae000
    4468  <... mmap resumed>)               = 0x74b3bad7d000
    4469  close(4 <unfinished ...>
    4468  close(4 <unfinished ...>
    4469  <... close resumed>)              = 0
    4468  <... close resumed>)              = 0
    4469  openat(AT_FDCWD,
"/dev/shm/sem.sem_server_to_client_1763446038_54883658", O_RDWR|O_NOFOLLOW
<unfinished ...>
    4468  openat(AT_FDCWD,
"/dev/shm/sem.sem_server_to_client_1763446038_54883658", O_RDWR|O_NOFOLLOW
<unfinished ...>
    4469  <... openat resumed>)             = 4
    4468  <... openat resumed>)             = 4
    4469  newfstatat(4, "",  <unfinished ...>
    4468  newfstatat(4, "",  <unfinished ...>
    4469  <... newfstatat resumed>{st_mode=S_IFREG|0600, st_size=32, ...},
AT_EMPTY_PATH) = 0
    4468  <... newfstatat resumed>{st_mode=S_IFREG|0600, st_size=32, ...},
AT_EMPTY_PATH) = 0
    4469  mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0
<unfinished ...>
    4468  mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0
<unfinished ...>
    4469  <... mmap resumed>)               = 0x7e98ea6ad000
    4468  <... mmap resumed>)               = 0x74b3bad7c000
    4469  close(4 <unfinished ...>
    4468  close(4 <unfinished ...>
    4469  <... close resumed>)              = 0
    4468  <... close resumed>)              = 0
    4469  futex(0x7e98ea6ae000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
    4468  futex(0x74b3bad7d000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
```

```
    4409  <... read resumed>"string1\n", 4088) = 8
    4409  futex(0x73029f25b000, FUTEX_WAKE, 1) = 1
    4468  <... futex resumed>)              = 0
    4409  futex(0x73029f259000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
    4468  write(1, "1gnirts\n", 8)          = 8
    4468  futex(0x74b3bad7c000, FUTEX_WAKE, 1 <unfinished ...>
    4409  <... futex resumed>)              = 0
    4468  <... futex resumed>)              = 1
    4409  read(0,  <unfinished ...>
    4468  futex(0x74b3bad7d000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
    4409  <... read resumed>"string2\n", 4088) = 8
    4409  futex(0x73029f25a000, FUTEX_WAKE, 1) = 1
    4469  <... futex resumed>)              = 0
    4409  futex(0x73029f259000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
    4469  write(1, "2gnirts\n", 8)          = 8
    4469  futex(0x7e98ea6ad000, FUTEX_WAKE, 1 <unfinished ...>
    4409  <... futex resumed>)              = 0
    4469  <... futex resumed>)              = 1
    4409  read(0,  <unfinished ...>
    4469  futex(0x7e98ea6ae000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
    4409  <... read resumed>"string with spaces\n", 4088) = 19
    4409  futex(0x73029f25b000, FUTEX_WAKE, 1) = 1
    4468  <... futex resumed>)              = 0
    4409  futex(0x73029f259000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
    4468  write(1, "secaps htiw gnirts\n", 19) = 19
    4468  futex(0x74b3bad7c000, FUTEX_WAKE, 1 <unfinished ...>
    4409  <... futex resumed>)              = 0
    4468  <... futex resumed>)              = 1
    4409  read(0,  <unfinished ...>
    4468  futex(0x74b3bad7d000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
    4409  <... read resumed>"qwerty\n", 4088) = 7
    4409  futex(0x73029f25a000, FUTEX_WAKE, 1) = 1
    4469  <... futex resumed>)              = 0
    4409  futex(0x73029f259000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
    4469  write(1, "ytrewq\n", 7)           = 7
    4469  futex(0x7e98ea6ad000, FUTEX_WAKE, 1 <unfinished ...>
    4409  <... futex resumed>)              = 0
    4469  <... futex resumed>)              = 1
    4409  read(0,  <unfinished ...>
    4469  futex(0x7e98ea6ae000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
    4409  <... read resumed>"123456789\n", 4088) = 10
    4409  futex(0x73029f25b000, FUTEX_WAKE, 1) = 1
    4468  <... futex resumed>)              = 0
    4409  futex(0x73029f259000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
    4468  write(1, "987654321\n", 10)       = 10
```

