МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №2
по курсу «Операционные системы»

Выполнил: А. В. Маркелов
Группа: М8О-207БВ-24
Преподаватель: Е. С. Миронов

Москва, 2025

Условие

Цель работы: Приобретение практических навыков в: - Освоение принципов работы с файловыми системами - Обеспечение обмена данных между процессами посредством технологии «File mapping»

Задание:Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или через отображаемые файлы (memory-mapped files). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Правило фильтрации: нечетные строки отправляются в file1, четные в file2. Дочерние процессы инвертируют строки.

Вариант: 21

Метод решения

При запуске программы через коммандную строку подаются три значения: путь к собранному файлу родительского процесса, файл для нечетных строк, файл для четных строк.

Родительский процесс:

1. Инициализирует:

- CreateMappedFile() для двух входных файлов

- MapFileToMemory() для отображения в память

- CreateNamedSemaphore() для четырех семафоров

2. Создает дочерние процессы (POSIX):

- fork() → child1

- fork() → child2

3. Выполняет главный цикл, который включает в себя:

- Чтение строки из stdin

- Завершение, если строка пустая

- Иначе:

- Если четная, отправляет в child1, ждет обработки

- Если нечетная, Отправляет в child2, ждет обработки

4. Завершает выполнение программы:

- Сигнализирует shutdown_flag обоим дочерним процессам

- WaitForChildren() для ожидания завершения

- UnmapFile() и CloseSemaphore() в рамках CleanupChildIpc() как очистка

Каждый из дочерних процессов:

1. Инициализирует:

- OpenNamedSemaphore() - открытие семафоров

- OpenObject() - открытие входного mmap

- MapFileToMemory() - отображение в память

- OpenObject() - создание выходного файла
- SendSignal() - уведомление parent что готов
2. Выполняет главный цикл, который включает в себя:
- WaitSemaphore(data_ready) - ожидание данных
- Проверка shutdown_flag:
- Если 1 → выход из цикла
- Иначе продолжение
- Копирование данных из shared_buffer
- Применение Reverse() функции
- WriteToObject() записывает результат в выходной файл
- PostSemaphore(processed) - сигнал завершения
3. Завершает своё выполнение:
- CloseObject() выходной файл
- UnmapFile() входной буфер
- CloseSemaphore() все семафоры

Описание программы

Входные данные: названия двух файлов для записи инвертированных строк, непустые строки.

В файле os.h объявлены функции-обертки, универсальные структуры и типы данных для реализации кроссплатформенности. Реализация для POSIX систем всех функций, перечисленных в os.h, приведена в os_linux.c.

В файле helpers.h объявлены вспомогательные функции и структуры данных, константы для parent.c и child.c файлов. В helpers.c содержатся реализации всех вспомогательных функций. Решение ввести helpers файлы было принято для умененьшения количества дублирования кода и повышения читаемости основного кода (parent.c и child.c).

В файле parent.c реализован алгоритм создания общих буфферов для сообщения с дочерними процессами, семафоров для передачи сигналов между процессами, создание дочерних процессов, чтение и перенаправление строк в соответсвующие дочерние процессы. В файле child.c производится открытие выходных файлов, чтение строк из общего с родительским процессом буффера, реверс полученных строк и запись в выходные файлы.

Результаты

При запуске программы, открытии файлов odd.txt и even.txt и вводе тестовых строк нечётные строки попали в файл odd.txt в перевёрнутом виде, а чётные — в файл even.txt так же в перевернутом виде.

Пример работы:

Ввод:

odd

even

text

asd

123das

Test

^D

Содержимое odd.txt:

txet

sad321

Содержимое even.txt:

dsa

tseT

Выводы

Реализована система межпроцессного взаимодействия, которая успешно демонстрирует использование механизмов POSIX для синхронизации и обмена данными между процессами. Система может быть дополнена реализацией небольшого количества оберточных функций для не POSIX систем и работать кроссплатформенно.

Все поставленные цели лабораторной работы успешно достигнуты: освоены принципы работы с файловыми системами, обеспечен обмен данных между процессами посредством технологии «File mapping». Создана эффективно работающая программа на языке программирования C, которая получает от пользователя строки, инвертирует их и записывает, в зависимости от четности порядкового номера, строки в выходные файлы, указанные при запуске пользователем.

В ходе выполнения лабораторной работы получены знания:

1. POSIX API:

- Все основные функции для работы с процессами
- File mapping для эффективной передачи информации между процессами
- Семафоры для синхронизации
- Сигналы для управления процессами

2. Многопроцессорное программирование:

- Race conditions и как их избежать
- Deadlock и механизмы для его предотвращения
- Правильное управление ресурсами
- Отладка многопроцессных приложений

3. Системное программирование:

- Абстракции ОС и их реальная работа
- Производительность и оптимизация
- Надежность и обработка ошибок

4. Инженерные практики:

- Модульный дизайн программы
- Разделение интерфейса и реализации
- Кроссплатформенноть кода
- Переиспользуемость кода
- Обработка граничных случаев

Исходная программа

```
#pragma once

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#define _GNU_SOURCE

#ifdef _WIN32

#else
#include <fcntl.h>
#include <semaphore.h>
#include <signal.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/wait.h>

typedef int pipe_t;
typedef pid_t process_id_t;
typedef void* mmap_handle_t;
typedef void (*signal_handler_t)(int);

typedef struct {
  void* addr;
  size_t size;
  int fd;
} mmap_info_t;

typedef sem_t* semaphore_t;

#endif

process_id_t CreateProc(const char* file,char* argv[],char*
envp[]);

int WaitObject(process_id_t proc_info,int* status,int options);

void TerminateProc(int status);

process_id_t GetProcessId(void);

process_id_t GetParentProcessId(void);

pipe_t OpenObject(const char* path,int flags,int mode);
```

```
      int CloseObject(pipe_t fd);

      ssize_t WriteToObject(pipe_t fd,const void* line,size_t count);

      int TruncateFile(pipe_t fd,off_t size);

      pipe_t CreateMappedFile(const char* path,size_t size);

      mmap_info_t MapFileToMemory(pipe_t fd,size_t size,int prot);

      int UnmapFile(mmap_info_t info);

      int SyncMappedFile(mmap_info_t info);

      int RegisterSignalHandler(int sig,signal_handler_t handler);

      int SendSignal(process_id_t pid,int sig);

      semaphore_t CreateNamedSemaphore(const char* name,int
initial_value);

      semaphore_t OpenNamedSemaphore(const char* name);

      int WaitSemaphore(semaphore_t sem);

      int PostSemaphore(semaphore_t sem);

      int CloseSemaphore(semaphore_t sem);

      int UnlinkSemaphore(const char* name);
```

