МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПЕУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра информационных систем и технологий

**ЛАБОРАТОРНАЯ РАБОТА** **№8.**

**РАБОТА С ПРОЦЕССАМИ СИСТЕМЫ**

Выполнили: Карпович П.И., Ральцевич Э.В., Коляго Л.В. ФИТ, 3 курс, 5 группа(ПОИТ)

Проверила: Герман Ю.О.

2017

**ЗАДАНИЕ**.

1. Набрать и выполнить приведенные файлы.

ПОИСК ФАЙЛА (-ОВ) ПО ИМЕНИ. В ЭТОЙ ПРОГРАММЕ ИЩЕМ ФАЙЛ, ИМЯ КОТОРОГО НАДО ИЗМЕНИТЬ.

#include "stdafx.h"

#include "windows.h"

#include <tlhelp32.h>

#include "string.h"

#include <iostream>

int \_tmain(int argc, \_TCHAR\* argv[])

{

WIN32\_FIND\_DATA FindFileData;

HANDLE hFind = INVALID\_HANDLE\_VALUE;

TCHAR directorySearch[] = TEXT("f:\\\*");

TCHAR filesearch[] = TEXT("f:\\fileF\*.txt");

// Find the first file in the directory.

hFind = FindFirstFile(filesearch, &FindFileData);

if (hFind == INVALID\_HANDLE\_VALUE)

{

printf("Invalid file handle. Error is %u.\n", GetLastError());

}

else

{

std::wcout << "Found: " << FindFileData.cFileName << '\n';

// List all the other files in the directory.

while (FindNextFile(hFind, &FindFileData) != 0)

{

std::wcout << "Found: " << FindFileData.cFileName << '\n';

}

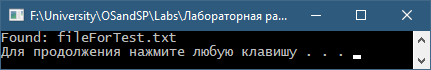
FindClose(hFind);

}

system("pause");

return 0;

}



ЗАПУСК ВНЕШНЕГО ПРОЦЕССА WIN API CreateProcess. Следующая небольшая программа запускает внешнее приложение через CREATEPROCESS (ЗАПУСКАЕМ mspaint.exe).

#include "stdafx.h"

#include <windows.h>

#include <iostream>

#include "string.h"

int \_tmain(int argc, \_TCHAR\* argv[])

{

STARTUPINFO cif;

ZeroMemory(&cif, sizeof(STARTUPINFO));

PROCESS\_INFORMATION pi;

TCHAR buf[] = TEXT("mspaint.exe");

if (!(CreateProcess(NULL, buf, NULL, NULL, FALSE, 0,

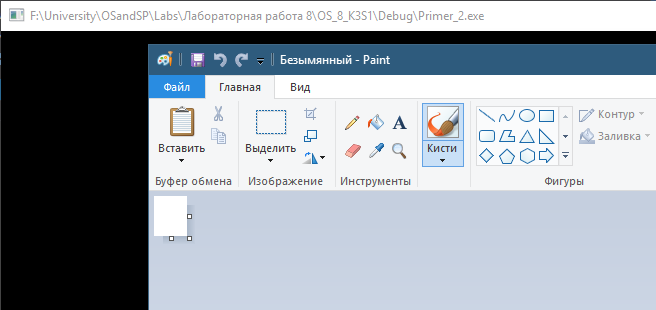
NULL, NULL, &cif, &pi)))

std::wcout << "Error " << '\n';

getchar();

return 0;

}



**ЗАДАЧА.** ОБЪЕДИНИТЬ 1А и 1Б. НАЙТИ ФАЙЛ EXE И ЗАПУСТИТЬ ЕГО НА ВЫПОЛНЕНИЕ.

#include "stdafx.h"

#include "windows.h"

#include <tlhelp32.h>

#include "string.h"

#include <iostream>

int \_tmain(int argc, \_TCHAR\* argv[])

{

WIN32\_FIND\_DATA FindFileData;

HANDLE hFind = INVALID\_HANDLE\_VALUE;

TCHAR temp[] = TEXT("C:\\Program Files (x86)\\Opera\\");

TCHAR directorySearch[] = TEXT("C:\\Program Files (x86)\\Opera\\\*");

TCHAR filesearch[] = TEXT("C:\\Program Files (x86)\\Opera\\lau\*.exe");

hFind = FindFirstFile(filesearch, &FindFileData);

if (hFind == INVALID\_HANDLE\_VALUE)

