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: Тарасов Егор Дмитриевич
Группа: М8О-209Б-23
Вариант: 19
Преподаватель: Ядров Артём Леонидович
Оценка:_____
Дата:_____
Подпись: _____

Москва, 2024

Содержание

Репозиторий

https://github.com/EgorTarasov1/os-mai-labs-v19

Постановка задачи

Цель работы

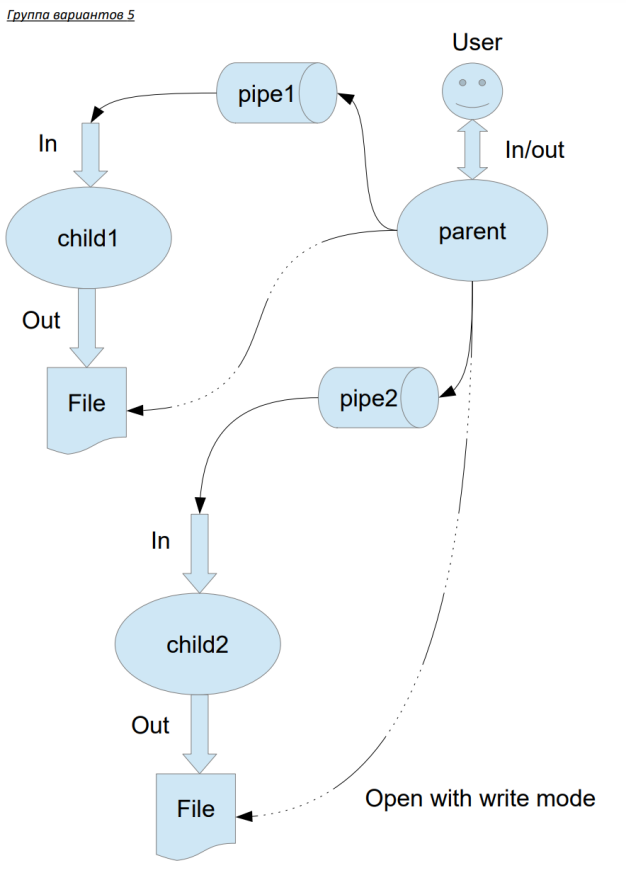
Приобретение практических навыков в:

1. Управление процессами в ОС
2. Обеспечение обмена данных между процессами посредством каналов

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 каналы (pipe). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

19 вариант) Правило фильтрации: с вероятностью 80% строки отправляются в pipe1, иначе в pipe2. Дочерние процессы удаляют все гласные из строк.



Общие сведения о программе

Программы собираются и компилируются при помощи CMake. В проекте существуют две программы: parent.cpp и main.cpp, предназначенные для описания родительского и дочерних процессов соответственно. В программе используются следующие функции Windows API:

1. **CreatePipe** – для создания каналов (аналог *pipe* в Unix)
2. **CreateProcess** – для создания дочерних процессов (аналог *fork* в Unix)
3. **WriteFile** – для записи данных в каналы (аналог *write* в Unix)
4. **ReadFile** – для чтения данных из каналов (аналог *read* в Unix)
5. **CloseHandle** – для закрытия дескрипторов (аналог *close* в Unix)
6. **WaitForSingleObject –** для ожидания завершения процесса (аналог *wait* в Unix)

Общий метод и алгоритм решения

Для реализации поставленной задачи необходимо:

1) Изучить принципы работы функций Windows API (CreatePipe, CreateProcess, WriteFile, ReadFile, CloseHandle, WaitForSingleObject)

2) Написать программу, которая будет работать с 3-мя процессами: один родительский и два дочерних, процессы связываются между собой при помощи pipe-ов.

3) Организовать работу по отбору используемого дочернего процесса для записи строк в файл, а также по непосредственно самой записи строк в файл.