Листинг 1: *Заголовочный файл для обеспечаения кроссплатформенности*

```
      #include "os.h"

      process_id_t CreateProc(const char* file,char* argv[],char* envp[])
{
        process_id_t result;
        pid_t pid = fork();

        if (pid == -1) {
          perror("CreateProc: fork() failed");
          result = -1;
          return result;
        }
```

```
  if (pid == 0) {
    execve(file,argv,envp);
    perror("CreateProc: execve() failed");
    TerminateProc(EXIT_FAILURE);
  }

  result = pid;
  return result;
}

int WaitObject(process_id_t proc_info,int* status,int options) {
  int res = waitpid(proc_info,status,options);
  if (res == -1) {
    perror("WaitObject: waitpid() failed");
  }
  return res;
}

void TerminateProc(int status) { _exit(status); }

process_id_t GetProcessId(void) { return getpid(); }

process_id_t GetParentProcessId(void) { return getppid(); }

pipe_t OpenObject(const char* path,int flags,int mode) {
  int fd = open(path,flags,mode);
  if (fd == -1) {
    perror("OpenObject: open() failed");
  }
  return fd;
}

int CloseObject(pipe_t fd) {
  if (fd <0) {
    return 0;
  }
  int res = close(fd);
  if (res == -1) {
    perror("CloseObject: close() failed");
  }
  return res;
}

ssize_t WriteToObject(pipe_t fd,const void* line,size_t count) {
  return write(fd,line,count);
}

int TruncateFile(pipe_t fd,off_t size) {
```

```
  int res = ftruncate(fd,size);
  if (res == -1) {
    perror("TruncateFile: ftruncate() failed");
  }
  return res;
}

pipe_t CreateMappedFile(const char* path,size_t size) {
  pipe_t fd = open(path,O_RDWR | O_CREAT | O_TRUNC,0666);
  if (fd == -1) {
    perror("CreateMappedFile: open() failed");
    return -1;
  }

  if (ftruncate(fd,size) == -1) {
    perror("CreateMappedFile: ftruncate() failed");
    close(fd);
    return -1;
  }

  return fd;
}

mmap_info_t MapFileToMemory(pipe_t fd,size_t size,int prot) {
  mmap_info_t result;
  result.fd = fd;
  result.size = size;

  result.addr = mmap(NULL,size,prot,MAP_SHARED,fd,0);
  if (result.addr == MAP_FAILED) {
    perror("MapFileToMemory: mmap() failed");
    result.addr = NULL;
    return result;
  }

  return result;
}

int UnmapFile(mmap_info_t info) {
  if (info.addr == NULL || info.addr == MAP_FAILED) {
    return 0;
  }

  int res = munmap(info.addr,info.size);
  if (res == -1) {
    perror("UnmapFile: munmap() failed");
  }
  return res;
```

```
}

int SyncMappedFile(mmap_info_t info) {
  if (info.addr == NULL || info.addr == MAP_FAILED) {
    return -1;
  }

  int res = msync(info.addr,info.size,MS_SYNC);
  if (res == -1) {
    perror("SyncMappedFile: msync() failed");
  }
  return res;
}

int RegisterSignalHandler(int sig,signal_handler_t handler) {
  struct sigaction sa;
  memset(&sa,0,sizeof(sa));
  sigemptyset(&sa.sa_mask);
  sa.sa_handler = handler;
  sa.sa_flags = 0;

  int res = sigaction(sig,&sa,NULL);
  if (res == -1) {
    perror("RegisterSignalHandler: sigaction() failed");
  }
  return res;
}

int SendSignal(process_id_t pid,int sig) {
  int res = kill(pid,sig);
  if (res == -1) {
    perror("SendSignal: kill() failed");
  }
  return res;
}

semaphore_t CreateNamedSemaphore(const char* name,int
initial_value) {
  sem_unlink(name);

  semaphore_t sem = sem_open(name,O_CREAT |
O_EXCL,0666,initial_value);
  if (sem == SEM_FAILED) {
    perror("CreateNamedSemaphore: sem_open() failed");
    return NULL;
  }
  return sem;
}
```

```
semaphore_t OpenNamedSemaphore(const char* name) {
  semaphore_t sem = sem_open(name,0);
  if (sem == SEM_FAILED) {
    perror("OpenNamedSemaphore: sem_open() failed");
    return NULL;
  }
  return sem;
}

int WaitSemaphore(semaphore_t sem) {
  if (sem == NULL) {
    fprintf(stderr,"WaitSemaphore: NULL semaphore");
    return -1;
  }

  int res = sem_wait(sem);
  if (res == -1) {
    perror("WaitSemaphore: sem_wait() failed");
  }
  return res;
}

int PostSemaphore(semaphore_t sem) {
  if (sem == NULL) {
    fprintf(stderr,"PostSemaphore: NULL semaphore");
    return -1;
  }

  int res = sem_post(sem);
  if (res == -1) {
    perror("PostSemaphore: sem_post() failed");
  }
  return res;
}

int CloseSemaphore(semaphore_t sem) {
  if (sem == NULL) {
    return 0;
  }

  int res = sem_close(sem);
  if (res == -1) {
    perror("CloseSemaphore: sem_close() failed");
  }
  return res;
}
```

```
int UnlinkSemaphore(const char* name) {
  int res = sem_unlink(name);
  if (res == -1) {
    perror("UnlinkSemaphore: sem_unlink() failed");
  }
  return res;
}
```

Листинг 2: *Реализация кроссплатформенности (только POSIX системы)*

```
#include <stdio.h>
#include <string.h>

#include "helpers.h"

volatile sig_atomic_t child1_ready = 0;
volatile sig_atomic_t child2_ready = 0;

void ChildReadyHandler(int sig) {
  if (sig == SIGUSR1) {
    child1_ready = 1;
  } else if (sig == SIGUSR2) {
    child2_ready = 1;
  }
}

static void WaitForChildrenReady(void) {
  while (!child1_ready || !child2_ready) {
    pause();
  }
}

int main(int argc,char* argv[],char* envp[]) {
  if (argc != 3) {
    fprintf(stderr,"usage: %s <file_odd><file_even>\n",argv[0]);
    return EXIT_FAILURE;
  }

  if (RegisterSignalHandler(SIGUSR1,ChildReadyHandler) == -1 ||
      RegisterSignalHandler(SIGUSR2,ChildReadyHandler) == -1) {
    fprintf(stderr,"[Parent] Ошибка регистрации обработчиков
сигналов\n");
    return EXIT_FAILURE;
  }

  ipc_handles_t h1 =

InitIpcHandles(1,"/tmp/child1_input.mmap","/sem_child1_data_ready",
```

```
                        "/sem_child1_processed");
        ipc_handles_t h2 =

InitIpcHandles(2,"/tmp/child2_input.mmap","/sem_child2_data_ready",
                        "/sem_child2_processed");

        if (h1.data_ready == NULL || h2.data_ready == NULL) {
          fprintf(stderr,"[Parent] Ошибка инициализации IPC\n");
          CleanupChildIpc(&h1);
          CleanupChildIpc(&h2);
          return EXIT_FAILURE;
        }

        char* args1[] =
{"child","1","/tmp/child1_input.mmap",argv[1],NULL};
        process_id_t pid1 = CreateProc("./child",args1,envp);

        if (pid1 == -1) {
          fprintf(stderr,"[Parent] Ошибка при создании child1\n");
          CleanupChildIpc(&h2);
          CleanupChildIpc(&h1);
          return EXIT_FAILURE;
        }

        char* args2[] =
{"child","2","/tmp/child2_input.mmap",argv[2],NULL};
        process_id_t pid2 = CreateProc("./child",args2,envp);

        if (pid2 == -1) {
          fprintf(stderr,"[Parent] Ошибка при создании child2\n");
          CleanupChildIpc(&h2);
          CleanupChildIpc(&h1);
          return EXIT_FAILURE;
        }

        WaitForChildrenReady();

        shared_buffer_t* buf1 = (shared_buffer_t*)h1.input_mmap.addr;
        shared_buffer_t* buf2 = (shared_buffer_t*)h2.input_mmap.addr;

        printf("Введите строки (пустая строка для завершения):\n");
        char line[MAX_LINE_LENGTH];
        int line_number = 0;

        while (fgets(line,sizeof(line),stdin) != NULL) {
          if (line[0] == '\n') {
            break;
          }
```

```
    line_number++;
    size_t len = strlen(line);

    if (line_number % 2 == 1) {
      SendDataToChild(buf1,line,len,h1.data_ready,h1.processed);
    } else {
      SendDataToChild(buf2,line,len,h2.data_ready,h2.processed);
    }
  }

  SignalChildShutdown(buf1,h1.data_ready);
  SignalChildShutdown(buf2,h2.data_ready);

  int status1,status2;
  WaitForChildren(pid1,pid2,&status1,&status2);

  CleanupChildIpc(&h1);
  CleanupChildIpc(&h2);

  return EXIT_SUCCESS;
}
```

