Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №4 по курсу**

**«Операционные системы»**

Студент: Пономарев Н.В.

Группа: М80-207Б-20

Преподаватель: Миронов Е.С.

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

Дата: 07.11.21

Содержание

1. Постановка задачи.
2. Общие сведения о программе.
3. Общий метод и алгоритм решения.
4. Код программы.
5. Демонстрация работы программы.
6. Вывод.

## **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 № 4, вариант 16:**

Составить и отладить программу на языке С++, родительский процесс которой считывает стандартный входной поток, отдает его дочернему процессу, который выполняет проверку строки на правило: «Строка начинается с большой буквы» и, если проверка пройдена успешно, то записывает строку в файл(имя файла также передается от родительского процесса), а в противном случае строки возвращаются в родительский процесс и выводятся в терминал.

## **Общие сведения о программе**

Программа состоит из одного файла main.cpp. В данном файле 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**

* создать два именованных семофора, проверить, задан ли файл для вывода как аргумент, создать два файла **pipe1.txt** и **pipe2.txt,** отобразить их в память c помощью **mmap** как **pipe\_1** и **pipe\_2**, создать дочерний процесс(с помощью **fork**) и обработать возможные ошибки.
* Из родительского процесса: блокируем семафор **input\_semaphore** и освобождаем **output\_semaphore,** записываем в **pipe\_1** очередную строку для проверки: cчитываем все символы, переданные во входной поток, и передаем их через **pipe\_1** в дочерний увеличив файл с помощью функции **ftruncate**, а затем посимвольно записать в файл строку. Будем продолжать такие действия до того момента, как не будет встречен символ конца ввода EOF.
* Из дочернего процесса: блокируем семафоры **input\_semaphore** и **output\_semaphore,** считываем из **pipe\_1** посимвольно строку **s.** Далее выполняем проверку на наше правило. Если строка не прошла проверку, то мы по **pipe\_2** передаем строку в родительский процесс, если же строка прошла проверку на правило, то полученная строка записывается в выходной файл с помощью функции **write**. Когда символы, переданные из родительского процесса закончатся, дочерний процесс закроет файл с помощью функции **close** и завершится с кодом выхода 0.
* После передачи символов входного потока в дочерний процесс, родительский процесс ожидает его завершения. После выполняется считывание строк, не прошедших правило, из **pipe\_2** и выполняется их вывод в консоль. Затем, родительский процесс закрывает семафоры и отзывает **mmap**, используя системный вызов **munmap,** удаляет и закрывает ненужные файлы.

## **Код программы**

#include <iostream>

#include <unistd.h>

#include <sstream>

#include <signal.h>

#include <fcntl.h>

#include <sys/mman.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <pthread.h>

#include <stdio.h>

#include <semaphore.h>

int main(){

const char\* in\_sem\_name = "input\_semaphor";

const char\* out\_sem\_name = "output\_semaphor";

sem\_unlink(in\_sem\_name);

sem\_unlink(out\_sem\_name);

sem\_t\* input\_semaphore = sem\_open(in\_sem\_name, O\_CREAT, 777, 1);

sem\_t\* output\_semaphore = sem\_open(out\_sem\_name, O\_CREAT, 777, 0);

std::string name\_of\_file;

std::string s;

int pipe1 = open("pipe1.txt", O\_RDWR | O\_CREAT, 0777);

int pipe2 = open("pipe2.txt", O\_RDWR | O\_CREAT, 0777);

if(pipe1 == 0){

std::cout << "Error during creating semaphore1**\n**";

exit(1);

}

if(pipe2 == 0){

std::cout << "Error during creating semaphore2**\n**";

exit(1);

}

if(pipe1 == -1){

std::cout << "Error during creating file1 for file mapping**\n**";

exit(1);

}

if(pipe2 == -1){

std::cout << "Error during creating file2 for file mapping**\n**";

exit(1);

}

char\* pipe\_1 = (char\*)mmap(nullptr, getpagesize(), PROT\_READ | PROT\_WRITE, MAP\_SHARED, pipe1, 0);

char\* pipe\_2 = (char\*)mmap(nullptr, getpagesize(), PROT\_READ | PROT\_WRITE, MAP\_SHARED, pipe2, 0);

if (pipe\_1 == MAP\_FAILED){

std::cout << "Error in creating file1 mapping**\n**";

exit(1);

