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 №6

на тему

Приложение для смены иконок файлов заданного расширения с сохранением события в журнале событий WINDOWS, использующее средства синхонизации и взаимного исключения

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 6](#_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, применяя средства для организации много поточности и синхронизации работы нескольких потоков.

## 2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Синхронизация необходима для обеспечения выполнения взаимозависимого кода в правильной последовательности или организации корректного доступа нескольких потоков к общим ресурсам. Существует ряд объектов, дескрипторы которых можно использовать для синхронизации нескольких потоков. К этим объектам относятся входные буферы консоли, собычия (Events), мьютексы (Mutexes), семафоры (Semaphores), критические секции (Critical Sections).

Состояние каждого из этих объектов либо сигнализируется, либо не сообщается. При указании дескриптора для любого из этих объектов в вызове одной из [функций ожидания](https://learn.microsoft.com/ru-ru/windows/win32/sync/wait-functions) выполнение вызывающего потока блокируется до тех пор, пока состояние указанного объекта не станет сигналом. [1]

Объекты-взаимоисключения (мьютексы, mutex - от MUTual EXclusion) позволяют координировать взаимное исключение доступа к разделяемому ресурсу. Сигнальное состояние объекта соответствует моменту времени, когда объект не принадлежит ни одному потоку и его можно "захватить". И наоборот, не сигнальное состояние соответствует моменту, когда какой-либо поток уже владеет этим объектом. Доступ к объекту разрешается, когда поток, владеющий объектом, освободит его.

Объекты-события используются для уведомления ожидающих потоков о наступлении какого-либо события. Различают два вида событий - с ручным и автоматическим сбросом. Ручной сброс осуществляется функцией ResetEvent. События с ручным сбросом используются для уведомления сразу нескольких потоков. При использовании события с автосбросом уведомление получит и продолжит свое выполнение только один ожидающий поток, остальные будут ожидать дальше.

Функция CreateEvent создает объект-событие, SetEvent - устанавливает событие в сигнальное состояние, ResetEvent-сбрасывает событие. Функция PulseEvent устанавливает событие, а после возобновления ожидающих это событие потоков (всех при ручном сбросе и только одного при автоматическом), сбрасывает его. Если ожидающих потоков нет, PulseEvent просто сбрасывает событие.

Объект-семафор ─ это фактически объект-взаимоисключение со счетчиком. Данный объект позволяет захватить себя определенному количеству потоков. Семафоры применяются для ограничения количества потоков, одновременно работающих с ресурсом. Объекту при инициализации передается максимальное число потоков, после каждого захвата счетчик семафора уменьшается. Сигнальному состоянию соответствует значение счетчика больше нуля.

Объект-критическая секция позволяет выделить участок кода, где поток получает доступ к разделяемому ресурсу, и предотвратить одновременное использование ресурса. Перед использованием ресурса поток входит в критическую секцию (вызывает функцию EnterCriticalSection). Если после этого какой-либо другой поток попытается войти в ту же самую критическую секцию, его выполнение приостановится, пока первый поток не покинет секцию с помощью вызова LeaveCriticalSection. В отличие от других (именованных) объектов, может применяться только в рамках одного процесса.

Преимуществом работы именно с критической секцией является наличие функции TryEnterCriticalSection, которая проверяет, занята ли критическая секция в данный момент. С ее помощью поток в процессе ожидания доступа к ресурсу может не блокироваться, а выполнять какие-то полезные действия.

Кроме того, существует ряд функций, позволяющих работать с глобальными переменными из всех потоков не заботясь о синхронизации, т.к. эти функции сами за ней следят. Это функции InterlockedIncrement / InterlockedDecrement, InterlockedExchange,InterlockedExchangeAdd и InterlockedCompareExchange. Например, функция InterlockedIncrement увеличивает значение 32-битной переменной на единицу - удобно использовать для различных счетчиков.[2]

## 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной работы было создано приложение, на основе предыдущей лабораторной работы, направленное на демонстрацию синхронного выполнения потоков с использованием различых средств синхронизации и взаимного исключения (события, мьютекс, критическую секцию). Мьютекс используется для организации записи в один файл из нескольких потоков (рисунок 1).

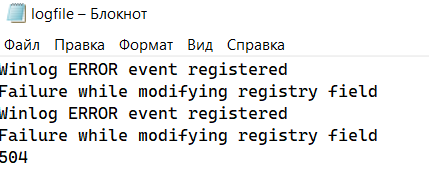


Рисунок 1 ─ результат записи в файл несколькими потоками

Кроме того, каждый поток ожидает наступление определенного события перед тем, как совершить запись в файл определенного сообщеия или вывода исчезающей строки на экран (рисунок 2).



Рисунок 2 ─ строка

Критическая секция используется для прекращения работы импровизированного таймера по закрытию окна и записи времени работы программы в файл (рисунок 1).

