

MÔN: HỆ ĐIỀU HÀNH

Bài thực hành số 5.1: Viết phần mềm giải quyết tương tranh giữa các thread

I. Mục tiêu

- Giúp SV củng cố kiến thức về phương pháp dùng semaphore nhị phân để giải quyết loại trừ tương hỗ giữa các thread khi chúng đồng thời truy xuất tài nguyên dùng chung.
- Giúp SV làm quen với việc dùng class Mutex của namespace System.Threading, sự hiện thực semaphore nhị phân của môi trường .NET, để loại trừ tương hỗ giữa các thread khi chúng cùng truy xuất vào tài nguyên dùng chung.


II. Nội dung

- Tìm cách giải quyết tương tranh từng cell màn hình khi các thread cùng hiển thị icon của mình lên cell đó.
- Dùng class Mutex để quản lý từng cell hiển thị, thread nào muốn hiển thị cell nào thì phải gọi tác vụ WaitOne() của Mutex của cell tương ứng (down(s)). Khi không dùng cell cũ nữa, thread phải gọi tác vụ ReleaseMutex() của Mutex của cell đó để giải phóng nó cho thread khác dùng (up(s)).

III. Chuẩn đầu ra

- Sinh viên nắm vững và sử dụng thành thạo class Thread để quản lý thread.
- Sinh viên nắm vững vấn đề tương tranh giữa các thread khi chúng truy xuất tài nguyên dùng chung.
- Sinh viên nắm vững và sử dụng thành thạo class Mutex để loại trừ tương hỗ giữa các thread khi chúng cùng truy xuất vào tài nguyên dùng chung.

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. ThreadDemo3), 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. Bài thực hành này không thiết kế form mà chỉ viết code cho chương trình vì form được hiệu chỉnh kích thước động và nội dung hiển thị trong form cũng được hiệu chỉnh động bởi các thread đang chạy.
4. Chọn Form, cửa sổ thuộc tính của nó sẽ hiển thị, click icon  để hiển thị danh sách các sự kiện của Form, duyệt tìm sự kiện Load, ấn kép chuột vào comboBox bên phải sự kiện Load 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 Form1_Load(object sender, EventArgs e) {
    //tạo đối tượng quản lý việc truy xuất tài nguyên trong project
    System.Reflection.Assembly myAssembly =
System.Reflection.Assembly.GetExecutingAssembly();
    threadLst = new MyThread[26];
    int i;
    //tạo ma trận semaphore Mutex để bảo vệ các cell màn hình
    mutList = new Mutex[yMax, xMax];
    int h, cot;
    for (h = 0; h < yMax; h++)
```

```

    for (cot = 0; cot < xMax; cot++)
        mutList[h, cot] = new Mutex();
    //Lắp thiết lập trạng thái ban đầu cho 26 thread từ A-Z
    for (i = 0; i < 26; i++)
    {
        threadLst[i] = new MyThread(rnd, xMax, yMax);
        threadLst[i].stop = threadLst[i].suspend = threadLst[i].start = false;
        char c = (char)(i + 65);
        //đọc bitmap miêu tả thread c từ file
        myStream =
myAssembly.GetManifestResourceStream("ThreadDemo3.Resources.Image" + c.ToString()
+ ".bmp");
        threadLst[i].Pic = new Bitmap(myStream);
        threadLst[i].Xmax = 25;
        threadLst[i].Ymax = 20;
    }
    ClientSize = new Size(25 * 30, 20 * 30);
    this.Location = new Point(0, 0);
    this.BackColor = Color.Black;
}

```

5. Tạo hàm xử lý sự kiện KeyDown cho Form. Cửa sổ mã nguồn sẽ hiển thị khung sườn của hàm vừa được tạo với thân rỗng, viết thân cho hàm này như sau :