}

if (pipe\_2 == MAP\_FAILED){

std::cout << "Error in creating file2 mapping**\n**";

exit(1);

}

std::cout << "Print input name of file: ";

std::cin >> name\_of\_file;

int f = open(name\_of\_file.c\_str(), O\_WRONLY | O\_CREAT, 0777);

if(f == -1){

std::cout << "Error during creating file**\n**";

exit(1);

}

int id = fork();

switch(id){

case -1:

std::cout << "Error during creating fork**\n**";

exit(1);

break;

case 0: {

printf("[%d] It's child**\n**", getpid());

sem\_wait(output\_semaphore);

sem\_wait(input\_semaphore);

sem\_post(output\_semaphore);

if(dup2(f, 1) == -1){

return 6;

}

s = "";

struct stat st;

if(fstat(pipe1, &st)){

std::cout << "Error during fstat**\n**";

exit(1);

}

int ind = 0, idx = 0;

int length\_1 = 0, length\_2 = 0;

while(ind <= st.st\_size){

if(pipe\_1[ind] != '**\n**'){

s += pipe\_1[ind++];

} else {

if('A' <= s[0] and s[0] <= 'Z'){

s = s + "**\n**";

length\_1 += s.length() \* sizeof(char);

if(ftruncate(f, length\_1)){

std::cout << "Error during ftruncate**\n**";

exit(1);

}

std::cout << s;

} else {

s = "Error in string " + s;

s = s + "**\n**";

length\_2 += s.length() \* sizeof(char);

if(ftruncate(pipe2, length\_2)){

std::cout << "Error during ftruncate**\n**";

exit(1);

}

for(int i = 0; i < s.length(); i++){

pipe\_2[idx++] = s[i];

}

}

ind++;

s = "";

}

}

sem\_post(input\_semaphore);

break;

}

default: {

printf("[%d] It's parent. Child id: %d**\n**", getpid(), id);

sem\_wait(input\_semaphore);

int idx = 0;

int length = 0;

sem\_post(output\_semaphore);

while(std::cin >> s && s != "EOF"){

s = s + "**\n**";

length += s.length() \* sizeof(char);

if(ftruncate(pipe1, length)){

std::cout << "Error during ftrancate**\n**";

exit(1);

}

for(int i = 0; i < s.length(); i++){

pipe\_1[idx++] = s[i];

}

}

sem\_post(input\_semaphore);

s = "";

int ind = 0;

sem\_wait(output\_semaphore);

sem\_wait(input\_semaphore);

struct stat st;

if(fstat(pipe2, &st)){

std::cout << "Error during fstat**\n**";

exit(1);

}

while(ind <= st.st\_size){

if(pipe\_2[ind] != '**\n**'){

s += pipe\_2[ind++];

} else {

s += "**\n**";

std::cout << s;

++ind;

s = "";

}

}

close(pipe1);

close(pipe2);

close(f);

if (munmap(pipe\_1, getpagesize()) == -1){

std::cout << "Munmap1 error has been dected!" << std::endl;

exit(EXIT\_FAILURE);

}

if (munmap(pipe\_2, getpagesize()) == -1){

std::cout << "Munmap2 error has been dected!" << std::endl;

exit(EXIT\_FAILURE);

}

remove("pipe1.txt");

remove("pipe2.txt");

sem\_destroy(input\_semaphore);

sem\_destroy(output\_semaphore);

return 0;

}

}

}

## **Использование утилиты strace**

execve("./a.out", ["./a.out"], 0x7ffcb3a75800 /\* 48 vars \*/) = 0

brk(NULL) = 0x562d2b23c000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffd67cae5c0) = -1 EINVAL (Недопустимый аргумент)

