Trasferimento di files tra Client e Server (seconda parte)

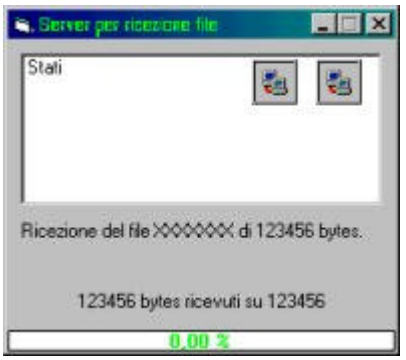
http://www.vbsimple.fbi/cliserv/clser_04_2.htm

Difficoltà: 4 / 5

[<< Continua dalla parte 1]

Visto il client nella parte precedente, rimane da vedere il server, leggermente più complesso del precedente.

Il secondo progetto si compone di un solo form che farà uso del controllo FBI Shape Progress Bar trattato nella sezione Controlli utente da aggiungere al progetto.



Sulla superficie del form avremo una *ListBox* di nome Stati, nella quale saranno inserite alcune informazioni sulla connessione e sul trasferimento dei dati.

Sono presenti anche due controlli *Winsock* di nome **Socket** e **Passivo**, due Label indicative di nome **StatoLabel** ed **Avanzamento**. Entrambe avranno la proprietà *Autosize* impostata a True, ma la prima avrà anche la proprietà *Wordwrap* impostata a True, mentre la seconda avrà la proprietà *Alignment* impostata a 2-Center.
Completiamo il nostro form con un'istanza della *FBI Shape Progress Bar* di nome **AvanzamentoProgress** con la proprietà *Align* impostata a 2-vbAlignBottom, *Alignment* impostata a 2-vbCenter e la proprietà *Decimali* impostata su 2.

Il server non potrà fornire alcun comando, eccetto che rispondere SI o NO alla richiesta di trasferimento del file per accettarlo o rifiutarlo.
Vediamo passo dopo passo il nostro codice:

```
1. Option Explicit
2.
3. Private FILEHANDLE As Integer
4. Private DIMENSIONEFILE As Long
5. Private FILEDASALVARE As String
6.
```

Abbiamo dichiarato 3 variabili che verranno utilizzate più volte all'interno del programma: **FILEHANDLE** è l'handle del file aperto per il trasferimento; **DIMENSIONEFILE** è la dimensione in bytes aspettata; infine **FILEDASALVARE** è il nome del file da salvare compreso del percorso della cartella.

```
7. Private Sub Form_Load()
8.    StatoLabel.Caption = ""
9.    Avanzamento.Caption = "0 bytes ricevuti su 0"
10.   Socket.LocalPort = 1500
11.   Socket.Listen
12. End Sub
```



```
13.
```

All'avvio del programma viene aperta la porta 1500 in ascolto per la connessione sul controllo **Socket** (righe 10 e 11).

```
14. Private Sub Form_Unload(Cancel As Integer)
15.   If Socket.State <> sckClosed Then Socket.Close
16. End Sub
17.
```

Così alla chiusura del form viene chiusa anche la connessione aperta con **Socket**.

```
18. Private Sub Socket_ConnectionRequest(ByVal requestID As Long)
19.   Socket.Close
20.   Socket.Accept requestID
21.   Stati.AddItem "Accettata connessione."
22. End Sub
23.
```

Nel momento in cui **Socket** riceve una richiesta di connessione, esso termina di ascoltare ed accetta la chiamata, senza preoccuparsi di chi sia.

```
24. Private Sub Socket_Close()
25.   If Passivo.State <> sckClosed Then Passivo.Close
26.   If Socket.State <> sckClosed Then Socket.Close
27.   Stati.AddItem "Connessione chiusa."
28.   Socket.Listen
29. End Sub
30.
```

Nel momento in cui la connessione viene chiusa (dal lato client), vengono chiusi tutti i socket aperti e **Socket** viene posto nuovamente in attesa di chiamata.

