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-213Б-23

Студент: Гига М.Я.

Преподаватель: Бахарев В.Д

Оценка: _____

Дата: 12.11.24

Москва, 2024

Постановка задачи

Вариант 6.

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. Дочерний процесс выводит в общую память. Родительский процесс читает из общей памяти и прочитанное выводит в стандартный поток вывода. Родительский и дочерний процесс должны быть представлены разными программами. В файле записаны команды вида: «число число число<endline>». Дочерний процесс считает их сумму и выводит результат в стандартный поток вывода. Числа имеют тип int. Количество чисел может быть произвольным. Для связи между родительским и дочерним процессом должны быть использованы общая память и семафоры.

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork(void); – создает дочерний процесс.
- int read(int fd, void* buf, size_t count); – считывает count байт из fd в buf.
- int write(int fd, void* buf, size_t count); – записывает count байт из buf в fd.
- int open(const char *pathname, int flags, ...); – открывает файловый дескриптор.
- int close(int fd); – закрывает файловый дескриптор.
- pid_t waitpid(pid_t pid, int *_Nullable wstatus, int options); – ожидает завершения процесса.
- int shm_open(const char *name, int oflag, mode_t mode); – создает или открывает объект общей памяти.
- int shm_unlink(const char *name); – удаляет объект общей памяти.
- void *mmap(void addr, size_t length, int prot, int flags, int fd, off_t offset); – выполняет отображение файла или устройства на память.
- int munmap(void addr, size_t length); – удаляет отображение файла или устройства на память.
- sem_t *sem_open(const char *name, int oflag, mode_t mode, unsigned int value); – создает или открывает семафор. Если создается новый (oflag & O_CREAT > 0), то необходимо указать разрешения на доступ и начальное значение.
- int sem_wait(sem_t *sem); – ожидает пока значение, хранимое в семафоре, не станет больше нуля, после чего уменьшает его на 1.
- int sem_post(sem_t *sem); – увеличивает значение, хранимое в семафоре на 1.
- int sem_unlink(const char *name); – удаляет семафор.

Программа main инициализирует переменную prog_name значением из argw[1] если оно есть, или строкой "./child.out". Выполняется считывание имени входного файла из stdin. Открывается файловый дескриптор, соответствующий имени файла, выполняется проверка на успешность открытия. Память, выделенная на имя, освобождается. Создается именная общая память, её размер ограничивается, производится проверка успешности операций. Создаются именные семафоры empty = 1 (должен выполнятся дочерний поток) и full = 0 (должен вызываться поток родитель).Вызывается fork. В процессе родителе выполняет отображение общей памяти на

память buf. Процесс в цикле ждет семафора full, выводит buf в stdout и освобождает семафор empty. Если buf пустой, то цикл завершается. После завершении данной операции, процесс родитель ждет завершения дочернего процесса. В дочернем процессе стандартный вход перенаправлен на открытый файл с использованием dup2. С использованием execv вызывается процесс с путем prog_name.

Программа child создает буфер для ввода с переменным размером. Открывается именная общая память, именные семафоры empty и full. и буфер для вывода с неизменяемым размером. выполняет отображение общей памяти на память buf. В цикле процесс ждет семафора empty. Используется метод read_line для построчного чтения стандартного ввода. Каждая строка разбивается на подстроки, разделенные пробелом, каждая подстрока переводится в int использую atoi. Полученные символы складываются. Результат записывается в буфер вывода методом itoa. Освобождается семафор full.

В обоих программах используется метод read_line. В него передаются файловый дескриптор fd, буфер **buf, и размер буфера *buf_size. Этот метод производит сдвиг памяти в буфере до 0 – конца сейчас записанной в буфер строки. Оставшаяся в буфере информация – это остаток после предыдущих чтений. Длина оставшейся строки измеряется и записывается в count. В цикле do происходит проверка наполнения буфера: если он полный, то производится перевыделение памяти. Происходит чтение из fd в буфер со сдвигом в count и размером buf_size - count. Count увеличивается на количество считанных символов. Цикл завершается, когда в буфере содержится символ переноса строки. После завершения цикла первое вхождение '\n' в буфер заменяется на '\0' – символ конца строки.

