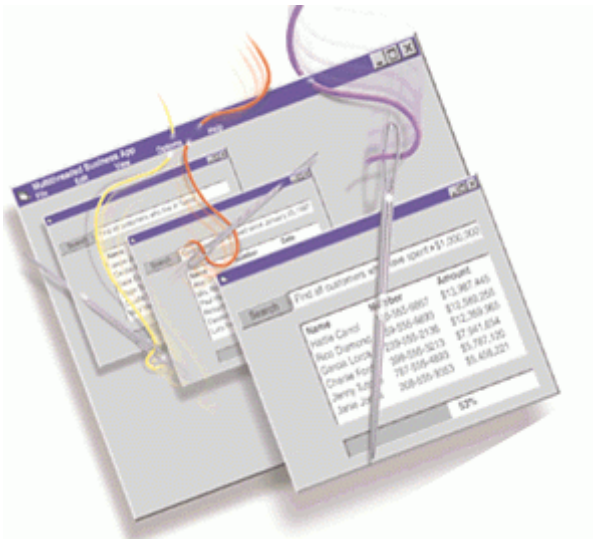
چند ریسمانی[1] در ویژوال بیسیک



۱- چند کارگی[2] چیست ؟

از آنجایی که ویندوز سیستم عاملی است ۳۲ بیتی ، بیش از یک کار[3] را در یک زمان در آن می توان انجام داد. برای مثال می توانید به سی دی موسیقی گوش دهید و در همان زمان چند فایل را هم کپی نمایید. اما آنها در یک ماشین تک پردازنده دریک لحظه با هم اجرا نمی شوند ، بلکه ویندوز کنترل پردازنده را بسیار سریع بین آنها تعویض می نماید. در این حالت این طور به نظر می رسد که کارها همزمان انجام می شوند. فقط برروی کامپیوترهای چند پردازنده چندکارگی واقعی امکان پذیر است و برنامه ها به موازات یکدیگر اجرا می شوند.

۲- چند ریسمانی چیست ؟

ریسمان در اینجا یک **Sub** یا **Function** است در **یک** برنامه. چند ریسمانی توانایی اجرای بیش از یک **Sub** یا **Function** در یک زمان ودر یک برنامه می باشد. تعداد ریسمانها در یک برنامه فقط توسط میزان حافظه‌ی مهیا محدود می شود.

---

[1] **Multithreading**
[2] **Multitasking**
[3] **Task**

۳- چگونه می توان چند ریسمانی را در ویژوال بیسیک پیاده نمود ؟

برای انجام اینکار باید از تعدادی از توابع **API** ویندوزبه شرح زیر استفاده نمود:

۱- CreateThread :

این تابع ریسمان جدیدی را در برنامه پدید می آورد .

```
Declare Function CreateThread Lib "kernel32" ( _
      ByVal lpThreadAttributes As Any, _
      ByVal dwStackSize As Long, _
      ByVal lpStartAddress As Long, _
      lpParameter As Any, _
      ByVal dwCreationFlags As Long, _
      lpThreadID As Long) As Long
```

شرح آرگومانهای تابع :

lpThreadAttributes صفات امنیتی ریسمان می باشد و بجای آن ByVal 0& قرار می گیرد.

dwStackSize میزان حافظه‌ی پشته لازم برای ریسمان می باشد و بجای آن ByVal 0& قرارمی گیرد.

lpStartAddress مهمترین آرگومان این تابع می باشد و به Windows می گوید که ریسمان ایجاد شده کدام تابع را باید اجرا کند. برای استفاده از آن باید از اپراتور AddressOf استفاده کرد. این اپراتور **فقط** با **public** functions در **public modules** بکارمی رود. بنابراین فقط کافی است تابعی را که می خواهید بعنوان ریسمان بکار ببرید در یک ماژول عمومی قرار دهید و سپس بجای این آرگومان قرار دهید :

```
AddressOf YourFunction
```

dwCreationFlags یک پرچم را برای ریسمان تنظیم می کند. در مشاهده گر **API** ویژوال‌بیسیک ثابت های CREATE_* را جستجو کنید. جالبترین پرچمی که در اینجا وجود دارد CREATE_SUSPENDED می باشد. که با تنظیم این پرچم ریسمان غیر فعال می شود. اگر به آن احتیاجی ندارید می توانید از ByVal 0& استفاده کنید .

lpThreadID که یک پارامتر ByRef می باشد بیانگر ID ریسمان ایجاد شده می باشد.

