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»

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 командную строку подаются два значения: файл для нечетных строк и файл для четных строк. Родительский процесс инициализирует разделяемую память (shared memory) с использованием shm_open() и создает четыре семафора для синхронизации между процессами: семафор для записи родителя, семафор сигнала для первого дочернего процесса, семафор сигнала для второго дочернего процесса и семафор подтверждения завершения обработки. Затем родитель создает два дочерних процесса посредством fork() и execve(). Родительский процесс выполняет главный цикл, в котором читает строки из stdin, определяет четность номера строки и отправляет данные соответствующему дочернему процессу через разделяемую память, используя семафоры для синхронизации. После получения всех данных или при пустой строке родитель сигнализирует обоим дочерним процессам о завершении через флаг shutdown_flag, ожидает завершения обоих процессов и осуществляет очистку ресурсов.

Каждый дочерний процесс открывает общую разделяемую память и семафоры, созданные родителем. Затем создает выходной файл для записи обработанных данных. Дочерний процесс выполняет цикл ожидания, в котором блокируется на своем семафоре сигнала до момента, когда родитель отправляет новые данные. После получения данных процесс проверяет флаг shutdown_flag, и если он установлен, завершает работу. В противном случае он инвертирует полученную строку с помощью локальной функции InvertLine(), записывает результат в выходной файл и сигнализирует родителю о завершении обработки посредством семафора подтверждения.

Описание программы

Входные данные: непустые строки со стандартного входа.

Архитектура решения построена на модульной организации кода с разделением на уровни абстракции. Файл os.h содержит объявления функций-оберток для работы с API POSIX, обеспечивающих независимость от платформы. Реализация для Linux расположена в os_linux.c и использует функции shm_open(), mmap(), sem_open() и другие примитивы синхронизации.

Файл helpers.h объявляет структуры данных для управления IPC ресурсами и вспомогательные функции для инициализации и очистки. В helpers.c реализованы функции

InitParentIpc() и InitChildIpc(), которые устанавливают разделяемую память и семафоры, функция SendDataToChild() для отправки данных с синхронизацией, и функции управления выходными файлами.

В файле parent.c реализуется логика родительского процесса: инициализация IPC, создание двух дочерних процессов с передачей им параметров (идентификатор процесса, имя разделяемой памяти и имя выходного файла), чтение строк со входа и маршрутизация их к соответствующим детям на основе четности номера строки. В файле child.c реализуется логика дочернего процесса: открытие общих ресурсов, создание выходного файла, цикл обработки данных с инверсией строк и запись результатов.

Функция InvertLine() в child.c осуществляет преобразование строки путем переворота символов в обратном порядке с сохранением символа новой строки в конце, если он был присутствен в исходной строке.

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 также в перевернутом виде.

Пример работы:

Ввод:

./parent odd.txt even.txt

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 технологий "shared memory"и "file mapping". Реализована стабильная система обмена данными между процессами, обеспечивающая корректную обработку параллельных операций. Программа демонстрирует практическое применение межпроцессного взаимодействия в системном программировании, включая управление жизненным циклом IPC ресурсов и обработку ошибок на уровне операционной системы.

В ходе выполнения лабораторной работы получены знания:

POSIX API:

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

Многопроцессорное программирование:

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

Системное программирование:

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

Инженерные практики:

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

Исходная программа

```
#pragma once

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

#define _GNU_SOURCE

#ifdef _WIN32

#else
#include <semaphore.h>
#include <signal.h>
#include <string.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 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);

void* MapSharedMemory(int fd, size_t size);

int UnmapMemory(void* addr, size_t size);

int CreateSharedMemory(const char* name, size_t size);

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

```
51
52 int UnlinkSharedMemory(const char* name);
53
54 semaphore_t CreateNamedSemaphore(const char* name, int initial_value);
55
56 semaphore_t OpenNamedSemaphore(const char* name);
57
58 int WaitSemaphore(semaphore_t sem);
59
60 int PostSemaphore(semaphore_t sem);
61
62 int CloseSemaphore(semaphore_t sem);
63
64 int UnlinkSemaphore(const char* name);
```

Листинг 1: *Заголовочный файл для обеспечения кроссплатформенности*

```
1  #include "os.h"
2
3  process_id_t CreateProc(const char* file, char* argv[], char* envp[]) {
4    pid_t pid = fork();
5    if (pid == -1) {
6      perror("fork");
7      return -1;
8    }
9
10   if (pid == 0) {
11     execve(file, argv, envp);
12     perror("execve");
13     TerminateProc(EXIT_FAILURE);
14   }
15
16   return pid;
17 }
18
19 int WaitObject(process_id_t proc_id, int* status, int options) {
20   int res = waitpid(proc_id, status, options);
21   if (res == -1) {
22     perror("waitpid");
23   }
24   return res;
25 }
26
27 void TerminateProc(int status) { _exit(status); }
28
29 process_id_t GetProcessId(void) { return getpid(); }
30
31 process_id_t GetParentProcessId(void) { return getppid(); }
32
33 pipe_t OpenObject(const char* path, int flags, int mode) {
```