Код программы

main.c

```
#include <errno.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <semaphore.h>

#include "misc.h"

int main(int argc, char *argw[]) {
    char *prog_name;
    if (argc >= 2)
        prog_name = argw[1];
    else
        prog_name = "./child.out";

    int buf_size = 64;
    char *fname = malloc(buf_size);
    *fname = 0;
    if (read_line(STDIN_FILENO, &fname, &buf_size) <= 0) {
        print(STDERR_FILENO,
              "ERROR: failed read filename from standart input\n");
        exit(-1);
    }
    int fd = open(fname, O_RDONLY);
    if (fd == -1) {
```

```
    print_error("failed to open file");
    exit(-1);
}
free(fname);

int mem_fd = shm_open(MEM, O_RDONLY | O_CREAT, 0777);
if (mem_fd == -1) {
    print_error("failed to open shared memory");
    close(fd);
    exit(-1);
}
if (ftruncate(mem_fd, MEM_SIZE) == -1) {
    print_error("failed to truncate shared memory");
    close(fd);
    shm_unlink(MEM);
    exit(-1);
}

sem_t *empty = sem_open(SEM_EMPTY, O_CREAT, 0777, 1);
if (empty == SEM_FAILED) {
    print_error("failed to create empty semaphore");
    close(fd);
    shm_unlink(MEM);
    exit(-1);
}

sem_t *full = sem_open(SEM_FULL, O_CREAT, 0777, 0);
if (full == SEM_FAILED) {
    print_error("failed to create full semaphore");
    close(fd);
    shm_unlink(MEM);
    sem_unlink(SEM_EMPTY);

    exit(-1);
}

pid_t pid = fork();
if (pid == 0) {
    dup2(fd, STDIN_FILENO);
    close(fd);
    char *argv[] = {prog_name, "", NULL};
    if (execv(prog_name, argv) == -1) {
        char out[1024] = {0};
        strcat(out, "ERROR: failed to launch process \"");
        strcat(out, prog_name);
        strcat(out, "\"\n");
        strcat(out, strerror(errno));
        strcat(out, "\n");
        print(STDERR_FILENO, out);
        exit(-1);
    }
} else if (pid == -1) {
    print_error("failed to fork process");
    exit(-1);
} else {
    char *buf = mmap(NULL, MEM_SIZE, PROT_READ, MAP_SHARED, mem_fd, 0);
    if (!buf) {
        print_error("failed to mmap");
        sem_unlink(SEM_EMPTY);
        sem_unlink(SEM_FULL);
        shm_unlink(MEM);
        munmap(buf, MEM_SIZE);
        exit(-1);
    }
```

```
        while (1) {
            sem_wait(full);
            if (*buf == 0)
                break;
            print(STDOUT_FILENO, buf);
            sem_post(empty);
        }
        waitpid(pid, 0, 0);
        munmap(buf, MEM_SIZE);
    }
    sem_unlink(SEM_EMPTY);
    sem_unlink(SEM_FULL);
    shm_unlink(MEM);

    return 0;
}
```

child.c

```
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/mman.h>
#include <semaphore.h>

#include "misc.h"

int main(void) {
    int buf_size = 2;
    char *input_buffer = malloc(buf_size);
    memset(input_buffer, 0, buf_size);

    if (!input_buffer) {
        return -1;
    }

    int mem_fd = shm_open(MEM, O_WRONLY, 0);
    if (mem_fd == -1) {
        print_error("failed to open shared memory");
        exit(-1);
    }

    sem_t *empty = sem_open(SEM_EMPTY, 0);
    if (empty == SEM_FAILED) {
        print(STDERR_FILENO, "ERROR: failed to open semaphore\n");
        exit(-1);
    }

    sem_t *full = sem_open(SEM_FULL, 0);
    if (empty == SEM_FAILED) {
        print(STDERR_FILENO, "ERROR: failed to open semaphore\n");
        exit(-1);
    }

    char *buf =
        mmap(NULL, MEM_SIZE, PROT_WRITE, MAP_SHARED, mem_fd, 0);
    if (!buf) {
        print_error("failed to mmap");
        exit(-1);
    }

    while (1) {
        sem_wait(empty);
        int count = read_line(STDIN_FILENO, &input_buffer, &buf_size);
```

```
        if (count <= 0) {
            buf[0] = 0;
            sem_post(full);
            break;
        }
        int res = 0;
        char *ptr = input_buffer;
        while (*ptr) {
            int f = atoi(ptr);
            res += f;
            ptr = strchr(ptr, ' ');
            if (!ptr)
                break;
            ptr++;
        }
        int n = itoa(res, buf, 1);
        buf[n++] = '\n';
        buf[n++] = 0;
        sem_post(full);
    }
    munmap(buf, MEM_SIZE);
    free(input_buffer);
    return 0;
}
```

misc.h

```
#ifndef __MISC_H__
#define __MISC_H__

#include <stddef.h>
#define MEM "/lab03_memory"
#define MEM_SIZE 128
#define SEM_EMPTY "/lab03_semaphore_empty"
#define SEM_FULL "/lab03_semaphore_full"


int itoa(long n, char *res, int d);
char *strnchr(const char *buf, char c, size_t len);
int read_line(int fd, char **buf, int *buf_size);
int print(int fd, const char *s);
int print_error(const char *s);

#endif
```

misc.c

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

int itoa(long n, char *res, int d) {
    int neg = 0;
    if (n < 0) {
        neg++;
        n *= -1;
        res[0] = '-';
        res++;
    }
    int i = 0;
    while (n > 0) {
        res[i++] = '0' + (n % 10);
```