## 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. Кроме того настроена система событий и оповещений с записью в файл, основанная на синхронизации многопоточности.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Создание классических приложений для Windows с использованием API Win32 [Электронный ресурс]. ─ Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/api ─ Дата доступа 15.11.2023

[2] Многозадачность в Windows [Электронный ресурс]. ─ Режим доступа: https://www.rsdn.org/article/baseserv/mt.xml ─ Дата доступа 15.11.2023

## ПРИЛОЖЕНИЕ А

## (обязательное)

## Листинг кода

**Lab5.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 WriteToFile(LPCWSTR msg, LPCWSTR path)

{

WaitForSingleObject(file\_mutex, INFINITE);

DWORD res = 0;

WriteFile(log\_file, msg, wcslen(msg)\*2, &res, NULL);

ReleaseMutex(file\_mutex);

}

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);

SetEvent(winlog\_success\_event);

}

}

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);

SetEvent(winlog\_failure\_event);

}

}

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);

SetWindowTextA(msg\_line, "Congratulations! File icons were changed successfully!");

SetEvent(reg\_success\_event);

}

else

{

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

SetWindowTextA(msg\_line, "Some troubles accured while trying to change the icon!");

SetEvent(reg\_failure\_event);

}

}

else

{

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

SetWindowTextA(msg\_line, "Some troubles accured while trying to change the icon!");

SetEvent(reg\_failure\_event);

}

SetEvent(clear\_event);

}

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);

// сообщение

msg\_line = CreateWindowA("static", "", WS\_VISIBLE | WS\_CHILD | ES\_LEFT, 10, 400, 450, 30, window, NULL, NULL, NULL);

}

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:

{

LeaveCriticalSection(&exit\_section);

Sleep(10);

DeleteCriticalSection(&exit\_section);

CloseHandle(clear\_event);

CloseHandle(reg\_success\_event);

CloseHandle(reg\_failure\_event);

CloseHandle(winlog\_success\_event);

CloseHandle(winlog\_failure\_event);

CloseHandle(file\_mutex);

CloseHandle(log\_file);

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);

}

}

DWORD WINAPI logSucsessRegistry(LPVOID lpParameter)

{

while (true) {

WaitForSingleObject(reg\_success\_event, INFINITE);

WriteToFile(log\_reg\_success, logfile\_path);

}

}

DWORD WINAPI logFailureRegistry(LPVOID lpParameter)

{

while (true) {

WaitForSingleObject(reg\_failure\_event, INFINITE);

WriteToFile(log\_reg\_failure, logfile\_path);

}

}

DWORD WINAPI logSucsessWinlog(LPVOID lpParameter)

{

while (true) {

WaitForSingleObject(winlog\_success\_event, INFINITE);

WriteToFile(log\_winlog\_success, logfile\_path);

}

}

DWORD WINAPI logFailureWinlog(LPVOID lpParameter)

{

while (true) {

WaitForSingleObject(winlog\_failure\_event, INFINITE);

WriteToFile(log\_winlog\_failure, logfile\_path);

}

}

DWORD WINAPI ClearMsg(LPVOID lpParameter)

{

while (true) {

WaitForSingleObject(clear\_event, INFINITE);

Sleep(2000);

SetWindowTextA(msg\_line, "");

}

}

DWORD WINAPI TimerSmth(LPVOID lpParameter)

{

DWORD secs = 0;

while (true) {

Sleep(1);

secs++;

// завершилась ли программа

if (TryEnterCriticalSection(&exit\_section)) {

WCHAR wsecs[10];

swprintf\_s(wsecs, 10, L"%d", secs);

WriteToFile(wsecs, logfile\_path);

break;

}

}

LeaveCriticalSection(&exit\_section);

ExitThread(0);

}

VOID AddASinchronizationObjects(VOID) {

InitializeCriticalSection(&exit\_section);

EnterCriticalSection(&exit\_section);

file\_mutex = CreateMutex(NULL, FALSE, L"file\_mutex");

clear\_event = CreateEvent(NULL, FALSE, FALSE, L"clear\_event");

reg\_success\_event = CreateEvent(NULL, FALSE, FALSE, L"reg\_success\_event");

reg\_failure\_event = CreateEvent(NULL, FALSE, FALSE, L"reg\_failure\_event");

winlog\_success\_event = CreateEvent(NULL, FALSE, FALSE, L"winlog\_success\_event");

winlog\_failure\_event = CreateEvent(NULL, FALSE, FALSE, L"winlog\_failure\_event");

log\_file = CreateFile(logfile\_path, FILE\_GENERIC\_WRITE, FILE\_SHARE\_READ | FILE\_SHARE\_WRITE, 0, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, 0);

thread1 = CreateThread(NULL, 0, logSucsessRegistry, NULL, 0, NULL);

thread2 = CreateThread(NULL, 0, logFailureRegistry, NULL,0, NULL);

thread3 = CreateThread(NULL, 0, logSucsessWinlog, NULL, 0, NULL);

thread4 = CreateThread(NULL, 0, logFailureWinlog, NULL,0, NULL);

thread5 = CreateThread(NULL, 0, ClearMsg, NULL, 0, NULL);

thread6 = CreateThread(NULL, 0, TimerSmth, NULL, 0, NULL);

}

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

AddASinchronizationObjects();

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(240, 240, 240)), 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;

CRITICAL\_SECTION exit\_section = { 0 };

HANDLE file\_mutex = NULL;

HANDLE clear\_event = NULL;

HANDLE reg\_success\_event = NULL;

HANDLE reg\_failure\_event = NULL;

HANDLE winlog\_success\_event = NULL;

HANDLE winlog\_failure\_event = NULL;

HANDLE thread1 = NULL;

HANDLE thread2 = NULL;

HANDLE thread3 = NULL;

HANDLE thread4 = NULL;

HANDLE thread5 = NULL;

HANDLE thread6 = NULL;

LPCWSTR log\_reg\_success = L"Registry field modified - icon changed\n";

LPCWSTR log\_reg\_failure = L"Failure while modifying registry field\n";

LPCWSTR log\_winlog\_success = L"Winlog INFO event registered\n";

LPCWSTR log\_winlog\_failure = L"Winlog ERROR event registered\n";

LPCWSTR logfile\_path = L"logfile.txt";

HANDLE log\_file = NULL;

HWND msg\_line = NULL;

#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

HANDLE nasty = NULL;