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=70876, ...}) = 0

mmap(NULL, 70876, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f87b4b0f000

close(3) = 0

openat(AT\_FDCWD, "/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`\341\t\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=1956992, ...}) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f87b4b0d000

mmap(NULL, 1972224, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f87b492b000

mprotect(0x7f87b49c1000, 1290240, PROT\_NONE) = 0

mmap(0x7f87b49c1000, 987136, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x96000) = 0x7f87b49c1000

mmap(0x7f87b4ab2000, 299008, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x187000) = 0x7f87b4ab2000

mmap(0x7f87b4afc000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1d0000) = 0x7f87b4afc000

mmap(0x7f87b4b0a000, 10240, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f87b4b0a000

close(3) = 0

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\3405\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=104984, ...}) = 0

mmap(NULL, 107592, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f87b4910000

mmap(0x7f87b4913000, 73728, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7f87b4913000

mmap(0x7f87b4925000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x15000) = 0x7f87b4925000

mmap(0x7f87b4929000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x18000) = 0x7f87b4929000

close(3) = 0

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\0\220\201\0\0\0\0\0\0"..., 832) = 832

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\345Ga\367\265T\320\374\301V)Yf]\223\337"..., 68, 824) = 68

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=157224, ...}) = 0

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\345Ga\367\265T\320\374\301V)Yf]\223\337"..., 68, 824) = 68

mmap(NULL, 140408, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f87b48ed000

mmap(0x7f87b48f4000, 69632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x7000) = 0x7f87b48f4000

mmap(0x7f87b4905000, 20480, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x18000) = 0x7f87b4905000

mmap(0x7f87b490a000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1c000) = 0x7f87b490a000

mmap(0x7f87b490c000, 13432, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f87b490c000

close(3) = 0

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\360q\2\0\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 68, 880) = 68

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2029224, ...}) = 0

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 68, 880) = 68

mmap(NULL, 2036952, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f87b46fb000

mprotect(0x7f87b4720000, 1847296, PROT\_NONE) = 0

mmap(0x7f87b4720000, 1540096, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x25000) = 0x7f87b4720000

mmap(0x7f87b4898000, 303104, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19d000) = 0x7f87b4898000

mmap(0x7f87b48e3000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7f87b48e3000

mmap(0x7f87b48e9000, 13528, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f87b48e9000

close(3) = 0

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\300\363\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=1369352, ...}) = 0

mmap(NULL, 1368336, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f87b45ac000

mmap(0x7f87b45bb000, 684032, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xf000) = 0x7f87b45bb000

mmap(0x7f87b4662000, 618496, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xb6000) = 0x7f87b4662000

mmap(0x7f87b46f9000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x14c000) = 0x7f87b46f9000

close(3) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f87b45aa000

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f87b45a7000

arch\_prctl(ARCH\_SET\_FS, 0x7f87b45a7740) = 0

mprotect(0x7f87b48e3000, 12288, PROT\_READ) = 0

mprotect(0x7f87b46f9000, 4096, PROT\_READ) = 0

mprotect(0x7f87b490a000, 4096, PROT\_READ) = 0

mprotect(0x7f87b4929000, 4096, PROT\_READ) = 0

mprotect(0x7f87b4afc000, 45056, PROT\_READ) = 0

mprotect(0x562d2955b000, 4096, PROT\_READ) = 0

mprotect(0x7f87b4b4e000, 4096, PROT\_READ) = 0

munmap(0x7f87b4b0f000, 70876) = 0

set\_tid\_address(0x7f87b45a7a10) = 11254

set\_robust\_list(0x7f87b45a7a20, 24) = 0

rt\_sigaction(SIGRTMIN, {sa\_handler=0x7f87b48f4bf0, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_SIGINFO, sa\_restorer=0x7f87b49023c0}, NULL, 8) = 0

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7f87b48f4c90, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7f87b49023c0}, 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) = 0x562d2b23c000

brk(0x562d2b25d000) = 0x562d2b25d000

futex(0x7f87b4b0a6bc, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

futex(0x7f87b4b0a6c8, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

statfs("/dev/shm/", {f\_type=TMPFS\_MAGIC, f\_bsize=4096, f\_blocks=1006019, f\_bfree=990187, f\_bavail=990187, f\_files=1006019, f\_ffree=1005910, f\_fsid={val=[0, 0]}, f\_namelen=255, f\_frsize=4096, f\_flags=ST\_VALID|ST\_NOSUID|ST\_NODEV}) = 0

futex(0x7f87b490f390, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

unlink("/dev/shm/sem.input\_semaphor") = 0

unlink("/dev/shm/sem.output\_semaphor") = 0

openat(AT\_FDCWD, "/dev/shm/sem.input\_semaphor", O\_RDWR|O\_NOFOLLOW) = -1 ENOENT (Нет такого файла или каталога)

