Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу

«Операционные системы»

Студентка:

Варламова Анна Борисовна

Группа: М80-207Б-20

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 20.12.2021

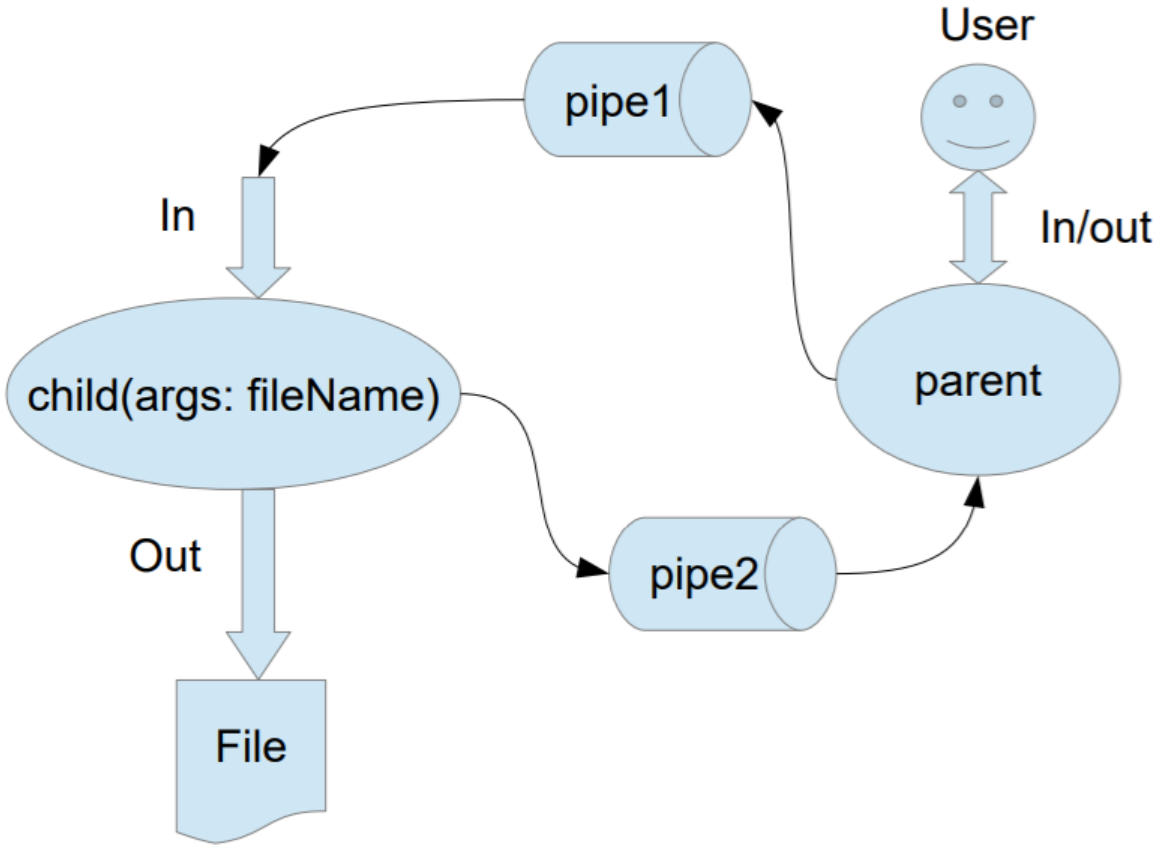
Москва, 2021

Постановка задачи

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 1



Группа вариантов № 1, вариант 4:

Родительский процесс создаёт дочерний. Первой строчкой пользователь вводит имя файла, которое будет передано в дочерний, туда запишутся результаты работы дочернего процесса. Далее родительский процесс считывается команды вида: <число число число endl>, числа типа float. Родительский процесс через pipe1 передаёт команды в дочерний, который в свою очередь делит первое число команды на последующие и записывает результат в открытый

вначале файл. Если встречается деление на 0, через pipe2 дочерний процесс передаёт информацию об этом родительскому процессу, оба процесса завершаются.

Общие сведения о программе

Программа состоит из файлов main.cpp, child.cpp, mem.h. В них используются заголовочные файлы ostream, unistd.h, sstream, signal.h, fcntl.h, sys/mman.h, sys/types.h, sys/stat.h, pthread.h, stdio.h, semaphore.h.

