Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №1 по курсу**

**«Операционные системы»**

Студент: Муханов Андрей Ильич

Группа: М8О-209Б-23

Вариант: 12

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2024

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Демонстрация работы программы
4. Выводы

**Репозиторий**

https://github.com/andreymuch/OC\_MAI

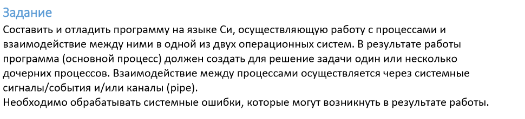
**Постановка задачи**

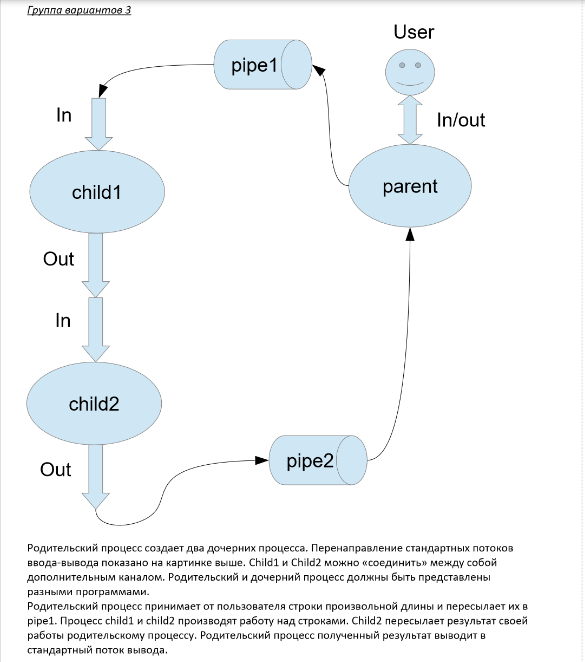
**Цель работы**

Приобретение практических навыков в:

1. Управление процессами в ОС
2. Обеспечение работы данных между процессами

**Задание**

****

****

****

**Демонстрация работы программы**

**Введите строку: hgjfdhgjkfhdkj hgjkfdhkjghjkfdhkjghd hgjfhgjfjgfj ghjfgjfhgjhf**

**Успешное подключение к pipe3 в child2**

**Результат: HGJFDHGJKFHDKJ HGJKFDHKJGHJKFDHKJGHD HGJFHGJFJGFJ GHJFGJFHGJHF  
  
  
  
  
Код**

**Родительский процесс  
#include <windows.h>**

**#include <iostream>**

**#include <string>**

**#include <locale>**

**#define BUFFER\_SIZE 1024**

**int main() {**

**HANDLE hPipe1, hPipe2; *// Дескрипторы для именованных каналов (pipe)***

**PROCESS\_INFORMATION pi1, pi2; *// Структуры для хранения информации о процессе (дескриптор и идентификаторы)***

**STARTUPINFO si1, si2; *// Структуры, содержащие информацию о запуске процесса, такие как окна, дескрипторы и т.д.***

**char buffer[BUFFER\_SIZE];**

**DWORD dwWritten, dwRead;**

**std::string input;**

***// Создаем именованный канал pipe1 для передачи данных в процессе***

**hPipe1 = CreateNamedPipe("\\\\.\\pipe\\Pipe1", *// Имя канала***

**PIPE\_ACCESS\_OUTBOUND, *// Канал для записи данных***

**PIPE\_TYPE\_BYTE | PIPE\_WAIT, *// PIPE\_TYPE\_BYTE: побайтовая передача данных. PIPE\_WAIT: режим ожидания, пока данные не будут записаны***

**PIPE\_UNLIMITED\_INSTANCES, BUFFER\_SIZE, BUFFER\_SIZE, 0, NULL); *// BUFFER\_SIZE: размер буфера для чтения и записи. PIPE\_UNLIMITED\_INSTANCES: неограниченное количество подключений***