```
        n /= 10;
    }
    while (i < d) {
        res[i++] = '0';
    }
    for (int j = 0; j < i / 2; j++) {
        char tmp = res[j];
        res[j] = res[i - j - 1];
        res[i - j - 1] = tmp;
    }
    res[i] = 0;
    return i + neg;
}

char *strnchr(const char *buf, char c, size_t len) {
    for (size_t i = 0; i < len; i++) {
        if (buf[i] == c)
            return (char *)(buf + i);
        if (buf[i] == 0)
            return 0;
    }
    return 0;
}

int read_line(int fd, char **buf, int *buf_size) {
    int count = strnchr(*buf, 0, *buf_size) - *buf + 1;
    for (int i = 0; i < *buf_size - count; i++) {
        (*buf)[i] = (*buf)[i + count];
    }
    count = strnchr(*buf, 0, *buf_size) - *buf;
    int read_cur;
    do {
        if (count + 1 >= *buf_size) {
            int new_size = *buf_size * 2;
            char *tmp = realloc(*buf, new_size);
            if (!tmp)
                return count;
            *buf = tmp;
            *buf_size = new_size;
        }
        read_cur = read(fd, *buf + count, *buf_size - count - 1);
        count += read_cur;
        (*buf)[count] = 0;
    } while (read_cur > 0 && !strnchr(*buf, '\n', *buf_size));
    int line_len = strnchr(*buf, '\n', *buf_size) - *buf;
    if (line_len < 0)
        return count;
    (*buf)[line_len] = 0;
    return line_len + 1;
}

int print(int fd, const char *s) {
    int n = strlen(s);
    return write(fd, s, n);
}

void print_error(const char *s) {
    char out[1024] = {0};
    strcat(out, "ERROR: ");
    strcat(out, s);
    strcat(out, "\n");
    strcat(out, strerror(errno));
    strcat(out, "\n");
    print(STDERR_FILENO, out);
```

```
}
```

Протокол работы программы

Тестирование:

```
$ cat input.txt
1
124 12 -1

2
1
-2317
-23 23 12 -11
2 4

$ ./main.out
a.txt
ERROR: failed to open file: "a.txt"
No such file or directory

$ ./main.out
input.txt
1
135
0
2
1
-2317
1
6
```

Strace:

```
$      PID/THRD  SYSCALL(args)           = return
dtrace: error on enabled probe ID 1696 (ID 845: syscall::stat64:return): invalid
address (0x0) in action #12 at DIF offset 12
 4444/0xa45f:  fork()             = 0 0
 4444/0xa45f:  munmap(0x100D18000, 0x84000)          = 0 0
 4444/0xa45f:  munmap(0x100D9C000, 0x8000)           = 0 0
 4444/0xa45f:  munmap(0x100DA4000, 0x4000)           = 0 0
 4444/0xa45f:  munmap(0x100DA8000, 0x4000)           = 0 0
 4444/0xa45f:  munmap(0x100DAC000, 0x48000)          = 0 0
 4444/0xa45f:  munmap(0x100DF4000, 0x4C000)          = 0 0
 4444/0xa45f:  crossarch_trap(0x0, 0x0, 0x0)         = -1 Err#45
 4444/0xa45f:  open(".\0", 0x100000, 0x0)            = 3 0
 4444/0xa45f:  fcntl(0x3, 0x32, 0x16F4B70A8)         = 0 0
 4444/0xa45f:  close(0x3)         = 0 0
 4444/0xa45f:  fsgetpath(0x16F4B70B8, 0x400, 0x16F4B7098)         = 49 0
 4444/0xa45f:  fsgetpath(0x16F4B70C8, 0x400, 0x16F4B70A8)         = 14 0
 4444/0xa45f:  csrctl(0x0, 0x16F4B74CC, 0x4)         = -1 Err#1
 4444/0xa45f:  __mac_syscall(0x191983D62, 0x2, 0x16F4B7410)              = 0 0
 4444/0xa45f:  csrctl(0x0, 0x16F4B74BC, 0x4)         = -1 Err#1
 4444/0xa45f:  __mac_syscall(0x191980B95, 0x5A, 0x16F4B7450)             = 0 0
 4444/0xa45f:  sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16F4B69B8, 0x16F4B69B0,
0x191982888, 0xD)          = 0 0
 4444/0xa45f:  sysctl([CTL_KERN, 157, 0, 0, 0, 0] (2), 0x16F4B6A68, 0x16F4B6A60, 0x0,
0x0)          = 0 0
 4444/0xa45f:  open("/\0", 0x20100000, 0x0)          = 3 0
 4444/0xa45f:  openat(0x3, "System/Cryptexes/OS\0", 0x100000, 0x0)             = 4 0
 4444/0xa45f:  dup(0x4, 0x0, 0x0)              = 5 0
 4444/0xa45f:  fstatat64(0x4, 0x16F4B6541, 0x16F4B64B0)           = 0 0
```

