

MÔN : LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Bài thực hành số 9.2 : Viết phần mềm demo tính hiệu quả của lập trình multi-thread

I. Mục tiêu :

- Giúp SV làm quen với việc dùng class Thread của namespace System.Threading để quản lý thread.
- Giúp SV thấy được tính hiệu quả của lập trình multi-thread so với lập trình tuần tự.


II. Nội dung :

- Xây dựng chương trình nhỏ cho phép người dùng chọn số thread cần dùng để tính tích của 2 ma trận có kích thước đủ lớn. Sau khi tính xong, chương trình sẽ hiển thị thời gian chạy để người dùng biết độ hiệu quả khi dùng số thread khác nhau.

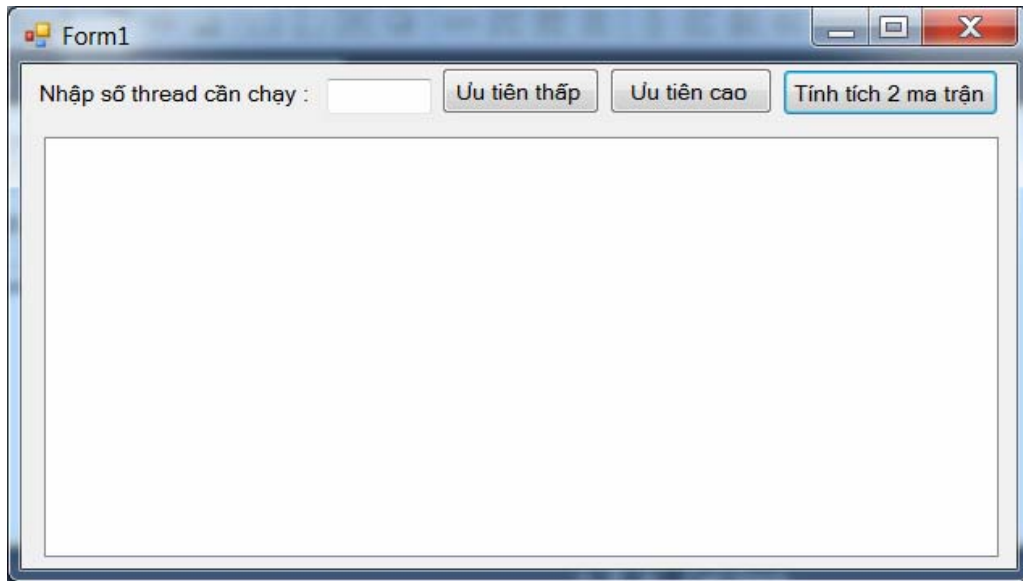
III. Chuẩn đầu ra :


- Sinh viên nắm vững và sử dụng thành thạo class Thread để quản lý thread.

IV. Qui trình :

1. Chạy VS .Net, chọn menu File.New.Project để hiển thị cửa sổ New Project.
2. Mở rộng mục Visual C# trong TreeView "Project Types", chọn mục Windows, chọn icon "Windows Form Application" trong listbox "Templates" bên phải, thiết lập thư mục chứa Project trong listbox "Location", nhập tên Project vào textbox "Name:" (td. ThreadDemo1), click button OK để tạo Project theo các thông số đã khai báo.
3. Form đầu tiên của ứng dụng đã hiển thị trong cửa sổ thiết kế, việc thiết kế form là quá trình lập 4 thao tác tạo mới/xóa/hiệu chỉnh thuộc tính/tạo hàm xử lý sự kiện cho từng đối tượng cần dùng trong form.
4. Nếu cửa sổ Toolbox chưa hiển thị chi tiết, chọn menu View.Toolbox để hiển thị nó (thường nằm ở bên trái màn hình). Click chuột vào button  (Auto Hide) nằm ở góc trên phải của cửa sổ Toolbox để chuyển nó về chế độ hiển thị thường trực.
5. Duyệt tìm phần tử Label (trong nhóm Common Controls), chọn nó, dời chuột về góc trên trái của form và vẽ nó với kích thước mong muốn. Xem cửa sổ thuộc tính của Label vừa vẽ (thường ở góc dưới phải màn hình), duyệt tìm và hiệu chỉnh thuộc tính Text = "Nhập số thread cần chạy :".
6. Duyệt tìm phần tử TextBox (trong nhóm Common Controls), chọn nó, dời chuột về ngay phải Label vừa vẽ và vẽ TextBox với kích thước đủ để nhập số nguyên nhỏ. Xem cửa sổ thuộc tính của TextBox vừa vẽ, duyệt tìm và hiệu chỉnh thuộc tính (Name) = txtThreads.
7. Duyệt tìm phần tử Button (trong nhóm Common Controls), chọn nó, dời chuột về bên phải TextBox vừa vẽ và vẽ Button với kích thước mong muốn. Xem cửa sổ thuộc tính của Button vừa vẽ, duyệt tìm và hiệu chỉnh thuộc tính Text = "Ưu tiên thấp", duyệt tìm và thay đổi thuộc tính (Name) = btnCham.
8. Lặp lại bước 7 để vẽ thêm button ngay bên phải button vừa vẽ với thuộc tính Text = "Ưu tiên cao", (Name) = btnNhanh.
9. Lặp lại bước 8 để vẽ thêm button ngay bên phải button btnNhanh với thuộc tính Text = "Tính tích 2 ma trận", (Name) = btnStart.
10. Duyệt tìm phần tử ListBox (trong nhóm Common Controls), chọn nó, dời chuột về bên dưới Label và vẽ ListBox với kích thước mong muốn. Xem cửa sổ thuộc tính của ListBox vừa vẽ, duyệt tìm và hiệu chỉnh thuộc tính (Name) = lbKetqua.