**if (hPipe1 == INVALID\_HANDLE\_VALUE) {**

**std::cerr << "Не удалось создать pipe1" << std::endl;**

**return 1;**

**}**

***// Создаем именованный канал pipe2 для получения данных***

**hPipe2 = CreateNamedPipe(TEXT("\\\\.\\pipe\\Pipe2"),**

**PIPE\_ACCESS\_INBOUND, *// Канал для чтения данных***

**PIPE\_TYPE\_BYTE | PIPE\_WAIT,**

**PIPE\_UNLIMITED\_INSTANCES, BUFFER\_SIZE, BUFFER\_SIZE, 0, NULL);**

**if (hPipe2 == INVALID\_HANDLE\_VALUE) {**

**std::cerr << "Не удалось создать pipe2" << std::endl;**

**return 1;**

**}**

***// Инициализируем структуры для запуска процессов***

**ZeroMemory(&si1, sizeof(si1)); *// Обнуляем структуру (инициализируем нулями)***

**si1.cb = sizeof(si1); *// Устанавливаем размер структуры STARTUPINFO***

**ZeroMemory(&si2, sizeof(si2));**

**si2.cb = sizeof(si2);**

***// Запускаем дочерний процесс 2***

**if (!CreateProcessA(TEXT("F:\\Dokument\\program\\CPP\\OC\\build\\lab1\\Debug\\child2.exe"), NULL, NULL, NULL, FALSE, 0, NULL, NULL, &si2, &pi2)) {**

**std::cerr << "Не удалось запустить процесс child2" << std::endl;**

**return 1;**

**}**

***// Запускаем дочерний процесс 1***

**if (!CreateProcess(TEXT("F:\\Dokument\\program\\CPP\\OC\\build\\lab1\\Debug\\child1.exe"), NULL, NULL, NULL, FALSE, 0, NULL, NULL, &si1, &pi1)) {**

