عنوان مقاله : چند رشته ای در C# بخش اول موضوع : برنامه نویسی تاریخ انتشار: زمستان ۸۷

چکیده:

همروندی *یک مشخصه سامانه های رایانه ای است که در آن اجرای چند پردازه بطور همزمان صورت می گیرد. پردازه ها ممکن است با هم در ارتباط باشند یا نتایج کار یک پردازه مورد استفاده پردازه دیگر قرار گیرد. یکی از روشهای اجرای همروندی استفاده از* ***چند رشته‌ای*** *میباشد. هرگاه بخواهیم در برنامه چند کار بطور همزمان اجرا شوند باید از سازوکار چند رشته ای استفاده کنیم.*

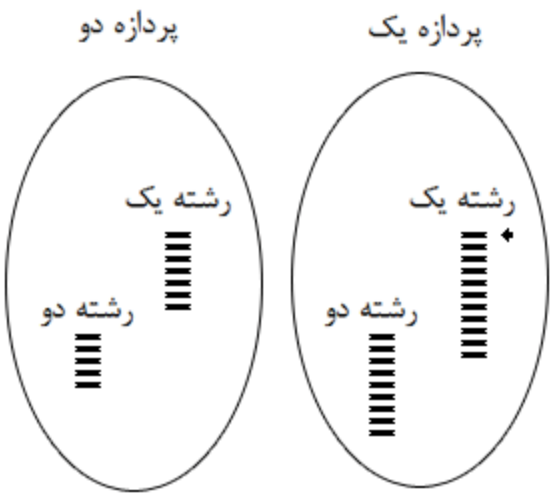
همروندی اجرای همزمان چند روند در سامانهء رایانه‌ای است. همروندی می تواند در سطوح مختلف مطرح شود. بطور معمول چهار سطح برای همروندی در نظر گرفته میشود. سطح اول همروندی در اجرای دستورات زبان ماشین است. سطح دوم اجرای همروندی در سطح دستورات برنامه است. سطح سوم همروندی در اجرای واحدهای برنامه است. و سطح چهارم اجرای همروندی در سطح برنامه است. مدیریت سطح چهارم کاملا بر عهده سامانه عامل است. و مدیریت سطح اول بر عهده سخت‌افزار می‌باشد. برای اجرای همزمان دستور العمل‌های زبان ماشین لازم است سخت افزار قابلیت اجرای موازی را داشته باشد. سطوح اول و چهارم از دید برنامه‌نویس پوشیده است. سطوح دوم و سوم در حوزه برنامه‌نویسی می‌باشند .

در بحث همروندی موضوع **همگام‌سازی** بسیار مورد توجه است. همگام سازی در دو حیطهء **رقابت** و **همکاری** مورد بررسی قرار می‌گیرد. رقابت به معنای آنست که دو یا چند پردازه بخواهند به طور همزمان به یک منبع دسترسی یابند و آن منبع در هر زمان بتواند فقط توسط یک پردازه مورد استفاده قرار گیرد. همکاری به معنای آنست که یک پردازه برای تکمیل عملیات خود منتظر نتیجهء کار یک پردازه دیگر باشد.همگام‌سازی سازماندهی و ترتیب اجرای پردازه‌ها را مشخص میکند.

پردازه‌های مختلف برای ارتباط با یکدیگر از سازوکارهایی نظیر استفاده از متغیرهای غیر محلی (non local variables) ، ارسال و دریافت پیام و ارسال پارامتر استفاده میکنند. همچنین برای مدیریت دسترسی به منابع انحصاری از سه روش *سمافور (Semaphore) ، مانیتور (Monitor) و ارسال پیام (Message Passing)* استفاده میشود

سمافور یک ساختار داده ای شامل یک صف و یک متغیر صحیح است و دو عمل استاندارد P و V روی آن تعریف شده است. سمافور اولین بار توسط دانشمندی به نام دایکسترا مطرح گردید. از سمافور برای کنترل دسترسی انحصاری به منابع استفاده می‌شود.

هر پردازه به مجموعه‌ای از رشته‌های اجرایی تقسیم می‌گردد. رشته‌های اجرایی در فضای Heap مشترک هستند و غالبا بر روی مجموعه مشترکی از داده‌ها پردازش می‌کنند.



