Wrocław University of Science and Technology

Faculty of Microsystem Electronics and Photonics

Operating systems

Lab. 4.

I. Issues to prepare

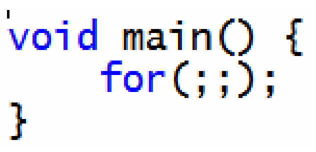
- Windows API library
- Win32 programming
- Threads is Windows

II. Outline

1. Threads implementation in C
2. Windows windows creation
3. Tasks

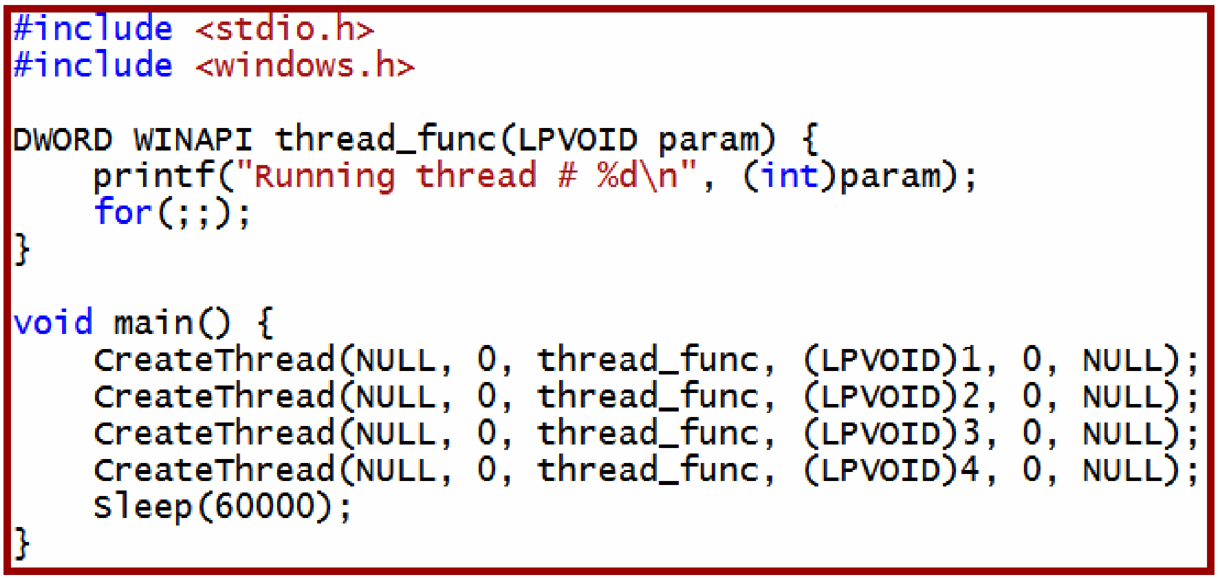
III. Tasks

1. Start C++ programming environment under Windows (choose your favourite one, CodeBlocks, Visual etc.).
2. Check CPU utilisation with Windows Task Manager during running present blocking function (infinite *for* loop):



```
void main() {
    for(;;);
}
```

3. Create few threads:



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

DWORD WINAPI thread_func(LPVOID param) {
    printf("Running thread # %d\n", (int)param);
    for(;;);
}

