Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и

системное программирование

ОТЧЁТ

к лабораторной работе №5

на тему

Приложение для смены иконок файлов заданного расширения с сохранением события в журнале событий WINDOWS

Выполнила: студентка группы 153502

Шайкова Вероника Андреевна

Проверил: Гриценко Никита Юрьевич

Минск 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc147885133)

[2 Теоретические сведения 4](#_Toc147885134)

[3 Результаты выполнения лабораторной работы 5](#_Toc147885135)

[Выводы](#_Toc147885136) 7

[Список использованных источников](#_Toc147885137) 8

[Приложение А (обязательное) Листинг кода](#_Toc147885138) 9

## 1 ПОСТАНОВКА ЗАДАЧИ

Целью выполнения лабораторной работы является создание оконного приложения на Win32 API, реализующее возможость изменять иконки файлов указанного расширения на пользовательские путем изменения данных в реестре с фиксацией данного события в журнале обытий Windows в случае удачной смены иконки или ошибки.

## 2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Реестр Windows представляет собой иерархическую БД, хранящую параметры и настройки ОС, аппаратного обеспечения, пользователей и имеет 5 главных разделов (HKEY\_CLASSES\_ROOT, HKEY\_CURRENT\_USER, HKEY\_LOCAL\_MACHINE, HKEY\_USERS, HKEY\_CURRENT\_CONFIG).

Для работы с реестром на WIN32 API необходимо первым делом получить дескриптор ключа HKEY (функция RegOpenKey). Создать новый ключ можно при помощи функции RegCreateKey, после чего получаем новый раздел, где можем создавать или редактировать существующие записи при помощи функций RegSetValue, RegDeleteValue. Для удаления раздела можем использовать функцию RegDeleteKey, но она не сработает для разделов, содержащих дочерние подразделы. В этом случае необходимо использовать RegDeleteTree. Для получения данных, находящиеся по определенному ключу можно использовать, например, функции RegGetValue или RegQueryInfo. По завершении работы с реестром закрываем дескриптор ключа при помощи RegCloseKey.

Журнал событий Windows ─ это выжный инструмент, который позволяет отслеживать ошибки, предупреждения, а также информационные сообщения, которые регистрируются операционной системой, ее компонентами и другими приложениями. Служба ведения журнала событий записывает события из различных источников и сохраняет их в единой коллекции.

Конечно, для работы с журналом Windows через средства Win32 API программисту необходимо получить дескриптор. Это возможно сделать при помощи функции RegisterEventSource. После этого все готово к работе, и можно создавать свои собственныезаписи в журнале, генерируемые автомаически при каждом повторе некоторого участка программы. Для этого используется функция ReportEvent, в параметрах которой можем указать сообщение события, его тип (EVENTLOG\_INFORMATION\_TYPE, EVENTLOG\_WARNING\_TYPE, EVENTLOG\_ERROR\_TYPE). Закрываем дескриптор прии помощи функции DeregisterEventSource.

## 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной работы было создано приложение, позволяющее программно изменять данные в реестре (рисунок 1), а именно изменять данные работы с различными расширениями файлов в подразделах HKEY\_CLASSES\_ROOT.

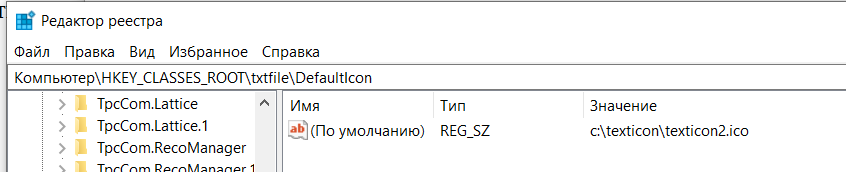


Рисунок 1 ─ изменяемая рапись в реестре

Была реализована возможность изменения иконок для файлов расширений \*.txt, \*.py, \*.torrent с выбором иконки (рисунок 2, рисунок 3).

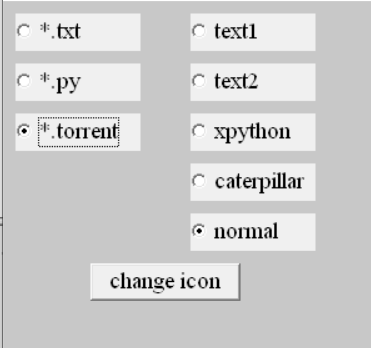


Рисунок 2 ─ интерфейс

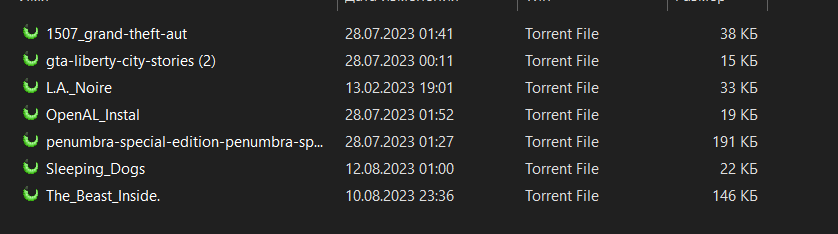


Рисунок 3 ─ измененная иконка .torrent файла

Кроме того реализована возможность работы с журналом событий Windows, а именно создание пользовательских сообщений (рисунок 4, рисунок 5):