Sau khi thiết kế xong, Form có dạng sau :



11. Dời chuột về button btnCham, ấn kép chuột vào nó để tạo hàm xử lý sự kiện Click chuột cho button, cửa sổ mã nguồn sẽ hiển thị để ta bắt đầu viết code cho hàm. Cách tổng quát để tạo hàm xử lý sự kiện là chọn đối tượng btnCham, cửa sổ thuộc tính của nó sẽ hiển thị, click icon  để hiển thị danh sách các sự kiện của đối tượng, duyệt tìm sự kiện quan tâm (Click), ấn kép chuột vào comboBox bên phải sự kiện Click để máy tạo tự động hàm xử lý cho sự kiện này. Cửa sổ mã nguồn sẽ hiển thị khung sườn của hàm vừa được tạo với thân rỗng, viết thân cho hàm này như sau :

```
private void btnCham_Click(object sender, EventArgs e)
{
    //thiết lập chế độ quyền ưu tiên realtime cho chương trình
    myPrio = ProcessPriorityClass.BelowNormal;
}
```

11. Dời chuột về button btnNhanh, ấn kép chuột vào nó để tạo hàm xử lý sự kiện Click chuột cho button, cửa sổ mã nguồn sẽ hiển thị khung sườn của hàm vừa được tạo với thân rỗng, viết thân cho hàm này như sau :

```
private void btnNhanh_Click(object sender, EventArgs e)
{
    //thiết lập chế độ quyền ưu tiên realtime cho chương trình
    myPrio = ProcessPriorityClass.RealTime;
}
```

12. Dời chuột về button btnStart, ấn kép chuột vào nó để tạo hàm xử lý sự kiện Click chuột cho button, cửa sổ mã nguồn sẽ hiển thị khung sườn của hàm vừa được tạo với thân rỗng, viết thân cho hàm này như sau :

```
private void btnStart_Click(object sender, EventArgs e)
{
    //xác định đối tượng quản lý process hiện hành
    MyProc = Process.GetCurrentProcess();
    //thay đổi quyền ưu tiên theo yêu cầu người dùng
    MyProc.PriorityClass = myPrio;
    //xác định số thread tham gia tính tích 2 ma trận
    int cnt = Int32.Parse(txtThreads.Text);
    int i;
    //ghi nhận thời điểm bắt đầu tính tích
    DateTime t1 = DateTime.Now;
    if (cnt == 1) { //dùng thuật giải tuần tự
```

```

    TinhTich(new Params(null, 0, N, 0));
}
else //dùng thuật giải song song gồm cnt-1 thread con và 1 thread chính có sẵn
{
    Thread t;
    for (i = 0; i < cnt-1; i++) { //lập tạo và chạy từng thread con
        stateLst[i] = 0; //ghi nhận thread i chưa chạy xong
        //tạo thread mới để chạy hàm TinhTich
        t = new Thread(new ParameterizedThreadStart(TinhTich));
        //thiết lập quyền ưu tiên cho thread i
        t.Priority = tPrio[i % 5];
        //hiển thị độ ưu tiên của thread i
        lbKetqua.Items.Add(String.Format("Thread {0:d} có độ ưu tiên = {1:d}", i,
t.Priority.ToString()));
        //kích hoạt thread i chạy và truyền các tham số cần thiết cho nó
        t.Start (new Params(t, i*N/cnt, (i+1)*N/cnt,i));
    }
    //bản thân thread cha cũng tính N/cnt hàng cuối của ma trận tích
    TinhTich(new Params(null, (cnt-1)*N/cnt, N, cnt-1));
    //chờ đợi các thread con hoàn thành
    for (i = 0; i < cnt-1; i++)
        while (stateLst[i] == 0); //cho
    }
    //ghi nhận thời điểm kết thúc tính tích
    DateTime t2 = DateTime.Now;
    System.TimeSpan diff;
    //hiển thị độ ưu tiên hiện hành của chương trình
    lbKetqua.Items.Add("Ung dụng đa chạy voi quyền " + myPrio.ToString());
    //hiển thị thời gian tính của từng thread con
    for (i = 0; i < cnt - 1; i++)
    {
        diff = dateLst[i];
        lbKetqua.Items.Add(String.Format("Thread {0:d} chạy ton {1:d2} phut {2:d2} giây
{3:d3} ms", i, diff.Minutes, diff.Seconds, diff.Milliseconds));
    }
    diff = t2.Subtract(t1);
    //hiển thị thời gian tổng cộng để tính tích
    lbKetqua.Items.Add(String.Format("{0:d} threads ==> Thời gian chạy là {1:d2} phut
{2:d2} giây {3:d3} ms", cnt, diff.Minutes, diff.Seconds, diff.Milliseconds));
}