```

//hàm xử lý gõ phím của user để quản lý thread
private void Form1_KeyDown(object sender, KeyEventArgs e) {
    //xác định mã phím ấn, nếu không phải từ A-Z thì phớt lờ
    int newch = e.KeyValue;
    if (newch < 0x41 || newch > 0x5a) return;
    //xác định chức năng mà user muốn và thực hiện
    if (e.Control && e.Shift)
    { //kill thread
        //dừng Thread
        threadLst[newch - 65].start = false;
    }
    else if (e.Control && e.Alt)
    { //tạm dừng thread
        if (threadLst[newch - 65].start && !threadLst[newch - 65].suspend)
        {
            threadLst[newch - 65].t.Suspend();
            threadLst[newch - 65].suspend = true;
        }
    }
    else if (e.Alt)
    { //cho thread chạy lại
        if (threadLst[newch - 65].start && threadLst[newch - 65].suspend)
        {
            threadLst[newch - 65].t.Resume();
            threadLst[newch - 65].suspend = false;
        }
    }
    else if (e.Shift)

```

```

{ //tăng độ ưu tiên tối đa
  threadLst[newch - 65].t.Priority = ThreadPriority.Highest;
  MessageBox.Show(threadLst[newch - 65].t.Priority.ToString());
}
else if (e.Control)
{ //giảm độ ưu tiên tối thiểu
  threadLst[newch - 65].t.Priority = ThreadPriority.Lowest;
  MessageBox.Show(threadLst[newch - 65].t.Priority.ToString());
}
else
{ //tạo mới thread và bắt đầu chạy
  if (!threadLst[newch - 65].start)
  {
    threadLst[newch - 65].start = true;
    threadLst[newch - 65].suspend = false;
    threadLst[newch - 65].t = new Thread(new
ParameterizedThreadStart(Running));
    if (newch == 65) threadLst[newch - 65].t.Priority = ThreadPriority.Highest;
    else threadLst[newch - 65].t.Priority = ThreadPriority.Lowest;
    threadLst[newch - 65].t.Start(threadLst[newch - 65]);
  }
}
}
}

```

6. Tạo hàm xử lý sự kiện FormClosed cho Form. 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 Form1_FormClosed(object sender, FormClosedEventArgs e) {
  int i;
  //lặp kiểm tra xem có thread con nào còn chạy không, nếu có thì xóa nó
  for (i = 0; i < 26; i++)
  {
    if (threadLst[i].start) {
      threadLst[i].start = false;
      while (!threadLst[i].stop) ;
    }
  }
}

```

7. 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 :

```

//định nghĩa các thuộc tính cần dùng
Stream myStream;
Mutex[,] mutList;
MyThread[] threadLst;
const int xCell = 30;
const int yCell = 30;
const int xMax = 25;
const int yMax = 20;
//tạo đối tượng sinh số ngẫu nhiên
public Random rnd = new Random();

//định nghĩa hàm giả lập hành vi của thread
void MySleep(long count)

```

```

{
    long i, j, k = 0;
    for (i = 0; i < count; i++)
        for (j = 0; j < 64000; j++) k = k + 1;
}

//định nghĩa hàm mà mỗi thread sẽ chạy
void Running(object obj)
{
    //ép kiểu tham số về MyThread theo yêu cầu xử lý
    MyThread p = (MyThread)obj;
    //tạo đối tượng vẽ
    Graphics g = this.CreateGraphics();
    //tạo chổi màu đen để xóa cell cũ
    Brush brush = new SolidBrush(Color.FromArgb(0, 0, 0));
    //xin khóa truy xuất cell (x1,y1)
    mutList[p.Pos.Y, p.Pos.X].WaitOne();
    int x1, y1;
    int x2, y2;
    int x, y;
    bool kq=true;
    try
    {
        while (p.start)
        { //lặp trong khi chưa có yêu cầu kết thúc
            //xác định tọa độ hiện hành của thread
            x1 = p.Pos.X; y1 = p.Pos.Y;
            //hiển thị logo của thread ở (x1,y1)
            g.DrawImage(p.Pic, xCell * x1, yCell * y1);
            Color c = p.Pic.GetPixel(1,1);
            int yR, yG, yB;
            if (c.R > 128) yR = 0; else yR = 255;
            if (c.G > 128) yG = 0; else yG = 255;
            if (c.B > 128) yB = 0; else yB = 255;
            Pen pen = new Pen(Color.FromArgb(yR, yG, yB), 2);
            if (p.tx >= 0 && p.ty >= 0) { //hiện mũi tên góc dưới phải
                x = xCell * x1 + xCell - 2;
                y = yCell * y1 + yCell - 2;
                g.DrawLine(pen, x, y, x - 10, y);
                g.DrawLine(pen, x, y, x, y - 10);
            } else if (p.tx >= 0 && p.ty < 0) { //hiện mũi tên góc trên phải
                x = xCell * x1 + xCell - 2;
                y = yCell * y1 + 2;
                g.DrawLine(pen, x, y, x - 10, y);
                g.DrawLine(pen, x, y, x, y + 10);
            } else if (p.tx < 0 && p.ty >= 0) { //hiện mũi tên góc dưới trái
                x = xCell * x1 + 2;
                y = yCell * y1 + yCell - 2;
                g.DrawLine(pen, x, y, x + 10, y);
                g.DrawLine(pen, x, y, x, y - 10);
            }
        }
    }
}