Листинг 3: *Родительский процесс*

```
#include <stdio.h>
#include <string.h>

#include "helpers.h"

static void Reverse(char* line) {
  size_t len = strlen(line);
  if (len == 0) {
    return;
  }

  int has_newline = (line[len -1] == '\n');
  if (has_newline) {
    len--;
  }

  for (size_t i = 0,j = len -1; i <j; i++,j--) {
    char tmp = line[i];
    line[i] = line[j];
    line[j] = tmp;
  }
```

```
        if (has_newline) {
          line[len] = '\n';
          line[len + 1] = '\0';
        } else {
          line[len] = '\0';
        }
      }

      int main(int argc,char* argv[]) {
        child_config_t config;
        if (InitChildConfig(argc,argv,&config) == -1) {
          return EXIT_FAILURE;
        }

        ipc_handles_t ipc = SetupChildIpc(&config);
        if (ipc.data_ready == NULL) {
          return EXIT_FAILURE;
        }

        int output_fd = CreateOutputFile(&config);
        if (output_fd == -1) {
          CleanupChildIpcSafe(&ipc);
          return EXIT_FAILURE;
        }

        if (SignalReady(&config) == -1) {
          fprintf(stderr,"[Child%d] Ошибка отправки сигнала
готовности\n",
                  config.id);
          CloseObject(output_fd);
          CleanupChildIpcSafe(&ipc);
          return EXIT_FAILURE;
        }

        char buffer[MAX_LINE_LENGTH];

        while (1) {
          if (WaitSemaphore(ipc.data_ready) == -1) {
            fprintf(stderr,"[Child%d] Ошибка ожидания
данных\n",config.id);
            break;
          }

          shared_buffer_t* shared_buf =
(shared_buffer_t*)ipc.input_mmap.addr;

          if (shared_buf->shutdown_flag == 1) {
            break;
```

```
        }

        int line_len = shared_buf->line_length;
        if (line_len >0 && line_len <MAX_LINE_LENGTH) {
          memcpy(buffer,shared_buf->data,line_len + 1);
          buffer[line_len] = '\0';

          Reverse(buffer);

          int reversed_len = strlen(buffer);
          if (WriteToOutput(output_fd,buffer,reversed_len,config.id) ==
-1) {
            PostSemaphore(ipc.processed);
            break;
          }
        }

        if (PostSemaphore(ipc.processed) == -1) {
          fprintf(stderr,"[Child%d] Ошибка сигнализации
завершения\n",config.id);
          break;
        }
      }

      CloseObject(output_fd);
      CleanupChildIpcSafe(&ipc);

      return EXIT_SUCCESS;
    }
```

Листинг 4: *Дочерний процесс*

```
    #include "os.h"

    #define MAX_LINE_LENGTH 1024
    #define MAPPED_FILE_SIZE (100 * MAX_LINE_LENGTH)

    typedef struct {
      volatile int line_length;
      volatile int shutdown_flag;
      char data[MAX_LINE_LENGTH];
    } shared_buffer_t;

    typedef struct {
      semaphore_t data_ready;
      semaphore_t processed;
      mmap_info_t input_mmap;
```

```
  pipe_t input_fd;
  const char *data_ready_name;
  const char *processed_name;
} ipc_handles_t;

typedef struct {
  int id;
  const char *input_mmap_path;
  const char *output_file_path;
} child_config_t;

ipc_handles_t InitIpcHandles(int child_id,const char *input_path,
                             const char *data_ready_name,
                             const char *processed_name);

void CleanupChildIpc(ipc_handles_t *handles);

int SendDataToChild(shared_buffer_t *buf,const char *data,size_t
len,
                    semaphore_t sem_ready,semaphore_t
sem_processed);

void SignalChildShutdown(shared_buffer_t *buf,semaphore_t
sem_ready);

void WaitForChildren(process_id_t pid1,process_id_t pid2,int
*status1,
                     int *status2);

int InitChildConfig(int argc,char *argv[],child_config_t *config);

ipc_handles_t SetupChildIpc(const child_config_t *config);

void CleanupChildIpcSafe(ipc_handles_t *handles);

int CreateOutputFile(const child_config_t *config);

int SignalReady(const child_config_t *config);

int WriteToOutput(int fd,const char *data,size_t data_len,int
child_id);
```

