Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №3 по курсу**

**«Операционные системы»**

**РАБОТА С ОБЩЕЙ ПАМЯТЬЮ**

Студент: Муратов Артём Алексеевич

Группа: М8О–212Б–22

Вариант: 9

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2023.

**Постановка задачи**

## Цель работы

Приобретение практических навыков в:

* Освоение принципов работы с файловыми системами
* Обеспечение обмена данных между процессами посредством технологии «File mapping»

## 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 системные сигналы/события и/или через отображаемые файлы (memory-mapped files). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Вариант 9:

В файле записаны команды вида:«число число число». Дочерний процесс производит деление первого числа команда, на последующие числа в команде, а результат выводит в стандартный поток вывода. Если происходит деление на 0, то тогда дочерний и родительский процесс завершают свою работу. Проверка деления на 0 должна осуществляться на стороне дочернего процесса. Числа имеют тип float. Количество чисел может быть произвольным.

**Общие сведения о программе**

Программа компилируется из файла main.c. Для сборки используется CMake. Также используется заголовочные файлы:

#include <stdio.h>

#include <fcntl.h>

#include <stdlib.h>

#include <sys/mman.h>

#include <sys/wait.h>

#include <sys/types.h>

#include <unistd.h>

#include <semaphore.h>

#include <string.h>

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы функции mmap().
2. Создать child.c и main.c.
3. Реализовать в main.c:
   1. Создание двух процессов и общение между программами child.c и main.c с помощью двух односторонних труб.
   2. Открытие заданного файла, считывание чисел из него и отправка считанных чисел в child.c.
   3. Прием и вывод ответа из child.c.
4. Реализовать в child.c:
   1. Считывание из main.c строки чисел с помощью read.
   2. Функцию,решающую задачу варианта.
   3. Передача ответа в main.c с помощью Buff.txt.

**Основные файлы программы**

**Main.cpp**

#include <stdio.h>

#include <fcntl.h>

#include <stdlib.h>

#include <sys/mman.h>

#include <sys/wait.h>

#include <sys/types.h>

#include <unistd.h>

#include <semaphore.h>

#include <string.h>

int main() {

size\_t B\_Size = 1024;

char file\_name[128];

if(fgets(file\_name,128,stdin) == NULL) {

perror("No filename");

return 3;

}

int pos = strlen(file\_name) - 1;

if (file\_name[pos] == '\n') {

file\_name[pos] = '\0';

}

int file = open(file\_name, O\_RDWR, B\_Size/2);

if( file == -1) {

perror("No such file");

return 1;

}

int buf\_file = open("Buff.txt", O\_RDWR | O\_CREAT, 0666);

if( buf\_file == -1) {

perror("No such file");

return 7;

}

if(ftruncate(buf\_file, B\_Size) == -1) {

return 4;

}

char\* mapped\_data = (char\*)mmap(NULL, B\_Size, PROT\_READ | PROT\_WRITE, MAP\_SHARED, buf\_file, 0);

if( mapped\_data == MAP\_FAILED) {

perror("error with memory map");

close(buf\_file);

return 4;

}

//закинь дату из теста в бафф.тхт

if(read(file,mapped\_data,B\_Size/2) == -1) {

return 8;

};

close(file);

pid\_t pid = fork();

if (pid == -1) {

perror("problem with fork");

return 2;

}

if (pid == 0) { //child

execlp("./child", "./child",mapped\_data, NULL);

}

else { //parent

int status;

waitpid(pid, &status,0);

if(!WIFEXITED(status)){

perror("error with child proccess");

return 10;

}

printf("%s \n",mapped\_data + (B\_Size/2 + 1));

}

memset(mapped\_data,0,B\_Size);

if(msync(mapped\_data,B\_Size,MS\_SYNC) == -1) {

perror("failed to sinchronize");

return -1;

}

munmap(mapped\_data, 1024);

close(buf\_file);

return 0;

}

**Child.cpp**

#include <stdio.h>

#include <fcntl.h>

#include <stdlib.h>

#include <sys/mman.h>

#include <sys/wait.h>

#include <sys/types.h>

#include <unistd.h>

#include <semaphore.h>

#include <string.h>

float solving\_func(char \* nums) {

float answer = strtof(nums, NULL);

float divider = 0;

int flag = 0;

for (int i = 0; i < strlen(nums)+1; ++i) {

if ((flag == 0) && ((nums[i] == ' ') || (nums[i] == '\0'))) {

flag = 1;

}

else if ((flag == 1) && (nums[i] != ' ') && (nums[i] != '\0')) {

divider = divider\*10 + ((int)nums[i] - 48);

}

else if ((flag == 1) && ((nums[i] == ' ') || (nums[i] == '\0'))) {

if (divider == 0) {

perror("Division by 0");

exit(1);

}

answer = answer / divider;

divider = 0;

}

}

return answer;

}

int main(int argc, char const \*argv[])

{

size\_t B\_Size = 1024;

char answer\_char[64];

float answer;

size\_t i = B\_Size/2 + 1;

int buf\_file = open("Buff.txt", O\_RDWR | O\_CREAT, 0666);

if( buf\_file == -1) {

perror("No such file");

return 7;

}

if(ftruncate(buf\_file, B\_Size) == -1) {

return 4;

}

char\* mapped\_data = (char\*)mmap(NULL, B\_Size, PROT\_READ | PROT\_WRITE, MAP\_SHARED, buf\_file, 0);

if( mapped\_data == MAP\_FAILED) {

perror("error with memory map");

close(buf\_file);

return 4;

}

answer = solving\_func(mapped\_data);

if(gcvt(answer, 12, answer\_char) == NULL) {

perror("gcvt problem");

return 6;

}

strcpy(mapped\_data + i,answer\_char);

if(msync(mapped\_data,B\_Size,MS\_SYNC) == -1) {

perror("failed to sinchronize");

return -1;

}

return 0;

}

**CMakeList.txt**

cmake\_minimum\_required (VERSION 3.22)

project(lab\_os\_3)

set(CMAKE\_C\_STANDARD 17)

set(CMAKE\_C\_STANDARD\_REQUIRED ON)

add\_executable(child child.c)

add\_executable(main main.c)

**Пример работы**

**1.** 10 2 5

virtbox@virtbox-VirtualBox:~/OS\_labs/lab\_os\_1/build$ ./main

/home/virtbox/OS\_labs/lab\_os\_1/tests/test\_1.txt

1

**2.** 120 2 5 4 3

virtbox@virtbox-VirtualBox:~/OS\_labs/lab\_os\_1/build$ ./main

/home/virtbox/OS\_labs/lab\_os\_1/tests/test\_2.txt

1

**3.** 2 0

virtbox@virtbox-VirtualBox:~/OS\_labs/lab\_os\_1/build$ ./main

/home/virtbox/OS\_labs/lab\_os\_1/tests/test\_3.txt

Division by 0

**4.** 127 34 23 3 5

virtbox@virtbox-VirtualBox:~/OS\_labs/lab\_os\_1/build$ ./main

/home/virtbox/OS\_labs/lab\_os\_1/tests/test\_4.txt

0.0108269397169

**5.** 0

virtbox@virtbox-VirtualBox:~/OS\_labs/lab\_os\_1/build$ ./main

/home/virtbox/OS\_labs/lab\_os\_1/tests/test\_5.txt

0

**6.** 0 1234 5678 90

virtbox@virtbox-VirtualBox:~/OS\_labs/lab\_os\_1/build$ ./main

/home/virtbox/OS\_labs/lab\_os\_1/tests/test\_6.txt

0

**Вывод**