Программа использует следующие системные вызовы:

1. **sem_open –** для создания нового именнованного семафора.
2. **sem_unlink** – для удаления именованного семафора.
3. **sem_destroy** – для уничтожения семафора.
4. **open -** для создания файла и его открытия.
5. **close** – для закрытия файлового дескриптора.
6. **mmap –** для отображения файла в память.
7. **fork** – для создания дочернего процесса.
8. **sem_wait –** для блокировки семафора.
9. **sem_post** – для разблокировки семафора.
10. **dup2** – для перенаправления потока вывода.
11. **getpid** - для получения id процесса.
12. **fstat** - для считывания состояния файла.
13. **ftruncate** — обрезает/расширяет файл до заданного размера.
14. **remove** — для удаления файла.
15. **munmap** — для удаления отображения в файл.

Общий метод и алгоритм решения

Pipes теперь заменяет mmap, семафором регулируем доступ к разделяемой памяти. Также теперь не нужна функция fflush, ведь другой процесс не может войти в свою критическую область, пока первый не закончит свою работу. В остальном решение не отличается от 2 лабораторной работы.

Код программы

mem.h

```
#ifndef SHRMEM_H
#define SHRMEM_H


#include <fcntl.h>


const char *CommonFile = "like_a_pipe";
const char *SemaphoreName = "my_semaphore";
unsigned mode = S_IWUSR | S_IRUSR | S_IRGRP | S_IROTH;


#endif // SHRMEM_H
```

main.cpp

```
#include <fcntl.h>
#include <semaphore.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/mman.h>
#include <unistd.h>
#include <iostream>

#include "mem.h"

using namespace std;

int main() {
            char *filename = NULL;
    size_t len = 0;
    std::cout << "Enter the name of file for answers: ";
    if (getline(&filename, &len, stdin) == -1) {
        perror("getline");
        _exit(EXIT_FAILURE);
    }
    filename[strlen(filename) - 1] = '\0';
```

```
std::cout << "Enter numerous:" << std::endl;

size_t map_size = 0;
char *in = (char *)malloc(sizeof(char));
char c;
while ((c = getchar()) != EOF) {
        in[map_size] = c;
                in = (char *)realloc(in, (++map_size + 1) * sizeof(char));
}
in[map_size++] = '\0';

int fd = shm_open(CommonFile, O_RDWR | O_CREAT, mode);
if (fd == -1) {
                perror("OPEN");
                _exit(EXIT_FAILURE);
}
sem_t *semptr = sem_open(SemaphoreName, O_CREAT, mode, 1);
if (semptr == SEM_FAILED) {
                perror("SEM_OPEN");
                _exit(EXIT_FAILURE);
}
int val;

ftruncate(fd, (off_t)map_size);
char* memptr = (char*)mmap(
        NULL,
        map_size,
        PROT_READ | PROT_WRITE,
        MAP_SHARED,
        fd,
        0);
if (memptr == MAP_FAILED) {
                perror("MMAP");
                _exit(EXIT_FAILURE);
}
sprintf(memptr, "%s", in);
free(in);
if (sem_getvalue(semptr, &val) != 0) {
                perror("SEM_GETVALUE");
                _exit(EXIT_FAILURE);
}
while (val++ < 1) {
   sem_post(semptr);
}
int pid = fork();

if (pid == 0) {
```

```
                munmap(memptr, map_size);
            close(fd);
            sem_close(semptr);
     execl("child", "child", filename, NULL);
            perror("EXECL");
    } else if (pid < 0) {
            perror("FORK");
            exit(EXIT_FAILURE);
    }
    while (true) {
                    if (sem_getvalue(semptr, &val) != 0) {
                       perror("SEM_GETVALUE");
                       _exit(EXIT_FAILURE);
                    }
                    if (val == 0) {
                       if (sem_wait(semptr) == -1) {
                          perror("SEM_WAIT");
                  _exit(EXIT_FAILURE);
                       }
                       cout << memptr;
                       return EXIT_SUCCESS;
                    }
    }
}
```

