Fedit 3.0 Changes

This document lists the changes to Fedit in version 3.0. I must take this opportunity to thank my beta testers, particularly Billy Steinberg and Robert Wiggins (in alphabetical order), for the hard work they have done to detect problems and their help in suggesting (and sometimes insisting on) better approaches to new features. Their help has been invaluable.

Firstly, a number of minor bugs have been fixed. As far as I remember they are:

> the search mechanism would spometimes fail if a search string was found starting on the last character of a sector.
>
> When selecting a volume, if a disk was inserted the name did not switch to that volume.
>
> An attempt to write a sector using an extended write beyond the end of the volume was sometimes not detected.
>
> The tags read from a disk would be incorrect immediately after some error conditions.
>
> The program would fail to print correctly with a laser printer.
>
> The program would permit tag searches even on volumes that do not support tags.

All the above have been corrected.

Next, some addtional features have been added:

The data and resource forks of a file can be interchanged (Options Menu).

The standard boot blocks can be written to a disk. If the control and option keys are held down when the command is selected, the pseudo 128K version of the boot blocks are written instead (File Menu).

The configuration data at the start of the boot blocks can be edited (File Menu).

Two file recovery mechanisms have been added. These are described below.

The Special Menu

One of the most difficult procedures in disk handling is recovering from disk problems. The tags on each sector are used for data recovery, but most hard disk manufacturers do not support tags. The items in the special menu can help make recovery possible on these types of disk as well as the Sony diskettes. Unfortunately, due to space restrictions none of the file recovery items will run on a 128K system. If you are running Fedit under Switcher you will also want to play with the partition size until you have the number of buffers that you require.

When Fedit is initialized, the program will allocate a number of buffers for use in storing data in memory. This will vary between 1 and 400 buffers depending on how much memory is available after making an allowance for desk accessories that you may wish to use. Using the items in this menu you may read data from any sector on disk into any buffer and then write the data out into another area of the disk or into a new file. This provides an easy way to reconstruct files.

A second method of file recovery is also available. This method allows you to select sectors and tell Fedit to add them to the end of a file. This method can be more convenient that using buffer copies, but cannot always be used, for example if there are problems with the disk directory or the volume allocation tables. In these cases you will have to copy the file into one or more buffers before writing them to a different disk.

Both the above methods have advantages and disadvantages. They are somewhat crude methods of recovering data, but without tags there appears to be no better methodology available. The real answer is for all disk manufacturers to support the Apple tag storage areas on each sector.

For users who have tags available, the routines in this menu can still be used. Alternate methods for file recovery and recovering from disk directory or volume allocation table errors will be provided soon.

The following items are in the special menu.

Next Display Buffer

This routine allows you to select the next buffer to be displayed on the screen.

Previous Display Buffer

This routine allows you to select the previous buffer to be displayed on the screen.

Specify Display Buffer

Using this routine you may select a buffer to be displayed on the screen. The currently displayed buffer is used for all single sector disk operations.

Multiple Sector Read

This command allows you to read a number of sectors from the currently open file or volume into memory. Each sector read from disk will be stored in a separate buffer starting with the buffer selected in this menu. Any sector in memory can be edited before writing it back to disk.

Multiple Sector Write

This command allows to to write data from memory buffers onto the current volume. The data is written directly to disk and not as part of any file that may be open. The volume allocation table is not updated to reflect any changes that may occur. This command can be useful if you need to write into the system areas of a disk.

Write Sectors to File

This command is a good tool for trashing disks. Be careful using it. It permits you to write data from the buffers onto a selected file on disk. No validity checks whatsoever are made on the data, so you are quite able to create some very strange files. All that is guaranteed is that the resultant disk file will conform to the requirements of the Macintosh file manager. The command is unique in Fedit because it will permit you to write to only part of a sector, thus allowing great flexibility in what data you change. By moving the end of file point you can write to any place in a file.

Link to Data Fork

This command is designed for file recovery. When invoked, the allocation block to which this sector belongs is linked in the Volume Allocation Table at the end of the Data Fork of the currently selected output file. You should note that the complete allocation block is linked in, not just the displayed sector. On a standard diskette this will be two sectors, but the number of sectors will vary on a hard disk typically between eight and twenty sectors.

Link to Resource Fork

This command is exactly the same as the previous command, except the allocation block is linked into the resource fork of the currently selected file.

Create File

This command allows you to create a file on a volume. The created file is automatically selected for output from the Write Data command or Link commands. The file is created with the type and creator fields set to question marks, but these items can be changed using the Display File Attributes command.

Select Output File

This command allows you to select a file for output from the Write Sectors to File command or for data to be linked in using the Link to Data Fork or Link to Resource Fork commands

[For better or worse, this is the end of the documentation. -ed.]