L'evento *DataArrival* sul primo socket servirà per accettare il trasferimento del file oppure annullarlo una volta avviato. Ricordiamo che il server accetta solo due comandi: `"/FILE NomeFile Dimensione"` e `"/FINE"`.

```
31. Private Sub Socket_DataArrival(ByVal bytesTotal As Long)
32.   Dim DATI() As Byte
33.   Dim NOMEFILE As String
34.   Dim DIMENSIONE As Long
35.   Dim POSIZIONE As Integer
36.   Dim TEMPSTR As String
37.
```

Saranno utilizzate una serie di variabili: la matrice di bytes **DATI** servirà per contenere i dati ricevuti dal client; **NOMEFILE** verrà utilizzata per contenere il nome del file in arrivo, **POSIZIONE** verrà utilizzata per effettuare l'estrazione dei parametri dai dati ricevuti dal client, **DIMENSIONE** sarà invece la dimensione in bytes del file in arrivo e **TEMPSTR** sarà utilizzata per manipolare le stringhe.

```
38.   Call Socket.GetData(DATI)
39.   DATI = StrConv(DATI, vbUnicode)
40.   Select Case Left(UCase(DATI), 5)
41.     Case "/FILE"
42.       TEMPSTR = Mid(DATI, 7)
43.       POSIZIONE = 0
44.       While InStr(POSIZIONE + 1, TEMPSTR, " ") > 0
45.         POSIZIONE = InStr(POSIZIONE + 1, TEMPSTR, " ")
46.       Wend
```



```
47.       NOMEFILE = Left(TEMPSTR, POSIZIONE - 1)
48.       DIMENSIONE = CLng(Mid(TEMPSTR, POSIZIONE + 1))
49.       TEMPSTR = "È in arrivo un file di nome " & NOMEFILE & " di " & CStr
    (DIMENSIONE) & " bytes."
50.       TEMPSTR = TEMPSTR & vbNewLine
51.       TEMPSTR = TEMPSTR & "Desideri accettarlo?"
```

Alla riga 38 vengono letti i dati in arrivo su **Socket** e subito successivamente vengono convertiti in Unicode, allo scopo di utilizzarli come stringhe.
Ottenuta la stringa inviata dal client, sarà necessario verificare se essa è uno dei due comandi accettati (righe 40, 41 e 80).

Se la stringa ricevuta inizia per `"/FILE"`, sarà necessario estrarre da essa il nome del file e la dimensione; l'inizio di tali informazioni è alla posizione 7 della stringa.

Alle righe 43-46 viene estratta la posizione dell'ultimo spazio separatore, al fine di rintracciare il punto dove termina il nome del file ed inizia la dimensione. Tale posizione sarà salvata nella variabile **POSIZIONE**.

Solo allora (righe 47 e 48) sarà possibile estrarre il nome del file (salvato in **NOMEFILE**) e la sua dimensione (memorizzata in **DIMENSIONE**).

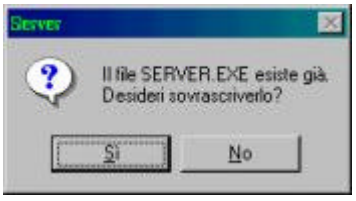
Alle righe 49-51 viene formata la stringa **TEMPSTR** che verrà utilizzata per mostrare un avviso all'utente. Essa conterrà anche il nome del file in arrivo e la sua dimensione.

```
52.       If MsgBox(TEMPSTR, vbYesNo + vbQuestion) = vbYes Then
53.         FILEDASALVARE = App.Path & "\RICEVUTI\" & NOMEFILE
54.         If Dir(FILEDASALVARE) <> "" Then
55.           TEMPSTR = "Il file " & NOMEFILE & " esiste già." & vbNewLine
56.           TEMPSTR = TEMPSTR & "Desideri sovrascriverlo?"
57.           If MsgBox(TEMPSTR, vbYesNo + vbQuestion) = vbYes Then
58.             Kill FILEDASALVARE
59.           Else
60.             Socket.SendData " -ERR: NONACCETTATO" & vbNewLine
61.             Stati.AddItem "File " & NOMEFILE & " rifiutato."
62.             Exit Sub
63.           End If
64.         End If
```

Tale avviso sarà mostrato all'utente. Se alla richiesta di trasferimento l'utente rispnderà ***NO***, sarà inviato al client il messaggio `" -ERR: NONACCETTATO"` e la procedura terminerà (righe 59-63).

Se l'utente invece risponderà ***SI***, verrà generato il percorso completo del file da salvare, composto dal percorso della cartella in cui il programma server si trova, dalla cartella ***RICEVUTI*** e dal nome del file in arrivo. Il percorso completo sarà memorizzato nella variabile **FILEDASALVARE**.

