Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №3 по курсу

«Операционные системы»

Группа: М8О-211Б-23

Студент: Воробьев Г. Я.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 18.11.24

Москва, 2024

Постановка задачи

Цель работы:

Целью является приобретение практических навыков в:

- **Управление процессами в ОС**
- **Обеспечение обмена данных между процессами посредством shared memory**

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 shared memory и memory mapping, для синхронизации чтения и записи из shared memory будем использовать семафор.

10 вариант) В файле записаны команды вида: «число<endline>». Дочерний процесс производит проверку этого числа на простоту. Если число составное, то дочерний процесс пишет это число в стандартный поток вывода. Если число отрицательное или простое, то тогда дочерний и родительский процессы завершаются. Количество чисел может быть произвольным.

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork(void); – создает дочерний процесс.
- int close(int fd); – сообщает операционной системе об окончании работы с файловым дескриптором, и закрывает файл(FD).
- int open(const char * file, int oflag, …); – используется для открытия файла для чтения, записи или и того, и другого.
- ssize_t write(int fd, const void * buf, size_t n); – Записывает N байт из буфер(BUF) в файл (FD). Возвращает количество записанных байт или -1.
- void exit(int status); – выполняет немедленное завершение программы. Все используемые программой потоки закрываются, и временные файлы удаляются, управление возвращается ОС или другой программе.
- int execlp(const char *file, const char *arg, …); – используется для замены текущего процесса новым процессом, выполняющим указанный исполняемый файл.
- pid_t wait(int * stat_loc); – используются для ожидания изменения состояния процесса-потомка вызвавшего процесса и получения информации о потомке, чьё состояние изменилось.
- int shm_open(const char *name, int oflag, mode_t mode); – создает и открывает новый (или открывает уже существующий) объект разделяемой памяти POSIX.
- int shm_unlink(const char *name); – удаляется имя объекта разделяемой памяти и, как только все процессы завершили работу с объектом и отменили его распределение, очищают пространство и уничтожают связанную с ним область памяти.
- int ftruncate(int fd, off_t length); – устанавливают длину файла с файловым дескриптором fd в length байт.
- void * mmap(void *start, size_t length, int prot , int flags, int fd, off_t offset); – отражает length байтов, начиная со смещения offset файла (или другого объекта), определенного файловым дескриптором fd, в память, начиная с адреса start.

- int munmap(void *start, size_t length); – удаляет все отражения из заданной области памяти, после чего все ссылки на данную область будут вызывать ошибку "неправильное обращение к памяти".
- sem_t *sem_open(const char *name, int oflag); ИЛИ sem_t *sem_open(const char *name, int oflag, mode_t mode, unsigned int value); – создаёт новый семафор или открывает уже существующий.
- int sem_wait(sem_t *sem); – уменьшает значение семафора на 1. Если семафор в данный момент имеет нулевое значение, то вызов блокируется до тех пор, пока либо не станет возможным выполнить уменьшение.
- int sem_post(sem_t *sem); – увеличивает значение семафора на 1.
- int sem_unlink(const char *name); – удаляет имя семафора из системы. После вызова этой функции другие процессы больше не смогут открыть этот семафор по имени.
- int sem_close(sem_t *sem); – закрывает указанный семафор, освобождая ресурсы, связанные с ним.

Родительский процесс создает сегмент общей памяти и инициализирует семафоры для синхронизации. Родительский процесс запрашивает у пользователя имя файла для чтения. Родительский процесс создает дочерний процесс с помощью fork(). В дочернем процессе выполняется программа shm_child с использованием execlp(), передавая имя файла и путь к общей памяти в качестве аргументов. Дочерний процесс перенаправляет стандартный ввод на указанный файл и проверяет, является ли файл пустым. Если файл пуст, дочерний процесс сигнализирует родительскому процессу о завершении и завершает выполнение. Дочерний процесс читает числа из файла. Если число простое, дочерний процесс сигнализирует родительскому процессу о завершении и завершает выполнение. Если число составное, дочерний процесс записывает его в общую память и сигнализирует родительскому процессу о готовности данных. Родительский процесс ожидает сигнал от дочернего процесса, затем читает данные из общей памяти и выводит их на стандартный вывод. После чтения данных родительский процесс сигнализирует дочернему процессу о продолжении. Процесс повторяется до тех пор, пока дочерний процесс не сигнализирует о завершении. Родительский процесс ожидает завершения дочернего процесса и очищает ресурсы общей памяти и семафоров.

Код программы

pshm_ucase.h

```
#ifndef PSHM_UCASE_H

#define PSHM_UCASE_H



#include <semaphore.h>

#include <stddef.h>

#include <stdio.h>

#include <stdlib.h>



#define errExit(msg)    do { perror(msg); exit(EXIT_FAILURE); \

                        } while (0)
```

```c
#define BUF_SIZE 1024   /* Максимальный размер передаваемой строки */

/* Определение структуры, которая будет наложена на объект общей памяти */

struct shmbuf {
   sem_t  *sem1;            /* POSIX именованный семафор */
   sem_t  *sem2;            /* POSIX именованный семафор */
   size_t cnt;             /* Количество байт, используемых в 'buf' */
   char   buf[BUF_SIZE];   /* Передаваемые данные */
};

// Объявления функций
void create_shared_memory(const char *shmpath, struct shmbuf **shmp);
void cleanup_shared_memory(const char *shmpath, struct shmbuf *shmp);
void wait_for_child(struct shmbuf *shmp);
void signal_parent(struct shmbuf *shmp);

#endif  // include guard
```