خروجی این تابع handle ریسمان ایجاد شده می باشد مانند handle فرمها (hWnd). که اجازه اعمال کنترل روی ریسمان را مهیا می سازد . اگر تابع موفق به ایجاد ریسمان نشود خروجی آن صفر خواهد بود .

۲- SetThreadPriority و GetThreadPriority :

```
Declare Function SetThreadPriority Lib "kernel32" ( _
      ByVal hThread As Long, ByVal nPriority As Long) As Long
```

```
Declare Function GetThreadPriority Lib "kernel32" Alias _
       "GetThreadPriority" (ByVal hThread As Long) As Long
```

تابع **SetThreadPriority** حق تقدم ریسمان مشخص شده را تنظیم می کند.

شرح آرگومانهای تابع :

hThread دستگیره (handle) ریسمان مربوطه می باشد . برای مثال می توانید از خروجی تابع قبل (CreateThread) استفاده کنید.

nPriority حق تقدم جدید ریسمان می باشد و یکی از مقادیر زیر را دارا است :

```
Const THREAD_BASE_PRIORITY_IDLE = -15
Const THREAD_BASE_PRIORITY_LOWRT = 15
Const THREAD_BASE_PRIORITY_MAX = 2
Const THREAD_BASE_PRIORITY_MIN = -2
Const MAXLONG = &H7FFFFFFF
Const THREAD_PRIORITY_HIGHEST = THREAD_BASE_PRIORITY_MAX
Const THREAD_PRIORITY_LOWEST = THREAD_BASE_PRIORITY_MIN
Const THREAD_PRIORITY_ABOVE_NORMAL = (THREAD_PRIORITY_HIGHEST - 1)
Const THREAD_PRIORITY_BELOW_NORMAL = (THREAD_PRIORITY_LOWEST + 1)
Const THREAD_PRIORITY_ERROR_RETURN = (MAXLONG)
Const THREAD_PRIORITY_IDLE = THREAD_BASE_PRIORITY_IDLE
Const THREAD_PRIORITY_NORMAL = 0
Const THREAD_PRIORITY_TIME_CRITICAL = THREAD_BASE_PRIORITY_LOWRT
```

و برای بدست آوردن حق تقدم ریسمان مربوطه از تابع **GetThreadPriority** استفاده می شود.

۳- SuspendThread و ResumeThread :

```
Declare Function SuspendThread Lib "kernel32" ( _
                                ByVal hThread As Long) As Long

Declare Function ResumeThread Lib "kernel32" ( _
                                ByVal hThread As Long) As Long
```

از این دو تابع برای فعال سازی و غیر فعال سازی یک ریسمان استفاده می شود.

hThread دستگیره (handle) ریسمان مربوطه می باشد

۴- TerminateThread :

```
Declare Function TerminateThread Lib "kernel32" ( _
      ByVal hThread As Long, ByVal dwExitCode As Long) As Long
```

این تابع بطور کامل یک ریسمان را متوقف می کند.

شرح آرگومانهای تابع :
hThread دستگیره (handle) ریسمان مربوطه می باشد .
dwExitCode کد خروج می باشد و به آن نیازی نیست و بجای آن می توانید از ByVal 0& استفاده کنید .

برای اینکه بتوان از این توابع حداکثر استفاده را برد کلاس زیر نوشته شده است :

```
'clsThreading:
'Simple class that allows you to implement multithreading
'in your app
'API Declarations
'Creates a new thread
Private Declare Function CreateThread Lib "kernel32" ( _
ByVal lpThreadAttributes As Any, ByVal dwStackSize As Long, _
ByVal lpStartAddress As Long, lpParameter As Any, _
 ByVal dwCreationFlags As Long, lpThreadID As Long) As Long
'Terminates a thread
Private Declare Function TerminateThread Lib "kernel32" _
 (ByVal hThread As Long, ByVal dwExitCode As Long) As Long
'Sets the priority of a thread
Private Declare Function SetThreadPriority Lib "kernel32" _
 (ByVal hThread As Long, ByVal nPriority As Long) As Long
'Returns the proirity of a thread
Private Declare Function GetThreadPriority Lib "kernel32" _
 (ByVal hThread As Long) As Long
'Enables a disabled Thread
Private Declare Function ResumeThread Lib "kernel32" _
 (ByVal hThread As Long) As Long
'Disables a thread
Private Declare Function SuspendThread Lib "kernel32" _
 (ByVal hThread As Long) As Long
'Returns the handle of the current thread
Private Declare Function GetCurrentThread Lib "kernel32"  _
 () As Long
'Returns the ID of the current thread
Private Declare Function GetCurrentThreadId Lib "kernel32" _
 () As Long

'Consts
Private Const MAXLONG = &H7FFFFFFF

'Thread priority consts
Private Const THREAD_BASE_PRIORITY_IDLE = -15
Private Const THREAD_BASE_PRIORITY_LOWRT = 15
Private Const THREAD_BASE_PRIORITY_MAX = 2
Private Const THREAD_BASE_PRIORITY_MIN = -2
Private Const THREAD_PRIORITY_HIGHEST = THREAD_BASE_PRIORITY_MAX
Private Const THREAD_PRIORITY_LOWEST = THREAD_BASE_PRIORITY_MIN
Private Const THREAD_PRIORITY_ABOVE_NORMAL = _
 (THREAD_PRIORITY_HIGHEST - 1)
```

