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.

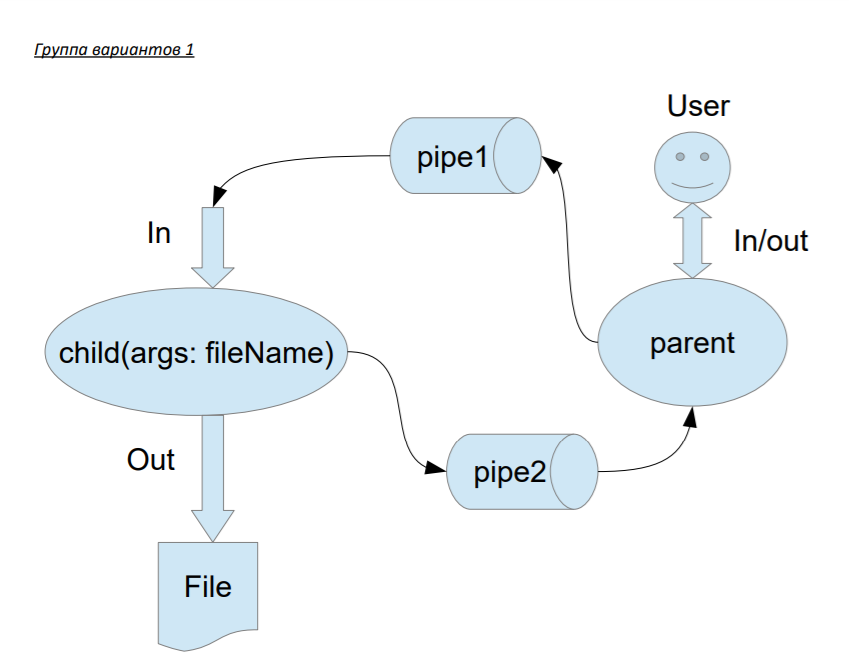
Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 20.12.2021

## **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, 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\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\0\200\0\0\0e\_a\_\0\0\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.