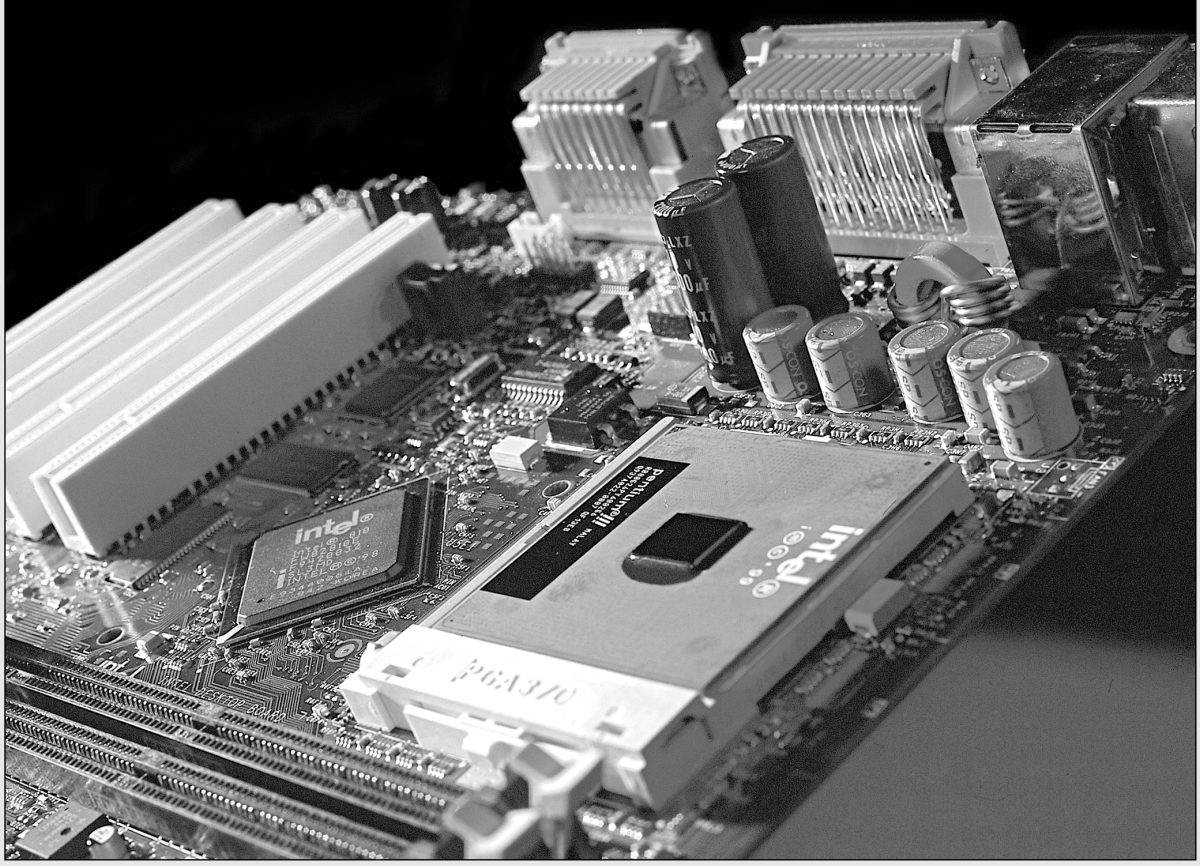
CHAPTER 24

File Systems and Data Recovery



File Systems

Physically, the hard disks and other media provide the basic technology for storing data. Logically, however, the file system provides the hierarchical structure of volumes and directories in which you store individual files and the organizational model that enables the system to locate data anywhere on a given disk or drive. File systems typically are an integrated part of an operating system (OS), and many of the newer OSes provide support for several file systems from which you can choose.

Several file systems are available from which to choose in a modern PC. Each file system has specific limitations, advantages, and disadvantages, and which ones you use can also be limited by the operating system you choose. Not all operating systems support all the file systems.

The primary file systems to choose from today include

- File allocation table (FAT), which includes FAT12, FAT16, and FAT32
- New Technology File System (NTFS)

Although other file systems, such as OS/2's High Performance File System (HPFS), are in use on PCs, these are the two you're most likely to find on a Windows-based PC. Consequently, this chapter focuses mainly on FAT and NTFS.

File Allocation Table

Until the release of Windows XP, the most commonly used file systems were based on a file allocation table (FAT), which keeps track of the data stored in each cluster on a disk. FAT is still the most universally understood file system, meaning it is recognized by virtually every operating system that runs on PCs, and even non-PCs. For example, FAT is even recognizable on Apple Mac systems. For this reason, although NTFS (covered later in this chapter) is usually recommended with Windows XP, for greater compatibility across systems and platforms, most external hard disks and removable-media drives use FAT as their native file systems. Also, if you want to dual-boot Windows XP and Windows 9x/Me, you need to use FAT-based file systems even on your main drives.

Three main varieties of the FAT system exist, called FAT12, FAT16, and FAT32—all of which are differentiated by the number of digits used in the allocation table numbers. In other words, FAT16 uses 16-bit numbers to keep track of data clusters, FAT32 uses 32-bit numbers, and so on. The various FAT systems are used as follows:

- *FAT12.* Used on all volumes smaller than 16MiB (for example, floppy disks).
- *FAT16.* Used on volumes from 16MiB through 2GiB by MS-DOS 3.0 and most versions of Windows. Windows NT, Windows 2000, and Windows XP support FAT16 volumes as large as 4GiB. However, FAT16 volumes larger than 2GiB cannot be used by MS-DOS or Windows 9x/Me.
- *FAT32.* Optionally used on volumes from 512MiB through 2GiB, and required on all FAT volumes over 2GiB, starting with Windows 95B (OSR 2.x) and subsequent versions.

FAT12 and FAT16 are the file systems originally used by DOS and Windows and are supported by every other PC operating system from past to present. An add-on to the FAT file systems called VFAT is found in Windows 95 and newer. VFAT is a driver in Windows that adds the capability to use long filenames on existing FAT systems. When running Windows 95 or newer, VFAT is automatically enabled for all FAT volumes.

Although all PC operating systems support FAT12 and FAT16, Windows 2000 and XP also have support for FAT32 as well as non-FAT file systems such as NTFS.

FAT12

FAT12 was the first file system used in the PC when it was released on August 12, 1981, and it is still used today on all floppy disks and FAT volumes less than 16MiB. FAT12 uses a table of 12-bit numbers to manage the clusters (also called *allocation units*) on a disk. A *cluster* is the storage unit in the data area of the disk where files are stored. Each file uses a minimum of one cluster, and files that are larger than one cluster use additional space in cluster increments. FAT12 cluster sizes are shown in Table 24.1.

Table 24.1 FAT12 Cluster Sizes

Media Type	Volume Size	Sectors per Cluster	Cluster Size
5 1/4'' DD floppy disk	360K	2	1KiB
3 1/2'' DD floppy disk	720K	2	1KiB
5 1/4'' HD floppy disk	1.2MB	1	0.5KiB
3 1/2'' HD floppy disk	1.44MB	1	0.5KiB
3 1/2'' ED floppy disk	2.88MB	2	1KiB
Other media	0–15.9MiB	8	4KiB

DD = Double density
HD = High density
ED = Extra-high density
KiB = Kibibyte = 1,024 bytes
MiB = Mebibyte = 1,048,576 bytes

Each cluster on a FAT12 volume is typically 8 sectors in size, except on floppy disks, where the size varies according to the particular floppy type. 12-bit cluster numbers range from 000h to FFFh (hexadecimal), which is 0–4,095 in decimal. This theoretically allows 4,096 total clusters; however, 11 of the cluster numbers are reserved and cannot be assigned to actual clusters on a disk. Cluster numbers start at 2 (0 and 1 are reserved), number FF7h is reserved to indicate a bad cluster, and numbers FF8h–FFFh indicate an end-of-chain in the FAT, leaving 4,085 clusters (4,096 – 11 = 4,085). Microsoft subtracts 1 from this to eliminate boundary problems, allowing up to exactly 4,084 clusters maximum in a FAT12 volume.

FAT12 volumes include 1 sector for the boot record and BPB (BIOS parameter block), two copies of the FAT (up to 12 sectors long each), up to 32 sectors for the root directory (less only on floppy disk media), and a data area with up to 4,084 clusters. Because each FAT12 cluster is 8 sectors (except on floppy disks), FAT12 volumes are limited to a maximum size of 32,729 sectors (1 sector for the boot record + 12 sectors per FAT × 2 FATs + 32 sectors for the root directory + 4084 clusters × 8 sectors per cluster). This equals 16.76MB or 15.98MiB. FAT12 volume limits are detailed in Table 24.2.

Table 24.2 FAT12 Volume Limits

Volume Limit	Clusters	Sectors per Cluster	Total Volume Sectors	Volume Size (Decimal)	Volume Size (Binary)
Maximum size	4,084	8	32,729	16.76MB	15.98MiB

MB = Megabyte = 1,000,000 bytes
MiB = Mebibyte = 1,048,576 bytes

PC/MS-DOS 1.x and 2.x use FAT12 exclusively, and all later versions of DOS and all Windows versions automatically create a FAT12 file system on any disks or partitions that are 32,729 sectors or less (16.76MB) in size. Anything larger than that is automatically formatted as FAT16, FAT32, or NTFS. Characteristics of FAT12 include the following:

- Is used on all floppy disks
- Has a default format on FAT volumes of 16.76MB (15.98MiB) or less

- Is supported by all versions of DOS and Windows
- Is supported by all operating systems capable of reading PC disks

FAT12 is still used in PCs today on very small media because the 12-bit tables are smaller than those for FAT16 and FAT32, which preserves the most space for data.

FAT16

FAT16 is similar to FAT12 except it uses 16-bit numbers to manage the clusters on a disk. FAT16 was introduced on August 14, 1984, along with PC/MS-DOS 3.0, with the intention of supporting larger hard drives. FAT16 picked up where FAT12 left off and was used on media or partitions larger than 32,729 sectors (15.98MiB or 16.76MB). FAT16 could theoretically support drives of up to 2GiB or 4GiB. However, even with FAT16, DOS 3.3 and earlier were still limited to a maximum partition size of 32MiB (33.55MB) because DOS 3.3 and earlier used only 16-bit sector addressing internally and in the BPB (BIOS parameter block, stored in the volume boot sector, which is the first logical sector in a FAT partition). The use of 16-bit sector values limited DOS 3.3 and earlier to supporting drives of up to 65,535 sectors of 512 bytes, which is 32MiB (33.55MB).

As a temporary way to address drives larger than 32MiB, PC/MS-DOS 3.3 (released on April 2, 1987) introduced the extended partition, which could internally support up to 23 subpartitions (logical drives) of up to 32MiB each. Combined with the primary partition on a disk, this allowed for a total of 24 partitions of up to 32MiB each, which would be seen by the operating system as logical drives C–Z.

To take full advantage of FAT16 and allow for larger drives and partition sizes, Microsoft collaborated with Compaq, who introduced Compaq DOS 3.31 in November 1987. It was the first OS to use 32-bit sector addressing internally and in the BPB. Then the rest of the PC world followed suit on July 19, 1988, when Microsoft and IBM released PC/MS-DOS 4.0. This enabled FAT16 to handle partition sizes up to 2GiB using 64 sectors per cluster.

Each cluster in a FAT16 volume is up to 64 sectors in size. 16-bit cluster numbers range from 0000h to FFFFh, which is 0–65,535 in decimal. This theoretically allows 65,536 total clusters. However, 11 of the cluster numbers are reserved and cannot be assigned to actual clusters on a disk. Cluster numbers start at 2 (0 and 1 are reserved), number FFF7h is reserved to indicate a bad cluster, and numbers FFF8h–FFFFh indicate an end of chain in the FAT, leaving 65,525 clusters (65,536 – 11 = 65,525). Microsoft subtracts 1 from this to eliminate boundary problems, allowing up to 65,524 clusters maximum in a FAT16 volume. FAT16 cluster sizes are shown in Table 24.3.

Table 24.3 FAT16 Cluster Sizes

Volume Size	Sectors per Cluster	Cluster Size
4.1MiB–15.96MiB[1]	2	1KiB
>15.96MiB–128MiB	4	2KiB
>128MiB–256MiB	8	4KiB
>256MiB–512MiB	16	8KiB
>512MiB–1GiB	32	16KiB
>1GiB–2GiB	64	32KiB
>2GiB–4GiB[2]	128	64KiB

1. Volumes smaller than 16MiB default to FAT12; however, FAT32 can be forced by altering the format parameters.

2. Volumes larger than 2GiB are supported only by Windows NT/2000/XP and are not recommended.

MB = Megabyte = 1,000,000 bytes
KiB = Kibibyte = 1,024 bytes
MiB = Mebibyte = 1,024KiB = 1,048,576 bytes
GiB = Gibibyte = 1,024MiB = 1,073,741,824 bytes

FAT16 volumes include 1 sector for the boot record and BPB, two copies of the FAT (default and backup) up to 256 sectors long each, 32 sectors for the root directory, and a data area with 4,085–65,524 clusters. Each FAT16 cluster can be up to 64 sectors (32KiB) in size, meaning FAT16 volumes are limited to a maximum size of 4,194,081 sectors (1 sector for the boot record + 256 sectors per FAT × 2 FATs + 32 sectors for the root directory + 65,524 clusters × 64 sectors per cluster). This equals a maximum capacity of 2.15GB or 2GiB. FAT16 volume limits are shown in Table 24.4.

Note

Windows NT/2000/XP can optionally create FAT16 volumes that use 128 sectors per cluster (64KiB) in size, bringing the maximum volume size to 4.29GB or 4GiB. However any volumes formatted in that manner are not readable in virtually any other OS. Additionally, 64KiB clusters cause many disk utilities to fail. For maximum compatibility, FAT16 volumes should be limited to 32KiB clusters and 2.15GB/2GiB in size.

Table 24.4 FAT16 Volume Limits

Volume Limit	Clusters	Sectors per Cluster	Total Volume Sectors	Volume Size (Decimal)	Volume Size (Binary)
Minimum size	4,167	2	8,401	4.3MB	4.1MiB
DOS 3.0-3.3 max.	16,343	4	65,533	33.55MB	32MiB
Win9x/Me max.	65,524	64	4,194,081	2.15GB	2GiB
NT/2000/XP max.	65,524	128	8,387,617	4.29GB	4GiB

MB = Megabyte = 1,000,000 bytes
GB = Gigabyte = 1,000 MB = 1,000,000,000 bytes
MiB = Mebibyte = 1,048,576 bytes
GiB = Gibibyte = 1,024MiB = 1,073,741,824 bytes

Some notable characteristics and features of FAT16 include

- FAT16 is fully supported by MS-DOS 3.31 and higher, all versions of Windows, and some Unix operating systems.
- FAT16 is fast and efficient on volumes smaller than 256MiB but relatively inefficient on larger volumes because the cluster size becomes much larger than with FAT32 and NTFS.
- The boot sector information is not automatically backed up, and if damaged or destroyed, access to the volume is lost.
- In case of a problem, you can boot the system using any MS-DOS bootable floppy to troubleshoot the problem and if necessary, repair the volume. Many third-party software tools can repair or recover data from FAT16 volumes.
- The root directory (folder) can handle up to a maximum of 512 entries, which is further reduced if any root entries use long filenames.
- FAT16 has no built-in security, encryption, or compression capability.
- File sizes on a FAT16 volume are limited only by the size of the volume. Because each file takes a minimum of one cluster, FAT16 volumes cannot have more than 65,524 total files.

FAT32

FAT32 is an enhanced version of the FAT file system first supported by Windows 95B (also known as OEM Service Release 2, released in August 1996). FAT32 is also supported in Windows 98/Me and Windows 2000/XP. FAT32 is not supported in the original release of Windows 95 or in any release of Windows NT.

One of the main reasons for creating FAT32 was to use disk space more efficiently. FAT32 uses smaller clusters (4KiB clusters for drives up to 8GiB in size), resulting in a 10%–15% more efficient use of disk space relative to large FAT16 drives. FAT32 also supports partitions of up to 2TiB in size, much larger than the 2GiB limit of FAT16. FAT32 cluster sizes are shown in Table 24.5.

Table 24.5 FAT32 Cluster Sizes

Volume Size	Sectors per Cluster	Cluster Size
32.52MiB–260MiB[1]	1	0.5KiB
>260MiB–8GiB	8	4KiB
>8GiB–16GiB	16	8KiB
>16GiB–32GiB	32	16KiB
>32GiB–2TiB[2]	64	32KiB

1. Volumes smaller than 512MiB will default to FAT16, although FAT32 can be forced by altering the format parameters.

2. Windows 2000 and XP format FAT32 volumes only up to 32GiB; however, they support existing FAT32 volumes up to 2TiB.

MB = Megabyte = 1,000,000 bytes
GB = Gigabyte = 1,000MB = 1,000,000,000 bytes
TB = Terabyte = 1,000GB = 1,000,000,000,000 bytes
KiB = Kibibytes = 1,024 bytes
MiB = Mebibyte = 1,024KiB = 1,048,576 bytes
GiB = Gibibyte = 1,024MiB = 1,073,741,824 bytes
TiB = Tebibytes = 1,024GiB = 1,099,511,627,776 bytes

Although the name implies that FAT32 uses 32-bit numbers to manage clusters (allocation units) on a disk, FAT32 actually uses only the first 28 bits of each 32-bit entry, leaving the high 4 bits reserved. The only time the high 4 bits are changed is when the volume is formatted, at which time the whole 32-bit entry is zeroed (including the high 4 bits). The high 4 bits are subsequently ignored when reading or writing FAT32 cluster entries; therefore, if those bits are non-zero, they are preserved. Microsoft has never indicated any purpose for them other than simply being reserved.

So, although the entries are technically 32-bit numbers, FAT32 really uses 28-bit numbers to manage the clusters on a disk. Each cluster on a FAT32 volume is from 1 sector (512 bytes) to 64 sectors (32KiB) in size. 28-bit cluster numbers range from 0000000h to FFFFFFFh, which is 0–268,435,455 in decimal. This theoretically allows for 268,435,456 total clusters. However 11 of the numbers are reserved and cannot be assigned to actual clusters on a disk. Cluster numbers start at 2 (0 and 1 are reserved), the number FFFFFF7h is reserved to indicate a bad cluster, and numbers FFFFFF8h–FFFFFFFh indicate an end of chain in the FAT, leaving exactly 268,435,445 clusters maximum (268,435,456 – 11 = 268,435,445).

FAT32 volumes reserve the first 32 sectors for the boot record, which includes both the default boot record (3 sectors long, starting at logical sector 0) and a backup boot record (also 3 sectors long, but starting at logical sector 6). The remaining sectors in that area are reserved and filled with 0s. Following the 32 reserved sectors are 2 FATs (default and backup) that can be anywhere from 512 sectors to 2,097,152 sectors in length, with a data area 65,525–268,435,445 clusters.

Each FAT32 cluster is up to 64 sectors (32KiB) in size, so FAT32 disks or partitions could theoretically be up to 17,184,062,816 sectors (32 sectors for the boot record + 2,097,152 sectors per FAT × 2 FATs + 268,435,445 clusters × 64 sectors per cluster), which equals a capacity of 8.8TB or 8TiB. This capacity is theoretical because the 32-bit sector numbering scheme used in the partition tables located in the master boot record (MBR) limits a disk to no more than 4,294,967,295 (2^{32}–1) sectors, which is 2.2TB or 2TiB. Therefore, although FAT32 can in theory handle a volume of up to 8.8TB (terabytes or trillions of bytes), the reality is that we are currently limited by the partition table format of the MBR to only 2.2TB. At the current rate of hard disk capacity growth, single drives of that size will debut between the years 2009 and 2011. Of course, by that time the MBR limitation should be addressed.

Note

If you attempt to format a FAT32 volume larger than 32GiB on a system running Windows 2000 or Windows XP, the format fails near the end of the process with the following error:

```
Logical Disk Manager: Volume size too big.
```

This is by design. The format tools included with Windows 2000 and XP will not format a volume larger than 32GiB using the FAT32 file system. Only the format tools are restricted as such; any preexisting volumes up to 2TiB are otherwise fully supported. Microsoft is attempting to force you to use NTFS on any newly created volumes larger than 32GiB. If you are formatting an external USB or FireWire drive that will be moved between various systems, some of which include Windows 98 or Me, you must format the drive on a Windows 98 or Me system.

Windows 95 OSR2 and Windows 98 have additional limitations with FAT32. The ScanDisk tool included with those operating systems is a 16-bit program that has a maximum allocation size for a single memory block of 16MB less 64KB. This means that ScanDisk cannot process volumes using the FAT32 file system that have a FAT larger than 16MB less 64KB in size. Each FAT32 entry uses 4 bytes, so ScanDisk cannot process the FAT on a volume using the FAT32 file system that defines more than 4,177,920 clusters, which after subtracting the two reserved clusters leaves 4,177,918 actual clusters. At 32KiB per cluster—including the boot sector reserved area and the FATs themselves—this results in a maximum volume size of 136.94GB or 127.53GiB. Windows Me and later do not have this limitation.

Table 24.6 lists the volume limits for a FAT32 file system.

Table 24.6 FAT32 Volume Limits

Volume Limit	Clusters	Sectors per Cluster	Total Volume Sectors	Volume Size (Decimal)	Volume Size (Binary)
Minimum size	65,535	1	66,601	34.1MB	32.52MiB
Win9x max.	4,177,918	64	267,452,072	136.94GB	127.53GiB
MBR max.	67,092,481	64	4,294,967,266	2.2TB	2TiB
Theoretical max.	268,435,445	64	17,184,062,816	8.8TB	8TiB

MB = Megabyte = 1,000,000 bytes
MiB = Mebibyte = 1,024KiB = 1,048,576 bytes
GB = Gigabyte = 1,000MB = 1,000,000,000 bytes
GiB = Gibibyte = 1,024MiB = 1,073,741,824 bytes
TB = Terabyte = 1,000GB = 1,000,000,000,000 bytes
TiB = Tebibytes = 1,024GiB = 1,099,511,627,776 bytes
KiB = Kibibytes = 1,024 bytes

FAT32 is more robust than FAT12 or FAT16. FAT12/16 uses the first sector of a volume for the volume boot record, which is a critical structure. If the boot sector is damaged or destroyed, access to the entire volume is lost. FAT32 improves on this by creating both a default and a backup volume boot record in the first 32 sectors of the volume, which are reserved for this purpose. Each FAT32 volume boot record is 3 sectors long. The default boot record is in logical sectors 0–2 (the first three sectors) in the partition, and the backup is in sectors 6–8. This feature has saved me on several occasions when the default boot record was damaged and access to the entire volume was lost. In these situations, I was still able to recover the entire volume by manually restoring the volume boot record from the backup by using a sector editor such as Norton Diskedit (included with Norton SystemWorks by Symantec). Because FAT12 and FAT16 do not create a backup boot sector, if the boot sector is destroyed on those volumes, it must be re-created manually from scratch—a much more difficult proposition.

As with FAT12/16, FAT32 also maintains two copies of the FAT and automatically switches to the backup FAT if a sector in the default FAT becomes unreadable. The root directory (folder) in a FAT16 system is exactly 32 sectors long and immediately follows the two FAT copies; however, in FAT32 the root directory is actually created as a subdirectory (folder) that is stored as a file, relocatable to anywhere in the

partition, and extendable in length. Therefore, a 512-file limit no longer exists in the root directory for FAT32 as there was with FAT12/16.

Some notable characteristics and features of FAT32 include the following:

- FAT32 is fully supported by Windows 95B (OSR2), 98, Me, 2000, XP, and later versions.
- MS-DOS 6.22 and earlier, Windows 95a, and Windows NT do not support FAT32 and cannot read or write FAT32 volumes.
- Mac OS 8.1 and later support FAT32 drives.
- FAT32 is the ideal format for large, external USB or FireWire drives that will be moved between various PC and Mac systems.
- The root directory (folder) is stored as a subdirectory file that can be located anywhere on the volume. Subdirectories (folders) including the root can handle up to 65,534 entries, which is further reduced if any entries use long filenames.
- FAT32 uses smaller clusters (4KiB for volumes up to 8GiB), so space is allocated much more efficiently than FAT16.
- The critical boot sector is backed up at logical sector 6 on the volume.
- Windows 2000 and XP format FAT32 volumes only up to 32GiB. To format FAT32 volumes larger than 32GiB, you must format the volume on a Windows 98/Me system.
- In case of a boot problem, you must start the system using a bootable disk created using Windows 95B (OSR2), 98, Me, 2000, or XP. A limited number of third-party software tools can repair or recover data from FAT32 volumes.
- FAT32 has no built-in security, encryption, or compression capability.
- File sizes on FAT32 volumes are limited to 4,294,967,295 bytes ($2^{32} - 1$), which is 1 byte less than 4GiB in size.

