1. Лабораторная работа № 2

Использование программного интерфейса Win API. Процессы и потоки в ОС Windows

* 1. Цель работы

Изучение возможности использования программного интерфейса приложений (API) операционных систем Windows 95-2000, NT, XP, 7. Приобретение практических навыков создания и управления процессами и потоками, используя Win API в средах программирования Borland Delphi, C++ Builder или Visual Studio.

* 1. Постановка задачи

Изучить способы создания процессов и потоков, управления ими с помощью WinAPI.

* 1. Ход выполнения работы
     1. Тексты программ

Ниже представлены тексты программ, написанные на языке программирования Pascal для компилятора Free Pascal Compiler 3.2.0 for x86\_64.

* + - 1. Программа Master

Program Master;

Uses Crt, Windows, SysUtils;

Var ProcInfo: array [1..2] of PROCESS\_INFORMATION;

StartInfo: array [1..2] of STARTUPINFOA;

begin

StartInfo[1].cb := sizeof(STARTUPINFOA);

StartInfo[2].cb := sizeof(STARTUPINFOA);

// Тест №1

StartInfo[1].lpTitle := 'Test #1; Process: 01; Priority: 01';

StartInfo[2].lpTitle := 'Test #1; Process: 02; Priority: 02';

CreateProcess(nil, '.\sort.exe', nil, nil, false, CREATE\_NEW\_CONSOLE or CREATE\_SUSPENDED, nil, nil, StartInfo[1], ProcInfo[1]);

CreateProcess(nil, '.\sort.exe', nil, nil, false, CREATE\_NEW\_CONSOLE or CREATE\_SUSPENDED, nil, nil, StartInfo[2], ProcInfo[2]);

SetPriorityClass(ProcInfo[1].hProcess, IDLE\_PRIORITY\_CLASS);

SetPriorityClass(ProcInfo[2].hProcess, NORMAL\_PRIORITY\_CLASS);

SetThreadAffinityMask(ProcInfo[1].hThread, 2);

SetThreadAffinityMask(ProcInfo[2].hThread, 2);

ResumeThread(ProcInfo[1].hThread);

ResumeThread(ProcInfo[2].hThread);

WaitForSingleObject(ProcInfo[1].hProcess, INFINITE);

WaitForSingleObject(ProcInfo[2].hProcess, INFINITE);

// Тест №2

StartInfo[1].lpTitle := 'Test #2; Process: 01; Priority: 01'; // Задать имя для процессов

StartInfo[2].lpTitle := 'Test #2; Process: 02; Priority: 03';

// Создать процессы

CreateProcess(nil, '.\sort.exe', nil, nil, false, CREATE\_NEW\_CONSOLE or CREATE\_SUSPENDED, nil, nil, StartInfo[1], ProcInfo[1]);

CreateProcess(nil, '.\sort.exe', nil, nil, false, CREATE\_NEW\_CONSOLE or CREATE\_SUSPENDED, nil, nil, StartInfo[2], ProcInfo[2]);

// Изменить приоритеты и ядра, на которых будут выполняться потоки процессов

SetPriorityClass(ProcInfo[1].hProcess, IDLE\_PRIORITY\_CLASS);

SetPriorityClass(ProcInfo[2].hProcess, HIGH\_PRIORITY\_CLASS);

SetThreadAffinityMask(ProcInfo[1].hThread, 2);

SetThreadAffinityMask(ProcInfo[2].hThread, 2);

// Запустить выполнение потоков

ResumeThread(ProcInfo[1].hThread);

ResumeThread(ProcInfo[2].hThread);

// Ждать завершения работы процессов

WaitForSingleObject(ProcInfo[1].hProcess, INFINITE);

WaitForSingleObject(ProcInfo[2].hProcess, INFINITE);

// Тест №3

StartInfo[1].lpTitle := 'Test #3; Process: 01; Priority: 02';

StartInfo[2].lpTitle := 'Test #3; Process: 02; Priority: 02';

CreateProcess(nil, '.\sort.exe', nil, nil, false, CREATE\_NEW\_CONSOLE or CREATE\_SUSPENDED, nil, nil, StartInfo[1], ProcInfo[1]);

CreateProcess(nil, '.\sort.exe', nil, nil, false, CREATE\_NEW\_CONSOLE or CREATE\_SUSPENDED, nil, nil, StartInfo[2], ProcInfo[2]);

SetPriorityClass(ProcInfo[1].hProcess, NORMAL\_PRIORITY\_CLASS);

SetPriorityClass(ProcInfo[2].hProcess, NORMAL\_PRIORITY\_CLASS);

SetThreadAffinityMask(ProcInfo[1].hThread, 2);

SetThreadAffinityMask(ProcInfo[2].hThread, 2);

ResumeThread(ProcInfo[1].hThread);

ResumeThread(ProcInfo[2].hThread);

WaitForSingleObject(ProcInfo[1].hProcess, INFINITE);

WaitForSingleObject(ProcInfo[2].hProcess, INFINITE);

end.

