Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №3 по курсу**

**«Операционные системы»**

Студент: Бойцов Иван Алексеевич

Группа: М8О–212Б–22

Вариант: 5

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2024

# Лабораторная работа №3

## Цель работы

Приобретение практических навыков в:

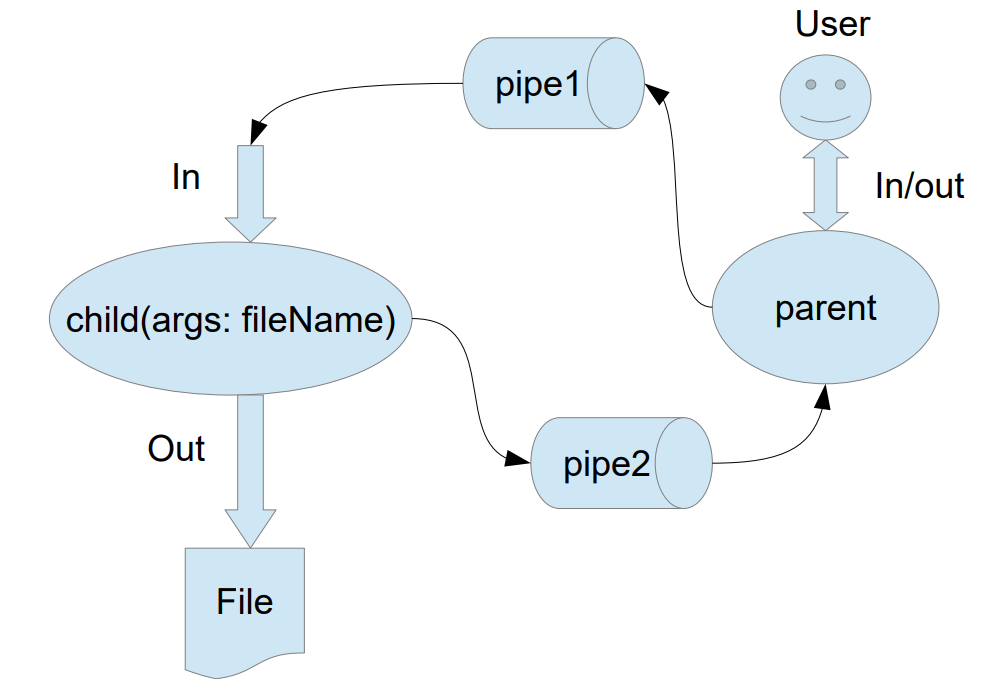
1. Освоение принципов работы с файловыми системами
2. Обеспечение обмена данных между процессами посредством технологии «File mapping»

## 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.

## Группа вариантов 1



Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса пишет имя файла, которое будет передано при создании дочернего процесса. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс передает команды пользователя через pipe1, который связан с стандартным входным потоком дочернего процесса. Дочерний процесс принеобходимости передает данные в родительский процесс через pipe2. Результаты своей работы дочерний процесс пишет в созданный им файл. Допускается просто открыть файл и писать туда, не перенаправляя стандартный поток вывода.

## Вариант 5

Пользователь вводит команды вида: «число». Далее это число передается от родительского процесса в дочерний. Дочерний процесс производит проверку на простоту. Если число составное, то в это число записывается в файл. Если число отрицательное или простое, то тогда дочерний и родительский процессы завершаются.

## Решение

Отладить программу и написать решение =)

## parent.cpp

#include <iostream>

#include <windows.h>

#include <string>

using namespace std;

*// Функция для преобразования строки в широкую строку*

wstring stringToWString(const string& str) {

int len;

int slength = (int)str.length() + 1;

len = MultiByteToWideChar(CP\_UTF8, 0, str.c\_str(), slength, 0, 0);

wstring r(len, L'\0');

MultiByteToWideChar(CP\_UTF8, 0, str.c\_str(), slength, &r[0], len);

return r;

}

int main() {

setlocale(LC\_ALL, "Russian");

HANDLE hMapFile;

LPCTSTR pBuf;

*// Создаем отображаемый файл*

hMapFile = CreateFileMapping(

INVALID\_HANDLE\_VALUE, *// используем файл подкачки*

NULL, *// стандартные атрибуты защиты*

PAGE\_READWRITE, *// доступ на чтение/запись*

0, *// размер файла в старших 32 битах*

256, *// размер файла в младших 32 битах*

TEXT("Global\\MyFileMappingObject")); *// имя отображаемого файла*

if (hMapFile == NULL) {

cerr << "Could not create file mapping object (" << GetLastError() << ")." << endl;

system("pause");

return 1;

}

pBuf = (LPTSTR)MapViewOfFile(hMapFile, *// дескриптор отображаемого файла*

FILE\_MAP\_ALL\_ACCESS, *// доступ на чтение/запись*

0,

0,

256);

if (pBuf == NULL) {

cerr << "Could not map view of file (" << GetLastError() << ")." << endl;