NTFS

Windows NT 3.1 (the first version of NT despite the 3.1 designation) was released in August 1993 and introduced the New Technology File System (NTFS), which is unique to NT-based operating systems (including Windows 2000 and Windows XP) and is not supported by Windows 9x/Me. NTFS includes many advanced features not found in the FAT file systems.

NTFS supports larger volumes (up to 16TiB), larger files, and more files per volume than FAT. NTFS also uses smaller cluster sizes than even FAT32, resulting in more efficient use of a volume. For example, a 30GiB NTFS volume uses 4KiB clusters, whereas the same size volume formatted with FAT32 uses 16KiB clusters. Smaller clusters reduce wasted space on the volume. NTFS cluster sizes are shown in Table 24.7.

Table 24.7 NTFS Cluster Sizes

Volume Size	Sectors per Cluster	Cluster Size
16MiB–512MiB	1	0.5KiB
>512MiB–1GiB	2	1KiB
>1GiB–2GiB	4	2KiB
>2GiB–2TiB[1]	8	4KiB

1. Larger cluster sizes can be forced by altering the format parameters; however, file compression is disabled if clusters larger than 8 sectors (4KiB) are selected.

MB = Megabyte = 1,000,000 bytes
GB = Gigabyte = 1,000MB = 1,000,000,000 bytes
TB = Terabyte = 1,000GB = 1,000,000,000,000 bytes
KiB = Kibibytes = 1,024 bytes
MiB = Mebibyte = 1,024KiB = 1,048,576 bytes
GiB = Gibibyte = 1,024MiB = 1,073,741,824 bytes
TiB = Tebibytes = 1,024GiB = 1,099,511,627,776 bytes

NTFS uses a special file structure called a master file table (MFT) and metadata files. The MFT is basically a relational database that consists of rows of file records and columns of file attributes. It contains at least one entry for every file on an NTFS volume. NTFS creates file and folder records for each file and folder created on an NTFS volume. These are stored in the MFT and consume 1KiB each. Each file record contains information about the position of the file record in the MFT, as well as file attributes and any other information about the file.

NTFS was designed to manage clusters using up to 64-bit numbers, which is an astronomical amount, but the current implementations use 32-bit numbers instead. Using 32-bit numbers allows for addressing up to 4,294,967,295 clusters, each of which is typically up to 4KiB.

NTFS reserves a total of 32 sectors for a 16-sector-long default volume boot sector and a backup boot sector. The default boot sector is located at the beginning (logical sector 0) of the volume, whereas the backup boot sector is written at the logical center of the volume (if it was formatted using NT 3.51 and earlier) or at end of the volume (if it was formatted with NT 4.0 or later, including 2000 and XP).

An NTFS volume can therefore comprise up to 34,359,738,392 total sectors (32 sectors reserved for the default and backup boot sectors, plus 4,294,967,295 clusters × 8 sectors), which is 17.59TB or 16.00TiB. Note that this capacity is theoretical because the 32-bit sector numbering scheme used in the partition tables located in the MBR limits a disk to no more than 4,294,967,295 (2^{32}–1) sectors, which is 2.2TB or 2TiB. Therefore, even though NTFS can in theory handle a volume of up to 17.59TB, the reality is that this is currently limited by the partition table format of the MBR to only 2.2TB.

Windows 2000 and XP can get around this on nonbootable drives by using a new storage format called a *dynamic disk*. Windows 2000 and XP Professional (but not XP Home) offer two types of storage: basic disks and dynamic disks. *Basic disks* use the same structures as before, with an MBR on the disk containing a partition table limited to four primary partitions per disk, or three primary partitions and one extended partition with unlimited logical drives. Primary partitions and logical drives on basic disks are known as *basic volumes*.

Dynamic disks were first introduced in Windows 2000 and provide the capability to create dynamic volumes that can be simple (using only one drive), spanned (using multiple drives), or striped (using multiple drives simultaneously for increased performance). Dynamic disks use a hidden database (contained in the last megabyte of the disk) to track information about dynamic volumes on the disk and about other dynamic disks in the computer. Because each dynamic disk in a computer stores a replica of the dynamic disk database, a corrupted database on one dynamic disk can be repaired using the database on another. By spanning or striping multiple drives using dynamic disk formats, you can exceed the 2TiB limit of a single MBR-based partition.

Although the 32-bit sector numbering in the partition tables on MBR disks limits NTFS basic disks to 2TiB volumes, you can use dynamic volumes to create NTFS volumes larger than 2TiB by spanning or striping multiple basic disks to create a larger dynamic disk. Because the dynamic volumes are managed in the hidden database, they are not affected by the 2TiB limit imposed by the partition tables in the MBR. In essence, dynamic disks enable Windows 2000 and XP Pro to create NTFS volumes as large as 16TiB. The volume limits for NTFS are listed in Table 24.8.

Table 24.8 NTFS Volume Limits

Volume Limit	Clusters	Sectors per Cluster	Total Volume Sectors	Volume Size (Decimal)	Volume Size (Binary)
Minimum size	32,698	1	32,730	16.76MB	15.98MiB
Basic disk max.	536,870,908	8	4,294,967,296	2.2TB	2TiB
Dynamic disk max.	4,294,967,295	8	34,359,738,392	17.59TB	16TiB

MB = Megabyte = 1,000,000 bytes
GB = Gigabyte = 1,000MB = 1,000,000,000 bytes
TB = Terabyte = 1,000GB = 1,000,000,000,000 bytes
KiB = Kibibytes = 1,024 bytes
MiB = Mebibyte = 1,024KiB = 1,048,576 bytes
GiB = Gibibyte = 1,024MiB = 1,073,741,824 bytes
TiB = Tebibytes = 1,024GiB = 1,099,511,627,776 bytes

Some notable characteristics and features of NTFS include

- *Files are limited in size to 16TiB less 64KiB or by the size of the volume, whichever is lower.* NTFS supports up to 4,294,967,295 (2^{32}–1) files on a volume.
- *NTFS is not normally used on removable media because NTFS does not flush data to the disk immediately.* In addition, removing NTFS-formatted media without using the Safe Removal application can result in data loss. For removable media that can be ejected unexpectedly, you should use FAT12, FAT16, or FAT32 instead.
- *NTFS incorporates transaction logging and recovery techniques.* In the event of a failure, upon reboot NTFS uses its log file and checkpoint information to restore the consistency of the file system.
- *NTFS dynamically remaps clusters found to contain bad sectors and then marks the defective cluster as bad so it will no longer be used.*
- *NTFS has built-in security features.* These enable you to set permissions on a file or folder.
- *NTFS has a built-in encrypting File System (EFS).* This performs dynamic encryption and decryption as you work with encrypted files or folders, while preventing others from doing so.
- *It enables the setting of disk quotas.* You can track and control space usage for NTFS volumes among various users.
- *NTFS has built-in dynamic compression, which compresses and decompresses files as you use them.*

Disk and File System Structures

To manage files on a disk and enable all applications to see a consistent interface to the file system no matter what type of storage hardware is being used, the operating system creates several structures on the disk. These structures are the same for any OS that supports the FAT file system, including Windows 9x, Windows Me, Windows NT, Windows 2000, and Windows XP. The following list shows all the structures and areas FAT uses to manage a disk, in roughly the same order in which they appear on the media:

- Master and extended partition boot records (sectors)
- Volume boot record
- Root directory
- File allocation tables
- Clusters (allocation units in the data area)
- Diagnostic read-and-write cylinder (not on all drives)

All these structures are detailed later in this section. A hard disk has all these disk-management structures, and a floppy disk has all but the master and extended partition boot records and diagnostic cylinder. The volume boot record through data area structures are repeated for each partition or volume on a drive. These structures are created on hard disk drives or other high-capacity media by the disk partitioning program included with all operating systems. You can't use a disk partitioning program such as FDISK (MS-DOS/Windows 9x/Me) or DISKPART or Disk Management (Windows NT/2000/XP) on a floppy disk because floppy disks can't be partitioned. Figure 24.1 is a simple diagram showing the relative locations of these FAT disk-management structures on an 8.4GB hard disk.

Note

Some removable cartridge drives, such as the SuperDisk (LS-120 and LS-240) and Iomega Zip drive, function like high-capacity floppy disk drives. They lack a master boot record (MBR) and diagnostic cylinder and can't be partitioned like hard disk drives. Other higher-capacity removable drives, such as the legacy Iomega Jaz or Castlewood Orb, can be partitioned like a hard disk drive.

All PC hard drives using the FAT16 file system are similar.

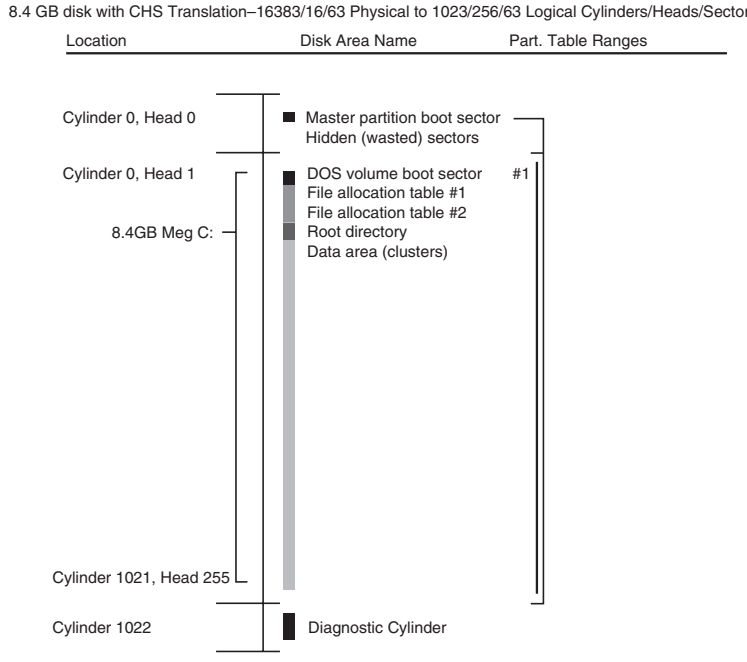


Figure 24.1 FAT16 file-management structures on a typical 8.4GB drive.

Each disk area has a purpose and function. If one of these special areas is damaged, serious consequences can result. Damage to one of these sensitive structures usually causes a domino effect, limiting access to other areas of the disk or causing further problems in using the disk. For example, the OS normally can't access a drive at all if the MBR is corrupted. Therefore, you should understand these data structures well enough to be able to repair them when necessary. Rebuilding these special tables and areas of the disk is essential to the art of data recovery.

Master Boot Record

The first PC OS to support hard disks, DOS 2.0 (released on March 8, 1983), was also the first to introduce the capability to partition a drive. *Partitioning* is basically dividing the drive into multiple volumes. One concept easily misunderstood is that all drives that *can* be partitioned *must* be partitioned; that is, you have to partition the drive even if you are going to set it up with only one partition. Another name for a partition is a *logical volume* because the partition shows up as an additional drive letter or volume to the OS.

Although the primary use for partitioning today is to divide a single drive into multiple volumes for use by the same OS, originally it was intended to allow multiple different OSes, each with different file systems, to coexist on a single drive. This multi-OS capability still exists today; however, additional aftermarket utilities often are required to manage and boot from multiple OSes on a single machine.

Tip

If you want to dual-boot Windows 9x/Me with Windows 2000/XP without purchasing aftermarket boot managers, install Windows 9x or Me first and then install Windows 2000 or XP either on the unused space on the first hard disk or in a primary partition on an additional hard disk. Windows 2000 or XP will set up a boot manager for you.

To use a hard disk with different operating systems, you can create partitions to logically divide the disk. You can, for example, create one or more FAT or NTFS partitions for use with Windows and leave the rest of the disk storage area for use by another OS's file system, such as Linux. Each of the FAT or NTFS partitions appear to an OS that supports it as a separate drive letter. For example, Windows 9x/Me ignores the unused or non-FAT partitions, whereas Windows 2000/XP sees both FAT and NTFS partitions but ignores others such as Linux and OS/2 HPFS.

Even though Windows NT, 2000, and XP have a command-line disk partitioning program called DISKPART, disk partitions are usually prepared with the GUI-based Disk Management tool (2000/XP) or Disk Administrator (NT).

◀◀ For more information about creating and formatting partitions with any of these operating systems, see Chapter 14, "Physical Drive Installation and Configuration," p. 797.

Information about each of the partitions on the disk is stored in a partition (or volume) boot record at the beginning of each partition. Additionally, a main table lists the partitions embedded in the master boot record.

The MBR, which is also sometimes called the master boot sector, is always located in the first physical sector of a disk (cylinder 0, head 0, sector 1) and consists of the following structures:

- *Bootstrap code*. The instructions used to locate and load the VBR from the active (bootable) partition.
- *Master partition table*. A table consisting of four 16-byte entries for up to four primary partitions, or three primary partitions and one extended partition. Each primary partition defines a logical drive, and an extended partition can be further partitioned into multiple logical drives. A given partition entry indicates which type of partition it is, whether it is bootable, where it is located physically on the disk, and how many sectors it occupies.
- *Signature bytes*. A 2-byte signature (55AAh) used by the motherboard ROM and other code to validate the sector.

Primary and Extended FAT Partitions

Most OSes are designed to support up to 24 volumes on a single hard disk drive (represented by the drive letters C:–Z:), but the partition table in the master boot record (MBR) can have a maximum of only four entries. This is handled by using a single primary partition, which is seen as the first logical drive (C:), and an extended partition, which is then further partitioned into additional logical drives (D:, E:, F:, and so on).

Note

Although Windows NT uses Disk Administrator and Windows 2000 and XP use Disk Management to create disk partitions instead of FDISK, the following discussion refers to FDISK for simplicity's sake. A FAT partition up to 2GiB is the same with any of these operating systems.

An *extended* partition is listed in the master partition table the same as a primary partition, but it differs in that you can use its disk space to create multiple logical partitions, or *volumes*. You can create only one extended partition on a single drive, meaning that typically there will never be more than two entries in the master partition table, one primary and one extended.

The logical volumes you create in the extended partition appear as separate drive letters to the operating system, but they are not listed in the master partition table. Volumes in the extended partition are not bootable. You can create up to 23 volumes out of a single extended partition (assuming that you have already created a primary partition, which brings the total number of volumes to 24).

Each of the subpartitions in an extended partition includes an extended partition table located in the first sector of the subpartition. The first sector of the extended partition contains an extended partition table that points to the first subpartition and, optionally, another extended partition. The first sector of that extended partition has another extended partition table that can reference another volume as

well as an additional extended partition. This chain of references continues, linking all the volumes in the extended partition to the master partition table. It is important to note that, if the entry for the extended partition in the MBR is lost or damaged, the chain will be broken at the start and all volumes contained within will be inaccessible—essentially meaning that they will dissapear.

Few people have any reason to create 24 partitions on a single disk drive, but the extended partition can create a chain of linked partitions on the disk that makes it possible to exceed the four-entry limitation of the master partition table.

Because the master boot record contains the first program loaded from disk that the system executes when you boot a PC, it is frequently a target for creators of computer viruses. A virus that infects or destroys the MBR can make it impossible for the BIOS to find the active partition, thus preventing the operating system from loading. Because the MBR contains the first program executed by the system, a virus stored there loads before any antivirus code can be loaded to detect it. To remove an MBR virus, you must first boot the system from a clean, uninfected disk, such as a floppy, bootable CD/DVD, or USB drive, and then run an antivirus program to test and possibly repair or restore the MBR.

Each partition on a disk contains a volume boot record starting in the first sector. With the FDISK or DISKPART utilities, you can designate a primary partition as active (or bootable). The master boot record bootstrap code causes the VBR from the active primary partition to receive control whenever the system is started.

Although FAT12, FAT16, FAT32, or NTFS partitions are mainly used when running Windows, you can also create additional disk partitions for Linux, Novell NetWare, OS/2's HPFS, AIX (Unix), XENIX, or other file systems or operating systems, using disk utilities provided with the alternative OS or in some cases a third-party disk partitioning tool such as PowerQuest's Partition Magic. A partition that is not recognized by a particular operating system is simply ignored. If you install multiple operating systems on a single drive, a boot manager program (which might be included with the operating systems or installed separately) can be used to allow you to select which partition to make active each time you boot the system. As another alternative, you could install different operating systems in different primary partitions and then use FDISK, DISKPART, or some other partitioning program to change the one you want to boot as active.

Table 24.9 shows the format of the master boot record and its partition tables. The table lists the fields in each of the master partition table's four entries, the location on the disk where each field begins (the offset), and its length.

Table 24.9 Master Boot Record Format

Offset (Hex)	Offset (Dec)	Name	Length	Description
000h	0	Boot Code	446	Bootstrap code; loads the VBR from the active partition.
		Partition Table Entry #1		
Offset (Hex)	**Offset (Dec)**	**Name**	**Length**	**Description**
1BEh	446	Boot Indicator	1 byte	Boot status; 80h = active (bootable). Otherwise, it's 00h.
1BFh	447	Starting Head	1 byte	Starting head (or side) of partition in CHS mode.
1C0h	448	Starting Cylinder/Sector	16 bits	Starting cylinder (10 bits) and sector (6 bits) in CHS mode.
1C2h	450	System Indicator	1 byte	Partition type/file system.
1C3h	451	Ending Head	1 byte	Ending head (or side) of partition in CHS mode.
1C4h	452	Ending Cylinder/Sector	16 bits	Ending cylinder (10 bits) and sector (6 bits) in CHS mode.
1C6h	454	Relative Sector	4 bytes	Count of sectors before partition, which is the starting sector of partition in LBA mode.
1CAh	458	Total Sectors	1 bytes	Total number of sectors in partition in LBA mode.

Table 24.9 Continued

Partition Table Entry #2

Offset (Hex)	Offset (Dec)	Description	Length	Description
1CEh	462	Boot Indicator	1 byte	Boot status; 80h = active (bootable). Otherwise, it's 00h.
1CFh	463	Starting Head	1 byte	Starting head (or side) of partition in CHS mode.
1D0h	464	Starting Cylinder/Sector	16 bits	Starting cylinder (10 bits) and sector (6 bits) in CHS mode.
1D2h	466	System Indicator	1 byte	Partition type/file system.
1D3h	467	Ending Head	1 byte	Ending head (or side) of partition in CHS mode.
1D4h	468	Ending Cylinder/Sector	16 bits	Ending cylinder (10 bits) and sector (6 bits) in CHS mode.
1D6h	470	Relative Sector	4 bytes	Count of sectors before partition, which is the starting sector of the partition in LBA mode.
1DAh	474	Total Sectors	4 bytes	Total number of sectors in the partition in LBA mode.

Partition Table Entry #3

Offset (Hex)	Offset (Dec)	Description	Length	Description
1DEh	478	Boot Indicator	1 byte	Boot status; 80h = active (bootable). Otherwise, it's 00h.
1DFh	479	Starting Head	1 byte	Starting head (or side) of partition in CHS mode.
1E0h	480	Starting Cylinder/Sector	16 bits	Starting cylinder (10 bits) and sector (6 bits) in CHS mode.
1E2h	482	System Indicator	1 byte	Partition type/file system.
1E3h	483	Ending Head	1 byte	Ending head (or side) of partition in CHS mode.
1E4h	484	Ending Cylinder/Sector	16 bits	Ending cylinder (10 bits) and sector (6 bits) in CHS mode.
1E6h	486	Relative Sector	4 bytes	Count of sectors before partition, which is the starting sector of partition in LBA mode.
1EAh	490	Total Sectors	4 bytes	Total number of sectors in partition in LBA mode.

Partition Table Entry #4

Offset (Hex)	Offset (Dec)	Description	Length	Description
1EEh	494	Boot Indicator	1 byte	Boot status; 80h = active (bootable). Otherwise, it's 00h.
1EFh	495	Starting Head	1 byte	Starting head (or side) of partition in CHS mode.
1F0h	496	Starting Cylinder/Sector	16 bits	Starting cylinder (10 bits) and sector (6 bits) in CHS mode.
1F2h	498	System Indicator	1 byte	Partition type/file system.
1F3h	499	Ending Head	1 byte	Ending head (or side) of partition in CHS mode.
1F4h	500	Ending Cylinder/Sector	16 bits	Ending cylinder (10 bits) and sector (6 bits) in CHS mode.
1F6h	502	Relative Sector	4 bytes	Count of sectors before partition, which is the starting sector of partition in LBA mode.
1FAh	506	Total Sectors	4 bytes	Total number of sectors in partition in LBA mode.

Signature Bytes

Offset (Hex)	Offset (Dec)	Description	Length	Description
1FEh	510	Signature	2 bytes	Boot sector signature; it should be 55AAh.

CHS = Cylinder head sector

LBA = Logical block address

The data in the partition table entries tells the system where each partition starts and ends on the drive, how big it is, whether it is bootable, and which type of file system is contained in the partition. The starting cylinder, head, and sector values are used only by systems running in CHS mode, which is standard for all drives of 8.4GB or less. CHS values do not work past 8.4GB and therefore cannot represent partitions on drives larger than that. Drives larger than 8.4GB can be fully addressed only in LBA mode. In that case, the starting cylinder, head, and sector values in the table are ignored, and only the Relative Sector and Total Sectors fields are used. The Relative Sector field indicates the precise LBA where the partition begins, and the Total Sectors field indicates the length, which is always contiguous. Thus, from those two values the system can know exactly where a partition is physically located on a disk.

Note

The processors on which the PC is based have a design characteristic that is important to know for anybody editing or interpreting boot sectors. Numbers larger than 1 byte are actually read backward! This is called *little endian format* (as in reading the number from the little end first) or *reverse-byte ordering*. People typically read numbers in *big endian format*, which means from left to right, from the big end first. However, because PC processors read in little endian format, most numeric values larger than 1 byte are stored so that the least significant byte appears first and the most significant byte appears last. For example, the value for the Relative Sector field in the MBR for the first partition is usually 63, which is 3Fh in hex, or 0000003Fh (4 bytes long) in standard big endian hexadecimal format. However, the same number stored in little endian format would appear as 3F000000h. As another example, if a partition had 23,567,292 total sectors (about 12GB), which is 01679BBCh in hexadecimal, the number would be stored in the MBR partition table Total Sectors field in reverse-byte/little endian format as BC9B6701h.

As an aside, the use of reverse-byte order numbers stems from the way processors evolved from 8-bit (1 byte) designs to 16-bit (2 byte), 32-bit (4 byte) designs and beyond. The way the internal registers are organized and implemented dictates how a processor deals with numbers. Many processors, such as the Motorola PowerPC chips used in Macintosh systems, read numbers in big endian format. PC processors, on the other hand, are all based on Intel designs dating back to the original Intel 8088 processor used in the first IBM PC. Of course, how a particular processor reads numbers doesn't make any difference to those using a system. In the PC, the only people who have to deal with reverse-byte order or little endian numbers directly are machine or assembly language programmers—and those who want to edit or interpret raw boot sectors!

Each partition table entry contains a system indicator byte that identifies the type of partition and file system used in the partition referenced by that entry. Table 24.10 shows the standard values and meanings of the system indicator bytes, and Table 24.11 lists the nonstandard values.

Table 24.10 Standard System Indicator Byte Values

Value	Partition Type	Address Mode	Partition Size
00h	None	—	—
01h	Primary FAT12	CHS	0–16MiB
04h	Primary FAT16	CHS	16MiB–32MiB
05h	Extended	CHS	0–2GiB
06h	Primary FAT16	CHS	32MiB–2GiB
07h	NTFS/HPFS	All	All
0Bh	Primary FAT32	CHS	512MiB–2TiB
0Ch	Primary FAT32	LBA	512MiB–2TiB
0Eh	Primary FAT16	LBA	32MiB–2GiB
0Fh	Extended	LBA	2GiB–2TiB

CHS = Cylinder head sector

LBA = Logical block address

Table 24.11 Nonstandard System Indicator Byte Values

Value	Partition Type	Value	Partition Type
02h	MS-XENIX Root	75h	IBM PC/IX
03h	MS-XENIX usr	80h	Minix v.1.1–v1.4a
08h	AIX File System Boot	81h	Minix v1.4b-up or Linux
09h	AIX Data	82h	Linux swap file
0Ah	OS/2 Bootmanager	83h	Linux native file system
12h	Compaq diagnostics	93h	Amoeba file system
40h	ENIX 80286	94h	Amoeba bad block table
50h	Ontrack Disk Manager read-only DOS	B7h	BSDI file system (secondary swap)
51h	Ontrack Disk Manager read/write DOS	B8h	BSDI file system (secondary file system)
52h	CP/M or Microport System V/386	DBh	DR Concurrent DOS/CPM-86/CTOS
54h	Ontrack Disk Manager non-DOS	E1h	SpeedStor 12-bit FAT extended
55h	Micro House EZ-Drive non-DOS	E4h	SpeedStor 16-bit FAT extended
56h	Golden Bow Vfeature Deluxe	F2h	DOS 3.3+secondary
61h	Storage Dimensions SpeedStor	F4h	SpeedStor primary
63h	IBM 386/ix or Unix System V/386	FEh	LANstep
64h	Novell NetWare 286	FFh	Unix/Xenix Bad Block Table Partition
65h	Novell NetWare 386		

These values can be useful for somebody trying to manually repair a partition table using a disk editor such as the Disk Edit program included with Norton Utilities (now part of Norton SystemWorks).

