**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторная работа № 4**

**По курсу:  
«Операционные системы»**

Студент: Патрикеева Лидия

Группа: 80-207Б-18

Преподаватель: Миронов Е.С.

Дата:

Оценка:

Москва, 2019

**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 решения 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 задания № 12:

Написание собственного простого целочисленного калькулятора с операцией "\*", " /".

В дочернем процессе должны происходить вычисления выражений. В родительском процессе должны происходить вывод/ввод и передача их дочернему процессу.

**Исходный код**

lab4.c

#include <Windows.h>

#include <stdio.h>

int main(){

STARTUPINFO si;

PROCESS\_INFORMATION pi;

ZeroMemory(&si, sizeof(STARTUPINFO));

HANDLE mutex;

mutex = CreateMutex(NULL, TRUE, "myMutex");

if(mutex == NULL){

printf("Cannot CreateMutex %ld\n", GetLastError());

return 1;

}

HANDLE mapFile;

const unsigned char\* fName = "mapFile";

mapFile = CreateFileMapping(INVALID\_HANDLE\_VALUE, NULL, PAGE\_READWRITE, 0, 4, fName);

if(mapFile == NULL){

printf("Cannot CreateFileMapping %ld\n", GetLastError());

return 1;

}

unsigned char \*buff = (unsigned char\*)MapViewOfFile(mapFile, FILE\_MAP\_ALL\_ACCESS, 0, 0, 4);

if (buff == NULL){

printf("Cannot MapViewOfFile %ld\n", GetLastError());

CloseHandle(mapFile);

return 1;

}

if(!(CreateProcess("child.exe", NULL, NULL, NULL, FALSE, NORMAL\_PRIORITY\_CLASS, NULL, NULL, &si, &pi))){

fprintf(stdout,"Cannot CreateProcess: Error %ld\n", GetLastError());

return 1;

}

while(1){

unsigned char \*bytes;

int num1, num2;

char symbol;

printf("Enter expression : ");

if(scanf("%d", &num1) == EOF) break;

scanf(" %c", &symbol);

scanf("%d", &num2);

bytes = (unsigned char\*)(&num1);

for(int i = 0; i < 4; i++){

\*(buff+i) = \*(bytes+i);

}

ReleaseMutex(mutex);

WaitForSingleObject(mutex, INFINITE);

bytes = (unsigned char\*)(&num2);

for(int i = 0; i < 4; i++){

\*(buff+i) = \*(bytes+i);

}

ReleaseMutex(mutex);

WaitForSingleObject(mutex, INFINITE);

\*(buff) = symbol;

ReleaseMutex(mutex);

WaitForSingleObject(mutex, INFINITE);

int res;

res = \*((int\*)buff);

printf("Result = %d\n",res);

}

CloseHandle(mutex);

CloseHandle(mapFile);

TerminateProcess(pi.hProcess, NO\_ERROR);

return 0;

}

child.c

#include <Windows.h>

#include <stdio.h>

int main(){

const unsigned char\* fName = "mapFile";

HANDLE mapFile;

mapFile = OpenFileMapping(FILE\_MAP\_ALL\_ACCESS, FALSE, fName);

if(mapFile == NULL){

printf("Cannot OpenFileMapping %ld\n", GetLastError());

}

unsigned char \*buff = (unsigned char\*)MapViewOfFile(mapFile, FILE\_MAP\_ALL\_ACCESS, 0, 0, 4);

if (buff == NULL){

printf("Cannot MapViewOfFile %ld\n", GetLastError());

CloseHandle(mapFile);

return 1;

}

HANDLE mutex;

mutex = OpenMutex(MUTEX\_ALL\_ACCESS, FALSE, "myMutex");

while(1){

unsigned char \*bytes;

int res;

int num1, num2;

unsigned char symbol;

WaitForSingleObject(mutex, INFINITE);

num1 = \*((int\*)buff);

ReleaseMutex(mutex);

WaitForSingleObject(mutex, INFINITE);

num2 = \*((int\*)buff);

ReleaseMutex(mutex);

WaitForSingleObject(mutex, INFINITE);

symbol = \*buff;

if(symbol == '\*'){

res = num1 \* num2;

}

else if(symbol == '/'){

if(num2\*num1 == 0){

res = 0;

}

else {

res = num1 / num2;

}

}

bytes = (unsigned char\*)(&res);

for(int i = 0; i < 4; i++){

\*(buff+i) = \*(bytes+i);