Листинг 5: *Заголовочный файл вспомогательных функций*

```
#include "helpers.h"

#include <stdio.h>
```

```
      #include <string.h>

      ipc_handles_t InitIpcHandles(int child_id,const char *input_path,
                                   const char *data_ready_name,
                                   const char *processed_name) {
        ipc_handles_t handles = {0};
        handles.data_ready_name = data_ready_name;
        handles.processed_name = processed_name;

        handles.input_fd =
CreateMappedFile(input_path,sizeof(shared_buffer_t));
        if (handles.input_fd == -1) {
          fprintf(stderr,"[Parent] Ошибка создания входного файла для
child%d\n",
                  child_id);
          return handles;
        }

        handles.input_mmap = MapFileToMemory(
            handles.input_fd,sizeof(shared_buffer_t),PROT_READ |
PROT_WRITE);
        if (handles.input_mmap.addr == NULL) {
          fprintf(stderr,"[Parent] Ошибка отображения входного файла для
child%d\n",
                  child_id);
          CloseObject(handles.input_fd);
          handles.data_ready = NULL;
          return handles;
        }

        handles.data_ready = CreateNamedSemaphore(data_ready_name,0);
        handles.processed = CreateNamedSemaphore(processed_name,0);

        if (handles.data_ready == NULL || handles.processed == NULL) {
          fprintf(stderr,"[Parent] Ошибка создания семафоров для
child%d\n",
                  child_id);
          UnmapFile(handles.input_mmap);
          CloseObject(handles.input_fd);

          if (handles.data_ready) {
            CloseSemaphore(handles.data_ready);
          }
          if (handles.processed) {
            CloseSemaphore(handles.processed);
          }
          handles.data_ready = NULL;
          return handles;
```

```
        }

        shared_buffer_t *buf = (shared_buffer_t
*)handles.input_mmap.addr;
        memset(buf,0,sizeof(shared_buffer_t));

        return handles;
      }

      void CleanupChildIpc(ipc_handles_t *handles) {
        if (handles == NULL) {
          return;
        }

        UnmapFile(handles->input_mmap);
        CloseObject(handles->input_fd);
        CloseSemaphore(handles->data_ready);
        CloseSemaphore(handles->processed);
        UnlinkSemaphore(handles->data_ready_name);
        UnlinkSemaphore(handles->processed_name);
      }

      int SendDataToChild(shared_buffer_t *buf,const char *data,size_t
len,
                          semaphore_t sem_ready,semaphore_t
sem_processed) {
        if (len >MAX_LINE_LENGTH -1) {
          fprintf(stderr,"[Parent] Ошибка: строка слишком длинная\n");
          return -1;
        }

        memcpy(buf->data,data,len + 1);
        buf->line_length = len;
        buf->shutdown_flag = 0;

        if (PostSemaphore(sem_ready) == -1) {
          return -1;
        }

        if (WaitSemaphore(sem_processed) == -1) {
          return -1;
        }

        return 0;
      }

      void SignalChildShutdown(shared_buffer_t *buf,semaphore_t
sem_ready) {
```

```
        buf->shutdown_flag = 1;
        PostSemaphore(sem_ready);
      }

      void WaitForChildren(process_id_t pid1,process_id_t pid2,int
*status1,
                           int *status2) {
        *status1 = 0;
        *status2 = 0;
        WaitObject(pid1,status1,0);
        WaitObject(pid2,status2,0);
      }

      int InitChildConfig(int argc,char *argv[],child_config_t *config) {
        if (argc != 4) {
          fprintf(stderr,"Child: неверное количество аргументов\n");
          fprintf(stderr,
                  "Usage: %s
<child_id><input_mmap_path><output_file_path>\n",
                  argv[0]);
          return -1;
        }

        config->id = atoi(argv[1]);
        config->input_mmap_path = argv[2];
        config->output_file_path = argv[3];

        return 0;
      }

      ipc_handles_t SetupChildIpc(const child_config_t *config) {
        ipc_handles_t handles;
        memset(&handles,0,sizeof(handles));

        const char *data_ready_name =
            (config->id == 1) ? "/sem_child1_data_ready" :
"/sem_child2_data_ready";
        const char *processed_name =
            (config->id == 1) ? "/sem_child1_processed" :
"/sem_child2_processed";

        handles.data_ready_name = data_ready_name;
        handles.processed_name = processed_name;

        handles.data_ready = OpenNamedSemaphore(data_ready_name);
        handles.processed = OpenNamedSemaphore(processed_name);

        if (handles.data_ready == NULL || handles.processed == NULL) {
```

```
          fprintf(stderr,"[Child%d] Ошибка открытия
семафоров\n",config->id);
          return handles;
        }

        handles.input_fd =
OpenObject(config->input_mmap_path,O_RDWR,0666);
        if (handles.input_fd == -1) {
          fprintf(stderr,"[Child%d] Ошибка открытия входного
файла\n",config->id);
          CloseSemaphore(handles.data_ready);
          CloseSemaphore(handles.processed);
          handles.data_ready = NULL;
          return handles;
        }

        handles.input_mmap = MapFileToMemory(
            handles.input_fd,sizeof(shared_buffer_t),PROT_READ |
PROT_WRITE);
        if (handles.input_mmap.addr == NULL) {
          fprintf(stderr,"[Child%d] Ошибка отображения входного файла\n",
                  config->id);
          CloseObject(handles.input_fd);
          CloseSemaphore(handles.data_ready);
          CloseSemaphore(handles.processed);
          handles.data_ready = NULL;
          return handles;
        }

        return handles;
      }

      void CleanupChildIpcSafe(ipc_handles_t *handles) {
        if (handles == NULL) {
          return;
        }

        UnmapFile(handles->input_mmap);
        CloseObject(handles->input_fd);
        CloseSemaphore(handles->data_ready);
        CloseSemaphore(handles->processed);
      }

      int CreateOutputFile(const child_config_t *config) {
        int fd =
            OpenObject(config->output_file_path,O_CREAT | O_WRONLY |
O_TRUNC,0666);
        if (fd == -1) {
```

```
          fprintf(stderr,"[Child%d] Ошибка создания выходного файла
%s\n",
                  config->id,config->output_file_path);
          return -1;
        }
        return fd;
      }

      int SignalReady(const child_config_t *config) {
        int sig = (config->id == 1) ? SIGUSR1 : SIGUSR2;
        return SendSignal(GetParentProcessId(),sig);
      }

      int WriteToOutput(int fd,const char *data,size_t data_len,int
child_id) {
        ssize_t written = WriteToObject(fd,data,data_len);
        if (written == -1 || (size_t)written != data_len) {
          fprintf(stderr,"[Child%d] Ошибка записи в выходной
файл\n",child_id);
          return -1;
        }
        return 0;
      }
```

