Sisteme de Operare

**Nicoleta Radu – Tema 5**

#undef UNICODE

#include <windows.h>

#include <math.h>

#include <process.h>

#include "resource.h"

#define REP 10000

#define Oprire 0

#define STATUS\_WORKING 1

#define STATUS\_DONE 2

#define STATUS\_NOTSTARTED 3

#define WM\_CALC\_DONE (WM\_USER + 0)

#define WM\_CALC\_ABORTED (WM\_USER + 1)

typedef struct

{

HWND hwnd;

HANDLE hEvent;

int contor;

BOOL bPornit;

BOOL bStop;

} PARAMS, \* PPARAMS;

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

BOOL CALLBACK DlgProc(HWND hDlg, UINT iMsg,

WPARAM wParam, LPARAM lParam);

BOOL dlgActive = FALSE; //variabila pentru caseta de dialog

HWND hwndMain; //variabila handle a ferestrei principale

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,

PSTR szCmdLine, int iCmdShow)

{

static char szAppName[] = "HelloWin";

HWND hwnd;

MSG msg;

WNDCLASSEX wndclass; // obiectul instantiat din structura WNDCLASSEX

wndclass.cbSize = sizeof(wndclass);//Dimensiunea structurii

wndclass.style = CS\_HREDRAW | CS\_VREDRAW; //Stilul ferestrei

wndclass.lpfnWndProc = WndProc; // Procedura de fereastra care trateaza mesajele

// Doua c�mpuri care permit rezervarea de spatiu suplimentar in structura class, respectiv structura window.

wndclass.cbClsExtra = 0;

wndclass.cbWndExtra = 0;

wndclass.hInstance = hInstance; //Identificatorul instantei ferestrei

//Stabileste o pictograma pentru toate ferestrele create pe baza clasei window

wndclass.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

//Stabileste o pictograma pentru cursor

wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW);

//Specifica culoarea de fundal a ferestrei

wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE\_BRUSH);

wndclass.lpszMenuName = NULL; // Specifica meniul feresterei

wndclass.lpszClassName = szAppName; // Specifica numele clasei feresterei

wndclass.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

//Inregistrarea clasei de fereastra

RegisterClassEx(&wndclass);

// Crearea ferestrei pe baza clasei de fereastra

hwnd = CreateWindow(szAppName, // window class name

"Un program simplu", // window caption

WS\_OVERLAPPEDWINDOW, // window style

CW\_USEDEFAULT, // initial x position

CW\_USEDEFAULT, // initial y position

CW\_USEDEFAULT, // initial x size

CW\_USEDEFAULT, // initial y size

NULL, // parent window handle

NULL, // window menu handle

hInstance, // program instance handle

NULL); // creation parameters

ShowWindow(hwnd, iCmdShow); // afiseaza fereastra pe ecran

//transmite catre WndProc un mesaj de tip WM\_PAINT, care are ca rezultat

//redesenarea ferestrei, adica reactualizarea zonei client a ferestrei

UpdateWindow(hwnd); //

// preia si distribuie mesaje pana se primeste WM\_QUIT

while (GetMessage(&msg, NULL, 0, 0))

{

TranslateMessage(&msg); // traduce mesajul

DispatchMessage(&msg); // transmite mesajul catre procedura de fereastra

}

return msg.wParam;

}

void Thread1(PVOID pvoid)

{

int i;

char text[100];

static HANDLE hEvent;

PPARAMS pparams = (PPARAMS)pvoid;

while (true)

{

WaitForSingleObject(pparams->hEvent, INFINITE);

// verificare daca bStop este TRUE, daca da se iese din while

if (pparams->bStop == TRUE) break;

// verificare daca avem butonul Pornire activ

if (pparams->bPornit == TRUE)

{

// daca da, setare eveniment

SetEvent(hEvent);

for (i = 0; i <= 500 && pparams->bPornit; i++) {

wsprintf(text, "%d", i);

pparams->contor = i;

}

}

// se asteapta dupa un eveniment semnalat

if (pparams->contor < 0)

{

pparams->contor = 0;

}

if (pparams->contor == 500)

{

SendMessage(pparams->hwnd, pparams->bStop, 0,500);

}

// suspendare 500ms

Sleep(500);

//incrementare contor si afisare in caseta de text

}

\_endthread();

}

LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)

{

static HINSTANCE hInstance;

switch (iMsg)

{

case WM\_CREATE: // operatiile ce se executa la crearea ferestrei

// se creaza caseta de dialog

hInstance = ((LPCREATESTRUCT)lParam)->hInstance;

if (!dlgActive) {

DialogBox(hInstance, MAKEINTRESOURCE(IDD\_DIALOG),

hwnd, (DLGPROC)DlgProc);

PostMessage(hwnd, WM\_CLOSE, 0, 0);

// insereaza un nou mesaj nou in coada de asteptare

}

return 0;

case WM\_DESTROY:

PostQuitMessage(0); // insereaza un mesaj de incheiere

return 0;

}

return DefWindowProc(hwnd, iMsg, wParam, lParam);

}

BOOL CALLBACK DlgProc(HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam) {

// Variabile tema - to send

HWND hEditControl;

HFONT hFont;

static HANDLE hEvent;

static INT iStatus;

static PARAMS params;

char text[200] = "0";

switch (iMsg) {

case WM\_INITDIALOG:

// setari initiale

hFont = CreateFont(40, 0, 0, 0, FW\_BOLD, FALSE, FALSE, FALSE, DEFAULT\_CHARSET,

OUT\_OUTLINE\_PRECIS, CLIP\_DEFAULT\_PRECIS, ANTIALIASED\_QUALITY,

VARIABLE\_PITCH, TEXT("Times New Roman"));

hEditControl = GetDlgItem(hDlg,IDC\_EDIT);

SendMessage(hEditControl, WM\_SETFONT, (WPARAM)hFont, MAKELPARAM(TRUE, 0));

SetDlgItemText(hDlg,IDC\_EDIT,text);

params.hEvent = CreateEvent(NULL,FALSE,FALSE,NULL);

params.hwnd = hDlg;

params.bPornit = FALSE;

params.bStop = FALSE;

iStatus = STATUS\_NOTSTARTED;

EnableWindow(GetDlgItem(hDlg, Oprire), FALSE);

\_beginthread(Thread1, 0, &params);

return TRUE;

case WM\_CLOSE:

//dlgActive = FALSE;

EndDialog(hDlg, 0);

return TRUE;

case WM\_COMMAND:

switch (LOWORD(wParam)) {

case IDC\_BUTTON\_PORNIRE:

iStatus = STATUS\_WORKING;

params.bPornit = TRUE;

params.bStop = FALSE;

SetEvent(hEvent);

InvalidateRect(hDlg, NULL, TRUE);

return TRUE;

case IDC\_BUTTON\_OPRIRE:

params.bStop = TRUE;

params.bPornit = FALSE;

InvalidateRect(hDlg, NULL, TRUE);

return TRUE;

case IDC\_CANCEL:

EndDialog(hDlg, 0);

return TRUE;

}

break;

}

return FALSE;

}

