Fat32 & File Systems Guide

What is FAT *(File Allocation Table)*:

The FAT is a roadmap, or index, that points to the location where all the information in files is stored on a floppy disk or hard drive. The FAT is extremely important because the system uses it to store and retrieve files containing information.

When you save a file in Windows, it is stored in multiple pieces (in clusters made up of multiple sectors) on the disk. Windows also saves the roadmap, or index, that points to these clusters in two copies of the FAT (File Allocation Table). The FAT contains the directions to all the pieces of your files, so that applications can find them again later.

FAT An Introduction :

The existing File Allocation Table (FAT) file system was invented in 1977 as a way to store data on floppy disks for Microsoft stand-alone Disk Basic. Although originally intended for floppy disks, FAT has since been modified to be a fast, and flexible system for managing data on both removable and fixed media. In 1996 Windows 95 OSR2 came out with FAT32 a new and improved FAT.

A new generation of very large hard disks will soon be shipping, and the existing FAT data structures have finally reached the limit of their ability to support ever larger media. FAT currently can support a single disk volume up to 2 Gigabytes in size. FAT32 is an enhancement of the FAT file system that supports larger hard drives with improved disk space efficiency.

FAT32 is an enhancement of the File Allocation Table file system that supports large drives with improved disk space efficiency. FAT32 is only currently supported by two operating systems:-

OEM Service Release 2 (OSR2) of Microsoft Windows 95 (commonly known as Windows95b). This release of windows was released in the fall of 1996, and is only available with new PC's.

Windows 98, FAT32 is included in the final release of the Windows 98 operating system. It also includes a FAT32 converter so you can convert an existing FAT drive to FAT32 without data loss.

The existing File Allocation Table (FAT) file system was invented in 1977 as a way to store data on floppy disks for Microsoft stand-alone Disk Basic. Although originally intended for floppy disks, FAT has since been modified to be a fast, and flexible system for managing data on both removable and fixed media. The current generation of large hard disks have finally reached the limit of the existing FAT data

structures. FAT currently can support a single disk volume up to 2 Gigabytes in size, with the increasing size of new hard drives this is an increasing problem.

Does my computer come with FAT32?

This generally depends on the computer and its manufacturer. Most manufacturers now set up FAT32 on computers, but some may not. There are two ways to know whether or not your computer is set up for FAT32:

1. Go into the Start menu, select Programs/Accessories/System Tools, and then select Drive Converter (FAT 32). When you run the program, it will display the status of your hard drives. If any of them say ALREADY FAT32, then they are set up for FAT32. If any say FAT or FAT16, then they need to be converted to FAT32. (Only for windows 98 users)

2. Go into My Computer and right-click on a hard drive icon. Select Properties. In the General section, the drive will be listed either as FAT or FAT32.

Most Common FAT (File System's) Explained:

FAT = File Allocation Table

The file system that is used/Or ordinarily designed for floppies and used by DOS, W 3.x, W95, Windows NT and OS/2. In technical terms refered as FAT12 and FAT16 in which 12 and 16 standing for bits. A FAT directory holds info such as name, file size, date & time stamp, the starting cluster number and the file attributes like (archive, hidden, system etc.). It's file system can support up to 65,525 clusters and is limited to 2 GB. Works best on small 500mb drives because of the cluster size. It seems to be about 2% faster than FAT32 and NTFS but windows is faster if confined to a small area. FAT performance drops off after 400mb's on up.

FAT32 = File Allocation Table 32 such as W-95b

FAT32 will not recognize FAT or NTFS volumes of other operating systems--so you can't use them. It supports drives up to 2 terabytes. It uses smaller clusters (e.g. 4k clusters up to 8 gigs).

So--What's the Difference (FAT12/16 or FAT32)

Remember that DOS 6.x or even versions of Windows prior to SR2 won't recognize the front end of a FAT32 partition. So if you must run the occasional old DOS app, move it into a FAT16 drive partition and then restart from an old DOS boot diskette. FAT16 does not support partitions larger than 2GB. FAT32 is an improvement, as it supports drives up to 2 Terabytes in size, and cluster sizes are 4K for partitions smaller than 8GB. So, if you can get FAT32 on the drive, it will work.

Fat12/16 and Fat32 is a Partition size/cluster size issue. FAT32 solves this problem by reducing to 4KB the default file cluster size for partitions between 260MB and 8GB. (Drives or partitions under 260MB use .5KB clusters.) Up to 16GB, FAT32's cluster size is 8KB; to 32GB, it's 16KB; and for partitions of 32GB and greater, the cluster size holds steady at 32KB. FAT32 adds a few other improvements. The root directory on a FAT32 drive is now an ordinary cluster chain, so it can be located anywhere on the drive. This removes FAT16's previous limitation of 512 root directory entries. In addition, the boot record on FAT32 drives has been expanded to allow a backup of critical data structures. This makes FAT32 drives less susceptible to failure. FAT32 partitions are also invisible to other operating systems, including other versions of Windows. To access a FAT32 partition from a boot floppy, you must create an SR2 start-up disk. You won't see your C: drive if you boot from an older Win95 or DOS start-up disk. If you start out with SR2 on a FAT32 partition and subsequently install Windows NT or OS/2, neither OS will be able to access the FAT32 partition. In addition, you can't run disk-compression software (such as Microsoft's DriveSpace) on a FAT32 partition. But it is possible to include both FAT32 and FAT16 partitions on a single hard disk and use DriveSpace compression on FAT16 partitions. (So SR2 includes the same DriveSpace 3 compression Microsoft ships with its Plus pack.)

FAT32/FAT32x

In the beginning, DOS and Windows systems used FAT12/FAT16. But when drives got larger, (meaning over 2 gigb) along came FAT32 and now FAT32x.
FAT16 was limited to 32 MB drives (hi hi hi) and it was updated over the years (by manipulated sector translation) until it became necessary to increase its basic structure from 16 to 32. FAT32 can safely handle drives up to 2 Terabytes--Err, they say--but it has this problem over 8.4 GB.
The ("x") refers to eXtensions to the FAT32 specification, because with the advent of drives exceeding 8.4 Gig, a new limitation was reached. Prior to this, all drives used some form of CHS (Cylinder Head Sector) translation. Under this scheme every sector was given three numbers. In anticipation of this limitation being exceeded, manufacturers developed Logical Block Addressing (LBA). With LBA, each sector is given a unique number depending on the BIOS.

To expand you knowledge base, let's talk about scenarios. Windows 95 and 98 comes with an enhanced 32-bit driver that is LBA compliant (OK) but a certain BIOS limitation can blow out Windows, but run ok in DOS. Why??, DOS does not use the Windows enhanced driver -- that's all. Conclusion, if you try playing with the DOS command FDISK in DOS Mode and then use it from inside a Windows Dos Prompt and funny stuff happens THEN you know to beware of DOS mode, even though there's 4 different versions of FDISK.

FAT32X is a form of FAT32 created by the Windows Fdisk utility when partitions over 8 GB in size are created, and the 1024 cylinder threshold of the disk is passed. The File Allocation Table is moved to the end of the disk in these cases.

As many of users have found, there seems to be a "mystical" limit to how big a hard drive can be used by DOS. At first glance this limit seems arbitrary and can be frustrating, especially with the sudden glut of 8GB+ drives on the market. There is, however, a valid reason behind it.
On most standard IDE drives (SCSI are different but similar rules apply) it is normal to have 16 heads and 63 sectors per track. Cylinders increase as drive size increases. We are seeing drives with as high as 24000 cylinders or as low as 50. To figure the drive's capacity, multiply the cylinder, head, and sector numbers together. Divide the product by 2048. As a formula it looks like this:

(cylinders * heads * sectors) / 2048 = megabytes

This will give you a number in megabytes that are equal to the size of the drive. The system uses these numbers to help it when reading and writing to the disk.

The first limit comes directly from how these numbers relate to the system. The system BIOS's INT13h interface allows for a maximum of 1024 cylinders, 255 heads, and 63 sectors per track. The standard IDE interface allows for a maximum of 65,536 cylinders, 16 heads, and 63 sectors. In order to satisfy the limits of both of these numbers, the minimum highest common number for each is used. This produces a maximum number of 1024 cylinders, 16 heads, and 63 sectors (504 MB per the calculation above).

That limit quickly became too restrictive and a work around method was developed. It is now possible to "translate" a drive by multiplying the number of heads to reduce the number of cylinders. For instance, if I had a drive that was 2046 cylinders, 16 heads, and 63 sectors, I could translate it by halving the number of cylinders and doubling the number of heads. This results in a drive that has 1023 cylinders, 32 heads, and 63 sectors.

By using a translator between the IDE interface and the BIOS INT13h interface, we can accomplish this translation and satisfy both limits. My IDE drive will still have the same physical number of cylinders, heads, and sectors that fit within its limits, but the numbers reported to the BIOS INT13h interface will be translated. This changes our limit to 1024 cylinders, 255 heads, and 63 sectors (8GB per the calculation above) or simply the limits BIOS have in the first place. We are now capable of working with anything up to 8GB.

There is another place that this limit can be found. Every partition table in a PC is set up with the same parameters. An entry for a single partition is 16 bytes long. Of those 16 bytes, three re dedicated to holding the beginning cylinder, head, and sector of a partition and three are dedicated to holding the end cylinder, head, and sector of a partition. A single byte can hold a number up to 255. If this number where strictly adhered to this would leave a limit of 255 cylinders, 255 heads, and 255 sectors. Of the heads byte, this is true. The other two bytes are manipulated slightly to allow for different numbers. The sector number is held in a six digit binary number and the cylinders in a 10 digit binary number. To fit this into the byte structure (remember 8 bits to a byte), the first two digits of the cylinder number are chopped off and put on the

front of the sector number. Learning to interpret the numbers can be interesting, but it effectively gives us a limit of 1023 cylinders, 255 heads, and 63 sectors. Once again 8GB. Now, if these limits look so built in, why are we seeing people breaking them? The answer is logical addressing. All of these limits are related to the number of cylinders, heads, and sectors that can be handled. When you start addressing a drive with logical numbers (start counting at the first sector and keep counting until you get to the end), much larger numbers are possible. For instance, the total number of sectors for a partition is stored in four bytes in the partition table. Four bytes allow partition sizes up to two terabytes. If we were to always use this number, then the 8GB limit would be non-existent.

But the limit persists. This is because all logical addressing is done by operating systems. Windows 95, Windows NT 4.0, and OS/2 Warp have systems that allow them to address the drive in a logical manner. By doing this, all of these operating systems can break the 8GB barrier. DOS and Windows 3.x don't have this ability. They still rely on the traditional cylinder, head, and sector addressing. For this reason any program that runs under these operating systems is also limited to 8GB unless it has its own system for using logical addressing.

Working in FAT32x partitions is essentially the same as working in FAT32 partitions. However, when attempting to manipulate a FAT32x partition, problems may occur. Procedures such as copying, imaging, resizing, and moving FAT32x partitions require different methods than those used for FAT32 partitions.

Many new computers have pre-installed FAT32x partitions. This has created numerous problems for individuals wishing to modify their partitions on their new systems. FAT32x partitions have a different file system flag in the partition table. Sometimes a FAT32x partition is erroneously created entirely within 1024 cylinders. This can be corrected, in some cases, by using a disk editing utility.

VFAT (Virtual File Allocation Table)

A protected-mode version of the FAT file system, used by Windows 95. It is compatible with the FAT system, the main difference being support for long filenames.

NTFS (New Technology File System)

This systems structure is the (MFT) or master file table. It uses too much space to use on a (e.g. 400mb) hard-drive because it keeps multiple copies of files in the MFT to protect against data loss. It also uses clusters to store data in small noncontiguous clusters and isn't broken up resulting in good performance on large hard-drives. It also supports Hot Fixing where bad sectors are automatically detected and marked.

HPFS (High Performance File System)

This system sorts the directory based on names and is better organized, is faster and is a better space saver. It allocates data to sectors instead of clusters, organized into 8mb bands. This banding improves performance because the read/write heads don't have to

return to track zero each time for access. NetWare File System: This is quick because Novell developed it for NetWare servers being NetWare 3.x and 4.x partitions. Linux Ext2: This is also quick because it is a developed version of UNIX. The Linux Ex12 volume supports up to 2 terabytes.

Disk Efficiency

Every file on your system is stored in clusters in your hard drive, the maximum of one file can be stored in a particular cluster, so this results in wastage if the file is under the cluster size. The current FAT version (FAT16) organises files in 32K clusters in drives over 1.2gig, while FAT32 will use a minimum cluster size of 4K. This means that a 3K file wastes only 1K of disk space on FAT32, while it wastes 29K of space on a standard FAT system. This wastage can result in over 50% of a 2gig drive being wasted. See the table below.

Average Cluster Efficiency

Note: Disk Size does not apply to FAT32 where the 4K cluster is usually used.

Cluster Size	Efficiency	Disk Size (applies to FAT 16 only)
2K	98.4%	0-127 MB
4K	96.6%	128-255 MB
8K	92.9%	256-511 MB
16K	85.8%	512-1023 MB
32K	73.8%	1024-2047 MB
64K	56.6%	2047 MB >

What's a cluster and why does cluster size matter?

The whole problem of wasted space arises from the fact that DOS allocates file space in "clusters". Clusters are sequentially numbered on the disk, starting at 0, and cluster numbers are used both in the FAT (file allocation table) and in the individual directory entry for each file.

Allocation by clusters means some space on the disk will be wasted. Regardless of the actual length of a file as reported by the DIR command, the file will actually occupy a whole number of clusters on the disk. So a 1-byte file will actually use a

whole cluster, a file that's 1 cluster plus 1 byte long will use 2 clusters, and so on.

Is this serious? It can be, depending on the pattern of file sizes on your disk. For instance, if you have an 2GB disk with 5,000 files on it, about 100 MB of your disk is being wasted. And the figures can be much worse, depending on the pattern of your usage. One user reported copying 450 MB of files to a 1.6 GB disk and having them take up 600 MB! As your disk approaches being full, you may wish you could squeeze some extra space out of it instead of buying a new disk.

How does cluster size depends on hard-disk size?

As mentioned above in the table the cluster size for various partition sizes so that you can make intelligent choices about how to partition your hard disk.

From the above table we see that even 2.1 GB drive is over the 1023 kilo-byte limit for 16 KB clusters and therefore its cluster size (unpartitioned) is 32 KB. With a 32 KB cluster, even a 1-byte file will use 32 KB of disk space. A file whose length is 32,769 to 65,536 bytes will likewise use two clusters (64 KB), and so on for higher file sizes.

Even so, you may be inclined to think this is no big deal. But think about it: if you have a 2.1 GB drive with 5,000 files, you're probably wasting about 160 MB.

How are cluster sizes determined?

Clusters are always some power of 2 times 512 bytes, but just which power of 2 depends on the disk size. Why should this be so? I mentioned above that clusters are numbered sequentially. The problem is that the directory structure and the FAT have room for only 16 bits for a cluster number. Since the largest unsigned number that will fit into a 16-bit field is 2^16-1 = 65535, the disk can hold at most 2^16 = 65536 clusters.

In general the wasted space per file will be half a cluster. We'll explore the implications of this after we look at cluster sizes for various disk sizes.

Features of Fat 32

FAT32 provides the following enhancements over previous implementations of the FAT tile system:

Supports up to 2 terabytes in size*:* Compared to 2GB of FAT16, 1000 times more.

Uses space more efficiently: FAT32 uses smaller clusters (e.g. 4k clusters for drives up to 8GB in size), resulting in IO to I 5% more efficient use of disk space relative

to large FAT drives. The minimum size for a FAT32 partition is about 260 MB.

Disk Size	Cluster Size	Efficiency
> 260meg	4K	96.6%
> 8gig	8K	92.9%
> 60gig	16K	85.8%
> 2tril	32K	73.8%

More robust: FAT32 has the ability to relocate the root directory and use the backup copy of the FAT instead of the default copy. In addition, the boot record on FAT32 drives has been expanded to include a backup of critical data structures. This means that FAT32 drives are less susceptible to a single point of failure than existing FAT volumes.

More flexible: The root directory of a FAT32 drive is now an ordinary cluster chain, so it can be abnormally large and located anywhere on the drive. In addition, FAT mirroring can be disabled, allowing a copy of the FAT other than the first to be active. These features allow for dynamic re-sizing of FAT32 partitions. Note: while the FAT32 design allows for this compatibility, it is not implemented by Microsoft in the initial release of OSR2.

Technical Implementation

Because of the compatibility considerations described above, the implementation of FAT32 involved very little change to Windows 95. The Major differences between FAT32 and earlier implementations of FAT are as follows:

Two new partition types are defined: OxB and OxC. Both indicate FAT32 volumes; type OxC indicates a FAT32 partition that requires extended INTI3 support (LBA).

The boot record on FAT32 drives requires 2 sectors (due to expansion and addition of fields within the BPB). As a result, the number of reserved sectors on FAT32 drives is higher than on FAT16, typically 32. This expanded reserved area allows two complete copies of the boot record to be stored there, as well as a sector in which free space count and other file system information is stored.

The FAT is now larger, because each entry now takes up 4 bytes and there are typically many more clusters than on FAT16 drives.

The root directory is no longer stored in a fixed location. A pointer to the starting cluster of the root directory is stored in the extended BPB. The on-disk format directory entries is unchanged, except that the two bytes previously reserved for Extended Attributes now contain the high order word of the starting cluster number.

MS-DOS APls that rely on intimate knowledge of the file system layout generally fail on FAT32 drives. For instance, GetDPB (int21 h, function 32h), Int 25/26h

Absolute disk read/write, and most of the Int 21 h, function 440Dh IOCTLs will fail on FAT32 drives. New forms of these APls are provided in OEM service release 2 which work on all FAT drives.

Win32 APls are not affected by FAT32, with the exception of one additional API called GetFreeSpaceEx() for determining the true free space on a FAT32 volume.

Notes

- ***Once you convert your hard drive to the FAT32 format, you cannot return to using the FAT16 format unless you repartition and format the FAT32 drive, or use a conversion utility like Partition Magic, but this cannot be 100% reliable.***

- *If you have a compressed drive, or want to compress your drive in the future, you should not convert to FAT32.* If you are really short of space use Free-space from Mijenix Corporation. It can selectively compress files or folders.

- If you have a removable disk that you use with another operating system, don't convert to FAT32.

- Hibernate features (suspend to disk, for example) will not work on a FAT32 drive.

- Although most programs are not affected by the conversion from FAT16 to FAT32, some disk utilities that depend on FAT16 do not work with FAT32 drives. Contact your disk utility manufacturer to see if there is an updated version that is compatible with FAT32.

- If you convert your hard drive to FAT32, you can no longer use dual boot to run earlier versions of Windows (Windows 95 [Version 4.00.950], Windows NT 3.x, Windows NT 4.0, and Windows 3.x). However, if you are on a network, earlier versions of Windows can still gain access to your FAT32 hard drive through the network.

- The minimum size for a FAT32 partition is about 260 MB. However, if you use the Windows98 FAT32 converter, it requires drives to be at least 512 MB in size before they can be converted. This is done to gain maximum performance.

Compatibility Considerations

In order to maintain the greatest compatibility possible with existing applications, networks and device drivers, FAT32 was implemented with as little change as possible to Windows 95's existing architecture, internal data structures, Application Programming interfaces (API's) and on-disk format. However, because 4 bytes are now required to store cluster values, many internal add on-disk data structures and published API’s will fail on FAT32 drives. Most applications will be unaffected by

these changes. Existing utilities and drivers should continue to work on FAT32 drives. however, MS-DOS block device drivers (e.g. ASPIDISK.SYS) and disk utilities for these will need to be revised to support FAT32 drives.

All of Microsoft's bundled disk utilities in Windows95b (format, FDISK, Defrag, MS-DOS and Windows Scandisk, and DriveSpace) have been revised to work with FAT32. In addition, Microsoft is working with leading device driver and disk utility vendors to support them in revising their products to support FAT32. Norton Utilities version 2 for Windows 95 fully supports FAT32.