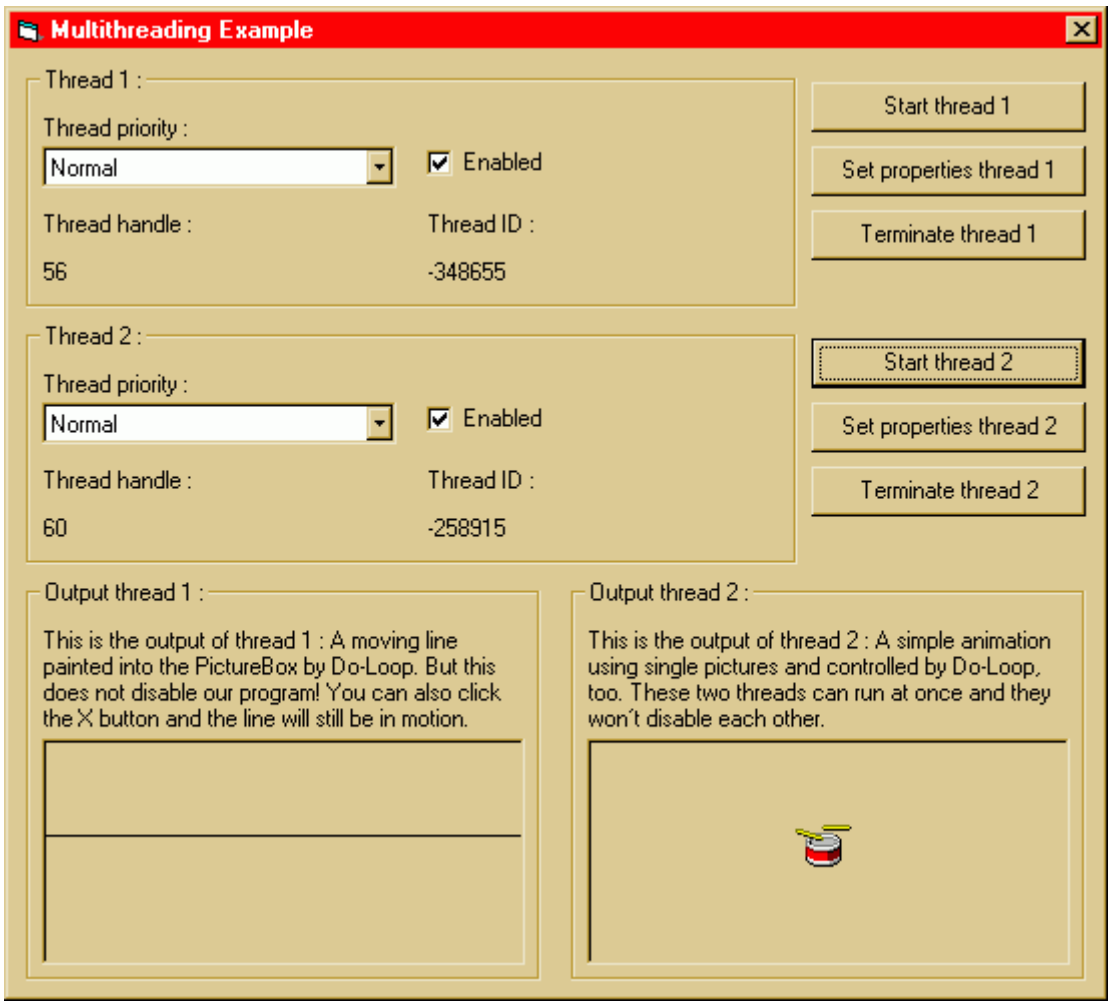
```
Private Const THREAD_PRIORITY_BELOW_NORMAL = _
 (THREAD_PRIORITY_LOWEST + 1)
Private Const THREAD_PRIORITY_ERROR_RETURN = (MAXLONG)
Private Const THREAD_PRIORITY_IDLE = THREAD_BASE_PRIORITY_IDLE
Private Const THREAD_PRIORITY_NORMAL = 0
Private Const THREAD_PRIORITY_TIME_CRITICAL =
THREAD_BASE_PRIORITY_LOWRT

'Thread creation flags
Private Const CREATE_ALWAYS = 2
Private Const CREATE_NEW = 1
Private Const CREATE_NEW_CONSOLE = &H10
Private Const CREATE_NEW_PROCESS_GROUP = &H200
Private Const CREATE_NO_WINDOW = &H8000000
Private Const CREATE_PROCESS_DEBUG_EVENT = 3
Private Const CREATE_SUSPENDED = &H4
Private Const CREATE_THREAD_DEBUG_EVENT = 2

'Types and Enums
Public Enum ThreadPriority
    tpLowest = THREAD_PRIORITY_LOWEST
    tpBelowNormal = THREAD_PRIORITY_BELOW_NORMAL
    tpNormal = THREAD_PRIORITY_NORMAL
    tpAboveNormal = THREAD_PRIORITY_ABOVE_NORMAL
    tpHighest = THREAD_PRIORITY_HIGHEST
End Enum

'Vars
Private mThreadHandle As Long
Private mThreadID As Long
Private mPriority As Long
Private mEnabled As Boolean
Private mCreated As Boolean

Public Function CreateNewThread(ByVal cFunction As Long, _
 Optional ByVal cPriority As Long = tpNormal, _
 Optional ByVal cEnabled As Boolean = True)
    'Creates a new Thread
    Dim mHandle As Long
    Dim CreationFlags As Long
    Dim lpThreadID As Long

    'Look if the thread has already been created
    If mCreated = True Then Exit Function

    'Look if the thread should be enabled
    If cEnabled = True Then
        CreationFlags = 0
    Else
        'Create a disabled thread, can be enabled later with the
        ''Enabled' property
        CreationFlags = CREATE_SUSPENDED
    End If

    'The CreateThread Function returns the handle of the created
    'thread;
    'if the handle is 0, it failed creating the thread
    mHandle = CreateThread(ByVal 0&, ByVal 0&, cFunction, _
                           ByVal 0&, CreationFlags, lpThreadID)
```

```
    If mHandle = 0 Then 'Failed creating the thread
        'Insert your own error handling