{

printf("Invalid file handle. Error is %u.\n", GetLastError());

}

else

{

WCHAR buf[260] = L"";

wcscat(buf, temp);

wcscat(buf, FindFileData.cFileName);

std::wcout << "Found: " << buf << '\n';

STARTUPINFO cif;

ZeroMemory(&cif, sizeof(STARTUPINFO));

PROCESS\_INFORMATION pi;

if (!(CreateProcess(NULL, buf, NULL, NULL, FALSE, 0, NULL, NULL, &cif, &pi)))

std::wcout << "Error " << '\n';

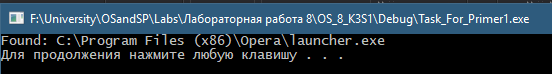
FindClose(hFind);

}

system("pause");

return 0;

}



Запуск процесса с выводом на консоль его системного номера (заголовка)

#include "stdafx.h"

#include <windows.h>

#include <iostream>

#include "string.h"

int \_tmain(int argc, \_TCHAR\* argv[])

{

DWORD processID;

STARTUPINFO cif;

ZeroMemory(&cif, sizeof(STARTUPINFO));

PROCESS\_INFORMATION pi;

TCHAR buf[] = TEXT("mspaint.exe");

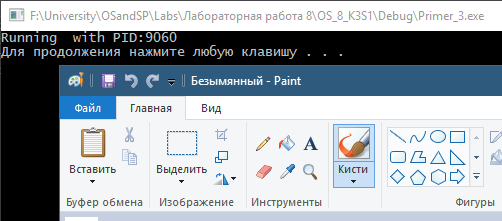
processID = CreateProcess(NULL, buf, NULL, NULL, FALSE, 0, NULL, NULL, &cif, &pi);

std::wcout << "Running with PID:" << pi.dwProcessId << '\n';

system("pause");

return 0;

}



Запуск процесса из консоли с удалением консольного окна через 0.2 секунды.

#include "stdafx.h"

#include <windows.h>

#include <iostream>

#include "string.h"

int \_tmain(int argc, \_TCHAR\* argv[])

{

DWORD processID;

STARTUPINFO cif;

ZeroMemory(&cif, sizeof(STARTUPINFO));

PROCESS\_INFORMATION pi;

TCHAR buf[] = TEXT("mspaint.exe");

if (!(processID = CreateProcess(NULL, buf, NULL, NULL, FALSE, 0, NULL, NULL, &cif, &pi)))

std::wcout << "Running " << '\n';

HANDLE hProcess = GetCurrentProcess();

if (NULL != hProcess)

{

Sleep(200);

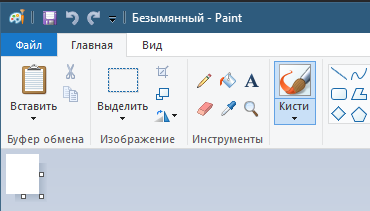
BOOL result = TerminateProcess(hProcess, 0);

CloseHandle(hProcess);

}

return 0;

}



ВЫВОД ИМЕН ВСЕХ ЗАПУЩЕННЫХ ПРОЦЕССОВ

#include "stdafx.h"

#include "windows.h"

#include <tlhelp32.h>

#include "string.h"

#include <iostream>

int \_tmain(int argc, \_TCHAR\* argv[])

{

HANDLE hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hStdout, FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY | BACKGROUND\_BLUE);

HANDLE hSnapshot = CreateToolhelp32Snapshot(TH32CS\_SNAPPROCESS, 0);

if (hSnapshot)

{

PROCESSENTRY32 pe32;

pe32.dwSize = sizeof(PROCESSENTRY32);

if (Process32First(hSnapshot, &pe32))

{

do

{

std::wcout << "Running ProcName :" << pe32.szExeFile << '\n';

} while (Process32Next(hSnapshot, &pe32));

}

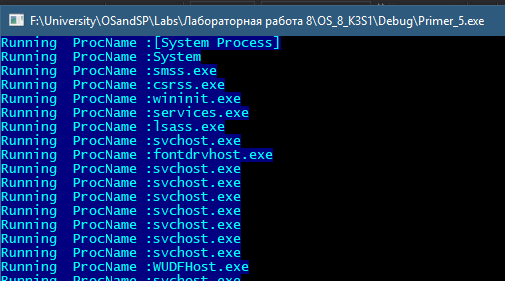
CloseHandle(hSnapshot);

