Московский авиационный институт
(Национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа № 4 по курсу «Операционные системы»

Вариант 7

Студент: Саженов Константин Станиславович

Группа: 8О-208

Преподаватель: Е. С. Миронов

Вариант: 7

Дата:

Оценка:

Москва, 2020

Лабораторная работа №4

1 Описание

Данная лабораторная работа будет выполняться в ОС Unix.

В данной работе требуется изучить основы принципов работы с файловыми системами, а также приобритение практических навыков в обеспечении обмена данными 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.

Группа вариантов № 2:
Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия файла с таким именем на чтение. Стандартный поток ввода дочернего процесса переопределяется открытым файлом. Дочерний процесс читает команды из стандартного потока ввода. Стандартный поток вывода дочернего процесса перенаправляется в pipe1. Родительский процесс читает из pipe1 и прочитанное выводит в свой стандартный поток вывода. Родительский и дочерний процесс должны быть представлены разными программами.

Вариант № 2:
В файле записаны команды вида: «число число число<endline>». Дочерний процесс считает их сумму и выводит результат в стандартный поток вывода. Числа имеют тип float.

2 Метод решения

Используемые системные вызовы:

void exit(int status);	Функция exit() приводит к обычному завершению программы, и величина status & 0377 (least significant byte of status) возвращается процессу-родителю.
pid_t fork(void);	fork создает процесс-потомок, который отличается от родительского только значениями PID (идентификатор процесса) и PPID (идентификатор родительского процесса), а также тем фактом, что

	счетчики использования ресурсов установлены в 0. Блокировки файлов и сигналы, ожидающие обработки, не наследуются.
int close(int fd);	close закрывает файловый дескриптор, который после этого не ссылается ни на один и файл и может быть использован повторно. Все блокировки, находящиеся на соответствующем файле, снимаются (независимо от того, был ли использован для установки блокировки именно этот файловый дескриптор).
int open(const char *pathname, int flags, mode_t mode);	Вызов open() используется, чтобы преобразовать путь к файлу в описатель файла. Если системный вызов завершается успешно, возвращенный файловый описатель является наименьшим описателем, который еще не открыт процессом. Новый описатель файла будет оставаться открытым при выполнении функции exec(2). Указатель устанавливается в начале файла.
pid_t waitpid(pid_t pid, int *status, int options);	Функция waitpid приостанавливает выполнение текущего процесса до тех пор, пока дочерний процесс, указанный в параметре pid, не завершит выполнение, или пока не появится сигнал, который либо завершает текущий процесс либо требует вызвать функцию-обработчик.
int open(const char *pathname, int flags);	Вызов open() используется, чтобы преобразовать путь к файлу в описатель файла (небольшое неотрицательно целое число, которое используется с вызовами read, write и т.п. при последующем вводе-выводе).
int fstat(int filedes, struct stat *buf);	fstat идентична stat, только возвращается информация об открытом файле, на который указывает filedes (возвращаемый open(2)), а не о file_name.
void * mmap(void *start, size_t length, int prot , int flags, int fd, off_t offset);	Функция mmap отражает length байтов, начиная со смещения offset файла (или другого объекта), определенного файловым описателем fd, в память, начиная с адреса start. Последний параметр (адрес) необязателен, и обычно бывает равен 0.
int munmap(void *start, size_t length);	Функция munmap () удаляет сопоставления для страниц в диапазоне [addr, addr + len), округляя аргумент len до следующего кратного размера страницы, возвращаемого sysconf (3C).
sem_t* sem_open(const char *name, int oflag, mode_t mode, unsigned int value);	Создает новый POSIX именнованный семафор или открывает уже существующий. name – название семафора, mode – такие же, как и у обычного файла, oflag – позволяет задавать права доступа к семафору, value – значение семафора по умолчанию.

Программа запускается без ключей, затем просит пользователя ввести файл, откуда считыать данные. Затем происходит создание файла, который будет использован для создания file mapping-а. Затем для проверки существования открывается файл, откуда будут считываться данные. Затем она создает новый именованный семафор для того, чтобы синхронизировать обмен данными между дочерним и родительским процессом. После этого происходит, собственно, сам процесс mmap в ранее открытый файл. Затем происходит очистка буфера, очистка семафора, если он уже был создан до этого и создание дочернего процесса. Перед execl-ем дочернего процесса происходит очистка всех буферов, mmaping-ов и семафоров. В родительском процессе в обрабатывающем цикле происходит считывание семафора, его значения и, если семафор открыт, происходит считывание данных, иначе ожидание. В дочернем процессе в обрабатывающем цикле также просиходит ожидание освобождения семафора, затем обработка данных и освобождение семафора для считывания данных родителем. После всех считываний в mmap отправляется терминальный символ и родитель ожидает завершения процесса ребенка, после чего закрывает семафоры и отзывает mmap, используя системный вызов munmap.