```
 4444/0xa45f:  openat(0x4, "System/Library/dyld/\0", 0x100000, 0x0)            = 6 0
 4444/0xa45f:  fcntl(0x6, 0x32, 0x16F4B6540)          = 0 0
 4444/0xa45f:  dup(0x6, 0x0, 0x0)              = 7 0
 4444/0xa45f:  dup(0x5, 0x0, 0x0)              = 8 0
 4444/0xa45f:  close(0x3)         = 0 0
 4444/0xa45f:  close(0x5)         = 0 0
 4444/0xa45f:  close(0x4)         = 0 0
 4444/0xa45f:  close(0x6)         = 0 0
 4444/0xa45f:  __mac_syscall(0x191983D62, 0x2, 0x16F4B6F30)              = 0 0
 4444/0xa45f:  shared_region_check_np(0x16F4B6B50, 0x0, 0x0)             = 0 0
 4444/0xa45f:  fsgetpath(0x16F4B70D0, 0x400, 0x16F4B6FF8)          = 82 0
 4444/0xa45f:  fcntl(0x8, 0x32, 0x16F4B70D0)          = 0 0
 4444/0xa45f:  close(0x8)         = 0 0
 4444/0xa45f:  close(0x7)         = 0 0
 4444/0xa45f:  getfsstat64(0x0, 0x0, 0x2)             = 10 0
 4444/0xa45f:  getfsstat64(0x100944050, 0x54B0, 0x2)        = 10 0
 4444/0xa45f:  getattrlist("/\0", 0x16F4B7010, 0x16F4B6F80)              = 0 0
 4444/0xa45f:  open("/Users/maxgiga/dev/mai/MAI_OS/lab03/src/main.out\0", 0x0, 0x0)
= 3 0
 4444/0xa45f:  mmap(0x0, 0x8B18, 0x1, 0x40002, 0x3, 0x0)           = 0x100944000 0
 4444/0xa45f:  fcntl(0x3, 0x32, 0x16F4B6938)          = 0 0
 4444/0xa45f:  close(0x3)         = 0 0
 4444/0xa45f:  munmap(0x100944000, 0x8B18)            = 0 0
 4444/0xa45f:  open("/Users/maxgiga/dev/mai/MAI_OS/lab03/src/main.out\0", 0x0, 0x0)
= 3 0
 4444/0xa45f:  __mac_syscall(0x191983D62, 0x2, 0x16F4B41B0)              = 0 0
 4444/0xa45f:  map_with_linking_np(0x16F4B3FF0, 0x1, 0x16F4B4020)        = 0 0
 4444/0xa45f:  close(0x3)         = 0 0
 4444/0xa45f:  mprotect(0x10093C000, 0x4000, 0x1)           = 0 0
 4444/0xa45f:  open("/dev/dtracehelper\0", 0x2, 0x0)        = 3 0
 4444/0xa45f:  ioctl(0x3, 0x80086804, 0x16F4B3538)          = 0 0
 4444/0xa45f:  close(0x3)         = 0 0
 4444/0xa45f:  shared_region_check_np(0xFFFFFFFFFFFFFFFF, 0x0, 0x0)             = 0 0
 4444/0xa45f:  access("/AppleInternal/XBS/.isChrooted\0", 0x0, 0x0)             = -1
Err#2
 4444/0xa45f:  bsdthread_register(0x191C860F4, 0x191C860E8, 0x4000)             =
1073746399 0
 4444/0xa45f:  getpid(0x0, 0x0, 0x0)           = 4444 0
 4444/0xa45f:  shm_open(0x191B1DF41, 0x0, 0xFFFFFFFF91CC4000)            = 3 0
 4444/0xa45f:  fstat64(0x3, 0x16F4B3BB0, 0x0)         = 0 0
 4444/0xa45f:  mmap(0x0, 0x8000, 0x1, 0x40001, 0x3, 0x0)           = 0x10094C000 0
 4444/0xa45f:  close(0x3)         = 0 0
 4444/0xa45f:  csops(0x115C, 0x0, 0x16F4B3CEC)              = 0 0
 4444/0xa45f:  ioctl(0x2, 0x4004667A, 0x16F4B3C5C)          = -1 Err#25
 4444/0xa45f:  ioctl(0x2, 0x40487413, 0x16F4B3C60)          = -1 Err#25
 4444/0xa45f:  mprotect(0x10095C000, 0x4000, 0x0)           = 0 0
 4444/0xa45f:  mprotect(0x100968000, 0x4000, 0x0)           = 0 0
 4444/0xa45f:  mprotect(0x10096C000, 0x4000, 0x0)           = 0 0
 4444/0xa45f:  mprotect(0x100978000, 0x4000, 0x0)           = 0 0
 4444/0xa45f:  mprotect(0x10097C000, 0x4000, 0x0)           = 0 0
 4444/0xa45f:  mprotect(0x100988000, 0x4000, 0x0)           = 0 0
 4444/0xa45f:  mprotect(0x100954000, 0xC8, 0x1)             = 0 0
 4444/0xa45f:  mprotect(0x100954000, 0xC8, 0x3)             = 0 0
 4444/0xa45f:  mprotect(0x100954000, 0xC8, 0x1)             = 0 0
 4444/0xa45f:  mprotect(0x10098C000, 0x4000, 0x1)           = 0 0
 4444/0xa45f:  mprotect(0x100990000, 0xC8, 0x1)             = 0 0
 4444/0xa45f:  mprotect(0x100990000, 0xC8, 0x3)             = 0 0
 4444/0xa45f:  mprotect(0x100990000, 0xC8, 0x1)             = 0 0
 4444/0xa45f:  mprotect(0x100954000, 0xC8, 0x3)             = 0 0
 4444/0xa45f:  mprotect(0x100954000, 0xC8, 0x1)             = 0 0
 4444/0xa45f:  mprotect(0x10098C000, 0x4000, 0x3)           = 0 0
 4444/0xa45f:  mprotect(0x10098C000, 0x4000, 0x1)           = 0 0
 4444/0xa45f:  issetugid(0x0, 0x0, 0x0)         = 0 0
 4444/0xa45f:  getentropy(0x16F4B32C8, 0x20, 0x0)           = 0 0
```

