How to download a file using FTP

The following function shows how to connect to a ftp server and download a file. It uses the functions from `wininet.dll`.

You need a *TProgressBar* to show the progress and a *TLabel* to show progress information.

```
uses
  WinInet, ComCtrls;

function FtpDownloadFile(strHost, strUser, strPwd: string;
  Port: Integer; ftpDir, ftpFile, TargetFile: string;
  ProgressBar: TProgressBar): Boolean;

  function FmtFileSize(Size: Integer): string;
  begin
    if Size >= $F4240 then
      Result := Format('%.2f', [Size / $F4240]) + ' Mb'
    else
    if Size < 1000 then
      Result := IntToStr(Size) + ' bytes'
    else
      Result := Format('%.2f', [Size / 1000]) + ' Kb';
  end;

const
  READ_BUFFERSIZE = 4096;  // or 256, 512, ...
var
  hNet, hFTP, hFile: HINTERNET;
  buffer: array[0..READ_BUFFERSIZE - 1] of Char;
  bufsize, dwBytesRead, fileSize: DWORD;
  sRec: TWin32FindData;
  strStatus: string;
  LocalFile: file;
  bSuccess: Boolean;
begin
  Result := False;

  { Open an internet session }
  hNet := InternetOpen(
    'Program_Name', // Agent
    INTERNET_OPEN_TYPE_PRECONFIG, // AccessType
    nil,  // ProxyName
    nil,  // ProxyBypass
    0     // or INTERNET_FLAG_ASYNC / INTERNET_FLAG_OFFLINE
  );

  {
    Agent contains the name of the application or
    entity calling the Internet functions
  }

  { See if connection handle is valid }
  if hNet = nil then
  begin
    ShowMessage('Unable to get access to WinInet.Dll');
    Exit;
  end;

  { Connect to the FTP Server }
  hFTP := InternetConnect(
    hNet, // Handle from InternetOpen
    PChar(strHost), // FTP server
    port, // (INTERNET_DEFAULT_FTP_PORT),
    PChar(StrUser), // username
    PChar(strPwd),  // password
    INTERNET_SERVICE_FTP, // FTP, HTTP, or Gopher?
    0, // flag: 0 or INTERNET_FLAG_PASSIVE
    0  // User defined number for callback
  );
```

```
if hFTP = nil then
begin
  InternetCloseHandle(hNet);
  ShowMessage(Format('Host "%s" is not available',[strHost]));
  Exit;
end;

{ Change directory }
bSuccess := FtpSetCurrentDirectory(hFTP, PChar(ftpDir));

if not bSuccess then
begin
  InternetCloseHandle(hFTP);
  InternetCloseHandle(hNet);
  ShowMessage(Format('Cannot set directory to %s.',[ftpDir]));
  Exit;
end;

{ Read size of file }
if FtpFindFirstFile(hFTP, PChar(ftpFile), sRec, 0, 0) <> nil then
begin
  fileSize := sRec.nFileSizeLow;
  // fileLastWritetime := sRec.lastWriteTime
end else
begin
  InternetCloseHandle(hFTP);
  InternetCloseHandle(hNet);
  ShowMessage(Format('Cannot find file ',[ftpFile]));
  Exit;
end;

{ Open the file }
hFile := FtpOpenFile(
  hFTP, // Handle to the ftp session
  PChar(ftpFile), // filename
  GENERIC_READ, // dwAccess
  FTP_TRANSFER_TYPE_BINARY, // dwFlags
  0 // This is the context used for callbacks.
);

if hFile = nil then
begin
  InternetCloseHandle(hFTP);
  InternetCloseHandle(hNet);
  Exit;
end;

{ Create a new local file }
AssignFile(LocalFile, TargetFile);
{$i-}
Rewrite(LocalFile, 1);
{$i+}

if IOResult <> 0 then
begin
  InternetCloseHandle(hFile);
  InternetCloseHandle(hFTP);
  InternetCloseHandle(hNet);
  Exit;
end;

dwBytesRead := 0;
bufsize := READ_BUFFERSIZE;

while (bufsize > 0) do
begin
  Application.ProcessMessages;

  if not InternetReadFile(
    hFile,
    @buffer, // address of a buffer that receives the data
    READ_BUFFERSIZE, // number of bytes to read from the file
    bufsize // receives the actual number of bytes read
  ) then Break;
```

```
    if (bufsize > 0) and (bufsize <= READ_BUFFERSIZE) then
      BlockWrite(LocalFile, buffer, bufsize);
    dwBytesRead := dwBytesRead + bufsize;

    { Show Progress }
    ProgressBar.Position := Round(dwBytesRead * 100 / fileSize);
    Form1.Label1.Caption :=
      Format(
        '%s of %s / %d %%',
        [
          FmtFileSize(dwBytesRead),
          FmtFileSize(fileSize),
          ProgressBar.Position
        ]
      );
  end;

  CloseFile(LocalFile);

  InternetCloseHandle(hFile);
  InternetCloseHandle(hFTP);
  InternetCloseHandle(hNet);
  Result := True;
end;
```

Original resource:	*Swiss Delphi Center*
Author:	Thomas Stutz
Contributor:	Frederik Smith
Added:	2010-12-17
Last updated:	2010-12-17