



## Rilevare se un processo è attivo

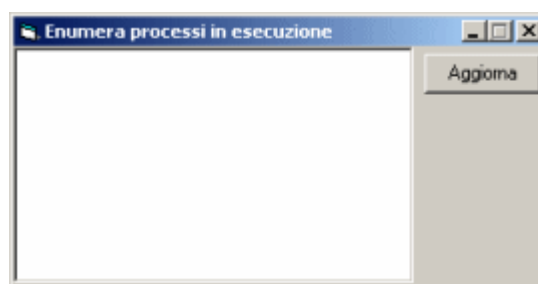
[http://www.vbsimple.net/activity/act\\_25.htm](http://www.vbsimple.net/activity/act_25.htm)

Richiesta di: [Nicola Mangiameli](#) - 15 Ottobre 2003

Difficoltà: 2 / 5

*Vorrei realizzare un programmino che in runtime controlli se è attivo un processo esterno (per intenderci se è in esecuzione un .exe specifico).*

Apparentemente il problema potrebbe sembrare più complesso di quanto non lo sia realmente ma vedremo come poche righe di codice possono risolvere questa necessità. A tal scopo aggiungeremo due semplici controlli sopra un form: una *ListBox* di nome **lbProcessi** ed un *CommandButton* di nome **cmdAggiorna**.



Il funzionamento è semplice quanto banale: il click sopra l'unico pulsante riporterà tutti i [processi](#) in esecuzione all'interno della *ListBox*, inizialmente vuota.

Il codice deve necessariamente utilizzare alcune funzioni dell'[API](#) di Windows; in particolare l'operazione si compone di due passi: la fotografia (*snapshot*) della situazione del sistema attuale ed il recupero di tutti i processi in esecuzione al momento della fotografia.


Iniziamo a vedere le definizioni API:


```
1. Option Explicit
2.
3. Private Const TH32CS_SNAPPROCESS = &H2
4. Private Const MAX_PATH As Integer = 260
5.
6. Private Type PROCESSENTRY32
7.     dwSize As Long
8.     cntUsage As Long
9.     th32ProcessID As Long
10.    th32DefaultHeapID As Long
11.    th32ModuleID As Long
12.    cntThreads As Long
13.    th32ParentProcessID As Long
14.    pcPriClassBase As Long
15.    dwFlags As Long
16.    szExeFile As String * MAX_PATH
17. End Type
18.
19. Private Declare Function CreateToolhelp32Snapshot Lib "Kernel32" (ByVal lFlags As Long, ByVal lProcessID As Long) As Long
20. Private Declare Function Process32First Lib "Kernel32" (ByVal hSnapshot As Long, uProcess As PROCESSENTRY32) As Long
21. Private Declare Function Process32Next Lib "Kernel32" (ByVal hSnapshot As Long, uProcess As PROCESSENTRY32) As Long
```

```

22. Private Declare Sub CloseHandle Lib "Kernel32" (ByVal hObject As Long)
23.

```

La costante  **TH32CS\_SNAPPROCESS** verrà utilizzata dalla funzione *CreateToolhelp32Snapshot* definita più giù per scattare la fotografia e congelare la situazione; questa costante indica alla funzione di memorizzare soltanto i processi in esecuzione (piuttosto che i [threads](#), i moduli, etc...). La costante **MAX\_PATH** definisce invece la lunghezza massima del percorso di ciascun file eseguibile. Vedremo in seguito a cosa serve.

Alle righe 6-17 è definito un nuovo tipo di dati  di nome **PROCESSENTRY32**; si tratta di una struttura necessaria per poter recuperare le informazioni sui singoli processi; fondamentalmente per il nostro scopo è necessario soltanto il campo **szExeFile** che andrà a ricevere il nome del file eseguibile di ogni processo rilevato.