```
  int fd = open(path, flags, mode);
  if (fd == -1) {
    perror("open");
  }
  return fd;
}

int CloseObject(pipe_t fd) {
  if (fd < 0) return 0;

  int res = close(fd);
  if (res == -1) {
    perror("close");
  }
  return res;
}

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

void* MapSharedMemory(int fd, size_t size) {
  void* addr = mmap(NULL, size, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
  if (addr == MAP_FAILED) {
    perror("mmap");
    return NULL;
  }
  return addr;
}

int UnmapMemory(void* addr, size_t size) {
  if (addr == NULL) return 0;

  int res = munmap(addr, size);
  if (res == -1) {
    perror("munmap");
  }
  return res;
}

int CreateSharedMemory(const char* name, size_t size) {
  shm_unlink(name);

  int fd = shm_open(name, O_CREAT | O_RDWR | O_EXCL, 0666);
  if (fd == -1) {
    perror("shm_open");
    return -1;
  }

  if (ftruncate(fd, size) == -1) {
    perror("ftruncate");
```

```
85      close(fd);
86      shm_unlink(name);
87      return -1;
88    }
89
90    return fd;
91  }
92
93  int OpenSharedMemory(const char* name) {
94    int fd = shm_open(name, O_RDWR, 0);
95    if (fd == -1) {
96      perror("shm_open");
97    }
98    return fd;
99  }
100
101 int UnlinkSharedMemory(const char* name) {
102   int res = shm_unlink(name);
103   if (res == -1) {
104     perror("shm_unlink");
105   }
106   return res;
107 }
108
109 semaphore_t CreateNamedSemaphore(const char* name, int initial_value) {
110   sem_unlink(name);
111
112   semaphore_t sem = sem_open(name, O_CREAT | O_EXCL, 0666, initial_value);
113   if (sem == SEM_FAILED) {
114     perror("sem_open");
115     return NULL;
116   }
117   return sem;
118 }
119
120 semaphore_t OpenNamedSemaphore(const char* name) {
121   semaphore_t sem = sem_open(name, 0);
122   if (sem == SEM_FAILED) {
123     perror("sem_open");
124     return NULL;
125   }
126   return sem;
127 }
128
129 int WaitSemaphore(semaphore_t sem) {
130   if (sem == NULL) {
131     fprintf(stderr, "WaitSemaphore: NULL semaphore\n");
132     return -1;
133   }
134
135   int res = sem_wait(sem);
```

```
  if (res == -1) {
    perror("sem_wait");
  }
  return res;
}

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

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

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

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

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

Листинг 2: *Реализация функций-оберток (только POSIX системы)*

```
#pragma once

#include "os.h"

#define MAX_LINE_LENGTH 1024
#define SHM_NAME "/ipc_buffer"
#define SHM_SIZE sizeof(shared_buffer_t)

typedef struct {
  volatile int shutdown_flag;
  volatile int line_length;
```

```
  char data[MAX_LINE_LENGTH];
} shared_buffer_t;

typedef struct {
  int shm_fd;
  void* shm_addr;
  semaphore_t sem_parent_write;
  semaphore_t sem_child1_read;
  semaphore_t sem_child2_read;
  semaphore_t sem_ack;
} ipc_handles_t;

typedef struct {
  int id;
  const char* shm_name;
  const char* output_file;
} child_config_t;

ipc_handles_t InitParentIpc(const char* shm_name, size_t shm_size);

ipc_handles_t InitChildIpc(const char* shm_name, size_t shm_size);

void CleanupIpc(ipc_handles_t* handles, size_t shm_size);

void CleanupIpcFull(ipc_handles_t* handles, const char* shm_name,
                    size_t shm_size);

int SendDataToChild(shared_buffer_t* buf, const char* data, size_t len,
                    int child_id, ipc_handles_t* ipc);

void SignalChildShutdown(shared_buffer_t* buf, ipc_handles_t* ipc);

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

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

int CreateOutputFile(const child_config_t* config);

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

int CloseOutputFile(int fd);
```

Листинг 3: *Заголовочный файл для вспомогательных функций*

```
#include "helpers.h"

#include <string.h>

ipc_handles_t InitParentIpc(const char* shm_name, size_t shm_size) {
```

```
  ipc_handles_t handles = {0};

  handles.shm_fd = CreateSharedMemory(shm_name, shm_size);
  if (handles.shm_fd == -1) {
    fprintf(stderr, "[Parent] Failed to create shared memory\n");
    return handles;
  }

  handles.shm_addr = MapSharedMemory(handles.shm_fd, shm_size);
  if (handles.shm_addr == NULL) {
    fprintf(stderr, "[Parent] Failed to map shared memory\n");
    CloseObject(handles.shm_fd);
    return handles;
  }

  shared_buffer_t* buf = (shared_buffer_t*)handles.shm_addr;
  memset(buf, 0, shm_size);

  handles.sem_parent_write = CreateNamedSemaphore("/sem_parent_write", 1);
  handles.sem_child1_read = CreateNamedSemaphore("/sem_child1_read", 0);
  handles.sem_child2_read = CreateNamedSemaphore("/sem_child2_read", 0);
  handles.sem_ack = CreateNamedSemaphore("/sem_ack", 0);

  if (!handles.sem_parent_write || !handles.sem_child1_read ||
      !handles.sem_child2_read || !handles.sem_ack) {
    fprintf(stderr, "[Parent] Failed to create semaphores\n");
    UnmapMemory(handles.shm_addr, shm_size);
    CloseObject(handles.shm_fd);
    handles.shm_fd = -1;
    return handles;
  }

  return handles;
}

ipc_handles_t InitChildIpc(const char* shm_name, size_t shm_size) {
  ipc_handles_t handles = {0};

  handles.shm_fd = OpenSharedMemory(shm_name);
  if (handles.shm_fd == -1) {
    fprintf(stderr, "[Child] Failed to open shared memory\n");
    return handles;
  }

  handles.shm_addr = MapSharedMemory(handles.shm_fd, shm_size);
  if (handles.shm_addr == NULL) {
    fprintf(stderr, "[Child] Failed to map shared memory\n");
    CloseObject(handles.shm_fd);
    return handles;
  }

