Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Институт информационных технологий и управления в технических системах

ОТЧЕТ

по лабораторной работе №2

Использование программного интерфейса Win API.

Процессы и потоки в ОС Windows.

Выполнил: ст. гр. ИС/б-21

Куркчи А.Е.

Проверил:

Шишкевич В.Е.

Севастополь

2015

# Цель

# 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
   1. Написать программу **Sort**, реализующую следующий алгоритм:
      1. Зафиксировать время начала **Tstrt** выполнения программы (пример получения текущего системного времени и вычисления разности между двумя значениями времени приведен в программе TimeDifference, представленной в Приложении).
      2. Вывести на экран время **Tstrt** в формате: минуты:секунды:миллисекунды;
      3. Зафиксировать текущий момент времени t1;
      4. Для **i** от 1 до 100(количество повторений может быть изменено в зависимости от быстродействия процессора) повторять:
         1. Заполнить массив целых чисел случайными значениями из диапазона 0-10000;
         2. Отсортировать массив;
      5. Зафиксировать текущий момент времени t2;
      6. Определить среднее время одной сортировки: (t2-t1)/100;
      7. Вывести на экран среднее время одной сортировки(в миллисекундах);
      8. Зафиксировать время окончания **Tend** выполнения программы;
      9. Вывести на экран время **Tend** в формате: минуты:секунды: миллисекунды;

Язык программирования, метод и направление сортировки, а также количество элементов массива N выбирается в соответствии с вариантом задания, приведенным в таблице 3.1.

* 1. Написать программу Master, выполняющую следующие действия:

3.2.1. Для **i** от 1 до 3 повторять:

3.2.2.1. Используя системные вызовы **CreateProcess,** создать два процесса **Sort** склассами приоритетов, в соответствии с вариантом задания (таблица 3.1). Необходимо, чтобы каждый процесс имел собственную консоль и окно консоли имело заголовок: "Process: *NP*; Prioritet: *PP*", где *NP* – номер процесса (1 или 2), а *PP* – приоритет соответствующего процесса. (Для изменения свойств окна консоли использовать структуру STARTUPINFO).

3.2.2.2. Ожидать окончания процессов, созданных в п. 3.2.2.1 (использовать функцию WaitForSingleObject).

* 1. Зафиксировать для отчета значения времени, получаемые при выполнении процессов в п 3.2.1.
  2. Написать программу **Threads,** содержащую процедуру сортировки массива (разработанную при выполнении пункта 3.1), содержащего N/50 элементов и процедуру вывода массива на экран – **mass\_print**. Программа должна выполнять следующие действия:
     1. Генерировать случайный массив, содержащий N/50 элементов.
     2. Используя системные вызовы **CreateThread,** создать программные потоки **sort** и **mass\_print** в приостановленном состоянии.
     3. Установить приоритеты потоков в THREAD\_PRIORITY\_NORMAL, используя системный вызов **SetThreadPriority.**
     4. Активизировать потоки, используя системные вызовы **ResumeThread**;
     5. Изменяя приоритеты потоков в разработанной программе в различных сочетаниях фиксировать получаемые результаты.

1. Текст программ

Программа Master:

#include "stdafx.h"

**using** **namespace** std**;**

int main**(**int argc**,** char **\***argv**[])** **{**

int i**,**j**,**

sost**;**

char cmd**[**128**],**

window1**[**64**],**

window2**[**64**],**

cmd1**[**128**],**

cmd2**[**128**];**

**if** **(**argc **!=** 1**)** **{**

strcpy**(**cmd**,** argv**[**1**]);**

**}** **else** **{**

strcpy**(**cmd**,** "Lab.02.Slave.exe"**);**

**}**

int pid **=** GetCurrentProcessId**();**

cout **<<** "Master: Starting:" **<<** cmd **<<** " PID: " **<<** pid **<<** "\n"**;**

