Комп'ютерний практикум №3
ПРОЦЕСИ ОС WINDOWS

Виконав:
Студент 2 курсу ФТІ
групи ФІ-92
Поночевний Назар Юрійович

Мета: ознайомлення з основами створення і управління процесами в ОС WINDOWS.

Завдання 10:

1. Створити системи каталогів типу FILE11/FILE12/FILE13/ та FILE21/FILE22/FILE23/ (Діє головний процес);
2. Скопіювати з якогось існуючого каталогу групу файлів (як бінарних так і текстових) у вказаній директорії після чого створити власні у обраних каталогах в яких є ключове слово (Діє перший та другий дочірні процеси);
3. Передача параметру Date через змінні середовища для одного з дочірніх процесів, який не успадковує (Діє головний процес);
4. Змінити змінну середовища Path для одного з дочірніх процесів (Діє головний процес);
5. Процеси шукають файли, де знаходять заданий зразок, доки не закінчать всі. Перший дочірній процес повідомляє другому зразок. Головний процес виводить список файлів зі зразком;
6. Дочірні процеси передають імена файлів, де знайдено зразок. Передача даних між дочірніми процесами;
7. 2 - Г1К 12К 2ГК результат у Вікно;
8. Відобразити системну інформацію. Про процеси, змінні оточення, а також виведення початкових даних та результатів змін на екран (Діє головний процес).