Undocumented FDISK

FDISK is a very powerful program. In DOS 5 and later versions, including Windows 9x/Me, it gained some additional capabilities (Windows NT uses Disk Administrator, whereas Windows 2000/XP use Disk Management or the DISKPART program to perform the functions of FDISK). Unfortunately, these capabilities were never documented in any of the Microsoft documentation for Windows or DOS. The most important undocumented parameter in FDISK is the `/MBR` (master boot record) parameter, which causes FDISK to rewrite the master boot record code area, leaving the partition table area intact.

The `/MBR` parameter is tailor-made for eliminating boot sector virus programs that infect the master boot record (located at cylinder 0, head 0, sector 1) of a hard disk. To use this feature, enter the following:

```
FDISK /MBR
```

FDISK then rewrites the boot record code, leaving the partition tables intact. This should not cause any problems on a normally functioning system, but just in case, I recommend backing up the partition table information to floppy disk before trying it. You can do this by using a third-party product such as Norton Utilities.

Be aware that using FDISK with the `/MBR` switch overwrites the partition tables if the two signature bytes at the end of the sector (55AAh) are damaged. This situation is highly unlikely, however. In fact, if these signature bytes were damaged, you would know—the system would not boot and would act as though there were no partitions at all. If you are unable to access your hard disk after booting from a clean floppy or removable-media drive, your system might be infected with a boot sector virus. You should scan for viruses with an up-to-date antivirus program and use it to guide repair.

Caution

Also note that **FDISK /MBR** should be used only on systems using the normal master boot record structure. If a disk-management program such as Disk Manager, Disc Wizard, EZ-Drive, MaxBlast, Data Lifeguard Tools, or similar is being used to allow your system to access the drive's full capacity, do not use **FDISK /MBR** because these programs use a modified MBR for disk access. Using **FDISK /MBR** will wipe out the changes they made to your drive and could make your data inaccessible.

The equivalent Windows NT/2000/XP Recovery Console feature to DOS/Windows 9x/Me's `FDISK /MBR` is called `FIXMBR`. The Recovery Console equivalent to FDISK is `DISKPART`. For details on the use of these commands, type **HELP** after loading the Recovery Console.

Volume Boot Records

The volume boot record is the first sector on any area of a drive addressed as a volume, including primary partitions or logical volumes inside an extended partition. On a floppy disk or removable cartridge (such as a Zip disk), for example, the volume boot record starts at the physical beginning of the disk because the disk is recognized as a volume without the need for partitioning. On a hard disk, the volume boot record is located as the first sectors within any disk area allocated as a primary partition, or as a logical drive (volume) inside an extended partition. Refer to Figure 24.1 for an idea of the physical relationship between this volume boot record and the other data structures on a disk. The volume boot record resembles the master boot record in that it contains the following elements:

- *Jump Instruction to Boot Code.* A 3-byte Intel x86 unconditional branch (or jump) instruction that jumps to the start of the operating system bootstrap code within the sector.
- *BIOS Parameter Block.* Contains specific information about the volume, such as its size, the number of disk sectors it uses, the size of its clusters, and the volume label name. Used by the file system driver to determine the type and status of the media. Varies according to the type of file system on the media.
- *Boot Code.* The instructions used to locate and load the initial operating system kernel or startup file, usually either `IO.SYS` or `NTLDR`.
- *Signature Bytes.* A two-byte signature (55AAh) used by the motherboard ROM and other code to validate the boot sector.

Either the motherboard ROM or the master boot record on a hard disk loads the volume boot record of the active partition on a disk. The program code in the volume boot record is given control of the system; it performs some tests and then attempts to load the first operating system file (in DOS/Windows 9x/Me the file is `IO.SYS` and in Windows NT/2000/XP the file is `NTLDR`). The volume boot record, similar to the master boot record, is transparent to the running system; it is outside the data area of the disk on which files are stored.

Note

Many of today's systems are capable of booting from drives other than standard floppy disk and hard disk drives. In these cases, the system BIOS must specifically support the boot drive. For example, some BIOS products enable you to select an ATAPI CD-ROM (or DVD) as a boot device, in addition to the floppy and hard disk drives. Many can also boot from drives connected to USB ports, adding even more flexibility to the system.

Other types of removable media, such as Zip cartridges and LS-120 disks, can also be made bootable. When the BIOS properly supports it, an LS-120 drive can replace the existing floppy disk drive as drive A:. Check the setup screens in your system BIOS to determine which types of drives can be used to start your system.

The VBR is typically created on a volume when the volume is high-level formatted. This can be done with the `FORMAT` command included with DOS and Windows, or you can also use Windows NT's Disk

Administrator and Windows 2000/XP's Disk Management programs to perform this task after partitioning the disk. All volumes have a VBR starting in the first sector of the volume.

The VBR contains both program code and data. The single data table in this sector is called the media parameter block or disk parameter block. The operating system needs the information this table contains to verify the capacity of the disk volume as well as the location of important structures, such as the FATs on FAT volumes or the Master File Table on NTFS volumes. The format of this data is very specific.

Although all VBRs contain boot code in addition to the BIOS parameter block (BPB) and other structures, only the boot code from the VBR in the bootable volume is executed. The others are read by the operating system during startup to determine the volume parameters.

The VBR on FAT12 and FAT16 volumes is 1 sector long and contains the jump instruction, the main BPB, bootstrap code, and signature bytes. Table 24.12 shows the format and layout of the FAT12/16 VBR.

Table 24.12 FAT12/16 Volume Boot Record Format

Offset (Hex)	Offset (Dec)	Name	Length (Bytes)	Description
000h	0	BS_jmpBoot	3	Jump instruction to boot code, usually EB3C90h.
003h	3	BS_OEMName	8	OEM ID. Indicates which system formatted the volume. Typically, it's `MSWIN4.1`. Not used by the OS after formatting.
00Bh	11	BPB_BytsPerSec	2	Bytes per sector; normally 512.
00Dh	13	BPB_SecPerClus	1	Sectors per cluster. It must be a power of 2 greater than 0; typically 1, 2, 4, 8, 16, 32, or 64.
00Eh	14	BPB_RsvdSecCnt	2	Number of sectors reserved for the boot record(s); it should be 1 on FAT12/16 volumes.
010h	16	BPB_NumFATs	1	Count of FAT structures on the volume; usually 2.
011h	17	BPB_RootEntCnt	2	Count of 32-byte directory entries in the root directory of FAT12 and FAT16 volumes; it should be 512 on FAT12/16 volumes.
013h	19	BPB_TotSec16	2	16-bit total count of sectors on volumes with less than 65,536 sectors. If 0, then BPB_TotSec32 contains the count.
015h	21	BPB_Media	1	Media descriptor byte; normally F8h on all nonremovable media, and F0h on most removable media.
016h	22	BPB_FATSz16	2	FAT12/16 16-bit count of sectors occupied by one FAT.
018h	24	BPB_SecPerTrk	2	Sectors per track geometry value for interrupt 13h; it's usually 63 on hard disks.
01Ah	26	BPB_NumHeads	2	Number of heads for interrupt 13h; it's usually 255 on hard disks.
01Ch	28	BPB_HiddSec	4	Count of hidden sectors preceding the partition that contains this volume; it's usually 63 for the first volume.
020h	32	BPB_TotSec32	4	32-bit total count of sectors on volumes with 65,536 or more sectors. If 0, then BPB_TotSec16 contains the count.
024h	36	BS_DrvNum	1	Int 13h drive number; it's usually 00h for floppy disks or 80h for hard disks.
025h	37	BS_Reserved1	1	Reserved (used by Windows NT); it should be 0.
026h	38	BS_BootSig	1	Extended boot signature; it should be 29h if the following three fields are present. Otherwise, it's 00h.

Table 24.12 Continued

Offset (Hex)	Offset (Dec)	Name	Length (Bytes)	Description
027h	39	BS_VolID	4	Volume serial number; used with BS_VolLab to support volume tracking on removable media. Normally generated using the date and time as a seed when the volume is formatted.
02Bh	43	BS_VolLab	11	Volume label. Matches the 11-byte volume label recorded in the root directory; it should be set to `NO NAME` if there is no volume label.
036h	54	BS_FilSysType	8	Should be `FAT12`, `FAT16`, or `FAT`. Not used by the OS after formatting.
03Eh	62	BS_BootCode	448	Bootstrap program code.
1FEh	510	BS_Signature	2	Signature bytes; should be 55AAh.

The VBR on a FAT32 volume is 3 sectors long, although 32 sectors are reserved at the beginning of the volume for the default and backup VBRs. The default VBR is in sectors 0, 1, and 2, and the backup VBR is in sectors 6, 7, and 8. These are all created at the time the volume is formatted and do not change during normal use. The first sector contains a jump instruction, the BPB, initial bootstrap code, and signature bytes. The second sector is called the FSInfo (file system information) sector and contains signature bytes and information used to assist the file system software; the third sector contains only additional bootstrap code and signature bytes. Table 24.13 shows the format and layout of the first sector of the 3-sector long FAT32 VBR.

Table 24.13 FAT32 VBR Format, BPB Sector 0

Offset (Hex)	Offset (Dec)	Name	Length (Bytes)	Description
000h	0	BS_jmpBoot	3	Jump instruction to boot code; it's usually EB5890h.
003h	3	BS_OEMName	8	OEM ID; indicates which system formatted the volume. It's typically `MSWIN4.1`. Not used by the OS after formatting.
00Bh	11	BPB_BytsPerSec	2	Bytes per sector; normally 512.
00Dh	13	BPB_SecPerClus	1	Sectors per cluster; it must be a power of 2 greater than 0. It's normally 1, 2, 4, 8, 16, 32, or 64.
00Eh	14	BPB_RsvdSecCnt	2	Number of sectors reserved for the boot record(s); it should be 32 on FAT32 volumes.
010h	16	BPB_NumFATs	1	Count of FAT structures on the volume; usually 2.
011h	17	BPB_RootEntCnt	2	Count of 32-byte directory entries in the root directory of FAT12 and FAT16 volumes; should be 0 on FAT32 volumes.
013h	19	BPB_TotSec16	2	16-bit total count of sectors on volumes with less than 65,536 sectors. If 0, then BPB_TotSec32 contains the count. Must be 0 for FAT32 volumes.
015h	21	BPB_Media	1	Media descriptor byte, normally F8h on all non-removable media, F0h on most removable media.
016h	22	BPB_FATSz16	2	FAT12/16 16-bit count of sectors occupied by one FAT; it should be 0 on FAT32 volumes, and BPB_FATSz32 contains the FAT size count.

Table 24.13 Continued

Offset (Hex)	Offset (Dec)	Name	Length (Bytes)	Description
018h	24	BPB_SecPerTrk	2	Sectors per track geometry value for interrupt 13h; usually 63 on hard disks.
01Ah	26	BPB_NumHeads	2	Number of heads for interrupt 13h; usually 255 on hard disks.
01Ch	28	BPB_HiddSec	4	Count of hidden sectors preceding the partition that contains this volume; usually 63 for the first volume.
020h	32	BPB_TotSec32	4	32-bit total count of sectors on volumes with 65,536 or more sectors. If 0, then BPB_TotSec16 contains the count. Must be non-zero on FAT32 volumes.
024h	36	BPB_FATSz32	4	FAT32 32-bit count of sectors occupied by one FAT. BPB_FATSz16 must be 0.
028h	40	BPB_ExtFlags	2	FAT32 only: Bits 0–3. Zero-based number of active FAT. Valid only if mirroring is disabled (bit 7 = 1). Bits 4–6. Reserved. Bit 7. 0 indicates FAT is mirrored; 1 indicates only the FAT referenced in bits 0–3 is active. Bits 8–15. Reserved.
02Ah	42	BPB_FSVer	2	Version number of the FAT32 volume. A high byte is a major revision number; a low byte is a minor revision number. It should be 00h:00h.
02Ch	44	BPB_RootClus	4	Cluster number of the first cluster of the root directory; usually 2.
030h	48	BPB_FSInfo	2	Sector number of extended FSInfo boot sector structure in the reserved area of the FAT32 volume; usually 1.
032h	50	BPB_BkBootSec	2	Sector number of the backup copy of the boot record; it's usually 6.
034h	52	BPB_Reserved	12	Reserved; should be 0.
040h	64	BS_DrvNum	1	Int 13h drive number; it's usually 00h for floppy disks or 80h for hard disks.
041h	65	BS_Reserved1	1	Reserved (used by Windows NT); it should be 0.
042h	66	BS_BootSig	1	Extended boot signature; it should be 29h if the following three fields are present. Otherwise, it's 00h.
043h	67	BS_VolID	4	Volume serial number; used with BS_VolLab to support volume tracking on removable media. Normally generated using the date and time as a seed when the volume is formatted.
047h	71	BS_VolLab	11	Volume label. Matches the 11-byte volume label recorded in the root directory, should be `NO NAME` if there is no volume label.
052h	82	BS_FilSysType	8	Should be `FAT32`. Not used by the OS after formatting.
05Ah	90	BS_BootCode	420	Bootstrap program code.
1FEh	510	BS_Signature	2	Signature bytes; it should be 55Aah.

Table 24.14 shows the format and layout of the FAT32 FSInfo sector, which is the second sector of the 3-sector-long FAT32 volume boot record.

Table 24.14 FAT32 VBR Format, FSInfo Sector 1

Offset (Hex)	Offset (Dec)	Name	Length (Bytes)	Description
000h	0	FSI_LeadSig	4	Lead signature, validates sector; it should be 52526141h.
004h	4	FSI_Reserved1	480	Reserved; it should be 0.
1E4h	484	FSI_StrucSig	4	Structure signature; it validates sector and should be 72724161h.
1E8h	488	FSI_Free_Count	4	Last known free cluster count on the volume. If FFFFFFFFh, the free count is unknown and must be recalculated by the OS.
1ECh	492	FSI_Nxt_Free	4	Next free cluster; it indicates where the system should start looking for free clusters. Usually set to the last cluster number allocated. If the value is FFFFFFFFh, the system should start looking at cluster 2.
1F0h	496	FSI_Reserved2	12	Reserved; it should be 0.
1FCh	508	FSI_TrailSig	4	Trailing signature; it should be 000055AAh.

Table 24.15 shows the format and layout of the FAT32 Boot Code sector, which is the third and final sector of the 3-sector-long FAT32 volume boot record.

Table 24.15 FAT32 VBR Format, Boot Code Sector 2

Offset (Hex)	Offset (Dec)	Name	Length (Bytes)	Description
000h	0	BS_BootCode	510	Boot program code
1FEh	510	BS_Signature	2	Signature bytes; should be 55AAh

It is interesting to note that this third sector has no system-specific information in it, which means the contents are the same from system to system. Thus, if this sector (and its backup at LBA 8) were damaged on one system, you could obtain a copy of this sector from any other FAT32 volume and use it to restore the damaged sector.

The VBR on NTFS volumes is 7 sectors long, although 16 sectors are reserved at the beginning of the disk for the VBR. A backup of the 16 sector VBR area is reserved at the end of the volume, which contains a backup VBR. The first sector of the 7 is the BPB sector, and it contains a jump instruction, the BPB, and signature bytes. Sectors 2–7 contain only additional boot code, with no signature bytes or any other structures. Because the boot code is not system specific, all but the first VBR sector should be the same on any NTFS volume. Table 24.16 shows the format and layout of the first sector of the 7-sector-long NTFS VBR.

Table 24.16 NTFS VBR Format, BPB Sector 0

Offset (Hex)	Offset (Dec)	Name	Length (Bytes)	Description
000h	0	BS_jmpBoot	3	Jump instruction to boot code; it's usually EB5290h.
003h	3	BS_OEMName	8	OEM ID; indicates which system formatted the volume. Typically, it's NTFS. Not used by the OS after formatting.
00Bh	11	BPB_BytsPerSec	2	Bytes per sector; it's usually 512.
00Dh	13	BPB_SecPerClus	1	Sectors per cluster; must be a power of 2 greater than 0. It's normally 1, 2, 4, or 8.

Table 24.16 Continued

Offset (Hex)	Offset (Dec)	Name	Length (Bytes)	Description
00Eh	14	BPB_RsvdSecCnt	2	Reserved sectors before the VBR; the value must be 0 or NTFS fails to mount the volume.
010h	16	BPB_Reserved	3	Value must be 0 or NTFS fails to mount the volume.
013h	19	BPB_Reserved	2	Value must be 0 or NTFS fails to mount the volume.
015h	21	BPB_Media	1	Media descriptor byte; it's normally F8h on all non-removable media and F0h on most removable media.
016h	22	BPB_Reserved	2	Value must be 0 or NTFS fails to mount the volume.
018h	24	BPB_SecPerTrk	2	Sectors per track geometry value for interrupt 13h; usually 63 on hard disks.
01Ah	26	BPB_NumHeads	2	Number of heads for interrupt 13h; usually 255 on hard disks.
01Ch	28	BPB_HiddSec	4	Count of hidden sectors preceding the partition that contains this volume; normally 63 for the first volume.
020h	32	BPB_Reserved	4	Value must be 0 or NTFS fails to mount the volume.
024h	36	Reserved	4	Not used or checked by NTFS; it's normally 80008000h.
028h	40	BPB_TotSec64	8	Total count of sectors on the volume.
030h	48	BPB_MftClus	8	Logical cluster number for the start of the `$MFT` file.
038h	56	BPB_MirClus	8	Logical cluster number for the start of the `$MFTMirr` file.
040h	64	BPB_ClusPerMft	1	Clusters per MFT file/folder record. If this number is positive (00h–7Fh), it represents clusters per MFT record. If the number is negative (80h–FFh), the size of the record is 2 raised to the absolute value of this number.
041h	65	Reserved	3	Not used by NTFS.
044h	68	BPB_ClusPerIndx	1	Clusters per index buffer; it's used to allocate space for directories. If this number is positive (00h–7Fh), it represents clusters per MFT record. If the number is negative (80h–FFh), the size of the record is 2 raised to the absolute value of this number.
045h	69	Reserved	3	Not used by NTFS.
048h	72	BS_VolID	8	Volume serial number; used to support volume tracking on removable media. Normally generated using the date and time as a seed when the volume is formatted.
050h	80	Reserved	4	Not used by NTFS.
054h	84	BS_BootCode	426	Bootstrap program code.
1FEh	510	BS_Signature	2	Signature bytes; should be 55AAh.

Root Directory

A *directory* is a simple database containing information about the files stored on a FAT partition. Each record in this database is 32 bytes long, with no delimiters or separating characters between the fields or records. A directory stores almost all the information that the operating system knows about a file, including the following:

- *Filename and extension.* The eight-character name and three-character extension of the file. The dot between the name and the extension is implied but not included in the entry.

Note

To see how Windows 9x/Me extends filenames to allow 255 characters within the 8.3 directory structure, see the section "VFAT and Long Filenames," later in this chapter.

- *File attribute byte.* The byte containing the flags representing the standard DOS file attributes, using the format shown in Table 24.18.
- *Date/Time of last change.* The date and time that the file was created or last modified.
- *File size.* The size of the file, in bytes.
- *Link to start cluster.* The number of the cluster in the partition where the beginning of the file is stored. To learn more about clusters, see the section "Clusters (Allocation Units)," later in this chapter.

Other information exists that a directory does not contain about a file. This includes where the rest of its clusters in the partition are located and whether the file is contiguous or fragmented. This information is contained in the FAT.

Two basic types of directories exist: the root directory (also called the root folder) and subdirectories (also called folders). Any given volume can have only one root directory. The root directory is always stored on a disk in a fixed location immediately following the two copies of the FAT. Root directories vary in size because of the different types and capacities of disks, but the root directory of a given disk is fixed. Using the `FORMAT` command creates a root directory that has a fixed length and can't be extended to hold more entries. The root directory entry limits are shown in Table 24.17. Subdirectories are stored as files in the data area of the disk and can grow in size dynamically; therefore, they have no fixed length limits.

Table 24.17 Root Directory Entry Limits

Drive Type	Maximum Root Directory Entries
Hard disk	512
1.44MB floppy disk	224
2.88MB floppy disk	448
Jaz and Zip	512
LS-120 and LS-240	512

Note

Two advantages of the FAT32 file system are that the root directory can be located anywhere on the disk and it can have an unlimited number of entries. FAT32 is discussed in more detail later in this chapter.

Every directory, whether it is the root directory or a subdirectory, is organized in the same way. Entries in the directory database store important information about individual files and how files are named on the disk. The directory information is linked to the FAT by the starting cluster entry. In fact, if no file on a disk were longer than one single cluster, the FAT would be unnecessary. The directory stores all the information needed by DOS to manage the file, with the exception of the list of clusters the file occupies other than the first one. The FAT stores the remaining information about the other clusters the file occupies.

To trace a file on a disk, use a disk editor, such as the Disk Edit program that comes with the Norton Utilities. Start by looking up the directory entry to get the information about the starting cluster of the file and its size. Then, using the appropriate editor commands, go to the FAT where you can follow the chain of clusters the file occupies until you reach the end of the file. By using the directory and FAT in this manner, you can visit all the clusters on the disk that are occupied by the file. This type of technique can be useful when these entries are corrupted and when you are trying to find missing parts of a file.

FAT directory entries are 32 bytes long and are in the format shown in Table 24.18, which shows the location (or offset) of each field within the entry (in both hexadecimal and decimal form) and the length of each field.

Table 24.18 FAT Directory Format

Offset (Hex)	Offset (Dec)	Field Length	Description
00h	0	8 bytes	Filename
08h	8	3 bytes	File extension
0Bh	11	1 byte	File attributes
0Ch	12	10 bytes	Reserved (00h)
16h	22	1 word	Time of creation
18h	24	1 word	Date of creation
1Ah	26	1 word	Starting cluster
1Ch	28	1 dword	Size in bytes

Filenames and extensions are left justified and padded with spaces, (which are represented as ASCII 32h bytes). In other words, if your filename is "AL", it is really stored as "AL------", where the hyphens are spaces. The first byte of the filename indicates the file status for that directory entry, shown in Table 24.19.

Table 24.19 Directory Entry Status Byte (First Byte)

Hex	File Status
00h	Entry never used; entries past this point not searched.
05h	Indicates that the first character of the filename is actually E5h.
E5h	σ (lowercase sigma). Indicates that the file has been erased.
2Eh	. (period). Indicates that this entry is a directory. If the second byte is also 2Eh, the cluster field contains the cluster number of the parent directory (0000h, if the parent is the root).

A word is 2 bytes read in reverse order, and a dword is two words read in reverse order.

Table 24.20 describes the FAT directory file attribute byte. Attributes are 1-bit flags that control specific properties of a file, such as whether it is hidden or designated as read-only. Each flag is individually activated (1) or deactivated (0) by changing the bit value. The combination of the eight bit values can be expressed as a single hexadecimal byte value; for example, 07h translates to 00000111, and the 1 bits in positions 3, 2, and 1 indicate the file is system, hidden, and read-only.

Table 24.20 FAT Directory File Attribute Byte

Bit Positions 7	6	5	4	3	2	1	0	Hex Value	Description
0	0	0	0	0	0	0	1	01h	Read-only file
0	0	0	0	0	0	1	0	02h	Hidden file
0	0	0	0	0	1	0	0	04h	System file
0	0	0	0	1	0	0	0	08h	Volume label
0	0	0	1	0	0	0	0	10h	Subdirectory
0	0	1	0	0	0	0	0	20h	Archive (updated since backup)

Table 24.20 Continued

Bit Positions 7 6 5 4 3 2 1 0	Hex Value	Description
0 1 0 0 0 0 0 0	40h	Reserved
1 0 0 0 0 0 0 0	80h	Reserved
Examples		
0 0 0 0 0 1 1 1	07h	System, hidden, read-only
0 0 1 0 0 0 0 1	21h	Read-only, archive
0 0 1 1 0 0 1 0	32h	Hidden, subdirectory, archive
0 0 1 0 0 1 1 1	27h	Read-only, hidden, system, archive

Note

The DOS/Windows command-line `ATTRIB` command can be used to change file attributes. In the Windows GUI, you can use the properties sheet for a file or folder to change the attributes. The archive bit changes automatically when a file is backed up or changed.

File Allocation Tables

The file allocation table is a list of numerical entries describing how each cluster in the partition is allocated. The data area of the partition has a single entry in the table for each of its clusters. Sectors in the non-data area of the partition are outside the range of the disk controlled by the FAT. Thus, the sectors that comprise the boot records, FATs, and root directory are outside the range of sectors controlled by the FAT.