3 Исходный код

shrmem.h:

```
//
// Created by sakost on 27.11.2020.
//

#ifndef INC_4_LAB_SHRMEM_H
#define INC_4_LAB_SHRMEM_H


#include <fcntl.h>

const size_t map_size = 4096;

const char * BackingFile = "os_lab4.back";
const char * SemaphoreName = "os_lab4.semaphore";
unsigned AccessPerms = S_IWUSR | S_IRUSR | S_IRGRP | S_IROTH;


#endif //INC_4_LAB_SHRMEM_H
```

parent.c:

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

```
#include <sys/wait.h>
#include <sys/mman.h>
#include <fcntl.h>
#include <semaphore.h>
#include <stdbool.h>


#include "shrmem.h"


int main(int argc, char* argv[]) {
    char *filename = NULL;
    size_t len;

    printf("Enter a filename with tests: ");
    if(getline(&filename, &len, stdin) == -1){
        perror("getline");
    }

    filename[strlen(filename) - 1] = '\0';


    int fd = shm_open(BackingFile, O_RDWR | O_CREAT, AccessPerms);
    int file = open(filename, O_RDONLY);
    if (fd == -1 || file == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }

    close(file);

    sem_t *semptr = sem_open(SemaphoreName, O_CREAT, AccessPerms, 2);
    if (semptr == SEM_FAILED){
        perror("sem_open");
        exit(EXIT_FAILURE);
    }
    int val;


    ftruncate(fd, map_size);  // resize file up to mmap size

    caddr_t memptr = mmap(
            NULL,
            map_size,
            PROT_READ | PROT_WRITE,
```

```
        MAP_SHARED,
        fd,
        0
);


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

if(sem_getvalue(semptr, &val) != 0){
    perror("sem_getvalue");
    exit(EXIT_FAILURE);
}

memset(memptr, '\0', map_size); // clean shared memory

while(val++ < 2){ // if semaphore already was created, just fill it up to 2
    sem_post(semptr);
}

pid_t pid = fork();

if(pid == 0){
    munmap(memptr, map_size);
    close(fd);
    sem_close(semptr);
    execl("4_lab_child", "4_lab_child", filename, NULL);
    perror("execl");
    exit(EXIT_FAILURE);
}
else if(pid == -1){
    perror("fork");
    exit(EXIT_FAILURE);
}


while (true){
    // get the value of semaphore
    if(sem_getvalue(semptr, &val) != 0){
        perror("sem_getvalue");
        exit(EXIT_FAILURE);
    }
    // if child process seems not to even capture shared memory and semaphore
```

```
        if(val == 2){
            continue;
        }
        // wait for semaphore
        if(sem_wait(semptr) != 0) {
            perror("sem_wait");
            exit(EXIT_FAILURE);
        }
        // just end of input
        if (memptr[0] == EOF) {
            break;
        }
        // if child process didn't write something to shared memory(we too fast locked
semaphore), just skip iteration
        if(memptr[0] == '\0'){
            if(sem_post(semptr) != 0){
                perror("sem_post");
                exit(EXIT_FAILURE);
            }
            continue;
        }

        char* string = (char*)malloc(strlen(memptr) * sizeof(char));
        // copy data to local storage and clean shared memory
        strcpy(string, memptr);
        memset(memptr, '\0', map_size);
        if(sem_post(semptr) != 0){ // unlock semaphore to let child work
            perror("sem_post");
            exit(EXIT_FAILURE);
        }
        // print the result
        printf("%s\n", string);
        free(string);
    }


    // some error checks...
    int status;
    if(wait(&status) == -1){
        perror("wait");
        exit(EXIT_FAILURE);
    }
    if(!WIFEXITED(status) || (WIFEXITED(status) && (WEXITSTATUS(status)) != 0)){
        fprintf(stderr, "Some error occurred in child process\n");
        return 1;
```

```
    }

    // cleanup
    if(munmap(memptr, map_size) != 0){
        perror("munmap");
        exit(EXIT_FAILURE);
    }

    close(fd);
    if(sem_close(semptr) != 0){
        perror("sem_close");
        exit(EXIT_FAILURE);
    }
    if(shm_unlink(BackingFile) != 0){
        perror("shm_unlink");
        exit(EXIT_FAILURE);
    }

    return 0;
}
```