Листинг 6: *Реализация вспомогательных функций*

```
      20333 execve("./parent",["./parent","odd","even"],0x7ffee849adf8 /*
36 vars */) = 0
      20333 brk(NULL)                           = 0x5653675e3000
      20333 arch_prctl(0x3001 /* ARCH_??? */,0x7ffd37bb8fb0) = -1 EINVAL
(Invalid argument)
      20333
mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) =
0x7ff4a428b000
      20333 access("/etc/ld.so.preload",R_OK) = -1 ENOENT (No such file
or directory)
      20333 openat(AT_FDCWD,"/etc/ld.so.cache",O_RDONLY|O_CLOEXEC) = 3
      20333
newfstatat(3,"",{st_mode=S_IFREG|0644,st_size=30076,...},AT_EMPTY_PATH) =
0
      20333 mmap(NULL,30076,PROT_READ,MAP_PRIVATE,3,0) = 0x7ff4a4283000
      20333 close(3)                            = 0
      20333
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libc.so.6",O_RDONLY|O_CLOEXEC) = 3
      20333
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\0"...,832)
= 832
```

```
     20333
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
     20333 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\0"...,48,848) = 48
     20333
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
     20333
newfstatat(3,"",{st_mode=S_IFREG|0755,st_size=2220400,...},AT_EMPTY_PATH)
= 0
     20333
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
     20333 mmap(NULL,2264656,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) =
0x7ff4a405a000
     20333 mprotect(0x7ff4a4082000,2023424,PROT_NONE) = 0
     20333
mmap(0x7ff4a4082000,1658880,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3
= 0x7ff4a4082000
     20333
mmap(0x7ff4a4217000,360448,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x1bd000)
= 0x7ff4a4217000
     20333
mmap(0x7ff4a4270000,24576,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,
= 0x7ff4a4270000
     20333
mmap(0x7ff4a4276000,52816,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,-1
= 0x7ff4a4276000
     20333 close(3)                          = 0
     20333
mmap(NULL,12288,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) =
0x7ff4a4057000
     20333 arch_prctl(ARCH_SET_FS,0x7ff4a4057740) = 0
     20333 set_tid_address(0x7ff4a4057a10)   = 20333
     20333 set_robust_list(0x7ff4a4057a20,24) = 0
     20333 rseq(0x7ff4a40580e0,0x20,0,0x53053053) = 0
     20333 mprotect(0x7ff4a4270000,16384,PROT_READ) = 0
     20333 mprotect(0x565343536000,4096,PROT_READ) = 0
     20333 mprotect(0x7ff4a42c5000,8192,PROT_READ) = 0
     20333
prlimit64(0,RLIMIT_STACK,NULL,{rlim_cur=8192*1024,rlim_max=RLIM64_INFINITY})
= 0
     20333 munmap(0x7ff4a4283000,30076)     = 0
     20333
rt_sigaction(SIGUSR1,{sa_handler=0x565343532529,sa_mask=[],sa_flags=SA_RESTORER,sa_re
= 0
     20333
```

```
rt_sigaction(SIGUSR2,{sa_handler=0x565343532529,sa_mask=[],sa_flags=SA_RESTORER,sa_re
= 0
     20333
openat(AT_FDCWD,"/tmp/child1_input.mmap",O_RDWR|O_CREAT|O_TRUNC,0666) = 3
     20333 ftruncate(3,1032)                 = 0
     20333 mmap(NULL,1032,PROT_READ|PROT_WRITE,MAP_SHARED,3,0) =
0x7ff4a42c4000
     20333 unlink("/dev/shm/sem.sem_child1_data_ready") = -1 ENOENT (No
such file or directory)
     20333 getrandom("\xfd\xf0\x37\x04\x53\xf8\x01\x6e",8,GRND_NONBLOCK)
= 8
     20333
newfstatat(AT_FDCWD,"/dev/shm/sem.3lvtej",0x7ffd37bb86a0,AT_SYMLINK_NOFOLLOW)
= -1 ENOENT (No such file or directory)
     20333
openat(AT_FDCWD,"/dev/shm/sem.3lvtej",O_RDWR|O_CREAT|O_EXCL,0666) = 4
     20333
write(4,"\0\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",32)
= 32
     20333 mmap(NULL,32,PROT_READ|PROT_WRITE,MAP_SHARED,4,0) =
0x7ff4a428a000
     20333
link("/dev/shm/sem.3lvtej","/dev/shm/sem.sem_child1_data_ready") = 0
     20333
newfstatat(4,"",{st_mode=S_IFREG|0644,st_size=32,...},AT_EMPTY_PATH) = 0
     20333 getrandom("\x59\x91\xe0\x66\x93\x75\x5c\xbe",8,GRND_NONBLOCK)
= 8
     20333 brk(NULL)                         = 0x5653675e3000
     20333 brk(0x565367604000)               = 0x565367604000
     20333 unlink("/dev/shm/sem.3lvtej")     = 0
     20333 close(4)                          = 0
     20333 unlink("/dev/shm/sem.sem_child1_processed") = -1 ENOENT (No
such file or directory)
     20333 getrandom("\x0c\x4a\x78\x4a\x2b\x07\x82\x95",8,GRND_NONBLOCK)
= 8
     20333
newfstatat(AT_FDCWD,"/dev/shm/sem.QNaeDy",0x7ffd37bb86a0,AT_SYMLINK_NOFOLLOW)
= -1 ENOENT (No such file or directory)
     20333
openat(AT_FDCWD,"/dev/shm/sem.QNaeDy",O_RDWR|O_CREAT|O_EXCL,0666) = 4
     20333
write(4,"\0\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",32)
= 32
     20333 mmap(NULL,32,PROT_READ|PROT_WRITE,MAP_SHARED,4,0) =
0x7ff4a4289000
     20333
link("/dev/shm/sem.QNaeDy","/dev/shm/sem.sem_child1_processed") = 0
     20333
```

```
newfstatat(4,"",{st_mode=S_IFREG|0644,st_size=32,...},AT_EMPTY_PATH) = 0
      20333 unlink("/dev/shm/sem.QNaeDy")     = 0
      20333 close(4)                          = 0
      20333
openat(AT_FDCWD,"/tmp/child2_input.mmap",O_RDWR|O_CREAT|O_TRUNC,0666) = 4
      20333 ftruncate(4,1032)                 = 0
      20333 mmap(NULL,1032,PROT_READ|PROT_WRITE,MAP_SHARED,4,0) =
0x7ff4a4288000
      20333 unlink("/dev/shm/sem.sem_child2_data_ready") = -1 ENOENT (No
such file or directory)
      20333 getrandom("\xab\x54\xd2\x5b\x79\x20\x08\xdb",8,GRND_NONBLOCK)
= 8
      20333
newfstatat(AT_FDCWD,"/dev/shm/sem.rFaRbZ",0x7ffd37bb86a0,AT_SYMLINK_NOFOLLOW)
= -1 ENOENT (No such file or directory)
      20333
openat(AT_FDCWD,"/dev/shm/sem.rFaRbZ",O_RDWR|O_CREAT|O_EXCL,0666) = 5
      20333
write(5,"\0\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",32)
= 32
      20333 mmap(NULL,32,PROT_READ|PROT_WRITE,MAP_SHARED,5,0) =
0x7ff4a4287000
      20333
link("/dev/shm/sem.rFaRbZ","/dev/shm/sem.sem_child2_data_ready") = 0
      20333
newfstatat(5,"",{st_mode=S_IFREG|0644,st_size=32,...},AT_EMPTY_PATH) = 0
      20333 unlink("/dev/shm/sem.rFaRbZ")     = 0
      20333 close(5)                          = 0
      20333 unlink("/dev/shm/sem.sem_child2_processed") = -1 ENOENT (No
such file or directory)
      20333 getrandom("\x96\x29\x76\x1a\x10\xef\x7f\x1d",8,GRND_NONBLOCK)
= 8
      20333
newfstatat(AT_FDCWD,"/dev/shm/sem.6el3XF",0x7ffd37bb86a0,AT_SYMLINK_NOFOLLOW)
= -1 ENOENT (No such file or directory)
      20333
openat(AT_FDCWD,"/dev/shm/sem.6el3XF",O_RDWR|O_CREAT|O_EXCL,0666) = 5
      20333
write(5,"\0\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",32)
= 32
      20333 mmap(NULL,32,PROT_READ|PROT_WRITE,MAP_SHARED,5,0) =