شکل یک - پردازنده به ترتیب دستورات را از رشته‌های مختلف اجرا می‌کند

رشته‌ها در موارد متعددی بکار می‌روند یکی از کاربردهای اصلی آنها شرایطی است که باید یک کار طولانی (مثلا جستجو) روی پایگاه داده‌ها به انجام رسد. در این حالت اگر برنامه چند رشته‌ای نباشد، به دلیل طولانی بودن پردازش، رویدادهای صفحه کلید و موس پردازش نمی‌شوند. همچنین فرم‌ها ترسیم نمی‌گردنند و برنامه هنگ می‌کند و به درخواست‌های کاربر پاسخ نمی‌دهد. در این حالت برنامه non responsive شده و کاربر امکان متوقف کردن برنامه را نخواهد داشت. از آنجایی که برنامه‌های کاربردی باید همیشه با کاربر اندرکنش (interaction) داشته باشند، لازم است عمل جستجو یا هر عمل زمان‌بر دیگر در یک رشته مستقل اجرا شود و رشته اصلی به درخواست‌های کاربر پاسخ دهد در این حالت کاربر می‌تواند اجرای رشته اول را متوقف کند.

نکته مهم:

در استفاده از رشته‌ها نباید زیاده‌روی کرد. بکارگیری بیش از حد رشته‌ها باعث پیچیدگی برنامه شده و فرآیند توسعه و رفع عیب را طولانی و دشوار می‌کند.

طرح مسئله:

می‌خواهیم یک برنامه ساده بنویسیم که در آن یک کار طولانی مانند شمارش از یک تا دویست ملیون در یک رشته جداگانه اجرا شود. در این حالت برنامه اصلی همچنان در تعامل (interaction) با کاربر می‌ماند. همچنین برنامه می‌تواند به درخواست کاربر برای نمایش پیام پاسخ دهد.

کاربر با زدن کلید شمارش عمل شمارش را در یک رشته جدا به نام (t) آغاز می‌کند. رشته اجرایی متغیر خاصی بنام flag را کنترل می‌کند، هرگاه کاربر کلید انصراف را بزند مقدار متغیر کنترلی false شده و رشته عمل شمارش را متوقف می‌کند.

یک زمان‌سنج در برنامه استفاده شده که در بازه‌های زمانی مشخص مقدار پیشرفت کار را از متغیر استتیک L1 دریافت کرده و میله پیشرفت (progress bar) را بروزرسانی می‌کند. پس از پایان عمل شمارش دوباره دکمه شمارش فعال می‌شود.

شکل دو - برنامه حین انجام کار به درخواست کاربر پاسخ می‌دهد.

کد برنامه:

```
static void Search()
  {
      while ((L1 < MAX)&&(flag))
      {
          L1++;
      }
      flag = true;
  }

  private void btnShowMsg_Click(object sender, EventArgs e)
  {
      MessageBox.Show("برنامه به درخواست کاربر پاسخ میدهد ");

  private void btnLengthyWork_Click(object sender, EventArgs e)
  {
      Thread t = new Thread(Search);
      if (iStatus)
      {
          flag = true;
          L1 = 0;
```

```
        DisableSearch();
        timer1.Enabled = true;
        timer1.Start();
        t.Start();
    }
    else
    {
        EnableSearch();
        timer1.Enabled = false;
        timer1.Stop();
        flag = false;
    }
}

private void timer1_Tick(object sender, EventArgs e)
{
    double D1;
    D1 = (double) L1;

    prsBar1.Value = (int)(Math.Round(100 * (D1 / MAX)));
    lblMsg.Text = "در حال شمارش رکورد " + L1.ToString() + "  " + "از " + MAX.ToString();
```

```
    if (L1 == MAX)
    {
        L1 = 0;
        EnableSearch();
        timer1.Enabled = false;
        timer1.Stop();
    }
}

private void EnableSearch()
{
    L1 = 0;
    lblMsg.Visible = false;
    prsBar1.Visible = false;
    btnLengthyWork.Text = "شمارش";
    iStatus = true;
}

private void DisableSearch()
{
    lblMsg.Visible = true;
    prsBar1.Visible = true;
```

```
        prsBar1.Value = 0;

        btnLengthyWork.Text = "انصراف";

        iStatus = false;

    }

}
```