main.c

```c
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/mman.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include "../inc/pshm_ucase.h"
```

```c
int main() {
    char shmpath[256];
    snprintf(shmpath, sizeof(shmpath), "/my_shared_memory");

    struct shmbuf *shmp;
    create_shared_memory(shmpath, &shmp);

    char filename[256];
    printf("Введите имя файла: ");
    if (fgets(filename, sizeof(filename), stdin) == NULL) {
        perror("fgets");
        exit(EXIT_FAILURE);
    }
    size_t len = strlen(filename);
    if (len > 0 && filename[len - 1] == '\n') {
        filename[len - 1] = '\0';
    }

    pid_t child = fork();
    if (child == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    }

    if (child == 0) {
        // Дочерний процесс
        printf("Дочерний процесс: выполнение дочернего процесса\n");
        execlp("./shm_child", "./shm_child", filename, shmpath, NULL);
        perror("execlp");
        exit(EXIT_FAILURE);
```

```
    } else {
        // Родительский процесс
        // Сигнализируем дочернему процессу о готовности
        if (sem_post(shmp->sem1) == -1) {
            perror("sem_post");
            exit(EXIT_FAILURE);
        }

        while (1) {
            printf("Родительский процесс: ожидание семафора 2\n");
            if (sem_wait(shmp->sem2) == -1) {
                perror("sem_wait");
                exit(EXIT_FAILURE);
            }

            if (shmp->cnt == 0) {
                break; // Выход из цикла, если дочерний процесс сигнализирует о
завершении
            }

            printf("Родительский процесс: чтение из общей памяти\n");
            write(STDOUT_FILENO, shmp->buf, shmp->cnt);
            write(STDOUT_FILENO, "\n", 1);

            // Сигнализируем дочернему процессу продолжить
            printf("Родительский процесс: сигнализируем дочернему процессу
продолжить\n");
            if (sem_post(shmp->sem1) == -1) {
                perror("sem_post");
                exit(EXIT_FAILURE);
            }
```

```
        }

        wait(NULL); // Ожидание завершения дочернего процесса
        cleanup_shared_memory(shmpath, shmp);
    }

    return 0;
}
```

core.c

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <unistd.h>
#include "pshm_ucase.h"

void create_shared_memory(const char *shmpath, struct shmbuf **shmp) {
    // Удаление объекта общей памяти, если он уже существует
    shm_unlink(shmpath);

    int fd = shm_open(shmpath, O_CREAT | O_EXCL | O_RDWR, 0600);
    if (fd == -1) {
        errExit("shm_open");
    }

    if (ftruncate(fd, sizeof(struct shmbuf)) == -1) {
        errExit("ftruncate");
    }

    *shmp = mmap(NULL, sizeof(struct shmbuf), PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
```

```
    if (*shmp == MAP_FAILED) {
        errExit("mmap");
    }

    // Инициализация семафоров
    (*shmp)->sem1 = sem_open("/sem1", O_CREAT, 0644, 0);
    if ((*shmp)->sem1 == SEM_FAILED) {
        errExit("sem_open-sem1");
    }
    (*shmp)->sem2 = sem_open("/sem2", O_CREAT, 0644, 0);
    if ((*shmp)->sem2 == SEM_FAILED) {
        sem_unlink("/sem1");
        errExit("sem_open-sem2");
    }

    close(fd); // Закрытие файлового дескриптора после mmap
}

void cleanup_shared_memory(const char *shmpath, struct shmbuf *shmp) {
    if (shmp != NULL) {
        if (shmp->sem1 != NULL) {
            sem_close(shmp->sem1);
            sem_unlink("/sem1");
        }
        if (shmp->sem2 != NULL) {
            sem_close(shmp->sem2);
            sem_unlink("/sem2");
        }
        munmap(shmp, sizeof(struct shmbuf));
    }
    shm_unlink(shmpath);
```

```
}


void wait_for_child(struct shmbuf *shmp) {
   if (sem_wait(shmp->sem1) == -1) {
       errExit("sem_wait-sem1");
   }
}


void signal_parent(struct shmbuf *shmp) {
   if (sem_post(shmp->sem2) == -1) {
       errExit("sem_post-sem2");
   }
}
```

child.c

```
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <unistd.h>
#include "../inc/pshm_ucase.h"


int is_prime(int num) {
   if (num <= 1)
       return 0;
   if (num <= 3)
       return 1;
   if (num % 2 == 0 || num % 3 == 0)
       return 0;
   for (int i = 5; i * i <= num; i += 6)
```

```
        if (num % i == 0 || num % (i + 2) == 0)
            return 0;
    return 1;
}

int main(int argc, char *argv[]) {
    if (argc != 3) {
        fprintf(stderr, "Использование: %s <filename> <shmpath>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    const char *filename = argv[1];
    const char *shmpath = argv[2];

    struct shmbuf *shmp;

    // Подключение к сегменту общей памяти
    int fd = shm_open(shmpath, O_RDWR, 0);
    if (fd == -1) {
        perror("shm_open");
        exit(EXIT_FAILURE);
    }

    shmp = mmap(NULL, sizeof(*shmp), PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
    if (shmp == MAP_FAILED) {
        perror("mmap");
        exit(EXIT_FAILURE);
    }

    // Перенаправление стандартного ввода на файл
    FILE *file = freopen(filename, "r", stdin);
```

```
if (file == NULL) {
    perror("freopen");
    exit(EXIT_FAILURE);
}

// Проверка, является ли файл пустым
fseek(file, 0, SEEK_END);
if (ftell(file) == 0) {
    printf("Дочерний процесс: файл пустой, завершение\n");
    shmp->cnt = 0; // Сигнализируем о завершении
    if (sem_post(shmp->sem2) == -1) {
        perror("sem_post");
        exit(EXIT_FAILURE);
    }
    exit(EXIT_SUCCESS);
}
fseek(file, 0, SEEK_SET); // Возвращаемся в начало файла