Исходный код

parent.cpp

```
#include <windows.h>
#include <iostream>
#include <string>
#include <cstdlib>
#include <ctime>

int main() {
    HANDLE hWritePipe1, hReadPipe1, hWritePipe2, hReadPipe2;
    SECURITY_ATTRIBUTES sa = {sizeof(SECURITY_ATTRIBUTES), NULL, TRUE};


    if (!CreatePipe(&hReadPipe1, &hWritePipe1, &sa, 0) || !CreatePipe(&hReadPipe2,
&hWritePipe2, &sa, 0)) {
        std::cerr << "Failed to create pipes.\n";
        return 1;
    }

    std::string fileName1, fileName2;
    std::cout << "Enter the name of the file for child process 1: ";
    std::getline(std::cin, fileName1);
    std::cout << "Enter the name of the file for child process 2: ";
    std::getline(std::cin, fileName2);

    STARTUPINFO si1 = {sizeof(STARTUPINFO)};
    STARTUPINFO si2 = {sizeof(STARTUPINFO)};
    PROCESS_INFORMATION pi1, pi2;
    ZeroMemory(&pi1, sizeof(PROCESS_INFORMATION));
    ZeroMemory(&pi2, sizeof(PROCESS_INFORMATION));


    si1.hStdInput = hReadPipe1;
    si1.hStdOutput = GetStdHandle(STD_OUTPUT_HANDLE);
    si1.hStdError = GetStdHandle(STD_ERROR_HANDLE);
    si1.dwFlags |= STARTF_USESTDHANDLES;

    si2.hStdInput = hReadPipe2;
    si2.hStdOutput = GetStdHandle(STD_OUTPUT_HANDLE);
    si2.hStdError = GetStdHandle(STD_ERROR_HANDLE);
    si2.dwFlags |= STARTF_USESTDHANDLES;


    char cmdLine1[256], cmdLine2[256];
    sprintf_s(cmdLine1, sizeof(cmdLine1), "child1.exe \"%s\"", fileName1.c_str());
    sprintf_s(cmdLine2, sizeof(cmdLine2), "child2.exe \"%s\"", fileName2.c_str());
```

```
    if (!CreateProcess(NULL, cmdLine1, NULL, NULL, TRUE, 0, NULL, NULL, &si1,
&pi1)) {
        std::cerr << "Failed to create child process 1.\n";
        return 1;
    }

    if (!CreateProcess(NULL, cmdLine2, NULL, NULL, TRUE, 0, NULL, NULL, &si2,
&pi2)) {
        std::cerr << "Failed to create child process 2.\n";
        return 1;
    }

    CloseHandle(hReadPipe1);
    CloseHandle(hReadPipe2);

    std::srand(static_cast<unsigned int>(std::time(0)));
    std::string inputLine;

    while (std::getline(std::cin, inputLine)) {
        DWORD written;
        if (std::rand() % 100 < 80) {
            WriteFile(hWritePipe1, inputLine.c_str(),
static_cast<DWORD>(inputLine.length()), &written, NULL);
            WriteFile(hWritePipe1, "\n", 1, &written, NULL);
        } else {
            WriteFile(hWritePipe2, inputLine.c_str(),
static_cast<DWORD>(inputLine.length()), &written, NULL);
            WriteFile(hWritePipe2, "\n", 1, &written, NULL);
        }
    }

    CloseHandle(hWritePipe1);
    CloseHandle(hWritePipe2);

    WaitForSingleObject(pi1.hProcess, INFINITE);
    WaitForSingleObject(pi2.hProcess, INFINITE);

    CloseHandle(pi1.hProcess);
    CloseHandle(pi1.hThread);
    CloseHandle(pi2.hProcess);
    CloseHandle(pi2.hThread);
    return 0;
}
```

child1.cpp/child2.cpp

```cpp
#include <iostream>
#include <fstream>
#include <string>

std::string removeVowels(const std::string &input) {
    const std::string vowels = "aeiouAEIOU";
    std::string result;
    for (char ch : input) {
        if (vowels.find(ch) == std::string::npos) {
            result += ch;
        }
    }
    return result;
}

int main(int argc, char *argv[]) {
    if (argc < 2) {
        std::cerr << "File name argument is missing." << std::endl;
        return 1;
    }

    std::ofstream outFile(argv[1]);
    if (!outFile.is_open()) {
        std::cerr << "Failed to open file for writing." << std::endl;
        return 1;
    }

    std::string inputLine;
    while (std::getline(std::cin, inputLine)) {
        std::string output = removeVowels(inputLine);
        outFile << output << std::endl;
    }

    outFile.close();
    return 0;
}
```

Демонстрация работы программы

```
PS C:\Users\Xiaomi\Desktop\ll1\build> ./parent

Enter the name of the file for child process 1:
C:\Users\Xiaomi\Desktop\ll1\first.txt

Enter the name of the file for child process 2:
C:\Users\Xiaomi\Desktop\ll1\second.txt

Monkey D. Luffy
nami usop
SANJI Chopper
Nico Robin
FRRRRAAAAAANKY!
Brook

PS C:\Users\Xiaomi\Desktop\ll1\build> cd ../

PS C:\Users\Xiaomi\Desktop\ll1> type first.txt
Mnky D. Lffy
zr
nm sp
SNJ Chppr
Nc Rbn
Brk

PS C:\Users\Xiaomi\Desktop\ll1> type second.txt
FRRRRNKY!
```

Выводы

Выполнив данную лабораторную работу, я понял принцип работы с процессами в ОС и обеспечения обмена данными между процессами посредствам каналов. Я познакомился с понятием pipe, fork, узнал о существовании и принципах работы таких функций Windows API, обеспечивающих работу с процессами, как: CreatePipe, CreateProcess, WriteFile, ReadFile, CloseHandle и WaitForSingleObject. Кроме того, благодаря заданию моего варианта я узнал, как можно написать генератор случайных чисел на языке C++. Данная работа не только развила мои навыки работы с процессами ОС, но и улучшила моё понимание языка C++.