▶▶ See "The Data Area," p. 1327.

The FAT does not manage every data sector specifically, but rather allocates space in groups of sectors called *clusters* or *allocation units*. A cluster is a unit of storage consisting of one or more sectors. The FDISK program determines the size of a cluster during the creation of the partition, based on the partition's size and the file system selected (FAT16 or FAT32). The smallest space a file can occupy in a partition is one cluster; all files use space in integer (whole) cluster units. If a file is one byte larger than one cluster, two whole clusters are used.

You can think of the FAT as a type of spreadsheet that tracks the allocation of the disk's clusters. Each cell in the spreadsheet corresponds to a single cluster on the disk. The number stored in that cell is a code indicating whether a file uses the cluster and, if so, where the next cluster of the file is located. Thus, to determine which clusters a particular file is using, you would start by looking at the first FAT reference in the file's directory entry. When you look up the referenced cluster in the FAT, the table contains a reference to the file's next cluster. Each FAT reference, therefore, points to the next cluster in what is called a FAT *chain* until you reach the cluster containing the end of the file. Numbers stored in the FAT are hexadecimal numbers that are either 12 or 16 bits long. The 16-bit FAT numbers are easy to follow in a disk sector editor because they take an even 2 bytes of space. The 12-bit numbers are 1 1/2 bytes long, which presents a problem because most disk sector editors show data in byte units. To edit a 12-bit FAT, you must do some hex/binary math to convert the displayed byte units to FAT numbers. Fortunately (unless you are using the DOS `DEBUG` program), most of the available tools and utility programs have a FAT-editing mode that automatically converts the numbers for you. Most of them also show the FAT numbers in decimal form, which most people find easier to handle. Table 24.21 shows the relationship between the directory and FAT for a given file (assuming the file is not fragmented).

Table 24.21 Directory and FAT Relationship (File Not Fragmented)

Directory

Name	Starting Cluster	Size
USCONST.TXT	1000	4

FAT16 File Allocation Table

FAT Cluster #	Value	Meaning
00002	0	First cluster available
…	…	…
00999	0	Cluster available
01000	1001	In use, points to next cluster
01001	1002	In use, points to next cluster
01002	1003	In use, points to next cluster
01003	FFFFh	End of file
01004	0	Cluster available
01005	0	Cluster available
…	…	…
65526	0	Last cluster available

In this example, the directory entry states that the file starts in cluster number 1000. In the FAT, that cluster has a nonzero value, which indicates it is in use; the specific value indicates where the file using it would continue. In this case, the entry for cluster 1000 is 1001, which means the file continues in cluster 1001. The entry in 1001 points to 1002, and from 1002 the entry points to 1003. The entry for 1003 is FFFFh, which is an indication that the cluster is in use but that the file ends here and uses no additional clusters.

The use of the FAT becomes clearer when you see that files can be fragmented. Let's say that before the `USCONST.TXT` file was written, another file already occupied clusters 1002 and 1003. If `USCONST.TXT` was written starting at 1000, it would not have been capable of being completely written before running into the other file. As such, the operating system would skip over the used clusters and continue the file in the next available cluster. The end result is shown in Table 24.22.

Table 24.22 Directory and FAT Relationship (Fragmented File)

Directory

Name	Starting Cluster	Size
PLEDGE.TXT	1002	2
USCONST.TXT	1000	4

FAT16 File Allocation Table

FAT Cluster #	Value	Meaning
00002	0	First cluster available
…	…	…
00999	0	Cluster available
01000	1001	In use, points to next cluster
01001	1004	In use, points to next cluster
01002	1003	In use, points to next cluster

Table 24.22 Continued

FAT16 File Allocation Table

FAT Cluster #	Value	Meaning
01003	FFFFh	End of file
01004	1005	In use, points to next cluster
01005	FFFFh	End of file
…	…	…
65526	0	Last cluster available

In this example, the `PLEDGE.TXT` file that was previously written interrupted `USCONST.TXT`, so those clusters are skipped over, and the pointers in the FAT reflect that. Note that the defrag programs included with DOS and Windows take an example like this and move the files so they are contiguous, one after the other, and update the FAT to indicate the change in cluster use.

The first two entries in the FAT are reserved and contain information about the table itself. All remaining entries correspond to specific clusters on the disk. Most FAT entries consist of a reference to another cluster containing the next part of a particular file. However, some FAT entries contain hexadecimal values with special meanings, as follows:

- *0000h.* Indicates that the cluster is not in use by a file
- *FFF7h.* Indicates that at least one sector in the cluster is damaged and that it should not be used to store data
- *FFF8h–FFFFh.* Indicates that the cluster contains the end of a file and that no reference to another cluster is necessary

The FDISK program generally determines whether a 12-bit, 16-bit, or 32-bit FAT is placed on a disk, even though the FAT isn't written until you perform a high-level format (using the `FORMAT` utility). On today's systems, usually only floppy disks use 12-bit FATs, but FDISK also creates a 12-bit FAT if you create a hard disk volume that is smaller than 16MiB. On hard disk volumes of more than 16MiB, FDISK creates a 16-bit FAT. On drives larger than 512MiB, the FDISK program included in Windows 95 OSR2, and Windows 98/Me enables you to create 32-bit FATs when you answer Yes to the question `Enable Large Disk Support?` when you start FDISK. You can also select FAT32 when you prepare a drive with the Windows 2000 or Windows XP Disk Management program (Windows NT doesn't support FAT32).

FAT volumes normally have two copies of the FAT. Each one occupies contiguous sectors on the disk, and the second FAT copy immediately follows the first. Unfortunately, the operating system uses the second FAT copy only if sectors in the first FAT copy become unreadable. If the first FAT copy is corrupted, which is a much more common problem, the operating system does not use the second FAT copy. Even the `CHKDSK` command does not check or verify the second FAT copy. Moreover, whenever the OS updates the first FAT, it automatically copies large portions of the first FAT to the second FAT. If the first copy was corrupted and subsequently updated by the OS, a large portion of the first FAT is copied over to the second FAT copy, damaging it in the process. After the update, the second copy is usually a mirror image of the first one, complete with any corruption. Two FATs rarely stay out of sync for very long. When they are out of sync and the OS writes to the disk, it updates the first FAT and overwrites the second FAT with the first. This is why disk repair and recovery utilities warn you to stop working as soon as you detect a FAT problem. Programs such as Norton Disk Doctor (included with the Norton Utilities which are a part of Norton SystemWorks) use the second copy of the FAT as a reference to repair the first one, but if the OS has already updated the second FAT, repair might be impossible.

Clusters (Allocation Units)

A *cluster* is often called an *allocation unit*. *Allocation unit* is appropriate because a single cluster is the smallest unit of the disk that the operating system can handle when it writes or reads a file. A cluster is equal to one or more 512-byte sectors, in a power of 2. Although a cluster can be a single disk sector, it is usually more than one. Having more than one sector per cluster reduces the size and processing overhead of the FAT and enables the operating system to run faster because it has fewer individual units to manage. The trade-off is in wasted disk space. Because operating systems manage space only in full-cluster units, every file consumes space on the disk in increments of one cluster.

Table 24.23 shows the default cluster (or allocation unit) sizes for the various floppy disk formats used over the years.

Table 24.23 Default Floppy Disk Cluster (Allocation Unit) Sizes

Drive Type	Cluster (Allocation Unit) Size	Density
5 1/4" 360KB	2 sectors (1,024 bytes)	Low
5 1/4" 1.2MB	1 sector (512 bytes)	High
3 1/2" 720KB	2 sectors (1,024 bytes)	Low
3 1/2" 1.44MB	1 sector (512 bytes)	High
3 1/2" 2.88MB	2 sectors (1,024 bytes)	Extra

It seems strange that the high-density disks, which have many more individual sectors than low-density disks, sometimes have smaller cluster sizes. The larger the FAT, the more entries the operating system must manage and the slower it seems to function. This sluggishness is due to the excessive overhead required to manage all the individual clusters; the more clusters to be managed, the slower things become. The trade-off is in the minimum cluster size. High-density floppy disk drives are faster than their low-density counterparts, so perhaps IBM and Microsoft determined that the decrease in cluster size balances the drive's faster operation and offsets the use of a larger FAT.

Because the operating system can allocate only whole clusters, inevitably a certain amount of wasted storage space results. File sizes rarely fall on cluster boundaries, so the last cluster allocated to a particular file is rarely filled completely. The extra space left over between the actual end of the file and the end of the cluster is called *slack*. A partition with large clusters has more slack space, whereas smaller clusters generate less slack.

For hard disks, the cluster size varies greatly among partition sizes and file systems. Table 24.24 shows the cluster sizes FDISK selects for a particular volume size when FAT16 is the file system used.

Table 24.24 Default Hard Disk Cluster (Allocation Unit) Sizes for FAT16

Volume Size	Sectors per Cluster	Cluster Size
4.1MiB–15.96MiB[1]	2	1KiB
>15.96MiB–128MiB	4	2KiB
>128MiB–256MiB	8	4KiB
>256MiB–512MiB	16	8KiB
>512MiB–1GiB	32	16KiB
>1GiB–2GiB	64	32KiB
>2GiB–4GiB[2]	128	64KiB

1. Volumes smaller than 16MiB default to FAT12; however, FAT32 can be forced in some cases by altering the format parameters.

2. Volumes larger than 2GiB are supported only by Windows NT/2000/XP and are not recommended.

MB = Megabyte = 1,000,000 bytes
GB = Gigabyte = 1,000MB = 1,000,000,000 bytes
TB = Terabyte = 1,000GB = 1,000,000,000,000 bytes
KiB = Kibibytes = 1,024 bytes
MiB = Mebibyte = 1,024KiB = 1,048,576 bytes
GiB = Gibibyte = 1,024MiB = 1,073,741,824 bytes

The effect of larger cluster sizes on disk utilization can be substantial. A 2GiB partition containing about 5,000 files, with average slack of one-half of the last 32KiB cluster used for each file, wastes more than 78MB (5000×(.5×32)KiB) of file space. When files under 32KiB in size are stored on a drive with a 32KiB allocation unit, waste (slack) factors can approach 40% of the drive's capacity.

16-bit cluster numbers range from 0000h to FFFFh, which is 0–65,535 in decimal. This theoretically allows 65,536 total clusters, but 11 of the cluster numbers are reserved and cannot be assigned to actual clusters on a disk. Cluster numbers start at 2 (0 and 1 are reserved), number FFF7h is reserved to indicate a bad cluster, and numbers FFF8h–FFFFh indicate an end-of-chain in the FAT, leaving 65,525 clusters (65,536 – 11). Microsoft subtracts 1 from this to eliminate boundary problems, allowing up to 65,524 clusters maximum in a FAT16 volume.

FAT16 volumes include 1 sector for the boot record and BPB, two copies of the FAT (default and backup) up to 256 sectors long each, 32 sectors for the root directory, and a data area with 4,085–65,524 clusters. Because each FAT16 cluster can be up to 64 sectors (32KiB) in size, FAT16 volumes are limited to a maximum size of 4,194,081 sectors (1 sector for the boot record + 256 sectors per FAT × 2 FATs + 32 sectors for the Root Directory + 65,524 clusters × 64 sectors per cluster), which equals a maximum capacity of 2.15GB or 2GiB.

Note

Windows NT/2000/XP support the use of 64KiB clusters on FAT16 partitions, enabling you to create partitions up to 4GiB in size. However, the huge amount of slack that generally results from clusters this large makes this practice undesirable. Also note that no other operating systems with FAT16 support are capable of accessing a FAT16 partition larger than 2GiB.

The Data Area

The data area of a partition is the place where the actual files are stored. It is located following the boot record, file allocation tables, and root directory. This is the area of the disk that is divided into clusters and managed by the FAT and root directory.

Diagnostic Read-and-Write Cylinder

The FDISK partitioning program always reserves the last cylinder of a hard disk for use as a special diagnostic read-and-write test cylinder. Because this cylinder is reserved, FDISK always reports fewer total cylinders than the drive manufacturer states are available. Operating systems do not use this cylinder for any normal purpose because it lies outside the partitioned area of the disk.

◀◀ See "Disk Formatting," p. 609.

On systems with IDE or SCSI disk interfaces, the drive and controller also might allocate an additional area past the logical end of the drive for a bad-track table and spare sectors. This situation can account for additional discrepancies between the FDISK and the drive manufacturer's reported sizes.

The diagnostics area enables software such as a manufacturer-supplied diagnostics disk to perform read-and-write tests on a hard disk without corrupting any user data. Many of these programs also swap spare cylinders for damaged cylinders if damaged cylinders are detected during testing.

VFAT and Long Filenames

The original Windows 95 release uses what is essentially the same FAT file system as DOS, except for a few important enhancements. Like much of the rest of Windows 95, the operating system support for the FAT file system was rewritten using 32-bit code and called *VFAT (virtual file allocation table)*. VFAT works in combination with the 32-bit protected mode VCACHE (which replaces the 16-bit real mode SMARTDrive cache used in DOS and Windows 3.1) to provide better file system performance. However, the most obvious improvement in VFAT is its support for long filenames. DOS and Windows 3.1 had been encumbered by the standard 8.3 filenaming convention for many years, and

adding long filename support was a high priority in Windows 95—particularly in light of the fact that Macintosh and OS/2 users had long enjoyed this capability.

The problem for the Windows 95 designers, as is often the case in the PC industry, was backward compatibility. It is no great feat to make long filenames possible when you are designing a new file system from scratch, as Microsoft did years before with Windows NT's NTFS. However, the Windows 95 developers wanted to add long filenames to the existing FAT file system and still make it possible to store those names on existing DOS volumes and for previous versions of DOS and Windows to access the files.

VFAT provides the capability to assign file and directory names that are up to 255 characters in length (including the length of the path). The three-character extension is maintained because, like previous Windows versions, Windows 9x relies on the extensions to associate file types with specific applications. VFAT's long filenames can also include spaces, as well as the following characters, which standard DOS 8.3 names can't: +,;=[ ].

The first problem when implementing the long filenames was how to make them usable to previous versions of DOS and 16-bit Windows applications that support only 8.3 names. The resolution to this problem was to give each file two names: a long filename and an alias that uses the traditional 8.3 naming convention. When you create a file with a long filename in Windows 9x/Me, VFAT uses the following process to create an equivalent 8.3 alias name:

1. The first three characters after the last dot in the long filename become the extension of the alias.
2. The first six characters of the long filename (excluding spaces, which are ignored) are converted into uppercase and become the first six characters of the alias filename. If any of these six characters are illegal under the standard 8.3 naming rules (that is, +,;=[ ]), VFAT converts those characters into underscores.
3. VFAT adds the two characters ~1 as the seventh and eighth characters of the alias filename, unless this will result in a name conflict, in which case it uses ~2, ~3, and so on, as necessary.

Aliasing in Windows NT/2000/XP

Note that Windows NT/2000/XP creates aliases differently than Windows 9x/Me (as shown later).

NT/2000/XP begins by taking the first six legal characters in the LFN and following them with a tilde and number. If the first six characters are unique, a number 1 follows the tilde.

If the first six characters aren't unique, a number 2 is added. NT/2000/XP uses the first three legal characters following the last period in the LFN for a file extension.

At the fifth iteration of this process, NT/2000/XP takes only the first two legal characters, performs a hash on the filename to produce four hexadecimal characters, places the four hex characters after the first two legal characters, and appends a ~5. The ~5 remains for all subsequent aliases; only the hex numbers change.

Tip

You can modify the behavior of the VFAT filename truncation mechanism to make it use the first eight characters of the long filename instead of the first six characters plus ~1. To do this, you must add a new binary value to the `HKEY_LOCAL_MACHINE\System\CurrentControlSet\control\FileSystem` Registry key called `NameNumericTail`, with a value of `0`. Changing the value to `1` returns the truncation process to its original state.

Although this Registry change creates "friendly" alias names, it causes many programs working with alias names to fail and is not recommended.

VFAT stores this alias filename in the standard name field of the file's directory entry. Any version of DOS or 16-bit Windows can therefore access the file using the alias name. The big problem that still remains, however, is where to store the long filenames. Clearly, storing a 255-character filename in a 32-byte directory entry is impossible (because each character requires 1 byte). However, modifying the structure of the directory entry would make the files unuseable by previous DOS versions.

The developers of VFAT resolved this problem by using additional directory entries to store the long filenames. Each of the directory entries is still 32 bytes long, so up to 8 might be required for each long name, depending on its length. To ensure that these additional directory entries are not misinterpreted by earlier DOS versions, VFAT flags them with a combination of attributes that is not possible for a normal file: read-only, hidden, system, and volume label. These attributes cause DOS to ignore the long filename entries, while preventing them from being mistakenly overwritten.

Caution

When using long filenames on a standard FAT12 or FAT16 partition, you should avoid storing them in the root directory. Files with long names that take up multiple directory entries can more easily use up the limited number of entries allotted to the root directory than files with 8.3 names. On FAT32 drives, this is not a problem because the root directory has an unlimited number of entries.

In an experiment, I created a small (1KiB) text file on a floppy disk and gave it a 135-character long filename using Windows 98. I copied the file and pasted it repeatedly into the root directory of a floppy disk using Windows Explorer. Before I could make 20 copies of the file, the system displayed a `File copying` error. The disk could not accept any more files because the extremely long filename had used up all the root directory entries.

This solution for implementing backward-compatible long filenames in Windows 9x is ingenious, but it is not without its problems. Most of these problems stem from the use of applications that can access only the 8.3 alias names assigned to files. In some cases, if you open a file with a long name using one of these programs and save it again, the connection to the additional directory entries containing the long name is severed and the long name is lost.

This is especially true for older versions of disk utilities, such as Norton Disk Doctor for MS-DOS, that are not designed to support VFAT. Most older applications ignore the additional directory entries because of the combination of attributes assigned to them, but disk repair utilities usually are designed to detect and "correct" discrepancies of this type. The result is that running an old version of Norton Disk Doctor on a partition with long filenames results in the loss of all the long names. In the same way, backup utilities not designed for use with VFAT can strip off the long filenames from a partition.

Note

When using VFAT's long filename capabilities, you definitely should use disk and backup utilities that are intended to support VFAT. Windows 9x includes VFAT-compatible disk repair, defragmentation, and backup programs. If, however, you are for some reason inclined to use an older program that does not support VFAT, Windows 9x includes a clumsy, but effective, solution.

A program is included on the Windows 9x CD-ROM called LFNBK.EXE. It doesn't install with the operating system, but you can use it to strip the long filenames from a VFAT volume and store them in a text file called `LFNBK.DAT`. You can then work with the files on the volume as though they were standard 8.3 FAT files. Afterward, you can use LFNBK.EXE to restore the long filenames to their original places (assuming the file and directory structure has not changed). This is not a convenient solution, nor is it recommended for use in anything but extraordinary circumstances, but the capability is there if needed. Some backup programs designed for disaster recovery (which enable you to reconstruct the contents of the hard drive without reloading Windows first) have used this feature to enable restoration of a Windows drive with long filenames from a DOS prompt (where only 8.3 alias names usually are supported).

Another problem with VFAT's long filenames involves the process by which the file system creates the 8.3 alias names. VFAT creates a new alias every time you create or copy a file into a new directory; therefore, the alias can change. For example, you might have a file called `Expenses-January98.doc` stored in a directory with the alias `EXPENS~1.DOC`. If you use Windows 9x Explorer to copy this file to a directory that already contains a file called `Expenses-December97.doc`, you are likely to find that this existing file is already using the alias `EXPENS~1.DOC`. In this case, VFAT assigns `EXPENS~2.DOC` as the alias of the newly copied file, with no warning to the user. This is not a problem for applications that support VFAT because the long filenames are unchanged, but a user running an older application might open the `EXPENS~1.DOC` file expecting to find the list of January 1998 expenses and see the December 1997 expenses list instead.

FAT32

When the FAT file system was being developed, 2GB hard disk drives were a fantasy and no one expected the partition size limitation imposed by the combination of 16-bit FAT entries and 32KiB cluster sizes to be a problem. Today, however, even low-end PCs come equipped with hard drives holding at least 20GB, and 80GB and larger drives are common. When you use the standard FAT16 file system with these larger drives, you must create several partitions, each no larger than approximately 2GiB. To many users, having multiple drive letters representing a single disk is confusing, making organizing and locating files difficult.

To address this problem, Microsoft released an enhanced version of the FAT file system, called FAT32. FAT32 works just like the standard FAT; the only difference is that it uses numbers with more digits, so it can manage more clusters on a disk. Unlike VFAT, which is a Windows 9x innovation that uses existing file system structures, FAT32 is an enhancement of the FAT file system itself. In other words, whereas VFAT is implemented as part of the Windows 9x Virtual Machine Manager (Vmm.vxd), FAT32 is implemented by the FDISK program before the Windows GUI is even loaded. FAT32 was first included in the Windows 95 OEM Service Release 2 (OSR2, also known as Windows 95B) and is also part of Windows 98/Me, Windows 2000, and Windows XP operating systems.

Note

Because the FDISK utility can implement the FAT32 file system when you partition the drive, you can't use FAT32 on a floppy disk or any removable cartridge that does not use a partition table. However, any medium you can partition with FDISK can use FAT32.

The most obvious improvement in FAT32 is that by using 32-bit values for FAT entries instead of 16-bit ones, the maximum number of clusters allowed in a single partition jumps from 65,536 (2^{16}) to 268,435,456. This value is equivalent to 2^{28}, not 2^{32}, because 4 bits out of the 32 are reserved for other uses. FAT32 also uses 32-bit values for the low-level operating system calls used to retrieve a specific disk sector.

These expanded values blow the top off the 2GiB limit on partition sizes. Because each FAT32 cluster is up to 64 sectors (32KiB) in size, FAT32 disks or partitions can theoretically be up to 17,184,062,816 sectors (32 sectors for the boot record + 2,097,152 sectors per FAT × 2 FATs + 268,435,445 clusters × 64 sectors per cluster), which equals a capacity of 8.8TB or 8TiB. Note that this capacity is theoretical because the 32-bit sector numbering scheme used in the partition tables located in the MBR limits a disk to no more than 4,294,967,295 (2^{32}–1) sectors, which is 2.2TB or 2TiB. Therefore, even though FAT32 can in theory handle a volume of up to 8.8TB, the reality is that it's currently limited by the partition table format of the MBR to only 2.2TB. At the current rate of hard disk capacity growth, single drives of that size will come into existence between the years 2009 and 2011. Of course, by that time the MBR limitation should be addressed.

Individual files can be up to 1 byte less than 4GiB in size and are limited by the size field in the directory entry, which is 4 bytes long. Because the clusters are numbered using 32-bit values instead of 16-bit ones, the format of the directory entries on a FAT32 partition must be changed slightly. The 2-byte Link to Start Cluster field is increased to 4 bytes, using 2 of the 10 bytes (bytes 12–21) in the directory entry that were reserved for future use.

Windows 2000 and Windows XP are intentionally limited to format FAT32 volumes of up to only 32GiB in size but can read larger FAT32 partitions. The limit is there not for any technical reason, but because Microsoft wants to encourage people to use NTFS instead of FAT32.

Another important difference in FAT32 partitions is the nature of the root directory. In a FAT32 partition, the root directory does not occupy a fixed position on the disk as in a FAT16 partition. Instead, it can be located anywhere in the partition and expand to any size. This eliminates the preset limit on root directory entries and provides the infrastructure necessary to make FAT32 partitions dynamically resizable. Unfortunately, Microsoft never implemented that feature in Windows, but third-party products such as PowerQuest's PartitionMagic can take advantage of this capability.

The main drawback of FAT32 is that it is not compatible with previous versions of DOS and Windows 95. You can't boot to a previous version of DOS or (pre-OSR2) Windows 95 from a FAT32 drive, nor can a system started with an old DOS or Windows 95 boot disk see FAT32 partitions. Most recent distributions of Linux now support FAT32, and even the Mac OS 8.1 and later can read and write FAT32 volumes. For all but the oldest of equipment, FAT32 is the most universally understandable format that supports larger volumes.

FAT32 Cluster Sizes

Because FAT32 partitions can have so many more clusters than FAT16 partitions, the clusters themselves can be smaller. Using smaller clusters reduces the wasted disk space caused by slack. For example, the same 2GiB partition with 5,000 files on it mentioned earlier would use 4KiB clusters with FAT32 instead of 32KiB clusters with FAT16. Assuming the same amount of slack for each file, the smaller cluster size reduces the average amount of wasted space on that partition from more than 78MB to less than 10MB.

To compare FAT16 and FAT32, you can look at how a file would be stored on each. For FAT16, I'll use the same example as before. With FAT16 the cluster numbers are stored as 16-bit entries, from 0000h to FFFFh. The largest value possible is FFFFh, which corresponds to 65,535 in decimal, but several numbers at the beginning and end are reserved for special use. The actual cluster numbers allowed in a FAT16 system range from 0002h to FFF6h, which is 2–65,526 in decimal. All files must be stored in cluster numbers within that range. That leaves only 65,524 valid clusters to use for storing files (cluster numbers below 2 and above 65,526 are reserved), meaning a partition must be broken up into that many clusters or less. A typical file entry under FAT16 might look like Table 24.25.