// Ожидание сигнала от родительского процесса
printf("Дочерний процесс: ожидание семафора 1\n");
if (sem_wait(shmp->sem1) == -1) {
    perror("sem_wait");
    exit(EXIT_FAILURE);
}

printf("Дочерний процесс: чтение из стандартного ввода\n");
char line[256];
while (fgets(line, sizeof(line), stdin) != NULL) {
    int num = atoi(line);

    printf("Дочерний процесс: прочитано число %d\n", num);
```

```
        if (num < 0 || is_prime(num)) {
            printf("Дочерний процесс: %d отрицательное или простое, завершение\n", num);
            shmp->cnt = 0; // Сигнализируем о завершении
            if (sem_post(shmp->sem2) == -1) {
                perror("sem_post");
                exit(EXIT_FAILURE);
            }
            exit(EXIT_SUCCESS);
        } else {
            printf("Дочерний процесс: %d составное, запись в общую память\n", num);
            shmp->cnt = snprintf(shmp->buf, sizeof(shmp->buf), "%d\n", num);
            if (sem_post(shmp->sem2) == -1) {
                perror("sem_post");
                exit(EXIT_FAILURE);
            }

            // Ожидание, пока родительский процесс прочитает данные
            printf("Дочерний процесс: ожидание семафора 1 для продолжения\n");
            if (sem_wait(shmp->sem1) == -1) {
                perror("sem_wait");
                exit(EXIT_FAILURE);
            }
        }
    }

    return 0;
}
```

Протокол работы программы

Тестирование:

```
glebvorobev@Glebs-MacBook-Air LAB_3 % ./shm_parent
Введите имя файла: test.txt
Родительский процесс: ожидание семафора 2
Дочерний процесс: выполнение дочернего процесса
Дочерний процесс: ожидание семафора 1
Дочерний процесс: чтение из стандартного ввода
Дочерний процесс: прочитано число 4
Дочерний процесс: 4 составное, запись в общую память
Дочерний процесс: ожидание семафора 1 для продолжения
Родительский процесс: чтение из общей памяти
4

Родительский процесс: сигнализируем дочернему процессу продолжить
Родительский процесс: ожидание семафора 2
Дочерний процесс: прочитано число 32
Дочерний процесс: 32 составное, запись в общую память
Дочерний процесс: ожидание семафора 1 для продолжения
Родительский процесс: чтение из общей памяти
32

Родительский процесс: сигнализируем дочернему процессу продолжить
Родительский процесс: ожидание семафора 2
Дочерний процесс: прочитано число 44
Дочерний процесс: 44 составное, запись в общую память
Дочерний процесс: ожидание семафора 1 для продолжения
Родительский процесс: чтение из общей памяти
44

Родительский процесс: сигнализируем дочернему процессу продолжить
Родительский процесс: ожидание семафора 2
Дочерний процесс: прочитано число 4
Дочерний процесс: 4 составное, запись в общую память
Дочерний процесс: ожидание семафора 1 для продолжения
```

Родительский процесс: чтение из общей памяти

4

Родительский процесс: сигнализируем дочернему процессу продолжить

Родительский процесс: ожидание семафора 2

Дочерний процесс: прочитано число -2

Дочерний процесс: -2 простое, завершение

Dtruss:

```
glebvorobev@Glebs-MacBook-Air LAB_3 % sudo dtruss ./shm_parent

SYSCALL(args)             = return

shm_parent(1207,0x1f12f3240) malloc: nano zone abandoned due to inability to reserve vm
space.

Введите имя файла: munmap(0x102858000, 0x84000)          = 0 0

munmap(0x1028DC000, 0x8000)              = 0 0

munmap(0x1028E4000, 0x4000)              = 0 0

munmap(0x1028E8000, 0x4000)              = 0 0

munmap(0x1028EC000, 0x48000)             = 0 0

munmap(0x102934000, 0x4C000)             = 0 0

crossarch_trap(0x0, 0x0, 0x0)            = -1 Err#45

open(".\0", 0x100000, 0x0)               = 3 0

fcntl(0x3, 0x32, 0x16D67B118)            = 0 0

close(0x3)               = 0 0

fsgetpath(0x16D67B128, 0x400, 0x16D67B108)               = 64 0

fsgetpath(0x16D67B138, 0x400, 0x16D67B118)               = 14 0

csrctl(0x0, 0x16D67B53C, 0x4)            = -1 Err#1

__mac_syscall(0x18C8ABC12, 0x2, 0x16D67B480)             = 0 0

csrctl(0x0, 0x16D67B52C, 0x4)            = -1 Err#1

__mac_syscall(0x18C8A8A45, 0x5A, 0x16D67B4C0)            = 0 0

sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16D67AA28, 0x16D67AA20, 0x18C8AA738, 0xD)
= 0 0

sysctl([CTL_KERN, 150, 0, 0, 0, 0] (2), 0x16D67AAD8, 0x16D67AAD0, 0x0, 0x0)
= 0 0
```