Alla riga 19 è definita la funzione API *CreateToolhelp32Snapshot* che si occuperà di creare uno snapshot del sistema con il tipo di informazioni specificate nell'argomento **IFlags**. Le due funzioni successive *Process32First* e *Process32Next* consentono di recuperare rispettivamente il primo ed il successivo processo da uno snapshot. I dati recuperati saranno memorizzati nella struttura **PROCESSENTRY32**. Infine alla riga 19 la funzione *CloseHandle* consente di chiudere l'[handle](#) allo snapshot creato.

Tutto il codice del programma è concentrato nell'evento Click dell'unico pulsante presente sul form:

```

24. Private Sub cmdAggiorna_Click()
25.     Dim hSnapShot As Long
26.     Dim uProcess As PROCESSENTRY32
27.     Dim lngRet As Long
28.     hSnapShot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0&)
29.     uProcess.dwSize = Len(uProcess)
30.     lngRet = Process32First(hSnapShot, uProcess)
31.     lbProcessi.Clear
32.     Do While lngRet
33.         lbProcessi.AddItem Left$(uProcess.szExeFile, InStr(1, uProcess.szExeFile,
vbNullChar) - 1)
34.         lngRet = Process32Next(hSnapShot, uProcess)
35.     Loop
36.     CloseHandle hSnapShot
37. End Sub

```

La variabile **hSnapShot** riceverà l'handle dello snapshot creato; **uProcess** riceverà invece i dati sul singolo processo rilevato ed infine **lngRet** sarà utilizzata per verificare la disponibilità di ulteriori processi nello snapshot.

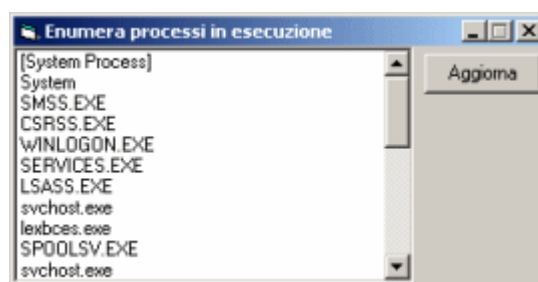
La prima operazione che verrà eseguita è naturalmente la creazione della fotografia (riga 28); come già detto l'handle recuperato resterà in **hSnapShot**. La costante specificata nella chiamata della funzione indicherà alla stessa di recuperare soltanto le informazioni sui processi in esecuzione.

Alla riga 29 è definita l'ampiezza della struttura **uProcess** e quindi sarà recuperato il primo processo dallo snapshot mediante *Process32First*. Il valore di ritorno di questa funzione se differente da zero indicherà il corretto recupero del processo dallo snapshot. Pertanto, una volta azzerato l'elenco **lbProcessi** sarà possibile avviare un ciclo che recuperi ogni processo. Alla riga 33 sarà inserito nell'elenco il nome del file eseguibile del processo rilevato; tale nome è naturalmente recuperato dal campo **szExeFile** della struttura. Poiché

trattasi di una stringa terminata con un [null](#) (sz indica appunto una stringa terminata con un carattere 0) sarà opportuno eliminare i caratteri in eccedenza.

Questo giro è ripetuto per ogni processo recuperato (la prima volta con *Process32First* e le successive con *Process32Next*). Al termine dell'operazione potremo chiudere l'handle allo snapshot creato e liberare quindi l'area in memoria.

Il risultato naturalmente riporterà il nome di tutti i processi in esecuzione, in maniera analoga al Task Manager di Windows NT/2000/XP. Da qui a capire se un determinato processo è in esecuzione il salto è breve; un semplice controllo su ciascun processo recuperato potrà rivelarci questa informazione.



Il codice non presenta alcun rischio particolare né difficoltà alcuna; la creazione dello snapshot congelerà la situazione dei processi al momento della fotografia e le successive interrogazioni saranno fatte su questa fotografia. È naturalmente opportuno creare uno snapshot ogni volta che si desidera analizzare le informazioni sui processi e non utilizzare la vecchia fotografia; i dati infatti dentro lo snapshot non saranno mai aggiornati e riporteranno sempre la medesima situazione.

[Fibia FBI](#)

2 Novembre 2003



[Torna all'introduzione delle Richieste dei lettori](#)

---