Performance*(discussed in detail in later parts)*

For most users with small cluster size **(4kb and below)**, FAT32 will have a negligible performance impact. Some applications may see a slight performance gain from FAT32. In other applications, particularly those heavily dependent on large sequential write operations, FAT32 may result in a modest performance degradation. The overall effect on raw disk performance is less than 2% however, and the overall impact on application performance as measured by Winstone is typically less than 1%. Drive defragmenting becomes a lengthy process due to the large number of clusters.

Dual-Boot Computers

At this time, Windows 95 OEM Service Release 2 and Windows 98 are the only operating systems capable of accessing FAT32 volumes. Windows 3.1, MS-DOS and the original version of Windows 95 will not recognise FAT32 partitions, and thus they are unable to boot from a FAT32 volume. Microsoft plans to add support for FAT32 in Windows NT5, but at this time , Windows NT is unable to access, or dual boot from FAT32 volumes. At minimum, Microsoft will provide a utility to convert a FAT32 volume to an NTFS volume in NT5.

Customers who run Windows 95 real mode (for example, to run a game) will be able to use FAT32 volumes. Windows95/98 protected DOS mode does include the FAT32 driver.

Creating FAT32 Drives *(discussed in detail in later parts)*

There are only currently three ways of creating a FAT32 partition:-

1. In OEM Service release 2 or in Windows 98, if you run the FDISK utility on a large system with a drive over 512MB, it will ask whether to enable large disk support. If you answer yes, any partition you create that's greater than 5I2MB will be marked as a FAT32 partition. FDISK cannot convert a FAT16 partition to FAT32, all contents of the drive are lost if you use this method.

2. Partition Magic 3 or ‘CVT.EXE’ from Microsoft can convert partitions to FAT32 on-the-fly without data loss, this is probably the best method currently available, it

can also convert back to FAT16 if required.

3. Windows 98 includes a graphical FAT32 conversion utility, which quickly and safely converts a hard drive from the original FAT to FAT32. You can also start FAT32 Converter by clicking Start, Programs, Accessories, System Tools, and then clicking FAT32 Converter.

Does FAT32 affect system performance ?

!! NO !!

To quote the Microsoft Knowledge Base article referenced below:

"For most users, FAT32 will have a negligible performance impact. Some programs may see a slight performance gain from FAT32. In other programs, particularly those heavily dependent on large sequential read or write operations, FAT32 may result in a modest performance degradation." **I do not agree with this statement !**

Basically, the more clusters on your hard drive, the slower the performance. This is true for any file system, FAT16, FAT32, NTFS, HPFS, whatever. However, since FAT32 allows for many more clusters on a single partition than FAT16, the effect may be noticable. Disk utilities are especially affected (slower) the more clusters in the partition. For instance, the closer to 8GB your partition gets, the more 4K clusters, and the slower the performance. (However, as your partition gets bigger, your slack vs. FAT16 improves.)

In preliminary benchmark testing, FAT32 and FAT16 benchmark roughly the same (within 2% either way) when partition size and cluster size are the same. (Note that in order to create a FAT32 partition with the same cluster size as FAT16, the /Z switch must be used when formatting.) However, when cluster sizes are made smaller and the number of clusters increases (only possible with FAT32), disk performance degrades.

Thus there is a battle between slack and performance: Small clusters mean less slack but worse performance. Large clusters mean more slack but better performance. Most users will notice slack differences much more than performance differences caused by varying cluster sizes. Microsoft has decided for us (in making 4K the default cluster size for FAT32) that 4K clusters is the best balance between slack and performance. However, with the /Z switch on the FORMAT command, the user has the ability to decide for his/herself what cluster sizes should be, based on the user's concerns about slack vs. performance.

Read the table below for choosing the cluster size on basis of disk size for better performance and limiting slack space.

Best recommended Cluster size for your drive

Disk Size	Cluster Size Recommended	Format Command Option*
Less than 1 GB	4 KB	Format x:
Less than 4 GB	8 KB	Format x: /z:16
Between 4 to 16 GB	16 KB **	Format x: /z:32
Above 16 GB	32 KB	Format x: /z:64

*x: is the drive i.e. e.g. for c drive it is format c:

** 16 KB Cluster Size is best for space savings and performance, even faster than FAT16. My drive runs on FAT32 with cluster size of 16 KB and Sysinfo by Norton Utilities 3.0 gives me cached read/write benchmark of 57 MB and 5 MB physical read/write speed compared to 57 MB and 5.1 MB on FAT16 (with 2GB partitions and cluster size of 32 KB) respectively on Seagate Medallist 4.3 GB.

TIPS :-

- The APM (Advance Power Management) "feature" of OSR2, and which spins down the hard disk when inactive, also results in slower performance. To turn off APM, go to Control Panel > Power and uncheck the "Allow Windows to manage power use on this computer" box.
- Also increase buffers in your config.sys in root of your boot drive to say 60 or above. If file is not there create it in C:\ (assuming C is your boot drive) with following lines :

  BUFFERS=65,8

- New in OSR2 is support for Hard Drive/CDROM Direct Memory Access. With DMA enabled, no processor time is used to access the hard drive or CDROM. The result is an improvement in overall system performance. However, the default for DMA is disabled (for the default OSR2 Bus Mastering IDE controller drivers)! Note that varying hardware configurations will show varying improvements in system performance when DMA is enabled. To enable HD DMA, go to Control Panel > System > Device Manager > Disk Drives. Highlight your hard drive, click properties. Click the "settings" tab. Check the box next to "DMA." Restart the system when prompted, and repeat for any additional hard or CDROM drives.
- NOTE that the DMA checkbox only appears 1) for IDE drives, and 2) only if the default OSR2 bus mastering IDE controller drivers are installed and configured correctly. Some IDE drives do not support Bus Mastering. In effect, OSR2 supports Bus Mastering, but only by checking the DMA box, is Bus Mastering turned "on."

- Note that the DMA check box only appears when using the default Windows95 OSR2 Bus Master drivers. If you are using non-bus mastering drivers, the DMA box won't show up because Bus Mastering cannot be enabled. If you are using other Bus Mastering drivers, the DMA box does not show up because it is enabled by default, and you don't have the option of disabling Bus Mastering.

- It has previously been written that FAT32 is not faster than FAT16 in the OSR 2 beta release notes, early Windows 98 beta release notes, and Introduction to Windows 98 from Microsoft Press. This is no longer true. In fact, the largest application-launch performance gains are associated with FAT32.

- FAT32 uses the disk cache (Vcache) more efficiently by allowing the memory manager to map directly into memory portions of an application that is in the disk cache. This process eliminates an additional memory copy of the mapped portions of the application. This feature is only available only on FAT32 volumes.

- FAT32 allows the memory manager to write pageable data very efficiently to the swap file while a system is idle. This feature is only available on FAT32 volumes and volumes with cluster sizes greater than 4K.

Creating FAT32 Drives

As discussed earlier there are two ways of creating a FAT32 partition:-

1. Using FDISK :- In OEM Service release 2 or in Windows 98, if you run the FDISK system utility it will ask whether to enable large disk support. If you answer yes, any partition you create that's greater than 5I2MB will be marked as a FAT32 partition. FDISK cannot convert a FAT16 partition to FAT32, all contents of the drive are lost if you use this method.

2. Using Converter :- To convert an existing partition without any data loss you can use :

- Partition Magic
- 'CVT.EXE' from Microsoft
- Windows 98 graphical FAT32 conversion utility

to convert partitions to FAT32 on-the-fly without data loss, this is probably the best method currently available, it can also convert back to FAT16 if required.

FAT32 conversion using FDISK

Step-by-step procedure

1. Do a **full** backup of your disk. If you're at all uncertain about your backup

hardware and software, do a second full backup. Don't give in to temptation to skip this step, because **when you run FDISK you're going to make all the old files on your hard disk disappear i.e. FDISK deletes everything and creates everything fresh and new.** ***HI HI HI*** **BEWARE BACKUP BEFORE STARTING.**

2. Make sure you have a **bootable floppy / Startup Disk**. Also make sure you have the following on a floppy :
   - *your* backup/restore software
   - FDISK
   - FORMAT
   - mouse driver and *CD-ROM driver*
   - You can create one bootable floppy by two ways :

   - If you already have Windows 95 OSR2 or Windows 98 installed go to Control Panel – Add Remove Programs – Startup Disk – CREATE DISK.
   - If you do not have windows 95 / 98 installed run Setup and follow instructions till you reach Creating Startup Disk. Create Startup Disk and then cancel the setup.

3. Decide what partition sizes you want.

You can partition your Hard Disk in two or more parts or in a Single entity as a whole. I recommend that go for a single big size because of speed and it is fast to defrag a single partition rather than multiple partitions.

In Microsoft world, you don't create N partitions on a disk. Instead you create one "primary DOS partition" and one "extended DOS partition" and then you create N-1 "logical drives" on the extended DOS partition. So if you're trying to partition your only hard disk into C:, D;, and E:, you'll create a primary DOS partition which will automatically be C:, then an extended DOS partition, then logical drives D: and E: on the extended partition.

4. Insert your boot disk and restart your computer. When the command prompt appears, type *FDISK*. At that point, FDISK will ask you if you want to enable large disk support.

FDISK asks if you want to enable large disk support.

- **Enter *Y* to enable the FAT-32 file system**. FDISK will then ask you what you want to do with the partition. Notice that the first line under FDISK Options tells you which fixed disk drive you're currently working on. If you're using a computer with multiple hard disks, enter *4* at the prompt to see the drive letter that corresponds to the drive. If this isn't the hard disk you intend to reconfigure as FAT-32, type *5* to change the current fixed disk drive.

- **Use FDISK to delete and re-create your DOS partition.**

- Once you've switched to the correct hard disk (if necessary), enter *3* at the command prompt. FDISK will now display the Delete DOS Partition or Logical DOS Drive screen. I'll assume that you'll be working with the primary partition. (If you have multiple partitions on the same physical drive, you may have to delete an extended partition before you can delete the primary partition.) Enter *1* to delete the primary partition.

- At this point, FDISK will warn you that it's about to delete all data in the partition. It will also ask you for the partition number you want to delete. For our example, enter *1*. FDISK will then ask you to enter the volume label of the partition you're deleting. This is an extra safeguard to make sure you're really deleting the partition that you think you are. Answer *Y* to the final confirmation prompt to allow FDISK to delete your partition.

- **FDISK warns you that you'll lose all data when you delete the partition. Press Y to carry on or N to do latter.**

- After FDISK deletes your partition, it takes you back to the main menu. Now you need to create a new primary DOS partition. First, enter *1* at the prompt. When FDISK asks if you'd like to use the maximum available size for a primary partition and make the partition active, enter *Y*. You'll soon see a message stating that FDISK created the partition. We're now done with our boot disk, so press [Esc] to exit FDISK.

- Next, reboot your PC using your boot disk. When you see the DOS prompt, type *FORMAT C: /S* /z:n (n is the cluster size you have to decide from the above table) and press [Enter] to make the partition bootable. You now have a new partition that uses FAT-32 and that you can install Windows 95 on.

- If you do not reboot between FDISKing and FORMATing, you will get strange-looking error messages.

- **TIP**: There is an undocumented flag for OSR2's FDISK, namely, /FPRMT. If you do a FDISK /FPRMT, you are able to format partitions smaller than 512MB as FAT32. For advanced users only!

- **TIP**: There is an undocumented switch in the FORMAT command in OSR2. Then syntax is:

- **FORMAT /z:n where n * 512bytes = cluster size** In other words, with this switch, you can set your own cluster size. Like mentioned above you can test this accordingly

Restore the files you backed up. (You don't need to restore them all to C. But unless you plan on reinstalling Windows from scratch, it's probably best to restore the Windows files to the same disk letter where they were before you repartitioned.)

Your CD-ROM drive letter will now be different. You'll have to change the drive letter in any Windows icons that point to it (click on each icon, then click File, click

Properties, then edit the command line and possibly the working directory, then click).

You don't need to make any changes in your CMOS settings. Even though DOS and Windows now behave as if you have several hard disks, the CMOS will still correctly show the actual hard disk(s). This completes the procedure.

FAT16 to FAT32 conversion by utility

PowerQuest has released one software part of Partition Magic in which it is possible to "force" an upgrade of an existing Windows95 installation, and then convert FAT16 to FAT32 after installation of OSR2.

On a 1.2GB drive, formatted as a single partition, before conversion from FAT16 to FAT32 (via Partition Magic 3.0), there was 58MB free space. After conversion, there was 268MB free space. Your results will vary.

Microsoft has also developed a FAT16 ---> FAT32 (one way only) conversion utility, named "CVT.EXE". This conversion utility has only been released to beta testers of OSR2 and Memphis. It has not been distributed via MSDN. CVT.EXE is currently available for download at http://www.memphis97.com/97download.html. Or search at www.ftpsearch.lycos.com.

FAT32 Conversion in windows 98

Windows 98 includes a converter that will convert a FAT16 volume to FAT32. The conversion process leverages features of the FAT32 file system specifically designed to allow for safe conversions from a FAT16 volume. These features include the ability to specify which of multiple FAT tables is active and the ability to relocate the root directory table.

The basic algorithm of the FAT32 converter is to construct a mirror file system that shares the same data space as the original file system. The FAT16 file system remains fully intact and usable while the new FAT32 file system is constructed. At no point during the building of the FAT32 file system would a power failure or unexpected reboot result in data loss. Once the FAT32 file system is constructed, it is "instantly" enabled with a single-sector write to the boot sector. That is, the transition from a fully functional FAT16 file system to a fully functional FAT32 file system comes down to a single sector write.

The nine phases of the conversion process are the following:

Phase 1: Ensures that it is safe to convert:

Runs ScanDisk to verify that the drive's file system structure is valid. Refuses to run if there are any clusters that have previously been marked as being bad. Watches for compression. Watches for incompatible BIOS versions. Watches for anti-virus software that will get in the way. Watches for drives that are not supported by interrupt 13 (Int 13). Watches for drivers that do not support FAT32.

Phase 2: Converts the directories. CVT (the conversion utility) makes a copy of the entire volume's directory structure to an unused portion of the disk. The conversion is performed entirely on this copy. Any failure or power loss during this phase leaves the volume in its original state.

Phase 3: Makes room for the 32-bit FAT. CVT makes room for the new 32-bit FAT by moving files and directories out of the way using the same method as Defrag. Any failure or power loss during this phase might leave minor problems in the form of lost clusters (the same as Defrag.) No user data is lost, and ScanDisk can easily throw away the lost clusters.

Phase 4: Converts the 16-bit FAT to a 32-bit FAT. CVT converts as it copies the 16-bit FAT to the new unused location for the 32-bit FAT. Any failure or power loss during this phase leaves the volume in its original state, except that a few files have been moved.

Phase 5: Frees unneeded clusters. CVT frees clusters marked as used in the 32-bit FAT that will no longer be needed because of the smaller FAT32 clusters. Any failure or power loss during this phase leaves the volume in its original state, except that a few files have been moved.

Phase 6: Updates the master boot record. CVT updates the master boot record with a new signature indicating that the volume is either a FAT32 volume or a FAT16 volume. Any failure or power loss during this phase leaves the volume as a FAT16 volume with all files intact.

Phase 7: Updates the boot record. CVT updates the boot record to the FAT32 boot record. Any failure or power loss during this phase leaves the volume as a FAT16 volume with all files intact.

Phase 8: Creates a second copy of the FAT. CVT creates a second copy of the 32-bit FAT and then enables it. Any failure or power loss during this phase leaves the volume as a FAT32 volume with all files intact. The volume will only have one FAT, but it will function properly.

Phase 9: Moves the root directory. CVT relocates the root directory to the front of the disk. Any failure or power loss during this phase leaves the volume as a FAT32 volume with all files intact, but it might leave minor problems in the form of lost clusters (the same as Defrag). No user data is lost, and ScanDisk can easily throw away the lost clusters.

Can I compress a FAT-32 drive:-

NO Windows 95 won't allow you to compress a FAT-32 drive. In fact, Microsoft has changed the manner in which you compress disks in OSR2. If you try to use the Compression Agent with OSR2, Windows 95 will display the error message It states that if you want to compress a disk, you must first install Microsoft Plus!. However, this message is deceptive, because OSR2 includes DriveSpace 3, which is the same compression utility that's found in Plus!. You can access DriveSpace 3 by choosing Accessories, System Tools, Drive Space. Unfortunately, DriveSpace 3 still can't compress a FAT-32 partition. Instead use Freespace from Mizenix Corporation instead of DriveSpace.

Will my disk utilities work with FAT-32?

Norton Utilities 2.0 and Norton AntiVirus 2.0 offer full FAT-32 support if you install the patch available at Symantec's Web site (http://www.symantec.com). Norton Utilities 1.0 is incompatible with FAT-32.

A little about Fdisk Command

FDISK is the utility provided with all versions of MS-DOS and Windows95/98 for partitioning hard drives. Partitioning a hard disk defines areas of the disk to be used by operating systems as volumes. In DOS, each volume is assigned a drive letter. A disk must be partitioned with FDISK before it can be formatted, even if the disk is to be left as one volume.

FDISK Version History:

- MS-DOS 2.x - First DOS version to support hard drives. Maximum partition size - 16MB.
- MS-DOS 3.2 - Maximum partition size - 32MB No extended partitions supported.
- MS-DOS 3.3 - Maximum partition size - 32MB First DOS version to allow extended DOS partitions. Maximum number of partitions - 24 (C: through Z:)
- MS-DOS 4.x - Maximum partition size - 2.1GB Maximum number of partitions - 24
- MS-DOS 5.x - Same as DOS 4, but now handles up to 8 physical drives
- **MS-DOS 6.x - Same as DOS 5**
- **Windows95 - (MS-DOS 7) Basically the same as DOS 4/5/6 but adds 2 new partition types - 0E and 0F - which will be seen as NON-DOS partitions by earlier DOS versions. Type 0E is used for a Primary partition if INT13 Extension support is present in the BIOS. Type 0F is used for an Extended partition if INT13 Extension support is present.**
- **WIN95B(OSR2) Adds support for FAT32. Adds 2 more new partition types 0B and 0C. 0B is used for a FAT32 partition. 0C is used for a FAT32 partition if INT13 Extension support is present. Type B and C partitions are seen as NON-DOS partitions by earlier DOS versions (including WIN95/WIN95A).**

Some Rules of FDISK:

Under DOS, the first physical drive must be the boot drive. The first physical drive

must contain a primary DOS partition and the primary partition must be the first partition on the drive. A drive can have only one primary DOS partition. The partition must be active in order to boot. Only partitions on drive 1 can be made active. Only drive 1 has to have a primary partition. Additional drives may be defined either as primary or as extended partitions. In addition to or instead of a primary partition, a drive may also have one extended partition. All volumes in extended partitions must further be defined in FDISK as logical DOS volumes. DOS through FDISK assigns drive letters first to all primary partitions in order, starting with the letter C, and then to all logical volumes in extended partitions. Drive letters A and B are reserved for floppy drives. Existing partitions must be deleted with FDISK before new partition scan be defined. FDISK from MS-DOS 4.01 and earlier will not remove NON-DOS partitions.

Some Considerations for Large Drives DOS 6.22 FDISK Does not support drives over 8.4GB. Will show total drive size for drives over 8.4GB as 7553MB or 8025MB or some similar value which will vary depending on how the drive is being handled by the BIOS.Also cannot correctly display the size of large drives - is limited to 4 characters (9999MB).

WIN95/WIN95A FDISK

Does support drives over 8.4GB, but is still limited. For drives over 8.4GB, the BIOS must support INT13 Extensions. If FDISK is started with /X option, it will be limited to 8.4GB total size and 0E or 0F partition types will not be used. Also still has same problem with displaying size of large drives.

FDISK will see only 8.4GB of a drive larger than 8.4GB if FDISK is used in a DOS window or through the Run box on the Start menu. To correctly set up a drive larger than 8.4GB, FDISK must be run in DOS mode. This can be accomplished by booting to "Command Prompt Only" from the Startup menu, or choosing "Restart in MS-DOS Mode" from the Shutdown menu, or just booting from a WIN95 startup disk.