0x7ff4a4286000
      20333
link("/dev/shm/sem.6el3XF","/dev/shm/sem.sem_child2_processed") = 0
      20333
newfstatat(5,"",{st_mode=S_IFREG|0644,st_size=32,...},AT_EMPTY_PATH) = 0
      20333 unlink("/dev/shm/sem.6el3XF")     = 0
      20333 close(5)                          = 0
```

```
     20333
clone(child_stack=NULL,flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,child_ti
= 20334
     20334 set_robust_list(0x7ff4a4057a20,24 <unfinished ...>
     20333
clone(child_stack=NULL,flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD
<unfinished ...>
     20334 <... set_robust_list resumed>)    = 0
     20334
execve("./child",["child","1","/tmp/child1_input.mmap","odd"],0x7ffd37bb9198
/* 36 vars */ <unfinished ...>
     20333 <... clone resumed>,child_tidptr=0x7ff4a4057a10) = 20335
     20333 pause( <unfinished ...>
     20335 set_robust_list(0x7ff4a4057a20,24) = 0
     20335
execve("./child",["child","2","/tmp/child2_input.mmap","even"],0x7ffd37bb9198
/* 36 vars */ <unfinished ...>
     20334 <... execve resumed>)             = 0
     20334 brk(NULL)                         = 0x56101d027000
     20335 <... execve resumed>)             = 0
     20334 arch_prctl(0x3001 /* ARCH_??? */,0x7ffe04250d50 <unfinished
...>
     20335 brk(NULL <unfinished ...>
     20334 <... arch_prctl resumed>)         = -1 EINVAL (Invalid
argument)
     20335 <... brk resumed>)                = 0x56114f6dc000
     20334
mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0
<unfinished ...>
     20335 arch_prctl(0x3001 /* ARCH_??? */,0x7ffdba5961e0 <unfinished
...>
     20334 <... mmap resumed>)               = 0x7f20270c1000
     20335 <... arch_prctl resumed>)         = -1 EINVAL (Invalid
argument)
     20334 access("/etc/ld.so.preload",R_OK <unfinished ...>
     20335
mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0
<unfinished ...>
     20334 <... access resumed>)             = -1 ENOENT (No such file
or directory)
     20335 <... mmap resumed>)               = 0x7ff89f0ec000
     20334 openat(AT_FDCWD,"/etc/ld.so.cache",O_RDONLY|O_CLOEXEC
<unfinished ...>
     20335 access("/etc/ld.so.preload",R_OK <unfinished ...>
     20334 <... openat resumed>)             = 5
     20335 <... access resumed>)             = -1 ENOENT (No such file
or directory)
     20334 newfstatat(5,"", <unfinished ...>
```

```
     20335 openat(AT_FDCWD,"/etc/ld.so.cache",O_RDONLY|O_CLOEXEC
<unfinished ...>
     20334 <... newfstatat
resumed>{st_mode=S_IFREG|0644,st_size=30076,...},AT_EMPTY_PATH) = 0
     20335 <... openat resumed>)             = 5
     20334 mmap(NULL,30076,PROT_READ,MAP_PRIVATE,5,0 <unfinished ...>
     20335 newfstatat(5,"", <unfinished ...>
     20334 <... mmap resumed>)               = 0x7f20270b9000
     20335 <... newfstatat
resumed>{st_mode=S_IFREG|0644,st_size=30076,...},AT_EMPTY_PATH) = 0
     20334 close(5 <unfinished ...>
     20335 mmap(NULL,30076,PROT_READ,MAP_PRIVATE,5,0 <unfinished ...>
     20334 <... close resumed>)              = 0
     20335 <... mmap resumed>)               = 0x7ff89f0e4000
     20334
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libc.so.6",O_RDONLY|O_CLOEXEC
<unfinished ...>
     20335 close(5)                          = 0
     20334 <... openat resumed>)             = 5
     20335
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libc.so.6",O_RDONLY|O_CLOEXEC
<unfinished ...>
     20334 read(5, <unfinished ...>
     20335 <... openat resumed>)             = 5
     20334 <... 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\0"...,832)
= 832
     20335 read(5, <unfinished ...>
     20334 pread64(5, <unfinished ...>
     20335 <... 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\0"...,832)
= 832
     20334 <... 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
     20335 pread64(5, <unfinished ...>
     20334 pread64(5, <unfinished ...>
     20335 <... 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
     20334 <... 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
     20335 pread64(5, <unfinished ...>
     20334 pread64(5, <unfinished ...>
     20335 <... 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
     20334 <... 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
     20335 pread64(5, <unfinished ...>
     20334 newfstatat(5,"", <unfinished ...>
     20335 <... 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
     20334 <... newfstatat
resumed>{st_mode=S_IFREG|0755,st_size=2220400,...},AT_EMPTY_PATH) = 0
     20335 newfstatat(5,"", <unfinished ...>
     20334 pread64(5, <unfinished ...>
     20335 <... newfstatat
resumed>{st_mode=S_IFREG|0755,st_size=2220400,...},AT_EMPTY_PATH) = 0
     20334 <... 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
     20335 pread64(5, <unfinished ...>
     20334 mmap(NULL,2264656,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,5,0
<unfinished ...>
     20335 <... 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
     20334 <... mmap resumed>)               = 0x7f2026e90000
     20335 mmap(NULL,2264656,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,5,0
<unfinished ...>
     20334 mprotect(0x7f2026eb8000,2023424,PROT_NONE <unfinished ...>
     20335 <... mmap resumed>)               = 0x7ff89eebb000
     20334 <... mprotect resumed>)           = 0
     20335 mprotect(0x7ff89eee3000,2023424,PROT_NONE <unfinished ...>
     20334
mmap(0x7f2026eb8000,1658880,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,5
<unfinished ...>
     20335 <... mprotect resumed>)           = 0
     20334 <... mmap resumed>)               = 0x7f2026eb8000
     20335
mmap(0x7ff89eee3000,1658880,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,5
<unfinished ...>
     20334
mmap(0x7f202704d000,360448,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,5,0x1bd000
<unfinished ...>
     20335 <... mmap resumed>)               = 0x7ff89eee3000
     20334 <... mmap resumed>)               = 0x7f202704d000
     20335
mmap(0x7ff89f078000,360448,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,5,0x1bd000)
= 0x7ff89f078000
     20334
mmap(0x7f20270a6000,24576,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,5,
<unfinished ...>
     20335
```

```
mmap(0x7ff89f0d1000,24576,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,5,
<unfinished ...>
     20334 <... mmap resumed>)               = 0x7f20270a6000
     20335 <... mmap resumed>)               = 0x7ff89f0d1000
     20334