child.cpp

```
#include <fcntl.h>
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <unistd.h>
#include <string>
#include <iostream>

#include "mem.h"

using namespace std;

int main(int argc, char **argv) {
    int map_fd = shm_open(CommonFile, O_RDWR, mode);
    if (map_fd < 0) {
            perror("SHM_OPEN");
            _exit(EXIT_FAILURE);
    }
```

```
struct stat statbuf;
fstat(map_fd, &statbuf);
const size_t map_size = statbuf.st_size;
char* memptr = (char*)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, mode, 1);
if (semptr == SEM_FAILED) {
            perror("SEM_OPEN");
            _exit(EXIT_FAILURE);
}
if (sem_wait(semptr) != 0) {
            perror("SEM_WAIT");
            _exit(EXIT_FAILURE);
}
char *out = (char *)malloc(sizeof(char));
size_t m_size = 0;
int flag = 0;
string first;
string second;

FILE *filename = fopen(argv[1], "w");
if (filename == NULL) {
                    perror("File not opened");
            _exit(EXIT_FAILURE);
        }

for (int i = 0; i + 1 < map_size; ++i) {// преобразование
            if (flag == 0) {
                        first.push_back(memptr[i]);
            } else if (flag == 1) {
                        second.push_back(memptr[i]);
            }
            if (memptr[i] == ' ' && flag == 0) {
                        flag = 1;
            } else if ((memptr[i] == ' ' || memptr[i] == '\n') && flag == 1) {
                        if (atof(second.c_str()) == 0) {
                                    perror("division by zero error\n");
                                    break;
```

```
                }
                first = to_string(atof(first.c_str()) / atof(second.c_str()));
                second = "";
                if (memptr[i] == '\n') {
                        fprintf(filename, "%s\n", first.c_str());
                    flag = 0;
                    first = "";
                    second = "";
                }
            }
    }
    fclose(filename);

    out[m_size++] = '\0';
    ftruncate(map_fd, (off_t)m_size);
    memset(memptr, '\0', m_size);
    sprintf(memptr, "%s", out);
    free(out);
    close(map_fd);
    sem_post(semptr);
    sem_close(semptr);
    return EXIT_SUCCESS;
}
```

Использование утилиты strace

```
ann@ann:~/os/lab4$ strace ./main
execve("./main", ["./main"], 0x7ffd00177910 /* 63 vars */) = 0
brk(NULL)                               = 0x55b70b0d0000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=79721, ...}) = 0
mmap(NULL, 79721, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f47f948a000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/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\0000b\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=144976, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7f47f9488000
mmap(NULL, 2221184, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7f47f9056000
mprotect(0x7f47f9070000, 2093056, PROT_NONE) = 0
```

```
mmap(0x7f47f926f000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19000) = 0x7f47f926f000
mmap(0x7f47f9271000, 13440, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f47f9271000
close(3)                        = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/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\0\0\"\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=31680, ...}) = 0
mmap(NULL, 2128864, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7f47f8e4e000
mprotect(0x7f47f8e55000, 2093056, PROT_NONE) = 0
mmap(0x7f47f9054000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x6000) = 0x7f47f9054000
close(3)                        = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/libstdc++.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\304\10\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=1594864, ...}) = 0
mmap(NULL, 3702848, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7f47f8ac5000
mprotect(0x7f47f8c3e000, 2097152, PROT_NONE) = 0
mmap(0x7f47f8e3e000, 49152, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x179000) = 0x7f47f8e3e000
mmap(0x7f47f8e4a000, 12352, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f47f8e4a000
close(3)                        = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/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\20\35\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030928, ...}) = 0
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7f47f86d4000
mprotect(0x7f47f88bb000, 2097152, PROT_NONE) = 0
mmap(0x7f47f8abb000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f47f8abb000
mmap(0x7f47f8ac1000, 15072, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f47f8ac1000
close(3)                        = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
```

```
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.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\200\272\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=1700792, ...}) = 0
mmap(NULL, 3789144, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7f47f8336000
mprotect(0x7f47f84d3000, 2093056, PROT_NONE) = 0
mmap(0x7f47f86d2000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19c000) = 0x7f47f86d2000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\300*\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=96616, ...}) = 0
mmap(NULL, 2192432, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7f47f811e000
mprotect(0x7f47f8135000, 2093056, PROT_NONE) = 0
mmap(0x7f47f8334000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x16000) = 0x7f47f8334000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7f47f9486000
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7f47f9483000
arch_prctl(ARCH_SET_FS, 0x7f47f9483740) = 0
mprotect(0x7f47f8abb000, 16384, PROT_READ) = 0
mprotect(0x7f47f8334000, 4096, PROT_READ) = 0
mprotect(0x7f47f86d2000, 4096, PROT_READ) = 0
mprotect(0x7f47f8e3e000, 40960, PROT_READ) = 0
mprotect(0x7f47f926f000, 4096, PROT_READ) = 0
mprotect(0x7f47f9054000, 4096, PROT_READ) = 0
mprotect(0x55b70935b000, 4096, PROT_READ) = 0
mprotect(0x7f47f949e000, 4096, PROT_READ) = 0
munmap(0x7f47f948a000, 79721)           = 0
set_tid_address(0x7f47f9483a10)         = 26737
set_robust_list(0x7f47f9483a20, 24)     = 0
rt_sigaction(SIGRTMIN, {sa_handler=0x7f47f905bcb0, sa_mask=[],
sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7f47f9068980}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x7f47f905bd50, sa_mask=[],
sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f47f9068980}, 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
brk(NULL)                               = 0x55b70b0d0000
brk(0x55b70b0f1000)                     = 0x55b70b0f1000
futex(0x7f47f8e4b09c, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f47f8e4b0a8, FUTEX_WAKE_PRIVATE, 2147483647) = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
write(1, "Enter the name of file for answe"..., 36Enter the name of file for answers: ) = 36
read(0, 1.txt
"1.txt\n", 1024)                = 6
write(1, "Enter numerous:\n", 16Enter numerous:
)       = 16
read(0, 67 5 6
"67 5 6\n", 1024)               = 7
read(0, 34 7 8
"34 7 8\n", 1024)               = 7
read(0, "", 1024)                       = 0
statfs("/dev/shm/", {f_type=TMPFS_MAGIC, f_bsize=4096, f_blocks=494308, f_bfree=411429,
f_bavail=411429, f_files=494308, f_ffree=493960, f_fsid={val=[0, 0]}, f_namelen=255,
f_frsize=4096, f_flags=ST_VALID|ST_NOSUID|ST_NODEV}) = 0
futex(0x7f47f9274370, FUTEX_WAKE_PRIVATE, 2147483647) = 0
openat(AT_FDCWD, "/dev/shm/like_a_pipe",
O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0644) = 3
openat(AT_FDCWD, "/dev/shm/sem.my_semaphore", O_RDWR|O_NOFOLLOW) = -1
ENOENT (No such file or directory)
getpid()                                = 26737
lstat("/dev/shm/RxIaDn", 0x7ffefdff24f0) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/dev/shm/RxIaDn", O_RDWR|O_CREAT|O_EXCL, 0644) = 4
write(4, "\1\0\0\0\0\0\0\200\0\0\0e_a_\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32) = 32
mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7f47f949d000
link("/dev/shm/RxIaDn", "/dev/shm/sem.my_semaphore") = 0
fstat(4, {st_mode=S_IFREG|0644, st_size=32, ...}) = 0
unlink("/dev/shm/RxIaDn")               = 0
close(4)                                = 0
ftruncate(3, 15)                        = 0
mmap(NULL, 15, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f47f949c000
clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f47f9483a10) = 26869
futex(0x7f47f949d000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
```

```
0xffffffff) = 0
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=26869, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---
exit_group(0)                           = ?
+++ exited with 0 +++
```