* + - 1. Программа Sort

Program Sort;

Uses Crt, Windows, SysUtils;

Const arr\_size = 4200;

repl\_count = 100;

Type arr = array [0..arr\_size - 1] of Integer;

procedure shell\_sort (var a: arr);

var d, i, j, tmp: Integer;

begin

d := arr\_size div 2;

while (d > 0) do

begin

for i := 0 to arr\_size do

begin

j := i + d;

while (j < arr\_size) do

begin

if (a[i] < a[j]) then

begin

tmp := a[i];

a[i] := a[j];

a[j] := tmp;

end;

j := j + d;

end;

end;

d := d div 2

end;

end;

procedure fill\_array (var a: arr);

var i: Integer;

begin

for i := 0 to arr\_size do

a[i] := 100;

end;

Var Tstart, Tstop, t\_start, t\_stop: SYSTEMTIME;

tmp: DWORD;

counter: Integer; a: arr;

begin

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

GetSystemTime(Tstart);

with Tstart do

writeln('Время Tstart: ', wMinute, ':', wSecond, '.', wMilliseconds);

GetSystemTime(t\_start);

for counter := 0 to repl\_count do

begin

fill\_array(a);

shell\_sort(a);

end;

GetSystemTime(t\_stop);

tmp := (t\_stop.wMilliseconds + t\_stop.wSecond \* 1000 + t\_stop.wMinute \* 60000) - (t\_start.wMilliseconds + t\_start.wSecond \* 1000 + t\_start.wMinute \* 60000);

writeln('Общее время сортировки, мс: ', tmp);

writeln('Время одной сортировки, мс: ', tmp / repl\_count);

GetSystemTime(Tstop);

with Tstop do

writeln('Время Tstop: ', wMinute, ':', wSecond, '.', wMilliseconds);

Readln();

end.

* + - 1. Программа Threads

Program Thread;

Uses Windows, SysUtils;

Const arr\_size = 50;

Type arr = array [0..arr\_size - 1] of Integer;

VAR a: arr;

procedure shell\_sort ();

var d, i, j, tmp: Integer;

begin

d := arr\_size div 2;

while (d > 0) do

begin

for i := 0 to arr\_size do

begin

j := i + d;

while (j < arr\_size) do

begin

if (a[i] < a[j]) then

begin

tmp := a[i];

a[i] := a[j];

a[j] := tmp;

end;

j := j + d;

end;

end;

d := d div 2

end;

writeln('Sorting is complete.');

end;

procedure fill\_array (var a: arr);

var i: Integer;

begin

for i := 0 to arr\_size do

a[i] := 100;

end;

procedure print\_array ();

var i: Integer;

begin

write('arr = [');

for i := 0 to (arr\_size - 1) do

write(a[i], ', ');

writeln(a[arr\_size - 1], ']');

writeln('Printing is complete.');

end;

Var ti1, ti2: DWORD;

th1, th2: HANDLE;

BEGIN

fill\_array(a);

th1 := CreateThread(nil, 0, @print\_array, nil, CREATE\_SUSPENDED, ti1);

th2 := CreateThread(nil, 0, @shell\_sort, nil, CREATE\_SUSPENDED, ti2);

SetThreadPriority(th1, THREAD\_PRIORITY\_NORMAL);

SetThreadPriority(th1, THREAD\_PRIORITY\_NORMAL);

ResumeThread(th1);

ResumeThread(th2);

WaitForSingleObject(th1, INFINITE);

WaitForSingleObject(th2, INFINITE);

END.

* + 1. Выполнение заданий

Тестовые примеры продемонстрированы на Рисунках 2.1 – 2.3 (работа с процессами) и на Рисунках 2.4–2.6 (работа с потоками).

На Рисунке 2.1 продемонстрировано выполнение программ с приоритетами IDLE (справа) и NORMAL (слева). Был сделан вывод, что выполнение процесса с приоритетом IDLE начинается не с момента его создания, а лишь через некоторое время.