Prima di procedere alla scrittura del file, sarà necessario verificare se il file esiste (le varie soluzioni possibili sono spiegate in un HowTo). Se esso esiste, sarà mostrato un avviso di sovrascrittura all'utente (righe 55-57). Se l'utente richiederà la sovrascrittura, il file originale sarà cancellato prima di ricevere il



file nuovo (righe 57-58).

In caso contrario il file originale sarà lasciato lì e verrà ritornato al server il messaggi di non accettazione del file. In altre situazioni potrebbe essere comodo specificare un nuovo nome di file in cui salvare il file in arrivo.

```
65.         DIMENSIONEFILE = DIMENSIONE
66.         If Passivo.State <> sckClosed Then Passivo.Close
67.         Passivo.LocalPort = 0
68.         Passivo.Listen
69.         Socket.SendData " +OK: PORTA " & Passivo.LocalPort & vbNewLine
70.         Stati.AddItem "Accettazione del file " & NOMEFILE
71.         Stati.AddItem "Ascolto sulla porta " & Passivo.LocalPort
72.         StatoLabel.Caption = "Ricezione del file " & NOMEFILE & " di " & DIMENSIONE
    & " bytes."
73.         Avanzamento.Caption = "0 bytes ricevuti su " & DIMENSIONE
74.         AvanzamentoProgress.Value = 0
75.         AvanzamentoProgress.Max = 0
76.       Else
77.         Socket.SendData " -ERR: NONACCETTATO" & vbNewLine
78.         Stati.AddItem "File " & NOMEFILE & " rifiutato."
79.       End If
```

Alla riga 65 viene impostata la variabile globale **DIMENSIONEFILE**, che verrà utilizzata dal resto del programma, uguale a **DIMENSIONE**.

Sarà necessario aprire un socket passivo in ascolto su una porta casuale e comunicare il numero di tale porta al client, nella forma `" +OK: PORTA Numero"` (righe 66-69).
Saranno anche azzerati tutti i controlli grafici per monitorare l'avanzamento del traferimento.

Se, invece, l'utete ha risposto NO alla richiesta di trasferimento, sarà inviata al client la risposta `" ERR: NONACCETATO"`.

```
80.     Case "/FINE"
81.       Passivo.Close
82.       Close FILEHANDLE
83.       FILEDASALVARE = ""
84.       Stati.AddItem "Ricezione del file completata."
85.   End Select
86. End Sub
87.
```

In caso che il client volesse terminare il trasferimento a metà, invierà il comando "/FINE" (non implementato in questo esempio) al cui ricevimento il server si occuperà di chiudere il socket passivo e l'handle del file in scrittura.

```
88. Private Sub Passivo_ConnectionRequest(ByVal requestID As Long)
89.   FILEHANDLE = FreeFile
90.   Open FILEDASALVARE For Binary As FILEHANDLE
91.   Passivo.Close
92.   Passivo.Accept requestID
93. End Sub
94.
```

Nel momento in cui il client richiede il collegamento con il socket passivo, viene aperto il file **FILEDASALVARE** in modalità binaria e viene accettata la richiesta di collegamento.

```
95. Private Sub Passivo_Close()
96.   Close FILEHANDLE
```



```
 97.    FILEDASALVARE = ""
 98.    Stati.AddItem "Ricezione del file completata."
 99.  End Sub
100.
```

Analogamente nel momento della disconnessione viene chiuso il file aperto.

Tutti i dati che arriveranno al socket passivo saranno scritti all'interno del file di output.

```
101.  Private Sub Passivo_DataArrival(ByVal bytesTotal As Long)
102.    Dim DATI() As Byte
103.    Call Passivo.GetData(DATI)
104.    If UBound(DATI) + LOF(FILEHANDLE) + 1 >= DIMENSIONEFILE Then
105.      ReDim Preserve DATI(DIMENSIONEFILE - LOF(FILEHANDLE) - 1)
106.      Put FILEHANDLE, , DATI
107.      Stati.AddItem "Salvato chunk di " & UBound(DATI) + 1 & " bytes."
108.      Avanzamento.Caption = LOF(FILEHANDLE) & " bytes ricevuti su " & DIMENSIONEFILE
109.      AvanzamentoProgress.Max = DIMENSIONEFILE
110.      AvanzamentoProgress.Value = LOF(FILEHANDLE)
111.      Socket.SendData " +OK: FINE"
112.      DoEvents
113.      Passivo.Close
114.      Close FILEHANDLE
115.      FILEDASALVARE = ""
116.      Stati.AddItem "Ricezione del file completata."
```