WIN95B (OSR2) FDISK

Adds support for FAT 32, which allows single partitions up to 2 terabytes (2000 gigabytes). Will always first ask "Do you wish to enable large disk support?" What this means is "Do you wish to use FAT32?" Answering NO to this question still allows support for large drives. The /X option limits FDISK to 8.4GB total size, even if answering YES to "Do you wish to enable large disk support?". FAT32 can be used on any drive over 512MB.

The problems with displaying the size correctly are fixed in the WIN95B version of FDISK. However, the FORMAT command, which also does not display large sizes correctly, is still not fixed.

The problem with FDISK not seeing drive capacity beyond 8.4GB in a DOS

window is fixed. **WIN98 FDISK** Problem with size shown by FORMAT command while formatting drive is still not fixed.

Explanation of * Remote * in Volume Label field in FDISK

In some cases, if an existing logical drive is not recognized, FDISK will show * Remote * in the Volume Label field for that drive. This can occur if a logical drive is not recognized by FDISK for some reason and if another drive such as CD-ROM drive or RAMDRIVE or network drive is using the drive letter that the logical drive would have been assigned. One reason that a logical drive might not be recognized is that the partition type may not be valid for that version of FDISK. We have seen a case where a logical drive showed up normally in WIN98 GUI mode or in FDISK in a DOS box, but the drive label field showed * Remote * in FDISK in DOS mode (with a RAMDRIVE loaded). In this case, when booting to Safe mode/Command prompt only, FDISK showed the partition but without the volume label (which dids how up in a DOS box) and the drive was not accessible. In this case the user had copied WIN95 from an old drive to a new drive installed as master, then installed WIN98 on top of WIN95, then converted the partition on the main drive to FAT32. The * Remote * string in the Volume label field is not meant as a volume label but is meant to indicate that FDISK thinks this drive maybe a network drive.

FDISK SWITCHES

/MBR - Recreate Master Boot Record on disk 1

This function is handy when an virus has infected the Master Boot Record. With /MBR you can wipe-out the virus.

FDISK does not build an MBR on any drive except the primary master. FDISK will only create an MBR on the primary master drive if the drive does not already have a valid MBR. The 55AAh signature at the end of the sector is checked by FDISK, if not present the MBR is written. Also if the drive is blank, an MBR is written.

Run Like this at command prompt in dos mode

Fdisk /mbr

FDISK /STATUS

displays partition information without starting FDISK and navigating thorough the menus. This works with MS-DOS version 5.00and higher. The first sector on the hard drive, (cylinder 0, head 0, sector 1) contains the master partition boot record. This 512-byte sector contains the partition loader and the partition table. At bootup, the BIOS loads the partition loader, the partition loader loads the bootstrap loader for the bootable partition and the bootstrap loader loads the operating system.

/PRI

Create primary partition. Partition is set to active Create a primary partition on disk number <disk> with the size of <size>. The partition is set to active. FDISK /PRI:<size> <disk>

If <size> is larger than the space on the HD all space is used for the primary partition.

/PRIO

Create primary partition with FAT16/FAT32 override. Partition is set to active

Works as /PRI.

/EXT

Create extended partition

Create an extended partition (to hold logical drives) on disk number <disk> with the size of <size>.

FDISK /EXT:<size> <disk>

If <size> is larger than remaining free space, all free space is used. That is, you don't have to know the exact remaining size in order to use this switch.

/LOG

Create logical drive

With /LOG you create a logical drive with the size of <size>. /LOG must be used together with /EXT.

FDISK /EXT:<size> <disk> /LOG:<size>

/LOG must be used together with /EXT and <size> must be the same for both switches. Furthermore, <size> must be smaller or equal to free space.

/LOGO

Create logical drive with FAT16/FAT32 override

Works as /LOG.

/FPRMT

Prompt for FAT32/FAT16 in interactive mode

With /FPRMT you won't get the FDISK startscreen where you are asked for support for large disks. Instead, you will be prompted for FAT16/FAT32 each time you create a partition.

FDISK /FPRMT

/X

Do not use LBA partitions

With /X you won't get any LBA partitions.

FDISK /X

/CMBR

Recreate Master Boot Record on specified disk

Works as /MBR with the exception that you specify the disk to have it's MBR recreated.

FDISK /CMBR <disk>

Notes on /PRI, /PRIO and /LOG, /LOGO

As far as I can tell, PRI and LOG creates FAT32 when partitions are larger than 512Mb and FAT16 when partitions are smaller than 512Mb. PRIO and LOGO creates FAT16 even if partitions are larger than 512Mb (in effect, it’s like FDISK from DOS 5/6).

A little about Format Command

First of all, (you don't have to FDISK/Partition your H-D first, because you might just want to Restore (Clean-Up) Windows or just want a extra storage area, so just FORMAT/Re-FORMAT it and it's done.

Formatting is the process of writing marks on the Hard Drive or (removable) Magnetic media that are used to mark tracks and sectors. Before a disk is formatted, its magnetic surface is a complete mess of magnetic signals. When it is formatted,

some order is brought into the chaos by essentially drawing lines where the tracks go, and where they are divided into sectors.

The actual details are not quite exactly like this, but that is irrelevant. What is important is that a disk cannot be used unless it has been formatted. The terminology is a bit confusing here: in MS-DOS, the word formatting is used to cover also the process of creating a file system. There, the two processes are often combined, especially for floppies. Where the distinction needs to be made, the real world Unix techies call formatting (low-level formatting, while making the file system is called high-level formatting) and we combine the two and call it formatting.

Ok, you've learned that your platter (disk) is a mass storage device generally made of metal, covered with a thin layer of iron oxide of which has good magnetic properties. Computers record data in bits, 8 bits to a byte and 512 bytes to a sector. The Windows OS knows where all your stuff is at upon request. Even the smallest hard drive can store millions of bits and must be organized---called (formatting).

First, a hard drive has to be physically formatted before it can be logically formatted. This (low-level) formatting is done by the drive manufacturer (IDE) and divides the platter into tracks, sectors and cylinders----these are called physical elements. Tracks are circular paths around your disk and are identified by a number starting with (0) at the outer edge. The set of tracks that lie at the same distance from the center on all sides of all platters are called a cylinder. Tracks are divided into areas called sectors.

Sectors are usually formatted to contain 4096 bits or 512 bytes. After this physical format--it's ready for logical formatting.

Logical formatting places a file system on your disk and a file system allows an operating system like (Windows-95) to use this space to store and retrieve files. So, this is what you do is a logical formatting using it's operating utility.

This means you can format a Hard-Drive and use it right away for a storage container just like you've been doing on the floppies since you know when, then use it as a back-up data area or what ever. A disk can be divided into partitions and then formatting is applied.

After you partition and format---it's called a (volume). This is why you should give it a name or (label) so you can identify it.

So, you now (hopefully) understand why you have to format.

Format Switches

FORMAT = places a file system on the disk for storage or a operating system.

***FORMAT /c* - Causes FORMAT to retest bad clusters, otherwise FORMAT will mark the clusters as bad but will not retest them.**

FORMAT /s = prepares a partition or disk to make it active or bootable.

FORMAT /mbr = creates a new boot sector and should only be used as a last resort.

FORMAT /q =this is mostly unknown but seems to work if your having problems getting the W95 setup to work (not recommended unless it's a last ditch effort).

FORMAT /U = Does a unconditional format, so do the SYS C: to get system files -- then a through scandisk to fix any errors. This FORMAT /U parameter performs an UNCONDITIONAL format, which DESTROYS every byte of data on a disk by overwriting it with.

***WARNING: You CANNOT UNFORMAT a disk formatted using the /U option*!**

FORMAT /SELECT /U This particular combination of FORMAT.COM parameters makes a disk UNREADABLE! WARNING: DO NOT use these two FORMAT switches TOGETHER on ANY drive!

FORMAT /Z:n formats a FAT32 drive with a cluster size of n times 512 bytes. Meaning: drive: = your hard drive letter (C:, D:, etc). n = number of sectors per cluster multiplied by 512 = cluster size in bytes. Examples: n = 1 creates a 512 bytes cluster; n = 2 creates a 1024 bytes (1 KB) cluster; n = ? creates a ? x 512 = ???? bytes (???? bytes : 1024 = ? KB) cluster.

NOTE: Almost all manufactures of hard drives (these days) come formatted and most will come with a program disk for formatting, partitioning and moving your system from your old hard drive to the new one.

Low-Level Formatting Explained:

Cylinder: concentrical tracks on one or more disksides the harddrive's read/write head can be positioned over. The heads are mounted on a "fork" which positions all heads in a certain cylinder position.

Head Side: this refers to the harddrive's read/write head currently active. Since most harddrive's have at least 2 heads also the term Side is used for referring which disk and side has an "active" head over it.

Track: the combination of the cylinder all heads are over, and the selected head.

Sector: the smallest unit that can be read from/written to a disk. Without special drivers, DOS and windows can only cope with 512-byte sectors.

BootSector: (ie:, the first sector in each partition). It stores information like the number of sectors/cluster, the partitionsize, the number of sectors/FAT and the number of sectors in the partition. Also it has code to load IO.SYS and MSDOS.SYS which are used by the boot process which is of course only used in the active, primary partition that is used to boot.

Formatting & Low-Level Formatting

This is very confusing to most people. Formatting a floppy generally means that BOTH a low-level and a high-level format take place. The low-level format overwrites full tracks consisting of multiple sectors, headers and trailers. Each sector is preceded by a header and gets a trailer written behind it. Only the sectors and trailers are updated when you copy a file to the floppy (effectively writing sectors), the headers are used for finding the start of each sector on the track and are only read from (except during a format operation). The low-level format calculates a CRC-checksum (CRC = Cyclic Redundancy Check, a special algorithm that is very sensitive to even the slightest change in data) from each sector it writes, and writes that calculated checksum to the trailer. After low-level-formatting each track, the format program simply reads all sectors and trailers in it, recalculates the checksum from each sector and compares it to the checksum in the trailer. If they don't match, the sector is considered "bad". It stores all bad sector locations in RAM for later use. After finishing ALL tracks on both sides, the format program will do a high-level format of the floppy. That means that it will write a proper bootsector, clear both FAT's and the rootdirectory of the drive. Then it will recall the bad sectors found and adjust the FAT's accordingly.

What is Low/High Level Formatting

A low-level format (first done at the factory) draws magnetic lines on the hard disk, these days you rarely need to redo a low-level format.

A high-level format creates a new FAT and scans the surface of the disk, finding and marking damaged sectors (those corrupted files). Performing a high-level format is deceptively easy; simply run the Format utility.

However, the DOS FORMAT program only knows how to low-level format floppies, not harddrive's, due to the fact that sector header/trailer information varies widely between harddrive manufacturers and models. Further, the number of sectors on a track is fixed for floppies in (DOS floppy formats) but varies on most harddrive's.

Finally, EIDE harddrive's are preformatted by the manufacturer typically with dedicated programs that leave some space on the drive to "remap" bad sectors to, should they show up over time. This means that if the microcontroller in the drive detects a bad sector, it will try to reread a number of times until the checksum matches. Then it will mark that sector as bad, and store the information in another sector. All this is invisible from DOS. However, for fast AV or multi-speed CDROM writers this may mean a hickup in the datastream coming from the drive, not only when the bad sector is detected, but also afterwards because the drive will have to get the sector data from another physical location on the drive involving extra head-seeks.

Therefore, instead of low-level formatting your EIDE drive when it was trashed,

first make sure that there are no other problems involved (i.e. too fast PIO mode, wd-ctrl in use for a large incompatible drives, buggy CDROM drivers, too long (ribbons) between your controller and drives, bad RAM or bad cache RAM etc).

I strongly advise you NOT to use the "format" option inside most BIOS's (reachable from the BIOS setup screen). This FORMAT option was useful for old MFM and RLL drives, and usually DESTROYS EIDE drives! If you need to "repair" an EIDE drive (bad sectors, or "indestructible" viruses etc.) it is best to use a tool that simply overwrites all sectors. This will force the drive to "remap" bad sectors itself. Nearly all bad sectors that are not remapped can be detected by either FORMAT or SCANDISK (surface scan option) and they will mark the clusters containing bad sectors as bad in the FAT.

The Harddisk Guide

The hard disk can have a huge impact on the performance of your PC: The fact is that the rotating magnetic media of the hard disk is one of the severest performance bottlenecks, causing second-long delays while fat programs spin off the disk and into RAM. Whereas disk access times are measured in milliseconds, system RAM performance is counted in nanoseconds. Understanding hard disk operation - and optimizing - can eliminate teeth-grinding delays.

The factors that affect the speed of a Hard disk:

- Rotation speed
- Number of sectors per track
- Seek time / head switch time / cylinder switch time
- Rotational latency
- Data access time
- Cache on the HD
- How data is organized on the disks
- Transfer rates
- Interface (EIDE / SCSI)

What are sectors, tracks, heads and cylinders?

On a Hard disk, data is stored in the magnetic coating of the disk. The so called head, held by an actor arm, is used to write and read data. This disk rotates with a

constant turn time, measured in revolutions per minute (rpm). Data is organized on a disk in cylinders, tracks and sectors. Cylinders are concentric tracks on the surface of the disk. A track is divided into sectors. A Hard disk has a head on each side of a disk. Nowadays, the actuator arm is moved by a servo-motor (not a step-motor which needs more time while swinging in after moving over the desired track). All harddisks have reserved sectors, which are used automatically by the drive logic if there is a defect in the media.

Rotation speed

Typical harddisks have a rotation speed from 4,500 to 7,200 rpm, a 10,000 rpm drive just hit the market. The faster the rotation, the higher the transfer rate, but also the louder and hotter the HD. You may need to cool a 7200 rpm disk with an extra fan, or its life would be much shorter. Modern HD's read all sectors of a track in one turn (Interleave 1:1). The rotation speed is constant.

Number of sectors per track

Modern harddisks use different track sizes. The outer parts of a disk have more space for sectors than the inner parts. Usually, HD's begin to write from the outside to the inside of a disk. Hence, data written or read at the beginning of a HD is accessed and transferred faster rate.

Seek time / head switch time / cylinder switch time

The fastest seek time occurs when moving from one track directly to the next. The slowest seek time is the so called full-stroke between the outer and inner tracks. Some harddisks (especially SCSI drives) don't execute the seek command correctly. These drives position the head somewhere close to the desired track or leave the head where it was. The seek time everyone is interested in is the average seek time, defined as the time it takes to position the drive's heads for a randomly located request. Yes, you are correct: seek time should be smaller if the disk is smaller (5 1/4", 3 1/2" etc.).

All heads of a Hard disk are carried on one actuator arm, so all heads are on the same cylinder. Head switch time measures the average time the drive takes to switch between two of the heads when reading or writing data.

Cylinder switch time is the average time it takes to move the heads to the next track when reading or writing data.

All these times are measured in milliseconds (ms).

Rotational latency

After the head is positioned over the desired track, it has to wait for the right sector.

This time is called rotational latency and is measured in ms. The faster the drives spins, the shorter the rotational latency time. The average time is the time the disk needs to turn half way around, usually about 4ms (7200rpm) to 6ms (5400rpm).

Data access time

Data access time is the combination of seek time, head switch time and rotational latency and is measured in ms.

As you now know, the seek time only tells you about how fast the head is positioned over a wanted cylinder. Until data is read or written you will have to add the head switch time for finding the track and also the rotational latency time for finding the wanted sector.

Cache

I guess you already know about cache. All modern HD's have their own cache varying in size and organization. The cache is normally used for writing and reading. On SCSI HD's you may have to enable write caching, because often it is disabled by default. This varies from drive to drive. You will have to check the cache status with a program like **ASPIID** from Seagate.

You may be surprised that it is not the cache size that is important, but the organization of the cache itself (write / read cache or look ahead cache).

With most EIDE drives, the PC's system memory is also used for storing the HD's firmware (e.g. software or "BIOS"). When the drive powers up, it reads the firmware from special sectors. By doing this, manufacturers save money by eliminating the need for ROM chips, but also give you the ability to easily update your drives "BIOS" if it is necessary (Like for the WD drives which had problems with some motherboard BIOS' resulting in head crashes!).

Organization of the data on the disks

You now know, a Hard disk has cylinders, heads and sectors. If you look in your BIOS you will find these 3 values listed for each Hard disk in your computer. You learned that a Hard disk don't have a fixed sector size as they had in earlier days.

Today, these values are only used for compatibility with DOS, as they have nothing to do with the physical geometry of the drive. The Hard disk calculates these values into a logical block address (LBA) and then this LBA value is converted into the real cylinder, head and sector values. Modern BIOS' are able to use LBA, so limitations like the 504 MB barrier are now gone.

Cylinder, heads and sectors are still used in DOS environments. SCSI drives have always used LBA to access data on the Hard disk. Modern operating systems access data via LBA directly without using the BIOS.

Transfer rates

In the pictures you can see the several ways how data can be stored physically on the Hard disk. With a benchmark program that calculates the transfer rate or seek time of the whole Hard disk you can see if your drive is using a 'vertical' or a 'horizontal' mapping. Depending on what kind of read/write heads and servo-motors (for positioning the actuator arm) are used it is faster to switch heads or to change tracks.

The Interface (EIDE / SCSI)

Currently there are 2 different common interfaces: EIDE and SCSI. You will find an EIDE controllers integrated with the motherboard and that EIDE harddisks are much cheaper than SCSI drives. For SCSI you need an extra controller, because there aren't a lot of motherboards with integrated SCSI controllers. Together with the higher price of a SCSI disk a SCSI system is more expensive than EIDE.

The EIDE interface has a primary and a secondary channel that will connect to two devices each, for a total of four. That could be a Hard disk, CD-ROM or disk changers. Lately there have been tape backups with EIDE connectors, but you need special backup software.

Scanners for example aren't available with EIDE interface, only with SCSI. You can connect up to 7 devices to a SCSI bus or 15 devices to a Wide SCSI. In a standard environment, the performance of single Hard disk won’t improve much from the SCSI interface. Rather, the power of SCSI is that several devices can use the bus at the same time, not using the bus while they don’t need it. So, we see the best benefit from SCSI when several devices are all used on the same bus.

On one EIDE channel, the 2 devices have to take turns controlling the bus. If there is a Hard disk and a CD-ROM on the same channel, the Hard disk has to wait until a request to the CD-ROM has finished. Because CD-ROM's are relatively slow, there is a degradation of performance. That's why everybody tells you to connect the CD-ROM to the secondary channel and your Hard disk to the primary. The primary and secondary channels work more or less independently of one another (it's a matter of the EIDE controller chip).

The SCSI interface comes in several types. 8-bit (50 wire data cable) or 16-bit (68 wire data cable, Wide SCSI). The clock can be 5 MHz (SCSI 1), 10 MHz (Fast SCSI), 20 MHz (Fast-20 or Ultra SCSI) or 40 MHz (Ultra-2 SCSI).

Possible transfer rates of the SCSI bus		
SCSIbus clock	**8-bit**	**16-bit**

	(50 wire data cable)	(68 wire data cable, Wide SCSI)
5 MHz (SCSI 1)	5 Mbytes/s	NA
10 MHz (Fast SCSI, SCSI II)	10 Mbytes/s	20 Mbytes/s
20 MHz (Fast-20, Ultra SCSI)	20 Mbytes/s	40 Mbytes/s
40 MHz (Fast-40, Ultra-2 SCSI)	40 Mbytes/s	80 Mbytes/s

The theoretical transfer rate of EIDE is up to 16.6 Mbytes/s in PIO mode 4 or **multi DMA mode** 2 (soon **33.3 Mbytes/s**) with all the problems you may have already faced. Here you will find a **table of several interfaces** and their speeds. However, today's CD-ROM's often use PIO mode 3, while older device use PIO mode 0 to 2. Sometimes devices lie about the PIO mode they support. There are harddisks that say they are able to use PIO mode 2 but they only work reliably in PIO mode 1! Whenever you get errors accessing your Hard disk, try to lower the PIO mode first!