child.c:

```
//
// Created by sakost on 23.09.2020.
//
#include <stdio.h>
#include <assert.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <string.h>
#include <semaphore.h>
#include <stdbool.h>

#include "shrmem.h"

void send_parent(caddr_t memptr, sem_t* semptr, const char *empty_string, double res){
    while(true) {
        if (sem_wait(semptr) == 0) {
            if(strcmp(memptr, empty_string) != 0) {
                if (sem_post(semptr) != 0) {
                    perror("sem_post");
                    exit(EXIT_FAILURE);
                }
                continue;
```

```
            }
            sprintf(memptr, "%lf\n", res);
            if (sem_post(semptr) != 0) {
                perror("sem_post");
                exit(EXIT_FAILURE);
            }
            break;
        } else {
            perror("sem_wait");
            exit(EXIT_FAILURE);
        }
    }
}



double m_eps (void)
{
    double e = 1.0;
    while (1.0 + e / 2.0 > 1.0) e /= 2.0;
    return e;
}


int main(int argc, char **argv){
    const double eps = m_eps();
    double a;
    char c;
    double res = 0;

    assert(argc == 2);

    char * empty_string = malloc(sizeof(char) * map_size);
    memset(empty_string, '\0', map_size);

    const char *filename = argv[1];

    FILE *file = fopen(filename, "r");
    int map_fd = shm_open(BackingFile, O_RDWR, AccessPerms);
    if(map_fd < 0){
        perror("shm_open");
        exit(EXIT_FAILURE);
    }

    caddr_t memptr = mmap(
```

```
        NULL,
        map_size,
        PROT_READ | PROT_WRITE,
        MAP_SHARED,
        map_fd,
        0
        );
if(memptr == MAP_FAILED){
    perror("mmap");
    exit(EXIT_FAILURE);
}

sem_t *semptr = sem_open(SemaphoreName, O_CREAT, AccessPerms, 2);


if(semptr == SEM_FAILED){
    perror("semptr");
    exit(EXIT_FAILURE);
}

if(sem_wait(semptr) != 0){
    perror("sem_wait");
    exit(EXIT_FAILURE);
}


while(fscanf(file,"%lf%c", &a, &c) != EOF) {
    res += a;
    if(c == '\n') {
        send_parent(memptr, semptr, empty_string, res);
        res = 0.;
        continue;
    }
}

if(!(res < eps && res > -eps))
    send_parent(memptr, semptr, empty_string, res);


while(true) {
    if (sem_wait(semptr) == 0) {
        if(strcmp(memptr, empty_string) != 0) {
            if (sem_post(semptr) != 0) {
                perror("sem_post");
                exit(EXIT_FAILURE);
```

```
            }
            continue;
        }
        memptr[0] = EOF;
        if (sem_post(semptr) != 0) {
            perror("sem_post");
            exit(EXIT_FAILURE);
        }
        break;
    } else {
        perror("sem_wait");
        exit(EXIT_FAILURE);
    }
  }

  if(munmap(memptr, map_size) != 0){
    perror("munmap");
    exit(EXIT_FAILURE);
  }
  sem_close(semptr);

  return EXIT_SUCCESS;
}
```

4 Консоль

```
┌─sakost@sakost-pc ~/university/2 course/os/4 lab/cmake-build-debug ‹master*›
└─$ ./4_lab
Enter a filename with tests: ../tests.txt
10.000000

┌─sakost@sakost-pc ~/university/2 course/os/4 lab/cmake-build-debug ‹master*›
└─$ cat ../tests.txt
1 2 3 4
```

5 Strace

```
execve("./4_lab", ["./4_lab"], 0x7ffe2a21f3e0 /* 55 vars */) = 0
brk(NULL)                               = 0x556702189000
arch_prctl(0x3001 /* ARCH_??? */, 0x7fff0cb07760) = -1 EINVAL (Недопустимый
аргумент)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
```

openat(AT_FDCWD, "tls/haswell/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/tls/haswell/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
stat("/usr/local/lib/tls/haswell/x86_64", 0x7fff0cb06900) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/tls/haswell/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
stat("/usr/local/lib/tls/haswell", 0x7fff0cb06900) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/tls/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
stat("/usr/local/lib/tls/x86_64", 0x7fff0cb06900) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/tls/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
stat("/usr/local/lib/tls", 0x7fff0cb06900) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/haswell/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
stat("/usr/local/lib/haswell/x86_64", 0x7fff0cb06900) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/haswell/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
stat("/usr/local/lib/haswell", 0x7fff0cb06900) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
stat("/usr/local/lib/x86_64", 0x7fff0cb06900) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
stat("/usr/local/lib", {st_mode=S_IFDIR|0755, st_size=12288, ...}) = 0
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

```
fstat(3, {st_mode=S_IFREG|0644, st_size=329160, ...}) = 0
mmap(NULL, 329160, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fd6b5238000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/librt.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P7\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=39408, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7fd6b5236000
mmap(NULL, 43520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7fd6b522b000
mmap(0x7fd6b522e000, 16384, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x3000) = 0x7fd6b522e000
mmap(0x7fd6b5232000, 8192, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x7000) = 0x7fd6b5232000
mmap(0x7fd6b5234000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x8000) = 0x7fd6b5234000
close(3)                                = 0
openat(AT_FDCWD, "tls/haswell/x86_64/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(Нет такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(Нет такого файла или каталога)
openat(AT_FDCWD, "tls/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет
такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(Нет такого файла или каталога)
openat(AT_FDCWD, "x86_64/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(Нет такого файла или каталога)
openat(AT_FDCWD, "libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет
такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/lib/libpthread.so.0", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0000\201\0\0\0\0\0\0"..., 832) = 832
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\307Y\373z\3054\277z\21\35\225\341\273\304<\
223"..., 68, 824) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=158744, ...}) = 0
mmap(NULL, 135600, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7fd6b5209000
mmap(0x7fd6b5210000, 65536, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x7000) = 0x7fd6b5210000
mmap(0x7fd6b5220000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|
```