```

```
57    handles.sem_parent_write = OpenNamedSemaphore("/sem_parent_write");
58    handles.sem_child1_read = OpenNamedSemaphore("/sem_child1_read");
59    handles.sem_child2_read = OpenNamedSemaphore("/sem_child2_read");
60    handles.sem_ack = OpenNamedSemaphore("/sem_ack");
61
62    if (!handles.sem_parent_write || !handles.sem_child1_read ||
63        !handles.sem_child2_read || !handles.sem_ack) {
64      fprintf(stderr, "[Child] Failed to open semaphores\n");
65      UnmapMemory(handles.shm_addr, shm_size);
66      CloseObject(handles.shm_fd);
67      handles.shm_fd = -1;
68      return handles;
69    }
70
71    return handles;
72  }
73
74  int SendDataToChild(shared_buffer_t* buf, const char* data, size_t len,
75                      int child_id, ipc_handles_t* ipc) {
76    if (len > MAX_LINE_LENGTH - 1) {
77      fprintf(stderr, "[Parent] String too long\n");
78      return -1;
79    }
80
81    if (WaitSemaphore(ipc->sem_parent_write) == -1) return -1;
82
83    memcpy((void*)buf->data, data, len);
84    buf->line_length = len;
85
86    semaphore_t child_sem =
87        (child_id == 1) ? ipc->sem_child1_read : ipc->sem_child2_read;
88
89    if (PostSemaphore(child_sem) == -1) {
90      PostSemaphore(ipc->sem_parent_write);
91      return -1;
92    }
93
94    if (WaitSemaphore(ipc->sem_ack) == -1) {
95      PostSemaphore(ipc->sem_parent_write);
96      return -1;
97    }
98
99    buf->line_length = 0;
100   PostSemaphore(ipc->sem_parent_write);
101
102   return 0;
103 }
104
105 void SignalChildShutdown(shared_buffer_t* buf, ipc_handles_t* ipc) {
106   WaitSemaphore(ipc->sem_parent_write);
107   buf->shutdown_flag = 1;
```

```
108   PostSemaphore(ipc->sem_child1_read);
109   PostSemaphore(ipc->sem_child2_read);
110 }
111
112 void WaitForChildren(process_id_t pid1, process_id_t pid2, int* status1,
113                      int* status2) {
114   *status1 = 0;
115   *status2 = 0;
116   WaitObject(pid1, status1, 0);
117   WaitObject(pid2, status2, 0);
118 }
119
120 int InitChildConfig(int argc, char* argv[], child_config_t* config) {
121   if (argc != 4) {
122     fprintf(stderr, "Child: incorrect argument count\n");
123     return -1;
124   }
125
126   config->id = atoi(argv[1]);
127   config->shm_name = argv[2];
128   config->output_file = argv[3];
129
130   if (config->id != 1 && config->id != 2) {
131     fprintf(stderr, "Child: invalid process ID (must be 1 or 2)\n");
132     return -1;
133   }
134
135   return 0;
136 }
137
138 int CreateOutputFile(const child_config_t* config) {
139   int fd = OpenObject(config->output_file, O_CREAT | O_WRONLY | O_TRUNC, 0666);
140   if (fd == -1) {
141     fprintf(stderr, "[Child$d]Failedtocreateoutputfile :$s\n",
142             config->id, config->output_file);
143   }
144   return fd;
145 }
146
147 int WriteToOutput(int fd, const char* data, size_t data_len) {
148   ssize_t written = WriteToObject(fd, data, data_len);
149   if (written == -1 || (size_t)written != data_len) return -1;
150   return 0;
151 }
152
153 int CloseOutputFile(int fd) {
154   if (fd < 0) return 0;
155   return CloseObject(fd);
156 }
157
158 void CleanupIpc(ipc_handles_t* handles, size_t shm_size) {
```

```
  if (handles == NULL) return;
  CloseSemaphore(handles->sem_parent_write);
  CloseSemaphore(handles->sem_child1_read);
  CloseSemaphore(handles->sem_child2_read);
  CloseSemaphore(handles->sem_ack);
  UnmapMemory(handles->shm_addr, shm_size);
  CloseObject(handles->shm_fd);
}

void CleanupIpcFull(ipc_handles_t* handles, const char* shm_name,
                    size_t shm_size) {
  CleanupIpc(handles, shm_size);
  UnlinkSemaphore("/sem_parent_write");
  UnlinkSemaphore("/sem_child1_read");
  UnlinkSemaphore("/sem_child2_read");
  UnlinkSemaphore("/sem_ack");
  UnlinkSharedMemory(shm_name);
}
```

Листинг 4: *Реализация вспомогательных функций*

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

#include "helpers.h"

int main(int argc, char* argv[], char* envp[]) {
  if (argc != 3) {
    fprintf(stderr, "Usage: $
    s < file1 >< file2 > argv[0]); return EXIT_FAILURE; ipc_handles_t ipc =
    InitParentIpc(SHM_NAME, SHM_SIZE); if (ipc.shm_fd ==
    -1) fprintf(stderr, "[Parent] Failed to initialize IPC"); return EXIT_FAILURE; char *
    args1[] = "child" 1 (char*)SHM_NAME, argv[1], NULL; process_id_t pid1 =
    CreateProc("./child args1, envp); if (pid1 ==
    -1) fprintf(stderr, "[Parent] Failed to create child 1"); CleanupIpcFull(ipc, SHM_NAME, SHM_SIZE); re
    args2[] = "child" 2 (char*)SHM_NAME, argv[2], NULL; process_id_t pid2 =
    CreateProc("./child args2, envp); if (pid2 ==
    -1) fprintf(stderr, "[Parent] Failed to create child 2"); CleanupIpcFull(ipc, SHM_NAME, SHM_SIZE); re
    shared_buf =
    (shared_buffer_t*)ipc.shm_addr; fprintf(stdout, "Enter lines (empty line or Ctrl + D to quit) :
    "); fflush(stdout); char line[MAX_LINE_LENGTH]; int line_number =
    0; while (fgets(line, sizeof(line), stdin) ! =
    NULL) if (line[0] == '
')break; line_number + +; size_t len = strlen(line); int target_child = (line_number
     $ 2 == 1) ? 1 : 2;

    if (SendDataToChild(shared_buf, line, len, target_child, &ipc) == -1) {
      fprintf(stderr, "[Parent] Failed to send data\n");
      break;
    }
```