mmap(0x7f20270ac000,52816,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,-1
<unfinished ...>
     20335
mmap(0x7ff89f0d7000,52816,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,-1
<unfinished ...>
     20334 <... mmap resumed>)               = 0x7f20270ac000
     20335 <... mmap resumed>)               = 0x7ff89f0d7000
     20334 close(5 <unfinished ...>
     20335 close(5 <unfinished ...>
     20334 <... close resumed>)              = 0
     20335 <... close resumed>)              = 0
     20334
mmap(NULL,12288,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0
<unfinished ...>
     20335
mmap(NULL,12288,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0
<unfinished ...>
     20334 <... mmap resumed>)               = 0x7f2026e8d000
     20335 <... mmap resumed>)               = 0x7ff89eeb8000
     20334 arch_prctl(ARCH_SET_FS,0x7f2026e8d740 <unfinished ...>
     20335 arch_prctl(ARCH_SET_FS,0x7ff89eeb8740 <unfinished ...>
     20334 <... arch_prctl resumed>)         = 0
     20335 <... arch_prctl resumed>)         = 0
     20334 set_tid_address(0x7f2026e8da10 <unfinished ...>
     20335 set_tid_address(0x7ff89eeb8a10 <unfinished ...>
     20334 <... set_tid_address resumed>)    = 20334
     20335 <... set_tid_address resumed>)    = 20335
     20334 set_robust_list(0x7f2026e8da20,24 <unfinished ...>
     20335 set_robust_list(0x7ff89eeb8a20,24 <unfinished ...>
     20334 <... set_robust_list resumed>)    = 0
     20335 <... set_robust_list resumed>)    = 0
     20334 rseq(0x7f2026e8e0e0,0x20,0,0x53053053 <unfinished ...>
     20335 rseq(0x7ff89eeb90e0,0x20,0,0x53053053 <unfinished ...>
     20334 <... rseq resumed>)               = 0
     20335 <... rseq resumed>)               = 0
     20335 mprotect(0x7ff89f0d1000,16384,PROT_READ <unfinished ...>
     20334 mprotect(0x7f20270a6000,16384,PROT_READ <unfinished ...>
     20335 <... mprotect resumed>)           = 0
     20334 <... mprotect resumed>)           = 0
     20335 mprotect(0x5611490d5000,4096,PROT_READ) = 0
     20334 mprotect(0x56100a1e5000,4096,PROT_READ <unfinished ...>
     20335 mprotect(0x7ff89f126000,8192,PROT_READ <unfinished ...>
     20334 <... mprotect resumed>)           = 0
```

```
      20335 <... mprotect resumed>)           = 0
      20334 mprotect(0x7f20270fb000,8192,PROT_READ <unfinished ...>
      20335 prlimit64(0,RLIMIT_STACK,NULL, <unfinished ...>
      20334 <... mprotect resumed>)           = 0
      20335 <... prlimit64
resumed>{rlim_cur=8192*1024,rlim_max=RLIM64_INFINITY}) = 0
      20334 prlimit64(0,RLIMIT_STACK,NULL, <unfinished ...>
      20335 munmap(0x7ff89f0e4000,30076 <unfinished ...>
      20334 <... prlimit64
resumed>{rlim_cur=8192*1024,rlim_max=RLIM64_INFINITY}) = 0
      20335 <... munmap resumed>)             = 0
      20335
openat(AT_FDCWD,"/dev/shm/sem.sem_child2_data_ready",O_RDWR|O_NOFOLLOW
<unfinished ...>
      20334 munmap(0x7f20270b9000,30076 <unfinished ...>
      20335 <... openat resumed>)             = 5
      20334 <... munmap resumed>)             = 0
      20335
newfstatat(5,"",{st_mode=S_IFREG|0644,st_size=32,...},AT_EMPTY_PATH) = 0
      20334
openat(AT_FDCWD,"/dev/shm/sem.sem_child1_data_ready",O_RDWR|O_NOFOLLOW
<unfinished ...>
      20335 getrandom( <unfinished ...>
      20334 <... openat resumed>)             = 5
      20335 <... getrandom
resumed>"\x52\xc9\x79\x6f\x3a\xab\xb4\x07",8,GRND_NONBLOCK) = 8
      20334 newfstatat(5,"", <unfinished ...>
      20335 brk(NULL <unfinished ...>
      20334 <... newfstatat
resumed>{st_mode=S_IFREG|0644,st_size=32,...},AT_EMPTY_PATH) = 0
      20335 <... brk resumed>)                = 0x56114f6dc000
      20334 getrandom( <unfinished ...>
      20335 brk(0x56114f6fd000 <unfinished ...>
      20334 <... getrandom
resumed>"\x39\xd3\x5f\x67\xd1\x84\xe1\xa9",8,GRND_NONBLOCK) = 8
      20335 <... brk resumed>)                = 0x56114f6fd000
      20334 brk(NULL <unfinished ...>
      20335 mmap(NULL,32,PROT_READ|PROT_WRITE,MAP_SHARED,5,0 <unfinished
...>
      20334 <... brk resumed>)                = 0x56101d027000
      20335 <... mmap resumed>)               = 0x7ff89f125000
      20334 brk(0x56101d048000 <unfinished ...>
      20335 close(5 <unfinished ...>
      20334 <... brk resumed>)                = 0x56101d048000
      20335 <... close resumed>)              = 0
      20334 mmap(NULL,32,PROT_READ|PROT_WRITE,MAP_SHARED,5,0 <unfinished
...>
      20335
```

```
openat(AT_FDCWD,"/dev/shm/sem.sem_child2_processed",O_RDWR|O_NOFOLLOW
<unfinished ...>
      20334 <... mmap resumed>)               = 0x7f20270fa000
      20335 <... openat resumed>)             = 5
      20334 close(5 <unfinished ...>
      20335 newfstatat(5,"", <unfinished ...>
      20334 <... close resumed>)              = 0
      20335 <... newfstatat
resumed>{st_mode=S_IFREG|0644,st_size=32,...},AT_EMPTY_PATH) = 0
      20334
openat(AT_FDCWD,"/dev/shm/sem.sem_child1_processed",O_RDWR|O_NOFOLLOW
<unfinished ...>
      20335 mmap(NULL,32,PROT_READ|PROT_WRITE,MAP_SHARED,5,0) =
0x7ff89f0eb000
      20334 <... openat resumed>)             = 5
      20335 close(5 <unfinished ...>
      20334 newfstatat(5,"", <unfinished ...>
      20335 <... close resumed>)              = 0
      20334 <... newfstatat
resumed>{st_mode=S_IFREG|0644,st_size=32,...},AT_EMPTY_PATH) = 0
      20335 openat(AT_FDCWD,"/tmp/child2_input.mmap",O_RDWR <unfinished
...>
      20334 mmap(NULL,32,PROT_READ|PROT_WRITE,MAP_SHARED,5,0 <unfinished
...>
      20335 <... openat resumed>)             = 5
      20334 <... mmap resumed>)               = 0x7f20270c0000
      20335 mmap(NULL,1032,PROT_READ|PROT_WRITE,MAP_SHARED,5,0
<unfinished ...>
      20334 close(5 <unfinished ...>
      20335 <... mmap resumed>)               = 0x7ff89f0ea000
      20334 <... close resumed>)              = 0
      20335 openat(AT_FDCWD,"even",O_WRONLY|O_CREAT|O_TRUNC,0666
<unfinished ...>
      20334 openat(AT_FDCWD,"/tmp/child1_input.mmap",O_RDWR) = 5
      20335 <... openat resumed>)             = 6
      20334 mmap(NULL,1032,PROT_READ|PROT_WRITE,MAP_SHARED,5,0
<unfinished ...>
      20335 getppid( <unfinished ...>
      20334 <... mmap resumed>)               = 0x7f20270bf000
      20335 <... getppid resumed>)            = 20333
      20334 openat(AT_FDCWD,"odd",O_WRONLY|O_CREAT|O_TRUNC,0666
<unfinished ...>
      20335 kill(20333,SIGUSR2)              = 0
      20334 <... openat resumed>)             = 6