MAP_DENYWRITE, 3, 0x17000) = 0x7fd6b5220000
mmap(0x7fd6b5225000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1b000) = 0x7fd6b5225000
mmap(0x7fd6b5227000, 12720, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fd6b5227000
close(3) = 0
openat(AT_FDCWD, "tls/haswell/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет
такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет
такого файла или каталога)
openat(AT_FDCWD, "tls/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого
файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет
такого файла или каталога)
openat(AT_FDCWD, "x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет
такого файла или каталога)
openat(AT_FDCWD, "libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого
файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет
такого файла или каталога)
openat(AT_FDCWD, "/usr/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\202\2\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\207\360\21\247\344\314?\306\nT\320\323\335i\
16t"..., 68, 880) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=2159552, ...}) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
mmap(NULL, 1868448, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7fd6b5040000
mmap(0x7fd6b5066000, 1363968, PROT_READ|PROT_EXEC, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x26000) = 0x7fd6b5066000
mmap(0x7fd6b51b3000, 311296, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x173000) = 0x7fd6b51b3000
mmap(0x7fd6b51ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1be000) = 0x7fd6b51ff000
mmap(0x7fd6b5205000, 12960, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fd6b5205000
close(3) = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -
1, 0) = 0x7fd6b503d000

```
arch_prctl(ARCH_SET_FS, 0x7fd6b503d740) = 0
mprotect(0x7fd6b51ff000, 12288, PROT_READ) = 0
mprotect(0x7fd6b5225000, 4096, PROT_READ) = 0
mprotect(0x7fd6b5234000, 4096, PROT_READ) = 0
mprotect(0x55670039a000, 4096, PROT_READ) = 0
mprotect(0x7fd6b52b5000, 4096, PROT_READ) = 0
munmap(0x7fd6b5238000, 329160)          = 0
set_tid_address(0x7fd6b503da10)         = 287359
set_robust_list(0x7fd6b503da20, 24)     = 0
rt_sigaction(SIGRTMIN, {sa_handler=0x7fd6b5210b90, sa_mask=[],
sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7fd6b521d0f0}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x7fd6b5210c30, sa_mask=[],
sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7fd6b521d0f0},
NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) =
0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
brk(NULL)                               = 0x556702189000
brk(0x5567021aa000)                     = 0x5567021aa000
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
write(1, "Enter a filename with tests: ", 29Enter a filename with tests: ) = 29
read(0, ../tests.txt
"../tests.txt\n", 1024)         = 13
statfs("/dev/shm/", {f_type=TMPFS_MAGIC, f_bsize=4096, f_blocks=999734,
f_bfree=956661, f_bavail=956661, f_files=999734, f_ffree=999522, f_fsid={val=[0, 0]},
f_namelen=255, f_frsize=4096, f_flags=ST_VALID|ST_NOSUID|ST_NODEV}) = 0
futex(0x7fd6b522a150, FUTEX_WAKE_PRIVATE, 2147483647) = 0
openat(AT_FDCWD, "/dev/shm/os_lab4.back", O_RDWR|O_CREAT|O_NOFOLLOW|
O_CLOEXEC, 0644) = 3
openat(AT_FDCWD, "../tests.txt", O_RDONLY) = 4
close(4)                                = 0
openat(AT_FDCWD, "/dev/shm/sem.os_lab4.semaphore", O_RDWR|O_NOFOLLOW) = 4
fstat(4, {st_mode=S_IFREG|0644, st_size=32, ...}) = 0
mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7fd6b5288000
close(4)                                = 0
ftruncate(3, 4096)                      = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) =
0x7fd6b5287000
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|
SIGCHLD, child_tidptr=0x7fd6b503da10) = 287385
futex(0x7fd6b5288000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANY) = -1 EAGAIN (Ресурс временно недоступен)
futex(0x7fd6b5288000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANY) = -1 EAGAIN (Ресурс временно недоступен)
```

```
futex(0x7fd6b5288000, FUTEX_WAKE, 1)    = 1
write(1, "10.000000\n", 1010.000000
)           = 10
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=287385, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---
write(1, "\n", 1
)                   = 1
wait4(-1, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 287385
munmap(0x7fd6b5287000, 4096)           = 0
close(3)                               = 0
munmap(0x7fd6b5288000, 32)             = 0
unlink("/dev/shm/os_lab4.back")        = 0
exit_group(0)                          = ?
+++ exited with 0 +++
```