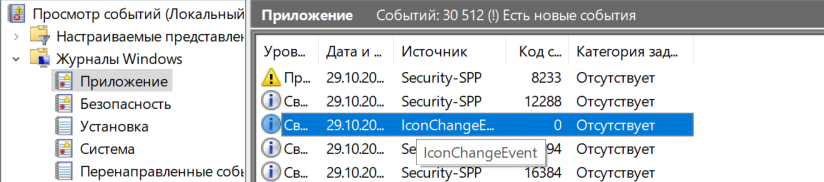


Рисунок 4 ─ информационнок сообщение о смене иконки

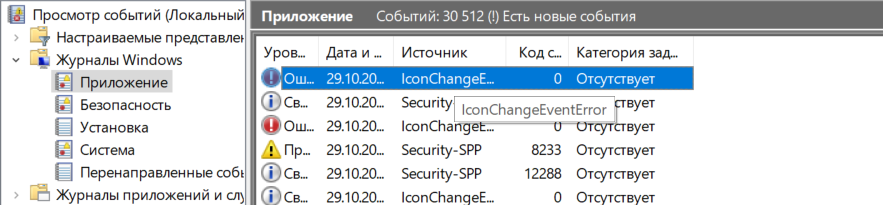


Рисунок 5 ─ сообщение об ошибке в процессе смены иконки

Новая запись в журнал добавляется при каждой попытке изменить иконку. В случае успешной смены ─ информационное, иначе (не удалось получить дескриптор ключа / не удалось установить значение по данному ключу) ─ сообщение об ошибке.

## ВЫВОДЫ

В ходе выполнения лабораторной работы было создано приложение на Win32 API, которое реализует возможность изменить иконки \*.txt, \*.py, \*.torrent файлов на 4 пользовательских либо вернуть стандартную иконку по умолчанию. Для этого реализован графический интерфейс пользователя, в котором пользователь может выбрать расширение, иконки файлов которого необходимо изменить, а также иконку (5 вариантов). Изменение происходит путем изменения соответствующих записей реестра и собровождается добавлением записи в журнал Windows. После смены иконки необходимо перезапустить устройство либо проводник через диспетчер задач, чтобы изменения вступили в силу.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Щупак Ю. Win32 API. Разработка приложений для Windows. ─ СПб: Питер, 2008. ─ 592 с.: ип.

[2] Создание классических приложений для Windows с использованием API Win32 [Электронный ресурс]. ─ Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/api ─ Дата доступа 20.09.2023

## ПРИЛОЖЕНИЕ А

## (обязательное)

## Листинг кода

**Lab4.cpp**

#include <windows.h>

#include <string>

#include "globals\_defines.h"

WNDCLASS createWindowClass(HBRUSH background\_color, HCURSOR cursor\_type, HINSTANCE hInstance, HICON Icon, LPCWSTR window\_name, WNDPROC procedure) {

// создание пустого класса

WNDCLASS new\_window = { 0 };

// наполняем пустой класс параметрами, требуемыми для конкретного окна

new\_window.hbrBackground = background\_color;

new\_window.hCursor = cursor\_type;

new\_window.hInstance = hInstance;

new\_window.hIcon = Icon;

new\_window.lpszClassName = window\_name;

new\_window.lpfnWndProc = procedure;

return new\_window;

}

void WriteToEventLogInfo(LPCWSTR message) {

event\_handle = RegisterEventSource(NULL, L"IconChangeEvent");

if (event\_handle) {

ReportEvent(event\_handle, EVENTLOG\_INFORMATION\_TYPE, 0, 0, NULL, 1, 0, &message, NULL);

DeregisterEventSource(event\_handle);

}

}

void WriteToEventLogError(LPCWSTR message) {

event\_handle = RegisterEventSource(NULL, L"IconChangeEventError");

if (event\_handle) {

ReportEvent(event\_handle, EVENTLOG\_ERROR\_TYPE, 0, 0, NULL, 1, 0, &message, NULL);

DeregisterEventSource(event\_handle);