```

```

    } else { //hiện mũi tên góc trên trái
        x = xCell * x1 + 2;
        y = yCell * y1 + 2;
        g.DrawLine(pen, x, y, x + 10, y);
        g.DrawLine(pen, x, y, x, y + 10);
    }
    //giả lập thực hiện công việc của thread tốn 500ms
    MySleep(500);
    //xác định vị trí mới của thread
    p.HieuchinhVitri();
    x2 = p.Pos.X; y2 = p.Pos.Y;
    //xin khóa truy xuất cell (x2,y2)
    mutList[y2, x2].WaitOne();
    // Xóa vị trí cũ
    g.FillRectangle(brush, xCell * x1, yCell * y1, xCell, yCell);
    //trả cell (x1,y1) cho các thread khác truy xuất
    mutList[y1, x1].ReleaseMutex();
}
}
catch (Exception e) { p.t.Abort(); }
//dọn dẹp thread trước khi ngừng
x1 = p.Pos.X; y1 = p.Pos.Y;
g.FillRectangle(brush, xCell * x1, yCell * y1, xCell, yCell);
//trả cell (x1,y1) cho các thread khác truy xuất
mutList[y1, x1].ReleaseMutex();
// dừng Thread
p.stop = true;
p.t.Abort();
}

```

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

```

using System.Threading;
using System.Resources;
using System.IO;

```

9. Ấn phải chuột vào phần tử gốc của cây Project trong cửa sổ Solution Explorer, chọn option Add.Class, đặt tên là MyThread.cs để tạo ra file đặc tả class chứa các tham số phục vụ cho từng thread con chạy. Khi cửa sổ hiển thị mã nguồn của class MyThread hiển thị, viết code cho class này như sau :

```

class MyThread {
    const double PI = 3.1416;
    public Thread t; //tham khảo đến thread hiện hành
    public Boolean start; //trạng thái Start của thread
    public Boolean stop; //trạng thái Stop của thread
    public Boolean suspend; //trạng thái Suspend của thread
    public Boolean WaitOne = false; //trạng thái chờ truy xuất cell
    public Bitmap Pic; //icon miêu tả thread
    internal int Xmax; //độ rộng vùng chạy của thread
    internal int Ymax; //độ cao vùng chạy của thread
    public Point Pos; //vị trí của thread trong vùng chạy
    double dblGocChay; //góc chạy của thread
    internal double tx, ty; //bước tăng theo x và y
}

```

```

//hàm khởi tạo các thông số của thread
public MyThread(Random rnd, int xMax, int yMax)
{
    Xmax = xMax; Ymax = yMax;
    Pos.X = (int)(rnd.Next(0, Xmax));
    Pos.Y = (int)(rnd.Next(0, Ymax));
    dblGocChay = ChinhGocChay(rnd.Next(0, 360));
}

//=====
//Hiệu chỉnh góc chạy của thread
//để tránh các trường hợp thread chạy thẳng đứng hay ngang
//=====
double ChinhGocChay(double dblGocChay)
{
    double goc = dblGocChay;
    if (0 <= goc && goc < 90) return 45;
    if (90 <= goc && goc < 180) return 135;
    if (180 <= goc && goc < 270) return 225;
    if (270 <= goc) return 315;
    return goc;
}

//=====
//Tính góc phản xạ mới khi thread đụng thành đứng (bên trái hay phải).
//=====
double DoiGocChayX(double dblGocChay)
{
    double goc;
    if (dblGocChay > 0 && dblGocChay < 180) goc = 180 - dblGocChay;
    else goc = 180 + 360 - dblGocChay;
    return ChinhGocChay(goc);
}

//=====
//Tính góc phản xạ mới khi thread đụng thành ngang (trên hay dưới).
//=====
double DoiGocChayY(double dblGocChay)
{
    return ChinhGocChay(360 - dblGocChay);
}

//=====
//Hiệu chỉnh vị trí của thread
//=====
public void HieuchinhVetri()
{
    int x, y;
    x = Pos.X;
    y = Pos.Y;
    if (x == 0 || x == Xmax - 1 || y == 0 || y == Ymax - 1)
    {
        //icon đụng thành ngang hay dọc -> thay đổi góc chạy
    }
}