Possible theoretical transfer rates of the IDE bus (ATA)	
single word DMA 0	2.1 Mbytes/s
PIO mode 0	3.3 Mbytes/s
single word DMA 1, multi word DMA 0	4.2 Mbytes/s
PIO mode 1	5.2 Mbytes/s
PIO mode 2, single word DMA 2	8.3 Mbytes/s
Possible theoretical transfer rates of the EIDE bus (ATA-2)	
PIO mode 3	11.1 Mbytes/s
multi word DMA 1	13.3 Mbytes/s
PIO mode 4, multi word DMA 2	16.6 Mbytes/s
Possible transfer rates of Ultra-ATA (Ultra DMA/33)	
multi word DMA 3	33.3 Mbytes/s

It is not only the interface transfer rate that determines how fast a Hard disk is. How fast the data can be written or read from the media, e.g. data density and rotation speed is more important. The fastest interface can't do anything faster than the 'inner values' of a Hard disk are capable of. Today, most harddisks are still under 10 Mbytes/s transfer rate physically. A faster interface is advantageous on when data is read from or written to the cache in a multitasking environment with several devices accessed simultaneously.

Multitasking environments especially benefit from SCSI, since simultaneous access occurs frequently. If you have a server or are working with large files like audio, video or disk-intense applications, you will benefit more from SCSI than EIDE. There are three reasons for this:

- All modern operating systems now supports SCSI very well. Windows 3.x didn't!
- Bus mastering really works better with a SCSI bus mastering controller.
- The fastest harddisks with the best performance are SCSI.

If you need large capacities and the highest transfer rates available on the market you need SCSI. This is not because EIDE is incapable of this, it’s because of the market. High-end disks with high capacities and high performance are intended to be used in servers and aren't build with EIDE interface. At the moment, EIDE disks are only built with up to a 5 Gigabyte capacity (there is a problem with a 4 GB barrier with some BIOS's again and for drives bigger than 8 GB you need a new BIOS that supports the INT 13 functions AH=41h bios 49h) and transfer rates of about 9 Mbytes/s. If you need more, you'll have to use SCSI. Also, SCSI harddisks have larger cache RAM than EIDE harddisks.

Performance, some thoughts

You need to know how a slow or fast Hard disk affects your overall system performance in a standard environment. If your operating system isn't constantly swapping (e.g. you have enough memory) the speed of a Hard disk is only a small part of a well balanced system. Let’s say you have a Hard disk that has 30% better performance than another older one; the benefit for standard applications would be from 2% up to 18%. Sometimes, you want or need the fastest components available. Other times, more capacity and reliability is needed.

There are several programs available that test the performance of a Hard disk. Some are crap, others are good. In any case, if you have one, you get numbers that tell you something. But do you have a point of comparison? Different benchmarks mean different numbers. Different environments mean different numbers. Modern benchmarks are independent from existing data on the Hard disk (only read

performance testing can be done). But a benchmark could be affected by several things:

- To which channel is the Hard disk connected
- Is the Hard disk alone or together with other devices connected to the controller
- Under which operating system is the Hard disk tested and used
- Which drivers are loaded or **not** loaded.
- Testing at Monday or Friday etc.

The File System

It is very important and I recommend you to go through my previous chapters again.

Upgrading Your Hard Disk

Choosing a Disk Type

There are many parts to a hard disk that a real "gear head" might argue are important in selecting a new drive. Most people really only use one criterion when considering which HD to choose: cost. You can get into a lot of trouble if that is your sole consideration, and I speak from experience. Here are the basic things to consider when comparing, in order of importance:

- *Interface type, such as EIDE (Enhanced Integrated Drive Electronics).* The interface type goes without saying. You wouldn't buy a SCSI (Small Computer System Interface) HD if all you have is an IDE controller, unless you want to add the benefits of a fast SCSI drive without eliminating the existing IDE drive.
- Also, make sure to match the controller with the drive. An Ultra Fast SCSI drive will not be "Ultra Fast" if you only use a SCSI 2 controller. The same can be said for an EIDE (Enhanced Integrated Drive Electronics) HD and an IDE controller.
- *Maximum formatted storage capacity.* HD manufacture's and vendors usually quote you the maximum storage capacity, which is always more than the formatted capacity due to many factors including file system (FAT16, FAT32, NTFS), cluster size or allocation unit, type of files being stored, and so on.
- *Transfer rate.* The transfer rate is the rate at which the drive *and* controller

can send data to the system. The greater the value, the better. Ultra Fast Wide SCSI subsystems hold the current record at 40M/sec.

- *Rotational speed (RPM).* Rotational speed and transfer rates are closely related. The faster the RPM, the more data passes under the read/write head in a set period of time, allowing for higher total transfer. Faster is not always better, however. The faster the RPM, the more chance you have to drop some data. Not a problem for digital video, as in AV drives, but not so good for your spreadsheet.

- *Average seek time (also known as access time).* The access time is the amount of time that lapses between a request for information and its delivery. The lower the value, the better. Most modern HDs have an access time of 10ms or less.

- *Cost per megabyte.* This is a good way to compare two drives of different storage capacity. For example, the IBM 12G drive cost $349 (349/12,000M), which translates to about 3 cents per 1M. A 1G drive costing $99 ran me 10 cents per 1M; therefore, I'm getting a better deal with the larger drive even though it might cost me more.

- *Onboard cache (or just cache).* Onboard cache acts as a buffer for data being transferred to and from the HD. The larger, the better.

Adding a SCSI Drive to an IDE System

If you would like to free yourself from the bindings that IDE imposes on you, but can't bear the idea of pitching a perfectly functional IDE drive, consider adding a SCSI drive as your upgrade instead of another IDE/EIDE drive. This will allow you to take advantage of the greater performance/expandability that SCSI has to offer, while retaining your current investment. And if you ever decide to invest in a new SCSI-based system, you can take the SCSI drives with you while retaining a completely functional IDE system.

> *In an IDE/SCSI system, the IDE drive must be the boot drive. Very few computer systems have the ability to allow the SCSI drive as the boot drive with an IDE drive installed.*

Planning the Installation

Once you have decided on the type of drive to buy (and you've picked out a controller, if necessary), it's time to plan your actual installation process. Drive upgrades are usually straightforward, but proper planning is the key to avoiding problems later on.

Data Backups

Before you attempt to perform any type of HD upgrade, you should perform a full system backup of your current hard disk's). If you are replacing a disk outright, a backup is vital for restoring your work to the new disk. If you are adding a second drive, a backup is not quite so critical, but will still protect you in the event you might accidentally lose data on your original disk.

Backups can take many forms: you can use floppy disks, SyQuest cartridges, Iomega Zip or Jazz cartridges, or any type of tape drive. Most of these drives come bundled with some form of backup software, so once the "backup drive" is connected, you should be able to start the backup software and proceed almost automatically.

Power Considerations

Power is the first issue to consider. A typical hard disk requires about 10 watts of power for proper operation. This may not sound like much, but you must be sure that your power supply is able to provide that much power. Otherwise, your supply may become overloaded. Overloaded supplies often cause unpredictable PC operation, or prevent the PC from booting at all. In extreme cases, a severely overloaded power supply can even break down. Here are some rules to help avoid power problems:

- If you're just replacing an existing hard drive with a new one, power should not be a problem.
- If your system has not been upgraded before, it can usually support an extra drive without any problem.
- If your system has been significantly upgraded already, pay attention to the system's operation after you add a new drive. If the system behaves erratically, you may need a higher voltage power supply.

You also need to have a 4-pin power connector available for the new drive. If you're just replacing an existing hard disk outright, you can just reuse the power connector on the replacement drive. When adding a new drive, be sure that there is an extra power connector available from the power supply.

If you don't have a power connector available, you can purchase a *Y-connector* that splits an existing connector into two separate connectors. You simply remove a power connector from a drive, install the Y-connector, plug one arm of the Y-connector back into the original drive, and you've got a spare connector for that new drive.

Choosing a Drive Bay

You also need to decide where your drive is going to go in the system. When replacing a drive outright, you can simply reuse the original drive bay. If you are adding a second hard disk, you need to locate an unused drive bay. Large desktop and tower chasses often have several drive bays available. You may have to hunt for an open bay in a mini-desktop enclosure.

Drive bays are typically referred to as *external* and *internal.* External drive bays are located behind those plastic plugs you see on so many plastic housings. Floppy disks, CD-ROM drives, and tape drives all demand these external drive bays. Internal drive bays are little more than metal brackets inside the chassis where you can bolt a drive securely. Because you do not need to insert or remove anything from a hard disk, you can use external or internal drive bays.

Most hard drives are 3 1/2 inches in size, so if you want to use a 5 1/2-inch bay, you need to use mounting brackets. These mounting brackets and slides usually come with the drive but can be purchased separately from a local computer store if necessary.

Drive Configuration

Configuration of your new hard disk is paramount to its proper function. Examine your new drive and locate any jumpers . For an IDE/EIDE drive, there are typically three ways to jumper the drive:

- As the only hard disk
- As the primary disk in a two-disk system
- As the secondary (slave) disk in a two-disk system

If the hard disk is to be the *only* one in your system, the factory settings are usually correct. In this case, the jumper is not being used. You can skip to the next section.

When you use the new disk as a secondary drive--that is, the second drive on the same ribbon cable--make sure it is jumpered as the slave drive, and set the original disk so that it is the master drive. This is the least intrusive method to adding a new disk to your system. You will still boot from the original drive with all files intact, but you just see an additional drive letter show up in your Windows 95 Explorer window (or File Manager if in Windows 3.x).

An EIDE controller can have two separate chains, each with its own master and slave drive for a total of four devices.

Making the new drive the master can get a bit messy. Because the drive is new, it will not have an operating system on it and will not boot. On an IDE/EIDE system, the master drive (the C: drive) has to be the boot drive. Therefore, you have to reinstall your OS on the new drive in order to use the system. All files should still be intact on the original drive; it will just be the next logical drive, usually D:. To get around this limitation, install the new drive as the slave. Boot the system, and

duplicate the entire old disk to the new one. Shut off the system and switch master/slave status. Reboot the system to test. If all works well, you can reformat the old drive and you are ready to rock.

Is Your Controller IDE, or Is It EIDE?

It's OK to use the same controller with the new drive, but unless the controller is an EIDE controller, you will not be able to take advantage of all the performance enhancements of your new drive. If your controller is embedded in the motherboard and you are not sure if it is IDE or EIDE, look for a Primary and a Secondary controller. If you have these two controller, it is EIDE; only EIDE supports four devices (two on each connection). If you only have one controller (IDE), check your BIOS (Basic Input Output System) setup to see if you can disable the on-board hard drive controller. If so, you can buy an EIDE controller card and use it instead.

SCSI disks are a bit easier to configure because you need only select the proper device ID for the drive. By convention, the boot disk (the C: drive) in a SCSI system is set as ID0, and a second SCSI disk (the D: drive) can be set as ID1.

If your new SCSI disk falls at the end of a SCSI bus, you also need to install a set of SCSI terminating resistors on the drive. Fortunately, SCSI kits typically include terminating resistors, and provide specific instructions on how to install the terminators. Make sure to check the installation manual that came with your SCSI controller card. It's not always simple to terminate a SCSI chain; it depends on the type of controller (for example, some controllers support both internal and external SCSI chains, and both need termination).

NOTE: IDE/EIDE and SCSI hard disks *can* coexist in the same system, but you cannot boot a PC from a SCSI disk if there is an IDE/EIDE disk in the system as well. IDE/EIDE disks automatically take precedence. If you need to boot from a SCSI disk, you have to remove any IDE/EIDE disks from the system first.

Installing a New Drive

Now that you've determined what type of drive configuration you have in the previous section, you can install the new drive. This section is divided into similarly appropriate parts, with steps common to all new installations here.

Before You Begin, You Need:

- A screwdriver
- An open drive bay (either 51/4- or 31/2-inch)
- Slide rails
- 3 1/2- to 5 1/4-inch mounting brackets (if you are putting the drive in a 5 1/4-inch bay)

- Mounting screws (come with the drive)
- Proper length ribbon cable

1. Shut down, turn off, and unplug the PC.

2. Open the system.

3. Discharge your static.

The following sections are specific to the type of drive configuration, and you should skip to the section that suits your needs.

Replacing an IDE Drive

This section describes the process of replacing a drive in a single drive system. For information on adding a new drive, skip to the next section.

TIP: It is highly recommended that you replace your old IDE controller with a new EIDE controller. The additional cost is more than worth it in performance gains.

1. Locate the old drive, unplug the power connector and the 40-pin ribbon cable *Remove the 4-pin power connector. Remove the 40-pin IDE ribbon cable.*

2. Locate and remove the mounting screws. These are found either on the side of the drive bay or in front *Remove the drive mounting screws.*

3. Carefully slide the drive out, making sure not to snag any other cables or wires in the process.

4. Remove and save any mounting brackets, slide rails, and screws that may be attached. You can reuse them on the new drive. That's the surest way of getting a good fit.
Remove the old drive.

5. Attach any mounting brackets and/or slide rails from the last step to the new drive.

6. Check the position of the key in your 40-pin ribbon cable. This key assures the correct alignment of the cable to the drive.

TIP: Don't panic if your ribbon cable does not have a key; not all do. There will be one colored wire at the side of the cable to indicate the #1 pin position, and the drive will also indicate this pin on its underside

7. Slide the new drive in place of the old one, and replace all mounting screws.

CAUTION: When securing a hard disk, be extremely careful to avoid stripping or

cross-threading a mounting hole. An unevenly mounted drive will vibrate excessively. This can lead to premature drive failure.

8. Reattach the ribbon cable, noting the position of the key, or the #1 pin position.

9. Reattach the 4-pin power connector.

TROUBLESHOOTING: My system doesn't even start when I turn the power on. What happened? You may have inadvertently replaced the ribbon cable reversed (#40 wire to the #1 pin). Turn the power off and double-check that the #1 pin positions are lined up. **Check that everything is affixed firmly.**

Adding a New EIDE Drive

This section describes the installation of an additional EIDE drive to a system with an IDE/EIDE drive already installed. This will be the most likely scenario because most people are unwilling--no matter how old and slow the old drive was--to simply discard it; it probably cost more than the new one! If this isn't you, you can skip to "Adding a New SCSI Drive" or back to "Replacing an IDE Drive."

Follow these steps to add a new EIDE drive:

1. Locate an open drive bay for your new drive, preferably one as close to the original drive as possible. The reason for this is the limited distance between drive connectors on the ribbon cable.

2. Attach any mounting brackets and/or slide rails, as required by your particular case design, to the new drive and slide it into place. Note that some cases do not require any mounting brackets (3 1/2-inch bays) or slide rails.

3. Check to see if you have an available 4-pin power connector; most systems will have at least one or two spares. If not, you can purchase a Y-connector that will split the power off one of the other power connectors.

4. Attach the 4-pin power connector to the new drive.

5. Attach an available drive connector from the 40-pin ribbon cable--note the key in the connector and the notch in the drive to the new drive.

NOTE: If you have an EIDE controller and the secondary IDE connector is unused, you can attach an additional 40-pin ribbon cable to it to control your new drive. In that case, the jumper settings on both the new drive and the old would be set to MASTER, because these controllers function as two separate controllers. The next device on either cable, whether hard disk or CD-ROM drive, would then be set to SLAVE.

There you have it. A new drive is born. Now you are ready to move to the next phase, "Preparing and Formatting the Drive."

Adding a New SCSI Drive

In this section, you learn how to add a new SCSI hard drive to your system. If you don't have to complete this task, skip to the next phase "Preparing and Formatting the Drive," or refer to the earlier sections "Adding a New EIDE Drive" or "Replacing an IDE Drive."

About the Controller Card

The actual process of installing the card is no different than any other expansion card you might add (see Chapter 6, "Basic Device Configuration and Installation"). Very few new motherboards come with an embedded SCSI controller, as EIDE does, and neither do most home systems you might want to upgrade.

The first thing to note is that most SCSI controllers can have both an internal chain and an external chain, and both need to be terminated

Most internal connectors appear similar to the IDE counterpart except bigger-- 50 pins instead of 40 pins. The exception is an Ultra Wide SCSI connector (16-bit path rather than the standard 8-bit) which is actually visually smaller but uses 68 pins. External connections can be different, too. Some older controllers use a 50-pin connector that resembles a parallel port. Most newer controllers use a 50-pin mini-connector or a 68-pin mini-connector for Ultra Wide. There are adapters available to change from one kind of connector to another, in the event a device you want to connect uses a different connection.

Compatibility and Recommendations

As you are beginning to see, there is not quite the same standardization in the SCSI world as there is in the IDE. This is one reason SCSI has been relegated to the world of the "high-end" machine or computer geek. Only these people were willing to take the time to understand all the factors necessary for a properly functioning SCSI subsystem.

> With this in mind, I recommend you purchase an Adaptec SCSI controller card because it is the most standardized SCSI board out there. In fact, Adaptec has been pushing the standardization of SCSI for years. The ASPI Advanced SCSI Programming Interface was once Adaptec SCSI Programming Interface, before they turned it over to public domain. There are other SCSI boards out there with as good, if not better, price/performance ratio, but the headache you could get with incompatibilities and poorly written device drivers isn't worth the difference.

Also note that the SCSI board has its own connection for the LED lights that flicker on the outside of your case to let you know there is some hard drive activity going on. It's not necessary to connect this, but some people like to use these lights for diagnostic reasons, or just because it looks cool.

Termination

Termination is one of the most important factors in setting up your SCSI subsystem. In a nutshell, the last device on a SCSI chain--internal *and* external--needs to have termination enabled. This can be accomplished in different ways, depending on the device, but this fact holds true of all SCSI chains.

For example, if you have only two devices--a controller card (yes, it counts as a device) and an HD--they both need to be terminated. It's usually automatic for the controller; you do not need to do anything (not always the case if you are not using an Adaptec controller, though). The HD will have either terminating resistors that plug into the underside of the drive or have a jumper/dip switch to set, as in the case of the IBM Ultrastar ES 2.16G Ultra SCSI hard disk I used in this upgrade. Now add another device to this party, like an internal SCSI CD-ROM drive. If you were to install it *between* the HD and the controller, no termination is necessary. On the other hand, if you installed it after the HD, you would need to remove the termination from the HD and terminate the CD-ROM drive.

Got it? Think so, huh? Let's add an external scanner to this scenario. Both the scanner and the HD (or CD-ROM, in the last case) need to be terminated. Why? The controller card auto-terminates. In the case where the only devices were internal, the card was one terminator and the drive was the other. Both ends were terminated. When you added the external device, the controller card becomes non-terminated and the scanner becomes the other end of the chain. Now the scanner is one terminator and the internal drive is the other.

Simple, right? The bottom line is you need two terminating devices in any SCSI chain--one at both ends.

Setting SCSI IDs

Each device on a SCSI chain has its own unique ID called its *SCSI ID.* You choose this ID either through jumpers or switches, and they can be set to any number from 0 to 7. This has no relation to the device's physical orientation on the SCSI chain, nor does it affect termination in any way. There are some general rules though:

- The boot disk is usually set to ID 0.
- The host adapter is usually set to ID 7.

Some new SCSI devices are supporting *SCAM (SCSI Configured Automatically)*, where the host adapter assigns the unique IDs at boot up automatically. If your devices support this, be sure to enable this feature in the host adapter's BIOS, as described in the next section.

Configuring the Host Adapter

The SCSI controller card (also known as the *host adapter*) uses its own BIOS, similar to the motherboard BIOS, to configure the devices under its control. These settings are different for each host adapter, but in general there are a few important features you should learn about that are common to all:

- *Extended BIOS translation*. Enabling this feature allows MS-DOS 5.0 and above, including Windows 95, to support drives larger than 1G. This option is not necessary under OS/2 or Windows NT.
- *BIOS support for bootable CD-ROM*. Enabling this feature allows you to boot your system from special bootable CD-ROMs.
- *Plug and Play SCAM support*. SCAM automatically assigns a unique SCSI ID to any device attached to the SCSI chain that supports this feature.
- *Target Boot ID.* This is the SCSI ID of the disk you want to boot from. With SCSI, you can choose which disk you want to boot from, unlike with IDE.