```
  }

  SignalChildShutdown(shared_buf, &ipc);

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

  CleanupIpcFull(&ipc, SHM_NAME, SHM_SIZE);

  return EXIT_SUCCESS;
}
```

Листинг 5: *Родительский процесс*

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

#include "helpers.h"

static int InvertLine(const char* input, int input_len, char* output) {
  int out_idx = 0;

  int line_end = input_len;
  if (input_len > 0 && input[input_len - 1] == '\n') {
    line_end = input_len - 1;
  }

  for (int i = line_end - 1; i >= 0; i--) {
    output[out_idx++] = input[i];
  }

  if (out_idx < MAX_LINE_LENGTH - 1) {
    output[out_idx++] = '\n';
  }

  return out_idx;
}

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

  ipc_handles_t ipc = InitChildIpc(config.shm_name, SHM_SIZE);
  if (ipc.shm_fd == -1) return EXIT_FAILURE;

  int output_fd = CreateOutputFile(&config);
  if (output_fd == -1) {
    CleanupIpc(&ipc, SHM_SIZE);
    return EXIT_FAILURE;
```

```
  }

  shared_buffer_t* shared_buf = (shared_buffer_t*)ipc.shm_addr;

  semaphore_t sem_read =
      (config.id == 1) ? ipc.sem_child1_read : ipc.sem_child2_read;

  while (1) {
    if (WaitSemaphore(sem_read) == -1) break;

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

    int line_len = shared_buf->line_length;
    if (line_len > 0 && line_len < MAX_LINE_LENGTH) {
      char output_buffer[MAX_LINE_LENGTH];
      int output_len = InvertLine(shared_buf->data, line_len, output_buffer);

      if (output_len > 0) {
        WriteToOutput(output_fd, output_buffer, output_len);
      }
    }

    if (PostSemaphore(ipc.sem_ack) == -1) break;
  }

  CloseOutputFile(output_fd);
  CleanupIpc(&ipc, SHM_SIZE);

  return EXIT_SUCCESS;
}
```