}

system("pause");

return 0;

}



ПРОВЕРЯЕТ, ВЫПОЛНЯЕТСЯ ЛИ ПРОЦЕСС

#include "stdafx.h"

#include "windows.h"

#include <tlhelp32.h>

#include "string.h"

#include <iostream>

#define STRLEN(x) (sizeof(x)/sizeof(TCHAR) - 1)

bool AreEqual(const TCHAR \*a, const TCHAR \*b)

{

while (\*a == \*b)

{

if (\*a == \_TEXT('\0'))return true;

a++; b++;

}

return false;

}

bool IsProcessRun()

{

bool RUN;

TCHAR buf[] = TEXT("Primer\_6.exe");

HANDLE hSnapshot = CreateToolhelp32Snapshot(TH32CS\_SNAPPROCESS, 0);

PROCESSENTRY32 pe;

pe.dwSize = sizeof(PROCESSENTRY32);

Process32First(hSnapshot, &pe);

if (AreEqual(pe.szExeFile, buf))

{

RUN = true;

return RUN;

}

else

RUN = false;

while (Process32Next(hSnapshot, &pe))

{

if (AreEqual(pe.szExeFile, buf))

{

RUN = true;

return RUN;

}

else

RUN = false;

}

return RUN;

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

HANDLE hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hStdout, FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY | BACKGROUND\_BLUE);

if (IsProcessRun())

{

std::wcout << "Running" << '\n';

}

else

{

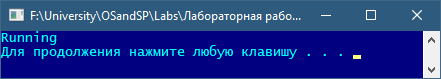
std::wcout << "NOT Running" << '\n';

}

system("pause");

return 0;

}



1. На основании представленных программок создать собственную программу, которая отыскивает *exe*-файл по части имени, запускает его на выполнение, затем проверяет, выполняется ли он, и выдает соответствующее сообщение.

#include "stdafx.h"

#include "windows.h"

#include <tlhelp32.h>

#include "string.h"

#include <iostream>

#define STRLEN(x) (sizeof(x)/sizeof(TCHAR) - 1)

bool AreEqual(const TCHAR \*a, const TCHAR \*b)

{

while (\*a == \*b)

{

if (\*a == \_TEXT('\0'))return true;

a++; b++;

}

return false;

}

bool IsProcessRun()

{

bool RUN;

TCHAR buf[] = TEXT("opera.exe");

HANDLE hSnapshot = CreateToolhelp32Snapshot(TH32CS\_SNAPPROCESS, 0);

PROCESSENTRY32 pe;

pe.dwSize = sizeof(PROCESSENTRY32);

Process32First(hSnapshot, &pe);

if (AreEqual(pe.szExeFile, buf))

{

RUN = true;

return RUN;

}

else

RUN = false;

while (Process32Next(hSnapshot, &pe))

{

if (AreEqual(pe.szExeFile, buf))

{

RUN = true;

return RUN;

}

else

RUN = false;

}

return RUN;

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

WIN32\_FIND\_DATA FindFileData;

HANDLE hFind = INVALID\_HANDLE\_VALUE;

TCHAR temp[] = TEXT("C:\\Program Files (x86)\\Opera\\");

TCHAR directorySearch[] = TEXT("C:\\Program Files (x86)\\Opera\\\*");

TCHAR filesearch[] = TEXT("C:\\Program Files (x86)\\Opera\\lau\*.exe");

hFind = FindFirstFile(filesearch, &FindFileData);

if (hFind == INVALID\_HANDLE\_VALUE)

{

printf("Invalid file handle. Error is %u.\n", GetLastError());

}

else

{

WCHAR buf[260] = L"";

wcscat(buf, temp);

wcscat(buf, FindFileData.cFileName);

std::wcout << "Found: " << buf << '\n';

STARTUPINFO cif;

ZeroMemory(&cif, sizeof(STARTUPINFO));

PROCESS\_INFORMATION pi;

if (!(CreateProcess(NULL, buf, NULL, NULL, FALSE, 0, NULL, NULL, &cif, &pi)))

std::wcout << "Error " << '\n';

FindClose(hFind);

HANDLE hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hStdout, FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY | BACKGROUND\_BLUE);

if (IsProcessRun())

{

std::wcout << "Running" << '\n';

}

else

{

std::wcout << "NOT Running" << '\n';

}

}

system("pause");

return 0;

}