getpid() = 11254

lstat("/dev/shm/0NMf3a", 0x7ffd67cae1a0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/dev/shm/0NMf3a", O\_RDWR|O\_CREAT|O\_EXCL, 01411) = 3

write(3, "\1\0\0\0\0\0\0\0\200\0\0\0\0\0\0\0\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, 3, 0) = 0x7f87b4b4d000

link("/dev/shm/0NMf3a", "/dev/shm/sem.input\_semaphor") = 0

fstat(3, {st\_mode=S\_IFREG|S\_ISVTX|0411, st\_size=32, ...}) = 0

unlink("/dev/shm/0NMf3a") = 0

close(3) = 0

openat(AT\_FDCWD, "/dev/shm/sem.output\_semaphor", O\_RDWR|O\_NOFOLLOW) = -1 ENOENT (Нет такого файла или каталога)

getpid() = 11254

lstat("/dev/shm/wYsBrc", 0x7ffd67cae1a0) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/dev/shm/wYsBrc", O\_RDWR|O\_CREAT|O\_EXCL, 01411) = 3

write(3, "\0\0\0\0\0\0\0\0\200\0\0\0\0\0\0\0\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, 3, 0) = 0x7f87b4b20000

link("/dev/shm/wYsBrc", "/dev/shm/sem.output\_semaphor") = 0

fstat(3, {st\_mode=S\_IFREG|S\_ISVTX|0411, st\_size=32, ...}) = 0

unlink("/dev/shm/wYsBrc") = 0

close(3) = 0

openat(AT\_FDCWD, "pipe1.txt", O\_RDWR|O\_CREAT, 0777) = 3

openat(AT\_FDCWD, "pipe2.txt", O\_RDWR|O\_CREAT, 0777) = 4

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0) = 0x7f87b4b1f000

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 4, 0) = 0x7f87b4b1e000

fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}) = 0

write(1, "Print input name of file: ", 26) = 26

fstat(0, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}) = 0

read(0, "test.txt\n", 1024) = 9

openat(AT\_FDCWD, "test.txt", O\_WRONLY|O\_CREAT, 0777) = 5

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f87b45a7a10) = 11275

getpid() = 11254

write(1, "[11254] It's parent. Child id: 1"..., 37) = 37

futex(0x7f87b4b20000, FUTEX\_WAKE, 1) = 1

read(0, "afsgdf\n", 1024) = 7

ftruncate(3, 7) = 0

read(0, "fgxc\n", 1024) = 5

ftruncate(3, 12) = 0

read(0, "zxcm,\n", 1024) = 6

ftruncate(3, 18) = 0

read(0, "shdh,\n", 1024) = 6

ftruncate(3, 24) = 0

read(0, "rtj\n", 1024) = 4

ftruncate(3, 28) = 0

read(0, "bxvmbcbv\n", 1024) = 9

ftruncate(3, 37) = 0

read(0, "ndshfk\n", 1024) = 7

ftruncate(3, 44) = 0

read(0, "fh.khdjfkg\n", 1024) = 11

ftruncate(3, 55) = 0

read(0, "YA\n", 1024) = 3

ftruncate(3, 58) = 0

read(0, "JKD\n", 1024) = 4

ftruncate(3, 62) = 0

read(0, "ajsd,j\n", 1024) = 7

ftruncate(3, 69) = 0

read(0, "zgfdhkf\n", 1024) = 8

ftruncate(3, 77) = 0

read(0, "AHD\n", 1024) = 4

ftruncate(3, 81) = 0

read(0, "HJH,\n", 1024) = 5

ftruncate(3, 86) = 0

read(0, "ARJDFJ\n", 1024) = 7

ftruncate(3, 93) = 0

read(0, "ahsjdk\n", 1024) = 7

ftruncate(3, 100) = 0

read(0, "AYSTJK\n", 1024) = 7

ftruncate(3, 107) = 0

read(0, "asffas\n", 1024) = 7

ftruncate(3, 114) = 0

read(0, "vxczv\n", 1024) = 6

ftruncate(3, 120) = 0

read(0, "SDFA\n", 1024) = 5

ftruncate(3, 125) = 0

read(0, "vaxc\n", 1024) = 5

ftruncate(3, 130) = 0

read(0, "DFS\n", 1024) = 4

ftruncate(3, 134) = 0

read(0, "FGD\n", 1024) = 4

ftruncate(3, 138) = 0

read(0, "BC\n", 1024) = 3

ftruncate(3, 141) = 0

read(0, "fdf\n", 1024) = 4

ftruncate(3, 145) = 0

read(0, "xbvc\n", 1024) = 5

ftruncate(3, 150) = 0

read(0, "xcn\n", 1024) = 4

ftruncate(3, 154) = 0

read(0, "", 1024) = 0