Листинг 6: *Дочерний процесс*

```
92290 execve("./parent", ["./parent", "file1.txt", "file2.txt"], 0x7ffe431078a8
    /* 36 vars */) = 0
92290 brk(NULL)                           = 0x55e86f011000
92290 arch_prctl(0x3001 /* ARCH_??? */, 0x7fff28a961d0) = -1 EINVAL (Invalid
    argument)
92290 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
     0x7fea9c108000
92290 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
92290 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
92290 newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=30076, ...},
    AT_EMPTY_PATH) = 0
92290 mmap(NULL, 30076, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fea9c100000
92290 close(3)                            = 0
92290 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) =
    3
92290 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
```

```
92290 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
92290 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
92290 pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O{\f\225\\=\201\327\312\301P\32$
    \230\266\235"..., 68, 896) = 68
92290 newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...},
    AT_EMPTY_PATH) = 0
92290 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
92290 mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0
    x7fea9bed7000
92290 mprotect(0x7fea9beff000, 2023424, PROT_NONE) = 0
92290 mmap(0x7fea9beff000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x28000) = 0x7fea9beff000
92290 mmap(0x7fea9c094000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x1bd000) = 0x7fea9c094000
92290 mmap(0x7fea9c0ed000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x215000) = 0x7fea9c0ed000
92290 mmap(0x7fea9c0f3000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
    MAP_ANONYMOUS, -1, 0) = 0x7fea9c0f3000
92290 close(3)                          = 0
92290 mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
    = 0x7fea9bed4000
92290 arch_prctl(ARCH_SET_FS, 0x7fea9bed4740) = 0
92290 set_tid_address(0x7fea9bed4a10)   = 92290
92290 set_robust_list(0x7fea9bed4a20, 24) = 0
92290 rseq(0x7fea9bed50e0, 0x20, 0, 0x53053053) = 0
92290 mprotect(0x7fea9c0ed000, 16384, PROT_READ) = 0
92290 mprotect(0x55e83200c000, 4096, PROT_READ) = 0
92290 mprotect(0x7fea9c142000, 8192, PROT_READ) = 0
92290 prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=
    RLIM64_INFINITY}) = 0
92290 munmap(0x7fea9c100000, 30076)     = 0
92290 unlink("/dev/shm/ipc_buffer")     = 0
92290 openat(AT_FDCWD, "/dev/shm/ipc_buffer", O_RDWR|O_CREAT|O_EXCL|O_NOFOLLOW|
    O_CLOEXEC, 0666) = 3
92290 ftruncate(3, 1032)                = 0
92290 mmap(NULL, 1032, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7fea9c141000
92290 unlink("/dev/shm/sem.sem_parent_write") = 0
92290 getrandom("\x68\x99\x08\xe1\x95\x58\x7b\x5e", 8, GRND_NONBLOCK) = 8
92290 newfstatat(AT_FDCWD, "/dev/shm/sem.QVvzXy", 0x7fff28a95980,
    AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)
92290 openat(AT_FDCWD, "/dev/shm/sem.QVvzXy", O_RDWR|O_CREAT|O_EXCL, 0666) = 4
92290 write(4, "\1\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\0
    \0", 32) = 32
92290 mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7fea9c107000
92290 link("/dev/shm/sem.QVvzXy", "/dev/shm/sem.sem_parent_write") = 0
92290 newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH)
    = 0
92290 getrandom("\x99\xbe\x5d\xf4\x92\x6d\x4e\x8a", 8, GRND_NONBLOCK) = 8
```

```
92290 brk(NULL)                           = 0x55e86f011000
92290 brk(0x55e86f032000)                 = 0x55e86f032000
92290 unlink("/dev/shm/sem.QVvzXy")       = 0
92290 close(4)                            = 0
92290 unlink("/dev/shm/sem.sem_child1_read") = 0
92290 getrandom("\xcd\xca\x63\xad\x10\xa6\xb4\x95", 8, GRND_NONBLOCK) = 8
92290 newfstatat(AT_FDCWD, "/dev/shm/sem.7h6jcD", 0x7fff28a95980,
    AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)
92290 openat(AT_FDCWD, "/dev/shm/sem.7h6jcD", O_RDWR|O_CREAT|O_EXCL, 0666) = 4
92290 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
    \0", 32) = 32
92290 mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7fea9c106000
92290 link("/dev/shm/sem.7h6jcD", "/dev/shm/sem.sem_child1_read") = 0
92290 newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH)
    = 0
92290 unlink("/dev/shm/sem.7h6jcD")       = 0
92290 close(4)                            = 0
92290 unlink("/dev/shm/sem.sem_child2_read") = 0
92290 getrandom("\x4a\x9f\x89\xfd\xe4\x06\xe4\xcd", 8, GRND_NONBLOCK) = 8
92290 newfstatat(AT_FDCWD, "/dev/shm/sem.qH3AFm", 0x7fff28a95980,
    AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)
92290 openat(AT_FDCWD, "/dev/shm/sem.qH3AFm", O_RDWR|O_CREAT|O_EXCL, 0666) = 4
92290 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
    \0", 32) = 32
92290 mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7fea9c105000
92290 link("/dev/shm/sem.qH3AFm", "/dev/shm/sem.sem_child2_read") = 0
92290 newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH)
    = 0
92290 unlink("/dev/shm/sem.qH3AFm")       = 0
92290 close(4)                            = 0
92290 unlink("/dev/shm/sem.sem_ack")      = 0
92290 getrandom("\x70\x0f\xb8\x68\x7a\x9b\x3c\x23", 8, GRND_NONBLOCK) = 8
92290 newfstatat(AT_FDCWD, "/dev/shm/sem.qwX17C", 0x7fff28a95980,
    AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)
92290 openat(AT_FDCWD, "/dev/shm/sem.qwX17C", O_RDWR|O_CREAT|O_EXCL, 0666) = 4
92290 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
    \0", 32) = 32
92290 mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7fea9c104000
92290 link("/dev/shm/sem.qwX17C", "/dev/shm/sem.sem_ack") = 0
92290 newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH)
    = 0
92290 unlink("/dev/shm/sem.qwX17C")       = 0
92290 close(4)                            = 0
92290 clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|
    SIGCHLD, child_tidptr=0x7fea9bed4a10) = 92291
92291 set_robust_list(0x7fea9bed4a20, 24 <unfinished ...>
92290 clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|
    SIGCHLD <unfinished ...>
92291 <... set_robust_list resumed>)      = 0
92291 execve("./child", ["child", "1", "/ipc_buffer", "file1.txt"], 0
    x7fff28a963b8 /* 36 vars */ <unfinished ...>
```

```
92290 <... clone resumed>, child_tidptr=0x7fea9bed4a10) = 92292
