How to search for a pattern in a file

Question

I need to locate a pattern in a file (both text and binary) - just like the *Pos* function does with the strings. Preferably, it should deal with *TFileStream*. Straightforward solution first seemed kind of expensive - that is to just plainly go through the stream comparing patterns on every step.

Answer 1:

You can do it that way but it is much faster to load chunks of data into a sizeable buffer and do the search in the buffer. Here is an example:

```
function ScanFile( const filename: String; const forString: String;
  caseSensitive: Boolean ): LongInt;
  { returns position of string in file or -1, if not found }
const
  BufferSize= $8001;  { 32K + 1 bytes }
var
  pBuf, pEnd, pScan, pPos: Pchar;
  filesize: LongInt;
  bytesRemaining: LongInt;
  bytesToRead: Word;
  F: File;
  SearchFor: Pchar;
  oldMode: Word;
begin
  Result := - 1;  { assume failure }
  if (Length( forString ) = 0) or (Length( filename ) = 0) then
    Exit;
  SearchFor := Nil;
  pBuf := Nil;
  { open file as binary, 1 byte recordsize }
  AssignFile( F, filename );
  oldMode := FileMode;
  FileMode := 0;  { read-only access }
  Reset( F, 1 );
  FileMode := oldMode;
  try  { allocate memory for buffer and pchar search string }
    SearchFor := StrAlloc( Length( forString ) +1 );
    StrPCopy( SearchFor, forString );
    if not caseSensitive then  { convert to upper case }
      AnsiUpper( SearchFor );
    GetMem( pBuf, BufferSize );
    filesize := System.Filesize( F );
    bytesRemaining := filesize;
    pPos := Nil;
    while bytesRemaining > 0 do
    begin
      { calc how many bytes to read this round }
      if bytesRemaining >= BufferSize then
        bytesToRead := Pred( BufferSize )
      else
        bytesToRead := bytesRemaining;
        { read a buffer full and zero-terminate the buffer }
      BlockRead( F, pBuf^, bytesToRead, bytesToRead );
      pEnd := @pBuf[ bytesToRead ];
      pEnd^ := #0;
      { scan the buffer. Problem: buffer may contain #0 chars! So we
      treat it as a concatenation of zero-terminated strings. }
      pScan := pBuf;
      while pScan < pEnd do
      begin
        if not caseSensitive then  { convert to upper case }
          AnsiUpper( pScan );
        pPos := StrPos( pScan, SearchFor );  { search for substring }
        if pPos <> Nil then
        begin  { Found it! }
```

```
          Result := FileSize - bytesRemaining + LongInt( pPos )
            - LongInt( pBuf );
          Break;
        end;
        pScan := StrEnd( pScan );
        Inc( pScan );
      end;
      if pPos <> Nil then
        Break;
      bytesRemaining := bytesRemaining - bytesToRead;
      if bytesRemaining > 0 then
      begin
        { no luck in this buffers load. We need to handle the case of
        the search string spanning two chunks of file now. We simply
        go back a bit in the file and read from there, thus inspecting
        some characters twice }
        Seek( F, FilePos(F) - Length( forString ));
        bytesRemaining := bytesRemaining + Length( forString );
      end;
    end;
  finally
    CloseFile( F );
    if SearchFor <> Nil then
      StrDispose( SearchFor );
    if pBuf <> Nil then
      FreeMem( pBuf, BufferSize );
  end;
end;
```

Tip by Peter Below

Answer 2:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  s: String;
  hFile: THandle;
  hFileMapObj: THandle;
  pSharedBuf:  Pointer;
  Time0: Integer;
  p: PChar;
begin
  if not OpenDialog1.Execute then
    Exit;
  s := InputBox('Find','Match','');
  Time0 := GetTickCount;
  hfile := 0;
  hFileMapObj := 0;
  pSharedBuf := nil;
  try
    hFile := FileOpen(OpenDialog1.FileName, fmOpenRead);
    Win32Check(hFileMapObj <> INVALID_HANDLE_VALUE);
    hFileMapObj :=
      CreateFileMapping(hFile, nil, PAGE_READONLY, 0, 0, nil);
    Win32Check(hFileMapObj <> 0);
    pSharedBuf := MapViewOfFile(hFileMapObj, FILE_MAP_READ, 0, 0, 0);
    Win32Check(pSharedBuf <> nil);
    P := StrPos(PChar(pSharedBuf), PChar(s));
  finally
    if pSharedBuf <> nil then
      UnMapViewOfFile(pSharedBuf);
    if hFileMapObj <> 0 then
      CloseHandle(hFileMapObj);
    if hFile <> 0 then
      CloseHandle(hFile);
  end;
  if P = nil then
    Caption := Format('Not found, ticks=%d', [GetTickCount - Time0])
  else
    Caption := Format(
      'Found it at pos %d, ticks=%d',
      [Integer(P - PChar(pSharedBuf)), GetTickCount - Time0]);
end;
```

Tip by Leonid Troyanovsky

Original resource:	The Delphi Pool
Author:	P Below & L Troyanovsky
Added:	2010-02-22
Last updated:	2010-02-22