futex(0x7f87b4b4d000, FUTEX\_WAKE, 1) = 1

futex(0x7f87b4b4d000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, FUTEX\_BITSET\_MATCH\_ANY) = 0

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=11275, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

fstat(4, {st\_mode=S\_IFREG|0775, st\_size=380, ...}) = 0

write(1, "Error in string afsgdf\n", 23) = 23

write(1, "Error in string fgxc\n", 21) = 21

write(1, "Error in string zxcm,\n", 22) = 22

write(1, "Error in string shdh,\n", 22) = 22

write(1, "Error in string rtj\n", 20) = 20

write(1, "Error in string bxvmbcbv\n", 25) = 25

write(1, "Error in string ndshfk\n", 23) = 23

write(1, "Error in string fh.khdjfkg\n", 27) = 27

write(1, "Error in string ajsd,j\n", 23) = 23

write(1, "Error in string zgfdhkf\n", 24) = 24

write(1, "Error in string ahsjdk\n", 23) = 23

write(1, "Error in string asffas\n", 23) = 23

write(1, "Error in string vxczv\n", 22) = 22

write(1, "Error in string vaxc\n", 21) = 21

write(1, "Error in string fdf\n", 20) = 20

write(1, "Error in string xbvc\n", 21) = 21

write(1, "Error in string xcn\n", 20) = 20

close(3) = 0

close(4) = 0

close(5) = 0

munmap(0x7f87b4b1f000, 4096) = 0

munmap(0x7f87b4b1e000, 4096) = 0

unlink("pipe1.txt") = 0

unlink("pipe2.txt") = 0

exit\_group(0) = ?

+++ exited with 0 +++

## **Демонстрация работы программы**

Print input name of file: test.txt

[11275] It's child

[11254] It's parent. Child id: 11275

afsgdf

fgxc

zxcm,

shdh,

rtj

bxvmbcbv

ndshfk

fh.khdjfkg

YA

JKD

ajsd,j

zgfdhkf

AHD

HJH,

ARJDFJ

ahsjdk

AYSTJK

asffas

vxczv

SDFA

vaxc

DFS

FGD

BC

fdf

xbvc

xcn

**Вывод в консоль:**

Error in string afsgdf

Error in string fgxc

Error in string zxcm,

Error in string shdh,

Error in string rtj

Error in string bxvmbcbv

Error in string ndshfk

Error in string fh.khdjfkg

Error in string ajsd,j

Error in string zgfdhkf

Error in string ahsjdk

Error in string asffas

Error in string vxczv

Error in string vaxc

Error in string fdf

Error in string xbvc

Error in string xcn

**Вывод в файл test.txt:**

YA

JKD

AHD

HJH,

ARJDFJ

AYSTJK

SDFA

DFS

FGD

BC

## **Вывод**

При выполнении данной работы я повторил понятие процессов, то, как следует работать с дочерными процессами, как работать с файловыми дескрипторами, а также при реализации мне понадобились семафоры, поэтому я познакомился с ними, узнал принцип их и работы и некоторые тонкости. Так, изначально я создавал семафор с помощью вызова функции sem\_init, передавая неверный аргумент при его инициализации и вследствие этого у меня возникла проблема, когда, казалось бы, все должно работать корректно, но возникала ошибка. Ее причиной оказалось то, что при инициализации я передавал неверный аргумент, запрещая использоваться ему несколькими процессами. Но я благодарен ей, ведь она помогла мне глубже понять устройство моей программы, так как пришлось разобраться во всем, чтобы найти, где мог возникнуть баг. В процессе выполнения данной лабораторной я понял, насколько важным является такой способ работы с файлами, как file mapping, ведь он позволяет не использовать буфер для чтения файла, можно лишь использовать file 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.