Preparing and Formatting the Drive

Now that you've installed your new drive, made all the proper connections, and replaced the case cover, you need to let the computer know the new drive is there. This is done through the BIOS Setup program.

CMOS Considerations

The *CMOS (Complementary Metal-Oxide Semiconductor)* is the chip that holds the information your motherboard's BIOS has recorded on it. Some people use these terms interchangeably.

The BIOS reads the system information contained in the CMOS, and then checks out the system and configures it. Next, the BIOS looks for an operating system on the boot drive (drive 1 or C: drive), launches that OS, and then turns control over.

Once the BIOS setup is activated, most new BIOS's have the IDE HDD Auto Detection option. This is a great improvement over older BIOS programs that required you to know things like the cylinders, heads, sectors, and so on. You need to go through this process so your computer can register the new drive and make it "visible" to your OS.

In case you have one of these older BIOS's or the auto detect didn't correctly ID your drive, you need to enter this information manually. Your new drive will most likely have this configuration information printed on a label pasted to the top of the mounting chassis. You might also find the information on the manufacturer's WWW site.

Because every BIOS works a bit different, you need to read the BIOS Setup instructions that came with your computer, but some general instructions follow:

1. Go to User Defined Settings.

2. Enter the parameters from your drive. Usually, only the Cyls (cylinders), Heads, and S/T (number of sectors per track) are necessary; landing zone and capacity are usually not required.

The Meg field should then reflect the drives' unformatted capacity in megabytes.

NOTE: Be sure to save your changes before exiting a CMOS Setup routine. If you forget to save, the new disk's parameters will be lost, and the drive will not be recognized.

Translating BIOS

It is important to note that not all older BIOS's will be able to recognize disk drives larger than 504M. Those that do are called *translating BIOS's*. Even if your BIOS correctly identifies the new drive's parameters, it doesn't mean the BIOS will translate those parameters. Here's how to check:

1. While at the BIOS Setup screen, look for a setting called LBA, Large Block Access, or Translation, and enable the option.

2. Check if the Auto configure Drive Type returns a heads value greater than 16; if so, you are probably OK.

3. Call your computer's vendor and/or manufacturer and ask.

What do you do if you don't have a translating BIOS? You can upgrade your BIOS (see Chapter 7, "Working with the BIOS," for help), or you can use software utilities like EZ-Drive that comes with all Western Digital hard drives over 528M (which is all of them these days), or Ontrack Disk Manager. The software works; I've used it. It does have some limitations, so read the instructions carefully and weigh these limitations against the cost of an upgraded BIOS.

Next task is to format the drive and done.

Hard Disk Crash Recovery Tools & Procedure

If you've lost data, suffered a virus attack or hard drive crash, or accidentally

reformatted your hard drive, there *is* hope. A new breed of PC disk recovery tools go far beyond the capabilities of DOS' undelete or Windows' ScanDisk. Here are some of the best tools. *There descriptions are taken as it is from their vendors site. Anything written in italics is edited by me. *** indicates the ranking more the better*

TOOLS Required in Disk Crash

1) Power Quest Lost & Found (http://www.powerquest.com/) *****

Rather than spending thousands of dollars to send your hard drive to a data recovery center, Lost & Found lets you automatically recover and restore data after accidental (or even intentional) data loss, or from corrupted media caused by a disk crash or logical system failure. Our patent-pending technology will even recover data if the partition has been reformatted or if the FAT tables have been destroyed! In fact, as long as your disk is still spinning, Lost & Found can locate and recover almost any file, anywhere on your disk.

Well I have tried out Lost & Found once. It is a DOS base software and is easy to operate. In my instant case it recovered all the data, while Norton Disk Doctor and Scandisk refused to operate and Norton disk doctor kept on crashing while operating on this drive. If your hard drive crash and to recover the data you have to keep ready other backup medium probably second hard disk or zip drive or box of floppy disks as it refuse to recover data on damaged disk (though it is good for safety) .

---

2) TIRAMISU for FAT32 V3.03 Ontrack Data International **** http://www.ontrack.com/op/op_7/op_7.asp

- TIRAMISU scans the drive even when there are physical damages. The found data are analyzed and reconstructed.TIRAMISU can handle drives without readable boot sector, readable Fat or readable directories. It can handle drives that are not recognized by DOS anymore.
- TIRAMISU automatically creates a VIRTUAL DRIVE in memory. This virtual drive looks like a usual file manager. You can see the lost directories and files of your crashed drive. Now files and directories can be viewed and copied to a safe medium.
- The extensive use of our sophisticated pattern recognition technology enables TIRAMISU to put the right pieces of data together again. Even disks with very few administrative informations left can reach a high recovery quality.
- TIRAMISU recovers much more data than any other 'disk doctor'.
- TIRAMISU is NON DESTRUCTIVE and READ ONLY. It does not put any data onto your crashed drive. Recovered data are restored to another destination (disk, diskette, network, interlink).
- TIRAMISU works on WIN95 rel. B platforms with 32-bit Fat.

- Versions for DOS/WINDOWS (16-Bit Fat), NOVELL and NTFS are available too.

3. *Stellar* version 7.0 www.stellarinfo.com *****

Well this is the best software for the data recovery. ***It won the PC World editors choice award*** *for the data recovery. This program sold like hot cakes when CIH virus destroyed many hard disks. While best product only one thing keeps it behind the Lost & Found - User Interface. Lost & Found can be operated even by general person but for Stellar it requires a novice user. But even after that it is best for your data recovery needs.* Contact Stellar Info Systems, 205 Skipper Corner, 88 Nehru Place, New Delhi - 110011, INDIA for more info.

4. Crash Proof http://unistal.com/crashproof.htm ****

Crash proof is a Hard Disk Crash Prevention and data recovery software. Once installed Revives the disk against any kind of software crash. Vendor claims about crash proof are:

protects the disk against any kind of software crash.
protects the disk against crash due to viruses like Win CIH, Worm, Explore...
protects the disk against Software malfunctioning.
the disk which has just been formatted.

You should use Crash Proof Because :

it displays the directory tree of the crashed disk. * *(It is important. If FAT gets corrupted you need this very much)*
it requires no technical knowledge to operate.
it requires only a few minutes to revive the disk.
it is the complete solution to multiple problems.

GET RID OF MESSAGES LIKE :	**No More Data Loss due to :**
General Failure Reading Drive	Accidental format
Invalid Media Type Error	Deltree Files
Invalid Drive Specification	Worm Explore.Zip Virus (*.Doc, *.PPT etc.)
Invalid Partition Table	Any software / virus Crash
System Fault, Sector Not Found	
Data Error Reading Drive	

Well it is both a protective and recovery software.

UNDELETE - I have emptied my recycle bin !!

What Happened to UNDELETE?

When Microsoft created Win95, they removed the UNDELETE program. In doing so, they made life miserable for people who either delete files from the Windows Recycle Bin or Quick-format their drives. You see, the smart guys at Microsoft figured it didn't matter if you deleted files from the recycle bin because the recycle bin protects you from accidentally deleted files.

But what happens when you delete the files from the recycle bin, and then want to get them back?

And then what happens if you Quick-format your drive under DOS or Windows? THAT causes the file names in the directories to appear erased, just as if had deleted them, but the data for those files is still on the drive. A FULL erase will actually write formatting information to all the file areas and totally wipe them off the disk, but a quick format leaves them there except for changing the directory entries. So what do you do about that?

5. How Repo 2000 Recovers Your Files

Repo 2000 is a Windows application program that runs on Win95, Win98, or Windows NT 4.0. The Master Boot Record must be intact on the drive you are recovering, and the drive to be recovered must not be the boot drive. ***Repo 2000*** scans the drive to be recovered for directory entries, and rebuilds a table of all directory structures and files on the drive. Then it allows you to specify which files are to be recovered and the target drive to which to send the recovered files. You never send the recovered files to the drive being recovered because that would risk overwriting other files.

Repo 2000 will recover files from FAT12 drives (floppy diskettes), FAT16 (DOS, Win95A, and NT4.0) drives, FAT32 (Win95B and Win98) drives, and NTFS drives; it does this locally or over a network. A special feature allows a Win95 or Win98 workstation to recover files from an NT 3.51 server across a network.

---

6. Other Undelete Tools

Undelete 1.2 from http://www.alsos.com/home.html

Norton Unerase from Norton Utilities www.symentac.com

Unerase from Mcafee Nuts & Bolts www.mcafee.com

Other General Data Recovery and Tools For Daily Users

Here are some utilities for Windows 9x systems :

1. **Norton Utilities** - It is the standard and contains with others three utilties you need most www.symantec.com.

- Norton Disk Doctor : It is available in two version. One for Windows and other replaces scandisk in DOS Mode. Windows version is though quite stable. DOS version I have tested from (Norton Utilities 3 till Norton 2000) causes General Protection Faults if large files or large disk is there. Norton Disk Doctor is just improved version of Scandisk with some bells and whistles and some feature extra. Norton Disk doctor is suitable for some minor glitches to FAT, daily routine checks and recovering data from floppy disks in case of bad sectors and does that job quite well. But in case of disk failure this is not the tool you can trust with full confidence. Lost and Found and Stellar are recommended for more safety.

- Norton Speed Disk : Used to optimize Disk. One of the best and better than windows defrag.

- Norton Unerase : Recovers deleted files. But still it lacks behind Lost and Found in recovering deleted files.

- It also includes other tools like Norton Disk Edit etc. which I think are for "super novice users".

- What Norton Utilities don't include from previous versions Norton Utilities 8.0 for DOS. DiskTool - It recovered bad floppies by putting new directory information on them It was able to repair the most floppies which NDD can't even read. (180 Kb) http://allf18.virtualave.net/disktool.zip.

2. ***Nuts and Bolts*** - Another good Set of utilities from the creators of Mcafee Antivirus. with more or less same features and better user interface from Norton Utilities. It contains all the programs what Norton offers and because of it's simple and elegant interface and design it is recommended. www.mcafee.com

3. ***FIX-IT Utilities*** - Well to be frank I have never run Fix-It. But I have read a review about it in a magazine. According to the editor FIX-IT Utilities was also good. One thing kept it ahead of Norton and Nuts & Bolts was the speed and small foot print the program required compared to its counterparts. Well Norton System Doctor was a Big System Hog and clogged about 20 to 30 % of system resources

compared to 5 % of Fix-IT. www.mijenix.com

4. **Anti Virus Packages**

- ***Norton Anti Virus***
- ***Mcafee Anti Virus* - *Get free trial version from here***

Choice is simple toss a coin and choose. But I recommend that you should use both because in these times of internet where virus spread like a boooom you should not trust on any one package and use both for more safety.

What To do before my HDD could Crash !!!

Well as old saying goes ***'prevention is better than cure'.***

Steps you should take before a crash *(Preventions)*

Rescue Disk

1. **Create a bootable Diskette**
2. **It should contains latest drivers and utilities**
3. **If you have any utility programs and Anti Virus packages such as Norton Utilities or Nuts & Bolts or Mcafee which provides for creation of rescue diskette. Create it.**
4. **Keep your windows operating system CD always ready and in safe place.**

Backups

1. **You should always do regular backup of most critical data of your system.**

!!! Crash !!!

??!@*!!*?* what to do !@*!!*?*??

You may face these problems

1. System can't boot from my disk. *!!! Pray God !!!* .
2. Though system can boot but some of my directories have been wiped out. *!!! Ha*

Ha Ha !!!

3. I can't even see the directory listing. Listing shows some wired characters with ting tong tunes from small forgotten PC speaker. *!!! Ting Tong Ting !!!*
4. My hard disk has been completely wiped out. *!!! Best lets laugh !!!* .
5. If hard disk is partitioned all or some of partitions have been wiped out. *!!! Hi Hi Hi !!!*
6. FAT is corrupted. *!!! ho ho ho !!!* .
7. More & more problems..... *!!! I am enjoying !!!* .

Solutions

1. Keep it cool. Take it easy. Have a mug of coffee, sit, relax and start the work.
2. First check for any loose cable or loose card inside your computer. Make sure all cable connections are tight, all the connectors are screwed down and fixed properly in.
3. Check whether your cmos info is all correct. May be it go corrupted and you get invalid drive specification. Re enter bios setup by rebooting the computer and check that all things are entered correctly. If all other fails lets start the dirty work.
4. Bring all your recovery rescue disks. Check their write protection tab. If not protected, lock them, there could be a virus.
5. Boot from your disk.
6. Run your anti-virus software.
7. If there is virus cure it. Many times a virus makes Hard Disk inaccessible. If all goes well you will have your data back.
8. Virus removed or if no virus found but hard disk is still damaged. Read on.
9. Now run scandisk or Norton Disk Doctor, or Disk Minder (nuts and bolts). Errors found. Repair them with undo option. It may be needed in case hard disk is not fully repaired and you may use advance tools.
10. Now turn of advance tools. Check above tools like Stellar or Lost & Found.
11. Follow their steps.
12. If they require another disk for data recovery borrow a hard disk from your friend. In case it is not possible then keep large box of floppy disks ready. (it depends on your data size).

13. All goes well recover data thank me and god.
14. Still at first stage - take your disk at data recovery shops. It's now there turn.

Data Recovered What to do now

1. Take a look again whether you have not left anything on disk. Backup up everything you think you need.
2. Is Disk still usable. Partitioned it again. Use Fdisk to delete old partitions, create new and then format. As it is good because many internal data structure on disk may have suffered damage so creating new partition is highly recommended.
3. Hard disk is not usable. Don't throw it. Use it as a paper weight or for keeping coffee mug on it. It will impress your friends. Purchase a new disk.

TASK COMPLETED

Some More Technical Info

Understanding Partitions

Individual hard disk drives can be divided into *partitions*, which are logical subdivisions of the hard disk device. Each hard disk drive can have up to four partitions. A partition can be either *primary* or *extended*. A hard disk can have up to four primary partitions, although only one primary partition can be active at a time. Extended partitions are made up of *logical drives*, which are distinct hard disk letters that appear for the operating system. A hard disk can have only one extended partition. Typically, hard disks that result in multiple drive letters are configured with a single primary partition and a single extended partition, with the extended partition containing from one to many logical drive letters. (You can have any number of logical drives within an extended partition; in reality, you are limited to the number of available drive letters, up to the letter *Z*.)

Partition data for a hard disk drive is stored as part of the Master Boot Record (MBR) located at cylinder 0, head 0, sector 1. A 64-byte section of the MBR contains the partition configuration for the hard disk. Each partition is defined by a 16-byte entry (which means that there can be no more than four partitions, because 16 bytes times four partitions is 64 bytes). Within the MBR sector, the partition data is stored starting at decimal offset 446 and uses up the remainder of the 512-byte MBR sector (the last 2 bytes are not related to the partition data, but are the end-of-sector marker, which is always 0x55AA).
Each 16-byte partition table entry is arranged as follows:

Byte 00 stores the boot indicator, which is always either 0x00 or 0x80. 0x80 indicates that the partition is used for booting the system; 0x00 indicates that the partition is not used for booting.

Byte 01 stores the starting head number of the partition.

Bytes 02 and 03 store a combined entry locating the starting sector and cylinder of the partition. The first 6 bits store the starting sector; the remaining 10 bits store the starting cylinder number.

Byte 04 stores the System ID. This ID indicates the file system being used on the partition and is set by the FORMAT command when the partition is formatted with a particular file system (such as FAT16, FAT32, or--on Windows NT--NTFS).

Byte 05 stores the ending head number.

Bytes 06 and 07 are another combined entry storing the ending sector and cylinder of the partition. Again, the first 6 bits store the ending sector; the remaining 10 bits store the ending cylinder number.

Bytes 08 to 11 store the *relative sector*, which is the relative sector number at which the partition starts.

Bytes 12 to 15 store the number of sectors within the partition.

Logical drives store their partition data somewhat differently than primary and extended partitions. What happens is this: An extended partition entry in the MBR partition table indicates the extended partition's first sector, which is the location of the first logical drive in the extended partition (an extended partition must contain at least one logical drive if it is to function). The first sector of the first logical drive stores *another* partition table. This logical drive partition table is stored in the last 64 bytes of that first sector (leaving 2 bytes for the end-of-sector marker) arranged just like the main partition table in the MBR. However, the logical drive's partition table contains only two entries: The first entry contains the configuration for that logical drive, and the second entry contains the configuration of the next logical drive. Entries three and four are empty and are not used. The second entry points to the next logical drive, which again contains its own logical drive partition table, and so forth. As you can see, logical drives within an extended partition are therefore defined by this linked list of partition tables, each one pointing to the next.

Understanding FAT

Any operating system relies on one or more *file systems*, which are the methods used to store files on storage devices. There are many different kinds of file systems, such as FAT,

New Technology File System (NTFS), High Performance File System (HPFS), CD-ROM File System (CDFS), and so forth. Windows 98 can use four different file systems: FAT16, FAT32, CDFS, and Universal File System (UFS) for DVD-ROM drives. Most of the following discussion about FAT concerns FAT16--differences between FAT16 and FAT32 are discussed in the following section.

FAT stands for File Allocation Table, a method for storing files and file directories on a hard disk. FAT has a long history, being used first under MS-DOS. Through a variety of techniques, including the new FAT32 variant, FAT has been extended and improved over the years.

A FAT-formatted volume is arranged starting with a Partition Boot Sector, followed by two identical copies of the FAT (FAT1 and FAT2), the root directory listing, and then the remainder of the volume. Two copies of the FAT are stored in case one is damaged.

The Partition Boot Sector contains the information needed to boot an operating system (if the partition is a primary partition intended for that purpose). The data in the Partition Boot Sector is described in the following chart.

Bytes	**Description**
3	Jump instruction
8	OEM Operating System name in text format
25	BIOS Parameter Block
26	Extended BIOS Parameter Block
448	Bootstrap code

The BIOS Parameter Blocks (both normal and extended) store additional configuration information about the volume, such as the number of bytes per sector, number of sectors per cluster, number of root directory entries, and so forth.

FAT volumes are divided into allocation units, called *clusters*. FAT16 can handle up to 16 bits worth of clusters (65,535). FAT32 can handle up to 32 bits worth of clusters (4,294,967,295). Depending on the size of the volume, clusters can be different sizes. The minimum cluster size is 512 bytes; larger clusters are always a power of 2 multiples of 512 bytes (for example, 1024 bytes, 2048 bytes, 4096 bytes, and so on).

The FAT table is a simple linked list. Each file's entry in the directory points to the first cluster used. Using the corresponding FAT table entry, the operating system can then work down the list of FAT entries for each cluster, locating each of the clusters occupied by a file. Consider the following example: A file is 70 K long, and the volume uses 32 K clusters. The file's entry in the directory says that the first cluster is number 2,345. The operating system then finds all the pieces of the file by first reading FAT entry 2,345 (for the first 32 K of the file). The FAT entry indicates that the next cluster is, say, number 4,123 (for the second 32 K of the file). Cluster 4,123 indicates that the next cluster is number 932 (the remaining 8 K of the file). The FAT entry for cluster 932 stores 0xFFFF instead of a pointer to the next cluster, thereby indicating that the last cluster for the file has been reached.

Each FAT entry corresponds to a cluster, and contains relatively simple information:

Whether or not the cluster is in use

Whether or not the cluster is marked bad

A pointer (a 16-bit entry on FAT16) for the next cluster in the chain, or a value

(0xFFFF) indicating that the cluster is the last one occupied by a file

File information is stored within the volume's data area, except for the root directory, which is in a fixed position on a FAT16 volume. The root directory is limited to 512 entries on FAT16.