```
     open("/\0", 0x20100000, 0x0)             = 3 0

     openat(0x3, "System/Cryptexes/OS\0", 0x100000, 0x0)              = 4 0

     dup(0x4, 0x0, 0x0)               = 5 0

     fstatat64(0x4, 0x16D67A5B1, 0x16D67A520)                 = 0 0

     openat(0x4, "System/Library/dyld/\0", 0x100000, 0x0)             = 6 0

     fcntl(0x6, 0x32, 0x16D67A5B0)            = 0 0

     dup(0x6, 0x0, 0x0)               = 7 0

     dup(0x5, 0x0, 0x0)               = 8 0

     close(0x3)               = 0 0

     close(0x5)               = 0 0

     close(0x4)               = 0 0

     close(0x6)               = 0 0

     __mac_syscall(0x18C8ABC12, 0x2, 0x16D67AFA0)             = 0 0

     shared_region_check_np(0x16D67ABC0, 0x0, 0x0)            = 0 0

     fsgetpath(0x16D67B140, 0x400, 0x16D67B068)               = 82 0

     fcntl(0x8, 0x32, 0x16D67B140)            = 0 0

     close(0x8)               = 0 0

     close(0x7)               = 0 0

     getfsstat64(0x0, 0x0, 0x2)               = 11 0

     getfsstat64(0x102784050, 0x5D28, 0x2)            = 11 0

     getattrlist("/\0", 0x16D67B080, 0x16D67AFF0)             = 0 0

     stat64("/System/Volumes/Preboot/Cryptexes/OS/System/Library/dyld/dyld_shared_cache_arm64
e\0", 0x16D67B3E0, 0x0)          = 0 0

     dtrace: error on enabled probe ID 1696 (ID 845: syscall::stat64:return): invalid address
(0x0) in action #11 at DIF offset 12

     stat64("/Users/glebvorobev/Documents/Projects/OS/build/LAB_3/shm_parent\0", 0x16D67A890,
0x0)            = 0 0

     open("/Users/glebvorobev/Documents/Projects/OS/build/LAB_3/shm_parent\0", 0x0, 0x0)
= 3 0

     mmap(0x0, 0xDFB8, 0x1, 0x40002, 0x3, 0x0)                = 0x102784000 0

     fcntl(0x3, 0x32, 0x16D67A9A8)            = 0 0

     close(0x3)               = 0 0
```

```
     munmap(0x102784000, 0xDFB8)              = 0 0

     stat64("/Users/glebvorobev/Documents/Projects/OS/build/LAB_3/shm_parent\0", 0x16D67AE00,
0x0)            = 0 0

     stat64("/usr/lib/libSystem.B.dylib\0", 0x16D679D50, 0x0)                 = -1 Err#2

     stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/libSystem.B.dylib\0", 0x16D679D00,
0x0)             = -1 Err#2

     open("@rpath/libclang_rt.asan_osx_dynamic.dylib\0", 0x0, 0x0)            = -1 Err#2

     open("@rpath\0", 0x100000, 0x0)          = -1 Err#2

     stat64("/Users/glebvorobev/Documents/Projects/OS/build/LAB_3/libclang_rt.asan_osx_dynami
c.dylib\0", 0x16D679C20, 0x0)            = -1 Err#2

     stat64("/Library/Developer/CommandLineTools/usr/lib/clang/16/lib/darwin/libclang_rt.asan
_osx_dynamic.dylib\0", 0x16D679C20, 0x0)                 = 0 0

     stat64("/Library/Developer/CommandLineTools/usr/lib/clang/16/lib/darwin/libclang_rt.asan
_osx_dynamic.dylib\0", 0x16D679650, 0x0)                 = 0 0

     open("/Library/Developer/CommandLineTools/usr/lib/clang/16/lib/darwin/libclang_rt.asan_o
sx_dynamic.dylib\0", 0x0, 0x0)           = 3 0

     mmap(0x0, 0x40B750, 0x1, 0x40002, 0x3, 0x0)              = 0x102788000 0

     fcntl(0x3, 0x32, 0x16D679768)            = 0 0

     close(0x3)               = 0 0

     open("/Library/Developer/CommandLineTools/usr/lib/clang/16/lib/darwin/libclang_rt.asan_o
sx_dynamic.dylib\0", 0x0, 0x0)           = 3 0

     fstat64(0x3, 0x16D678DF0, 0x0)           = 0 0

     fcntl(0x3, 0x61, 0x16D6793E8)            = 0 0

     fcntl(0x3, 0x62, 0x16D6793E8)            = 0 0

     mmap(0x102B94000, 0xA8000, 0x5, 0x40012, 0x3, 0x32C000)          = 0x102B94000 0

     mmap(0x102C3C000, 0x4000, 0x3, 0x40012, 0x3, 0x3D4000)           = 0x102C3C000 0

     mmap(0x102C40000, 0x4000, 0x3, 0x40012, 0x3, 0x3D8000)           = 0x102C40000 0

     mmap(0x103628000, 0x30000, 0x1, 0x40012, 0x3, 0x3DC000)          = 0x103628000 0

     close(0x3)               = 0 0

     munmap(0x102788000, 0x40B750)            = 0 0

     open("/Users/glebvorobev/Documents/Projects/OS/build/LAB_3/shm_parent\0", 0x0, 0x0)
= 3 0

     __mac_syscall(0x18C8ABC12, 0x2, 0x16D678440)             = 0 0

     map_with_linking_np(0x16D678220, 0x1, 0x16D678250)               = 0 0
```

```
     close(0x3)               = 0 0

     mprotect(0x102778000, 0x4000, 0x1)               = 0 0

     open("/Library/Developer/CommandLineTools/usr/lib/clang/16/lib/darwin/libclang_rt.asan_o
sx_dynamic.dylib\0", 0x0, 0x0)           = 3 0

