How to detect the types of executable files (part 1 of 3)

Contents, abstract and outline design

Contents

- *Abstract*
- *Outline design*
- *Developing the function*
- *Putting it all together*
- *Demo program*
- *Conclusion*
- *Credits*
- *Feedback*

Abstract

This article looks at how we examine a file to check if it is a DOS or Windows executable and, if so, whether it is a program file or a DLL.

We will develop a function – *ExeType* – that will test a file and return a value that describes what type of excutable file it is, if any.

An install program may need this information, or you may need to check a given file is a program or DLL before attempting to run or load it.

This article was revised on 9 September 2003 to add checks for a valid MS-DOS file. It was revised again on 10 September 2003 to add checks for Virtual Device Driver files.

Outline design

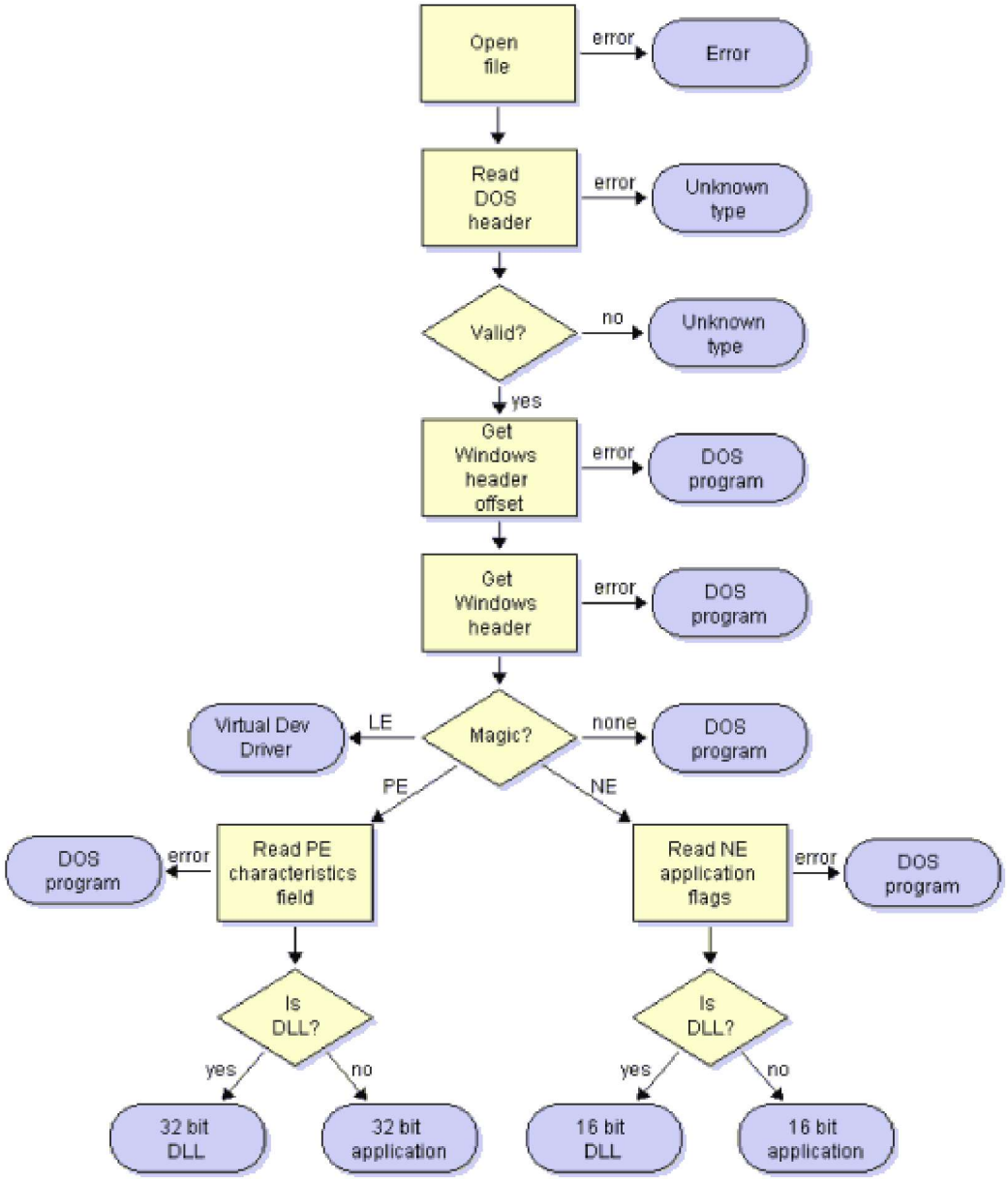
Before we start coding our function, let us look at how we're going to accomplish this task. Our approach will be to open the file of interest and scan through it looking for markers to indicate its file type.

All Windows executable files begin with a MS-DOS executable stub, so we first test for a valid MS-DOS executable using information from the MS-DOS program header that is present in every executable file. We then check for markers for a 16 bit or 32 bit Windows executable or for a virtual device driver (VXD). If we establish the file is a Windows executable we look for information that determines whether the file is an application or is a DLL. A review of the MS-DOS, Windows NE (16 bit) and PE (32 bit) executable file formats leads us to note the following:

- All DOS program files (and therefore Windows executables) begin with a "magic number"; the word value $5A4D ("MZ" in ASCII).
- We use the DOS header to check that the file length exceeds or is equal to the minimum length of the DOS executable and that the offset of the DOS relocation table lies within the file.
- Windows executables have a header record whose offset in the file is given by the long word at offset $3C.

- The Windows header begins with a "magic number" word whose value indicates whether this is a 16bit (NE format) or 32 bit (PE format) executable or a virtual device driver (LE format). The word is $454E ("NE" in ASCII), $4550 ("PE") or $454C ("LE").
- 32 bit Windows executables have an "image header" immediately following the $4550 magic number. This header structure has a *Characteristics* field which is a bit mask. If the bit mask contains the flag *IMAGE_FILE_DLL* then the file is a DLL, otherwise it is a program file.
- 16 bit Windows programs have a byte sized field at offset $0D from the start of the Windows header which is a bit mask providing information about the file. If this field contains the flag $80 then the file is a DLL, otherwise it is a program.

The following flowchart illustrates the tests we will use, based on the information above:



So, in pseudo-code, our function can be described as:

```
Open file (Return Error type if doesn't exist)
Read DOS header from file (Return Unknown file type if can't read)
If first word = 'MZ' and file size is valid then
  Get offset of Window header record (Return DOS type if can't read)
  Get Windows header record data (Return DOS type if can't read)
  case first word of
    'PE':
      Read Windows exe header record (Return DOS type if can't read)
      if Characteristics field contains IMAGE_FILE_DLL flag then
        Return 32 bit Windows DLL type
      else
        Return 32 bit Windows application type
      end
```

```
    'NE':
      Read in byte at offest $0D (Return DOS type if can't read)
      if byte field contains flag $80 then
        Return 16 bit Windows DLL type
      else
        Return 16 bit Windows application type
      end
    'LE':
      Return VXD device driver type
    else:
      Return DOS type
  end
else
  Return Unknown file type
end
```

Listing 1

We now have an outline design from which to work. In the *next part* of this article we'll begin coding the function.