```
 4444/0xa45f:  getattrlist("/Users/maxgiga/dev/mai/MAI_OS/lab03/src/main.out\0",
0x16F4B3B50, 0x16F4B3B6C)         = 0 0
 4444/0xa45f:  access("/Users/maxgiga/dev/mai/MAI_OS/lab03/src\0", 0x4, 0x0)
= 0 0
 4444/0xa45f:  open("/Users/maxgiga/dev/mai/MAI_OS/lab03/src\0", 0x0, 0x0)
= 3 0
 4444/0xa45f:  fstat64(0x3, 0x132E044E0, 0x0)         = 0 0
 4444/0xa45f:  csrctl(0x0, 0x16F4B3D3C, 0x4)          = -1 Err#1
 4444/0xa45f:  fgetattrlist(0x3, 0x16F4B3DE0, 0x16F4B3D60)         = 0 0
 4444/0xa45f:  __mac_syscall(0x19DC2E505, 0x2, 0x16F4B3D60)               = 0 0
 4444/0xa45f:  fcntl(0x3, 0x32, 0x16F4B3A38)          = 0 0
 4444/0xa45f:  close(0x3)         = 0 0
 4444/0xa45f:  open("/Users/maxgiga/dev/mai/MAI_OS/lab03/src/Info.plist\0", 0x0, 0x0)
= -1 Err#2
 4444/0xa45f:  proc_info(0x2, 0x115C, 0xD)            = 64 0
 4444/0xa45f:  csops_audittoken(0x115C, 0x10, 0x16F4B3DC0)         = 0 0
 4444/0xa45f:  sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16F4B4118, 0x16F4B4110,
0x195397D3A, 0x15)          = 0 0
 4444/0xa45f:  sysctl([CTL_KERN, 155, 0, 0, 0, 0] (2), 0x16F4B41A8, 0x16F4B41A0, 0x0,
0x0)          = 0 0
1
135
0
2
1
-2317
1
6
dtrace: error on enabled probe ID 1694 (ID 287: syscall::execve:return): invalid
address (0x10093be34) in action #12 at DIF offset 12
dtrace: error on enabled probe ID 1696 (ID 845: syscall::stat64:return): invalid
address (0x0) in action #12 at DIF offset 12
 4444/0xa45f:  read(0x0, "input.txt\n\0", 0x3F)              = 8 0
 4444/0xa45f:  open("input.txt\0", 0x0, 0x0)          = 3 0
 4444/0xa45f:  shm_open(0x10093BE85, 0x200, 0x1FF)          = 4 0
 4444/0xa45f:  ftruncate(0x4, 0x80, 0x0)              = 0 0
 4444/0xa45f:  sem_open(0x10093BED1, 0x200, 0x1FF)          = 5 0
 4444/0xa45f:  sem_open(0x10093BF09, 0x200, 0x1FF)          = 6 0
 4444/0xa45f:  fork()             = 4445 0
 4445/0xa483:  fork()             = 0 0
 4445/0xa483:  thread_selfid(0x0, 0x0, 0x0)           = 42115 0
 4445/0xa483:  bsdthread_register(0x191C860F4, 0x191C860E8, 0x4000)             = -1
Err#22
 4444/0xa45f:  mmap(0x0, 0x80, 0x1, 0x40001, 0x4, 0x0)             = 0x100998000 0
 4445/0xa483:  mprotect(0x100990000, 0xC8, 0x3)             = 0 0
 4445/0xa483:  mprotect(0x100990000, 0xC8, 0x1)             = 0 0
 4445/0xa483:  dup2(0x3, 0x0, 0x0)             = 0 0
 4445/0xa483:  close(0x3)         = 0 0
 4445/0xa484:  fork()             = 0 0
 4445/0xa484:  mprotect(0x103144000, 0x8000, 0x1)           = 0 0
 4445/0xa484:  thread_selfid(0x0, 0x0, 0x0)           = 42116 0
 4445/0xa484:  crossarch_trap(0x0, 0x0, 0x0)          = -1 Err#45
 4445/0xa484:  shared_region_check_np(0x16CE9F770, 0x0, 0x0)              = 0 0
 4445/0xa484:  thread_selfid(0x0, 0x0, 0x0)           = 42116 0
 4445/0xa484:  getpid(0x0, 0x0, 0x0)           = 4445 0
 4445/0xa484:  proc_info(0xF, 0x115D, 0x0)            = 0 0
 4445/0xa484:  munmap(0x1030C0000, 0x84000)           = 0 0
 4445/0xa484:  munmap(0x103144000, 0x8000)            = 0 0
 4445/0xa484:  munmap(0x10314C000, 0x4000)            = 0 0
 4445/0xa484:  munmap(0x103150000, 0x4000)            = 0 0
 4445/0xa484:  munmap(0x103154000, 0x48000)           = 0 0
 4445/0xa484:  munmap(0x10319C000, 0x4C000)           = 0 0
 4445/0xa484:  crossarch_trap(0x0, 0x0, 0x0)          = -1 Err#45
 4445/0xa484:  open(".\0", 0x100000, 0x0)             = 3 0
```