Table 24.25 FAT16 File Entries

Directory		
Name	**Starting Cluster**	**Size**
`USCONST.TXT`	1000	4
FAT16 File Allocation Table		
FAT Cluster #	**Value**	**Meaning**
00002	0	First cluster available
…	…	…
00999	0	Cluster available
01000	1001	In use, points to next cluster
01001	1002	In use, points to next cluster
01002	1003	In use, points to next cluster
01003	FFFFh	End of file
01004	0	Cluster available
…	…	…
65526	0	Last cluster available

With FAT32, the cluster numbers range from 00000000h to FFFFFFFFh, which is 0–4,294,967,295 in decimal. Again, some values at the low and high ends are reserved, and only values between 00000002h and FFFFFFF6h are valid, which means values 2–4,294,967,286 are valid. This leaves 4,294,967,284 valid entries, so the drive must be split into that many clusters or less. Because a drive can be split into so many more clusters, the clusters can be smaller, which conserves disk space. The same file as shown earlier could be stored on a FAT32 system as illustrated in Table 24.26.

Table 24.26 FAT32 File Entries

Directory		
Name	**Starting Cluster**	**Size**
USCONST.TXT	1000	8
FAT32 File Allocation Table		
FAT Cluster #	**Value**	**Meaning**
0000000002	0	First cluster available
…	…	…
0000000999	0	Cluster available
0000001000	1001	In use, points to next cluster
0000001001	1002	In use, points to next cluster
0000001002	1003	In use, points to next cluster
0000001003	1004	In use, points to next cluster
0000001004	1005	In use, points to next cluster
0000001005	1006	In use, points to next cluster
0000001006	1007	In use, points to next cluster
0000001007	FFFFFFFFh	End of file
0000001008	0	Cluster available
…	…	…
4,294,967,286	0	Last cluster available

Because the FAT32 system enables many more clusters to be allocated, the cluster size is usually smaller. So, although files overall use more individual clusters, less wasted space results because the last cluster is, on average, only half filled.

Note

How much space does your current cluster size waste? To find out, you can download a free Windows utility called Karen's Disk Slack Checker from former *Windows Magazine* editor Karen Kenworthy's Web site. Go to `http://www.karenware.com/powertools/ptslack.asp`.

Table 24.27 lists the cluster sizes for various FAT32 partition sizes.

Table 24.27 FAT32 Cluster Sizes

Volume Size	Maximum Volume Sectors	Sectors per Cluster	Cluster Size
0–32.52MiB[1]	66,600	FAT12/16	FAT12/16
>32.52MiB–260MiB[2]	532,480	1	0.5KiB
>260MiB–8GiB	16,777,216	8	4KiB

Table 24.27 Continued

Volume Size	Maximum Volume Sectors	Sectors per Cluster	Cluster Size
>8GiB–16GiB	33,554,432	16	8KiB
>16GiB–32GiB	67,108,864	32	16KiB
>32GiB–2TiB[3]	4,294,967,295	64	32KiB

1. FAT32 cannot be used on volumes 66,600 sectors (32.52MiB) or less.

2. Volumes smaller than 512MiB will default to FAT16, although FAT32 can be forced by altering the format parameters.

3. Windows 2000 and XP format FAT32 volumes only up to 32GiB; however, they do support existing FAT32 volumes up to 2TiB.

MB = Megabyte = 1,000,000 bytes
GB = Gigabyte = 1,000MB = 1,000,000,000 bytes
TB = Terabyte = 1,000GB = 1,000,000,000,000 bytes
KiB = Kibibytes = 1,024 bytes
MiB = Mebibyte = 1,024KiB = 1,048,576 bytes
GiB = Gibibyte = 1,024MiB = 1,073,741,824 bytes
TiB = Tebibytes = 1,024GiB = 1,099,511,627,776 bytes

Some additional limitations exist on FAT32 volumes. Volumes less than 512MiB default to FAT16, but FAT32 partitions as small as 32.52MiB can be forced using the proper utilities or commands. Anything smaller than that, however, must be either FAT16 or FAT12.

Windows 2000/XP or later will not format a volume larger than 32GiB using FAT32. If you attempt to format a FAT32 partition larger than 32GiB under Windows 2000/XP or later, the format operation will fail near the end of the process and you might receive the following error message: `Logical Disk Manager: Volume size too big`. If you want to use a larger FAT32 partition under Windows 2000/XP or later, you can format the partition using Windows 9x/Me because Windows 2000/XP or later will mount and support larger FAT32 partitions without any problems. The only limitation with FAT32 partitions under Windows 2000/XP or later is in the formatting process. Finally, remember that DOS 6.22 and earlier, Windows 95a, and Windows NT (through 4.0) do not recognize FAT32 partitions and are incapable of booting from a FAT32 volume.

Using smaller clusters results in many more clusters and more entries in the FAT. A 2GiB partition using FAT32 requires up to 524,288 FAT entries, whereas the same drive needs only 65,536 entries using FAT16. Thus, the size of one copy of the FAT16 table is 128KiB (65,536 entries × 16 bits = 1,048,576 bits/8 = 131,072 bytes/1,024 = 128KiB), whereas the FAT32 table is 2MiB in size.

The size of the FAT has a definite impact on the performance of the file system. Windows 9x/Me uses VCACHE to keep the FAT in memory at all times to improve file system performance. The use of 4KiB clusters for drives up to 8GiB in size is, therefore, a reasonable compromise for the average PC's memory capacity. If the file system were to use clusters equal to one disk sector (1 sector = 512KiB) in an attempt to minimize slack as much as possible, the FAT table for a 2GiB drive would contain 4,194,304 entries and be 16MB in size.

This would monopolize a substantial portion of the memory in the average system, probably resulting in noticeably degraded performance. Although at first it might seem as though even a 2MB FAT is quite large when compared to 128KiB, keep in mind that hard disk drives are a great deal faster now than they were when the FAT file system was originally designed. In practice, FAT32 typically results in a minor (less than 5%) improvement in file system performance. However, systems that perform a great many sequential disk writes might see an equally minor degradation in performance.

FAT Mirroring

FAT32 also takes greater advantage of the two copies of the FAT stored on a disk partition. On a FAT16 partition, the first copy of the FAT is always the primary copy and replicates its data to the secondary FAT, sometimes corrupting it in the process. On a FAT32 partition, when the system detects a problem with the primary copy of the FAT, it can switch to the other copy, which then becomes the primary.

The system can also disable the FAT mirroring process to prevent the viable FAT from being corrupted by the other copy. This provides a greater degree of fault tolerance to FAT32 partitions, often enabling you to repair a damaged FAT without an immediate system interruption or a loss of table data.

Creating FAT32 Partitions

Despite the substantial changes FAT32 provides, the new file system does have a large effect on the procedures you use to create and manage partitions. You create new FAT32 partitions in Windows 9x/Me using the FDISK utility from the command prompt, just as you would create FAT16 partitions. When you launch FDISK, the program examines your hard disk drives and, if they have a capacity greater than 512MiB, presents the following message:

```
Your computer has a disk larger than 512MB. This version of Windows
includes improved support for large disks, resulting in more efficient
use of disk space on large drives, and allowing disks over 2GB to be
formatted as a single drive.

IMPORTANT: If you enable large disk support and create any new drives on this
disk, you will not be able to access the new drive(s) using other operating
systems, including some versions of Windows 95 and Windows NT, as well as
earlier versions of Windows and MS-DOS. In addition, disk utilities that
were not designed explicitly for the FAT32 file system will not be able
to work with this disk. If you need to access this disk with other operating
systems or older disk utilities, do not enable large drive support.

Do you wish to enable large disk support (Y/N).....? [N]
```

If you answer Yes to this question, any partitions you create that are larger than 512MiB will be FAT32 partitions. If you want to create partitions larger than 2GiB, you must use FAT32. Otherwise, you can choose which file system you prefer. All the screens following this one are the same as those in previous versions of FDISK.

Normally, the FDISK utility determines the cluster size used when you format the partition, based on the partition's size and the file system used. However, you can override FDISK using an undocumented switch for the `FORMAT` utility. If you use the command

```
FORMAT /Z:n
```

where *n* multiplied by 512 equals the desired cluster size in bytes, you can create a partition that uses cluster sizes that are larger or smaller than the defaults for the file system.

Caution

The `/Z` switch does not override the 65,534-cluster limit on FAT16 partitions, so it is recommended that you use it only with FAT32. In addition, you should not use this switch on a production system without extensive testing first. Modifying the cluster size can increase or decrease the amount of slack on the partition, but it also can have a pronounced effect on the performance of the drive. Some disk utilities might not work with nonstandard cluster sizes.

Converting FAT16 to FAT32

If you want to convert an existing FAT16 partition to FAT32, Windows 98 and Me later include a FAT32 Conversion Wizard that enables you to migrate existing partitions in place.

The wizard gathers the information needed to perform the conversion, informs you of the consequences of implementing FAT32, and attempts to prevent data loss and other problems. After you have selected the drive you want to convert, the wizard performs a scan for applications (such as disk utilities) that might not function properly on the converted partition. The wizard gives you the opportunity to remove these and warns you to back up the data on the partition before proceeding

with the conversion. Even if you don't use the Microsoft Backup utility the wizard offers, backing up your data is a strongly recommended precaution.

Because the conversion must deal with the existing partition data in addition to creating new volume boot record information, FATs, and clusters, the process can take far longer than partitioning and formatting an empty drive. Depending on the amount of data involved and the new cluster size, the conversion can take several hours to complete.

After you convert a FAT16 partition to FAT32, you can't convert it back with Windows tools, except by destroying the partition and using FDISK to create a new one. Aftermarket partitioning utilities are available that can convert FAT32 back to FAT16 if you want. You should take precautions before beginning the conversion process, such as connecting the system to a UPS. A power failure during the conversion could result in a loss of data.

Third-Party Partitioning Utilities

Windows includes only basic tools for creating FAT32 partitions, but programs such as Partition Commander from VCOM (`www.v-com.com`) and PartitionMagic from PowerQuest (`www.powerquest.com`) provide many other partition manipulation features. These programs can easily convert partitions back and forth between FAT16, FAT32, NTFS and other file systems, as well as resize, move, and copy partitions without destroying the data they contain. They also allow changes in cluster sizes beyond what the standard Windows tools create.

FAT File System Errors

File system errors can, of course, occur because of hardware problems, but you are more likely to see them result from software crashes and improper system handling. Turning off a system without shutting down Windows properly, for example, can result in errors that cause clusters to be incorrectly listed as in use when they are not. Some of the most common file system errors that occur on FAT partitions are described in the following sections.

Lost Clusters

Probably the most common file system error, *lost* clusters are clusters the FAT designates as being in use when they actually are not. Most often caused by an interruption of a file system process due to an application crash or a system shutdown, for example, the FAT entry of a lost cluster might contain a reference to a subsequent cluster. However, the FAT chain stemming from the directory entry has been broken somewhere along the line.

Lost clusters appear in the file structure as shown in Table 24.28.

Table 24.28 Lost Clusters in a File Structure

Directory		
Name	**Starting Cluster**	**Size**
(no entry)	0	0
FAT16 File Allocation Table		
FAT Cluster #	**Value**	**Meaning**
00002	0	First cluster available
…	…	…
00999	0	Cluster available

Table 24.28 Continued

FAT16 File Allocation Table		
FAT Cluster #	**Value**	**Meaning**
01000	1001	In use, points to next cluster
01001	1002	In use, points to next cluster
01002	1003	In use, points to next cluster
01003	FFFFh	End of file
01004	0	Cluster available
…	…	…
65526	0	Last cluster available

The operating system sees a valid chain of clusters in the FAT but no corresponding directory entry to back it up. Programs that are terminated before they can close their open files typically cause this. The operating system usually modifies the FAT chain as the file is written, and the final step when closing is to create a matching directory entry. If the system is interrupted before the file is closed (such as by shutting down the system improperly), lost clusters are the result. Disk repair programs check for lost clusters by tracing the FAT chain for each file and subdirectory in the partition and building a facsimile of the FAT in memory. After compiling a list of all the FAT entries that indicate properly allocated clusters, the program compares this facsimile with the actual FAT. Any entries denoting allocated clusters in the real FAT that do not appear in the facsimile are lost clusters because they are not part of a valid FAT chain.

The utility typically gives you the opportunity to save the data in the lost clusters as a file before it changes the FAT entries to show them as unallocated clusters. If your system crashed or lost power while you were working with a word processor data file, for example, you might be able to retrieve text from the lost clusters in this way. When left unrepaired, lost clusters are unavailable for use by the system, reducing the storage capacity of your drive.

The typical choices you have for correcting lost clusters are to assign them a made-up name or zero out the FAT entries. If you assign them a name, you can at least look at the entries as a valid file and then delete the file if you find it useless. The CHKDSK and SCANDISK programs are designed to fix lost clusters by assigning them names starting with `FILE0001.CHK`. If more than one lost chain exists, sequential numbers following the first one are used. The lost clusters shown earlier could be corrected by CHKDSK or SCANDISK as shown in Table 24.29.

Table 24.29 Finding Lost Clusters

Directory		
Name	**Starting Cluster**	**Size**
`FILE0001.CHK`	1000	4
FAT16 File Allocation Table		
FAT Cluster #	**Value**	**Meaning**
00002	0	First cluster available
…	…	…
00999	0	Cluster available
01000	1001	In use, points to next cluster
01001	1002	In use, points to next cluster
01002	1003	In use, points to next cluster

Table 24.29 Continued

FAT16 File Allocation Table		
FAT Cluster #	**Value**	**Meaning**
01003	FFFFh	End of file
01004	0	Cluster available
…	…	…
65526	0	Last cluster available

As you can see, a new entry was created to match the FAT entries. The name is made up because there is no way for the repair utility to know what the original name of the file might have been.

Cross-Linked Files

Cross-linked files occur when two directory entries improperly reference the same cluster in their Link to Start Cluster fields. The result is that each file uses the same FAT chain. Because the clusters can store data from only one file, working with one of the two files can inadvertently overwrite the other file's data.

Cross-linked files would appear in the file structure as shown in Table 24.30.

Table 24.30 Cross-Linked Files

Directory		
Name	**Starting Cluster**	**Size**
USCONST.TXT	1000	4
PLEDGE.TXT	1002	2
FAT16 File Allocation Table		
FAT Cluster #	**Value**	**Meaning**
00002	0	First cluster available
…	…	…
00999	0	Cluster available
01000	1001	In use, points to next cluster
01001	1002	In use, points to next cluster
01002	1003	In use, points to next cluster
01003	FFFFh	End of file
01004	0	Cluster available
…	…	…
65526	0	Last cluster available

In this case, two files claim ownership of clusters 1002 and 1003, so these files are said to be cross-linked on 1002. When a situation such as this arises, one of the files typically is valid and the other is corrupt, being that only one actual given set of data can occupy a given cluster. The normal repair is to copy both files involved to new names, which duplicates their data separately in another area of the disk, and then delete *all* the cross-linked files. Deleting them all is important because by deleting only one of them, the FAT chain is zeroed, which further damages the other entries. Then, you can examine the files you copied to determine which one is good and which is corrupt.

Detecting cross-linked files is a relatively easy task for a disk repair utility because it must examine only the partition's directory entries and not the file clusters themselves. However, by the time the utility detects the error, the data from one of the two files is probably already lost—although you might be able to recover parts of it from lost clusters.

Invalid Files or Directories

Sometimes the information in a directory entry for a file or subdirectory can be corrupted to the point at which the entry is not just erroneous (as in cross-linked files) but invalid. The entry might have a cluster or date reference that is invalid, or it might violate the rules for the entry format in some other way. In most cases, disk repair software can correct these problems, permitting access to the file.

FAT Errors

As discussed earlier, accessing its duplicate copy can sometimes repair a corrupted FAT. Disk repair utilities typically rely on this technique to restore a damaged FAT to its original state, as long as the mirroring process has not corrupted the copy. FAT32 tables are more likely to be repairable because their more advanced mirroring capabilities make the copy less likely to be corrupted.

An example of a damaged FAT might appear to the operating system as shown in Table 24.31.

Table 24.31 Damaged FAT

Directory		
Name	**Starting Cluster**	**Size**
USCONST.TXT	1000	4
FAT16 File Allocation Table		
FAT Cluster #	**Value**	**Meaning**
00002	0	First cluster available
…	…	…
00999	0	Cluster available
01000	1001	In use, points to next cluster
01001	0	Cluster available
01002	1003	In use, points to next cluster
01003	FFFFh	End of file
01004	0	Cluster available
…	…	…
65526	0	Last cluster available

This single error would cause multiple problems to appear. The file USCONST.TXT would now come up as having an allocation error—in which the size in the directory no longer matches the number of clusters in the FAT chain. The file would end after cluster 1001 in this example, and the rest of the data would be missing if you loaded this file for viewing. Also, two lost clusters would exist; that is, 1002 and 1003 appear to have no directory entry that owns them. When multiple problems such as these appear, a single incident of damage is often the cause. The repair in this case could involve copying the data from the backup FAT back to the primary FAT, but in most cases, the backup is similarly damaged. Normal utilities would truncate the file and create an entry for a second file out of the lost clusters. You would have to figure out yourself that they really belong together. This is where having knowledge in data recovery can help over using automated utilities that can't think for themselves.

FAT File System Utilities

The CHKDSK, RECOVER, and SCANDISK commands are the DOS damaged-disk recovery team. These commands are crude, and their actions sometimes are drastic, but at times they are all that is available or necessary. RECOVER is best known for its function as a data recovery program, and CHKDSK typically is used for inspection of the file structure. Many users are unaware that CHKDSK can implement repairs to a damaged file structure. DEBUG, a crude, manually controlled program, can help in the case of a disk disaster—if you know exactly what you are doing.

SCANDISK is a safer, more automated, more powerful replacement for CHKDSK and RECOVER in Windows 9x/Me. Windows 2000 and XP do not include SCANDISK and instead have beefed up CHKDSK to handle NTFS.

If you are using MS-DOS 5.0 or older, the only disk testing utilities supplied with your version of MS-DOS are CHKDSK and RECOVER. To learn more about how CHKDSK works, see *Upgrading and Repairing PCs, 11th Edition*, included in electronic form on the DVD packaged with this book.

The *RECOVER* Command

The DOS RECOVER command is designed to mark clusters as bad in the FAT when the clusters can't be read properly. When the system can't read a file because of a problem with a sector on the disk going bad, the RECOVER command can mark the FAT so another file does not use those clusters. When used improperly, this program is highly dangerous. The RECOVER utility has not been included in DOS since version 5 and is not supplied in Windows 9x/Me because its functionality has been replaced by SCANDISK.

Caution

Be very careful when you use **RECOVER**. Used improperly, it can do severe damage to your files and the FAT. If you enter the **RECOVER** command without a filename for it to work on, the program assumes you want every file on the disk recovered and operates on every file and subdirectory on the disk. It converts all subdirectories to files, places all filenames in the root directory, and gives them new names (**FILE0000.REC**, **FILE0001.REC**, and so on). This process essentially wipes out the file system on the entire disk. Do not use **RECOVER** without providing a filename for it to work on. This program should be considered as dangerous as the **FORMAT** command. **RECOVER** is so notorious for this behavior that older versions of the Norton Utilities had a "recover from DOS RECOVER" option that could sometimes undo the damage caused by improper use of **RECOVER**.

An improved version of RECOVER that recovers data from a specified file is only one of the command-line programs provided with Windows NT, 2000, and XP. To use this version of RECOVER, which works with both FAT and NTFS file systems, open a command prompt and enter the command as shown here:

```
RECOVER (drive\folder\filename)
```

For example, to recover all readable sectors from a file called Mynovel.txt stored in C:\My Documents\Writings, you would enter the following command:

```
RECOVER C:\My Documents\Writings\Mynovel.txt
```

Because the NT/2000/XP version of RECOVER requires you to specify a filename and path, it cannot destroy a file system the way the DOS RECOVER command could. However, you should not use wildcards with the NT/2000/XP version of RECOVER. Instead, specify a single filename as shown in this example, or use a third-party tool such as Norton Disk Doctor to check the drive and attempt data recovery from damaged files.

SCANDISK

You should check your FAT partitions regularly for the problems discussed in this chapter and any other difficulties that might arise. By far an easier and more effective solution for disk diagnosis and repair than CHKDSK and RECOVER is the SCANDISK utility, included with DOS 6 and higher versions, as

well as with Windows 9x/Me. This program is more thorough and comprehensive than CHKDSK or RECOVER and can perform the functions of both of them—and a great deal more.

Note

The equivalent of SCANDISK in Windows NT/2000 and Windows XP is called CHKDSK, but it is far more powerful than the old MS-DOS CHKDSK program. For more information on the old MS-DOS CHKDSK program, see *Upgrading and Repairing PCs, 12th Edition*, included in its entirety on the DVD accompanying this book.

SCANDISK is similar to a scaled-down version of third-party disk repair programs such as Norton Disk Doctor, and it can verify both file structure and disk sector integrity. If SCANDISK finds problems, it can repair directories and FATs. If the program finds bad sectors in the middle of a file, it marks the clusters (allocation units) containing the bad sectors as bad in the FAT and attempts to read the file data by rerouting around the defect.

Windows 9x includes both DOS and Windows versions of SCANDISK, which are named SCANDISK.EXE and SCANDSKW.EXE, respectively. Windows scans your drives at the beginning of the operating system installation process and automatically loads the DOS version of SCANDISK whenever you restart your system after turning it off without completing the proper shutdown procedure. You can also launch SCANDISK.EXE from a DOS prompt or from a batch file using the following syntax:

```
Scandisk x: [/a] [/n] [/p] [dblspace.nnn/drvspace.nnn]
x: - designator of the drive that you want to scan
/a - scans all local fixed hard disks
/n - noninteractive mode; requires no user input
/p - scans only, without correcting errors
/custom - runs Scandisk with the options configured in the
➥[CUSTOM] section of the Scandisk.ini file
dblspace.nnn or drvspace.nnn - scans a compressed volume file,
➥where nnn is replaced by the file extension (such as 001)
```

The SCANDISK.INI file, located in the C:\WINDOWS\COMMAND directory on a Windows system by default, contains extensive and well-documented parameters you can use to control the behavior of SCANDISK.EXE. Note that the options in the SCANDISK.INI file are applied only to the DOS version of the utility and have no effect on the Windows GUI version.

You also can run the GUI version of the utility by opening the Start menu and selecting Programs, Accessories, System Tools. Both versions scan and repair the FAT and the directory and file structures, repair problems with long filenames, and scan volumes that have been compressed with DriveSpace or DoubleSpace.

SCANDISK provides two basic testing options: Standard and Thorough. The difference between the two is that the Thorough option causes the program to scan the entire surface of the disk for errors in addition to the items just mentioned. You also can select whether to run the program interactively or let it automatically repair any errors it finds.

The DOS and Windows versions of SCANDISK also test the FAT in different ways. The DOS version scans and, if necessary, repairs the primary copy of the file allocation table. After this, it copies the repaired version of the primary to the backup copy. The Windows version, however, scans both copies of the FAT. If the program finds discrepancies between the two copies, it uses the data from the copy that it judges to be correct and reassembles the primary FAT using the best data from both copies. If the FAT information is not reconstructed correctly, some or all of your data might become inaccessible.

SCANDISK also has an Advanced Options dialog box that enables you to set the following parameters:

- Whether the program should display a summary of its findings
- Whether the program should log its findings

- How the program should repair cross-linked files (two directory entries pointing to the same cluster)
- How the program should repair lost file fragments
- Whether to check files for invalid names, dates, and times

Although `SCANDISK` is good, and is certainly a vast improvement over the old DOS `CHKDSK`, I recommend using one of the commercial packages, such as the Norton Utilities (included with Norton SystemWorks), for any major disk problems. These utilities go far beyond what is included in DOS or Windows.

Disk Defragmentation

The entire premise of the FAT file systems is based on the storage of data in clusters that can be located anywhere on the disk. This enables the computer to store a file of nearly any size at any time. The process of following a FAT chain to locate all the clusters holding the data for a particular file can force the hard disk drive to access many locations on the disk. Because of the physical work involved in moving the disk drive heads, reading a file that is heavily fragmented in this way is slower than reading one that is stored on consecutive clusters.

As you regularly add, move, and delete files on a disk over a period of time, the files become increasingly fragmented, which can slow down disk performance. You can relieve this problem by periodically running a disk defragmentation utility on your drives, such as the one included with Windows 9x. When you run Disk Defragmenter, the program reads each of the files in the disk, using the FAT table to access its clusters, wherever they might be located.