```
    4468  futex(0x74b3bad7c000, FUTEX_WAKE, 1 <unfinished ...>
    4409  <... futex resumed>)              = 0
    4468  <... futex resumed>)              = 1
    4409  read(0,  <unfinished ...>
    4468  futex(0x74b3bad7d000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
    4409  <... read resumed>"\n", 4088)     = 1
    4409  futex(0x73029f25b000, FUTEX_WAKE, 1) = 1
    4468  <... futex resumed>)              = 0
    4409  futex(0x73029f25a000, FUTEX_WAKE, 1 <unfinished ...>
    4468  write(2, "Child PID 4468 exit successfully"..., 33
<unfinished ...>
    4409  <... futex resumed>)              = 1
    4469  <... futex resumed>)              = 0
    4409  wait4(4468,  <unfinished ...>
    4468  <... write resumed>)              = 33
    4469  write(2, "Child PID 4469 exit successfully"..., 33
<unfinished ...>
    4468  munmap(0x74b3bad7c000, 32 <unfinished ...>
    4469  <... write resumed>)              = 33
    4468  <... munmap resumed>)             = 0
    4469  munmap(0x7e98ea6ad000, 32 <unfinished ...>
    4468  munmap(0x74b3bad7d000, 32 <unfinished ...>
    4469  <... munmap resumed>)             = 0
    4468  <... munmap resumed>)             = 0
    4469  munmap(0x7e98ea6ae000, 32 <unfinished ...>
    4468  munmap(0x74b3bad7e000, 4096 <unfinished ...>
    4469  <... munmap resumed>)             = 0
    4468  <... munmap resumed>)             = 0
    4469  munmap(0x7e98ea6af000, 4096 <unfinished ...>
    4468  close(3 <unfinished ...>
    4469  <... munmap resumed>)             = 0
    4468  <... close resumed>)              = 0
    4469  close(3 <unfinished ...>
    4468  exit_group(0 <unfinished ...>
    4469  <... close resumed>)              = 0
    4468  <... exit_group resumed>)         = ?
    4469  exit_group(0)                     = ?
    4468  +++ exited with 0 +++
    4409  <... wait4 resumed>NULL, 0, NULL) = 4468
    4469  +++ exited with 0 +++
    4409  --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=4468,
si_uid=0, si_status=0, si_utime=0, si_stime=0} ---
    4409  wait4(4469, NULL, 0, NULL)        = 4469
    4409  write(1, "Parent exit successfully\n\0", 26) = 26
    4409  munmap(0x73029f25b000, 32)        = 0
    4409  unlink("/dev/shm/sem.sem_client_to_server_1_1763446038_807639635")
= 0
    4409  munmap(0x73029f25a000, 32)        = 0
    4409  unlink("/dev/shm/sem.sem_client_to_server_2_1763446038_1802162044")
= 0
    4409  munmap(0x73029f259000, 32)        = 0
    4409  unlink("/dev/shm/sem.sem_server_to_client_1763446038_54883658") =
0
```

```
4409  munmap(0x73029f25c000, 4096)      = 0
4409  close(5)                          = 0
4409  unlink("/dev/shm/shm_1763446038_1983967848") = 0
4409  exit_group(0)                     = ?
4409  +++ exited with 0 +++
```

Вывод

Приобрел практические навыки в управлении процессами в ОС Linux и обеспечении обмена данными между процессами с помощью разделяемой памяти (shared memory), отображения памяти (memory mapping) и семафоров, а также в использовании инструмента отладки strace.

Во время работы столкнулся с ошибкой Bus error во время повторного запуска программы после прерывания с помощью Ctrl+C, решение заключается в привязывании функции очистки ресурсов к соответствующему сигналу.