}

}

void ChangeFileIcon(LPWSTR key\_path, LPWSTR chosen\_icon) {

txt\_key = NULL;

if (RegOpenKey(HKEY\_CLASSES\_ROOT, key\_path, &txt\_key) == ERROR\_SUCCESS) {

if (RegSetValue(txt\_key, L"DefaultIcon", REG\_SZ, chosen\_icon, 50) == 0) {

WriteToEventLogInfo(L"Extention icons changed successfully");

RegCloseKey(txt\_key);

}

else WriteToEventLogError(L"Icon Change Error - trouble changing value");

}

else WriteToEventLogError(L"Icon Change Error - trouble while opening registry key");

}

void AddRadioButtons(HWND window) {

HINSTANCE second = NULL;

// расширения

SendMessageA(

CreateWindowA("button", "\*.txt", WS\_CHILD | WS\_VISIBLE | BS\_RADIOBUTTON, 10, 10, 100, 30, window, (HMENU)TXT\_CHOSEN, someinstance, NULL),

WM\_SETFONT, WPARAM(chelovecheskiy\_shrift), TRUE);

SendMessageA(

CreateWindowA("button", "\*.py", WS\_CHILD | WS\_VISIBLE | BS\_RADIOBUTTON, 10, 50, 100, 30, window, (HMENU)PY\_CHOSEN, someinstance, NULL),

WM\_SETFONT, WPARAM(chelovecheskiy\_shrift), TRUE);

SendMessageA(

CreateWindowA("button", "\*.torrent", WS\_CHILD | WS\_VISIBLE | BS\_RADIOBUTTON, 10, 90, 100, 30, window, (HMENU)TORRENT\_CHOSEN, someinstance, NULL),

// иконки

WM\_SETFONT, WPARAM(chelovecheskiy\_shrift), TRUE);

SendMessageA(

CreateWindowA("button", "text1", WS\_CHILD | WS\_VISIBLE | BS\_RADIOBUTTON, 150, 10, 100, 30, window, (HMENU)ICON1\_CHOSEN, someinstance, NULL),

WM\_SETFONT, WPARAM(chelovecheskiy\_shrift), TRUE);

SendMessageA(

CreateWindowA("button", "text2", WS\_CHILD | WS\_VISIBLE | BS\_RADIOBUTTON, 150, 50, 100, 30, window, (HMENU)ICON2\_CHOSEN, someinstance, NULL),

WM\_SETFONT, WPARAM(chelovecheskiy\_shrift), TRUE);

SendMessageA(

CreateWindowA("button", "xpython", WS\_CHILD | WS\_VISIBLE | BS\_RADIOBUTTON, 150, 90, 100, 30, window, (HMENU)ICON\_PY\_CHOSEN, someinstance, NULL),

WM\_SETFONT, WPARAM(chelovecheskiy\_shrift), TRUE);

SendMessageA(

CreateWindowA("button", "caterpillar", WS\_CHILD | WS\_VISIBLE | BS\_RADIOBUTTON, 150, 130, 100, 30, window, (HMENU)ICON\_CATERPILLAR\_CHOSEN, someinstance, NULL),

WM\_SETFONT, WPARAM(chelovecheskiy\_shrift), TRUE);

SendMessageA(

CreateWindowA("button", "normal", WS\_CHILD | WS\_VISIBLE | BS\_RADIOBUTTON, 150, 170, 100, 30, window, (HMENU)NORMAL\_ICON\_CHOSEN, someinstance, NULL),

WM\_SETFONT, WPARAM(chelovecheskiy\_shrift), TRUE);

// кнопка

SendMessageA(

CreateWindowA("button", "change icon", WS\_CHILD | WS\_VISIBLE, 70, 210, 120, 30, window, (HMENU)LOAD\_ICON\_CLICKED, someinstance, NULL),

WM\_SETFONT, WPARAM(chelovecheskiy\_shrift), TRUE);

}

LRESULT CALLBACK createNewProcedure(HWND window, UINT message, WPARAM wp, LPARAM lp) {

switch (message) {

case WM\_CREATE:

{

//ChangeTxtIcon();

AddRadioButtons(window);

CheckRadioButton(window, 1, 1, 1);

CheckRadioButton(window, 8, 8, 8);

break;

}

case WM\_DESTROY:

{

PostQuitMessage(0);

break;

}

case WM\_COMMAND:

// Проверяем, не щелкнули ли на первой группе радиокнопок.

if (1 <= LOWORD(wp) && LOWORD(wp) <= 3)

{

// Отмечаем кпопку из первой группы.

CheckRadioButton(window, 1, 3, LOWORD(wp));

}

// Проверяем, не щелкнули ли на второй группе радиокнопок.

if (4 <= LOWORD(wp) && LOWORD(wp) <= 8)

{

// Отмечаем кпопку из второй группы.