The program then writes the file to a series of contiguous clusters and deletes the original. By progressively reading, erasing, and writing files, the program eventually leaves the disk in a state where all its files exist on contiguous clusters. As a result, the drive is capable of reading any file on the disk with a minimum of head movement, thus providing what is often a noticeable performance increase.

The Windows Defragmentation utility provides this basic defragmenting function. It also enables you to select whether you want to arrange the files on the disk to consolidate the empty clusters into one contiguous free space (which takes longer). The Windows 98/Me version adds a feature that examines the files on the disk and arranges them with the most frequently used program files grouped together at the front of the disk, which can make programs load more quickly.

To show how defragmenting works, see the example of a fragmented file shown in Table 24.32.

Table 24.32 Fragmented File

Directory

Name	Starting Cluster	Size
`PLEDGE.TXT`	1002	2
`USCONST.TXT`	1000	4

FAT16 File Allocation Table

FAT Cluster #	Value	Meaning
00002	0	First cluster available
…	…	…
00999	0	Cluster available
01000	1001	In use, points to next cluster
01001	1004	In use, points to next cluster
01002	1003	In use, points to next cluster

Table 24.32 Continued

FAT16 File Allocation Table		
FAT Cluster #	**Value**	**Meaning**
01003	FFFFh	End of file
01004	1005	In use, points to next cluster
01005	FFFFh	End of file
…	…	…
65526	0	Last cluster available

In the preceding example, the file `USCONST.TXT` is fragmented in two pieces. If you ran a defragmenting program, the files would be read off the disk and rewritten in a contiguous fashion. One possible outcome is shown in Table 24.33.

Table 24.33 Defragmented File

Directory		
Name	**Starting Cluster**	**Size**
`PLEDGE.TXT`	1004	2
`USCONST.TXT`	1000	4
FAT16 File Allocation Table		
FAT Cluster #	**Value**	**Meaning**
00002	0	First cluster available
…	…	…
00999	0	Cluster available
01000	1001	In use, points to next cluster
01001	1002	In use, points to next cluster
01002	1003	In use, points to next cluster
01003	FFFFh	End of file
01004	1005	In use, points to next cluster
01005	FFFFh	End of file
…	…	…
65526	0	Last cluster available

Although it doesn't look like much was changed, you can see that now both files are in one piece, stored one right after the other. Because defragmenting involves reading and rewriting a possibly large number of files on your drive, it can take a long time, especially if you have a large drive with a lot of fragmented files and not very much free working space on the drive.

Third-party defragmentation utilities, such as the Speed Disk program included in the Norton Utilities, provide additional features, such as the capability to select specific files that should be moved to the front of the disk. Speed Disk also can defragment the Windows swap file and files that are flagged with the system and hidden attributes, which Disk Defragmenter will not touch.

Caution

Although the disk-defragmentation utilities included with Windows and third-party products are usually quite safe, you should always be aware that defragmenting a disk is an inherently dangerous procedure. The program reads, erases, and rewrites every file on the disk and has the potential to cause damage to your data when interrupted improperly. Although I have never seen a problem result from the process, an unforeseen event—such as a power failure—during a defragmentation procedure can conceivably be disastrous. I strongly recommend that you always run a disk-repair utility, such as `SCANDISK`, on your drives before defragmenting them and have a current backup ready.

Windows NT 4.0 does not include a defragmentation utility, but Windows 2000 and Windows XP do include such a utility. Note that third-party defragmentation utilities available for recent and current versions of Windows, such as Golden Bow Systems' VoptXP (`www.vopt.com`), are often faster and offer more features than Windows's own defragmentation programs.

Third-Party Programs

When you get a `Sector not found` error reading drive C:, the best course of action is to use one of the third-party disk-repair utilities on the market, rather than DOS's `RECOVER` or even `SCANDISK`. The Norton Utilities by Symantec (also included in Norton SystemWorks) stands as perhaps the premier data recovery package on the market today. This package is comprehensive and automatically repairs most types of disk problems.

Norton Utilities and Norton SystemWorks

Programs such as Norton Disk Doctor can perform much more detailed repairs with a greater amount of safety. Disk Doctor preserves as much of the data in the file as possible and can mark the FAT so the bad sectors or clusters of the disk are not used again. These programs also save Undo information, enabling you to reverse any data recovery operation.

Disk Doctor is part of Symantec's Norton Utilities package, which includes a great many other useful tools. For example, Norton Utilities has an excellent sector editor (Norton Disk Editor) that enables you to view and edit any part of a disk, including the master and volume boot records, FATs, and other areas that fall outside the disk's normal data area. Currently, no other program is as comprehensive or as capable of editing disks at the sector level. The disk editor included with Norton Utilities can give the professional PC troubleshooter or repairperson the ability to work directly with any sector on the disk, but this does require extensive knowledge of sector formats and disk structures. The documentation with the package is excellent and can be very helpful if you are learning data recovery on your own.

Note

Data recovery is a lucrative service that the more advanced technician can provide. People are willing to pay much more to get their data back than to replace a hard drive.

You also can create a rescue disk for restarting your system and testing the drive in case of emergencies, unerase accidentally deleted files (even if the Recycle Bin was bypassed), and unformat accidentally formatted drives.

Norton Utilities is now available in version 2003 as part of Norton SystemWorks 2003 (for Windows 98/Me/NT/2000/XP). SystemWorks 2003 also includes Norton Anti-Virus and Norton CleanSweep uninstaller; the Professional version also includes Norton Ghost disk imaging. Some of the programs provided with Norton Utilities are designed to be run from the command line or from a DOS prompt, such as the following:

- Norton Disk Doctor (`NDD.EXE`)
- Disk Editor (`DISKEDIT.EXE`)
- UnErase (`UNERASE.EXE`)
- UnFormat (`UNFORMAT.EXE`)
- Rescue Restore (`RESCUE.EXE`)

However, most Norton Utilities programs are designed to be run from within the Windows GUI. You must use the Windows GUI to work on NTFS-based drives. You should not use older versions of Norton Utilities, such as version 8.0 (designed for Windows 3.1 and MS-DOS), with 32-bit versions of Windows because of the possibility of data loss due to a lack of support for long filenames and large drives.

SpinRite 5

The SpinRite 5 program from Gibson Research (`http://grc.com`) provides testing of both IDE and SCSI hard drives as well as removable-media drives of all sizes that have FAT-type file systems, including FAT32. During testing, SpinRite 5 statistically analyzes hard-to-read disk sectors to determine their original contents; it then moves the data to a properly working area of the disk. This process takes longer than with a program such as DiskDoctor, but for heavily damaged disks and drives for which no backup is available, this process can be more successful at recovering data. Another handy feature of SpinRite 5 is its capability to query your drive's partition table to determine the disk geometry originally used to set up the disk. This enables you to access data on a disk even if the factory disk geometry values were not used to prepare the disk with FDISK and were not written down for future use.

File Systems and Third-Party Disk Utilities

The most important consideration when you purchase third-party disk utilities is to choose products that support your file system. For example, Norton Utilities supports both FAT (including FAT32) and NTFS files systems. On the other hand, SpinRite 5 supports all FAT file systems but not NTFS. Never use a disk utility not designed for your file system, and never use an out-of-date disk utility (designed for an earlier operating system) on your disk. In both cases, you could cause irreparable damage to the data on your drive.

New Technology File System

New Technology File System (NTFS) is the native file system of Windows NT, Windows 2000, and Windows XP. Windows 2000 and Windows XP use an enhanced version of NTFS called NTFS 5 or NTFS 2000; Windows NT 4.0 must have Service Pack 4 or above installed to be capable of accessing an NTFS 5/NTFS 2000 disk. Although NT/2000/XP support FAT partitions (and Windows 2000/XP even support FAT32), NTFS provides many advantages over FAT, including long filenames, support for larger files and partitions, extended attributes, and increased security. NTFS, like all of Windows NT, was newly designed from the ground up. Backward-compatibility with previous Microsoft operating systems was not a concern because the developers were intent on creating an entirely new 32-bit platform. As a result, no operating systems other than Windows NT and Windows 2000/XP, which are based on Windows NT, can read NTFS partitions.

This is an important consideration. All the other Windows operating systems are fundamentally based on DOS and give you the alternative of booting to a DOS prompt in special situations. Whether you have to run a special DOS utility to configure a piece of hardware or perform an emergency disk repair, the DOS option is always there.

Windows NT/2000/XP are not DOS based. You can open a window that provides a DOS-like command prompt in Windows NT/2000/XP, but this is actually a DOS emulation—not a true shell. You can't boot the system to a command prompt, avoiding the GUI, as you can with Windows 9x unless you install the Windows 2000 or Windows XP Recovery Console. As the name implies, the Recovery Console is designed for system repair and maintenance procedures, not for routine use. You can, of course, bypass NT/2000/XP entirely by starting the system with a DOS boot disk, but if you have NTFS partitions on your drives, you can't access them from the DOS prompt.

NTFS supports filenames of up to 255 characters, using spaces, multiple periods, and any other standard characters except the following: *?/\;<>|. Because NTFS file offsets are 64 bits long, files and partitions can be truly enormous—up to 16 EiB (exbibytes) in size (1 EiB = 2^{60} bytes = 1,048,576TiB)! To give you an idea of just how much data this is, it was estimated that all the words ever spoken by humans throughout history would occupy 5 EiB of storage.

Since Windows NT 3.51, NTFS has also supported compression on a file-by-file (or folder-by-folder) basis through each file or folder's properties sheet. No third-party program, such as WinZip or PKZip, is needed to compress or decompress files stored on an NTFS drive. NTFS 5 also supports encryption on a file-by-file or folder-by-folder properties sheet.

Tip

To access advanced file attributes such as compression, encryption, and indexing, right-click the file or folder, select Properties, and click the Advanced button to bring up the Advanced properties dialog box.

NTFS Architecture

Although NTFS partitions are very different from FAT partitions internally, they do comply with the extra-partitional disk structures described earlier in this chapter. NTFS partitions are listed in the master partition table of a disk drive's master boot record, just like FAT partitions, and they have a volume boot record as well, although it is formatted somewhat differently.

When a volume is formatted with NTFS, system files are created in the root directory of the NTFS volume. These system files can be stored at any physical location on the NTFS volume. This means that damage to any specific location on the disk will probably not render the entire partition inaccessible.

Typically, 12 NTFS system files (often referred to as *metadata* files) are created when you format an NTFS volume. Table 24.34 shows the names and descriptions for these files.

Table 24.34 NTFS System Files

Filename	Meaning	Description
`$mft`	Master file table (MFT)	Contains a record for every file on the NTFS volume in its `Data` attribute
`$mftmirr`	Master file table2 (MFT2)	Mirror of the MFT used for recoverability purposes; contains the first four records in `$mft` or the first cluster of `$mft`, whichever is larger
`$badclus`	Bad cluster file	Contains all the bad clusters on the volume
`$bitmap`	Cluster allocation bitmap	Contains the bitmap for the entire volume, showing which clusters are used
`$boot`	Boot file	Contains the volume's bootstrap if the volume is bootable
`$attrdef`	Attribute definitions table	Contains the definition of all system- and user-defined attributes on the volume
`$logfile`	Log file	Logs file transactions; used for recoverability purposes
`$quota`	Quota table	Table used to indicate disk quota usage for each user on a volume; used in NTFS 5
`$upcase`	Upcase table	Table used for converting uppercase and lowercase characters to the matching uppercase Unicode characters
`$volume`	Volume	Contains volume information, such as volume name and version
`$extend`	NTFS extension file	Stores optional extensions, such as quotas, object identifiers, and reparse point data
(no name)	Root filename index	The root folder (directory)

An NTFS partition is based on a structure called the master file table (MFT). The MFT concept expands on that of the FAT. Instead of using a table of cluster references, the MFT contains much more detailed information about the files and directories in the partition. In some cases, it even contains the files and directories themselves.

The first record in the MFT is called the *descriptor*, which contains information about the MFT itself. The volume boot record for an NTFS partition contains a reference that points to the location of this descriptor record. The second record in the MFT is a mirror copy of the descriptor, which provides fault tolerance, should the first copy be damaged.

The third record is the log file record. All NTFS transactions are logged to a file that can be used to restore data in the event of a disk problem. The bulk of the MFT consists of records for the files and directories stored on the partition. NTFS files take the form of objects that have both user- and system-defined attributes. Attributes on NTFS partitions are more comprehensive than the few simple flags used on FAT partitions. All the information on an NTFS file is stored as attributes of that file. In fact, even the file data itself is an attribute. Unlike FAT files, the attributes of NTFS files are part of the file itself; they are not listed separately in a directory entry. Directories exist as MFT records as well, but they consist mainly of indexes listing the files in the directory—they do not contain the size, date, time, and other information about the individual files.

Thus, an NTFS drive's MFT is much more than a cluster list, like a FAT; it is actually the primary data storage structure on the partition. If a file or directory is relatively small (less than approximately 1,500 bytes), the entire file or directory might even be stored in the MFT. For larger amounts of storage, the MFT record for a file or directory contains pointers to external clusters in the partition. These external clusters are called *extents*. All the records in the MFT, including the descriptors and the log file, are capable of using extents for storage of additional attributes. The attributes of a file that are part of the MFT record are called *resident* attributes, whereas those stored in extents are called *nonresident* attributes.

NTFS 5 (NTFS 2000)

Along with Windows 2000 came a new variation of NTFS called NTFS 5 (also called NTFS 2000); NTFS 5 is also used by Windows XP. This update of the NT file system includes several new features that are exploited and even required by Windows 2000 and Windows XP. Because of this, when you install Windows 2000 or Windows XP, any existing NTFS volumes automatically are upgraded to NTFS 5 (there is no way to override this option). If you also run Windows NT versions earlier than Windows NT 4 Service Pack 4 (SP4), NT 4 is no longer capable of accessing the NTFS 5 volumes. If you want to run both NT 4 and Windows 2000 or Windows XP on the same system (as in a dual-boot configuration), you must upgrade NT 4 by installing Service Pack 4 or later. An updated `NTFS.SYS` driver in Service Pack 4 enables NT 4 to read from and write to NTFS 5 volumes.

New features of the NTFS 5 file system include:

- *Disk quotas*. System administrators can limit the amount of disk space users can consume on a per-volume basis. The three quota levels are Off, Tracking, and Enforced.
- *Encryption*. NTFS 5 can automatically encrypt and decrypt files as they are read from and written to the disk (not available in Windows XP Home Edition).
- *Reparse points*. Programs can trap open operations against objects in the file system and run their own code before returning file data. This can be used to extend file system features such as mount points, which you can use to redirect data read and written from a folder to another volume or physical disk.
- *Sparse files*. This feature enables programs to create very large files as placeholders but to consume disk space by adding to the files only as necessary.
- *USN (Update Sequence Number) Journal*. Provides a log of all changes made to files on the volume.
- *Mounted drives*. You can attach volumes including drives and folders to an empty folder stored on an NTFS drive. For example, you can create a folder on one drive that allows you to access content stored on another drive.

Because these features—especially the USN Journal—are required for Windows 2000 to run, a Win2000/Server 2003 domain controller must use an NTFS 5 partition as the system volume.

NTFS Changes in Windows XP

The location of the MFT has changed in Windows XP versus Windows 2000 and Windows NT. In Windows 2000 and Windows NT, the MFT is typically located at the start of the disk space used by the NTFS file system. In Windows XP, the `$logfile` and `$bitmap` metadata files are located 3GB from the start of the disk space used by NTFS. As a result, system performance has been increased by 5%–8% in Windows XP over Windows 2000 or Windows NT.

Another improvement in Windows XP's implementation of NTFS is the amount of MFT information read into memory. During bootup, Windows XP reads only a few hundred kilobytes of MFT information if all drives are formatted with NTFS. However, if some or all of the drives are formatted with FAT32, many megabytes of information (the amount varies by the number of drives and the size of the drives) must be read during bootup. Thus, using NTFS utilizes system memory more efficiently.

NTFS Compatibility

Although NTFS partitions are not directly accessible by DOS and other operating systems, Windows NT/2000 is designed for network use, so other operating systems are expected to be capable of accessing NTFS files via the network. For this reason, NTFS continues to support the standard DOS file attributes and the 8.3 FAT naming convention.

One of the main reasons for using NTFS is the security it provides for its files and directories. NTFS security attributes are called *permissions* and are designed to enable system administrators to control access to files and directories by granting specific rights to users and groups. This is a much more granular approach than the FAT file system attributes, which apply to all users.

However, you can still set the FAT-style attributes on NTFS files using the standard Windows NT/2000 file management tools, including Windows NT/2000 Explorer and even the command-prompt `ATTRIB` command. When you copy FAT files to an NTFS drive over the network, the FAT-style attributes remain in place until you explicitly remove them. This can be an important consideration because the FAT-style attributes take precedence over the NTFS permissions. A file on an NTFS drive that is flagged with the FAT read-only attribute, for example, can't be deleted by a Windows NT/2000 user, even if that user has NTFS permissions that grant her full access.

To enable DOS and 16-bit Windows systems to access files on NTFS partitions over a network, the file system maintains an 8.3 alias name for every file and directory on the partition. The algorithm for deriving the alias from the long filename is the same as that used by Windows 95's VFAT. Windows NT/2000 also provides its FAT partitions with the same type of long filename support used by VFAT, allocating additional directory entries to store the long filenames as necessary.

Creating NTFS Drives

NTFS is for use on hard disk drives. You can't create an NTFS floppy disk (although you can format removable media, such as Iomega Zip and Jaz cartridges to use NTFS). Three basic ways to create an NTFS disk partition are as follows:

- Create a new NTFS volume out of unpartitioned disk space during the Windows NT/2000 installation process or after the installation with the Disk Administrator utility.
- Format an existing partition to NTFS (destroying its data in the process), using the Windows NT Format dialog box (accessible from Windows NT Explorer or Disk Administrator) or the `FORMAT` command (with the `/fs:ntfs` switch) from the command prompt.
- Convert an existing FAT partition to NTFS (preserving its data) during the Windows NT/2000 installation process or after the installation with the command-line `CONVERT` utility.

NTFS Tools

Because it uses a fundamentally different architecture, virtually none of the troubleshooting techniques outlined earlier in this chapter are applicable when dealing with NTFS partitions, nor can the disk utilities intended for use on FAT partitions address them. Windows NT has a rudimentary capability to check a disk for file system errors and bad sectors with its own version of `CHKDSK`, but apart from that, the operating system contains no other disk repair or defragmentation utilities. Windows 2000 and Windows XP include a command-line and GUI version of `CHKDSK` and also include a defragmenting tool that is run from the Windows Explorer GUI. Windows XP Professional also includes `DSKPROBE`, a direct disk sector editor, in its Windows Support Tools (the Windows 2000 Resource Kit also contains `DSKPROBE`).

One difference between Windows 2000/XP's CHKDSK and Windows 9x/Me's SCANDISK is that CHKDSK cannot fix file system errors if it is run within the Windows GUI. If you run CHKDSK and select the Automatically Fix File System Errors option, you must schedule CHKDSK to run at the next system startup. You can run CHKDSK without this option to find file system problems; you can also run CHKDSK within the Windows GUI to look for and attempt to fix bad sectors.

Tip

You can use the **`FSUTIL`** command-line program in Windows 2000 and Windows XP to learn more about a particular drive's file system, including whether you need to run CHKDSK or a third-party disk repair tool. For example, to determine whether drive D: is *dirty* (has file system errors requiring repair), you would open a Windows 2000/XP command prompt and enter this command:

```
FSUTIL DIRTY QUERY D:
```

For more examples of **`FSUTIL`** commands and syntax, enter **`FSUTIL`** with no options at a Windows 2000/XP command prompt.

The NTFS file system, however, does have its own automatic disk repair capabilities. In addition to Windows NT/2000/XP's fault-tolerance features, such as disk mirroring (maintaining the same data on two separate drives) and disk striping with parity (splitting data across several drives with parity information for data reconstruction), the OS has two features to help improve reliability:

- Transaction management
- Cluster remapping

NTFS can roll back any *transaction* (its term for a change to a file stored on an NTFS volume) if it isn't completed properly due to disk errors, running out of memory, or errors such as removing media or disconnecting a device before the transaction process is complete. Each transaction has five steps:

1. NTFS creates a log file of the metadata operations of the transaction (file updates, erasure, and so on) and caches the file in system memory.
2. NTFS stores the actual metadata operations in memory.
3. NTFS marks the transaction record in the log file as committed.
4. NTFS saves the log file to disk after the transaction is complete.
5. NTFS saves the actual metadata operations to disk after the transaction is complete.

This process is designed to prevent random data (lost clusters) on NTFS drives.

With cluster remapping, when Windows (NT/2000/XP) detects a bad sector on an NTFS partition, it automatically remaps the data in that cluster to another cluster. If the drive is part of a fault-tolerant drive array, any lost data is reconstructed from the duplicate data on the other drives.

Despite these features, however, there is still a real need for third-party disk repair and defragmentation utilities for Windows NT/2000/XP. These were scarce when Windows NT was first released, but third-party utilities that can repair and defragment NTFS drives are now widely available. One I recommend is Norton Utilities 2003 (also included in Norton SystemWorks 2003 and Norton SystemWorks Pro 2003) by Symantec, which works with Windows 98, Windows Me, Windows 2000, and Windows XP. Earlier versions work with Windows NT 4.0 and Windows 95. If you are looking for even faster defragmentation, Golden Bow's VoptXP (which also supports Windows 9x, Me, and 2000) is a longtime favorite.

High Performance File System

The High Performance File System (HPFS) was first introduced in OS/2 version 1.2 and included many additional features over the existing FAT16 system available at the time. HPFS was also supported in Windows NT 3.x versions but not any later versions. In many ways, HPFS was the predecessor to the NTFS that later debuted in Windows NT.

For more information about HPFS, see the Technical Reference section of the DVD packaged with this book.

Data Recovery

Recovering lost data can be as simple as opening the Recycle Bin, or it might require spending hundreds of dollars on specialized data recovery software or services. In the worst-case scenario, you might even need to send your drive to a data recovery center. Several factors affect the degree of difficulty you can have in recovering your data, including

- How the data was deleted
- Which file system was used by the drive on which the data was stored
- Whether the drive uses magnetic, optical, magneto-optical, or flash memory to store data
- Which version of Windows or other OS you use
- Whether you already have data-protection software installed on your system
- Whether the drive has suffered physical damage to heads, platters, or its circuit board

The Windows Recycle Bin and File Deletion

The simplest data recovery of all takes place when you send files to the Windows Recycle Bin (a standard part of Windows since Windows 95). Pressing the Delete key when you have a file or group of files highlighted in Windows Explorer or My Computer or clicking the Delete button sends files to the Recycle Bin. Although a file sent to the Recycle Bin is no longer listed in its normal location by Windows Explorer, the file is actually protected from being overwritten. By default, Windows 95 and above reserve 10% of the disk space on each hard disk for the Recycle Bin (removable-media drives don't have a Recycle Bin). Thus, a 10GB drive reserves about 1GB for its Recycle Bin. In this example, as long as less than 1GB of files has been sent to the Recycle Bin, a so-called *deleted* file is protected by Windows. However, after more than 1GB of files has been sent to the Recycle Bin, Windows allows the oldest files to be overwritten. Thus, the quicker you realize that a file has been sent to the Recycle Bin, the more likely it is you can retrieve it.

To retrieve a file from the Recycle Bin, open the Recycle Bin, select the file, right-click it, and select Restore. Windows lists the file in its original location and removes it from the Recycle Bin.

If you hold down the Shift key when you select Delete or press the Delete key, the Recycle Bin is bypassed. Retrieving lost data at this point requires third-party data recovery software.

Recovering Files That Are Not in the Recycle Bin

The Recycle Bin is a useful first line of defense against data loss, but it is quite limited. As you learned in the previous section, it can be bypassed when you select files for deletion, and files stored in the Recycle Bin are eventually kicked out by newer deleted files. Also, the Recycle Bin isn't used for files deleted from a command prompt or when an older version of a file is replaced by a newer version.

Products such as Norton UnErase (part of the Norton Utilities and Norton SystemWorks) are necessary if you want to retrieve files not in the Recycle Bin. However, the effectiveness of Norton UnErase and how you should use it depends on the version of Windows you use and the file system used by your drives.

Norton UnErase and Norton Protected Recycle Bin—Win9x/Me

With Windows 9x/Me, which use the FAT file system, retrieving data from a drive that doesn't have Norton Utilities installed isn't difficult. However, installing Norton Utilities before you start to delete files that you might want to retrieve makes it even easier. You can run Norton UnErase from the bootable CD included in current versions and run UnErase as a command-prompt program if you don't have it already installed and need to retrieve erased data. You will need to provide the first letter of each file you want to unerase.

Caution

Do *not* install data-recovery software to a drive you are attempting to retrieve data from because you might overwrite the data you are attempting to retrieve. If you are trying to recover data from your Windows startup drive, install another hard disk into your system, configure it as a boot drive in the system BIOS, install a working copy of Windows on it, boot from that drive, and install your data recovery software to that drive. If possible, install a drive large enough (at least 10GB or larger) so that you have several GB of free space on it for storing recovered data.

