Extended Attributes (EAs) and INI Files

BY RICK BYRLEY

If I were asked to list the two aspects of OS/2 a user must have to maintain a production ready system, I would list Extended Attributes (EAs) and INI files.

Perhaps the most attractive feature of the Workplace Shell and the OS/2 Desktop is its seemingly limitless customizability. Objects from folders to applications may be set to behave exactly as the user desires. Colors, backgrounds, associations, opening behavior and view — all are within the user's grasp within the pages of the object settings notebook. This object persistent information — information about the object that is maintained across boots — is stored in a couple of places. This month's column examines OS/2 Extended Attributes (EAs) and INI files.

Perhaps the most attractive feature of the Workplace Shell and the OS/2 Desktop is its seemingly limitless customizability.

Most of you are probably familiar with the standard DOS file attributes: Read Only, Hidden, System and Archive. OS/2 allows for much more information to be associated with a file or directory. These Extended Attributes are not only convenient, they are critical to the functioning of the WPS, containing the information the system needs to associate files with programs, icons with objects, long file names with files copied from HPFS to FAT, and other system functions.

THE STRUCTURE OF EAS

There are nine standard EA types which are identified to the operating system by a word (two-byte) identifier. As shown in Figure 1, the EA types may be grouped into two categories: Single-Valued, which contain only one data item per EA, and Multi-Valued, which may contain multiple items. A programmer may define his own EA type by simply specifying a value between 0x0000 and 0x7FFF for the hex identifier. Values above this range are reserved. For all Single-Valued EAs, the first word following the hex type identifier indicates the length of the data which follows. For Multi-Valued EAs, the first word following the hex type identifier indicates the code page to be used. The code page is followed by a word specifying the number of entries in the EA. For Multi-Valued, Single-Typed EAs, the next word indicates the type of entries contained in the EA. Although this sounds slightly confusing, it is actually quite simple: the first type specification denotes that the EA contains multiple values, or entries; the second type specification indicates the type of those entries. This second type specification is followed by a series of 'records',each consisting of two bytes indicating the length of the data followed by the data itself. Since Multi-Typed EAs contain, as their name implies, various EA types, each value or entry must define its type; thus the 'record' includes two bytes indicating the type, followed by the length and data as in the Single-Typed Multi-Valued EA.

The most common problem with EAs is related to its use with the FAT File system. The FAT file system uses a pointer in the directory structure, to an entry in the EA DATA. SF file, which maintains EA data for FAT files. If two files reference the same pointer the result is cross-linked EAs. The only sure method of addressing this problem is to correct the pointers with a sector editor, although copying the files to another location has a fifty-fifty shot at working.

Figure 1: EA Types

```
Single-Valued:
EAT_BINARY      FFFE    Binary data
EAT_ASCII       FFFD    Text
EAT_BITMAP      FFFB    Bitmap
EAT_METAFILE    FFFA    Metafile
EAT_ICON        FFF9    Icon EAT_EA FFFE Another associated EA

Multi-Valued:
EAT_MVMT        FFDF    Multi-Valued, Multi-Typed
EAT_MVST        FFDE    Multi-Valued, Single Typed
EAT_ASN1        FFDD    ASN.1 field data
```

INI FILES

INI files date back to the DOS days. Applications must be able to store user-specific data which tailors the application to the user's needs. The strategy under DOS was to maintain a file which was read when the program was initialized, which is why the INI extension was used. The disadvantage of this method is that it leaves initialization information scattered in many different locations. For this reason,Microsoft Windows provided the WIN.INI and SYSTEM.INI files to store this information. Unfortunately these files were simple ASCII files and were easy to get to but inefficient.

Windows 9x and NT, like OS/2 before them, use a different strategy for maintaining initialization information, a hierarchy of binary data. At the top level is the Application Name, subdivided into Keys defined by the application, with each Key having an associated value. It is left to the application to format and interpret the value.

The system initialization file, OS2SYS.INI, should never be used by applications. You may need to alter it, however, to solve a system problem.

Applications may write their own INI files using the OS/2 APIs, but most use the standard User INI file provided by OS/2, OS2.INI. The system initialization file, OS2SYS.INI, should never be used by applications. You may need to alter it, however, to solve a system problem.

Which brings up the first of several management problems with OS/2 INI files. There is no utility provided with OS/2 to display or directly modify INI files. Your best bet is to either utilizing REXX or one of the various INI editors available. A second problem is that INI entries are not portable. The ASCII format of Windows 3.x INI files and the application specific nature of PIF files meant a user could install an application on one system and then move it, with customization, to another Windows system. This is not possible under OS/2 except using a third-party utility such as UniMaint from SofTouch Systems. Still another problem with OS/2 INI files is housekeeping. Very few applications bother to clean up after themselves, leaving obsolete and potentially system-killing entries in the INI files. And finally, since OS/2 always has the User and System INI files open, there is no way to create a backup of these files except during the boot process. Fortunately, these problems are all addressed by third-party utilities available on the Internet, including FM/2 and WPTool (both available at www.musthave.com) or from vendors such as UniMaint and the GammaTech Utilities from SofTouch Systems. *ts*

Rick Byrley is senior workstation division technician for SofTouch Systems, Oklahoma City, Okla., which provides both mainframe and PC software solutions. His primary focus is object-oriented programming. He can be reached at rbyrley@softouch.com.