     __mac_syscall(0x18C8ABC12, 0x2, 0x16D678440)             = 0 0

     map_with_linking_np(0x16D677620, 0x1, 0x16D677650)               = 0 0

     close(0x3)               = 0 0

     mprotect(0x102C3C000, 0x4000, 0x1)               = 0 0

     open("/dev/dtracehelper\0", 0x2, 0x0)            = 3 0

     ioctl(0x3, 0x80086804, 0x16D677988)              = 0 0

     close(0x3)               = 0 0

     shared_region_check_np(0xFFFFFFFFFFFFFFFF, 0x0, 0x0)             = 0 0

     access("/AppleInternal/XBS/.isChrooted\0", 0x0, 0x0)             = -1 Err#2

     bsdthread_register(0x18CBAE0F4, 0x18CBAE0E8, 0x4000)             = 1073746399 0

     getpid(0x0, 0x0, 0x0)            = 1207 0

     shm_open(0x18CA45F41, 0x0, 0xFFFFFFFF9A60A000)           = 3 0

     fstat64(0x3, 0x16D678050, 0x0)           = 0 0

     mmap(0x0, 0x8000, 0x1, 0x1, 0x3, 0x0)            = 0x10278C000 0

     close(0x3)               = 0 0

     csops(0x4B7, 0x0, 0x16D67818C)           = 0 0

     sysctl([CTL_HW, 7, 0, 0, 0, 0] (2), 0x1F12FACF0, 0x16D677FD8, 0x0, 0x0)          = 0 0

     mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102794000 0

     stat64("/\0", 0x16D676310, 0x0)          = 0 0

     getattrlist("/Users\0", 0x18CAC2D20, 0x16D677C20)                = 0 0

     getattrlist("/Users/glebvorobev\0", 0x18CAC2D20, 0x16D677C20)            = 0 0

     getattrlist("/Users/glebvorobev/Documents\0", 0x18CAC2D20, 0x16D677C20)          = 0 0

     getattrlist("/Users/glebvorobev/Documents/Projects\0", 0x18CAC2D20, 0x16D677C20)
= 0 0

     getattrlist("/Users/glebvorobev/Documents/Projects/OS\0", 0x18CAC2D20, 0x16D677C20)
= 0 0

     getattrlist("/Users/glebvorobev/Documents/Projects/OS/build\0", 0x18CAC2D20,
0x16D677C20)               = 0 0
```

```
     getattrlist("/Users/glebvorobev/Documents/Projects/OS/build/LAB_3\0", 0x18CAC2D20,
0x16D677C20)          = 0 0

     getattrlist("/Users/glebvorobev/Documents/Projects/OS/build/LAB_3/shm_parent\0",
0x18CAC2D20, 0x16D677C20)               = 0 0

     munmap(0x102794000, 0x4000)              = 0 0

     mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102794000 0

     getattrlist("/Users\0", 0x18CAC2D20, 0x16D677C20)                = 0 0

     getattrlist("/Users/glebvorobev\0", 0x18CAC2D20, 0x16D677C20)            = 0 0

     getattrlist("/Users/glebvorobev/Documents\0", 0x18CAC2D20, 0x16D677C20)          = 0 0

     getattrlist("/Users/glebvorobev/Documents/Projects\0", 0x18CAC2D20, 0x16D677C20)
= 0 0

     getattrlist("/Users/glebvorobev/Documents/Projects/OS\0", 0x18CAC2D20, 0x16D677C20)
= 0 0

     getattrlist("/Users/glebvorobev/Documents/Projects/OS/build\0", 0x18CAC2D20,
0x16D677C20)               = 0 0

     getattrlist("/Users/glebvorobev/Documents/Projects/OS/build/LAB_3\0", 0x18CAC2D20,
0x16D677C20)          = 0 0

     getattrlist("/Users/glebvorobev/Documents/Projects/OS/build/LAB_3/shm_parent\0",
0x18CAC2D20, 0x16D677C20)               = 0 0

     munmap(0x102794000, 0x4000)              = 0 0

     mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102794000 0

     mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102798000 0

     sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16D678018, 0x16D678010, 0x102C3434A, 0xE)
= 0 0

     sysctl([CTL_KERN, 2, 0, 0, 0, 0] (2), 0x16D6780B0, 0x16D6780A8, 0x0, 0x0)
= 0 0

     mmap(0x0, 0x390000, 0x3, 0x1002, 0x63000000, 0x0)                = 0x10279C000 0

     mprotect(0x102B2C000, 0x4000, 0x1)               = 0 0

     getrlimit(0x1004, 0x16D678100, 0x0)              = 0 0

     setrlimit(0x1004, 0x16D678100, 0x0)              = 0 0

     sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16D677DF8, 0x16D677DF0, 0x102C34752, 0x15)
= 0 0

     sysctl([CTL_KERN, 145, 0, 0, 0, 0] (2), 0x16D677E90, 0x16D677ED8, 0x0, 0x0)
= 0 0
```

```
     mmap(0x700001C000, 0xE00008000, 0x3, 0x1052, 0x63000000, 0x0)            = 0x700001C000
0

     mmap(0x27E00024000, 0xDF1FFFFC000, 0x3, 0x1052, 0x63000000, 0x0)                 =
0x27E00024000 0

     mmap(0x7E00024000, 0x20000000000, 0x0, 0x1052, 0x63000000, 0x0)          = 0x7E00024000
0

     sigaltstack(0x0, 0x16D6780D0, 0x0)              = 0 0

     mmap(0x0, 0x80000, 0x3, 0x1002, 0x63000000, 0x0)                = 0x103658000 0

     sigaltstack(0x16D6780E8, 0x0, 0x0)              = 0 0

     sigaction(0xB, 0x16D6780A8, 0x0)                = 0 0

     sigaction(0xA, 0x16D6780A8, 0x0)                = 0 0

     sigaction(0x8, 0x16D6780A8, 0x0)                = 0 0

     mmap(0x600000000000, 0x40000004000, 0x0, 0x1052, 0x63000000, 0x0)                =
0x600000000000 0

     mmap(0x640000000000, 0x4000, 0x3, 0x1012, 0x63000000, 0x0)               =
0x640000000000 0