```
 4445/0xa484:  fcntl(0x3, 0x32, 0x16CE8F0A8)         = 0 0
 4445/0xa484:  close(0x3)         = 0 0
 4445/0xa484:  fsgetpath(0x16CE8F0B8, 0x400, 0x16CE8F098)         = 50 0
 4445/0xa484:  fsgetpath(0x16CE8F0C8, 0x400, 0x16CE8F0A8)         = 14 0
 4445/0xa484:  csrctl(0x0, 0x16CE8F4CC, 0x4)         = -1 Err#1
 4445/0xa484:  __mac_syscall(0x191983D62, 0x2, 0x16CE8F410)              = 0 0
 4445/0xa484:  csrctl(0x0, 0x16CE8F4BC, 0x4)         = -1 Err#1
 4445/0xa484:  __mac_syscall(0x191980B95, 0x5A, 0x16CE8F450)             = 0 0
 4445/0xa484:  sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16CE8E9B8, 0x16CE8E9B0,
0x191982888, 0xD)          = 0 0
 4445/0xa484:  sysctl([CTL_KERN, 157, 0, 0, 0, 0] (2), 0x16CE8EA68, 0x16CE8EA60, 0x0,
0x0)          = 0 0
 4445/0xa484:  open("/\0", 0x20100000, 0x0)          = 3 0
 4445/0xa484:  openat(0x3, "System/Cryptexes/OS\0", 0x100000, 0x0)              = 4 0
 4445/0xa484:  dup(0x4, 0x0, 0x0)              = 7 0
 4445/0xa484:  fstatat64(0x4, 0x16CE8E541, 0x16CE8E4B0)           = 0 0
 4445/0xa484:  openat(0x4, "System/Library/dyld/\0", 0x100000, 0x0)             = 8 0
 4445/0xa484:  fcntl(0x8, 0x32, 0x16CE8E540)         = 0 0
 4445/0xa484:  dup(0x8, 0x0, 0x0)              = 9 0
 4445/0xa484:  dup(0x7, 0x0, 0x0)              = 10 0
 4445/0xa484:  close(0x3)         = 0 0
 4445/0xa484:  close(0x7)         = 0 0
 4445/0xa484:  close(0x4)         = 0 0
 4445/0xa484:  close(0x8)         = 0 0
 4445/0xa484:  __mac_syscall(0x191983D62, 0x2, 0x16CE8EF30)              = 0 0
 4445/0xa484:  shared_region_check_np(0x16CE8EB50, 0x0, 0x0)             = 0 0
 4445/0xa484:  fsgetpath(0x16CE8F0D0, 0x400, 0x16CE8EFF8)         = 82 0
 4445/0xa484:  fcntl(0xA, 0x32, 0x16CE8F0D0)         = 0 0
 4445/0xa484:  close(0xA)         = 0 0
 4445/0xa484:  close(0x9)         = 0 0
 4445/0xa484:  getfsstat64(0x0, 0x0, 0x2)             = 10 0
 4445/0xa484:  getfsstat64(0x102F6C050, 0x54B0, 0x2)        = 10 0
 4445/0xa484:  getattrlist("/\0", 0x16CE8F010, 0x16CE8EF80)              = 0 0
 4445/0xa484:  open("/Users/maxgiga/dev/mai/MAI_OS/lab03/src/child.out\0", 0x0, 0x0)
= 3 0
 4445/0xa484:  mmap(0x0, 0x8A08, 0x1, 0x40002, 0x3, 0x0)          = 0x102F6C000 0
 4445/0xa484:  fcntl(0x3, 0x32, 0x16CE8E938)         = 0 0
 4445/0xa484:  close(0x3)         = 0 0
 4445/0xa484:  munmap(0x102F6C000, 0x8A08)            = 0 0
 4445/0xa484:  open("/Users/maxgiga/dev/mai/MAI_OS/lab03/src/child.out\0", 0x0, 0x0)
= 3 0
 4445/0xa484:  __mac_syscall(0x191983D62, 0x2, 0x16CE8C1B0)              = 0 0
 4445/0xa484:  map_with_linking_np(0x16CE8C020, 0x1, 0x16CE8C050)        = 0 0
 4445/0xa484:  close(0x3)         = 0 0
 4445/0xa484:  mprotect(0x102F64000, 0x4000, 0x1)           = 0 0
 4445/0xa484:  open("/dev/dtracehelper\0", 0x2, 0x0)        = 3 0
 4445/0xa484:  ioctl(0x3, 0x80086804, 0x16CE8B538)          = 0 0
 4445/0xa484:  close(0x3)         = 0 0
 4445/0xa484:  shared_region_check_np(0xFFFFFFFFFFFFFFFF, 0x0, 0x0)             = 0 0
 4445/0xa484:  access("/AppleInternal/XBS/.isChrooted\0", 0x0, 0x0)             = -1
Err#2
 4445/0xa484:  bsdthread_register(0x191C860F4, 0x191C860E8, 0x4000)             =
1073746399 0
 4445/0xa484:  getpid(0x0, 0x0, 0x0)           = 4445 0
 4445/0xa484:  shm_open(0x191B1DF41, 0x0, 0xFFFFFFFF9F6FC000)            = 3 0
 4445/0xa484:  fstat64(0x3, 0x16CE8BBB0, 0x0)         = 0 0
 4445/0xa484:  mmap(0x0, 0x8000, 0x1, 0x40001, 0x3, 0x0)           = 0x102F74000 0
 4445/0xa484:  close(0x3)         = 0 0
 4445/0xa484:  csops(0x115D, 0x0, 0x16CE8BCEC)              = 0 0
 4445/0xa484:  ioctl(0x2, 0x4004667A, 0x16CE8BC5C)          = -1 Err#25
 4445/0xa484:  ioctl(0x2, 0x40487413, 0x16CE8BC60)          = -1 Err#25
 4445/0xa484:  mprotect(0x102F84000, 0x4000, 0x0)           = 0 0
 4445/0xa484:  mprotect(0x102F90000, 0x4000, 0x0)           = 0 0
 4445/0xa484:  mprotect(0x102F94000, 0x4000, 0x0)           = 0 0
```