cout**.**flush**();**

STARTUPINFO info1**,**

info2**;**

memset**(&**info1**,** 0**,** **sizeof(**info1**));**

info1**.**cb **=** **sizeof(**info1**);**

memset**(&**info2**,** 0**,** **sizeof(**info2**));**

info2**.**cb **=** **sizeof(**info2**);**

**for(**i**=**1**;**i**<=**3**;**i**++)** **{**

sprintf**(**cmd1**,**"%s %d %d"**,**cmd**,**i**,**1**);**

sprintf**(**cmd2**,**"%s %d %d"**,**cmd**,**i**,**2**);\**

cout **<<** cmd1 **<<** endl**;**

cout **<<** cmd2 **<<** endl**;**

DWORD flags1**,**

flags2**;**

**switch** **(**i**)** **{**

**case** 1**:**

sprintf**(**window1**,**"Process: 1; Prioritet: %d"**,**1**);**

sprintf**(**window2**,**"Process: 2; Prioritet: %d"**,**1**);**

flags1 **=** CREATE\_NEW\_CONSOLE **|** IDLE\_PRIORITY\_CLASS**;**

flags2 **=** CREATE\_NEW\_CONSOLE **|** IDLE\_PRIORITY\_CLASS**;**

**break;**

**case** 2**:**

sprintf**(**window1**,**"Process: 1; Prioritet: %d"**,**2**);**

sprintf**(**window2**,**"Process: 2; Prioritet: %d"**,**3**);**

flags1 **=** CREATE\_NEW\_CONSOLE **|** NORMAL\_PRIORITY\_CLASS**;**

flags2 **=** CREATE\_NEW\_CONSOLE **|** HIGH\_PRIORITY\_CLASS**;**

**break;**

**case** 3**:**

sprintf**(**window1**,**"Process: 1; Prioritet: %d"**,**3**);**

sprintf**(**window2**,**"Process: 2; Prioritet: %d"**,**1**);**

flags1 **=** CREATE\_NEW\_CONSOLE **|** HIGH\_PRIORITY\_CLASS**;**

flags2 **=** CREATE\_NEW\_CONSOLE **|** IDLE\_PRIORITY\_CLASS**;**

**break;**

**}**

info1**.**lpTitle **=** window1**;**

info2**.**lpTitle **=** window2**;**

PROCESS\_INFORMATION pinfo1**,**

pinfo2**;**

CreateProcess**(NULL,**cmd1**,NULL,NULL,**TRUE**,**flags1**,NULL,NULL,&**info1**,&**pinfo1**);**

SetProcessAffinityMask**(**pinfo1**.**hProcess**,**1**);**

CreateProcess**(NULL,**cmd2**,NULL,NULL,**TRUE**,**flags2**,NULL,NULL,&**info2**,&**pinfo2**);**

SetProcessAffinityMask**(**pinfo2**.**hProcess**,**1**);**

WaitForSingleObject**(**pinfo1**.**hProcess**,**INFINITE**);**

WaitForSingleObject**(**pinfo2**.**hProcess**,**INFINITE**);**

**}**

cout **<<** "Master: Pause\n"**;**

system**(**"pause"**);**

cout **<<** "Master: Exiting\n"**;**

return **0;**

**}**

Программа Slave:

#include "stdafx.h"

**using** **namespace** std**;**

const int N **=** 3700**;**

const int T **=** 1000**;**

void fill**(**int**\*** a**,**int n**)** **{**

**for(**int i**=**0**;**i**<**n**;**i**++)** **{**

a**[**i**]** **=** rand**()%**10000**;**

**}**

**}**

void sort**(**int**\*** a**,**int n**)** **{**

**for(**int i**=**0**;**i**<**n**-**1**;**i**++)** **{**

int min **=** i**;**

**for(**int j**=**i**+**1**;**j**<**n**;**j**++)** **{**

**if(**a**[**j**]** **<** a**[**min**])** **{**

min **=** j**;**

**}**

**}**

**if(**min **!=** i**)** **{**

int tmp **=** a**[**i**];**

a**[**i**]** **=** a**[**min**];**

a**[**min**]** **=** tmp**;**

**}**

**}**

**}**

int main**(**int argc**,** char **\***argv**[])** **{**

**if(**argc **!=** 3**)** **{**

**return** 1**;**

**}**

int np **=** argv**[**1**][**0**]-**'0'**;**

int pp **=** argv**[**2**][**0**]-**'0'**;**

int**\*** a **=** **new** int**[**N**+**5**];**

auto start\_time **=** chrono**::**high\_resolution\_clock**::**now**();**

**for(**int t**=**0**;**t**<**T**;**t**++)** **{**

fill**(**a**,**N**);**

sort**(**a**,**N**);**

**}**

auto end\_time **=** chrono**::**high\_resolution\_clock**::**now**();**

long long time **=** chrono**::**duration\_cast**<**chrono**::**milliseconds**>(**end\_time**-**start\_time**).**count**();**

cout **<<** "Full time: " **<<** time **<<** "ms" **<<** endl**;**

cout **<<** "Per operation time: " **<<** **(**time**\***1.0**/**T**)** **<<** "ms" **<<** endl**;**

system**(**"pause"**);**

**return** 0**;**

**}**

Программа Thread:

#include "stdafx.h"