        'Debug.Print "InitializeThread Function in clsThreading
        'failed creating a new thread"
    Else
        mThreadHandle = mHandle
        mThreadID = lpThreadID
        mCreated = True
    End If
End Function

Public Function TerminateCurrentThread()
    'Terminates the current thread

    'Ignore errors to prevent crashing if no thread has been created
    On Error Resume Next
    'Terminate the thread to prevent crashing if the app is closed
    'and the thread is still running (dangerous!)
    Call TerminateThread(mThreadHandle, ByVal 0&)
    mCreated = False
End Function

Public Property Get ThreadHandle() As Long
    'Returns the Handle of the current Thread
    ThreadHandle = mThreadHandle
End Property

Public Property Get ThreadID() As Long
    'Returns the ID of the current thread
    ThreadID = mThreadID
End Property

Public Property Get Priority() As Long
    'Returns a long value because the thread might have other
    ' priorities
    'than our five in the enum

    'Ignore errors to prevent crashing if no thread has been created
    On Error Resume Next
    Priority = GetThreadPriority(mThreadHandle)
End Property

Public Property Let Priority(ByVal tmpValue As Long)
    'Sets the Thread Priority of the actual thread
    mPriority = tmpValue
    Call SetThreadPriority(mThreadHandle, tmpValue)
End Property

Public Property Get Enabled() As Boolean
    'Returns whether the Thread is enabled or not
    Enabled = mEnabled
End Property

Public Property Let Enabled(ByVal tmpValue As Boolean)
    'Enables/Disables the Thread

    'Ignore errors to prevent crashing if no thread has been created
    On Error Resume Next
```

```
    If tmpValue = True Then
        'Enable the thread
        Call ResumeThread(mThreadHandle)
    ElseIf tmpValue = False Then
        'Disable the thread
        Call SuspendThread(mThreadHandle)
    End If
End Property

Private Sub Class_Terminate()
    'Terminate the thread to prevent crashing if the app is closed
    'and the thread is still running (dangerous!)
    Call TerminateCurrentThread
End Sub
```