void main() {
    CreateThread(NULL, 0, thread_func, (LPVOID)1, 0, NULL);
    CreateThread(NULL, 0, thread_func, (LPVOID)2, 0, NULL);
    CreateThread(NULL, 0, thread_func, (LPVOID)3, 0, NULL);
    CreateThread(NULL, 0, thread_func, (LPVOID)4, 0, NULL);
    Sleep(60000);
}
```

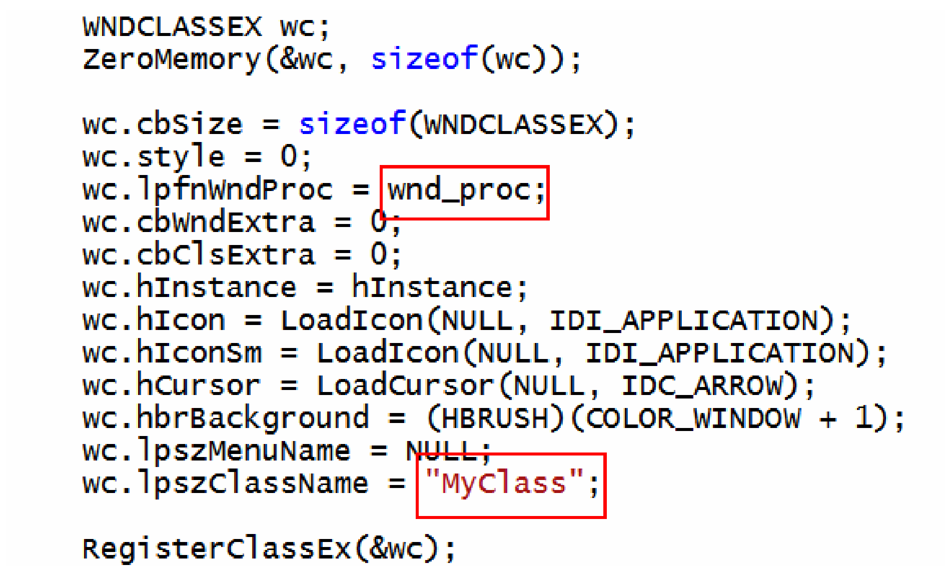
4. What is the utilisation of CPU cores right now? How many threads do you need to fill all logical cores?
5. Make one threads which is changing some variables values all the time and the second one which is displaying present values of those variables.

```c
#include <stdio.h>
#include <windows.h>

int a;
int b;

DWORD WINAPI thread_calculations(LPVOID param) {
    for(;;) { a=1; a=2; a=3; b=1; b=2; b=3; }
}
DWORD WINAPI thread_display(LPVOID param) {
    for(;;) { printf("a=%d b=%d\n", a, b); Sleep(100); }
}
void main() {
    CreateThread(NULL, 0, thread_calculations, NULL, 0, NULL);
    CreateThread(NULL, 0, thread_display, NULL, 0, NULL);
    Sleep(60000);
}
```

6. How displaying of variables behaves? If you can, stabilize it by utilizing critical sections: *EnterCriticalSection* and *LeaveCriticalSection*.

WinAPI:

7. Create basic Win32 windows application. You can use template from programming environment or google it.



```c
WNDCLASSEX wc;
ZeroMemory(&wc, sizeof(wc));

wc.cbSize = sizeof(WNDCLASSEX);
wc.style = 0;
wc.lpfnWndProc = wnd_proc;
wc.cbWndExtra = 0;
wc.cbClsExtra = 0;
wc.hInstance = hInstance;
wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);
wc.hIconSm = LoadIcon(NULL, IDI_APPLICATION);
wc.hCursor = LoadCursor(NULL, IDC_ARROW);
wc.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
wc.lpszMenuName = NULL;
wc.lpszClassName = "MyClass";

RegisterClassEx(&wc);
```

8. Use the messages handling function to change window title after left mouse button clicking:

```
LRESULT CALLBACK wnd_proc(
    HWND hwnd, UINT message, WPARAM wp, LPARAM lp)
{
    switch(message) {
    case WM_LBUTTONDOWN:
        SetWindowText(hwnd, "User clicked left mouse button");
        break;
    case WM_DESTROY:
        PostQuitMessage(0);
    default:
        return DefWindowProc(hwnd, message, wp, lp);
    }
    return 0;
}
```

9. Modify code to change windows title only after double click.
10. Check different window styles.
11. Try to execute blocking function (*for(;;)*) after clicking the mouse button. How window behaves?
12. Repair above situation by introducing separate thread.
13. Close the windows by double click.