However, if you have already installed Norton Utilities, you probably have the Norton Protected Recycle Bin on your desktop in place of the regular Recycle Bin. Compared to the Windows standard-model Recycle Bin, the Norton Protected Recycle Bin protects files that have been replaced with newer versions and files that were deleted from a command prompt. To retrieve a file stored in the Norton Protected Recycle Bin, open the Recycle Bin, select the file you want to retrieve, right-click it, and select Retrieve to put it back in its original location.

Alternatively, you can start the Norton Unerase Wizard from the Norton Utilities menu. You can search for recently deleted files (these files are stored in the Recycle Bin), all protected files on local drives (also stored in the Recycle Bin), and any recoverable files on local drives. When you select the last option, you can narrow down the search with wildcards or file types and specify which drives to search. You must supply the first letter of the filename for files that were not stored in the Recycle Bin; you can also see which files were deleted by a particular program. To undelete a file with the Unerase Wizard, select the file, provide the first letter of the filename if necessary, click Quick View to view the file (if your file viewer supports the file format), and click Recover to restore the file to its original location.

With Windows 9x/Me, you can search both hard and removable-media (floppy, flash memory) drives for lost files, although the Recycle Bin works only for hard drives.

Norton UnErase and Norton Protected Recycle Bin—Win 2000/XP

Norton UnErase and Norton Protected Recycle Bin work in a similar fashion with Windows 2000/XP as with Windows 9x/Me, but with a significant exception: The Unerase Wizard can search only hard drives. Removable-media drives are not supported.

Tip

If you want to use Norton Unerase Wizard to retrieve data from a floppy drive and don't have Windows 9x/Me installed on your system, start your computer with the Norton bootable CD and run Unerase from the CD startup menu.

If you have a dual-boot system that can run Windows 9x/Me and Windows 2000/XP and you also want to use Norton Unerase Wizard with Zip, USB, and flash memory devices, start the computer with Windows 9x/Me, install Norton Utilities or SystemWorks under Windows 9x/Me, and start your computer with Windows 9x/Me if you need to search removable media drives for lost data.

Alternatives to Norton UnErase

VCOM's System Suite 4.0 (previously sold by Ontrack) is an integrated utility suite that offers an undelete feature similar in many ways to Norton UnErase. However, System Suite's FileUndeleter works with removable-media drives as well as hard drives under all supported versions of Windows, including Windows XP.

Although it's not an automatic tool, you can use Norton's Disk Editor (DISKEDIT.COM) to retrieve lost data from hard, floppy, and most types of removable-media drives under any file system and most operating systems, including Linux. See the section "Using the Norton Disk Editor" later in this chapter.

Undeleting Files in NTFS

Because the file structure of NTFS is much more complex than any FAT file system version and some files might be compressed using NTFS's built-in compression, you should use an NTFS-specific file undeletion program to attempt to recover deleted files from an NTFS drive. For example, you should use a version of Norton Utilities or Norton SystemWorks compatible with NTFS, such as the 2002 or later versions. Also, you should enable the Norton Protection feature, which stores deleted files for a specified period of time before purging them from the system. Using Norton Protection will greatly enhance Norton UnErase's capability to recover deleted files.

If you need to recover deleted files and have not already installed an undelete program such as Norton Utilities or Norton SystemWorks' Norton UnErase, you should consider a standalone file recovery program, such as

- *Active Undelete*. This series of products also works with flash memory cards; more information and a free demo are available from `http://www.active-undelete.com`.
- *Restorer 2000*. Available in FAT, NTFS, and Professional versions; more information and a free demo are available from `http://www.bitmart.net/r2k.shtml`.
- *Ontrack EasyRecovery*. More information and a free demo are available from `http://www.ontrack.com`.

Tip

Some file-undelete products for NTFS can undelete only files created by the currently logged-in user, whereas others require the administrator to be logged in. Check the documentation for details, particularly if you are trying to undelete files from a system with more than one user.

Retrieving Data from Partitioned and Formatted Drives

When a hard disk, floppy disk, or removable-media drive has been formatted, its file allocation table, which is used by programs such as Norton UnErase or VCOM System Suite's FileUndeleter to determine the location of files, is lost. If a hard drive has been repartitioned with FDISK or another partitioning program (such as Windows 2000/XP's Disk Management), the original file system and partition information is lost (as is the FAT).

In such cases, more powerful data-recovery tools must be used to retrieve data. To retrieve data from an accidentally formatted drive, you have two options:

- Use a program that can unformat the drive.
- Use a program that can bypass the newly created FAT and read disk sectors directly to discover and retrieve data.

To retrieve data from a drive that has been partitioned, you must use a program that can read disk sectors directly.

Norton Unformat and Its Limitations

Norton Utilities and Norton SystemWorks offer Norton Unformat, which can be launched from the bootable CD to unformat an accidentally formatted FAT drive. However, Norton Unformat has significant limitations with today's file systems and drive types, including the following:

- *Norton Unformat doesn't support NTFS drives*. This means many Windows 2000 and XP-based systems can't use it for data recovery.
- *Norton Unformat cannot be used with drives that require device drivers to function, such as removable-media drives.*
- *Norton Unformat works best if the Norton Image program has been used to create a copy of the FATs and root directory.* If the image file is out-of-date, Unformat might fail; if the image file is not present, Unformat cannot restore the root directory and the actual names of folders in the root directory will be replaced by sequentially numbered folder names.
- *Norton Unformat cannot copy restored files to another drive or folder.* It restores data back to the same drive and partition. If Unformat uses an out-of-date file created by Norton Image to determine where data is located, it could overwrite valid data on the drive being unformatted.

For these reasons, Norton Unformat is not the most desirable method for unformatting a drive. You can use the powerful, but completely manual, Norton Disk Editor (DISKEDIT) to unformat a drive or retrieve data from a formatted drive, but other alternatives are simpler.

Retrieving Lost Data to Another Drive

Many products on the market can retrieve lost data to another drive, whether the data loss was due to accidental formatting or disk partitioning. One of the best and most comprehensive products is the EasyRecovery product line from Ontrack DataRecovery Services, a division of Kroll Ontrack, Inc. The EasyRecovery product line includes the following products:

- *EasyRecovery DataRecovery.* Recovers data from accidentally formatted or deleted hard, floppy, and removable-media drives and repairs damaged or corrupted Zip and Microsoft Word files. Local and network folders can be used for recovered files.
- *EasyRecovery FileRepair.* Repairs and recovers data from damaged or corrupted Zip and Microsoft Office (Word, Excel, Access, PowerPoint, and Outlook) files. Local and network folders can be used for recovered files.
- *EasyRecovery Professional.* Combines the features of DataRecovery and FileRecovery and adds features such as file type search, RawRecovery, and user-defined partition parameters to help recover data from more severe forms of file system corruption and accidental partitioning. A free trial version displays files that can be recovered (and repairs and recovers Zip files at no charge); it can be downloaded from the Ontrack Web site (`http://www.ontrack.com`).

An earlier version of EasyRecovery Data Recovery Lite can recover up to 50 files and is included as part of VCOM's System Suite (previously sold by Ontrack).

When you start EasyRecovery Professional, you can choose from several recovery methods, including these:

- *DeletedRecovery.* Recovers deleted files
- *FormatRecovery.* Recovers files from accidentally formatted drives
- *RawRecovery.* Recovers files with direct sector reads using file-signature matching technology
- *AdvancedRecovery.* Recovers data from deleted or corrupted partitions

In each case, you need to specify another drive to receive the retrieved data. This read-only method preserves the contents of the original drive and enables you to use a different data-recovery method if the first method doesn't recover the desired files.

Which options are best for data recovery? Table 24.35 shows the results of various data-loss scenarios and recovery options when EasyRecovery Professional was used to recover data from a 19GB logical drive formatted with the NTFS file system under Windows XP.

Table 24.35 Data Recovery Options and Results with EasyRecovery Professional

Type of Data Loss	Data Recovery Method	Data Recoverable?	Details	Notes
Deleted folder	DeletedRecovery	Yes	All files recovered.	All long file and folder names preserved.
Formatted drive (full format)	FormatRecovery	Yes	All files recovered.	New folders created to store recovered files; long filenames preserved for files and folders beneath root folder level.
Logical drive deleted with Disk Management	AdvancedRecovery	Yes	All files and folders recovered.	All long file and folder names preserved.
Formatted drive with new data copied to it	FormatRecovery	Partial	Files and folders that were not overwritten were recovered.	Long filenames and folders preserved.
Formatted, repartitioned drive reformatted as FAT32 (117MB Disk 1)	AdvancedRecovery	No	Could not locate any files to recover.	
	RawRecovery	Partial	Nonfragmented files recovered.	Original directory structure and filenames lost; each file type stored in a separate folder and files numbered sequentially.
Formatted, repartitioned drive formatted as NTFS (18.8GB Disk 2)	AdvancedRecovery	No	Could not locate any files to recover.	
	RawRecovery	Partial	Nonfragmented files recovered.	Original directory structure and filenames lost; each file type stored in a separate folder and files numbered sequentially.

As Table 24.35 makes clear, as long as the data areas of a drive are not overwritten, complete data recovery is usually possible—even if the drive has been formatted or repartitioned. Thus, it's critical that you react quickly if you suspect you have partitioned or formatted a drive containing valuable data. The longer you wait to recover data, the less data will be available for recovery. In addition, if you must use a sector-by-sector search for data (a process called RawRecovery by Ontrack), your original folder structure and long filenames will not be saved. You will therefore need to re-create the desired directory structure and rename files after you recover them—a very tedious process.

Tip

If you use EasyRecovery Professional or EasyRecovery DataRecovery to repair damaged Zip or Microsoft Office files, use the Properties menu to select a location for repaired files (the original location or another drive or folder). By default, repaired Outlook files are copied to a different folder, whereas other file types are repaired in place unless you specify a different location.

As you can see from this example, dedicated data-recovery programs such as Ontrack EasyRecovery Professional are very powerful. However, they are also very expensive. If you have Norton Utilities or Norton SystemWorks and don't mind taking some time to learn about disk structures, you can perform data recovery with the Norton Disk Editor.

Using the Norton Disk Editor

In my PC Hardware (Upgrading and Repairing) and Data Recovery/Computer Forensics seminars, I frequently use the Norton Disk Editor—an often-neglected program that's part of the Norton Utilities and Norton SystemWorks—to explore drives. I also use Disk Editor to retrieve lost data. Because Disk Editor is a manual tool, it can sometimes be useful even when friendlier automatic programs don't work correctly or are unavailable. For example, in physical sector mode, Disk Editor can be used with any drive regardless of what file system was used, since at that level it is working underneath the OS. Additionally, because Disk Editor displays the structure of your drive in a way other programs don't, it's a perfect tool for learning more about disk drive structures as well as recovering lost data. This section discusses two of the simpler procedures you can perform with Disk Editor:

- Undeleting a file on a floppy disk
- Copying a deleted file on a hard disk to a different drive

If you have Norton SystemWorks, SystemWorks Professional, or Norton Utilities for Windows, you have Norton Disk Editor. To determine whether it's installed on your system, look in the Norton Utilities folder under the Program Files folder for the following files: `DISKEDIT.EXE` and `DISKEDIT.HLP`.

If you don't find these files on your hard disk, you can run them directly from the Norton installation CD. If you have SystemWorks or SystemWorks Professional, look for the CD folder called `\NU` to locate these files.

Disk Edit is a command prompt program designed primarily to access FAT-based file systems such as FAT12 (floppy disks), FAT16 (MS-DOS and early Windows 95 hard disks), and FAT32 (Windows 95B/Windows 98/Me hard disks). You can use Disk Edit with Windows NT, Windows 2000, and Windows XP if you prepared the hard disks with the FAT16 or FAT32 file systems. Disk Edit will also work on NTFS volumes, however in that case it can only be used in physical sector mode.

I strongly recommend that you first use Disk Editor with floppy disks you have prepared with noncritical files before you use it with a hard disk or vital files. Because Disk Editor is a completely manual program, the opportunities for error are high.

The Disk Edit files can easily fit on a floppy disk, but if you are new to the program, you might want to put them on a different drive from one you will be examining or repairing. *Never* copy Disk Edit

files (or any other data recovery program) to a drive that contains data you are trying to recover because the files might overwrite the data area and destroy the files you want to retrieve. For example, if you are planning to examine or repair floppy disks, create a folder on your hard disk called `Disk Edit` and copy the files to that folder.

You can use Disk Editor without a mouse by using keyboard commands, but if you want to use it with a mouse, you can do so if your mouse attaches to the serial or PS/2 mouse ports (USB mice generally don't work from the command prompt, but if your USB mouse has a PS/2 mouse port adapter, you can use it by plugging the mouse and adapter into the PS/2 port). You must load an MS-DOS mouse driver (usually MOUSE.COM) for your mouse before you start Disk Editor. If you have a Logitech mouse, you can download an MS-DOS mouse driver from the Logitech Web site. If you have a Microsoft mouse, Microsoft doesn't provide MS-DOS drivers you can download, but you can get them from the following Web site:

`http://www.bootdisk.com/readme.htm#mouse`

For other mice, try the Microsoft or Logitech drivers, or contact the vendor for drivers. Keep in mind that scroll wheels and other buttons won't work with an MS-DOS driver.

I recommend you copy your mouse driver to the same folder in which Disk Editor is located.

Using Disk Editor to Examine a Drive

To start the program, do the following:

1. Boot the computer to a command prompt (not Windows); Disk Editor needs exclusive access to the drives you plan to examine. If you use Windows 9x, press F8 or Ctrl to bring up the startup menu and select Safe Mode Command Prompt, or use the Windows 9x/Me Emergency Startup disk (make one with Add/Remove Programs). If you use Windows 2000 or XP, insert a blank floppy disk into drive A:, right-click drive A: in My Computer, and select Format. Select the Create an MS-DOS Startup Disk option and use this disk to start your computer.
2. Change to the folder containing your mouse driver and Disk Editor.
3. Type **`MOUSE`** (if your mouse driver is called MOUSE.COM or MOUSE.EXE; otherwise, substitute the correct name if it's called something else). Then press Enter to load the mouse driver.
4. Type **`DISKEDIT`** and press Enter to start the program. If you don't specify a drive, Disk Editor scans the drive on which it's installed. If you are using it to work with a floppy disk, enter the command **`DISKEDIT A:`** to direct it to scan your floppy disk. Disk Editor scans your drive to determine the location of files and folders on the disk.
5. The first time you run Disk Editor, a prompt appears to remind you that Disk Editor runs in read-only mode until you change its configuration through the Tools menu. Click OK to continue.

After Disk Editor has started, you can switch to the drive you want to examine or recover data from. To change to a different drive, follow these steps:

1. Press Alt+O to open the Object menu.
2. Select Drive.
3. Select the drive you want to examine from the Logical Disks menu.
4. The disk structure is scanned and displayed in the Disk Editor window.

Disk Editor normally starts in Directory mode, but you can change it to other modes with the View menu. When you view a drive containing data in Directory mode, you will see a listing similar to the one shown in Figure 24.2.

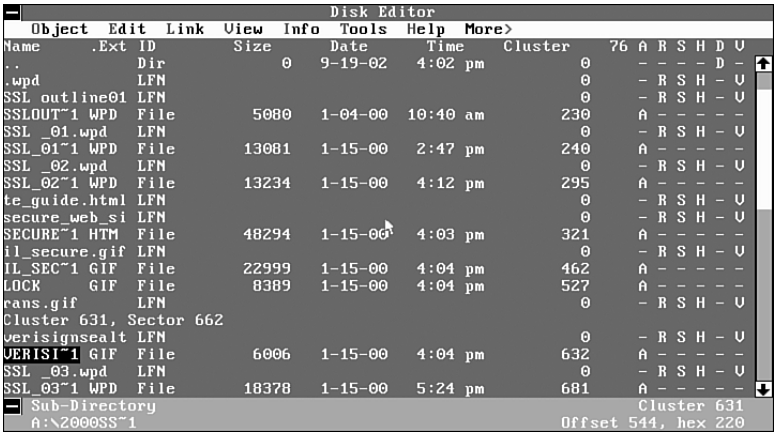


Figure 24.2 The Norton Disk Editor directory view of a typical floppy disk.

The Name column lists the names of the directory entries, and the .EXT column lists the file/folder extensions (if any). The ID column lists the type of directory entry, including

- *Dir.* A directory (folder).
- *File.* A data file.
- *LFN.* A portion of a Windows long filename. Windows stores the start of the LFN before the actual filename. If the LFN is longer than 13 characters, one or more additional directory entries is used to store the rest of the LFN. The next three columns list the file size, date, and time.

The Cluster column indicates the cluster in which the first portion of the file is located. Drives are divided into clusters or allocation units when they are formatted, and a *cluster* (allocation unit) is the smallest unit that can be used to store a file. Cluster sizes vary with the size of the drive and the file system used to format the drive.

The letters *A*, *R*, *S*, *H*, *D*, and *V* refer to attributes for each directory entry. *A* (archive) means the file hasn't been backed up since it was last modified. *R* is used to indicate that the directory entry is read-only, and *S* indicates that the directory entry has the System attribute. *H* indicates that the directory entry has the Hidden attribute, whereas *D* indicates that the entry is a directory. Finally, *V* is the attribute for an LFN entry.

The file `VERISI~1.GIF` (highlighted in black near the bottom of Figure 24.2) is interesting for several reasons. The tilde (~) and number at the end of the filename indicate that the file was created with a 32-bit version of Windows. 32-bit versions of Windows (Windows 9x/Me, 2000, and XP) allow the user to save a file with a long (more than eight characters) filename (plus the three-character file extension such as `.EXE`, `.BMP`, or `.GIF`). In addition, long filenames can have spaces and other characters not allowed by earlier versions of Windows and MS-DOS. The process used by various versions of Windows to create LFN entries is discussed earlier in this chapter in the section called "VFAT and Long Filenames."

When you view the file in Windows Explorer or My Computer, you see the long filename. To see the DOS alias name within the Windows GUI, right-click the file and select Properties from My Computer or Windows Explorer. Or, you can use the `DIR` command in a command-prompt window. The LFN is

stored as one or more separate directory entries just before the DOS alias name. Because the actual long name for VERISI~1.GIF (Verisignsealtrans.gif) is 21 characters, two additional directory entries are required to store the long filename (each directory entry can store up to 13 characters of an LFN), as seen in Figure 24.2.

Determining the Number of Clusters Used by a File

As discussed earlier in this chapter, an area of the disk called the file allocation table stores the starting location of the file and each additional cluster used to store the file. VERISI~1.GIF starts at cluster 632. Clusters are the smallest disk structures used to store files, and they vary in size depending on the file system used to create the disk on which the files are stored and on the size of the drive. In this case, the file is stored on a 1.44MB floppy disk, which has a cluster size of 512 bytes (one sector). The cluster size of the drive is very important to know if you want to retrieve data using Disk Editor.

To determine the cluster size of a drive, you can open a command-prompt window and run CHKDSK C: to display the allocation unit size (cluster size) and other statistics about the specified drive. You can also look up the information in Tables 24.10, 24.11, and 24.14 included earlier in this chapter.

To determine how many clusters are used to store a file, look at the size of the file and compare it to the cluster size of the drive on which it's stored. The file VERISI~1.GIF contains 6,006 bytes. Because this file is stored on a floppy disk that has a cluster size of 512 bytes, the file must occupy several clusters. How many clusters does it occupy? To determine this, divide the file size by the number of clusters and round the result up to the next whole number. The math is shown in Table 24.36.

Table 24.36 Determining the Number of Clusters Used by a File

File Size (FS) of VERISI~1.GIF	Cluster Size (CS)	Result of (FS) Divided by (CS) Equals (CR)	(CR) Rounded Up to Next Whole Number
6,006	512	11.73046875	12

From these calculations, you can see that VERISI~1.GIF uses 12 clusters on the floppy disk; it would use fewer clusters on a FAT16 or FAT32 hard disk (the exact number depends on the file system and size of the hard disk). The more clusters a file contains, the greater the risk is that some of its data area could be overwritten by newer data if the file is deleted. Consequently, if you need to undelete a file that was not sent to the Windows Recycle Bin or was deleted from a removable-media drive or floppy drive (these types of drives don't support the Recycle Bin), the sooner you attempt to undelete the file, the more likely it is that you can retrieve the data.

The normal directory display in Norton Disk Editor shows the starting cluster (632) for VERISI~1.GIF. If a file is stored on a drive with a lot of empty space, the remainder of the clusters will probably immediately follow the first two—a badly fragmented drive might use noncontiguous clusters to store the rest of the file. Because performing data recovery when the clusters are contiguous is much easier, I strongly recommend that you defragment your drives frequently.

To see the remainder of the clusters used by a file, move the cursor to the file, press Alt+L or click the Link menu, and select Cluster Chain (FAT); you can also press Ctrl+T to go directly to this view. The screen changes to show the clusters as listed in the FAT for this file, as shown in Figure 24.3. The clusters used by the file are highlighted in red, and the filename is shown at the bottom of the screen. The symbol <EOF> stands for *end of file*, indicating the last cluster in the file.

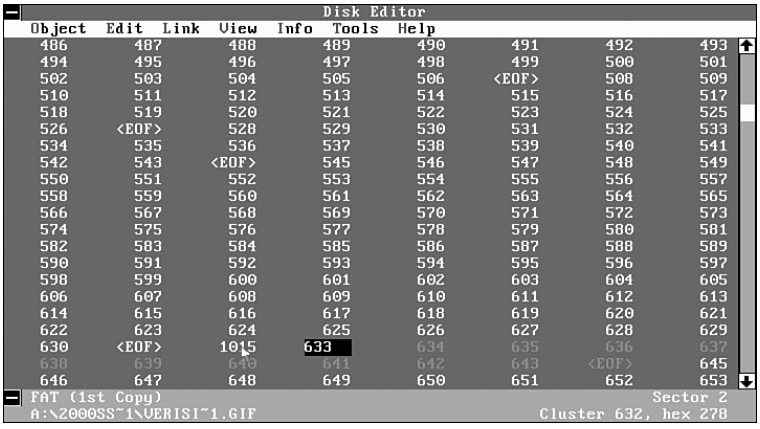


Figure 24.3 The FAT view of `VERISI~1.GIF`. All its clusters are contiguous.

How the Operating System Marks a File When It Is Deleted

If a file (`VERISI~1.GIF`, in this example) is deleted, the following changes happen to the disk where the file is stored, as shown in Figure 24.4:

- The default directory view shows that the first character of the filename (`V`) has been replaced with a σ (lowercase sigma) character.
- There are now two new types of entries in the ID column for this file and its associated LFN:
  - *Erased.* An erased file
  - *Del LFN.* An LFN belonging to an erased file

Note also that the beginning cluster (632) is still shown in the Cluster column.

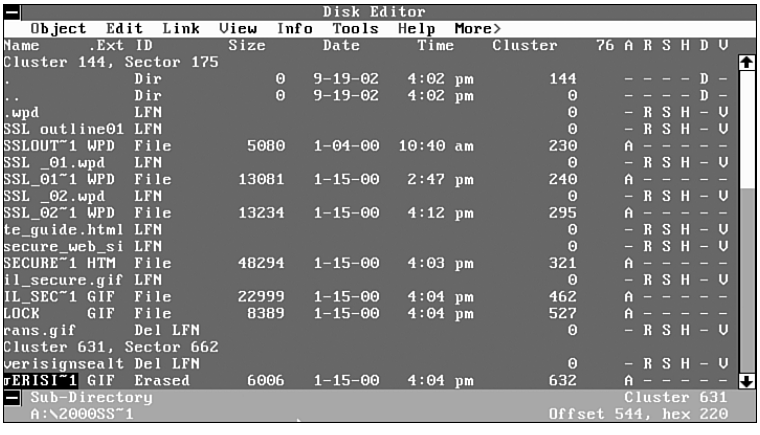


Figure 24.4 The Directory view after `VERISI~1.GIF` has been deleted.

Zeroes have also replaced the entries for the cluster locations after the beginning cluster in the FAT. This indicates to the operating system that these clusters are now available for reuse. Thus, if an undelete process is not started immediately, some or all of the clusters could be overwritten by new data. Because the file in question is a GIF graphics file, the loss of even one cluster will destroy the file.

As you can see from analyzing the file-deletion process, the undelete process involves four steps:

- Restoring the original filename
- Locating the clusters used by the file
- Re-creating the FAT entries for the file
- Relinking the LFN entries for the file to the file

Of these four, the most critical are locating the clusters used by the file and re-creating the FAT entries for the file. However, if the file is a program file, restoring the original name is a must for proper program operation (assuming the program can't be reloaded), and restoring the LFN entries enables a Windows user accustomed to long filenames to more easily use the file.