```
 4445/0xa484:  mprotect(0x102FA0000, 0x4000, 0x0)           = 0 0
 4445/0xa484:  mprotect(0x102FA4000, 0x4000, 0x0)           = 0 0
 4445/0xa484:  mprotect(0x102FB0000, 0x4000, 0x0)           = 0 0
 4445/0xa484:  mprotect(0x102F7C000, 0xC8, 0x1)             = 0 0
 4445/0xa484:  mprotect(0x102F7C000, 0xC8, 0x3)             = 0 0
 4445/0xa484:  mprotect(0x102F7C000, 0xC8, 0x1)             = 0 0
 4445/0xa484:  mprotect(0x102FB4000, 0x4000, 0x1)           = 0 0
 4445/0xa484:  mprotect(0x102FB8000, 0xC8, 0x1)             = 0 0
 4445/0xa484:  mprotect(0x102FB8000, 0xC8, 0x3)             = 0 0
 4445/0xa484:  mprotect(0x102FB8000, 0xC8, 0x1)             = 0 0
 4445/0xa484:  mprotect(0x102F7C000, 0xC8, 0x3)             = 0 0
 4445/0xa484:  mprotect(0x102F7C000, 0xC8, 0x1)             = 0 0
 4445/0xa484:  mprotect(0x102FB4000, 0x4000, 0x3)           = 0 0
 4445/0xa484:  mprotect(0x102FB4000, 0x4000, 0x1)           = 0 0
 4445/0xa484:  issetugid(0x0, 0x0, 0x0)        = 0 0
 4445/0xa484:  getentropy(0x16CE8B2C8, 0x20, 0x0)           = 0 0
 4445/0xa484:  getattrlist("/Users/maxgiga/dev/mai/MAI_OS/lab03/src/child.out\0",
0x16CE8BB50, 0x16CE8BB6C)         = 0 0
 4445/0xa484:  access("/Users/maxgiga/dev/mai/MAI_OS/lab03/src\0", 0x4, 0x0)
= 0 0
 4445/0xa484:  open("/Users/maxgiga/dev/mai/MAI_OS/lab03/src\0", 0x0, 0x0)
= 3 0
 4445/0xa484:  fstat64(0x3, 0x14B6044E0, 0x0)         = 0 0
 4445/0xa484:  csrctl(0x0, 0x16CE8BD3C, 0x4)          = -1 Err#1
 4445/0xa484:  fgetattrlist(0x3, 0x16CE8BDE0, 0x16CE8BD60)         = 0 0
 4445/0xa484:  __mac_syscall(0x19DC2E505, 0x2, 0x16CE8BD60)               = 0 0
 4445/0xa484:  fcntl(0x3, 0x32, 0x16CE8BA38)          = 0 0
 4445/0xa484:  close(0x3)          = 0 0
 4445/0xa484:  open("/Users/maxgiga/dev/mai/MAI_OS/lab03/src/Info.plist\0", 0x0, 0x0)
= -1 Err#2
 4445/0xa484:  proc_info(0x2, 0x115D, 0xD)            = 64 0
 4445/0xa484:  csops_audittoken(0x115D, 0x10, 0x16CE8BDC0)         = 0 0
 4445/0xa484:  sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16CE8C118, 0x16CE8C110,
0x195397D3A, 0x15)          = 0 0
 4445/0xa484:  sysctl([CTL_KERN, 155, 0, 0, 0, 0] (2), 0x16CE8C1A8, 0x16CE8C1A0, 0x0,
0x0)           = 0 0
 4445/0xa484:  shm_open(0x102F63F04, 0x1, 0x0)              = 3 0
 4445/0xa484:  sem_open(0x102F63F2F, 0x0, 0x0)              = 4 0
 4445/0xa484:  sem_open(0x102F63F67, 0x0, 0x0)              = 7 0
 4445/0xa484:  mmap(0x0, 0x80, 0x2, 0x40001, 0x3, 0x0)             = 0x102FC0000 0
 4445/0xa484:  sem_wait(0x4, 0x0, 0x0)          = 0 0
 4445/0xa484:  read(0x0, "1\0", 0x1)            = 1 0
 4445/0xa484:  read(0x0, "\n1\0", 0x2)          = 2 0
 4445/0xa484:  sem_post(0x7, 0x0, 0x0)          = 0 0
 4444/0xa45f:  sem_wait(0x6, 0x0, 0x0)          = 0 0
 4444/0xa45f:  write(0x1, "1\n\0", 0x2)         = 2 0
 4444/0xa45f:  sem_post(0x5, 0x0, 0x0)          = 0 0
 4445/0xa484:  sem_wait(0x4, 0x0, 0x0)          = 0 0
 4445/0xa484:  read(0x0, "24\0", 0x2)           = 2 0
 4445/0xa484:  read(0x0, " 12 \0", 0x4)         = 4 0
 4445/0xa484:  read(0x0, "-1\n\n2\n1\n\0", 0x8)             = 8 0
 4445/0xa484:  sem_post(0x7, 0x0, 0x0)          = 0 0
 4444/0xa45f:  sem_wait(0x6, 0x0, 0x0)          = 0 0
 4444/0xa45f:  write(0x1, "135\n\0", 0x4)              = 4 0
 4444/0xa45f:  sem_post(0x5, 0x0, 0x0)          = 0 0
 4445/0xa484:  sem_wait(0x4, 0x0, 0x0)          = 0 0
 4445/0xa484:  read(0x0, "-2317\n-23 \0", 0xA)              = 10 0
 4445/0xa484:  sem_post(0x7, 0x0, 0x0)          = 0 0
 4444/0xa45f:  sem_wait(0x6, 0x0, 0x0)          = 0 0
 4444/0xa45f:  write(0x1, "0\n\0", 0x2)         = 2 0
 4444/0xa45f:  sem_post(0x5, 0x0, 0x0)          = 0 0
 4445/0xa484:  sem_wait(0x4, 0x0, 0x0)          = 0 0
 4445/0xa484:  read(0x0, "2\0", 0x1)            = 1 0
 4445/0xa484:  sem_post(0x7, 0x0, 0x0)          = 0 0
```