92292 set_robust_list(0x7fea9bed4a20, 24 <unfinished ...>
92290 clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=1, tv_nsec=0},  <unfinished
    ...>
92292 <... set_robust_list resumed>)    = 0
92292 execve("./child", ["child", "2", "/ipc_buffer", "file2.txt"], 0
    x7fff28a963b8 /* 36 vars */ <unfinished ...>
92291 <... execve resumed>)             = 0
92291 brk(NULL <unfinished ...>
92292 <... execve resumed>)             = 0
92291 <... brk resumed>)                = 0x56314f8fc000
92292 brk(NULL <unfinished ...>
92291 arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe0ffd1770 <unfinished ...>
92292 <... brk resumed>)                = 0x556bff424000
92291 <... arch_prctl resumed>)         = -1 EINVAL (Invalid argument)
92292 arch_prctl(0x3001 /* ARCH_??? */, 0x7ffce061f730 <unfinished ...>
92291 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <
    unfinished ...>
92292 <... arch_prctl resumed>)         = -1 EINVAL (Invalid argument)
92291 <... mmap resumed>)               = 0x7f6170aa8000
92292 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <
    unfinished ...>
92291 access("/etc/ld.so.preload", R_OK <unfinished ...>
92292 <... mmap resumed>)               = 0x7f7aad206000
92291 <... access resumed>)             = -1 ENOENT (No such file or directory)
92292 access("/etc/ld.so.preload", R_OK <unfinished ...>
92291 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
92292 <... access resumed>)             = -1 ENOENT (No such file or directory)
92291 <... openat resumed>)             = 3
92292 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
92291 newfstatat(3, "",  <unfinished ...>
92292 <... openat resumed>)             = 3
92291 <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=30076, ...},
    AT_EMPTY_PATH) = 0
92292 newfstatat(3, "",  <unfinished ...>
92291 mmap(NULL, 30076, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>
92292 <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=30076, ...},
    AT_EMPTY_PATH) = 0
92291 <... mmap resumed>)               = 0x7f6170aa0000
92292 mmap(NULL, 30076, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>
92291 close(3 <unfinished ...>
92292 <... mmap resumed>)               = 0x7f7aad1fe000
92291 <... close resumed>)              = 0
92292 close(3 <unfinished ...>
92291 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC <
    unfinished ...>
92292 <... close resumed>)              = 0
92292 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC <
    unfinished ...>
92291 <... openat resumed>)             = 3
92292 <... openat resumed>)             = 3
```

```
92291 read(3,  <unfinished ...>
92292 read(3,  <unfinished ...>
92291 <... 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
92292 <... 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
92291 pread64(3,  <unfinished ...>
92292 pread64(3,  <unfinished ...>
92291 <... 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
92292 <... 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
92291 pread64(3,  <unfinished ...>
92292 pread64(3,  <unfinished ...>
92291 <... 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
92292 <... 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
92291 pread64(3,  <unfinished ...>
92292 pread64(3,  <unfinished ...>
92291 <... pread64 resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O{\f\225\\=\201\327\
    312\301P\32$\230\266\235"..., 68, 896) = 68
92292 <... pread64 resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O{\f\225\\=\201\327\
    312\301P\32$\230\266\235"..., 68, 896) = 68
92291 newfstatat(3, "",  <unfinished ...>
92292 newfstatat(3, "",  <unfinished ...>
92291 <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...},
    AT_EMPTY_PATH) = 0
92292 <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...},
    AT_EMPTY_PATH) = 0
92292 pread64(3,  <unfinished ...>
92291 pread64(3,  <unfinished ...>
92292 <... 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
92291 <... 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
92292 mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished
     ...>
92291 mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished
     ...>
92292 <... mmap resumed>)               = 0x7f7aacfd5000
92291 <... mmap resumed>)               = 0x7f6170877000
92292 mprotect(0x7f7aacffd000, 2023424, PROT_NONE <unfinished ...>
92291 mprotect(0x7f617089f000, 2023424, PROT_NONE <unfinished ...>
92292 <... mprotect resumed>)           = 0
92291 <... mprotect resumed>)           = 0
92292 mmap(0x7f7aacffd000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x28000 <unfinished ...>
92291 mmap(0x7f617089f000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x28000 <unfinished ...>
92292 <... mmap resumed>)               = 0x7f7aacffd000
```

```
92292 mmap(0x7f7aad192000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x1bd000 <unfinished ...>
92291 <... mmap resumed>)               = 0x7f617089f000
92292 <... mmap resumed>)               = 0x7f7aad192000
92291 mmap(0x7f6170a34000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x1bd000 <unfinished ...>
92292 mmap(0x7f7aad1eb000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x215000 <unfinished ...>
92291 <... mmap resumed>)               = 0x7f6170a34000
92292 <... mmap resumed>)               = 0x7f7aad1eb000
92291 mmap(0x7f6170a8d000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x215000 <unfinished ...>
92292 mmap(0x7f7aad1f1000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
    MAP_ANONYMOUS, -1, 0) = 0x7f7aad1f1000
92291 <... mmap resumed>)               = 0x7f6170a8d000
92292 close(3)                          = 0
92291 mmap(0x7f6170a93000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
    MAP_ANONYMOUS, -1, 0 <unfinished ...>
92292 mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
    = 0x7f7aacfd2000
92291 <... mmap resumed>)               = 0x7f6170a93000
92292 arch_prctl(ARCH_SET_FS, 0x7f7aacfd2740 <unfinished ...>
92291 close(3 <unfinished ...>
92292 <... arch_prctl resumed>)         = 0
92291 <... close resumed>)              = 0
92292 set_tid_address(0x7f7aacfd2a10)   = 92292
92291 mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <
    unfinished ...>