**using** **namespace** std**;**

const int N **=** 3700**;**

const int T **=** 1000**;**

void fill**(**int**\*** a**,**int n**)** **{**

**for(**int i**=**0**;**i**<**n**;**i**++)** **{**

a**[**i**]** **=** rand**()%**10000**;**

**}**

**}**

void sort**(**int**\*** a**,**int n**)** **{**

**for(**int i**=**0**;**i**<**n**-**1**;**i**++)** **{**

int min **=** i**;**

**for(**int j**=**i**+**1**;**j**<**n**;**j**++)** **{**

**if(**a**[**j**]** **<** a**[**min**])** **{**

min **=** j**;**

**}**

**}**

**if(**min **!=** i**)** **{**

int tmp **=** a**[**i**];**

a**[**i**]** **=** a**[**min**];**

a**[**min**]** **=** tmp**;**

**}**

**}**

**}**

DWORD WINAPI test**(**const LPVOID lpParam**)** **{**

int**\*** a **=** **new** int**[**N**+**5**];**

auto start\_time **=** chrono**::**high\_resolution\_clock**::**now**();**

**for(**int t**=**0**;**t**<**T**;**t**++)** **{**

fill**(**a**,**N**);**

sort**(**a**,**N**);**

**}**

auto end\_time **=** chrono**::**high\_resolution\_clock**::**now**();**

long long time **=** chrono**::**duration\_cast**<**chrono**::**milliseconds**>(**end\_time**-**start\_time**).**count**();**

cout **<<** "Thread id: " **<<** **((**char**\*)** lpParam**)** **<<** " Avarage time: " **<<** **(**time**\***1.0**/**T**)** **<<** "ms" **<<** endl**;**

**return** 0**;**

**}**

int main**(**int argc**,** char **\***argv**[])** **{**

int pid **=** GetCurrentProcessId**();**

cout **<<** "Thread PID: " **<<** pid **<<** "\n"**;**

HANDLE hT1**,**

hT2**;**

char tN1**[**10**],**

tN2**[**10**];**

**for(**int i**=**1**;**i**<=**3**;**i**++)** **{**

sprintf**(**tN1**,**"%d.%d"**,**i**,**1**);**

sprintf**(**tN2**,**"%d.%d"**,**i**,**2**);**

hT1 **=** CreateThread**(NULL,**0**,&**test**,**tN1**,**CREATE\_SUSPENDED**,NULL);**

hT2 **=** CreateThread**(NULL,**0**,&**test**,**tN2**,**CREATE\_SUSPENDED**,NULL);**

**switch** **(**i**)** **{**

**case** 1**:**

SetThreadPriority**(**hT1**,**THREAD\_PRIORITY\_BELOW\_NORMAL**);**

SetThreadPriority**(**hT2**,**THREAD\_PRIORITY\_BELOW\_NORMAL**);**

**break;**

**case** 2**:**

SetThreadPriority**(**hT1**,**THREAD\_PRIORITY\_NORMAL**);**

SetThreadPriority**(**hT2**,**THREAD\_PRIORITY\_ABOVE\_NORMAL**);**

**break;**

**case** 3**:**

SetThreadPriority**(**hT1**,**THREAD\_PRIORITY\_ABOVE\_NORMAL**);**

SetThreadPriority**(**hT2**,**THREAD\_PRIORITY\_BELOW\_NORMAL**);**

**break;**

**}**

ResumeThread**(**hT1**);**

ResumeThread**(**hT2**);**

WaitForSingleObject**(**hT1**,**INFINITE**);**

WaitForSingleObject**(**hT2**,**INFINITE**);**

**}**

cout **<<** "Thread pause" **<<** endl**;**

system**(**"pause"**);**

**return** 0**;**

**}**

1. Результат

Результат работы программы Master показан на рисунках 1-3.

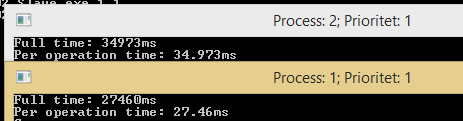


Рисунок 1 – первый запуск процессов

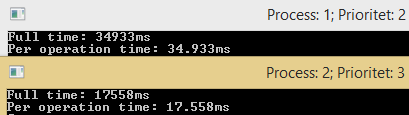


Рисунок 2 – второй запуск процессов

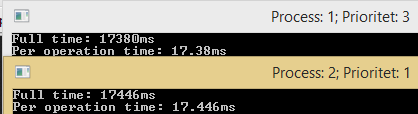


Рисунок 3 – третий запуск процессов

Результат работы программы Thread показан на рисунке 4.

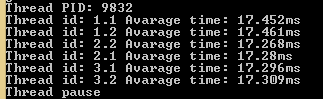


Рисунок 4 – запуск потоков

Выводы

# В ходе лабораторной работы были изучены методы запуска и управления процессами и потоками средствами WinAPI и языка Visual C++. В процессе замеров были получены различающиеся в зависимости от приоритета времена выполнения задач. Из-за невозможности в полной мере изолировать программы от внешнего воздействия и вынести из на строго определённые процессорные мощности возникают погрешности, а иногда и скачки значений из-за занятости процессом, не зависящим от хода эксперимента.