If you want to make these changes to the original disk, Disk Editor must be configured to work in Read-Write mode.

To change to Read-Write mode, follow these steps:

1. Press Alt+T to open the Tools menu.
2. Press N to open the Configuration dialog box.
3. Press the spacebar to clear the check mark in the Read Only option box.
4. Press the Tab key until the Save box is highlighted.
5. Press Enter to save the changes and return to the main display.

Caution

As a precaution, I recommend that you use DISKCOPY to make an exact sector-by-sector copy of a floppy disk before you perform data recovery on it, and you should work with the copy of the disk, not the original. By working with a copy, you keep the original safe from any problems you might have; plus, you can make another copy if you need to.

After you change to Read-Write mode, Disk Editor stays in this mode and uses Read-Write mode every time you use it. To change back to Read-Only mode, repeat the previously listed steps but check the Read-Only box. If you are using Disk Editor in Read-Write mode, you will see the message `Drive x is Locked` when you scan a drive.

Undeleting an Erased File

After you have configured Disk Editor to work in Read-Write mode, you can use it to undelete a file.

To recover an erased file, follow this procedure:

1. To change to the folder containing the erased file, highlight the folder containing the erased file and press Enter. In this example, you will recover the erased file `VERISI~1.GIF`.
2. Place the cursor under the lowercase sigma symbol and enter the letter you want to use to rename the file.
3. If the keyboard is in Insert mode, the lowercase sigma will move to the right; press the Delete key to delete this symbol.
4. This restores the filename, but even though the ID changes from Erased to File, this does *not* complete the file-retrieval process. You must now find the rest of the clusters used by the file. To the right of the filename, the first cluster used by the file is listed.
5. To go to the next cluster used by the file, press Ctrl+T to open the Cluster Chain command. Because you changed the name of the file, you are prompted to write the changes to the disk before you can continue. Press W or click Write to save the changes and continue.
6. Disk Editor moves to the first cluster used by the deleted file. Instead of cluster numbers, as seen earlier in Figure 24.3, each cluster contains a zero (0). Because this file uses 12 clusters, there should be 12 contiguous clusters that have been zeroed out if the file is unfragmented.

7. To determine whether these are the correct clusters for the file, press Alt+O or click Object to open the Object menu. Type **C** to open the Cluster dialog box (or press Alt+C to go to the Cluster dialog box). Enter the starting cluster number (**632** in this example) and the ending cluster number (**644** in this example). Click OK to display these clusters.

Disk Editor automatically switches to the best view for the specified object, and in this case, the best view is the Hex view (see Figure 24.5). Note that the first entry in cluster 632 is GIF89a (as shown in the right column). Because the deleted file is a GIF file, this is what we expected. Also, a GIF file is a binary graphics file, so the rest of the information in the specified sectors should not be human-readable. Note that the end of the file is indicated by a series of 0s in several disk sectors before another file starts.

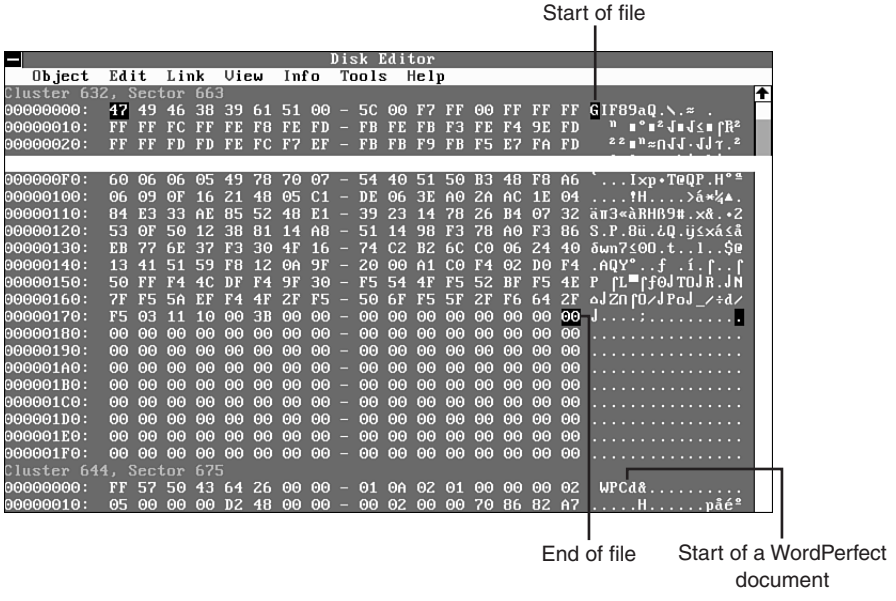


Figure 24.5 The start and end of the file `VERISI~1.GIF`.

Because the area occupied by the empty clusters (632–644) contains binary data starting with GIF89a, you can feel confident that these clusters contain the data you need.

8. To return to the FAT to fill in the cluster numbers for the file, open the Object menu and select Directory. The current directory is selected, so click OK.

9. Move the cursor down to the entry for `VERISI~1.GIF`, open the Link menu, and click Cluster Chain (FAT). The Cluster Chain refers to the clusters after the initial cluster (632); enter **633** in the first empty field, and continue until you enter **643** and place the cursor in the last empty field. This field needs to have the `<EOF>` marker placed in it to indicate the end of the file. Press Alt+E to open the Edit menu and select Mark (or press Ctrl+B). Open the Edit menu again and select Fill. Then, select End of File from the menu and click OK. Refer to Figure 24.3 to see how the FAT looks after these changes have been made.

10. To save the changes to the FAT, open the Edit menu again and select Write. When prompted to save the changes, click Write; then click Rescan the Disk.

11. To return to Directory view, open the Object menu and select Directory. Click OK.
12. The LFN entries directly above the `VERISI~1.GIF` file are still listed as Del LFN. To reconnect them to `VERISI~1.GIF`, select the first one (`verisignsealt`), open the Tools menu (press Alt+T), and select Attach LFN. Click Yes when prompted. Repeat the process for `rans.gif`.
13. To verify that the file has been undeleted successfully, exit Disk Editor and open the file in a compatible program. If you have correctly located the clusters and linked them, the file will open.

As you can see, this is a long process, but it is essentially the same process that a program such as Norton UnErase performs automatically. However, Disk Editor can perform these tasks on all types of disks that use FAT file systems, including those that use non-DOS operating systems; it's a favorite of advanced Linux users.

Retrieving a File from a Hard Disk or Flash Memory Card

What should you do if you need to retrieve an erased file from the hard disk or a flash memory card? It's safer to write the retrieved file to another disk (preferably a floppy disk if the file is small enough) or to a different drive letter on the hard disk. You can also perform this task with Disk Editor.

Tip

If you want to recover data from a hard disk and copy the data to another location, set Disk Editor back to its default Read-Only mode to avoid making any accidental changes to the hard disk. If you use Disk Editor in a multitasking environment such as Windows, it defaults to Read-Only mode.

The process of locating the file is the same as that described earlier:

1. Determine the cluster (allocation unit) size of the drive on which the file is located.
2. Run Disk Editor to view the name of the erased file and determine which clusters contain the file data.

However, you don't need to restore the filename because you will be copying the file to another drive.

The clusters will be copied to another file, so it's helpful to use the Object menu to look at the clusters and ensure that they contain the necessary data. To view the data stored in the cluster range, open the Object menu, select Cluster, and enter the range of clusters that the cluster chain command indicates should contain the data. In some cases, the first cluster of a particular file indicates the file type. For example, a GIF file has `GIF89a` at the start of the file, whereas a WordPerfect document has `WPC` at the start of the file.

Tip

Use Norton Disk Editor to view the starting and ending clusters of various types of files you create before you try to recover those types of files. This is particularly important if you want to recover files from formatted media. You might consider creating a database of the hex characters found at the beginning and ending of the major file types you want to recover.

If you are trying to recover a file that contains text, such as a Microsoft Word or WordPerfect file, you can switch Disk Edit into different view modes. To see text, press F3 to switch to Text view. However, to determine where a file starts or ends, use Hex mode (press F2 to switch to this mode). Figure 24.6 shows the start of a Microsoft Word file in Text format and the end of the file in Hex format.

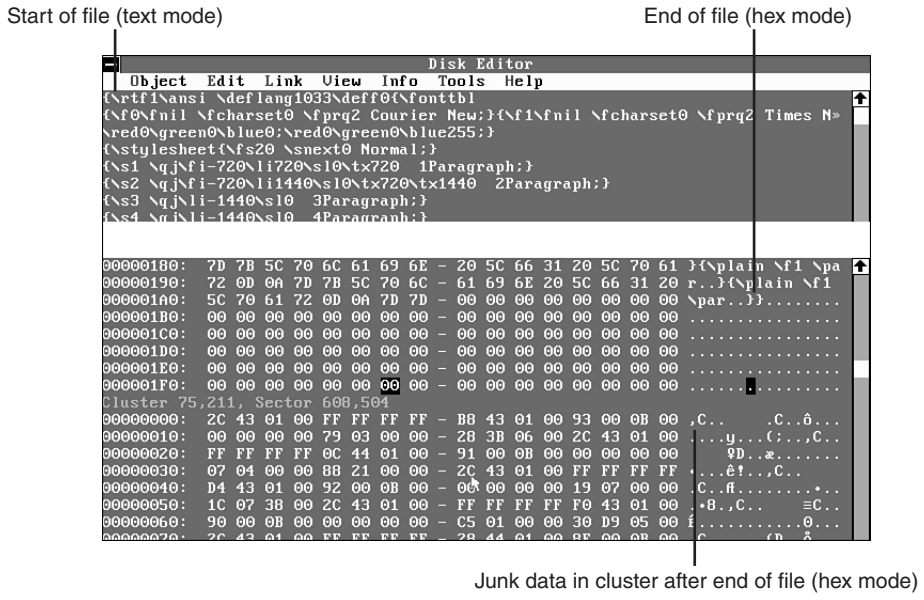


Figure 24.6 Scrolling through an erased file with Disk Editor.

To copy the contents of these clusters to a file safely, you should specify the sectors that contain the file. The top of the Disk Editor display shows the sector number as well as the cluster number. For example, the file shown in Figure 24.6 starts at cluster 75207, which is also sector 608470. The end of the file is located in sector 608503.

To write these sectors to a new file, do the following:

1. Open the Object menu.
2. Select Sector.
3. Specify the starting and ending sectors.
4. Click OK.
5. Scroll through the sectors to verify that they contain the correct data.
6. Open the Tools menu.
7. Click Write Object to.
8. Click To a File.
9. Click the drive on which you want to write the data.
10. Specify a DOS-type filename (8 characters plus a 3-character extension); you can rename the file to a long filename after you exit Disk Edit.
11. Click OK, and then click Yes to write the file. A status bar appears as the sectors are copied to the file.
12. Exit Disk Edit and open the file in a compatible program. If the file contains the correct data, you're finished. If not, you might have specified incorrect sectors or the file might be fragmented.

Norton Disk Editor is a powerful tool you can use to explore drives and retrieve lost data. However, your best data recovery technique is to avoid the need for data recovery. Think before you delete files or format a drive, and make backups of important files. That way, you won't need to recover lost data very often.

Data Recovery from Flash Memory Devices

Flash memory devices such as USB keychain drives and cards used in digital cameras and digital music players present a unique challenge to data recovery programs. Although, from a user standpoint, these devices emulate conventional disk drives, have file allocation tables similar to those found on floppy disks, and can usually be formatted through the Windows Explorer, many data recovery programs that work well with conventional drives cannot be used to recover data from flash memory devices—especially when the device has been formatted.

Under several conditions, data loss can occur with a flash memory device. Some of them, such as formatting of the media or deletion of one or more photos or files, can occur when the device is connected to the computer through a card reader or when the flash memory device is inserted into a digital camera. When photos are deleted, the file locations and name listings in the file allocation tables are changed in the same way as when files are deleted from magnetic media: The first character of the filename is changed to a lowercase sigma (refer to Table 24.19), indicating the file has been erased. Just as with magnetic media, undelete programs that support removable-media drives and the Norton Disk Editor can be used to retrieve deleted files on flash memory devices in the same way that they retrieve deleted files from magnetic media. Note that Disk Editor must be run in read-only mode and works best on systems running Windows 9x/Me. Data files can also be damaged if the flash memory card is removed from a device before the data-writing process is complete.

However, retrieving data from a formatted flash memory device, whether it has been formatted by a digital camera or through Windows, is much more difficult. Traditional unformat programs such as the command-line Norton Unformat program provided with Norton Utilities and Norton SystemWorks can't be used because flash memory devices are accessible only from within the Windows GUI and command-line programs are designed to work with BIOS-compatible devices such as hard and floppy drives.

Programs that rely on the file system, such as Ontrack EasyRecovery Personal Edition Lite (incorporated into VCOM System's Suite) and Ontrack EasyRecovery Personal Edition, do not work either because the previous file system is destroyed when the flash memory devices are formatted.

Note

When a digital camera formats a flash memory card, it usually creates a folder in which photos are stored. Some cameras might also create another folder for storing drivers or other information.

If you need to recover data from a formatted flash memory device, the following programs work extremely well:

- *Ontrack EasyRecovery Professional Edition*; free evaluation and more information are available from `http://www.ontrack.com`
- *PhotoRescue*; free evaluation and more information are available from `http://www.datarescue.com/photorescue/`

Norton Disk Editor (incorporated into Norton SystemWorks, and Norton SystemWorks Pro) can also be used to recover data if you can determine the starting and ending clusters used by the data stored on the device.

To recover data from a formatted flash memory card with EasyRecovery Professional Edition, the RawRecovery option (which recovers data on a sector-by-sector basis) must be used. This option bypasses the file system and can be used on all supported media types. A built-in file viewer enables you to determine whether the recovered data is readable.

PhotoRescue, which works only with standard photo image types such as JPG, BMP, and TIF, can access the media in either logical drive mode (which worked quite well in our tests) or physical drive mode. Physical mode uses a sector-by-sector recovery method somewhat similar to that used by EasyRecovery Professional Edition. PhotoRescue also displays recovered photos in a built-in viewer.

With both products, you might recover data from not just the most recent use before format, but also leftover data from previous uses. As long as the data area used by a particular file hasn't been overwritten, the data can be recovered—even if the device has been formatted more than once.

Table 24.37 provides an overview of our results when trying to recover data from two common types of flash memory devices: a Compact Flash card used in digital cameras and a USB keychain storage device.

Table 24.37 Retrieving Lost Data from Flash Memory Devices

		Results by Data-Recovery Program			
Device	**Cause of Data Loss**	**Norton Utilities**	**Ontrack/Vcom System Suite**	**DataRescue Photo Rescue**	**Ontrack EasyRecovery Professional**
Compact Flash 64MB	Deleted selected files in camera	Recovered data back to device when used with Windows 9x/Me only.[1, 2]	Recovered data to user-specified folder.[1, 3]	Recovered data from most recent format and from previous card uses to specified folder.[3, 4]	RawRecovery recovered deleted files from current and previous uses; refer to Table 24.35 for limitations.[3, 4]
Compact Flash 64MB	Deleted selected files with Windows Explorer	Recovered data back to device when used with Windows 9x/Me only.[1,2]	Recovered data to specified folder when used with any supported version of Windows.[3]	Recovered data from most recent format and from previous card uses to specified folder (files and folders renamed).	DeletedRecovery recovered deleted data from current use (first character of file/folder name lost).
Compact Flash 64MB	Format in camera	Drive could not be unformatted; Disk Edit could retrieve data from current and previous uses to user-specified folder.[3, 5, 6]	Could not locate data. No data was recovered.	Recovered data from most recent format and from previous card uses to user-specified folder.[3, 4]	RawRecovery recovered all readable data, including data from previous card uses to user-specified folder.[3, 4]
Compact Flash 64MB	Format in card reader	Drive could not be unformatted; Disk Edit could retrieve data from current and previous uses.[5, 6]	Could not locate data. No data was recovered.	Recovered data from most recent format and from previous card uses to user-specified folder.[4]	RawRecovery recovered all readable data, including data from previous card uses to user-specified folder.[4]
USB keychain drive (128MB)	Deleted folder with My Computer	Disk Edit can retrieve data from current use.[3, 6]	Partial success: Recovered some files.[3]	We recovered photo files only.[3, 4]	RawRecovery retrieved most files.[3, 4]
USB keychain drive (128MB)	Formatted by Windows Explorer	Disk Edit can retrieve data from current use.[3, 6]	Partial success: Recovered some files (folder names and structure lost).[3]	Recovered photo files only.[3, 4]	RawRecovery retrieved most files.[3, 4]

1. User supplied first letter of filename during undelete process.
2. Norton UnErase doesn't support removable-media drives in Windows NT/2000/XP.
3. Program operates in read-only mode on the drive containing the lost data.
4. Original file and folder names were not retained; files are numbered sequentially and might need to be renamed after recovery.
5. Windows must be used to access flash memory devices, and Norton Unformat can't be used in a multitasking environment such as Windows.
6. Disk Edit requires the user to manually locate the starting and ending sectors of each file and write the sectors to another drive with a user-defined filename.

Common Drive Error Messages and Solutions

Several common error messages are associated with problems in the file system or drives. This section covers the common ones, listing their causes and possible solutions.

Some of the most common file system errors are as follows:

- `Missing Operating System`
- `NO ROM BASIC - SYSTEM HALTED`
- `Boot Error Press F1 to Retry`
- `Invalid Drive Specification`
- `Invalid Media Type`
- `Hard Disk Controller Failure`

These usually occur when booting the system or when trying to log in to or access a drive. The following sections describe each of these errors and offer possible solutions.

Missing Operating System

This error indicates problems in the master boot record or partition table entries. The partition table entries might be pointing to a sector that is not the actual beginning of a partition. This can also be caused by invalid BIOS settings, in some cases resulting from a dead or dying battery. In fact, in my experience, corrupt or improper BIOS settings—such as incorrect geometry (cylinder/head/sector) or LBA mode settings—are the leading cause of this problem. Another cause can be virus damage to the MBR. This error also can occur if no active partition is defined in the partition table.

The normal solution is to correct the invalid BIOS settings. The BIOS settings for drive parameters and LBA translation must be set to the same values as when the drive was partitioned and formatted to read the drive correctly. If the MBR on a FAT drive is damaged or virus infected, you can try `FDISK /MBR` to repair it. Use `FIXMBR` with an NTFS drive. Other types of damage require more sophisticated use of a disk editor utility or repartitioning and reformatting the drive to start over.

NO ROM BASIC - SYSTEM HALTED

This error is generated by the AMI BIOS when the boot sector or master boot record of the boot drive is damaged or missing. This error also can occur if the boot device has been improperly configured or is not configured at all in the BIOS. In this case, data in the partition might be valid and undamaged, but no bootable partition exists.

IBM systems in this situation used to drop into a built-in BIOS version of BASIC, but most non-IBM BIOS manufacturers did not license this code from Microsoft. So, instead of dropping into BASIC, they displayed this cryptic message. Because the most common cause of this type of error is a failure to set at least one partition as active (bootable), the typical solution is to run FDISK and set the primary partition as active. If this is not the problem, the solution is to repair the damaged MBR or correct the improper BIOS settings.

Other possible problems include corrupted or missing drive parameters in the BIOS Setup.

Boot Error Press F1 to Retry

This error is generated by the Phoenix BIOS when the hard disk is missing a master boot record or boot sector or when there is a problem accessing the boot drive. This has the same meaning as `NO ROM BASIC` does on an AMI BIOS. The most common cause of this message is having no partitions defined as active (bootable). See the previous section for more information.

Invalid Drive Specification

This error occurs when you attempt to log in to a drive that has not been partitioned or for which the partition table entry has been damaged or is incorrect. Use FDISK to partition the drive or to check out the existing partitions. If they are damaged, you probably should use a data recovery utility such as the Norton Utilities to correct the problem. Such correction might require manual editing of the

partition table with the Disk Edit program included with the Norton Utilities. Another solution is to repartition the drive from scratch, but this causes any existing data on the drive to be overwritten.

Invalid Media Type

This indicates the partition table is valid, but the volume boot sector, directory, or file allocation tables are corrupt, damaged, or not yet initialized. For example, this is the standard error you would receive if you tried to access a drive that had been partitioned but not yet formatted. The `FORMAT` command is what creates the volume boot record (VBR), file allocation tables, and directories on the disk.

The repair typically involves using a data recovery utility, such as the Norton Utilities Disk Doctor, or redoing the high-level format on the drive. Because high-level formatting does not actually destroy the data, one technique to recover is to high-level format (OS Format) the volume and then immediately unformat it using the unformat utility included with the Norton Utilities.

Hard Disk Controller Failure

This message indicates the hard disk controller has failed, the hard disk controller is not set up properly in the BIOS, or the controller can't communicate with the attached drives (such as with cable problems).

The solution is to check out the drive installation and ensure that the cables to the drive are properly installed, the drive is receiving power, it is spinning, and the BIOS Setup definitions are correct. If all these are correct, the drive, cable, or controller might be physically damaged. Replace them with known-good spares one at a time until the problem is solved.

General File System Troubleshooting for MS-DOS, Windows 9x, and Windows Me

Here are some general procedures to follow for troubleshooting drive access, file system, or boot problems:

1. Start the system using a Windows startup disk, or any bootable MS-DOS disk that contains `FDISK.EXE`, `FORMAT.COM`, `SYS.COM`, and `SCANDISK.EXE` (Windows 95B or later versions preferred).
2. If your system can't boot from the floppy, you might have more serious problems with your hardware. Check the floppy drive and the motherboard for proper installation and configuration. On some systems, the BIOS configuration doesn't list the floppy as a boot device or puts it after the hard disk. Reset the BIOS configuration to make the floppy disk the first boot device if necessary and restart your computer.
3. Run FDISK from the Windows startup disk. Select option 4 (Display partition information).
4. If the partitions are listed, make sure that the bootable partition (usually the primary partition) is defined as active (look for an uppercase *A* in the Status column).
5. If no partitions are listed and you do not want to recover any of the data existing on the drive now, use FDISK to create new partitions, and then use `FORMAT` to format the partitions. This overwrites any previously existing data on the drive.
6. If you want to recover the data on the drive and no partitions are being shown, you must use a data recovery program, such as the Norton Utilities by Symantec or Lost and Found by PowerQuest, to recover the data.
7. If all the partitions appear in `FDISK.EXE` and one is defined as active, run the `SYS` command as follows to restore the system files to the hard disk:

   ```
   SYS C:
   ```

8. For this to work properly, it is important that the disk you boot from be a startup disk from the same operating system (or version of Windows) you have on your hard disk.

9. You should receive the message `System Transferred` if the command works properly. Remove the disk from drive A:, and restart the system. If you still have the same error after you restart your computer, your drive might be improperly configured or damaged.
10. Run `SCANDISK` from the Windows startup disk or an aftermarket data-recovery utility, such as the Norton Utilities, to check for problems with the hard disk.
11. Using `SCANDISK`, perform a surface scan. If `SCANDISK` reports any physically damaged sectors on the hard disk, the drive might need to be replaced.

General File System Troubleshooting for Windows 2000/XP

The process for file system troubleshooting with Windows 2000/XP is similar to that used for Windows 9x. The major difference is the use of the Windows 2000/XP Recovery Console, which is clarified here:

- If the Recovery Console was added to the boot menu, start the system normally, log in as Administrator if prompted, and select the Recovery Console.
- If the Recovery Console was not previously added to the boot menu, start the system using the Windows CD-ROM or the Windows Setup disks. Select Repair from the Welcome to Setup menu, and then press C to start the Recovery Console when prompted.

If your system can't boot from CD-ROM or the floppy, you might have more serious problems with your hardware. Check your drives, BIOS configuration, and motherboard for proper installation and configuration. Set the floppy disk as the first boot device and the CD-ROM as the second boot device and restart the system.

After you start the Recovery Console do the following:

1. Type **`HELP`** for a list of Recovery Console commands and assistance.
2. Run `DISKPART` to examine your disk partitions.
3. If the partitions are listed, make sure that the bootable partition (usually the primary partition) is defined as active.
4. If no partitions are listed and you do not want to recover any of the data existing on the drive now, use FDISK to create new partitions, and then use `FORMAT` to format the partitions. This overwrites any previously existing data on the drive.
5. If you want to recover the data on the drive and no partitions are being shown, you must use a data recovery program, such as the Norton Utilities by Symantec or Lost and Found by PowerQuest, to recover the data.
6. If all the partitions appear in `DISKPART` and one is defined as active, run the `FIXBOOT` command as follows to restore the system files to the hard disk:

   ```
   FIXBOOT
   ```

7. Type **`EXIT`** to restart your system. Remove the disk from drive A: or the Windows 2000 or XP CD-ROM from the CD-ROM drive.
8. If you still have the same error after you restart your computer, your drive might be improperly configured or damaged.
9. Restart the Recovery Console and run `CHKDSK` to check for problems with the hard disk.