Each directory on a FAT volume is actually a file, but one that is marked as being a directory entry so that the operating system knows how to deal with it. (Unless you are editing the byte-by-byte information on a disk, you won't be aware that a directory is actually a file.) Within the directory "file" are entries for all the files and subdirectories in the directory. When you enter a DIR command at a command prompt, you are simply displaying the contents of the directory file, formatted so that it's easy to read. Directories consume clusters just as files do. Notice that directories are *not* the FAT; the FAT is simply a table that lets the operating system locate parts of the files and directories listed in a directory, starting with the root directory.

Each directory entry contains the following information:

The name of the file or directory, stored as 11 bytes (inz 8.3 format; the period is not stored)

8 bits indicating the attributes of the entry

24 bits indicating the time the file was created

16 bits indicating the date the file was created

16 bits indicating the date the file was last accessed

16 bits indicating the time the file was last modified

16 bits indicating the date the file was last modified

16 bits (on FAT16) indicating the first cluster number occupied by the entry

32 bits indicating the size of the entry

The attribute bits indicate whether an entry is for a file or another directory (a subdirectory), whether or not the entry is for a volume label, and the user-settable attributes (read-only, system, hidden, and archive).

To tie all the parts of this discussion together, let's examine an extended example: A file called TEST.FIL is stored in the directory C:\Windows\System\, is 50 K long, and is being read into an application. The sample volume uses clusters that are 32 K long. (Some steps are simplified because they're not relevant to this discussion).

1. The application requests the file's data from the operating system. The application sends the operating system the file's name and directory, in the form of a fully qualified pathname: C:\Windows\System\TEST.FIL.

2. The operating system locates the file by first scanning the entries in the root directory of drive C for an entry called Windows that has the directory attribute set (indicating that it's a directory).

3. The Windows directory entry indicates that it starts at cluster 555. The FAT is then read; using the linked list in the FAT described earlier in this section, the operating system discovers that the Windows directory occupies clusters 1123, 2342, 523, and 4923. Using that information, the operating system reads the Windows directory and scans it for an entry called System.

4. An entry called System is found in the \Windows directory listing, and it has its directory attribute set. The System entry indicates that number 1154 is its first cluster.

5. The FAT is read again, starting at cluster 1154 and following the chain until all System directory clusters are known. Using that information, the operating system reads the System directory table into memory and scans it for an entry called TEST.FIL. The entry is found, and its directory attribute is clear, indicating that it's a file. By reading that entry, the operating system finds that the first cluster of TEST.FIL is number 2987.

6. The FAT is read again, starting at cluster 2987. Using the linked list, the operating system locates both of the clusters holding TEST.FIL and can then read both clusters' contents into memory.

7. The operating system then passes the cluster contents (the file's contents) to the application as a stream of bytes.

As you can see, retrieving a file is a lot of work! Fortunately, the system keeps most directory entries--as well as the entire FAT table--in RAM, so the necessity to read the directories and FAT entries doesn't require much disk activity. However, notice that writing changes to a file requires quite a few steps (all of which require disk writes), because all the following things must take place when a file is saved:

Based on the size of the file, the operating system must scan the FAT for free clusters that can be assigned to the file.

Both copies of the FAT must have the new linked list for the file written to them.

The directory that contains the file must have its entry for the file created or modified.

Finally, the file's contents are saved.

When you realize all the work that has to be done to open, read, and write files, it's a wonder that it doesn't take more time to do it all!

Understanding FAT32

FAT32 works substantially the same way as FAT16, and Microsoft worked to keep differences minimal to reduce compatibility problems. However, there are some key differences:

Each FAT entry now consumes 4 bytes rather than 2 bytes (32 bits versus 16 bits). This change plus the fact that there may be many more FAT entries means that the FAT itself is much larger under FAT32.

The MBR is expanded from one to two sectors to allow for a larger BIOS Parameter Block, as well as duplicate copies of the boot record (providing additional redundancy over FAT16).

The root directory is no longer located in a fixed position immediately following the two copies of the FAT, as is the case with FAT16. Instead, the root directory is now treated the same as any other directory and can be placed anywhere on the volume. Because of this, the root directory is no longer limited to 512 entries.

Directory entries themselves are unchanged, except that the 2 bytes unused under MS-DOS and Windows (all versions) that were set aside in the directory entry and intended for OS/2 Extended Attributes are now used to store the extra 16 bits required to index to a full 32-bit cluster number.

The minimum cluster size under FAT32 is 4 K instead of the 512-byte clusters supported for very small FAT16 drives.

All these changes, as well as some other optimizations, offer these advantages over FAT16:

Single partitions can now extend to 2 terabytes (1024 G).

On larger partitions, such as those up to 8 G, cluster sizes of only 4 K can be used, instead of the 128 K theoretical cluster size that FAT16 would have to use--were it even possible for FAT16 to support 8 G partitions. Overall, because the clusters under FAT32 are generally smaller than those in FAT16, about 15 percent more disk space is available because of reduced slack space in the clusters.

FAT32 is more reliable than FAT16, and the possible points of failure have been reduced, partly by keeping a redundant copy of the boot record on FAT32 drives.

Because FAT32 partitions generally use 4 K clusters (which also happens to be the size of the pages of memory used by the virtual memory manager), paging activity is faster and requires less overhead.

Applications launch faster from FAT32 drives.

There are also some drawbacks to consider when thinking about using FAT32:

FAT32 is not yet supported by Windows NT (up to version 4), so you cannot convert the primary partition if you want to dual-boot Windows 98 and Windows NT. Logical drives formatted with FAT32 are ignored by Windows NT 4 (it is not yet clear whether Windows NT 5 will support FAT32 drives, although it seems unthinkable that it would not).

You cannot reverse the Windows 98 conversion process that easily upgrades FAT16 drives to FAT32. Instead, to return a drive to FAT16 from FAT32, you must repartition and reformat the drive.

All utilities that rely on the FAT16 file system do not work with FAT32, including disk sector editors, older defragmentation programs, and all (at the time of this writing) disk compression programs (including the DriveSpace program that comes with Windows 98). However, updated versions should be available shortly after Windows 98 is available.

If you convert the primary partition to FAT32, you will be unable to access the drive by booting MS-DOS (such as from an MS-DOS bootable floppy disk).

Understanding Long Filenames on FAT

As is probably clear from the preceding discussions about FAT16 and FAT32, there is no room set aside in either file system to store more than the 8.3 filenames supported under DOS. And yet Windows 95 and Windows 98 (not to mention Windows NT) support long filenames on such FAT volumes. It is possible as the individual directory entries have a number of attribute bits, one of which is set if the entry indicates a volume label. In earlier MS-DOS versions, there was usually only a single such entry, located in the root directory, that stored the volume label for the drive. MS-DOS ignores additional directory entries that have this attribute set, as do most utility programs.

Microsoft developed a scheme whereby long filenames could be supported on FAT volumes by using this volume label directory attribute. When you create a long filename under Windows 98, such as MYLONGFILE.TXT, multiple directory entries may be created, with the second and successive entries being flagged with the volume label attribute (so that MS-DOS and other utilities ignore those entries). Windows 98 stores pieces of each long filename in each of these entries, with up to 11 characters stored per entry. In the case of MYLONGFILE.TXT, there are two directory entries: the first is stored as MYLONG~1.TXT, while the second entry stores the remainder of the full filename and is not individually viewable, except with a disk sector editor.

Long filenames under Windows 98 must follow these rules:

They can have multiple spaces and multiple periods.

The short portion of the filename can use only the following special characters: $ % ` - _ @ ~ ´ ! ( ) ^ # &

Long filenames can additionally use these special characters: + , ; = [ ]

Maximum long filename length is 254 characters, including the extension.

The maximum length of a fully qualified pathname under Windows 98 is 260 characters. In other words, if you have a filename that is 254 characters long, it can have a directory name associated with it that is only 4 characters long.

If you copy or move files with a utility that doesn't support long filenames, or copy or move files to a file server that doesn't support long filenames, the long filenames are lost.

When using a long filename at the MS-DOS prompt, if the filename includes spaces, you must surround the filename with quotation marks (").

One problem of which you should be aware occurs at the MS-DOS command prompt. The command-line buffer at the command prompt is limited to 128 characters, by default. Because of that, you can't use MS-DOS commands for long filenames if the command plus the filename exceeds 128 characters. You can expand the MS-DOS command-line buffer to help address this problem by placing the command **SHELL=C:\WINDOWS\COMMAND.COM /U:255 /P** in the CONFIG.SYS file (255 is the absolute maximum value).

Understanding NTFS

New Technology File System (NTFS) was designed to be an "industrial strength" file system, appropriate for use on file servers and other high-end systems. NTFS includes the following features:

It is transaction oriented, and failures to complete a disk transaction can be reversed during a system CHKDSK, safely and easily.

It supports *hot-fixing*, in which data on marginal areas of the disk are automatically moved to another part of the disk, and the questionable part of the disk is marked so that it is not used again.

It provides full support for long filenames in the file system.

It retains the time of last file access (FAT simply stores the date of last access).

Volumes and files under NTFS can occupy up to 2^{64} bytes (16 *exabytes*).

It makes use of a binary tree-based file table, which can be searched much more quickly than a FAT.

NTFS stores clusters more efficiently than does FAT; the file system works to avoid fragmentation much more effectively than FAT. Moreover, because of the B-tree search tree NTFS uses to find clusters, fragmentation causes less degradation with NTFS than with FAT.

Unlike High Performance File System (HPFS, designed for OS/2), NTFS uses clusters of varying sizes, similar to the FAT scheme. However, cluster sizes under NTFS are much smaller than FAT16 clusters, as follows:

Partition Size	**Cluster Size**
512 M	512 bytes
1024 M	1 K
2048 M	2 K
4096 M	4 K
8192 M	8 K
16384 M	16 K
32768 M	32 K
>32768 M	64 K

NTFS volumes are arranged into the following areas:

Partition Boot Sector

Master File Table

NTFS System Files

File Area

The Master File Table area and NTFS System Files are actually part of the same area on the disk, made up of 16 records (only 11 are used at this time), each one having a filename, as follows:

System File	Filename	Description
Master File Table	$Mft	Contains the contents of the NTFS volume
Master File Table #2	$MftMirr	Duplicate of the first three records of the Master File Table
Log File	$LogFile	Transaction log used to recover disk transactions in case of error (such as a power failure while writing a transaction)
Volume	$Volume	Volume information, such as name, NTFS version information, and so on
Attribute Definition	$AttrDef	A table listing attributes used on the volume
Table		
Root Filename Index	$.	The root directory contents
Cluster Bitmap	$Bitmap	A bitmap showing used and unused clusters on the volume; used for rapidly allocating clusters
Partition Boot Sector	$Boot	Bootstrap program for the volume
Bad Cluster Fil le	$BadClus	List of bad clusters on the volume
Quota Table	$Quota	Stores disk quota information for volume users
Upcase Table	$Upcase	Table used to convert lowercase letters to uppercase letters, using Unicode

Each file on an NTFS system is seen as a set of *file attributes* (which are quite unlike the file attribute flags used under FAT). NTFS can have file attributes for data in the file, for security information, for *file metadata* (such as icons), and even for the name of the file. (These file attributes are similar to the OLE file format in which individual files can be made up of separate streams of data.)

File System in UNIX

The file system is the primary means of file storage in UNIX. Each file system houses directories, which, as a group, can be placed almost anywhere in the UNIX directory tree. The topmost level of the directory tree, the root directory, begins at /. Subdirectories nested below the root directory may traverse as deep as you like so long as the longest absolute path is less than 1,024 characters.

With the proliferation of vendor-enhanced versions of UNIX, you will find a number of "enhanced" file systems :

- SunOS 4.1.x, which uses 4.2
- Solaris, which uses ufs
- Linux, which uses ext2

- IRIX, which uses efs and xfs

Note that the ufs and 4.2 file systems are actually the same.

A file system, however, is only a part of the grand scheme of how UNIX keeps its data on disk. At the top level, you'll find the disks themselves. These disks are then broken into partitions, each varying in size depending on the needs of the administrator. It is on each partition that the actual file system is laid out. Within the file system, you'll find directories, subdirectories, and, finally, the individual files.

Although you will rarely have to deal with the file system at a level lower than the individual files stored on it, it is critical that you understand two key concepts: *inodes* and the *superblock.*

inodes

An inode maintains information about each file. Depending on the type of file system, the inode can contain upwards of 40+ pieces of information. Most of it, however, is only useful to the kernel and doesn't concern us. The fields that do concern us are

mode	The permission mask and type of file.
link count	The number of directories that contain an entry with this inode number.
user ID	The ID of the file's owner.
group ID	The ID of the file's group.
size	Number of bytes in this file.
access time	The time at which the file was last accessed.
mod time	The time at which the file was last modified.
inode time	The time at which this inode structure was last modified.
block list	A list of disk block numbers which contain the first segment of the file.
indirect list	A list of other block lists.

The mode, link count, user ID, group ID, size, and access time are used when generating file listings. Note that the inode does not contain the file's name. That information is held in the directory file (see below for details).

Superblocks

This is the most vital information stored on the disk. It contains information on the disk's geometry (number of heads, cylinders, and so on), the head of the inode list, and free block list. Because of its importance, the system automatically keeps mirrors of this data scattered around the disk for redundancy. You only have to deal with superblocks if your file system becomes heavily corrupted.

Types of Files

Files come in 8 flavors:

- Normal Files
- Directories
- Hard Links
- Symbolic links
- Sockets
- Named Pipes
- Character Devices

- Block Devices

Normal Files These are the files you use the most. They can be either text or binary files; however, their internal structure is irrelevant from a System Administrator standpoint. A file's characteristics are specified by the inode in the file system that describes it.

Directories These are a special kind of file that contains a list of other files. Although there is a one-to-one mapping of inode to disk blocks, there can be a many-to-one mapping from directory entry to inode. When viewing a directory listing using the ls -l command, you can identify directories by their permissions starting with the d character.

Hard Links

A hard link is actually a normal directory entry except instead of pointing to a unique file , it points to an already existing file . This gives the illusion that there are two identical files when you do a directory listing. Because the system sees this as just another file, it treats it as such. This is most apparent during backups because hard-linked files get backed up as many times as there are hard links to them. Because a hard link shares an inode, it cannot exist across file systems. Hard links are created with the ln command.

Symbolic Links

A symbolic link (sometimes referred to as a *symlink*) differs from a hard link because it doesn't point to another inode but to another filename. This allows symbolic links to exist across file systems as well as be recognized as a special file to the operating system. You will find symbolic links to be crucial to the administration of your file systems, especially when trying to give the appearance of a seamless system when there isn't one. Symbolic links are created using the ln -s command. A common thing people do is create a symbolic link to a directory that has moved.

Sockets

Sockets are the means for UNIX to network with other machines. Typically, this is done using network ports; however, the file system has a provision to allow for interprocess communication through socket files. (A popular program that uses this technique is the X Windows system.)

Named Pipes

Similar to sockets, named pipes enable programs to communicate with one another through the file system. You can use the mknod command to create a named pipe. Named pipes are recognizable by their permissions settings beginning with the p character.

Character Devices

These special files are typically found in the /dev directory and provide a mechanism for communicating with system device drivers through the file system one character at a time. They are easily noticed by their permission bits starting with the c character. Each character file contains two special numbers, the major and minor. These two numbers identify which device driver that file communicates with.

Block Devices

Block devices also share many characteristics with character devices in that they exist in the /dev directory, are used to communicate with device drivers, and have major and minor numbers. The key difference is that block devices typically transfer large blocks of data at a time versus one character at a time. (A hard disk is a block device, whereas a terminal is a

character device.) Block devices are identified by their permission bits starting with the b character.

Mounting and Unmounting File Systems

Power in UNIX stems from its flexibility in placing file systems anywhere in the directory tree. This feat is accomplished by mounting file systems.
Before you can mount a file system, you need to select a mount point. A mount point is the directory entry in the file system where the root directory of a different file system will overlay it. UNIX keeps track of mount points, and accesses the correct file system, depending on which directory the user is currently in. A mount point may exist anywhere in the directory tree.

> While it is technically true that you can mount a file system anywhere in the directory tree, there is one place you will NOT want to mount it: the root directory. Remember that once a file system is mounted at a directory, that directory is overshadowed by the contents of the mounted file system. Hence, by mounting on the root directory, the system will no longer be able to see its own kernel or local configuration files. How long your system goes on before crashing depends on your vendor.
>
> There is an exception to the rule. Some installation packages will mount a network file system to the root directory. This is done to give the installation software access to many packages that may not be able to fit on your boot disk. Unless you fully understand how to do this yourself, don't.

Creating File Systems

Now that you understand the nuances of maintaining a file system, it's time to understand how they are created. This section walks you through the three steps of:

- Picking the right kind of disk for your system
- Creating partitions
- Creating the file system

Disk Types

Although there are many different kinds of disks, UNIX systems have come to standardize on SCSI for workstations. Many PCs also sport SCSI interfaces, but because of the lower cost and abundance, you'll find a lot of IDE drives on UNIX PC's as well.
SCSI itself comes in a few different flavors now. There is regular SCSI, SCSI-2, SCSI-Wide, SCSI-Fast and Wide, and now SCSI-3. Although it is possible to mix and match these devices with converter cables, you may find it both easier on your sanity and your performance if you stick to one format. As of this writing, SCSI-2 is the most common interface.
When attaching your SCSI drive, there are many important points to remember.

- Terminate your SCSI chain. Forgetting to do this causes all sorts of non-deterministic behavior (a pain to track down). SCSI-2 requires active termination, which is usually indicated by terminators with LEDs on them.
- If a device claims to be self-terminating, you can take your chances, but you'll be less likely to encounter an error if you put a terminator on anyway.

- There is a limit of eight devices on a SCSI chain with the SCSI card counting as a device. Some systems may have internal SCSI devices, so be sure to check for those.
- Be sure all your devices have unique SCSI IDs. A common symptom of having two devices with the same ID is their tendency to frequently reset the SCSI chain. Of course, many devices simply won't work under those conditions.
- When adding or removing a SCSI disk, be sure to power the system down first. There is power running through the SCSI cables, and failing to shut them down first may lead to problems in the future.

Although SCSI is king of the workstation, PCs have another choice: IDE. IDE tends to be cheaper and more available than SCSI devices with many motherboards offering direct IDE support. The advantage of using this kind of interface is its availability as well as lower cost. They are also simpler and require less configuration on your part.

The down side to IDEs is that their simplicity comes at the cost of configurability and expandability. The IDE chain can only hold two devices, and not all motherboards come with more than one IDE chain. If your CD-ROM is IDE, you only have space for one disk. This is probably okay with a single person workstation, but as you can imagine, it's not going to fly well in a server environment. Another consideration is speed. SCSI was designed with the ability to perform I/O without the aid of the main CPU, which is one of the reasons it costs more. IDE, on the other hand, was designed with cost in mind. This resulted in a simplified controller; hence, the CPU takes the burden for working the drive.

While IDE did manage to simplify the PC arena, it did come with the limitation of being unable to handle disks greater than 540M. Various tricks were devised to circumvent this, however, the clean solution is now predominantly available. Known as EIDE (Enhanced IDE), it is capable of supporting disks up to 8G and can support up to 4 devices on one chain.

In weighing the pros and cons of EIDE versus SCSI in the PC environment, don't forget to think about the cost-to-benefit ratio. Having a high speed SCSI controller in a single person's workstation may not be as necessary as the user is convinced it is. Plus, with disks being released in 2+ gigabyte configurations, there is ample room on the typical IDE disk.

Once you have decided on the disk subsystem to install, read the documentation that came with the machine for instructions on physically attaching the disk to the system.

What Are Partitions and Why Do I Need Them?

Partitions are UNIX's way of dividing the disk into usable pieces. UNIX requires that there be at least one partition; however, you'll find that creating multiple partitions, each with a specific function, is often necessary.