     mmap(0x0, 0x800000, 0x0, 0x1042, 0x63000000, 0x0)               = 0x1036D8000 0

     mmap(0x0, 0x10000, 0x3, 0x1002, 0x63000000, 0x0)                = 0x102B30000 0

     mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102B40000 0

     mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102B44000 0

     getrlimit(0x1003, 0x16D677F90, 0x0)             = 0 0

     mmap(0x702D9F4000, 0xFC000, 0x3, 0x1052, 0x63000000, 0x0)                = 0x702D9F4000
0

     sigaltstack(0x0, 0x16D6780A0, 0x0)              = 0 0

     mmap(0x0, 0x200000, 0x3, 0x1002, 0x63000000, 0x0)               = 0x103ED8000 0

     munmap(0x103ED8000, 0x28000)            = 0 0

     munmap(0x104000000, 0xD8000)            = 0 0

     mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102B48000 0

     __mac_syscall(0x198B13505, 0x2, 0x16D677E80)            = 0 0

     mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102B4C000 0

     __mac_syscall(0x198B13505, 0x2, 0x16D677E80)            = 0 0

     __mac_syscall(0x198B13505, 0x2, 0x16D677E80)            = 0 0

     __mac_syscall(0x198B13505, 0x2, 0x16D677E80)            = 0 0
```

```
    __mac_syscall(0x198B13505, 0x2, 0x16D677E80)            = 0 0

    __mac_syscall(0x198B13505, 0x2, 0x16D677E80)            = 0 0

    stat64("/usr/bin/atos\0", 0x16D677F80, 0x0)             = 0 0

    mmap(0x0, 0x200000, 0x3, 0x1002, 0x63000000, 0x0)               = 0x104000000 0

    munmap(0x104100000, 0x100000)           = 0 0

    mmap(0x0, 0x200000, 0x3, 0x1002, 0x63000000, 0x0)               = 0x104100000 0

    munmap(0x104200000, 0x100000)           = 0 0

    munmap(0x102B4C000, 0x4000)             = 0 0

    mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)         = 0x102B4C000 0

    mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)         = 0x102B50000 0

    ioctl(0x2, 0x4004667A, 0x16D6780FC)             = 0 0

    mprotect(0x102B5C000, 0x4000, 0x0)              = 0 0

    mprotect(0x102B68000, 0x4000, 0x0)              = 0 0

    mprotect(0x102B6C000, 0x4000, 0x0)              = 0 0

    mprotect(0x102B78000, 0x4000, 0x0)              = 0 0

    mprotect(0x102B7C000, 0x4000, 0x0)              = 0 0

    mprotect(0x102B88000, 0x4000, 0x0)              = 0 0

    mprotect(0x102B54000, 0xC8, 0x1)                = 0 0

    mprotect(0x102B54000, 0xC8, 0x3)                = 0 0

    mprotect(0x102B54000, 0xC8, 0x1)                = 0 0

    mprotect(0x102B2C000, 0x4000, 0x3)              = 0 0

    mprotect(0x102B2C000, 0x4000, 0x1)              = 0 0

    mprotect(0x102B8C000, 0xC8, 0x1)                = 0 0

    write(0x2, "shm_parent(1207,0x1f12f3240) malloc: nano zone abandoned due to inability to
reserve vm space.\n\0", 0x5F)           = 95 0

    proc_info(0x2, 0x4B7, 0x11)             = 56 0

    socket(0x1, 0x2, 0x0)           = 3 0

    fcntl(0x3, 0x2, 0x1)            = 0 0

    connect(0x3, 0x16D6770B6, 0x6A)         = 0 0

    sendto(0x3, 0x16D6771A0, 0xE3)          = 227 0

    issetugid(0x0, 0x0, 0x0)                = 0 0
```

```
     mmap(0x61B000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x61B000000000 0

     mmap(0x61BE00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x61BE00000000 0

     mmap(0x0, 0x100000, 0x3, 0x1002, 0x63000000, 0x0)               = 0x104200000 0

     mmap(0x0, 0x800000, 0x3, 0x1042, 0x63000000, 0x0)               = 0x104300000 0

     mmap(0x606000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x606000000000 0

     mmap(0x606E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x606E00000000 0

     getentropy(0x16D676EF8, 0x20, 0x0)              = 0 0

     mmap(0x603000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x603000000000 0

     mmap(0x603E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x603E00000000 0

     mmap(0x602000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x602000000000 0

     mmap(0x602E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x602E00000000 0

     mmap(0x624000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x624000000000 0

     mmap(0x624E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x624E00000000 0

     mmap(0x604000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x604000000000 0

     mmap(0x604E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x604E00000000 0

     mmap(0x616000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x616000000000 0

     mmap(0x616E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x616E00000000 0

     mmap(0x608000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x608000000000 0

     mmap(0x608E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x608E00000000 0

     mmap(0x614000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x614000000000 0
```

```
     mmap(0x614E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x614E00000000 0

     mmap(0x612000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x612000000000 0

     mmap(0x612E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x612E00000000 0

     getattrlist("/Users/glebvorobev/Documents/Projects/OS/build/LAB_3/shm_parent\0",
0x16D677FF0, 0x16D67800C)               = 0 0

     access("/Users/glebvorobev/Documents/Projects/OS/build/LAB_3\0", 0x4, 0x0)
= 0 0

     open("/Users/glebvorobev/Documents/Projects/OS/build/LAB_3\0", 0x0, 0x0)
= 4 0

     fstat64(0x4, 0x612000000090, 0x0)               = 0 0

     csrctl(0x0, 0x16D6781DC, 0x4)            = 0 0

     fcntl(0x4, 0x32, 0x16D677ED8)            = 0 0

     close(0x4)               = 0 0

     mmap(0x60C000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x60C000000000 0

     mmap(0x60CE00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x60CE00000000 0

     open("/Users/glebvorobev/Documents/Projects/OS/build/LAB_3/Info.plist\0", 0x0, 0x0)
= -1 Err#2

     mmap(0x60A000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x60A000000000 0

     mmap(0x60AE00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x60AE00000000 0

     mmap(0x607000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x607000000000 0

     mmap(0x607E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x607E00000000 0

     proc_info(0x2, 0x4B7, 0xD)               = 64 0

     csops_audittoken(0x4B7, 0x10, 0x16D678260)               = 0 0

     sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16D677D48, 0x16D677D40, 0x1902D8D3A, 0x15)
= 0 0

     sysctl([CTL_KERN, 148, 0, 0, 0, 0] (2), 0x16D678648, 0x16D678640, 0x0, 0x0)
= 0 0
```