L'evento *DataArrival* indica l'arrivo di dati al socket **Passivo**. Abbiamo detto che tali dati andranno scritti all'interno del file. Ma è necessario effettuare un controllo d'obbligo, ovvero che la somma dei dati ricevuti sia maggiore o uguale della dimensione aspettata (**DIMENSIONEFILE**). In tal caso sarà necessario eliminare i dati inutili che sono stati inviati mediante l'utilizzo dell'istruzione *Redim*, avendo cura di specificare la parola chiave ***Preserve*** per non distruggere il resto dei dati (riga 105).
Solo allora sarà possibile completare la scrittura dei dati nel file, inviare la risposta di fine del trasferimento " +OK: FINE" e chiudere socket passivo e file.

```
117.    Else
118.      Put FILEHANDLE, , DATI
119.      Stati.AddItem "Salvato chunk di " & UBound(DATI) + 1 & " bytes."
120.      Avanzamento.Caption = LOF(FILEHANDLE) & " bytes ricevuti su " & DIMENSIONEFILE
121.      AvanzamentoProgress.Max = DIMENSIONEFILE
122.      AvanzamentoProgress.Value = LOF(FILEHANDLE)
123.      Socket.SendData " +OK: RECV " & LOF(FILEHANDLE)
124.    End If
125.  End Sub
126.
```

Se invece la somma dei dati inviati non corrisponde alla fine del file, l'intero pacchetto sarà salvato nel file (riga 118), la barra di avanzamento sarà aggiornata (righe 121 e 122) e verrà rimandato indietro l'ACK `" +OK: RECV Dimensione"` del totale dei dati finora ricevuti.

```
127.  Private Sub Stati_DblClick()
128.    Stati.Clear
129.  End Sub
130.
```

Un'ultima funzioncina prima di vedere il funzionamento del programma è legata al doppio click sopra la LisBox **Stati**. L'intera lista sarà cancellata. Può essere utile quando essa diviene troppo piena o per verificare l'andamento del trasferimento senza riavviare il programma.

Avviare sia il client che il server su uno o due computer collegati. Inserire nel client l'indirizzo IP del computer in cui viene eseguito il server e premere il pulsante ***Connetti***. Se il server viene eseguito nella stesso computer del client, specificare come IP l'indirizzo di loopback **127.0.0.1**.
Specificare un file da trasferire e premere il pulsante ***Invia***.
Se il file non esiste sarà mostrato un messaggio di errore. Specificare quindi un nome di file valido e premere il pulsante Invia.

Il server riceverà la richiesta di trasferimento. Premendo il pulsante Si alla richiesta di trasferimento il programma client potrà avviare il trasferimento.

Man mano che il trasferimento procede saranno mostrate le informazioni di avanzamento nella ListBox del server e nella sua barra di avanzamento

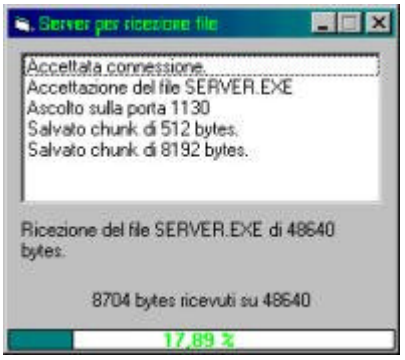


Figura 7

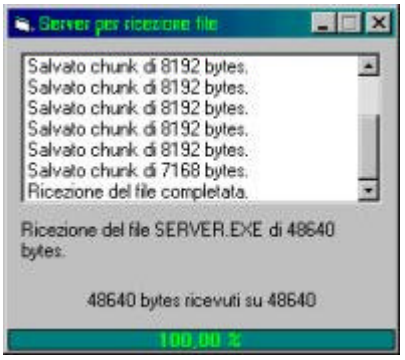


Figura 8

Il termine del trasferimento sarà notificato anche sul programma client con una finestra di messaggio apposita.

Il nostro programma è terminato.
Il protocollo di comunicazione utilizzato è molto simile all'FTP.
L'interfaccia utente è molto semplice e non effettua controlli severi sulla correttezza dei dati inseriti. Naturalmente questo esempio non pretende di essere un completo programma di trasferimento dati ma soltanto uno spunto quasi completo per comprendere il funzionamento.

Fibia FBI
2 Giugno 2001
Corretto il 25 Gennaio 2004

Torna all'indice Client/Server