      20333 <... pause resumed>)              = ? ERESTARTNOHAND (To be
restarted if no handler)
      20335
futex(0x7ff89f125000,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
```

```
<unfinished ...>
     20333 ---SIGUSR2
{si_signo=SIGUSR2,si_code=SI_USER,si_pid=20335,si_uid=1000} ---
     20334 getppid( <unfinished ...>
     20333 rt_sigreturn({mask=[]} <unfinished ...>
     20334 <... getppid resumed>)            = 20333
     20333 <... rt_sigreturn resumed>)       = -1 EINTR (Interrupted
system call)
     20334 kill(20333,SIGUSR1 <unfinished ...>
     20333 pause( <unfinished ...>
     20334 <... kill resumed>)               = 0
     20333 <... pause resumed>)              = ? ERESTARTNOHAND (To be
restarted if no handler)
     20334
futex(0x7f20270fa000,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
<unfinished ...>
     20333 ---SIGUSR1
{si_signo=SIGUSR1,si_code=SI_USER,si_pid=20334,si_uid=1000} ---
     20333 rt_sigreturn({mask=[]})           = -1 EINTR (Interrupted
system call)
     20333
newfstatat(1,"",{st_mode=S_IFCHR|0620,st_rdev=makedev(0x88,0x8),...},AT_EMPTY_PATH)
= 0
     20333
write(1,"\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\321\201\321\202\321\200\320\276\320\272\320\270 (\320\277\321"...,85) =
85
     20333
newfstatat(0,"",{st_mode=S_IFCHR|0620,st_rdev=makedev(0x88,0x8),...},AT_EMPTY_PATH)
= 0
     20333 read(0,"text\n",1024)           = 5
     20333 futex(0x7ff4a428a000,FUTEX_WAKE,1) = 1
     20334 <... futex resumed>)              = 0
     20333
futex(0x7ff4a4289000,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
<unfinished ...>
     20334 write(6,"txet\n",5)              = 5
     20334 futex(0x7f20270c0000,FUTEX_WAKE,1 <unfinished ...>
     20333 <... futex resumed>)              = 0
     20334 <... futex resumed>)              = 1
     20333 read(0, <unfinished ...>
     20334
futex(0x7f20270fa000,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
<unfinished ...>
     20333 <... read resumed>"asd\n",1024)  = 4
     20333 futex(0x7ff4a4287000,FUTEX_WAKE,1) = 1
     20335 <... futex resumed>)              = 0
     20333
```

```
futex(0x7ff4a4286000,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
<unfinished ...>
     20335 write(6,"dsa\n",4)              = 4
     20335 futex(0x7ff89f0eb000,FUTEX_WAKE,1 <unfinished ...>
     20333 <... futex resumed>)             = 0
     20335 <... futex resumed>)             = 1
     20333 read(0, <unfinished ...>
     20335
futex(0x7ff89f125000,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
<unfinished ...>
     20333 <... read resumed>"123das\n",1024) = 7
     20333 futex(0x7ff4a428a000,FUTEX_WAKE,1) = 1
     20334 <... futex resumed>)             = 0
     20333
futex(0x7ff4a4289000,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
<unfinished ...>
     20334 write(6,"sad321\n",7)           = 7
     20334 futex(0x7f20270c0000,FUTEX_WAKE,1 <unfinished ...>
     20333 <... futex resumed>)             = 0
     20334 <... futex resumed>)             = 1
     20333 read(0, <unfinished ...>
     20334
futex(0x7f20270fa000,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
<unfinished ...>
     20333 <... read resumed>"Test\n",1024) = 5
     20333 futex(0x7ff4a4287000,FUTEX_WAKE,1) = 1
     20335 <... futex resumed>)             = 0
     20333
futex(0x7ff4a4286000,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
<unfinished ...>
     20335 write(6,"tseT\n",5)             = 5
     20335 futex(0x7ff89f0eb000,FUTEX_WAKE,1) = 1
     20333 <... futex resumed>)             = 0
     20333 read(0, <unfinished ...>
     20335
futex(0x7ff89f125000,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSET_MATCH
<unfinished ...>
     20333 <... read resumed>"",1024)       = 0
     20333 futex(0x7ff4a428a000,FUTEX_WAKE,1) = 1
     20334 <... futex resumed>)             = 0
     20333 futex(0x7ff4a4287000,FUTEX_WAKE,1 <unfinished ...>
     20334 close(6 <unfinished ...>
     20333 <... futex resumed>)             = 1
     20335 <... futex resumed>)             = 0
     20333 wait4(20334, <unfinished ...>
     20334 <... close resumed>)             = 0
     20335 close(6 <unfinished ...>
     20334 munmap(0x7f20270bf000,1032)      = 0
```

```
      20334 close(5)                            = 0
      20334 munmap(0x7f20270fa000,32 <unfinished ...>
      20335 <... close resumed>)                = 0
      20335 munmap(0x7ff89f0ea000,1032 <unfinished ...>
      20334 <... munmap resumed>)               = 0
      20335 <... munmap resumed>)               = 0
      20334 munmap(0x7f20270c0000,32 <unfinished ...>
      20335 close(5 <unfinished ...>
      20334 <... munmap resumed>)               = 0
      20335 <... close resumed>)                = 0
      20334 exit_group(0 <unfinished ...>
      20335 munmap(0x7ff89f125000,32 <unfinished ...>
      20334 <... exit_group resumed>)           = ?
      20335 <... munmap resumed>)               = 0
      20335 munmap(0x7ff89f0eb000,32 <unfinished ...>
      20334 +++ exited with 0 +++
      20335 <... munmap resumed>)               = 0
      20333 <... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) ==
0}],0,NULL) = 20334
      20335 exit_group(0 <unfinished ...>
      20333 ---SIGCHLD
{si_signo=SIGCHLD,si_code=CLD_EXITED,si_pid=20334,si_uid=1000,si_status=0,si_utime=0,s
---
      20335 <... exit_group resumed>)           = ?
      20333 wait4(20335, <unfinished ...>
      20335 +++ exited with 0 +++
      20333 <... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) ==
0}],0,NULL) = 20335
      20333 ---SIGCHLD
{si_signo=SIGCHLD,si_code=CLD_EXITED,si_pid=20335,si_uid=1000,si_status=0,si_utime=0,s
---
      20333 munmap(0x7ff4a42c4000,1032)         = 0
      20333 close(3)                             = 0
      20333 munmap(0x7ff4a428a000,32)           = 0
      20333 munmap(0x7ff4a4289000,32)           = 0
      20333 unlink("/dev/shm/sem.sem_child1_data_ready") = 0
      20333 unlink("/dev/shm/sem.sem_child1_processed") = 0
      20333 munmap(0x7ff4a4288000,1032)         = 0
      20333 close(4)                             = 0
      20333 munmap(0x7ff4a4287000,32)           = 0
      20333 munmap(0x7ff4a4286000,32)           = 0
      20333 unlink("/dev/shm/sem.sem_child2_data_ready") = 0
      20333 unlink("/dev/shm/sem.sem_child2_processed") = 0
      20333 exit_group(0)                        = ?
      20333 +++ exited with 0 +++
```

Листинг 7: *Strace логи*