**1 获取cpu物理内核数目以及逻辑内核数目**

需要在引用中添加Management才能使用

// Get the WMI class

ManagementClass c = new ManagementClass(

new ManagementPath("Win32\_Processor"));

// Get the properties in the class

ManagementObjectCollection moc = c.GetInstances();

foreach (ManagementObject mo in moc)

{

PropertyDataCollection properties = mo.Properties;

//获取内核数代码

string cpu= "物理内核数:" + properties["NumberOfCores"].Value + "\r";

string total = "逻辑内核数:" + properties["NumberOfLogicalProcessors"].Value + "\r";

Console.WriteLine(cpu);

其他属性获取代码

foreach (PropertyData property in properties)

{

Console.WriteLine( property.Name + ":" + property.Value + "\r");

}

}

**2 线程绑定cpu函数**

**static extern UIntPtr SetThreadAffinityMask(IntPtr hThread,**

**UIntPtr dwThreadAffinityMask);**

指定线程hThread运行的核心dwThreadAffinityMask

**static extern IntPtr GetCurrentThread();**

获取当前线程handler

详细使用以下示例

**3 cpu线程绑定以及获取核心数目示例**

需要在引用中添加Management才能使用

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Runtime.InteropServices;

using System.Threading;

using System.Diagnostics;

using System.Management;

///创建8个线程 每两个线程绑定到一个核心上

namespace noBand

{

class Program

{

//获取系统运行时间毫秒级别

[DllImport("kernel32.dll")]

static extern uint GetTickCount();

//SetThreadAffinityMask 指定hThread 运行在 核心 dwThreadAffinityMask

[DllImport("kernel32.dll")]

static extern UIntPtr SetThreadAffinityMask(IntPtr hThread,

UIntPtr dwThreadAffinityMask);

//得到当前线程的handler

[DllImport("kernel32.dll")]

static extern IntPtr GetCurrentThread();

//设置线程数目

static int threads = 8;

public static void changeValue(object pid)

{

int core = (int)pid;

int len = 10000000;

uint[] data = new uint[len];

//将当前线程绑定到指定的cpu核心上

SetThreadAffinityMask(GetCurrentThread(), new UIntPtr(SetCpuID(core)));

uint startTime = GetTickCount();

Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

for (int i = 0; i < len; ++i)

data[i] = GetTickCount();

stopwatch.Stop();

Console.WriteLine("运行时间" + stopwatch.ElapsedMilliseconds.ToString());

}

//将指定的cpu例如0，1，2，3转换为cpu的id号

static ulong SetCpuID(int id)

{

ulong cpuid = 0;

if (id < 0 || id >= System.Environment.ProcessorCount)

{

id = 0;

}

cpuid |= 1UL << id;

return cpuid;

}

static void Main(string[] args)

{

// Get the WMI class

ManagementClass c = new ManagementClass(

new ManagementPath("Win32\_Processor"));

// Get the properties in the class

ManagementObjectCollection moc = c.GetInstances();

foreach (ManagementObject mo in moc)

{

PropertyDataCollection properties = mo.Properties;

//获取内核数代码

string cpu = "物理内核数:" + properties["NumberOfCores"].Value + "\r";

string total = "逻辑内核数:" + properties["NumberOfLogicalProcessors"].Value + "\r";

Console.WriteLine(cpu);

//其他属性获取代码

//foreach (PropertyData property in properties)

//{

// Console.WriteLine( property.Name + ":" + property.Value + "\r");

//}

}

//线程数目

int num = threads;

Console.WriteLine("线程数目" + num);

Thread[] t = new Thread[num];

Stopwatch stopwatch = new Stopwatch();

for (int i = 0; i < num; ++i)

t[i] = new Thread(new ParameterizedThreadStart(changeValue)); ;

stopwatch.Start();

for (int i = 0; i < num; ++i)

{

t[i].Start(i / 2);

}

for (int i = 0; i < num; ++i)

t[i].Join();

stopwatch.Stop();

Console.WriteLine("总运行时间" + stopwatch.ElapsedMilliseconds.ToString());

Console.ReadKey();

}

}

}

**4 自旋锁的简单使用**

**自旋锁的详细解释请参考msdn**

**https://msdn.microsoft.com/zh-cn/library/system.threading.spinlock.aspx?cs-save-lang=1&cs-lang=csharp#code-snippet-2**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace spinLock

{

class Program

{

//得到当前线程的handler

[DllImport("kernel32.dll")]

static extern IntPtr GetCurrentThread();

//创建自旋锁

private static SpinLock spin = new SpinLock();

public static void doWork1()

{

bool lockTaken = false;

try

{

//申请获取锁

spin.Enter(ref lockTaken);

//下面为临界区

for(int i=0;i<10;++i)

{

Console.WriteLine(2);

}

}

finally

{

//工作完毕，或者发生异常时，检测一下当前线程是否占有锁，如果咱有了锁释放它

//以避免出现死锁的情况

if (lockTaken)

spin.Exit();

}

}

public static void doWork2()

{

bool lockTaken = false;

try

{

spin.Enter(ref lockTaken);

for (int i = 0; i < 10; ++i)

{

Console.WriteLine(1);

}

}

finally

{

if (lockTaken)

spin.Exit();

}

}

static void Main(string[] args)

{

Thread[] t = new Thread[2];

t[0] = new Thread(new ThreadStart(doWork1));

t[1] = new Thread(new ThreadStart(doWork2));

t[0].Start();

t[1].Start();

t[0].Join();

t[1].Join();

Console.ReadKey();

}

}

}