```
mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102B8C000 0

mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102B90000 0

shm_unlink(0x16D68B2E0, 0x0, 0x0)                = -1 Err#2

shm_open(0x16D68B2E0, 0xA02, 0x180)              = 4 0

ftruncate(0x4, 0x418, 0x0)               = 0 0

mmap(0x0, 0x418, 0x3, 0x40001, 0x4, 0x0)                 = 0x103ED8000 0

sem_open(0x102777B00, 0x200, 0x1A4)              = 5 0

sem_open(0x102777B40, 0x200, 0x1A4)              = 6 0

close(0x4)               = 0 0

fstat64(0x1, 0x16D68B040, 0x0)           = 0 0

ioctl(0x1, 0x4004667A, 0x16D68B08C)              = 0 0

mmap(0x621000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x621000000000 0

mmap(0x621E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x621E00000000 0

fstat64(0x0, 0x16D68A8E0, 0x0)           = 0 0

ioctl(0x0, 0x4004667A, 0x16D68A92C)              = 0 0

write_nocancel(0x1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\270\320\274\321\217 \321\204\320\260\320\271\320\273\320\260: \0", 0x22)              =
34 0

test.txt

Родительский процесс: ожидание семафора 2

Дочерний процесс: выполнение дочернего процесса

shm_child(1292,0x1f12f3240) malloc: nano zone abandoned due to inability to reserve vm
space.

Дочерний процесс: ожидание семафора 1

Дочерний процесс: чтение из стандартного ввода

Дочерний процесс: прочитано число 4

Дочерний процесс: 4 составное, запись в общую память

Дочерний процесс: ожидание семафора 1 для продолжения

Родительский процесс: чтение из общей памяти

4
```

```
Родительский процесс: сигнализируем дочернему процессу продолжить

Родительский процесс: ожидание семафора 2

Дочерний процесс: прочитано число 32

Дочерний процесс: 32 составное, запись в общую память

Дочерний процесс: ожидание семафора 1 для продолжения

Родительский процесс: чтение из общей памяти

32


Родительский процесс: сигнализируем дочернему процессу продолжить

Родительский процесс: ожидание семафора 2

Дочерний процесс: прочитано число 44

Дочерний процесс: 44 составное, запись в общую память

Дочерний процесс: ожидание семафора 1 для продолжения

Родительский процесс: чтение из общей памяти

44


Родительский процесс: сигнализируем дочернему процессу продолжить

Родительский процесс: ожидание семафора 2

Дочерний процесс: прочитано число 4

Дочерний процесс: 4 составное, запись в общую память

Дочерний процесс: ожидание семафора 1 для продолжения

Родительский процесс: чтение из общей памяти

4


Родительский процесс: сигнализируем дочернему процессу продолжить

Родительский процесс: ожидание семафора 2

Дочерний процесс: прочитано число -2

Дочерний процесс: -2 простое, завершение

read_nocancel(0x0, "test.txt\n\0", 0x1000)              = 9 0

fork()           = 1292 0

sem_post(0x5, 0x0, 0x0)          = 0 0
```

```
     write_nocancel(0x1,
"\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320
\271 \320\277\321\200\320\276\321\206\320\265\321\201\321\201:
\320\276\320\266\320\270\320\264\320\260\320\275\320\270\320\265
\321\201\320\265\320\274\320\260\321\204\320\276\321\200\320\260 2\n\0", 0x4D)
= 77 0

     sem_wait(0x6, 0x0, 0x0)          = 0 0

     write_nocancel(0x1,
"\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320
\271 \320\277\321\200\320\276\321\206\320\265\321\201\321\201:
\321\207\321\202\320\265\320\275\320\270\320\265 \320\270\320\267
\320\276\320\261\321\211\320\265\320\271
\320\277\320\260\320\274\321\217\321\202\320\270\n\0", 0x53)                 = 83 0

     write(0x1, "4\n\0", 0x2)                 = 2 0

     write(0x1, "\n\0", 0x1)          = 1 0

     write_nocancel(0x1,
"\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320
\271 \320\277\321\200\320\276\321\206\320\265\321\201\321\201:
\321\201\320\270\320\263\320\275\320\260\320\273\320\270\320\267\320\270\321\200\321\203\320\
265\320\274 \320\264\320\276\321\207\320\265\321\200\320\275\320\265\320\274\321\203
\320\277\321\200\320\276\321\206\320\265\321\201\321\201\321\203
\320\277\321\200\320\276\320\264\320\276\320\273\320\266\320\270\321\202\321\214\n\0", 0x7D)
= 125 0

     sem_post(0x5, 0x0, 0x0)          = 0 0

     write_nocancel(0x1,
"\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320
\271 \320\277\321\200\320\276\321\206\320\265\321\201\321\201:
\320\276\320\266\320\270\320\264\320\260\320\275\320\270\320\265
\321\201\320\265\320\274\320\260\321\204\320\276\321\200\320\260 2\n\0", 0x4D)
= 77 0

     sem_wait(0x6, 0x0, 0x0)          = 0 0

     write_nocancel(0x1,
"\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320
\271 \320\277\321\200\320\276\321\206\320\265\321\201\321\201:
\321\207\321\202\320\265\320\275\320\270\320\265 \320\270\320\267
\320\276\320\261\321\211\320\265\320\271
\320\277\320\260\320\274\321\217\321\202\320\270\n\0", 0x53)                 = 83 0

     write(0x1, "32\n\0", 0x3)                = 3 0

     write(0x1, "\n\0", 0x1)          = 1 0

     write_nocancel(0x1,
"\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320
\271 \320\277\321\200\320\276\321\206\320\265\321\201\321\201:
```