Код (Main Process):

```
#include <tchar.h>
#include <windows.h>
#include <stdio.h>
#include <strsafe.h>

#define BUFSIZE 4096

HANDLE child1InRead = NULL;
HANDLE W11 = NULL;
HANDLE R11 = NULL;
```

```
HANDLE child2InRead = NULL;
HANDLE W21 = NULL;
HANDLE R21 = NULL;

void CreateChild1Process() {

    TCHAR chNewEnv[BUFSIZE];
    LPTSTR lpszCurrentVariable;
    DWORD dwFlags = 0;
    TCHAR applicationName[] = TEXT("\"D:\\Microsoft Visual
Studio\\Workspace\\SysProga\\Lab3_1\\Debug\\Lab3_1.exe\" \"D:\\Microsoft
Visual Studio\\Workspace\\SysProga\\files\" \"D:\\Microsoft Visual
Studio\\Workspace\\SysProga\\Lab3\\Debug\\FILE21\\FILE22\\FILE23\"");
    PROCESS_INFORMATION pi;
    STARTUPINFO si;
    BOOL success = FALSE;

    // Change env vars and not inherit it from parent process

    lpszCurrentVariable = (LPTSTR)chNewEnv;
    StringCchCopy(lpszCurrentVariable, BUFSIZE, TEXT("Date=05-2021"));

    lpszCurrentVariable += lstrlen(lpszCurrentVariable) + 1;
    *lpszCurrentVariable = (TCHAR)0;

    ZeroMemory(&pi, sizeof(PROCESS_INFORMATION));
    ZeroMemory(&si, sizeof(STARTUPINFO));

    si.cb = sizeof(STARTUPINFO);
    si.hStdError = W11;
    si.hStdOutput = W11;
    si.dwFlags |= STARTF_USESTDHANDLES;

#ifdef UNICODE
    dwFlags = CREATE_UNICODE_ENVIRONMENT | CREATE_NEW_CONSOLE;
#endif

    // Start process of files coping

    success = CreateProcess(NULL, applicationName, NULL, NULL, TRUE,
dwFlags, (LPVOID)chNewEnv, NULL, &si, &pi);

    if (!success) {
        printf("Error creating child process \n");
        return;
    }
```

```
    WaitForSingleObject(pi.hProcess, INFINITE);
    CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);

}

void CreateChild2Process() {

    TCHAR applicationName[] = TEXT("\"D:\\Microsoft Visual
Studio\\Workspace\\SysProga\\Lab3_2\\Debug\\Lab3_2.exe\" \"D:\\Microsoft
Visual Studio\\Workspace\\SysProga\\files\"");
    PROCESS_INFORMATION pi;
    STARTUPINFO si;
    BOOL success = FALSE;

    // Set env var with inheritence from parent process

    if (!SetEnvironmentVariable(TEXT("Path"), TEXT("Test")))
    {
        printf("SetEnvironmentVariable failed (%d)\n", GetLastError());
        return;
    }

    ZeroMemory(&pi, sizeof(PROCESS_INFORMATION));
    ZeroMemory(&si, sizeof(STARTUPINFO));

    si.cb = sizeof(STARTUPINFO);
    si.hStdError = W21;
    si.hStdOutput = W21;
    si.dwFlags |= STARTF_USESTDHANDLES;

    // Create process of looking for files

    success = CreateProcess(NULL, applicationName, NULL, NULL, TRUE,
CREATE_NEW_CONSOLE, NULL, NULL, &si, &pi);

    if (!success) {
        printf("Error creating child process \n");
        return;
    }

    WaitForSingleObject(pi.hProcess, INFINITE);
    CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);
```

```
}

int main()
{
    DWORD MainProcessId = GetCurrentProcessId();
    printf("(%d) Parent process running.... \n", MainProcessId);

    // Create new directories

    if (!CreateDirectory(TEXT("FILE11"), NULL))
    {
        printf("CreateDirectory failed (%d)\n", GetLastError());
        return 1;
    }
    if (!CreateDirectory(TEXT("FILE11\\FILE12"), NULL))
    {
        printf("CreateDirectory failed (%d)\n", GetLastError());
        return 1;
    }
    if (!CreateDirectory(TEXT("FILE11\\FILE12\\FILE13"), NULL))
    {
        printf("CreateDirectory failed (%d)\n", GetLastError());
        return 1;
    }

    if (!CreateDirectory(TEXT("FILE21"), NULL))
    {
        printf("CreateDirectory failed (%d)\n", GetLastError());
        return 1;
    }
    if (!CreateDirectory(TEXT("FILE21\\FILE22"), NULL))
    {
        printf("CreateDirectory failed (%d)\n", GetLastError());
        return 1;
    }
    if (!CreateDirectory(TEXT("FILE21\\FILE22\\FILE23"), NULL))
    {
        printf("CreateDirectory failed (%d)\n", GetLastError());
        return 1;
    }
    printf("\n(%d) Directories created\n\n\n", MainProcessId);

    DWORD dRead, dWritten;
    CHAR chBuf[BUFSIZE] = "hello";
    BOOL bSuccess = FALSE;
```

```
    SECURITY_ATTRIBUTES secAttr;
    secAttr.nLength = sizeof(SECURITY_ATTRIBUTES);
    secAttr.bInheritHandle = TRUE;
    secAttr.lpSecurityDescriptor = NULL;

    if (!CreatePipe(&R11, &W11, &secAttr, 0)) {
        printf("\nerror creating first pipe \n");
    }

    if (!SetHandleInformation(R11, HANDLE_FLAG_INHERIT, 0)) {
        printf("\nR1 SetHandleInformation \n");
    }

    if (!CreatePipe(&R21, &W21, &secAttr, 0)) {
        printf("\nerror creating second pipe \n");
    }

    if (!SetHandleInformation(R21, HANDLE_FLAG_INHERIT, 0)) {
        printf("\nR2 SetHandleInformation \n");
    }

    HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
    HANDLE hStdIn = GetStdHandle(STD_INPUT_HANDLE);

    CreateChild1Process();
    CreateChild2Process();

    bSuccess = ReadFile(R11, chBuf, BUFSIZE, &dRead, NULL);
    if (!bSuccess) {
        printf("error reading \n");
        return 0;
    }

    bSuccess = WriteFile(hStdOut, chBuf, 619, &dWritten, NULL);
    if (!bSuccess) {
        printf("error reading \n");
        return 0;
    }

    bSuccess = ReadFile(R21, chBuf, BUFSIZE, &dRead, NULL);
    if (!bSuccess) {
        printf("error reading \n");
        return 0;
    }

    bSuccess = WriteFile(hStdOut, chBuf, 2100, &dWritten, NULL);
```

```
    if (!bSuccess) {
        printf("error reading \n");
        return 0;
    }

    return 0;

}
```