92292 set_robust_list(0x7f7aacfd2a20, 24 <unfinished ...>
92291 <... mmap resumed>)               = 0x7f6170874000
92292 <... set_robust_list resumed>)    = 0
92292 rseq(0x7f7aacfd30e0, 0x20, 0, 0x53053053 <unfinished ...>
92291 arch_prctl(ARCH_SET_FS, 0x7f6170874740 <unfinished ...>
92292 <... rseq resumed>)               = 0
92291 <... arch_prctl resumed>)         = 0
92291 set_tid_address(0x7f6170874a10 <unfinished ...>
92292 mprotect(0x7f7aad1eb000, 16384, PROT_READ <unfinished ...>
92291 <... set_tid_address resumed>)    = 92291
92292 <... mprotect resumed>)           = 0
92291 set_robust_list(0x7f6170874a20, 24 <unfinished ...>
92292 mprotect(0x556bfe938000, 4096, PROT_READ <unfinished ...>
92291 <... set_robust_list resumed>)    = 0
92292 <... mprotect resumed>)           = 0
92291 rseq(0x7f61708750e0, 0x20, 0, 0x53053053 <unfinished ...>
92292 mprotect(0x7f7aad240000, 8192, PROT_READ <unfinished ...>
92291 <... rseq resumed>)               = 0
92292 <... mprotect resumed>)           = 0
92292 prlimit64(0, RLIMIT_STACK, NULL,  <unfinished ...>
92291 mprotect(0x7f6170a8d000, 16384, PROT_READ <unfinished ...>
92292 <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
92291 <... mprotect resumed>)           = 0
```

```
92292 munmap(0x7f7aad1fe000, 30076 <unfinished ...>
92291 mprotect(0x563128bee000, 4096, PROT_READ <unfinished ...>
92292 <... munmap resumed>)             = 0
92291 <... mprotect resumed>)           = 0
92292 openat(AT_FDCWD, "/dev/shm/ipc_buffer", O_RDWR|O_NOFOLLOW|O_CLOEXEC <
    unfinished ...>
92291 mprotect(0x7f6170ae2000, 8192, PROT_READ <unfinished ...>
92292 <... openat resumed>)             = 3
92291 <... mprotect resumed>)           = 0
92292 mmap(NULL, 1032, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0 <unfinished ...>
92291 prlimit64(0, RLIMIT_STACK, NULL,  <unfinished ...>
92292 <... mmap resumed>)               = 0x7f7aad23f000
92291 <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
92292 openat(AT_FDCWD, "/dev/shm/sem.sem_parent_write", O_RDWR|O_NOFOLLOW <
    unfinished ...>
92291 munmap(0x7f6170aa0000, 30076 <unfinished ...>
92292 <... openat resumed>)             = 4
92291 <... munmap resumed>)             = 0
92292 newfstatat(4, "",  <unfinished ...>
92291 openat(AT_FDCWD, "/dev/shm/ipc_buffer", O_RDWR|O_NOFOLLOW|O_CLOEXEC <
    unfinished ...>
92292 <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=32, ...},
    AT_EMPTY_PATH) = 0
92291 <... openat resumed>)             = 3
92292 getrandom( <unfinished ...>
92291 mmap(NULL, 1032, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0 <unfinished ...>
92292 <... getrandom resumed>"\x56\xf5\xd5\x1f\x1c\x5f\xd2\x19", 8,
    GRND_NONBLOCK) = 8
92291 <... mmap resumed>)               = 0x7f6170ae1000
92292 brk(NULL <unfinished ...>
92291 openat(AT_FDCWD, "/dev/shm/sem.sem_parent_write", O_RDWR|O_NOFOLLOW <
    unfinished ...>
92292 <... brk resumed>)                = 0x556bff424000
92291 <... openat resumed>)             = 4
92292 brk(0x556bff445000 <unfinished ...>
92291 newfstatat(4, "",  <unfinished ...>
92292 <... brk resumed>)                = 0x556bff445000
92291 <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=32, ...},
    AT_EMPTY_PATH) = 0
92292 mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0 <unfinished ...>
92291 getrandom( <unfinished ...>
92292 <... mmap resumed>)               = 0x7f7aad205000
92291 <... getrandom resumed>"\xea\xe6\x45\xf2\x62\x7b\x0c\xb2", 8,
    GRND_NONBLOCK) = 8
92292 close(4 <unfinished ...>
92291 brk(NULL <unfinished ...>
92292 <... close resumed>)              = 0
92291 <... brk resumed>)                = 0x56314f8fc000
92292 openat(AT_FDCWD, "/dev/shm/sem.sem_child1_read", O_RDWR|O_NOFOLLOW <
    unfinished ...>
92291 brk(0x56314f91d000 <unfinished ...>
```

```
92292 <... openat resumed>)             = 4
92291 <... brk resumed>)                = 0x56314f91d000
92292 newfstatat(4, "",  <unfinished ...>
92291 mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0 <unfinished ...>
92292 <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=32, ...},
    AT_EMPTY_PATH) = 0
92291 <... mmap resumed>)               = 0x7f6170aa7000
92292 mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0 <unfinished ...>
92291 close(4 <unfinished ...>
92292 <... mmap resumed>)               = 0x7f7aad204000
92291 <... close resumed>)              = 0
92292 close(4 <unfinished ...>
92291 openat(AT_FDCWD, "/dev/shm/sem.sem_child1_read", O_RDWR|O_NOFOLLOW <
    unfinished ...>
92292 <... close resumed>)              = 0
92292 openat(AT_FDCWD, "/dev/shm/sem.sem_child2_read", O_RDWR|O_NOFOLLOW <
    unfinished ...>
92291 <... openat resumed>)             = 4
92292 <... openat resumed>)             = 4
92291 newfstatat(4, "",  <unfinished ...>
92292 newfstatat(4, "",  <unfinished ...>
92291 <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=32, ...},
    AT_EMPTY_PATH) = 0
92292 <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=32, ...},
    AT_EMPTY_PATH) = 0
92291 mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0 <unfinished ...>
92292 mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0 <unfinished ...>
92291 <... mmap resumed>)               = 0x7f6170aa6000
92292 <... mmap resumed>)               = 0x7f7aad203000
92291 close(4 <unfinished ...>
92292 close(4 <unfinished ...>
92291 <... close resumed>)              = 0
92292 <... close resumed>)              = 0
92291 openat(AT_FDCWD, "/dev/shm/sem.sem_child2_read", O_RDWR|O_NOFOLLOW <
    unfinished ...>
92292 openat(AT_FDCWD, "/dev/shm/sem.sem_ack", O_RDWR|O_NOFOLLOW <unfinished
    ...>
92291 <... openat resumed>)             = 4
92292 <... openat resumed>)             = 4
92291 newfstatat(4, "",  <unfinished ...>
92292 newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH)
    = 0
92291 <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=32, ...},
    AT_EMPTY_PATH) = 0
92292 mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0 <unfinished ...>
92291 mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0 <unfinished ...>
92292 <... mmap resumed>)               = 0x7f7aad202000
92291 <... mmap resumed>)               = 0x7f6170aa5000
92292 close(4 <unfinished ...>
92291 close(4 <unfinished ...>
92292 <... close resumed>)              = 0
```

```