CloseHandle(hMapFile);

system("pause");

return 1;

}

*// Создание дочернего процесса*

STARTUPINFO si;

PROCESS\_INFORMATION pi;

ZeroMemory(&si, sizeof(si));

si.cb = sizeof(si);

ZeroMemory(&pi, sizeof(pi));

string filename;

cout << "Введите имя файла для записи результатов: ";

getline(cin, filename);

string commandLine = "C:\\Users\\ivanb\\source\\repos\\OS\_lab3\_child\\x64\\Debug\\OS\_lab3\_child.exe " + filename;

wstring wCommandLine = stringToWString(commandLine);

*// Преобразование строки в формат, приемлемый для CreateProcess*

if (!CreateProcess(NULL, &wCommandLine[0], NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi)) {

cerr << "CreateProcess failed (" << GetLastError() << ")." << endl;

UnmapViewOfFile(pBuf);

CloseHandle(hMapFile);

system("pause");

return 1;

}

string input;

while (true) {

cout << "Введите число: ";

getline(cin, input);

*// Копируем введенное число в отображаемую область памяти*

CopyMemory((PVOID)pBuf, input.c\_str(), (input.size() + 1) \* sizeof(TCHAR));

*// Ждем завершения дочернего процесса*

WaitForSingleObject(pi.hProcess, INFINITE);

DWORD exitCode;

if (GetExitCodeProcess(pi.hProcess, &exitCode)) {

if (exitCode == 0) break;

}

else {

cerr << "Failed to get exit code (" << GetLastError() << ")." << endl;

}

}

UnmapViewOfFile(pBuf);

CloseHandle(hMapFile);

CloseHandle(pi.hProcess);

CloseHandle(pi.hThread);

system("pause");

return 0;

}

## child.cpp

#include <iostream>

#include <windows.h>

#include <fstream>

#include <string>

#include <cctype>

#include <algorithm>

#include <cmath>

using namespace std;

bool is\_prime(int number) {

if (number <= 1) return false;

if (number <= 3) return true;

if (number % 2 == 0 || number % 3 == 0) return false;

for (int i = 5; i \* i <= number; i += 6) {

if (number % i == 0 || number % (i + 2) == 0) return false;

}

return true;

}

*// Функция для проверки, является ли строка числом*

bool is\_number(const string& s) {

return !s.empty() && all\_of(s.begin(), s.end(), ::isdigit);

}

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

if (argc < 2) {

cerr << "Usage: " << argv[0] << " <filename>\n";

return 1;

}

const char\* filename = argv[1];

ofstream file(filename, ios::app);

if (!file.is\_open()) {

cerr << "Failed to open file: " << filename << endl;

return 1;

}

HANDLE hMapFile;

LPCSTR pBuf;

hMapFile = OpenFileMapping(

FILE\_MAP\_ALL\_ACCESS, *// доступ на чтение/запись*

FALSE, *// наследование дескриптора*

L"Global\\MyFileMappingObject"); *// имя отображаемого файла*

if (hMapFile == NULL) {

cerr << "Could not open file mapping object (" << GetLastError() << ")." << endl;

return 1;

}

pBuf = (LPCSTR)MapViewOfFile(hMapFile, *// дескриптор отображаемого файла*

FILE\_MAP\_ALL\_ACCESS, *// доступ на чтение/запись*

0,

0,

256);

if (pBuf == NULL) {

cerr << "Could not map view of file (" << GetLastError() << ")." << endl;

CloseHandle(hMapFile);

return 1;

}

*// Чтение числа из отображаемого файла*

string input(pBuf);

if (!is\_number(input)) {

cerr << "Invalid input: '" << input << "' is not a number." << endl;

UnmapViewOfFile(pBuf);

CloseHandle(hMapFile);

file.close();

return 1;

}

int number;

try {

number = stoi(input);

}

catch (const invalid\_argument& e) {

cerr << "Invalid input: " << e.what() << endl;

UnmapViewOfFile(pBuf);

CloseHandle(hMapFile);

file.close();

return 1;

}

catch (const out\_of\_range& e) {

cerr << "Number out of range: " << e.what() << endl;

UnmapViewOfFile(pBuf);

CloseHandle(hMapFile);

file.close();

return 1;

}

if (number < 0 || is\_prime(number)) {

UnmapViewOfFile(pBuf);

CloseHandle(hMapFile);

file.close();

return 0;

}

else {

file << number << endl;

}

UnmapViewOfFile(pBuf);

CloseHandle(hMapFile);

file.close();

return 1;

}

## Вывод

В процессе выполнения лабораторной работы были получены практические навыки в обеспечении обмена данных между процессами посредством технологии "File mapping" в операционной системе Windows.