The most visible reason for creating separate partitions is to protect the system from the users. The one required partition mentioned earlier is called the root partition. It is here that critical system software and configuration files (the kernel and mount tables) must reside. This partition must be carefully watched so that it never fills up. If it fills up, your system may not be able to come back up in the event of a system crash. Because the root partition is not meant to hold the users' data, you must create separate partitions for the users' home directories, temporarily files, and so forth. This enables their files to grow without the worry of crowding out the key system files.

Dual boot configurations are becoming another common reason to partition, especially with the ever-growing popularity of Linux. You may find your users wanting to be able to boot

to either Windows or Linux; therefore, you need to keep at least two partitions to enable them to do this.

The last, but certainly not least, reason to partition your disks is the issue of backups. Backup software often works by dumping entire partitions onto tape. By keeping the different types of data on separate partitions, you can be explicit about what gets backed up and what doesn't. For example, daily backup of the system software isn't necessary, but backups of home directories are. By keeping the two on separate partitions, you can be more concise in your selection of what gets backed up and what doesn't.

Another example relates more to company politics. It may be possible that one group does not want their data to be backed up to the same tape as another group's. (Note: common sense doesn't always apply to inter-group politicsÉ) By keeping the two groups on separate partitions, you can exclude one from your normal backups and exclude the others during your special backups.

Which Partitions To Create As I mentioned earlier, the purpose of creating partitions is to separate the users from the system areas. So how many different partitions need to be created? While there is no right answer for every installation, here are some guidelines to take into account.

You always need a root partition. In this partition, you'll have your /bin, /etc, and /sbin directories at the very least. Depending on your version of UNIX, this could require anywhere from 30 to 100 megabytes.

/tmp	The /tmp directory is where your users, as well as programs, store temporarily files. The usage of this directory can quickly get out of hand, especially if you run a quota-based site. By keeping it a separate partition, you do not need to worry about its abuse interfering with the rest of the system. Many operating systems automatically clear the contents of /tmp on boot. Size /tmp to fit your site's needs. If you use quotas, you will want to make it a little larger, whereas sites without quotas may not need as much space.
	Under Solaris, you have another option when setting up /tmp. Using the *tmpfs* filesystem, you can have your swap space and /tmp partition share the same physical location on disk. While it appears to be an interesting idea, you'll quickly find that it isn't a very good solution, especially on a busy system. This is because as more users do their work, more of /tmp will be used. Of course, if there are more users, there is a greater memory requirement to hold them all. The competition for free space can become very problematic.
/var	The /var directory is where the system places its spool files (print spool, incoming/outgoing mail queue, and so on) as well as system log files. Because of this, these files constantly grow and shrink with no warning. Especially the mail spool. Another possibility to keep in mind is the creation of a separate partition just for mail. This enables you to export the mail spool to all of your machines without having to worry about your print spools being exported as well. If you use a backup package that requires its own spool space, you may wish to keep this a separate partition as well.
/home	The /home directory is where you place your users' account directories. You may need to use multiple partitions to keep your home directories (possibly broken up by department) and have each partition mount to /home/*dept* where *dept* is the name of the respective department.
/usr	The /usr directory holds noncritical system software, such as editors and lesser used utilities. Many sites hold locally compiled software in the /usr/local directory where they either export it to other machines, or mount other machines' /usr/local to their own. This makes it easy for

	a site to maintain one /usr/local directory and share it amongst all of its machines. Keeping this a separate partition is a good idea since local software inevitably grows.
swap	This isn't a partition you actually keep files on, but it is key to your system's performance. The swap partition should be allocated and swapped to instead of using swap files on your normal file system. This enables you to contain all of your swap space in one area that is out of your way. A good guideline for determining how much swap space to use is to double the amount of RAM installed on your system.

TIP: Several new versions of UNIX are now placing locally compiled software in the /opt directory. Like /usr/local, this should be made a separate partition as well. If your system does not use /opt by default, you should make a symbolic link from there to /usr/local. The vice versa is true as well, if your system uses /opt, you should create a symbolic link from /usr/local to /opt.

To add to the confusion, the Redhat Distribution of Linux has brought the practice of installing precompiled software (RPMs) in the /usr/bin directory. If you are using Redhat, you may want to make your /usr directory larger since locally installed packages will consume that partition.

The Device Entry

Most implementations of UNIX automatically create the correct device entry when you boot it with the new drive attached. Once this entry has been created, you should check it for permissions. Only root should be given read/write access to it. If your backups run as a nonroot user, you may need to give group read access to the backup group. Be sure that no one else is in the backup group. Allowing world read/write access to the disk is the easiest way to have your system hacked, destroyed, or both.

Device entries under Linux IDE disks under Linux use the following scheme to name the hard disks:

/dev/hd[drive][partition]

Each IDE drive is lettered starting from a. So the primary disk on the first chain is a; the slave on the first chain is b; the primary on the secondary chain is c; and so on. Each disk's partition is referenced by number. For example, the third partition of the slave drive on the first chain is /dev/hdb3.

SCSI disks use the same scheme except instead of using /dev/hd as the prefix, /dev/sd is used. So to refer to the second partition of the first disk on the SCSI chain, you would use /dev/sda2.

To refer to the entire disk, specify all the information except the partition. For example, to refer to the entire primary disk on the first IDE chain, you would use /dev/hda.

Device entries under IRIX SCSI disks under IRIX are referenced in either the /dev/dsk or /dev/rdsk directories. The following is the format:

/dev/[r]dsk/dksCdSP

where C is the controller number, S is the SCSI address, and P is the partition, s0,s1,s2, and so on. The partition name can also be vh for the volume header or vol to refer to the entire disk.

Device entries under Solaris The SCSI disks under Solaris are referenced in either the /dev/dsk or /dev/rdsk directories. The following is the format:
/dev/[r]dsk/cCtSd0sP
where C is the controller number, S is the SCSI address, and P is the partition number. Partition 2 always refers to the entire disk and label information. Partition 1 is typically used for swap.
Device entries under SunOS Disks under SunOS are referenced in the /dev directory. The following is the format:
/dev/sdTP
where T is the target number and P is the partition. Typically, the root partition is a, the swap partition is b, and the entire disk is referred to as partition c. You can have partitions from a through f.
An important aspect to note is an oddity with the SCSI target and unit numbering: Devices that are target three need to be called target zero, and devices that are target zero need to be called target three.

A Note About Formatting Disks

"Back in the old days," disks needed to be formatted and checked for bad blocks. The procedure of formatting entailed writing the head, track, and sector numbers in a sector preamble and a checksum in the postamble to every sector on the disk. At the same time, any sectors that were unusable due to flaws in the disk surface were marked and, depending on the type of disk, an alternate sector mapped into its place.
Thankfully, we have moved on.
Both SCSI and IDE disks now come pre-formatted from the factory. Even better, they transparently handle bad blocks on the disk and remap them without any assistance from the operating system.

> **CAUTION:** You should NEVER attempt to low level format an IDE disk. Doing so will make your day very bad as you watch the drive quietly kill itself. Be prepared to throw the disk away should you feel the need to low level format it.

Partitioning Disks and Creating File Systems

In this section, we will cover the step by step procedure for partitioning disks under Linux, IRIX, SunOS, and Solaris. Since the principles are similar across all platforms, each platform will also cover another method of determining how a disk should be partitioned up depending on its intended usage.
Linux To demonstrate how partitions are created under Linux, we will setup a disk with a single user workstation in mind. It will need not only space for system software, but for application software and the user's home directories.
Creating Partitions For this example, we'll create the partitions on a 1.6 GB IDE disk located on /dev/hda. This disk will become the boot device for a single user workstation. We will create the boot /usr, /var, /tmp, /home, and swap partitions.
During the actual partitioning, we don't name the partitions. Where the partitions are mounted is specified with the /etc/fstab file. Should we choose to mount them in different locations later on, we could very well do that. However, by keeping the function of each partition in mind, we have a better idea of how to size them.

A key thing to remember with the Linux fdisk command is that it does not commit any changes made to the partition table to disk until you explicitly do so with the w command.
With the drive installed, we begin by running the fdisk command:

```
# fdisk /dev/hda
```

This brings us to the fdisk command prompt. We start by using the p command to print what partitions are currently on the disk.

```
   Command (m for help): p

   Disk /dev/hda: 64 heads, 63 sectors, 786 cylinders
   Units = cylinders of 4032 * 512 bytes

      Device Boot  Begin   Start     End  Blocks   Id  System

   Command (m for help):
```

We see that there are no partitions on the disk. With 1.6 GB of space, we can be very liberal with allocating space to each partition. Keeping this policy in mind, we begin creating our partitions with the n command:

```
   Command (m for help): n
    e   extended
    p   primary partition (1-4)
   p
   Partition number (1-4): 1
   First cylinder (1-786): 1
   Last cylinder or +size or +sizeM or +sizeK ([1]-786): +50M
   Command (m for help):
```

The 50 MB partition we just created becomes our root partition. Because it is the first partition, it is referred to as /dev/hda1. Using the p command, we see our new partition:

```
   Command (m for help): p

   Disk /dev/hda: 64 heads, 63 sectors, 786 cylinders
   Units = cylinders of 4032 * 512 bytes

      Device Boot  Begin   Start     End  Blocks   Id  System
   /dev/hda1           1       1      26   52384+  83  Linux native

   Command (m for help):
```

With the root partition out of the way, we will create the swap partition. Our sample machine has 32 MB of RAM and will be running X-Windows along with a host of development tools. It is unlikely that the machine will get a memory upgrade for a while, so we'll allocate 64 MB to swap.

```
   Command (m for help): n
   Command action
      e   extended
      p   primary partition (1-4)
   p
   Partition number (1-4): 2
   First cylinder (27-786): 27
```

Last cylinder or +size or +sizeM or +sizeK ([27]-786): +64M

Command (m for help):

Because this partition is going to be tagged as swap, we need to change its file system type to swap using the t command.

Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): 82
Changed system type of partition 2 to 82 (Linux swap)

Command (m for help):

Because of the nature of the user, we know that there will be a lot of local software installed on this machine. With that in mind, we'll create /usr with 500 MB of space.

Command (m for help): n
Command action
e extended
p primary partition (1-4)
p
Partition number (1-4): 3
First cylinder (60-786): 60
Last cylinder or +size or +sizeM or +sizeK ([60]-786): +500M

If you've been keeping your eyes open, you've noticed that we can only have one more primary partition to use, but we want to have /home, /var, and /tmp to be in separate partitions. How do we do this?

Extended partitions.

The remainder of the disk is created as an extended partition. Within this partition, we can create more partitions for use. Let's create this extended partition:

Command (m for help): n
Command action
e extended
p primary partition (1-4)
e
Partition number (1-4): 4
First cylinder (314-786): 314
Last cylinder or +size or +sizeM or +sizeK ([314]-786): 786

Command (m for help):

We can now create /home inside the extended partition. Our user is going to need a lot of space, so we'll create a 500 MB partition. Notice that we are no longer asked whether we want a primary or extended partition.

Command (m for help): n
First cylinder (314-786): 314
Last cylinder or +size or +sizeM or +sizeK ([314]-786): +500M

Command (m for help):

Using the same pattern, we create a 250 MB /tmp and a 180 MB /var partition.

Command (m for help): n

First cylinder (568-786): 568
Last cylinder or +size or +sizeM or +sizeK ([568]-786): +250M

Command (m for help): n
First cylinder (695-786): 695
Last cylinder or +size or +sizeM or +sizeK ([695]-786): 786

Command (m for help):

Notice on the last partition we created that I did not specify a size, but instead specified the last track. This is to ensure that all of the disk is used.

Using the p command, we look at our final work:

Command (m for help): p

Disk /dev/hda: 64 heads, 63 sectors, 786 cylinders
Units = cylinders of 4032 * 512 bytes

Device	Boot	Begin	Start	End	Blocks	Id	System
/dev/hda1		1	1	26	52384+	83	Linux native
/dev/hda2		27	27	59	66528	82	Linux swap
/dev/hda3		60	60	313	512064	83	Linux native
/dev/hda4		314	314	786	953568	5	Extended
/dev/hda5		314	314	567	512032+	83	Linux native
/dev/hda6		568	568	694	256000+	83	Linux native
/dev/hda7		695	695	786	185440+	83	Linux native

Command (m for help):

Everything looks good. To commit this configuration to disk, we use the w command:

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
(Reboot to ensure the partition table has been updated.)
Syncing disks.

Reboot the machine to ensure that the partition has been updated and you're done creating the partitions.

Creating File Systems in Linux Creating a partition alone isn't very useful. In order to make it useful, we need to make a file system on top of it. Under Linux, this is done using the mke2fs command and the mkswap command.

To create the file system on the root partition, we use the following commands:

mke2fs /dev/hda1

The program only takes a few seconds to run and generates output similar to this:

mke2fs 0.5b, 14-Feb-95 for EXT2 FS 0.5a, 95/03/19
128016 inodes, 512032 blocks
25601 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)

63 block groups
8192 blocks per group, 8192 fragments per group
2032 inodes per group
Superblock backups stored on blocks:
8193,16385,24577,32769,40961,49153,57345,65537,73729,
81921,90113,98305,106497,114689,122881,131073,139265,147457,
155649,163841,172033,180225,188417,196609,204801,212993,221185,
229377,237569,245761,253953,262145,270337,278529,286721,294913,
303105,311297,319489,327681,335873,344065,352257,360449,368641,
376833,385025,393217,401409,409601,417793,425985,434177,442369,
450561,458753,466945,475137,483329,491521,499713,507905

Writing inode tables: done
Writing superblocks and file system accounting information: done
You should make a note of these superblock backups and keep them in a safe place. Should the day arise that you need to use fsck to fix a superblock gone bad, you will want to know where the backups are.
Simply do this for all of the partitions, except for the swap partition.
To create the swap file system, you need to use the mkswap command like this:
mkswap /dev/hda2
Replace /dev/hda2 with the partition you chose to make your swap space.
The result of the command will be similar to:
Setting up swapspace, size = 35090432 bytes
And the swap space is ready.
To make the root file system bootable, you need to install the lilo boot manager. This is part of all the standard Linux distributions, so you shouldn't need to hunt for it on the Internet.
Simply modify the /etc/lilo.conf file so that /dev/hda1 is set to be the boot disk and run:
lilo
The resulting output should look something like:
Added linux *
where linux is the name of the kernel to boot, as specified by the name= field in /etc/lilo.conf.
SunOS In this example, we will be preparting a Seagate ST32550N as an auxiliary disk to an existing system. The disk will be divided into three partitions: one for use as a mail spool, one for use as a /usr/local, and the third as an additional swap partition.
Creating the partitions

CAUTION: The procedure for formatting disks is not the same for SunOS and Solaris. Read each section to note the differences.

Once a disk has been attached to the machine, you should verify its connection and SCSI address by running the probe-scsi command from the PROM monitor if the disk is attached to the internal chain, or the probe-scsi-all command to see all the SCSI devices on the system. When you are sure the drive is properly attached and verified to be functioning, you're ready to start accessing the drive from the OS.

After the machine has booted, run the dmesg command to collect the system diagnostic messages. You may want to pipe the output to grep so that you can easily find the information on disks. For example:

dmesg | grep sd

On our system this generated the following output:

```
sd0: <SUN0207 cyl 1254 alt 2 hd 9 sec 36>
sd1 at esp0 target 1 lun 0
sd1:    corrupt label - wrong magic number
sd1: Vendor 'SEAGATE', product 'ST32550N', 4194058 512 byte blocks
root on sd0a fstype 4.2
swap on sd0b fstype spec size 32724K
dump on sd0b fstype spec size 32712K
```

This result tells us that we have an installed disk on sd0 that the system is aware of and using. The information from the sd1 device is telling us that it found a disk, but it isn't usable because of a corrupt label. Don't worry about the error. Until we partition the disk and create file systems on it, the system doesn't know what to do with it, hence the error.

If you are using SCSI address 0 or 3, remember the oddity we mentioned earlier where device 0 needs to be referenced as 3 and device 3 needs to be referenced as 0.

Even though we do not have to actually format the disk, we do need to use the format program that come with SunOS because it also creates the partitions and writes the label to the disk.

To invoke the format program, simply run:

format sd1

where sd1 is the name of the disk we are going to partition.

The format program displays the following menu:

```
FORMAT MENU:
        disk       - select a disk
        type       - select (define) a disk type
        partition  - select (define) a partition table
        current    - describe the current disk
        format     - format and analyze the disk
        repair     - repair a defective sector
        show       - translate a disk address
        label      - write label to the disk
        analyze    - surface analysis
        defect     - defect list management
        backup     - search for backup labels
        quit
format>
```

We need to enter **type** at the format> prompt so that we can tell SunOS the kind of disk we have. The resulting menu looks something like:

```
AVAILABLE DRIVE TYPES:
        0. Quantum ProDrive 80S
        1. Quantum ProDrive 105S
        2. CDC Wren IV 94171-344
        3. SUN0104
      ...
```

13. other

Specify disk type (enter its number):

Because we are adding a disk this machine has not seen before, we need to select option 13, other. This begins a series of prompts requesting the disk's geometry. Be sure to have this information from the manufacturer before starting this procedure.

The first question, Enter number of data cylinders: is actually a three-part question. After you enter the number of data cylinders, the program asks for the number of alternative cylinders and then the number of physical cylinders. The number of physical cylinders is the number your manufacturer provided you. Subtract two from there to get the number of data cylinders, and then just use the default value of 2 for the number of alternate cylinders.

Note that even though our sample drive actually rotates at 7200 rpm, we stick with the default of 3600 rpm because the software will not accept entering a higher speed. Thankfully, this doesn't matter because the operating system doesn't use the information.

Even though format reported that the disk was formatted, it really wasn't. It only acquired information needed to later write the label.

Now we are ready to begin preparations to partition the disk.

These preparations entail computing the amount each cylinder holds and then approximating the number of cylinders we want in each partition.

With our sample disk, we know that each cylinder is composed of 108 sectors on a track, with 11 tracks composing the cylinder.

From the information we saw in dmesg, we know that each block is 512 bytes long. Hence, if we want our mail partition to be 1 GB in size, we perform the following math to compute the necessary blocks:

1 gigabyte = 1048576 kilobytes

One cylinder = 108 sectors * 11 heads = 1188 blocks

1188 blocks = 594 kilobytes

1048576 / 594 = 1765 cylinders

1765 * 1188 = 2096820 blocks

Obviously, there are some rounding errors since the exact one GB mark occurs in the middle of a cylinder and we need to keep each partition on a cylinder boundary. 1,765 cylinders is more than close enough. The 1,765 cylinders translates to 2,096,820 blocks.

The new swap partition we want to make needs to be 64 MB in size. Using the same math as before, we find that our swap needs to be 130,680 blocks long. The last partition on the disk needs to fill the remainder of the disk. Knowing that we have a 2 GB disk, a 1 GB mail spool, and a 64 MB swap partition, this should leave us with about 960 MB for /usr/local.

Armed with this information, we are ready to tackle the partitioning. From the format> prompt, type **partition** to start the partitioning menu. The resulting screen looks something like this:

```
format> partition
PARTITION MENU:
      a      - change 'a' partition
      b      - change 'b' partition
      c      - change 'c' partition
      d      - change 'd' partition
      e      - change 'e' partition
      f      - change 'f' partition
      g      - change 'g' partition
```

```
        h      - change 'h' partition
        select - select a predefined table
        name   - name the current table
        print  - display the current table
        label  - write partition map and label to the disk
        quit
    partition>
```

To create our mail partition, we begin by changing partition a. At the partition> prompt, type **a**.

```
partition> a
```

This brings up a prompt for entering the starting cylinder and the number of blocks to allocate. Because this is going to be the first partition on the disk, we start at cylinder 0. Based on the math we did earlier, we know that we need 2,096,820 blocks.

```
        partition a - starting cyl      0, # blocks        0 (0/0/0)

    Enter new starting cyl [0]: 0
    Enter new # blocks [0, 0/0/0]: 2096820
    partition>
```

Now we want to create the b partition, which is traditionally used for swap space. We know how many blocks to use based on our calculations, but we don't know which cylinder to start from.