}

ReleaseMutex(mutex);

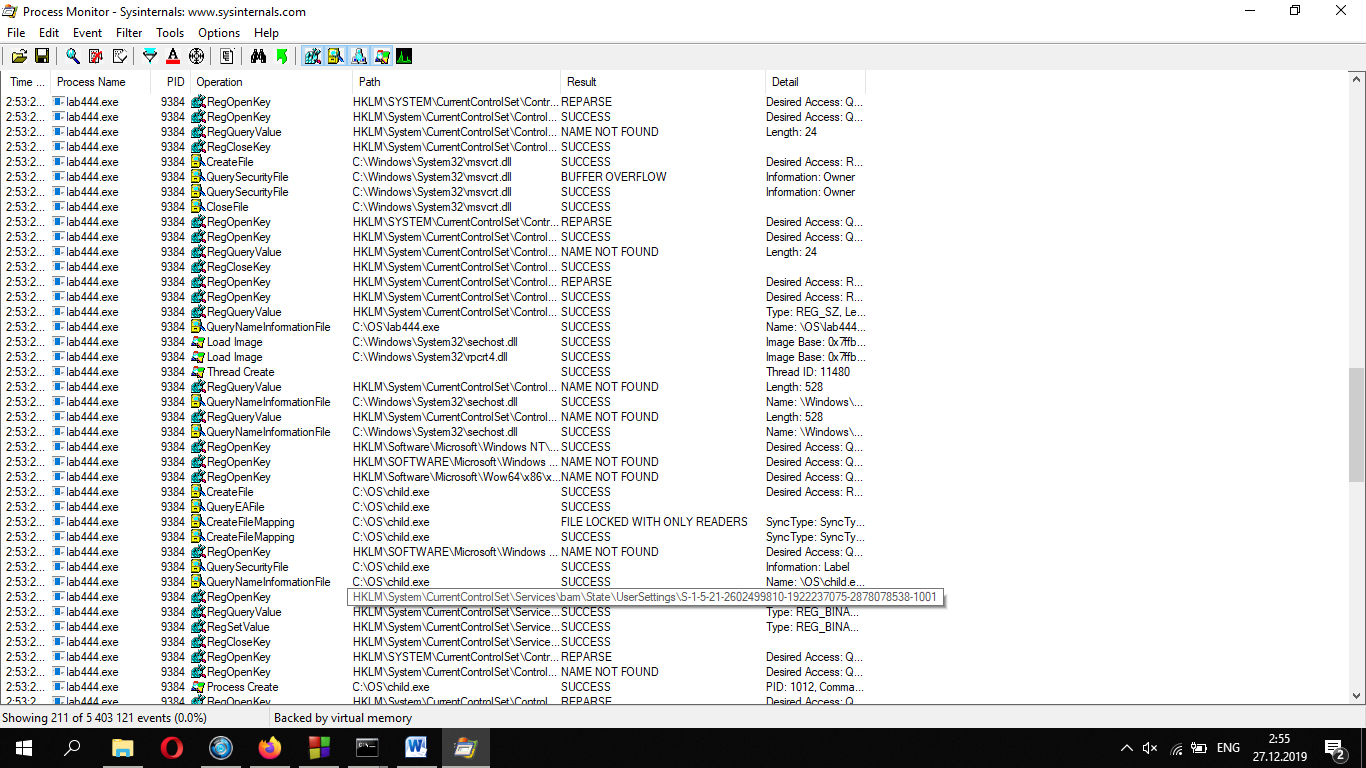
}

return 0;

}

**Procmon**

Ниже представлен скриншот из программы Process Monitor, которая позволяет наблюдать в реальном масштабе времени за действиями различных процессов в среде операционной системы Windows.



**Вывод**

В процессе выполнения данной лабораторной работы я научилась обеспечивать обмен данными между процессами посредством технологии «File mapping». До этого я умела передавать данные между двумя процессами только с помощью пайпов. Как мне кажется для решения подобных задач проще использовать как раз таки пайпы. Для реализации такой программы с помощью технологии «File mapping» необходимо было прибегнуть к помощи мьютексов, чтобы программа могла последовательно обрабатывать данные.

Однако у такого способа есть свои недостатки. Используя данную технологию, у меня не получилось реализовать задачу, когда надо было перенаправить ввод и вывод процесса в хендл, возвращаемый функцией CreateFileMapping.