****Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работ №4 по курсу**

**«Операционные системы»**

Группа: М80 – 206Б-18

Студент: Касимов М.М.

Преподаватель: Соколов А.А.

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

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2019.

**Содержание**

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

**Постановка задачи**.

Рекурсивное вычисление факториала, где каждый отдельный уровень рекурсии

вычисляется в отдельном процессе.

**Общие сведения о программе**

Программа компилируется из одного файла lab4.c. В данном файле используются заголовочные файлы stdio.h, unistd.h, stdbool.h, stdlib.h, wait.h, sys/types.h, semaphore.h, fcntl.h. В программе используются следующие вызовы:

1. **mkstemp** – для создания временного файла
2. **sem\_open** – для создания нового именованного семафора или открытия уже существующего.
3. **sem\_unlink** – для удаления именованного семафора.
4. **fork** – для создания дочернего процесса.
5. **sem\_post** – для увеличения(разблокировки) семафора.
6. **sem\_wait –** для уменьшения(блокировки) семафора.
7. **mmap –** для отображения файла в адресное пространство процесса.

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1.Произвести проверки корректности входных данных.

2.Создать временный файл для последующего маппинга, заполнить его нужным количеством нулевых символов, произвести маппинг.

3.Создать два семафора, для синхронизации работы с файлом, отображенным в память.

4. В рекурсивной функции вычисления факториала манипулировать семафорами, чтобы родительский процесс записал и ждал пока дочерний не прочитает и не напишет в файл.

**Основные файлы программы**.

**Файл lab4.c**

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

#include <wait.h>

#include <stdlib.h>

#include <stdbool.h>

#include <sys/mman.h>

#include <fcntl.h>

#include <semaphore.h>

#include <sys/stat.h>

#include <string.h>

#define BUFFER\_SIZE 10

//создание временного файла.

int create\_tmp() {

char tmp\_name[] = "/tmp/tmpf.XXXXXX";

int tmp\_fd = mkstemp(tmp\_name);

if ( tmp\_fd == -1) {

printf("error\n");

exit(1);

}

int size = BUFFER\_SIZE + 1;

char array[size];

for ( int i = 0; i < size; ++i ) {

array[i] = '\0';

}

write(tmp\_fd, array, size);

return tmp\_fd;

}

//рекурсивное вычисление факториала.

//создаём семафор и заставляем родительский процесс ждать пока дочерний не завершит свои вычисления.

unsigned long long fact(int n, int\* map){

if (n == 0){

return 1;

}

else {

const char\* out\_sem\_name = "/o\_s";

sem\_unlink(out\_sem\_name);

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

pid\_t proc = fork();

if (proc < 0){

printf("Error: fork\n");

exit(1);

}

if (proc == 0){ //дочерний процесс

unsigned long long res;

res = fact(n - 1, map);

map[0] = res;

sem\_post(out);

exit(0);

}

if (proc > 0) { //родительский процесс

sem\_wait(out);

unsigned long long res;

res = map[0];

return n \* (res);

}

}

}

int main(){

char a[132] = "Instruction.\n Enter only one nonnegative integer number less than 14. In case of incorrect input, the program will simply terminate:";

for ( int i = 0 ; i < 132 ; ++i ) {

write(STDOUT\_FILENO,&a[i],sizeof(char));

} //вывод короткой инструкции.

int flag = 0,flagPlus = 0,flagTabs = 0,flagNumber = 0;

int n = 0;

char c;

while(true) {

read(STDIN\_FILENO,&c,1);

if (c <= '9' && c >= '0') {

flagPlus++;

flagNumber++;

n \*= 10;

n += c - '0';

continue;

}

if (c=='\n')

break;

if (c == '+' && flagPlus == 0) {

flagPlus++;

continue;

}

if ((c == ' ' || c == '\t') && (flagTabs == 0)) {

continue;

}

else

++flag;

}

if (flag != 0 || flagNumber == 0) {

return 0;

}

if (n > 13)

return 0; // парсер.

unsigned long long k;

int fd = create\_tmp();// дескриптор временного файла.

//mapping файла.

int\* map = (int\*) mmap(NULL,10,PROT\_WRITE | PROT\_READ, MAP\_SHARED, fd, 0);

if (map == NULL) {

printf("error mapping\n");

exit(1);

}

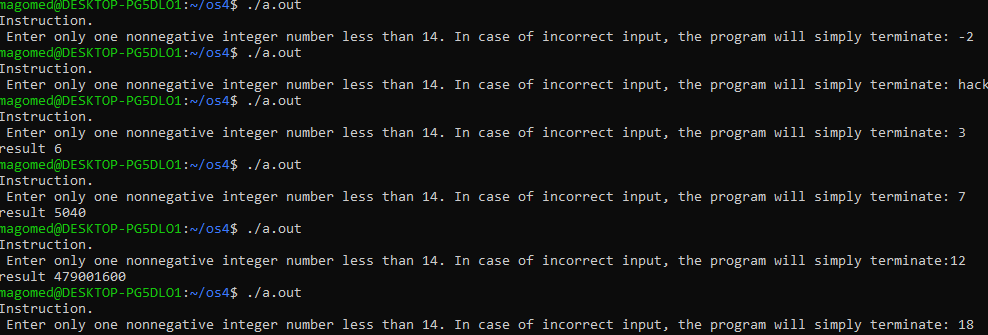
k=fact(n,map);

printf("result %lld\n",k);

return 0;

}

**Демонстрация работы программы.**



**Вывод.**

Я научился создавать семафоры, используя системный вызов sem\_open() и пользоваться ими, обрел навыки межпроцессного взаимодействия посредством файла, над которым производился маппинг. Улучшил навыки программирования на языке программирования Си.