Демонстрация работы программы

ann@ann:~/os/lab4$ ./main

Enter the name of file for answers: 1.txt

Enter numerous:

78 6 4

34 66 4

33445 345

55555 4 3 2

3 5 0 4

566 33

division by zero error

: Success

ann@ann:~/os/lab4$ cat 1.txt

3.250000

0.128788

96.942029

2314.791666

```
ann@ann:~/os/lab4$ ./main

Enter the name of file for answers: 2.txt

Enter numerous:

6785 4

0 664 4

5678 557 6

ann@ann:~/os/lab4$ cat 2.txt

1696.250000

0.000000

1.698983
```

Вывод

При выполнении данной работы я повторила понятие процессов, то, как следует работать с дочерными процессами, как работать с файловыми дескрипторами, а также при реализации мне понадобились семафоры, поэтому я познакомилась с ними, узнала принцип их и работы и некоторые тонкости. Достаточно сложно учитывать счётчик семафора, поэтому такой способ синхронизации процессов очень неудобный, но он даёт понимание, как устроены более высокоуровневые синхронизаторы, не обязательно семафоры, любые. В процессе выполнения данной лабораторной я поняла, насколько важным является такой способ работы с файлами, как memor mapping, ведь он позволяет не использовать буфер для чтения файла, можно лишь использовать memory mapping определенной области, синхронизация памяти с файлом лежит на ОС, есть возможность использовать файл несколькими процессами одновременно, но это и является проблемой. Хотя использование файловых отображений и безопасно, но в таком случае программисту необходимо следить за тем,

чтобы обмен данными между процессами осуществлялся корректно, например с помощью семафоров, как было сделано мной, ведь иначе могут возникнуть необратимые ошибки, которые нарушат выполнение всей программы.