CheckRadioButton(window, 4, 8, LOWORD(wp));

}

switch (wp)

{

case 1:

file\_type = 1;

key\_path = (LPWSTR)text\_key\_path;

CheckRadioButton(window, 4, 8, 8);

chosen\_icon = (LPWSTR)text\_standart\_path;

break;

case 2:

file\_type = 2;

key\_path = (LPWSTR)py\_key\_path;

CheckRadioButton(window, 4, 8, 8);

chosen\_icon = (LPWSTR)py\_standart\_path;

break;

case 3:

file\_type = 3;

key\_path = (LPWSTR)torrent\_key\_path;

CheckRadioButton(window, 4, 8, 8);

chosen\_icon = (LPWSTR)torrent\_standart\_path;

break;

case 4:

chosen\_icon = (LPWSTR)icon1\_path;

break;

case 5:

chosen\_icon = (LPWSTR)icon2\_path;

break;

case 6:

chosen\_icon = (LPWSTR)icon\_xpython\_path;

break;

case 7:

chosen\_icon = (LPWSTR)icon\_caterpillar\_path;

break;

case 8:

switch (file\_type)

{

case 1:

chosen\_icon = (LPWSTR)text\_standart\_path;

break;

case 2:

chosen\_icon = (LPWSTR)py\_standart\_path;

break;

case 3:

chosen\_icon = (LPWSTR)torrent\_standart\_path;

break;

default:

break;

}

break;

case 9:

ChangeFileIcon(key\_path, chosen\_icon);

break;

default:

break;

}

break;

default: return DefWindowProc(window, message, wp, lp);

}

}

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR args, int ncmdshow) {

someinstance = hInst;

chelovecheskiy\_shrift = CreateFontA(21, 8, 0, 0, FW\_EXTRALIGHT, FALSE, FALSE, FALSE, DEFAULT\_CHARSET, OUT\_OUTLINE\_PRECIS, CLIP\_DEFAULT\_PRECIS, ANTIALIASED\_QUALITY, FF\_ROMAN, "Font1");

WNDCLASS main\_window = createWindowClass(CreateSolidBrush(RGB(200, 200, 200)), LoadCursor(NULL, IDC\_CROSS), hInst, LoadIcon(NULL, IDI\_ASTERISK), L"MainWindow", createNewProcedure);

if (!RegisterClassW(&main\_window)) {

return -1;

}

MSG main\_window\_message = { 0 };

HWND mainwnd = CreateWindow(L"MainWindow", L"Icon Change", WS\_OVERLAPPEDWINDOW | WS\_VISIBLE, 200, 200, 600, 500, NULL, NULL, NULL, NULL);

while (GetMessage(&main\_window\_message, NULL, NULL, NULL)) {

TranslateMessage(&main\_window\_message);

DispatchMessage(&main\_window\_message);

}

return 0;

}

**globals\_defines.h**

HINSTANCE someinstance;

HWND button1;

HWND button2;

HWND button3;

LPCWSTR torrent\_standart\_path = L"C:\\Users\\Lenovo\\AppData\\Roaming\\uTorrent Web\\utweb.exe,0";

LPCWSTR py\_standart\_path = L"\"C:\\Windows\\py.exe\",1";

LPCWSTR text\_standart\_path = L"%SystemRoot%\\System32\\imageres.dll,-102";

LPCWSTR icon1\_path = L"c:\\texticon\\texticon.ico";

LPCWSTR icon2\_path = L"c:\\texticon\\texticon2.ico";

LPCWSTR icon\_xpython\_path = L"c:\\texticon\\xpython.ico";

LPCWSTR icon\_caterpillar\_path = L"c:\\texticon\\caterpillar.ico";

LPCWSTR text\_key\_path = L"txtfile";

LPCWSTR torrent\_key\_path = L"Torrent File";

LPCWSTR py\_key\_path = L"Python.File";

LPWSTR key\_path = (LPWSTR)text\_key\_path;

LPWSTR chosen\_icon = (LPWSTR)text\_standart\_path;

WORD file\_type = 1;

HFONT chelovecheskiy\_shrift;

HKEY txt\_key;

HANDLE event\_handle;

#define TXT\_CHOSEN 1

#define PY\_CHOSEN 2

#define TORRENT\_CHOSEN 3

#define ICON1\_CHOSEN 4

#define ICON2\_CHOSEN 5

#define ICON\_PY\_CHOSEN 6

#define ICON\_CATERPILLAR\_CHOSEN 7

#define NORMAL\_ICON\_CHOSEN 8

#define LOAD\_ICON\_CLICKED 9