**std::cerr << "Не удалось запустить процесс child1" << std::endl;**

**return 1;**

**}**

**std::cout << "Введите строку: ";**

**std::getline(std::cin, input);**

**std::cout << "Отправляем: " << input << std::endl;**

***// Обнуляем буфер перед записью данных***

**memset(buffer, 0, sizeof(buffer));**

***// Записываем данные в pipe1 (в канал)***

***/\****

***hPipe1: дескриптор канала, в который записываем данные.***

***input.c\_str(): строка данных для отправки. Метод c\_str() преобразует std::string в C-строку (массив символов).***

***input.size() + 1: размер данных для записи, включая нулевой символ в конце строки.***

***&dwWritten: указатель на переменную, в которую будет записано количество отправленных байтов.***

***NULL: параметр OVERLAPPED для асинхронных операций, NULL означает синхронный режим.***

***\*/***

**WriteFile(hPipe1, input.c\_str(), input.size() + 1, &dwWritten, NULL);**

**CloseHandle(hPipe1);**

***// Ожидаем завершения второго дочернего процесса***

***/\****

***pi2.hProcess: дескриптор процесса, который был создан с помощью CreateProcess.***

***INFINITE: означает, что функция WaitForSingleObject будет ждать бесконечно, пока процесс не завершится.***

***\*/***

**WaitForSingleObject(pi2.hProcess, INFINITE);**

***/\****

***Чтение данных из pipe2:***

***hPipe2: дескриптор канала, из которого мы будем читать данные.***

***buffer: массив для хранения прочитанных данных.***

***BUFFER\_SIZE: максимальное количество байтов для чтения.***

***&dwRead: указатель на переменную, в которую будет записано количество прочитанных байтов.***

***NULL: параметр OVERLAPPED для асинхронных операций, NULL означает синхронный режим.***

***\*/***

***// Читаем данные из pipe2 после завершения дочернего процесса***

**ReadFile(hPipe2, buffer, BUFFER\_SIZE, &dwRead, NULL);**

**CloseHandle(hPipe2);**

***// Выводим результат***

**std::cout << "Результат: " << buffer << std::endl;**

***// Закрываем дескрипторы дочерних процессов***

***// После завершения процесса необходимо закрыть его дескрипторы: pi1.hProcess и pi1.hThread***

**CloseHandle(pi1.hProcess);**

**CloseHandle(pi1.hThread);**

**CloseHandle(pi2.hProcess);**

**CloseHandle(pi2.hThread);**

**return 0;**

**}**

**Child1.exe  
#include <windows.h>**

**#include <iostream>**

**#include <cctype>**

**#include <locale>**

**#define BUFFER\_SIZE 1024**

**int main() {**

**HANDLE hPipe, hOut;**

**char buffer[BUFFER\_SIZE];**

**DWORD dwRead, dwWritten;**

**std::locale loc("ru\_RU.UTF-8");**

**SetConsoleOutputCP(CP\_UTF8);**

**SetConsoleCP(CP\_UTF8);**

***// Подключение к pipe1 для чтения данных от родительского процесса***

**hPipe = CreateFile("\\\\.\\pipe\\Pipe1",**

**GENERIC\_READ,*//прова на чтение***

**0, NULL, OPEN\_EXISTING, 0, NULL);**

**if (hPipe == INVALID\_HANDLE\_VALUE) {**

**std::cerr << "Failed to connect to pipe1" << std::endl;**

**return 1;**

**}**

***// Инициализация буфера***

**memset(buffer, 0, sizeof(buffer));**

***// Чтение данных из pipe1***

**ReadFile(hPipe, buffer, BUFFER\_SIZE, &dwRead, NULL);**

**CloseHandle(hPipe);**

***// Преобразование строки в верхний регистр***

**for (int i = 0; buffer[i]; i++) {**

**buffer[i] = toupper(buffer[i], loc);**

**}**

**while (true) {**

**hOut = CreateFile("\\\\.\\pipe\\Pipe3",**

**GENERIC\_WRITE,*//прова на запись***

**0, NULL, OPEN\_EXISTING, 0, NULL);**

**if (hOut != INVALID\_HANDLE\_VALUE) {**

**std::cerr << "Успешное подключение к pipe3 в child2" << std::endl;**

**break;**

**}**

**else{**

**std::cerr << "неудачное подключение к pipe3 в child2" << std::endl;**

**}**

**Sleep(100);**

**}**

**WriteFile(hOut, buffer, dwRead, &dwWritten, NULL);**

**CloseHandle(hOut);**

**return 0;**

**}**

**Chuld2.exe  
#include <windows.h>**

**#include <iostream>**

**#include <string>**

**#define BUFFER\_SIZE 1024**

**void remove\_duplicate\_spaces(char\* str) {**

**int count = 0;**

**bool last\_was\_space = false;**

**for (int i = 0; str[i]; i++) {**

**if (str[i] != ' ') {**

**str[count++] = str[i];**

**last\_was\_space = false;**

**} else if (!last\_was\_space) {**

**str[count++] = str[i];**

**last\_was\_space = true;**

**}**

**}**

**str[count] = '\0';**

**}**

**int main() {**

**HANDLE hPipe;**

**char buffer[BUFFER\_SIZE];**

**DWORD dwRead, dwWritten;**

**memset(buffer, 0, sizeof(buffer));**

**HANDLE hPipe3 = CreateNamedPipe("\\\\.\\pipe\\Pipe3",**

**PIPE\_ACCESS\_INBOUND,**

**PIPE\_TYPE\_BYTE | PIPE\_WAIT,**

**PIPE\_UNLIMITED\_INSTANCES,**

**BUFFER\_SIZE,**

**BUFFER\_SIZE,**

**0,**

**NULL);**

**if (hPipe3 == INVALID\_HANDLE\_VALUE) {**

**std::cerr << "Не удалось создать pipe3" << std::endl;**

**return 1;**

**}**

***// Ожидание подключения (ожидает, пока child1 подключится и начнет запись)***

**BOOL connected = ConnectNamedPipe(hPipe3, NULL) ? TRUE : (GetLastError() == ERROR\_PIPE\_CONNECTED);**

**if (!connected) {**

**std::cerr << "Ошибка при подключении к pipe3" << std::endl;**

**CloseHandle(hPipe3);**

**return 1;**

**}**

**ReadFile(hPipe3, buffer, BUFFER\_SIZE, &dwRead, NULL);**

**CloseHandle(hPipe3);**

**remove\_duplicate\_spaces(buffer);**

***// Подключение к pipe2***

**hPipe = CreateFile("\\\\.\\pipe\\Pipe2",**

**GENERIC\_WRITE,**

**0, NULL, OPEN\_EXISTING, 0, NULL);**

**if (hPipe == INVALID\_HANDLE\_VALUE) {**

**std::cerr << "Failed to connect to pipe2" << std::endl;**

**return 1;**

**}**

**WriteFile(hPipe, buffer, dwRead, &dwWritten, NULL);**

**CloseHandle(hPipe);**

**return 0;**

**}**

**Выводы**

В процессе работы были изучены механизмы взаимодействия между процессами через каналы. Программа успешно выполняет обработку строк в соответствии с заданием, преобразуя их в верхний регистр и убирая лишние пробелы.