مثال زیر طرز استفاده از این کلاس را بیان می کند :

یک پروژه جدید باز کنید و ظاهر آنرا به شکل زیر تغییر دهید :



کد زیر مربوط به این فرم می باشد :

```
'API Declarations
Private Declare Function TerminateProcess Lib "kernel32" ( _
  ByVal hProcess As Long, ByVal uExitCode As Long) As Long
Private Declare Function GetCurrentProcess Lib "kernel32" () _
 As Long

Private ThreadControl1 As clsThreading
Private ThreadControl2 As clsThreading

Private Sub cmdSetProperties_Click(Index As Integer)
    Dim mThreadPriority As Long
    Dim mEnabled As Boolean
    If Index = 0 Then
        'Get the thread priority
        Select Case cmbThreadPriority(0).Text
            Case "Lowest"
                mThreadPriority = tpLowest
            Case "Below normal"
                mThreadPriority = tpBelowNormal
            Case "Normal"
                mThreadPriority = tpNormal
            Case "Above normal"
                mThreadPriority = tpAboveNormal
            Case "Highest"
                mThreadPriority = tpHighest
        End Select

        'Get the 'Enabled' value
        If chkEnabled(0).Value = 1 Then
            mEnabled = True
        ElseIf chkEnabled(0).Value = 0 Then
            mEnabled = False
        End If

        'Set the properties
        ThreadControl1.Priority = mThreadPriority
        ThreadControl1.Enabled = mEnabled
    ElseIf Index = 1 Then
        'Get the thread priority
        Select Case cmbThreadPriority(1).Text
            Case "Lowest"
                mThreadPriority = tpLowest
            Case "Below normal"
                mThreadPriority = tpBelowNormal
            Case "Normal"
                mThreadPriority = tpNormal
            Case "Above normal"
                mThreadPriority = tpAboveNormal
            Case "Highest"
                mThreadPriority = tpHighest
        End Select

        'Get the 'Enabled' value
        If chkEnabled(1).Value = 1 Then
            mEnabled = True
        ElseIf chkEnabled(1).Value = 0 Then
```

```
            mEnabled = False
        End If

        'Set the properties
        ThreadControl2.Priority = mThreadPriority
        ThreadControl2.Enabled = mEnabled
    End If
End Sub

Private Sub cmdStartThread_Click(Index As Integer)
    Dim mThreadPriority As Long
    Dim mEnabled As Boolean
    If Index = 0 Then
        'Get the thread priority
        Select Case cmbThreadPriority(0).Text
            Case "Lowest"
                mThreadPriority = tpLowest
            Case "Below normal"
                mThreadPriority = tpBelowNormal
            Case "Normal"
                mThreadPriority = tpNormal
            Case "Above normal"
                mThreadPriority = tpAboveNormal
            Case "Highest"
                mThreadPriority = tpHighest
        End Select

        'Get the 'Enabled' value
        If chkEnabled(0).Value = 1 Then
            mEnabled = True
        ElseIf chkEnabled(0).Value = 0 Then
            mEnabled = False
        End If

        'Create the thread
        ThreadControl1.CreateNewThread AddressOf ShowMovingLine, _
                                       mThreadPriority, mEnabled
        'Display the thread handle and the thread ID
        lblThreadHandle(0).Caption = ThreadControl1.ThreadHandle
        lblThreadID(0).Caption = ThreadControl1.ThreadID
    ElseIf Index = 1 Then
        'Get the thread priority
        Select Case cmbThreadPriority(1).Text
            Case "Lowest"
                mThreadPriority = tpLowest
            Case "Below normal"
                mThreadPriority = tpBelowNormal
            Case "Normal"
                mThreadPriority = tpNormal
            Case "Above normal"
                mThreadPriority = tpAboveNormal
            Case "Highest"
                mThreadPriority = tpHighest
        End Select

        'Get the 'Enabled' value
        If chkEnabled(1).Value = 1 Then
            mEnabled = True
        ElseIf chkEnabled(1).Value = 0 Then
```

```
            mEnabled = False
        End If

        'Create the thread
        ThreadControl2.CreateNewThread AddressOf ShowAnimation, _
                      mThreadPriority, mEnabled
        'Display the thread handle and the thread ID
        lblThreadHandle(1).Caption = ThreadControl2.ThreadHandle
        lblThreadID(1).Caption = ThreadControl2.ThreadID
    End If
End Sub

Private Sub cmdTerminateThread_Click(Index As Integer)
    'Terminate the thread
    If Index = 0 Then
        ThreadControl1.TerminateCurrentThread
    ElseIf Index = 1 Then
        ThreadControl2.TerminateCurrentThread
    End If
End Sub

Private Sub Form_Load()
    Set ThreadControl1 = New clsThreading
    Set ThreadControl2 = New clsThreading
    'Select the 'Normal' items into the combo boxes
    cmbThreadPriority(0).ListIndex = 2
    cmbThreadPriority(1).ListIndex = 2
    'Center picDisplay in the PictureBox
    picDisplay.Left = picOutput(1).Width / 2 - picDisplay.Width / 2
    picDisplay.Top = picOutput(1).Height / 2 - picDisplay.Height / 2
End Sub

Private Sub Form_Unload(Cancel As Integer)
    'Terminate the Threads
    ThreadControl1.TerminateCurrentThread
    ThreadControl2.TerminateCurrentThread
    'Fully terminate the current process
    Call TerminateProcess(GetCurrentProcess, ByVal 0&)
End Sub
```