92291 <... close resumed>)              = 0
92292 openat(AT_FDCWD, "file2.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666 <unfinished
    ...>
92291 openat(AT_FDCWD, "/dev/shm/sem.sem_ack", O_RDWR|O_NOFOLLOW <unfinished
    ...>
92292 <... openat resumed>)             = 4
92291 <... openat resumed>)             = 4
92292 futex(0x7f7aad203000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
    FUTEX_BITSET_MATCH_ANY <unfinished ...>
92291 newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH)
    = 0
92291 mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7f6170aa4000
92291 close(4)                          = 0
92291 openat(AT_FDCWD, "file1.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 4
92291 futex(0x7f6170aa6000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
    FUTEX_BITSET_MATCH_ANY <unfinished ...>
92290 <... clock_nanosleep resumed>0x7fff28a95d50) = 0
92290 newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x9), ...},
     AT_EMPTY_PATH) = 0
92290 write(1, "Enter lines (empty line or Ctrl+"..., 44) = 44
92290 newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x9), ...},
     AT_EMPTY_PATH) = 0
92290 read(0, "text\n", 1024)           = 5
92290 futex(0x7fea9c106000, FUTEX_WAKE, 1) = 1
92291 <... futex resumed>)              = 0
92290 futex(0x7fea9c104000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
    FUTEX_BITSET_MATCH_ANY <unfinished ...>
92291 write(4, "txet\n", 5)             = 5
92291 futex(0x7f6170aa4000, FUTEX_WAKE, 1 <unfinished ...>
92290 <... futex resumed>)              = 0
92291 <... futex resumed>)              = 1
92290 read(0,  <unfinished ...>
92291 futex(0x7f6170aa6000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
    FUTEX_BITSET_MATCH_ANY <unfinished ...>
92290 <... read resumed>"asd\n", 1024)  = 4
92290 futex(0x7fea9c105000, FUTEX_WAKE, 1) = 1
92292 <... futex resumed>)              = 0
92290 futex(0x7fea9c104000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
    FUTEX_BITSET_MATCH_ANY <unfinished ...>
92292 write(4, "dsa\n", 4)              = 4
92292 futex(0x7f7aad202000, FUTEX_WAKE, 1 <unfinished ...>
92290 <... futex resumed>)              = 0
92292 <... futex resumed>)              = 1
92290 read(0,  <unfinished ...>
92292 futex(0x7f7aad203000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
    FUTEX_BITSET_MATCH_ANY <unfinished ...>
92290 <... read resumed>"123das\n", 1024) = 7
92290 futex(0x7fea9c106000, FUTEX_WAKE, 1) = 1
92291 <... futex resumed>)              = 0
92290 futex(0x7fea9c104000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
    FUTEX_BITSET_MATCH_ANY <unfinished ...>
```

```
92291 write(4, "sad321\n", 7)           = 7
92291 futex(0x7f6170aa4000, FUTEX_WAKE, 1 <unfinished ...>
92290 <... futex resumed>)              = 0
92291 <... futex resumed>)              = 1
92290 read(0,  <unfinished ...>
92291 futex(0x7f6170aa6000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
    FUTEX_BITSET_MATCH_ANY <unfinished ...>
92290 <... read resumed>"Test\n", 1024) = 5
92290 futex(0x7fea9c105000, FUTEX_WAKE, 1) = 1
92292 <... futex resumed>)              = 0
92290 futex(0x7fea9c104000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
    FUTEX_BITSET_MATCH_ANY <unfinished ...>
92292 write(4, "tseT\n", 5)             = 5
92292 futex(0x7f7aad202000, FUTEX_WAKE, 1 <unfinished ...>
92290 <... futex resumed>)              = 0
92292 <... futex resumed>)              = 1
92290 read(0,  <unfinished ...>
92292 futex(0x7f7aad203000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
    FUTEX_BITSET_MATCH_ANY <unfinished ...>
92290 <... read resumed>"", 1024)       = 0
92290 futex(0x7fea9c106000, FUTEX_WAKE, 1) = 1
92291 <... futex resumed>)              = 0
92290 futex(0x7fea9c105000, FUTEX_WAKE, 1 <unfinished ...>
92291 close(4 <unfinished ...>
92290 <... futex resumed>)              = 1
92292 <... futex resumed>)              = 0
92290 wait4(92291,  <unfinished ...>
92292 close(4 <unfinished ...>
92291 <... close resumed>)              = 0
92292 <... close resumed>)              = 0
92291 munmap(0x7f6170aa7000, 32 <unfinished ...>
92292 munmap(0x7f7aad205000, 32 <unfinished ...>
92291 <... munmap resumed>)             = 0
92292 <... munmap resumed>)             = 0
92291 munmap(0x7f6170aa6000, 32 <unfinished ...>
92292 munmap(0x7f7aad204000, 32 <unfinished ...>
92291 <... munmap resumed>)             = 0
92292 <... munmap resumed>)             = 0
92291 munmap(0x7f6170aa5000, 32 <unfinished ...>
92292 munmap(0x7f7aad203000, 32 <unfinished ...>
92291 <... munmap resumed>)             = 0
92292 <... munmap resumed>)             = 0
92291 munmap(0x7f6170aa4000, 32 <unfinished ...>
92292 munmap(0x7f7aad202000, 32 <unfinished ...>
92291 <... munmap resumed>)             = 0
92292 <... munmap resumed>)             = 0
92291 munmap(0x7f6170ae1000, 1032 <unfinished ...>
92292 munmap(0x7f7aad23f000, 1032 <unfinished ...>
92291 <... munmap resumed>)             = 0
92292 <... munmap resumed>)             = 0
92291 close(3 <unfinished ...>
```

```
92292 close(3 <unfinished ...>
92291 <... close resumed>)              = 0
92292 <... close resumed>)              = 0
92292 exit_group(0 <unfinished ...>
92291 exit_group(0 <unfinished ...>
92292 <... exit_group resumed>)         = ?
92291 <... exit_group resumed>)         = ?
92292 +++ exited with 0 +++
92291 +++ exited with 0 +++
92290 <... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) =
    92291
92290 --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=92292, si_uid
    =1000, si_status=0, si_utime=0, si_stime=1} ---
92290 wait4(92292, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 92292
92290 munmap(0x7fea9c107000, 32)        = 0
92290 munmap(0x7fea9c106000, 32)        = 0
92290 munmap(0x7fea9c105000, 32)        = 0
92290 munmap(0x7fea9c104000, 32)        = 0
92290 munmap(0x7fea9c141000, 1032)      = 0
92290 close(3)                          = 0
92290 unlink("/dev/shm/sem.sem_parent_write") = 0
92290 unlink("/dev/shm/sem.sem_child1_read") = 0
92290 unlink("/dev/shm/sem.sem_child2_read") = 0
92290 unlink("/dev/shm/sem.sem_ack")    = 0
92290 unlink("/dev/shm/ipc_buffer")     = 0
92290 exit_group(0)                     = ?
92290 +++ exited with 0 +++
```

Листинг 7: *Strace логи*