Отслеживание дочерних процессов:
Отслеживать дерево процессов целиком помогает флаг -f, с которым strace отслеживает системные вызовы в процессах-потомках. К каждой строке вывода при этом добавляется pid процесса, делающего системный вывод:

```
execve("./4_lab", ["./4_lab"], 0x7ffc56ded3e8 /* 55 vars */) = 0
brk(NULL)                              = 0x559ca6898000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffecebe8e30) = -1 EINVAL (Недопустимый
аргумент)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет
такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет
такого файла или каталога)
openat(AT_FDCWD, "tls/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого
файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет
такого файла или каталога)
openat(AT_FDCWD, "x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет
такого файла или каталога)
openat(AT_FDCWD, "librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого
файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/tls/haswell/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC)
= -1 ENOENT (Нет такого файла или каталога)
stat("/usr/local/lib/tls/haswell/x86_64", 0x7ffecebe7fd0) = -1 ENOENT (Нет такого файла
или каталога)
```

```
openat(AT_FDCWD, "/usr/local/lib/tls/haswell/librt.so.1", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
stat("/usr/local/lib/tls/haswell", 0x7ffecebe7fd0) = -1 ENOENT (Нет такого файла или
каталога)
openat(AT_FDCWD, "/usr/local/lib/tls/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
stat("/usr/local/lib/tls/x86_64", 0x7ffecebe7fd0) = -1 ENOENT (Нет такого файла или
каталога)
openat(AT_FDCWD, "/usr/local/lib/tls/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(Нет такого файла или каталога)
stat("/usr/local/lib/tls", 0x7ffecebe7fd0) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/haswell/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -
1 ENOENT (Нет такого файла или каталога)
stat("/usr/local/lib/haswell/x86_64", 0x7ffecebe7fd0) = -1 ENOENT (Нет такого файла или
каталога)
openat(AT_FDCWD, "/usr/local/lib/haswell/librt.so.1", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
stat("/usr/local/lib/haswell", 0x7ffecebe7fd0) = -1 ENOENT (Нет такого файла или
каталога)
openat(AT_FDCWD, "/usr/local/lib/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
stat("/usr/local/lib/x86_64", 0x7ffecebe7fd0) = -1 ENOENT (Нет такого файла или
каталога)
openat(AT_FDCWD, "/usr/local/lib/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(Нет такого файла или каталога)
stat("/usr/local/lib", {st_mode=S_IFDIR|0755, st_size=12288, ...}) = 0
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=329160, ...}) = 0
mmap(NULL, 329160, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f1642059000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/librt.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P7\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=39408, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7f1642057000
mmap(NULL, 43520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f164204c000
mmap(0x7f164204f000, 16384, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x3000) = 0x7f164204f000
mmap(0x7f1642053000, 8192, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x7000) = 0x7f1642053000
mmap(0x7f1642055000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x8000) = 0x7f1642055000
close(3)                                = 0
openat(AT_FDCWD, "tls/haswell/x86_64/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1
```

ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "x86_64/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/lib/libpthread.so.0", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0000\201\0\0\0\0\0\0"..., 832) = 832
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\307Y\373z\3054\277z\21\35\225\341\273\304<\223"..., 68, 824) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=158744, ...}) = 0
mmap(NULL, 135600, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f164202a000
mmap(0x7f1642031000, 65536, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x7f1642031000
mmap(0x7f1642041000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17000) = 0x7f1642041000
mmap(0x7f1642046000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b000) = 0x7f1642046000
mmap(0x7f1642048000, 12720, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f1642048000
close(3) = 0
openat(AT_FDCWD, "tls/haswell/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

```
openat(AT_FDCWD, "x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет
такого файла или каталога)
openat(AT_FDCWD, "libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого
файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет
такого файла или каталога)
openat(AT_FDCWD, "/usr/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\202\2\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\207\360\21\247\344\314?\306\nT\320\323\335i\
16t"..., 68, 880) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=2159552, ...}) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0"..., 784, 64) = 784
mmap(NULL, 1868448, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f1641e61000
mmap(0x7f1641e87000, 1363968, PROT_READ|PROT_EXEC, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x26000) = 0x7f1641e87000
mmap(0x7f1641fd4000, 311296, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x173000) = 0x7f1641fd4000
mmap(0x7f1642020000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x1be000) = 0x7f1642020000
mmap(0x7f1642026000, 12960, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f1642026000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -
1, 0) = 0x7f1641e5e000
arch_prctl(ARCH_SET_FS, 0x7f1641e5e740) = 0
mprotect(0x7f1642020000, 12288, PROT_READ) = 0
mprotect(0x7f1642046000, 4096, PROT_READ) = 0
mprotect(0x7f1642055000, 4096, PROT_READ) = 0
mprotect(0x559ca65c1000, 4096, PROT_READ) = 0
mprotect(0x7f16420d6000, 4096, PROT_READ) = 0
munmap(0x7f1642059000, 329160)          = 0
set_tid_address(0x7f1641e5ea10)         = 289392
set_robust_list(0x7f1641e5ea20, 24)     = 0
rt_sigaction(SIGRTMIN, {sa_handler=0x7f1642031b90, sa_mask=[],
sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7f164203e0f0}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x7f1642031c30, sa_mask=[],
sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f164203e0f0},
NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) =
0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
```

```
brk(NULL)                               = 0x559ca6898000
brk(0x559ca68b9000)                     = 0x559ca68b9000
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
write(1, "Enter a filename with tests: ", 29Enter a filename with tests: ) = 29
read(0, ../tests.txt
"../tests.txt\n", 1024)         = 13
statfs("/dev/shm/", {f_type=TMPFS_MAGIC, f_bsize=4096, f_blocks=999734,
f_bfree=956661, f_bavail=956661, f_files=999734, f_ffree=999522, f_fsid={val=[0, 0]},
f_namelen=255, f_frsize=4096, f_flags=ST_VALID|ST_NOSUID|ST_NODEV}) = 0
futex(0x7f164204b150, FUTEX_WAKE_PRIVATE, 2147483647) = 0
openat(AT_FDCWD, "/dev/shm/os_lab4.back", O_RDWR|O_CREAT|O_NOFOLLOW|
O_CLOEXEC, 0644) = 3
openat(AT_FDCWD, "../tests.txt", O_RDONLY) = 4
close(4)                                = 0
openat(AT_FDCWD, "/dev/shm/sem.os_lab4.semaphore", O_RDWR|O_NOFOLLOW) = 4
fstat(4, {st_mode=S_IFREG|0644, st_size=32, ...}) = 0
mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7f16420a9000
close(4)                                = 0
ftruncate(3, 4096)                      = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) =
0x7f16420a8000
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|
SIGCHLDstrace: Process 289427 attached
, child_tidptr=0x7f1641e5ea10) = 289427
[pid 289427] set_robust_list(0x7f1641e5ea20, 24) = 0
[pid 289427] munmap(0x7f16420a8000, 4096) = 0
[pid 289427] close(3)                   = 0
[pid 289427] munmap(0x7f16420a9000, 32) = 0
[pid 289427] execve("4_lab_child", ["4_lab_child", "../tests.txt"], 0x7ffecebe8f18 /* 55 vars */)
= 0
[pid 289427] brk(NULL)                  = 0x55e656fe7000
[pid 289427] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe47894b10) = -1 EINVAL
(Недопустимый аргумент)
[pid 289427] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Нет такого файла или
каталога)
[pid 289427] openat(AT_FDCWD, "tls/haswell/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC)
= -1 ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "tls/haswell/librt.so.1", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "tls/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "tls/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "haswell/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) =
-1 ENOENT (Нет такого файла или каталога)
```

[pid 289427] openat(AT_FDCWD, "haswell/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "/usr/local/lib/tls/haswell/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] stat("/usr/local/lib/tls/haswell/x86_64", 0x7ffe47893cb0) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "/usr/local/lib/tls/haswell/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] stat("/usr/local/lib/tls/haswell", 0x7ffe47893cb0) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "/usr/local/lib/tls/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] stat("/usr/local/lib/tls/x86_64", 0x7ffe47893cb0) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "/usr/local/lib/tls/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] stat("/usr/local/lib/tls", 0x7ffe47893cb0) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "/usr/local/lib/haswell/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] stat("/usr/local/lib/haswell/x86_64", 0x7ffe47893cb0) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "/usr/local/lib/haswell/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] stat("/usr/local/lib/haswell", 0x7ffe47893cb0) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "/usr/local/lib/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] stat("/usr/local/lib/x86_64", 0x7ffe47893cb0) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "/usr/local/lib/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] stat("/usr/local/lib", {st_mode=S_IFDIR|0755, st_size=12288, ...}) = 0
[pid 289427] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
[pid 289427] fstat(3, {st_mode=S_IFREG|0644, st_size=329160, ...}) = 0
[pid 289427] mmap(NULL, 329160, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f8cb300b000
[pid 289427] close(3)   = 0
[pid 289427] openat(AT_FDCWD, "/usr/lib/librt.so.1", O_RDONLY|O_CLOEXEC) = 3
[pid 289427] read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P7\0\0\0\0\0\0"..., 832) = 832
[pid 289427] fstat(3, {st_mode=S_IFREG|0755, st_size=39408, ...}) = 0