```
4444/0xa45f:  sem_wait(0x6, 0x0, 0x0)          = 0 0
4444/0xa45f:  write(0x1, "2\n\0", 0x2)         = 2 0
4444/0xa45f:  sem_post(0x5, 0x0, 0x0)          = 0 0
4445/0xa484:  sem_wait(0x4, 0x0, 0x0)          = 0 0
4445/0xa484:  read(0x0, "3 \0", 0x2)           = 2 0
4445/0xa484:  sem_post(0x7, 0x0, 0x0)          = 0 0
4444/0xa45f:  sem_wait(0x6, 0x0, 0x0)          = 0 0
4444/0xa45f:  write(0x1, "1\n\0", 0x2)         = 2 0
4444/0xa45f:  sem_post(0x5, 0x0, 0x0)          = 0 0
4445/0xa484:  sem_wait(0x4, 0x0, 0x0)          = 0 0
4445/0xa484:  read(0x0, "12\0", 0x2)           = 2 0
4445/0xa484:  sem_post(0x7, 0x0, 0x0)          = 0 0
4444/0xa45f:  sem_wait(0x6, 0x0, 0x0)          = 0 0
4444/0xa45f:  write(0x1, "-2317\n\0", 0x6)           = 6 0
4444/0xa45f:  sem_post(0x5, 0x0, 0x0)          = 0 0
4445/0xa484:  sem_wait(0x4, 0x0, 0x0)          = 0 0
4445/0xa484:  read(0x0, " -11\n2\0", 0x6)            = 6 0
4445/0xa484:  sem_post(0x7, 0x0, 0x0)          = 0 0
4444/0xa45f:  sem_wait(0x6, 0x0, 0x0)          = 0 0
4444/0xa45f:  write(0x1, "1\n\0", 0x2)         = 2 0
4444/0xa45f:  sem_post(0x5, 0x0, 0x0)          = 0 0
4445/0xa484:  sem_wait(0x4, 0x0, 0x0)          = 0 0
4445/0xa484:  read(0x0, " 4\n\0", 0xE)         = 3 0
4445/0xa484:  sem_post(0x7, 0x0, 0x0)          = 0 0
4444/0xa45f:  sem_wait(0x6, 0x0, 0x0)          = 0 0
4444/0xa45f:  write(0x1, "6\n\0", 0x2)         = 2 0
4444/0xa45f:  sem_post(0x5, 0x0, 0x0)          = 0 0
4445/0xa484:  sem_wait(0x4, 0x0, 0x0)          = 0 0
4445/0xa484:  read(0x0, "\0", 0xF)             = 0 0
4445/0xa484:  sem_post(0x7, 0x0, 0x0)          = 0 0
4445/0xa484:  munmap(0x102FC0000, 0x80)              = 0 0
4444/0xa45f:  sem_wait(0x6, 0x0, 0x0)          = 0 0
4444/0xa45f:  wait4(0x115D, 0x0, 0x0)          = 4445 0
4444/0xa45f:  munmap(0x100998000, 0x80)              = 0 0
4444/0xa45f:  sem_unlink(0x10093BED1, 0x0, 0x0)            = 0 0
4444/0xa45f:  sem_unlink(0x10093BF09, 0x0, 0x0)            = 0 0
4444/0xa45f:  shm_unlink(0x10093BE85, 0x0, 0x0)            = 0 0
```

Красным обозначены системные вызовы, произведенные родительским процессом, синим – системные вызовы дочернего процесса.

Вывод

В ходе выполнения лабораторной работы была составлена и отлажена программа на языке C, осуществляющая вызов дочернего процесса, переопределения стандартных ввода вывода для него и взаимодействие между дочерним и родительским процессами в операционной системе macOS с использованием общей памяти и семафоров. Была также написана программа, осуществляющая ввод из стандартного ввода, обработку и вывод информации в стандартный вывод без использования stdio.h из стандартной библиотеки языка c.