```

```

        if (x == 0 || x == Xmax - 1)
        {
            dblGocChay = DoiGocChayX(dblGocChay);
        }
        else if (y == 0 || y == Ymax - 1)
            dblGocChay = DoiGocChayY(dblGocChay);
    }
    //Hiệu chỉnh tọa độ x của thread
    tx = 2 * Math.Cos(dblGocChay * PI / 180);
    x = x + (int)tx;
    if (x < 0)
    {
        x = 0;
    }
    else if (x >= Xmax)
    {
        x = Xmax - 1;
    }
    //Hiệu chỉnh tọa độ y của thread
    ty = 2 * Math.Sin(dblGocChay * PI / 180);
    y = y + (int)ty;
    if (y < 0)
    {
        y = 0;
    }
    else if (y >= Ymax)
    {
        y = Ymax - 1;
    }
    //chỉnh góc chạy khi đụng 1 trong 4 góc
    if (x == 0 && y == 0) //góc trên trái
        ChinhGocChay(dblGocChay + 45);
    else if (x == 0 && y == Ymax - 1) //góc dưới trái
        ChinhGocChay(dblGocChay + 45);
    else if (x == Xmax - 1 && y == 0) //góc trên phải
        ChinhGocChay(dblGocChay + 45);
    else if (x == Xmax - 1 && y == Ymax - 1) //góc dưới phải
        ChinhGocChay(dblGocChay + 45);
    //Lưu vị trí mới
    Pos.X = (int)x;
    Pos.Y = (int)y;
}
}

```

10. Dời chuột về đầu file mã nguồn của class MyThread rồi thêm lệnh using như sau :

```
using System.Threading;
```

```
using System.Drawing;
```

11. Ấn phải chuột vào phần tử gốc của cây Project trong cửa sổ Solution Explorer, chọn option Add.New Folder để thêm folder với tên là Resources, ta sẽ dùng folder này chứa các file bitmap được dùng trong chương trình.

12. Ấn phải chuột vào folder Resources, chọn option Existing Items, duyệt chọn 26 file bitmap miêu tả 26 icon A-Z để add chúng vào folder Resources.
12. Chọn 26 mục vừa add vào folder Resources để hiển thị của sổ thuộc tính chúng của chúng, hiệu chỉnh lại thuộc tính Build Action về giá trị mới là "Embedded Resource".
13. Chọn menu Debug.Start Debugging để dịch và chạy thử ứng dụng.
14. Khi Form chương trình hiển thị, hãy thực hiện gõ phím qui định như sau để quản lý các thread :
 - Ấn phím từ A-Z để kích hoạt chạy thread có tên tương ứng.
 - Ấn phím Ctrl-Alt-X để tạm dừng chạy thread X.
 - Ấn phím Alt-X để chạy tiếp thread X.
 - Ấn phím Shift-X để tăng độ ưu tiên chạy cho thread X.
 - Ấn phím Ctrl-X để giảm độ ưu tiên chạy cho thread X.
 - Ấn phím Ctrl-Shift-X để dừng và thoát thread X.

Khi số thread chạy tương đối nhiều, hãy quan sát hiện tượng icon của thread này tiến đến và đè icon của thread khác không còn nữa, bây giờ thread A phải chờ thread B nếu A muốn hiển thị cell mà B đang hiển thị. Hiện tượng dùng chờ này có thể gây ra deadlock, nghĩa là các thread có thể dùng chờ lẫn nhau và không có thread nào chạy tiếp được. Tóm lại deadlock là hậu quả của việc xử lý tương tranh giữa các thread. Hiện nay Windows và các HĐH phổ biến khác đều không xử lý deadlock tự động, do đó chương trình ứng dụng và/hoặc người dùng phải tự giải quyết lấy.