```

13. Dời chuột về đầu class Form1 rồi thêm lệnh định nghĩa các kiểu dữ liệu, các thuộc tính, các hàm dịch vụ cần dùng như sau :

```

class Params //định nghĩa class đối tượng chứa các tham số cần truyền cho thread con
{
    public Thread t;        //đối tượng quản lý thread tương ứng
    public int sr;          //hàng bắt đầu tính
    public int er;          //hàng kết thúc tính +1
    public int id;          //chỉ số thread trong danh sách quản lý
    public Params(Thread t, int s, int e, int i)
    {
        this.t = t; sr = s;
        er = e;
    }
}

```

```

        id = i;
    }
};

//định nghĩa 3 biến ma trận
double[,] A;
double[,] B;
double[,] C;
//định nghĩa biến chứa số hàng/cột của ma trận
int N;
//định nghĩa danh sách trạng thái thi hành của các thread con
int[] stateLst = new int [20];
//định nghĩa danh sách thời gian thi hành của các thread con
System.TimeSpan[] dateLst = new System.TimeSpan[20];
//định nghĩa biến miêu tả quyền ưu tiên của chương trình
ProcessPriorityClass myPrio = ProcessPriorityClass.Normal;
Process MyProc;
//định nghĩa danh sách các quyền ưu tiên cho các thread
ThreadPriority[] tPrio = {
    ThreadPriority.Lowest, ThreadPriority.BelowNormal, ThreadPriority.Normal,
    ThreadPriority.AboveNormal, ThreadPriority.Highest};

//định nghĩa hàm các hàng ma trận tích theo yêu cầu trong tham số
void TinhTich (object obj) {
    DateTime t1 = DateTime.Now;
    Params p = (Params)obj;
    int h, c, k;
    for (h = p.sr; h < p.er; h++)
        for (c = 0; c < N; c++)
        {
            double s = 0;
            for (k = 0; k < N; k++)
                s = s + A[h, k] * B[k, c];
            C[h, c] = s;
        }
    //ghi nhận đã hoàn thành nhiệm vụ
    stateLst[p.id] = 1;
    //ghi nhận thời gian tính
    dateLst[p.id] = DateTime.Now.Subtract(t1);
}

```

14. Tìm hàm khởi tạo Form1 rồi thêm các lệnh khởi tạo các ma trận A và B như sau :

```

public Form1()
{
    InitializeComponent();
    //thêm vào các lệnh khởi tạo sau đây
    lbKetqua.Items.Clear();
    //khởi tạo các ma trận A, B, C
    N = 1000;
    A = new double[N,N];
    B = new double[N,N];
    C = new double[N,N];
    int h, c;

```

```
for (h = 0 ; h < N; h++)  
    for (c = 0; c < N; c++)  
        A[h,c] = B[h,c] = c;  
}
```

15. Dời chuột về đầu file mã nguồn Form1 rồi thêm lệnh using như sau :

using System.Diagnostics;

using System.Threading;

16. Chọn menu Debug.Start Debugging để dịch và chạy thử ứng dụng.

17. Khi Form chương trình hiển thị, hãy nhập số 1 vào Textbox số thread cần chạy, click chuột vào button "Tính tích 2 ma trận", chờ xem thời gian chạy tốn bao lâu rồi ghi nhận.

18. Lặp lại bước 17 với số thread tăng dần, rồi ghi nhận lại thời gian chạy của từng trường hợp.

19. So sánh thời gian chạy để đánh giá độ hiệu quả của multi-thread với số lượng thread khác nhau.