18 File Format

"Practice hospitality."

All except the text file (with .txt extension) use their own standards to save and organize their instruction. For example, the EXE file put up "MZ" in its first two bytes. Thus each file got its own architecture or *File Format*. If we know the file format of a particular file, we can read or create those files. For example if we know the file format of BMP file, we can read it or even we can create it. We must understand that each and every file type uses its own *file formats*. Each file format has its own advantages and drawbacks. The software that creates a file of specific type should be aware of its *file format*. For example, the Linker must know the file format of EXE file, Paintbrush must know the file format of BMP file and so on.

Usually all files contain what is called as *file header* and it is nothing but the first few bytes of a file. Each file type uses specific size for the file header. For example, the size of File Header for EXE is 28 bytes, for BMP file it is 14 bytes. The file Header contains many useful information such as its file types i.e. whether EXE or BMP or GIF. The file type is identified by what is known as *signature*. The signature of the EXE file is "MZ", the signature of BMP file is 19778 and so on. After the File Header, the files may contain instructions or some other header. For example, most of the image files have got the file header in the beginning, then color table and then instructions.

If you know the file format you can do miracles. Most of the software vendors *document the file format* whenever they introduce a new file type. But certain narrow-minded vendors may keep the file format as secret. In such a case you have to crack the file format with the help of certain software (usually DEBUG & simple C programs).

In this chapter, I just introduce the concept. But in the following chapters and in CD you can see some real examples. You can get almost all file formats from the File Format Encyclopedia that is available in the CD.

18.1 Example

The following shows the file format of EXE file format:

.EXE - DOS EXE File Structure		
Offset	**Size**	**Description**
00	word	"MZ" - Link file .EXE signature (Mark Zbikowski?)
02	word	length of image mod 512
04	word	size of file in 512 byte pages
06	word	number of relocation items following header
08	word	size of header in 16 byte paragraphs, used to locate the beginning of the load module
0A	word	min # of paragraphs needed to run program
0C	word	max # of paragraphs the program would like
0E	word	offset in load module of stack segment (in paras)
10	word	initial SP value to be loaded
12	word	negative checksum of pgm used while by EXEC loads
14	word	pgm
16	word	program entry point, (initial IP value)
18	word	offset in load module of the code segment (in paras)
1A	word	offset in .EXE file of first relocation item overlay number (0 for root program)

- relocation table and the program load module follow the header
- relocation entries are 32 bit values representing the offset into the load module needing patched
- once the relocatable item is found, the CS register is added to the value found at the calculated offset

Registers at load time of the EXE file are as follows:

AX:	contains number of characters in command tail, or 0
BX:CX	32 bit value indicating the load module memory size
DX	zero
SS:SP	set to stack segment if defined else, SS = CS and SP=FFFFh or top of memory.
DS	set to segment address of EXE header
ES	set to segment address of EXE header
CS:IP	far address of program entry point, (label on "END" statement of program)

Suggested Projects

After reading all the chapters of this book only, you will get thorough ideas about file formats and its usage. Then you can try the following projects:

1. Write your own EXE2BIN utility.
2. Remove relocation found in EXE files.
3. Check out all the available file formats in the File Format Encyclopedia found in the CD. Crack the file types for which file format is not yet available and try to document the file format. (Of course it is illegal!) (Hint: Use DEBUG or simple C programs to read byte by byte)
4. Write your own compression utility and thus develop your own file format for that. Compare its efficiency with PKZIP.
5. Write software to split and join files. For the good quality, it needs that you have to use your own file Header or file format.
6. Write a BMP file creator (i.e. Paintbrush) in high resolution VESA mode. The software has to use both mouse and graphics stylus as input devices.
7. Write a PDF to TXT (text) conversion utility.
8. Write your own image creation utility that uses MP3 compression algorithm and thus develop your own file format for that.
9. Add help (the one which always get invoked when we press Ctrl+F1) for the library that you created. For example if you create a mouse library, and you have `InitMouse( )` function, when you press Ctrl+F1, you should get the help for that function. (Hint: You should know the file format of Turbo C's help file).