```
\321\201\320\270\320\263\320\275\320\260\320\273\320\270\320\267\320\270\321\200\321\203\320\
265\320\274 \320\264\320\276\321\207\320\265\321\200\320\275\320\265\320\274\321\203
\320\277\321\200\320\276\321\206\320\265\321\201\321\201\321\203
\320\277\321\200\320\276\320\264\320\276\320\273\320\266\320\270\321\202\321\214\n\0", 0x7D)
= 125 0

     sem_post(0x5, 0x0, 0x0)          = 0 0

     write_nocancel(0x1,
"\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320
\271 \320\277\321\200\320\276\321\206\320\265\321\201\321\201:
\320\276\320\266\320\270\320\264\320\260\320\275\320\270\320\265
\321\201\320\265\320\274\320\260\321\204\320\276\321\200\320\260 2\n\0", 0x4D)
= 77 0

     sem_wait(0x6, 0x0, 0x0)          = 0 0

     write_nocancel(0x1,
"\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320
\271 \320\277\321\200\320\276\321\206\320\265\321\201\321\201:
\321\207\321\202\320\265\320\275\320\270\320\265 \320\270\320\267
\320\276\320\261\321\211\320\265\320\271
\320\277\320\260\320\274\321\217\321\202\320\270\n\0", 0x53)                 = 83 0

     write(0x1, "44\n\0", 0x3)               = 3 0

     write(0x1, "\n\0", 0x1)          = 1 0

     write_nocancel(0x1,
"\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320
\271 \320\277\321\200\320\276\321\206\320\265\321\201\321\201:
\321\201\320\270\320\263\320\275\320\260\320\273\320\270\320\267\320\270\321\200\321\203\320\
265\320\274 \320\264\320\276\321\207\320\265\321\200\320\275\320\265\320\274\321\203
\320\277\321\200\320\276\321\206\320\265\321\201\321\201\321\203
\320\277\321\200\320\276\320\264\320\276\320\273\320\266\320\270\321\202\321\214\n\0", 0x7D)
= 125 0

     sem_post(0x5, 0x0, 0x0)          = 0 0

     write_nocancel(0x1,
"\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320
\271 \320\277\321\200\320\276\321\206\320\265\321\201\321\201:
\320\276\320\266\320\270\320\264\320\260\320\275\320\270\320\265
\321\201\320\265\320\274\320\260\321\204\320\276\321\200\320\260 2\n\0", 0x4D)
= 77 0

     sem_wait(0x6, 0x0, 0x0)          = 0 0

     write_nocancel(0x1,
"\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320
\271 \320\277\321\200\320\276\321\206\320\265\321\201\321\201:
\321\207\321\202\320\265\320\275\320\270\320\265 \320\270\320\267
\320\276\320\261\321\211\320\265\320\271
\320\277\320\260\320\274\321\217\321\202\320\270\n\0", 0x53)                 = 83 0
```

```
     write(0x1, "4\n\0", 0x2)                 = 2 0

     write(0x1, "\n\0", 0x1)          = 1 0

     write_nocancel(0x1,
"\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320
\271 \320\277\321\200\320\276\321\206\320\265\321\201\321\201:
\321\201\320\270\320\263\320\275\320\260\320\273\320\270\320\267\320\270\321\200\321\203\320\
265\320\274 \320\264\320\276\321\207\320\265\321\200\320\275\320\265\320\274\321\203
\320\277\321\200\320\276\321\206\320\265\321\201\321\201\321\203
\320\277\321\200\320\276\320\264\320\276\320\273\320\266\320\270\321\202\321\214\n\0", 0x7D)
= 125 0

     sem_post(0x5, 0x0, 0x0)          = 0 0

     write_nocancel(0x1,
"\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320
\271 \320\277\321\200\320\276\321\206\320\265\321\201\321\201:
\320\276\320\266\320\270\320\264\320\260\320\275\320\270\320\265
\321\201\320\265\320\274\320\260\321\204\320\276\321\200\320\260 2\n\0", 0x4D)
= 77 0

     sem_wait(0x6, 0x0, 0x0)          = 0 0

     wait4(0xFFFFFFFFFFFFFFFF, 0x0, 0x0)              = 1292 0

     sem_close(0x5, 0x0, 0x0)                 = 0 0

     sem_unlink(0x102777B00, 0x0, 0x0)                = 0 0

     sem_close(0x6, 0x0, 0x0)                 = 0 0

     sem_unlink(0x102777B40, 0x0, 0x0)                = 0 0

     munmap(0x103ED8000, 0x418)               = 0 0

     shm_unlink(0x16D68B2E0, 0x0, 0x0)                = 0 0
```

Вывод

В процессе выполнения данной лабораторной работы я изучил новые системные вызовы на языке Си, которые позволяют эффективно работать с разделяемой памятью и семафорами. Освоил передачу данных между процессами через shared memory и управление доступом с использованием семафоров. Затруднений в ходе выполнения лабораторной работы не возникло, все задачи удалось успешно реализовать.