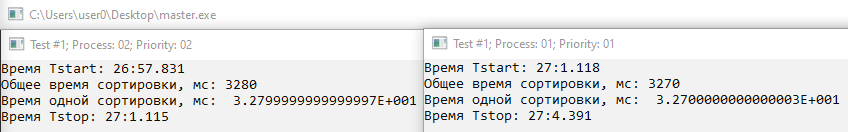


Рисунок 2.1 ­– Процессы с приоритетами IDLE и NORMAL

На Рисунке 2.2 продемонстрировано выполнение программ с приоритетами IDLE и HIGH (слева и справа соответственно). Был сделан вывод, что выполнение процесса с приоритетом IDLE начинается лишь только после того, как процесс HIGH завершит свою работу.

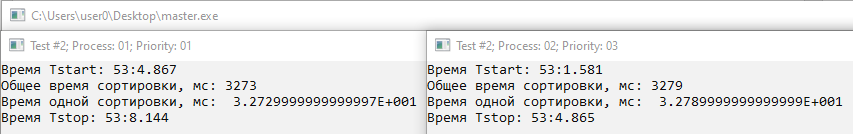


Рисунок 2.2 – Процессы с приоритетами IDLE и HIGH

На Рисунке 2.3 продемонстрировано выполнение программ с приоритетами NORMAL и NORMAL. Был сделан вывод, что вследствие одинакового приоритета на выполнение каждого процесса отведено одинаковое количество процессорного времени, а значит они завершат работу в одно и то же время при одинаковом времени работы.

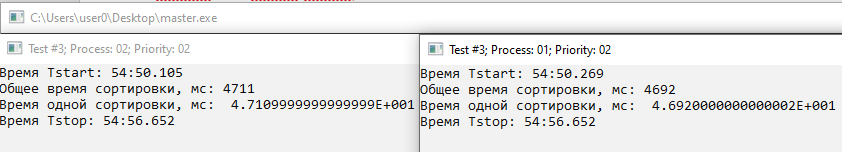


Рисунок 2.3 – Процессы с приоритетами NORMAL и NORMAL

На Рисунке 2.4 оба потока имеют приоритет NORMAL. Из результатов тестирования видно, что функция сортировки завершает свою работу до функции вывода массива на экран.

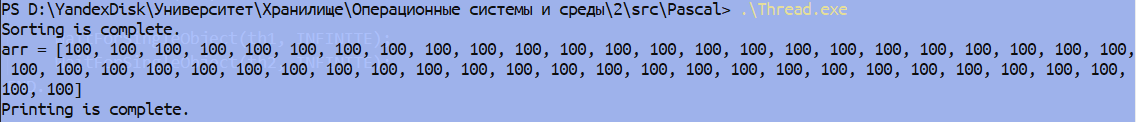


Рисунок 2.4 – Тест с потоками №1

На Рисунке 2.5 потоки имеют приоритеты TIME\_CRITICAL (печать массива) и NORMAL (сортировка массива). Из результатов тестирования видно, что функция сортировки завершает своё выполнение во время вывода массива на экран.

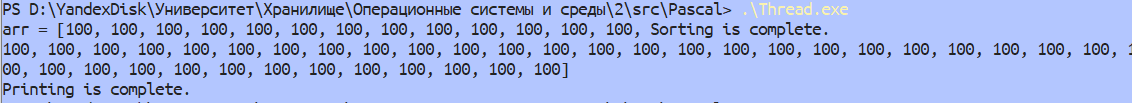


Рисунок 2.4 – Тест с потоками №2

На Рисунке 2.6 потоки имеют приоритеты TIME\_CRITICAL (печать массива) и LOWEST (сортировка массива). Из результатов тестирования видно, что функция сортировки завершает своё выполнение во время вывода массива на экран.

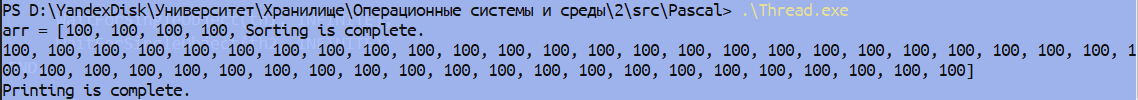


Рисунок 2.5 – Тест с потоками №3

Выводы

При выполнении данной лабораторной работы были получены навыки работы с процессами и потоками с помощью библиотеки WinAPI. Также были повторно закреплены навыки написания методов сортировки, работы с командной строкой Windows. Полученные знания позволят создавать многопоточные приложения, которые на многоядерных системах будут работать эффективнее.