[pid 289427] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_ANONYMOUS, -1, 0) = 0x7f8cb3009000
[pid 289427] mmap(NULL, 43520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0)
= 0x7f8cb2ffe000
[pid 289427] mmap(0x7f8cb3001000, 16384, PROT_READ|PROT_EXEC, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7f8cb3001000
[pid 289427] mmap(0x7f8cb3005000, 8192, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x7000) = 0x7f8cb3005000
[pid 289427] mmap(0x7f8cb3007000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x8000) = 0x7f8cb3007000
[pid 289427] close(3)   = 0
[pid 289427] openat(AT_FDCWD, "tls/haswell/x86_64/libpthread.so.0", O_RDONLY|
O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "tls/haswell/libpthread.so.0", O_RDONLY|O_CLOEXEC)
= -1 ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "tls/x86_64/libpthread.so.0", O_RDONLY|O_CLOEXEC) =
-1 ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "tls/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "haswell/x86_64/libpthread.so.0", O_RDONLY|
O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "haswell/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "x86_64/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "/usr/local/lib/libpthread.so.0", O_RDONLY|O_CLOEXEC)
= -1 ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "/usr/lib/libpthread.so.0", O_RDONLY|O_CLOEXEC) = 3
[pid 289427] read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0000\201\0\0\0\0\0\0"...,
832) = 832
[pid 289427] pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\307Y\373z\3054\277z\
21\35\225\341\273\304<\223"..., 68, 824) = 68
[pid 289427] fstat(3, {st_mode=S_IFREG|0755, st_size=158744, ...}) = 0
[pid 289427] mmap(NULL, 135600, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7f8cb2fdc000
[pid 289427] mmap(0x7f8cb2fe3000, 65536, PROT_READ|PROT_EXEC, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x7f8cb2fe3000
[pid 289427] mmap(0x7f8cb2ff3000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x17000) = 0x7f8cb2ff3000
[pid 289427] mmap(0x7f8cb2ff8000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x1b000) = 0x7f8cb2ff8000
[pid 289427] mmap(0x7f8cb2ffa000, 12720, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f8cb2ffa000

```
[pid 289427] close(3)                   = 0
[pid 289427] openat(AT_FDCWD, "tls/haswell/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC)
= -1 ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "tls/haswell/libc.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "tls/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "haswell/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) =
-1 ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "haswell/libc.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "/usr/local/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
[pid 289427] openat(AT_FDCWD, "/usr/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
[pid 289427] read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\202\2\0\0\0\0"...,
832) = 832
[pid 289427] 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
[pid 289427] pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0",
32, 848) = 32
[pid 289427] pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\207\360\21\247\344\314?\306\nT\
320\323\335i\16t"..., 68, 880) = 68
[pid 289427] fstat(3, {st_mode=S_IFREG|0755, st_size=2159552, ...}) = 0
[pid 289427] 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
[pid 289427] mmap(NULL, 1868448, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7f8cb2e13000
[pid 289427] mmap(0x7f8cb2e39000, 1363968, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x26000) = 0x7f8cb2e39000
[pid 289427] mmap(0x7f8cb2f86000, 311296, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x173000) = 0x7f8cb2f86000
[pid 289427] mmap(0x7f8cb2fd2000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x1be000) = 0x7f8cb2fd2000
[pid 289427] mmap(0x7f8cb2fd8000, 12960, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f8cb2fd8000
[pid 289427] close(3)                   = 0
[pid 289427] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_ANONYMOUS, -1, 0) = 0x7f8cb2e10000
[pid 289427] arch_prctl(ARCH_SET_FS, 0x7f8cb2e10740) = 0
```