یک ماژول جدید باز کنید و کد زیر را در آن بنویسید :

```
'This Module contains the functions called in the threads.
'They have to be in a module because you have to use the AddressOf-
'function which will only work with public functions

'API Declarations
Private Declare Function timeGetTime Lib "winmm.dll" () As Long
Private Declare Sub Sleep Lib "kernel32" ( _
                                    ByVal dwMilliseconds As Long)

Public Function ShowMovingLine()
    'Displays a line moving from the top of the PictureBox
    'to the bottom
    'in frmTest
    Dim PosBuf As Integer
    Dim TimeBuf As Long
```

```
    Do
        TimeBuf = timeGetTime
        'Clear the PictureBox
        frmTest.picOutput(0).Cls
        'Paint the line into the PictureBox
        frmTest.picOutput(0).Line (0, PosBuf)- _
                        (frmTest.picOutput(0).Width, PosBuf)

        PosBuf = PosBuf + 1
        If PosBuf > frmTest.picOutput(0).Height Then
            'Set position back to the top of the PictureBox
            PosBuf = 0
        End If

        'Wait some milliseconds...
        Do While timeGetTime - TimeBuf < 5
            Sleep 2
        Loop
    Loop
End Function

Public Function ShowAnimation()
    'Displays a simple Animation using single pictures in frmTest
    Dim PicNumber As Integer
    Dim TimeBuf As Long
    Do
        TimeBuf = timeGetTime
        'Put the actual picture into the PictureBox
        frmTest.picDisplay.Picture =
frmTest.imgAnimation(PicNumber).Picture

        PicNumber = PicNumber + 1
        If PicNumber > 3 Then PicNumber = 0

        'Wait some milliseconds...
        Do While timeGetTime - TimeBuf < 5
            Sleep 100
        Loop
    Loop
End Function
```

ا خطار!:

کدهای فوق را پس وارد کردن در محیط ویژوال بیسیک اجرا نکنید ! از آن فایل EXE بسازید و سپس فایل حاصل را اجرا نمایید.

نکاتی در مورد برنامه نویسی چند ریسمانی:

۱- همزمانی[4]

گاهی اوقات که چند ریسمان در برنامه شما کار می کنند لازم خواهد بود که فعالیت ریسمان های مجزا همزمان شوند. اگر هر ریسمان از دیگری بی اطلاع باشد ممکن است

[4] **Synchronization**

داده های مهم از بین بروند. برای اینکه ریسمان ها بتوانند با هم ارتباط داشته باشند می توان متغیری عمومی را تعریف کرد که توسط سایر ریسمان ها در برنامه شما قابل دیدن باشد. برای مثال اگر می خواهید بدانید که آیا کار یک ریسمان تمام شده است یا خیر می توانید یک متغیر عمومی از نوع **Boolean** تعریف کنید که در صورت در حال انجام بودن ریسمان **True** باشد و گرنه **False**.

Ref:

www.vb.net
www.planet-source-code.com/vb