Код (Child 1 Process):

```
#include <tchar.h>
#include <windows.h>
#include <stdio.h>
#include <strsafe.h>

int _tmain(int argc, TCHAR* argv[])
{

    TCHAR           szSearchDir[MAX_PATH];
    WIN32_FIND_DATA FileData;
    HANDLE          hSearch;
    BOOL            fFinished;
    TCHAR           szPath[MAX_PATH];
    TCHAR           szNewPath[MAX_PATH];
    TCHAR           sTargetFileDirectory[MAX_PATH];

    LPTSTR lpszVariable;
    LPTCH lpvEnv;

    lpvEnv = GetEnvironmentStrings();

    if (lpvEnv == NULL)
    {
        printf("GetEnvironmentStrings failed (%d)\n", GetLastError());
        return 1;
    }

    if (argc != 3)
    {
        _tprintf(TEXT("Usage: %s <search dir> <copy dir>\n"), argv[0]);
        return 1;
    }
```

```
    StringCchCopy(sTargetFileDirectory, MAX_PATH, argv[2]);

    DWORD Child1ProcessId = GetCurrentProcessId();
    printf("(%d) Child1 process running.... \n", Child1ProcessId);

    // Copy "*.txt" and "*.exe" files to FILE23 folder

    StringCchCopy(szSearchDir, MAX_PATH, argv[1]);
    StringCchCat(szSearchDir, MAX_PATH, TEXT("\\*.txt"));

    hSearch = FindFirstFile(szSearchDir, &FileData);
    if (hSearch == INVALID_HANDLE_VALUE)
    {
        printf("No *.txt files found.\n");
        return 1;
    }

    fFinished = FALSE;
    while (!fFinished)
    {
        StringCchPrintf(szPath, sizeof(szPath) / sizeof(szPath[0]),
TEXT("%s\\%s"), argv[1], FileData.cFileName);
        StringCchPrintf(szNewPath, sizeof(szNewPath) /
sizeof(szNewPath[0]), TEXT("%s\\%s"), sTargetFileDirectory,
FileData.cFileName);

        if (!CopyFile(szPath, szNewPath, FALSE))
        {
            _tprintf(TEXT("Could not copy file %s to %s\n"), szPath,
szNewPath);
            return 1;
        }

        if (!FindNextFile(hSearch, &FileData))
        {
            if (GetLastError() == ERROR_NO_MORE_FILES)
            {
                _tprintf(TEXT("\n(%d) Copied %s to %s\n"),
Child1ProcessId, szSearchDir, sTargetFileDirectory);
                fFinished = TRUE;
            }
            else
            {
                printf("Could not find next file.\n");
                return 1;
            }
```

```
        }
    }

    FindClose(hSearch);

    StringCchCopy(szSearchDir, MAX_PATH, argv[1]);
    StringCchCat(szSearchDir, MAX_PATH, TEXT("\\*.exe"));

    hSearch = FindFirstFile(szSearchDir, &FileData);
    if (hSearch == INVALID_HANDLE_VALUE)
    {
        printf("No *.exe files found.\n");
        return 1;
    }

    fFinished = FALSE;
    while (!fFinished)
    {
        StringCchPrintf(szPath, sizeof(szPath) / sizeof(szPath[0]),
TEXT("%s\\%s"), argv[1], FileData.cFileName);
        StringCchPrintf(szNewPath, sizeof(szNewPath) /
sizeof(szNewPath[0]), TEXT("%s\\%s"), sTargetFileDirectory,
FileData.cFileName);

        if (!CopyFile(szPath, szNewPath, FALSE))
        {
            _tprintf(TEXT("Could not copy file %s to %s\n"), szPath,
szNewPath);
            return 1;
        }

        if (!FindNextFile(hSearch, &FileData))
        {
            if (GetLastError() == ERROR_NO_MORE_FILES)
            {
                _tprintf(TEXT("(%d) Copied %s to %s\n"),
Child1ProcessId, szSearchDir, sTargetFileDirectory);
                fFinished = TRUE;
            }
            else
            {
                printf("Could not find next file.\n");
                return 1;
            }
        }
    }
```

```
    FindClose(hSearch);

    lpszVariable = (LPTSTR)lpvEnv;

    _tprintf(TEXT("\n(%d) EnvVars:\n"), Child1ProcessId);
    while (*lpszVariable)
    {
        _tprintf(TEXT("%s\n"), lpszVariable);
        lpszVariable += lstrlen(lpszVariable) + 1;
    }
    FreeEnvironmentStrings(lpvEnv);

    return 0;

}
```

