

USING DISK EDIT

Mark E. Donaldson

You might not know it, but there is a disk editor utility for Windows NT in the style of the venerable Norton Disk Edit for DOS. What's more, the utility understands FAT and NTFS and it's free. Microsoft apparently shipped DiskEdit accidentally, a tool that must be an internal debugging tool for their file systems teams, on the Windows NT 4.0 Service Pack 4 CD. DiskEdit has a peculiar interface that would take a small manual to document, but I'm going to get you started with a simple walkthrough. I'll focus on using DiskEdit to unravel the NTFS file system format, since DiskEdit is the only publicly available tool I know of that understands NTFS.

First you need to get DiskEdit from the Service Pack 4 (SP4) CD-ROM. Simply copy it from the \i386 directory to your hard disk. If you want to use DiskEdit under Win2K you'll need to create a private directory for it and copy the following DLLs from a SP4 winnt\system32 directory (or SP4 CD) to the same directory as DiskEdit:

- IFSUTIL.DLL
- ULIB.DLL
- UNTFSS.DLL
- UFAT.DLL.

Now you can start DiskEdit.

For this walkthrough create a directory called TEMP at the root of a NTFS drive and create a file called OUT.TXT in that directory by typing the following command in a command-prompt window with TEMP as the current directory: `echo hello > out.txt`. Select the drive with your new OUT.TXT file in DiskEdit by choosing the File|Open menu item and entering the drive's letter in the Volume Name field of the resulting dialog box. Make sure you include the colon e.g. "d:". Virtually all of DiskEdit's functionality requires that you've opened a drive.

We're going to locate the OUT.TXT file by starting at the root directory of the NTFS drive. Select the menu entry Read|NTFS File Record to open a dialog box that lets you view any MFT record entry just by entering its number. The first 16 MFT record entries of every NTFS drive are reserved and correspond to pre-defined NTFS metadata files. Here are the number assignments (note that DiskEdit interprets all input as hexadecimal):

- 0: \$MFT - MFT
- 1: \$MFTMirr - MFT Mirror (a copy of the first 4 entries of the MFT)
- 2: \$LogFile - NTFS LogFile
- 3: \$Volume - volume information file
- 4: \$AttrDef - the attribute definition file
- 5: . - the root directory
- 6: \$Bitmap - the volume allocation bitmap file
- 7: \$Boot - the boot sector
- 8: \$BadClus - the bad cluster database file
- 9: \$Secure - new to SP4, the security attribute database
- A: \$UpCase - the lower-to-upper case mapping file
- B: \$Extend - new to Win2K, the directory that contains the reparse, object ID, and quota metadata files
- C-F: Unused as of NTFS v5.0 (Win2K)

USING DISK EDIT

Mark E. Donaldson

Go ahead and look at some of these MFT entries. You'll start to notice a common theme: they all consist of attributes like \$INDEX_ROOT, \$FILE_NAME, and \$DATA. It's in attributes where data specific to a file is stored. When attribute data is small NTFS stores the data within the file's MFT record as "resident" data, and when the data is large NTFS stores the data external to the record in clusters on disk as "non-resident" data.

Now enter "5" as the file number and you'll be viewing the root directory's file. We're going to look at the files and directories that are in the root directory by viewing the directory's \$INDEX_ALLOCATION attribute, an attribute specific to directories that records a directory's contents. To do so select the Read|NTFS Attribute menu entry, which opens another dialog box. DiskEdit is sensitive to case so enter the following precisely as I've listed it:

Base Frs Number: 5

Base Frs (Base File Record Segment) is another name for MFT number. You enter to 5 to specify that you want to read an attribute from the root directory.

Attribute Type: \$INDEX_ALLOCATION

This indicates to DiskEdit that you want to read the directory's content data. I recommend using the pull-down menu to chose the attribute you want since DiskEdit is very picky about the way the attribute type is entered.

Attribute Name: \$I30

If you view the \$INDEX_ALLOCATION attribute of the root directory you'll see that "\$I30" is listed as its name in its "Type code, name" line, so that's what you enter as the attribute name.

Press OK and you'll see a hexadecimal dump of the attribute's contents. We want to see something more intelligible so select the View|NTFS Index Buffer menu entry. You'll be presented with the listing of the directory's contents. Scroll through the listing until you see the entry that has the name "TEMP". If you don't see it, the entry might be located in the root directory's \$INDEX_ROOT attribute, an attribute type also associated with directories, and that always has its value stored in the MFT record. Index root entries and allocation entries together form a B+ tree structure storing all of a directory's entries. If you have to view the \$INDEX_ROOT attribute just follow the same steps for viewing that attribute as you did for viewing the \$INDEX_ALLOCATION attribute. As you scroll through an index buffer you may come upon double-lines of asterisks: these designate the end of one index buffer and the beginning of the next.

Once you've found the TEMP directory's entry make a note of its File Reference (FRS). Select Read|NTFS File Record and enter TEMP's FRS. Now you're looking at the MFT record for the TEMP directory. We want to find the OUT.TXT file, so we'll have to look through TEMP's contents to find it. View the \$INDEX_ALLOCATION (or \$INDEX_ROOT) attribute of the TEMP directory, switch to viewing the data as a NTFS Index Buffer, and locate the OUT.TXT file. Remember to enter TEMP's FRS as the base FRS number in the attribute selection dialog. If you just created TEMP then it will only have an \$INDEX_ROOT (if you mistype something you'll get the pleasure of seeing on of DiskEdit's empty error dialogs).

USING DISK EDIT

Mark E. Donaldson

When you've found OUT.TXT and determined its FRS use Read|NTFS File Record to look at its MFT entry. Scroll down until you find the \$DATA attribute. You're now looking at the location of OUT.TXT's data. Since we made a small file, the data is stored in the MFT record. If you try to view OUT.TXT's \$DATA attribute using DiskEdit you'll see nothing, since DiskEdit doesn't properly show resident data (one of DiskEdit's many bugs). So, copy a largish (> 2KB) text file to \TEMP\ OUT.TXT. Now you can view the OUT.TXT data in one of two ways: you can examine the start of the data (or all of it if its contiguously stored on disk) by using Read|NTFS Cluster and specifying the first "lcn" value you see in OUT.TXT's \$DATA attribute "Extent List"; or you can use Read|NTFS Attribute and enter "\$DATA" as the attribute type and nothing (as in don't type anything into that field) as the attribute name.

Extent lists describe the location of an attribute's non-resident data. Each contiguous block of data of up to 16 clusters in length is described by one extent list entry. An extent list entry specifies a virtual cluster number (vcn), logical cluster number (lcn), and run length. A Vcn is the cluster within the file at which the data described by the entry starts. A Lcn designates the logical cluster where the data is stored on disk, and the runlength is the number of bytes of attribute data at that location (remember, DiskEdit is showing you hexadecimal values).

I walked you through the long way of finding the OUT.TXT file's MFT record by showing you how to scan directory contents. There's a shortcut, however: select Crack|NTFS Path and enter TEMP\OUT.TXT. You'll be presented with OUT.TXT's FRS and you can use Read|NTFS File Record to go right to it.

That brings me to the end of my DiskEdit primer. I encourage you to play around with this nifty tool by browsing your FAT or NTFS drives. Its highly unlikely that you'll ever find occasion to use DiskEdit to modify data in order to get your disk out of a jam, but if you're curious about the NTFS on-disk format (the FAT format is well-documented) this is the perfect tool for investigating.