[pid 289427] mprotect(0x7f8cb2fd2000, 12288, PROT_READ) = 0
[pid 289427] mprotect(0x7f8cb2ff8000, 4096, PROT_READ) = 0
[pid 289427] mprotect(0x7f8cb3007000, 4096, PROT_READ) = 0
[pid 289427] mprotect(0x55e6564df000, 4096, PROT_READ) = 0
[pid 289427] mprotect(0x7f8cb3088000, 4096, PROT_READ) = 0
[pid 289427] munmap(0x7f8cb300b000, 329160) = 0
[pid 289427] set_tid_address(0x7f8cb2e10a10) = 289427
[pid 289427] set_robust_list(0x7f8cb2e10a20, 24) = 0
[pid 289427] rt_sigaction(SIGRTMIN, {sa_handler=0x7f8cb2fe3b90, sa_mask=[], sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7f8cb2ff00f0}, NULL, 8) = 0
[pid 289427] rt_sigaction(SIGRT_1, {sa_handler=0x7f8cb2fe3c30, sa_mask=[], sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f8cb2ff00f0}, NULL, 8) = 0
[pid 289427] rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
[pid 289427] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
[pid 289427] brk(NULL) = 0x55e656fe7000
[pid 289427] brk(0x55e657008000) = 0x55e657008000
[pid 289427] openat(AT_FDCWD, "../tests.txt", O_RDONLY) = 3
[pid 289427] statfs("/dev/shm/", {f_type=TMPFS_MAGIC, f_bsize=4096, f_blocks=999734, f_bfree=956660, f_bavail=956660, f_files=999734, f_ffree=999521, f_fsid={val=[0, 0]}, f_namelen=255, f_frsize=4096, f_flags=ST_VALID|ST_NOSUID|ST_NODEV}) = 0
[pid 289427] futex(0x7f8cb2ffd150, FUTEX_WAKE_PRIVATE, 2147483647) = 0
[pid 289427] openat(AT_FDCWD, "/dev/shm/os_lab4.back", O_RDWR|O_NOFOLLOW|O_CLOEXEC) = 4
[pid 289427] **mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7f8cb305b000**
[pid 289427] openat(AT_FDCWD, "/dev/shm/sem.os_lab4.semaphore", O_RDWR|O_NOFOLLOW) = 5
[pid 289427] fstat(5, {st_mode=S_IFREG|0644, st_size=32, ...}) = 0
[pid 289427] mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 5, 0) = 0x7f8cb305a000
[pid 289427] close(5) = 0
[pid 289427] fstat(3, {st_mode=S_IFREG|0644, st_size=8, ...}) = 0
[pid 289427] read(3, "1 2 3 4\n", 4096) = 8
[pid 289392] futex(0x7f16420a9000, FUTEX_WAKE, 1 <unfinished ...>
[pid 289427] futex(0x7f8cb305a000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 289392] <... futex resumed>) = 0
[pid 289427] <... futex resumed>) = -1 EAGAIN (Ресурс временно недоступен)
[pid 289392] futex(0x7f16420a9000, FUTEX_WAKE, 1) = 0
[pid 289427] read(3, "", 4096) = 0
[pid 289392] write(1, "10.000000\n", 1010.000000
 <unfinished ...>
[pid 289427] **munmap(0x7f8cb305b000, 4096 <unfinished ...>**

```
[pid 289392] <... write resumed>)       = 10
[pid 289427] <... munmap resumed>)      = 0
[pid 289392] write(1, "\n", 1
 <unfinished ...>
[pid 289427] munmap(0x7f8cb305a000, 32 <unfinished ...>
[pid 289392] <... write resumed>)       = 1
[pid 289427] <... munmap resumed>)      = 0
[pid 289392] wait4(-1,  <unfinished ...>
[pid 289427] exit_group(0)              = ?
[pid 289427] +++ exited with 0 +++
<... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 289427
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=289427, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---
munmap(0x7f16420a8000, 4096)            = 0
close(3)                                = 0
munmap(0x7f16420a9000, 32)              = 0
unlink("/dev/shm/os_lab4.back")         = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Красным выделено создание файлов семафоров(сами семафоры создаются одной-двумя строками ниже), а жирным выделены операции mmap/munmap

Статистика системных вызовов:
С помощью опции -c — можно получить наглядную статистику выполнения программы:

```
strace: Process 292327 attached
10.000000
```

% time	seconds	usecs/call	calls	errors	syscall
11,04	0,000089	8	10		mprotect
10,92	0,000088	88	1		clone
9,93	0,000080	0	82	68	openat
8,68	0,000070	7	9		read
8,68	0,000070	14	5	1	futex
7,69	0,000062	31	2		statfs
6,95	0,000056	7	8		munmap
6,45	0,000052	17	3		write
4,47	0,000036	0	38		mmap
3,85	0,000031	2	13		fstat
3,35	0,000027	4	6		brk
2,73	0,000022	5	4		rt_sigaction
2,61	0,000021	21	1		wait4
2,61	0,000021	21	1		unlink

2,23	0,000018	1	13		close
1,49	0,000012	6	2		rt_sigprocmask
1,36	0,000011	2	4	2	arch_prctl
1,36	0,000011	3	3		set_robust_list
1,36	0,000011	5	2		prlimit64
1,24	0,000010	5	2		set_tid_address
0,99	0,000008	8	1		ftruncate
0,00	0,000000	0	16	14	stat
0,00	0,000000	0	10		pread64
0,00	0,000000	0	2	2	access
0,00	0,000000	0	2		execve
------	-----------	-----------	---------	---------	----------
100,00	0,000806	3	240	87	total

6 Выводы

При выполнении данной работы я освежил в памяти то, как следует работать с дочерными процессами, как работать с файловыми дескрипторами, а также приобрел навыки по работе с именованными семафорами.

File mapping является одной из наиболее востребованных и важных функций операционных систем и вот почему:

- Существует возможность не использовать буфер для чтения файла вовсе – можно лишь осуществить file mapping в определенную область в памяти, однако это есть и недостаток из-за того, что приходится узнавать размер файла и выделять область памяти данного размера под файл.
- Сдвиг «указателя» на область памяти не затаратен с точки зрения системных вызовов, в то время как при обычном чтении приходится осуществлять один лишний системный вызов.
- ОС сама осуществляет синхронизацию памяти с файлом и программисту практически не нужно думать о ней.
- Нет необходимости постоянно помнить о расположении файла относительно текущей директории.
- Есть возможность исполнять фрагмент машинного кода, записанного в исполняемый файл, прямо в текущем процессе.

Самым востребованным является последний пункт, т.к. он используется при каждом создании процесса, однако сущестует случай, представленный в данной лабораторной работе – когда требуется создать разделяемую память, доступную нескольким процессам, для обмена данными между ними.

Использование файловых отображений – крайне популярное и безопасное средство для разделения памяти между несколькими процессами, посредством своей простоты и более частого использовании средств языка, а не средств ОС, что является, несомненно, огромным преимуществом.