Код (Child 2 Process):

```
#include <tchar.h>
#include <windows.h>
#include <stdio.h>
#include <strsafe.h>

int _tmain(int argc, TCHAR* argv[])
{

    TCHAR           szSearchDir[MAX_PATH];
    WIN32_FIND_DATA FileData;
    HANDLE          hSearch;
    BOOL            fFinished;
    TCHAR           szPath[MAX_PATH];

    LPTSTR lpszVariable;
    LPTCH lpvEnv;

    lpvEnv = GetEnvironmentStrings();

    if (lpvEnv == NULL)
    {
        printf("GetEnvironmentStrings failed (%d)\n", GetLastError());
        return 1;
    }
```

```
    if (argc != 2)
    {
        _tprintf(TEXT("Usage: %s <search dir>\n"), argv[0]);
        return 1;
    }

    DWORD Child2ProcessId = GetCurrentProcessId();
    printf("(%d) Child2 process running.... \n\n", Child2ProcessId);

    // Search "*.txt" files in directory

    StringCchCopy(szSearchDir, MAX_PATH, argv[1]);
    StringCchCat(szSearchDir, MAX_PATH, TEXT("\\*.txt"));

    hSearch = FindFirstFile(szSearchDir, &FileData);
    if (hSearch == INVALID_HANDLE_VALUE)
    {
        printf("No *.txt files found.\n");
        return 1;
    }

    fFinished = FALSE;
    while (!fFinished)
    {
        StringCchPrintf(szPath, sizeof(szPath) / sizeof(szPath[0]),
TEXT("%s\\%s"), argv[1], FileData.cFileName);
        _tprintf(TEXT("(%d) Found: %s\n"), Child2ProcessId, szPath);

        if (!FindNextFile(hSearch, &FileData))
        {
            if (GetLastError() == ERROR_NO_MORE_FILES)
                fFinished = TRUE;
            else
            {
                printf("Could not find next file.\n");
                return 1;
            }
        }
    }

    FindClose(hSearch);

    lpszVariable = (LPTSTR)lpvEnv;

    _tprintf(TEXT("\n(%d) EnvVars:\n"), Child2ProcessId);
    while (*lpszVariable)
```

```
    {
        _tprintf(TEXT("%s\n"), lpszVariable);
        lpszVariable += lstrlen(lpszVariable) + 1;
    }
    FreeEnvironmentStrings(lpvEnv);

    return 0;

}
```

Скріншоти:

```
D:\Microsoft Visual Studio\Workspace\SysProga\Lab3\Debug>Lab3.exe
(14120) Parent process running....

(14120) Directories created


(13440) Child1 process running....

(13440) Copied D:\Microsoft Visual Studio\Workspace\SysProga\files\*.txt to D:\Microsoft Visual Studio\Workspace\SysPr
oga\Lab3\Debug\FILE21\FILE22\FILE23
(13440) Copied D:\Microsoft Visual Studio\Workspace\SysProga\files\*.exe to D:\Microsoft Visual Studio\Workspace\SysPr
oga\Lab3\Debug\FILE21\FILE22\FILE23

(13440) EnvVars:
Date=05-2021


(13400) Child2 process running....

(13400) Found: D:\Microsoft Visual Studio\Workspace\SysProga\files\test.txt
(13400) Found: D:\Microsoft Visual Studio\Workspace\SysProga\files\text.txt

(13400) EnvVars:
=::=::\
=C:=C:\Users\NazarPonochevnyi
=D:=D:\Microsoft Visual Studio\Workspace\SysProga\Lab3\Debug
=ExitCode=00000000
ALLUSERSPROFILE=C:\ProgramData
ANDROID_HOME=D:\AndroidSdk
```

```
OneDriveConsumer=C:\Users\NazarPonochevnyi\OneDrive
OS=Windows_NT
Path=Test
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.PY;.PYW
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_ARCHITEW6432=AMD64
PROCESSOR_IDENTIFIER=Intel64 Family 6 Model 58 Stepping 9, GenuineIntel
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=3a09
ProgramData=C:\ProgramData
ProgramFiles=C:\Program Files (x86)
ProgramFiles(x86)=C:\Program Files (x86)
ProgramW6432=C:\Program Files
PROMPT=$P$G
PSModulePath=C:\Users\NazarPonochevnyi\Documents\WindowsPowerShell\Modules;D:\Google\Cloud SDK\google-cloud-sdk\platfo
rm\PowerShell
PUBLIC=C:\Users\Public
SESSIONNAME=Console
SystemDrive=C:
SystemRoot=C:\WINDOWS
TEMP=C:\Users\NAZARP~1\AppData\Local\Temp
TMP=C:\Users\NAZARP~1\AppData\Local\Temp
USERDOMAIN=DESKTOP
USERDOMAIN_ROAMINGPROFILE=DESKTOP
USERNAME=NazarPonochevnyi
USERPROFILE=C:\Users\NazarPonochevnyi
VBOX_MSI_INSTALL_PATH=D:\Oracle\VirtualBox\
windir=C:\WINDOWS

D:\Microsoft Visual Studio\Workspace\SysProga\Lab3\Debug>
```

Повний код можна знайти у GitHub-репозиторії:
https://github.com/NazarPonochevnyi/Programming-Labs/blob/master/System%20Programming/Lab3/lab3.cpp