To solve this, we simply display the current partition information for the entire disk using the p command:

```
partition> p
Current partition table (unnamed):
    partition a - starting cyl      0, # blocks  2096820 (1765/0/0)
    partition b - starting cyl      0, # blocks        0 (0/0/0)
    partition c - starting cyl      0, # blocks        0 (0/0/0)
    partition d - starting cyl      0, # blocks        0 (0/0/0)
    partition e - starting cyl      0, # blocks        0 (0/0/0)
    partition f - starting cyl      0, # blocks        0 (0/0/0)
    partition g - starting cyl      0, # blocks        0 (0/0/0)
    partition h - starting cyl      0, # blocks        0 (0/0/0)

partition>
```

We can see that partition a is allocated with 2,096,820 blocks and is 1,765 cylinders long. Because we don't want to waste space on the disk, we start the swap partition on cylinder 1765.

(Remember to count from zero!)

```
partition> b

    partition b - starting cyl      0, # blocks        0 (0/0/0)

Enter new starting cyl [0]: 1765
Enter new # blocks [0, 0/0/0]: 130680
partition>
```

Before we create our last partition, we need to take care of some tradition first, namely partition c. This is usually the partition that spans the entire disk. Before creating this partition, we need to do a little math.
108 cylinders x 11 heads x 3508 data cylinders = 4167504 blocks
Notice that the number of blocks we compute here does not match the number actually on the disk. This number was computed based on the information we entered when giving the disk type information.
It is important that we remain consistent.
Since the c partition spans the entire disk, we specify the starting cylinder as 0. Creating this partition should look something like this:
partition> c

```
partition c - starting cyl      0, # blocks        0 (0/0/0)

Enter new starting cyl [0]: 0
Enter new # blocks [0, 0/0/0]: 4167504
partition>
```

We have only one partition left to create: /usr/local. Because we want to fill the remainder of the disk, we need to do one last bit of math to compute how many blocks are still free.
This is done by taking the size of partition c (the total disk) and subtracting the sizes of the existing partitions. For our example, this works out to be:
4167504 - 2096820 - 130680 = 1940004 remaining blocks
Now we need to find out which cylinder to start from.
To do so, we run the p command again:
partition> p
Current partition table (unnamed):

```
partition a - starting cyl      0, # blocks  2096820 (1765/0/0)
partition b - starting cyl   1765, # blocks   130680 (110/0/0)
partition c - starting cyl      0, # blocks  4167504 (3508/0/0)
partition d - starting cyl      0, # blocks        0 (0/0/0)
partition e - starting cyl      0, # blocks        0 (0/0/0)
partition f - starting cyl      0, # blocks        0 (0/0/0)
partition g - starting cyl      0, # blocks        0 (0/0/0)
partition h - starting cyl      0, # blocks        0 (0/0/0)

partition>
```

To figure out which cylinder to start from, we add the number of cylinders used so far. Remember not to add the cylinders from partition c since it encompasses the entire disk.
1765 + 110 = 1875
Now that we know which cylinder to start from and how many blocks to make it, we create our last partition.
partition> d

```
partition d - starting cyl      0, # blocks        0 (0/0/0)

Enter new starting cyl [0]: 1875
Enter new # blocks [0, 0/0/0]: 1940004
```

partition
Congratulations! You've made it through the ugly part. Before we can truly claim victory, we need to commit these changes to disk using the label command. When given the prompt, Ready to label disk, continue? simply answer y.
partition> label
Ready to label disk, continue? y

partition
To leave the format program, type **quit** at the partition> prompt, and then **quit** again at the format> prompt.
Creating File Systems Now comes the easy part. Simply run the newfs command on all the partitions we created except for the swap partition and the entire disk partition.
Be sure to note the superblock backups. This is critical information when fsck discovers heavy corruption in your file system. Remember to add your new entries into /etc/fstab if you want them to automatically mount on boot.
If you created the first partition with the intention of making it bootable, you have a few more steps to go. First, mount the new file system to /mnt.
mount /dev/sd1a /mnt
Once the file system is mounted, you need to clone your existing boot partition using the dump command like this:
cd /mnt
dump 0f - / | restore -rf -
With the root partition cloned, use the installboot command to make it bootable:
/usr/kvm/mdec/installboot /mnt/boot /usr/kvm/mdec/bootsd /dev/rsd1a
Be sure to test your work by rebooting and making sure everything mounts correctly. If you created a bootable partition, be sure you can boot from it now. Don't wait for a disaster to find out whether or not you did it right.
Solaris For this example, we are partitioning a disk that is destined to be a web server for an intranet. We need a minimal root partition, adequate swap, tmp, var, and usr space, and a really large partition, which we'll call /web. Because the web logs will remain on the /web partition, and there will be little or no user activity on the machine, /var and /tmp will be set to smaller values. /usr will be a little larger because it may be destined to house web development tools.
Creating partitions

> **TIP:** In another wondrous effort on its part to be just a little different, Sun has decided to call partitions "slices." With the number of documents regarding the file system so vast, you'll find that not all of them have been updated to use this new term, so don't be confused by the mix of "slices" with "partitions"--they are both the same.

Once a disk has been attached to the machine, you should verify its connection and SCSI address by running the probe-scsi command from the PROM monitor if the disk is attached to the internal SCSI chain, probe-scsi-all to list all the SCSI devices on the system Once this shows that the drive is properly attached and verified to be functioning, you're ready to start accessing the drive from the OS. Boot the machine and login as root.

In order to find the device name, we are going to use for this, we again use the dmesg command.

```
# dmesg | grep sd
   ...
   sd1 at esp0: target 1 lun 0
   sd1 is /sbus@1,f8000000/esp@0,800000/sd@1,0
   WARNING: /sbus@1,f8000000/esp@0,800000/sd@1,0 (sd1):
     corrupt label - wrong magic number
     Vendor 'SEAGATE', product 'ST32550N', 4194058 512 byte blocks
   ...
```

From this message, we see that our new disk is device /dev/[r]dsk/c0t1d0s2. The disk hasn't been set up for use on a Solaris machine before, which is why we received the corrupt label error.

If you recall the layout of Solaris device names, you'll remember that the last digit on the device name is the partition number. Noting that, we see that Solaris refers to the entire disk in partition 2, much the same way SunOS refers to the entire disk as partition c.

Before we can actually label and partition the disk, we need to create the device files. This is done with the drvconfig and disks commands. They should be invoked with no parameters:

```
# drvconfig ; disks
```

Now that the kernel is aware of the disk, we are ready to run the format command to partition the disk.

```
# format /dev/rdsk/c0t1d0s2
```

This brings up the format menu as follows:

```
FORMAT MENU:
        disk       - select a disk
        type       - select (define) a disk type
        partition  - select (define) a partition table
        current    - describe the current disk
        format     - format and analyze the disk
        repair     - repair a defective sector
        label      - write label to the disk
        analyze    - surface analysis
        defect     - defect list management
        backup     - search for backup labels
        verify     - read and display labels
        save       - save new disk/partition definitions
        inquiry    - show vendor, product and revision
        volname    - set 8-character volume name
        quit
format>
```

To help the format command with partitioning, we need to tell it the disk's geometry by invoking the type command at the format> prompt. We will then be asked to select what kind of disk we have. Because this is the first time this system is seeing this disk, we need to select other. This should look something like this:

```
format> type
```

AVAILABLE DRIVE TYPES:

0. Auto configure
1. Quantum ProDrive 80S
2. Quantum ProDrive 105S
3. CDC Wren IV 94171-344

. . .

16. other

Specify disk type (enter its number): 16

The system now prompts for the number of data cylinders. This is two less than the number of cylinders the vendor specifies because Solaris needs two cylinders for bad block mapping.

Enter number of data cylinders: 3508
Enter number of alternate cylinders[2]: 2
Enter number of physical cylinders[3510]: 3510

The next question can be answered from the vendor specs as well.

Enter number of heads: 14

The followup question about drive heads can be left as default.

Enter physical number of heads[default]:

The last question you must answer can be pulled from the vendor specs as well.

Enter number of data sectors/track: 72

The remaining questions should be left as default.

Enter number of physical sectors/track[default]:
Enter rpm of drive[3600]:
Enter format time[default]:
Enter cylinder skew[default]:
Enter track skew[default]:
Enter tracks per zone[default]:
Enter alternate tracks[default]:
Enter alternate sectors[default]:
Enter cache control[default]:
Enter prefetch threshold[default]:
Enter minimum prefetch[default]:
Enter maximum prefetch[default]:

The last question you must answer about the disk is its label information. Enter the vendor name and model number in double quotes for this question. For our sample disk, this would be:

Enter disk type name (remember quotes): "SEAGATE ST32550N"

With this information, Solaris makes creating partitions easy. Dare I say, fun?

After the last question from the type command, you will be placed at the format> prompt. Enter **partition** to start the partition menu.

format> partition

PARTITION MENU:

0 - change '0' partition
1 - change '1' partition
2 - change '2' partition
3 - change '3' partition

```
        4      - change '4' partition
        5      - change '5' partition
        6      - change '6' partition
        7      - change '7' partition
        select - select a predefined table
        modify - modify a predefined partition table
        name   - name the current table
        print  - display the current table
        label  - write partition map and label to the disk
        quit
partition>
```

At the partition> prompt, enter **modify** to begin creating the new partitions. This brings up a question about what template to use for partitioning. We want the All Free Hog method.

```
partition> modify
Select partitioning base:
  0. Current partition table (unnamed)
  1. All Free Hog
Choose base (enter number)[0]? 1
```

The All Free Hog method enables you to select one partition to receive the remainder of the disk once you have allocated a specific amount of space for the other partitions. For our example, the disk hog would be the /web partition because you want it to be as large as possible.

As soon as you select option 1, you should see the following screen:

```
Part    Tag        Flag    Cylinders    Size    Blocks
 0    root        wm       0          0    (0/0/0)
 1    swap        wu       0          0    (0/0/0)
 2    backup      wu    0 - 3507    1.99GB    (3508/0/0)
 3    unassigned  wm       0          0    (0/0/0)
 4    unassigned  wm       0          0    (0/0/0)
 5    unassigned  wm       0          0    (0/0/0)
 6    usr         wm       0          0    (0/0/0)
 7    unassigned  wm       0          0    (0/0/0)
Do you wish to continue creating a new partition
table based on above table [yes]? yes
```

Because the partition table appears reasonable, agree to use it as a base for your scheme. You will now be asked which partition should be the Free Hog Partition, the one that receives whatever is left of the disk when everything else has been allocated.

For our scheme, we'll make that partition number 5.

```
Free Hog Partition[6]? 5
```

Answering this question starts the list of questions asking how large to make the other partitions. For our web server, we need a root partition to be about 200 MB for the system software, a swap partition to be 64 MB, a /tmp partition to be 200 MB, a /var partition to be 200 MB, and a /usr partition to be 400 MB. Keeping in mind that partition 2 has already been tagged as the "entire disk" and that partition 5 will receive the remainder of the disk, you will be prompted as follows:

```
Enter size of partition '0' [0b, 0c, 0.00mb]: 200mb
Enter size of partition '1' [0b, 0c, 0.00mb]: 64mb
```

Enter size of partition '3' [0b, 0c, 0.00mb]: 200mb
Enter size of partition '4' [0b, 0c, 0.00mb]: 200mb
Enter size of partition '6' [0b, 0c, 0.00mb]: 400mb
Enter size of partition '7' [0b, 0c, 0.00mb]: 0
As soon as you finish answering these questions, the final view of all the partitions appears looking something like:

Part	Tag	Flag	Cylinders	Size	Blocks
0	root	wm	0 - 344	200.13mb	(345/0/0)
1	swap	wu	345 - 455	64.39mb	(111/0/0)
2	backup	wu	0 - 3507	1.99GB	(3508/0/0)
3	unassigned	wm	456 - 800	200.13mb	(345/0/0)
4	unassigned	wm	801 - 1145	200.13mb	(345/0/0)
5	unassigned	wm	1146 - 2817	969.89mb	(1672/0/0)
6	unassigned	wm	2818 - 3507	400.25mb	(690/0/0)
7	unassigned	wm	0	0	(0/0/0)

This is followed by the question:
Okay to make this the correct partition table [yes]? yes
Answer **yes** since the table appears reasonable. This brings up the question:
Enter table name (remember quotes): "SEAGATE ST32550N"
Answer with a description of the disk you are using for this example. Remember to include the quote symbols when answering. Given all of this information, the system is ready to commit this to disk. As one last check, you will be asked:

Ready to label disk, continue? y
As you might imagine, we answer yes to the question and let it commit the changes to disk. You have now created partitions and can quit the program by entering **quit** at the partition> prompt and again at the format> prompt.
Creating file systems To create a file system, simply run:
newfs /dev/c0t1d0s0
where */dev/c0t1d0s0* is the partition on which to create the file system. Be sure to create a file system on all the partitions except for partitions 2 and 3, the swap, and entire disk, respectively. Be sure to note the backup superblocks that were created. This information is very useful when fsck is attempting to repair a heavily damaged file system.
After you create the file systems, be sure to enter them into the /etc/vfstab file so that they are mounted the next time you reboot.
If you need to make the root partition bootable, you still have two more steps. The first is to clone the root partition from your existing system to the new root partition using:

```
# mount /dev/dsk/c0t1d0s0 /mnt
# ufsdump 0uf - / | ufsrestore -rf -
```

Once the file root partition is cloned, you can run the installboot program like this:
/usr/sbin/installboot /usr/lib/fs/ufs/bootblk /dev/rdsk/c0t1d0s0
Be sure to test your new file systems before you need to rely on them in a disaster situation.
IRIX For this example, we are creating a large scratch partition for a user who does modeling and simulations. Although IRIX has many GUI-based tools to perform these tasks, it is always a good idea to learn the command line versions just in case you need to do any kind of remote administration.

Creating partitions Once the drive is attached, run a program called hinv to take a "hardware inventory." On the sample system, you saw the following output:

```
...
Integral SCSI controller 1: Version WD33C93B, revision D
  Disk drive: unit 6 on SCSI controller 1
Integral SCSI controller 0: Version WD33C93B, revision D
  Disk drive: unit 1 on SCSI controller 0
...
```

Our new disk is external to the system, so we know it is residing on controller 1. Unit 6 is the only disk on that chain, so we know that it is the disk we just added to the system.

To partition the disk, run the fx command without any parameters. It prompts us for the device name, controller, and drive number. Choose the default device name and enter the appropriate information for the other two questions.

On our sample system, this would look like:

```
# fx
fx version 6.2, Mar  9, 1996
fx: "device-name" = (dksc)
fx: ctlr# = (0) 1
fx: drive# = (1) 6
fx: lun# = (0)
...opening dksc(1,6,0)
...controller test...OK
Scsi drive type == SEAGATE ST32550N        0022

----- please choose one (? for help, .. to quit this menu)-----
[exi]t              [d]ebug/            [l]abel/
[b]adblock/         [exe]rcise/         [r]epartition/
fx>
```

We see that fx found our Seagate and is ready to work with it. From the menu we select **r** to repartition the disk. fx displays what it knows about the disk and then presents another menu specifically for partitioning the disk.

```
fx> r
----- partitions-----
part  type      cyls            blocks          Megabytes   (base+size)
  7: xfs        3 + 3521        3570 + 4189990        2 + 2046
  8: volhdr     0 + 3             0 + 3570          0 + 2
 10: volume     0 + 3524          0 + 4193560       0 + 2048

capacity is 4194058 blocks

----- please choose one (? for help, .. to quit this menu)-----
[ro]otdrive         [u]srrootdrive      [o]ptiondrive       [re]size
fx/repartition>
```

Looking at the result, we see that this disk has never been partitioned in IRIX before. Part 7 represents the amount of partitionable space, part 8 the volume header, and part 10 the entire disk.

Because this disk is going to be used as a large scratch partition, we want to select the optiondrive option from the menu. After you select that, you are asked what kind of file system you want to use. IRIX 6 and above defaults to xfs, while IRIX 5 defaults to efs. Use the one appropriate for your version of IRIX.
Our sample system is running IRIX 6.3, so we accept the default of xfs:
fx/repartition> o

fx/repartition/optiondrive: type of data partition = (xfs)
Next we are asked whether we want to create a /usr log partition. Because our primary system already has a /usr partition, we don't need one here. Type **no**.
fx/repartition/optiondrive: create usr log partition? = (yes) no
The system is ready to partition the drive. Before it does, it gives one last warning allowing you to stop the partitioning before it completes the job. Because you know you are partitioning the correct disk, you can give it "the go-ahead":
Warning: you must reinstall all software and restore user data from backups after changing the partition layout. Changing partitions causes all data on the drive to be lost. Be sure you have the drive backed up if it contains any user data. Continue? **y**
The system takes a few seconds to create the new partitions on the disk. Once it is done, it reports what the current partition list looks like.

```
----- partitions-----
part  type      cyls            blocks          Megabytes   (base+size)
  7: xfs        3 + 3521        3570 + 4189990   2 + 2046
  8: volhdr     0 + 3           0 + 3570         0 + 2
 10: volume     0 + 3524        0 + 4193560      0 + 2048

capacity is 4194058 blocks

----- please choose one (? for help, .. to quit this menu)-----
[ro]otdrive     [u]srrootdrive     [o]ptiondrive     [re]size
fx/repartition>
```

Looks good. We can exit fx now by typing **..** at the fx/repartition> prompt and **exit** at the fx> prompt.
Our one large scratch partition is now called /dev/dsk/dks1d6s7.
Creating the filesystem To create the file system, we use the mkfs command like this:
mkfs /dev/rdsk/dks1d6s7
This generates the following output:

```
meta-data=/dev/dsk/dks1d6s7      isize=256    agcount=8, agsize=65469 blks
data     =                       bsize=4096   blocks=523748, imaxpct=25
log      =internal log           bsize=4096   blocks=1000
realtime =none                   bsize=65536  blocks=0, rtextents=0
```

Remember to add this entry into the /etc/fstab file so that the system automatically mounts the next time you reboot.

Summary

As you've seen in this chapter, creating, maintaining, and repairing filesystems is not a trivial task. It is, however, a task which should be well understood. An unmaintained file

system can quickly lead to trouble and without its stability, the remainder of the system is useless.
Let's make a quick rundown of the topics we covered:

- Disks are broken into partitions (sometimes called slices).
- Each partition has a file system.
- A file system is the primary means of file storage in UNIX.
- File systems are made of inodes and superblocks.
- Some partitions are used for raw data such as swap.
- The /proc file system really isn't a file system, but an abstraction to kernel data.
- An inode maintains critical file information.
- Superblocks track disk information as well as the location of the heads of various inode lists.
- In order for you to use a file system, it must be mounted.
- No one must be accessing a file system in order for it to be unmounted.
- File systems can be mounted anywhere in the directory tree.
- /etc/fstab (vfstab in Solaris) is used to by the system to automatically mount file systems on boot.
- The root file system should be kept away from users.
- The root file system should never get filled.
- Be sure to watch how much space is being used.
- fsck is the tool to use to repair file systems.
- Don't forget to terminate your SCSI chain!

•In short, file systems administration is not a trivial task and should not be taken lightly. Good maintenance techniques not only help maintain your uptime, but your sanity as well.

File System Conversion

Reversing FAT32 to FAT16

Since I strongly recommend to stay with FAT32 or NTFS (Windows NT or 2000). Here is summary procedure to convert back to FAT16 :

1. DO a complete Backup.
2. Check again that you have done backup as whole disk is to be wipe out.
3. Create a bootable disk and check that it contains your cd drivers for DOS.
4. Shutdown windows and boot to DOS.
   Start -> shutdown -> Restart in MS DOS ---OR---
   Start -> shutdown -> Restart and press F8 key before you get starting windows 98 message.

Run Fdisk.
Fdisk will prompt that whether you want to enable large disk support. **Select NO 'N'.** Here below is what fdisk asks

1. Delete old fat32 partition.
2. Create a new partition.
3. exit fdisk and reboot.
4. Format the disk with format.com command.
5. Then type **fdisk /mbr . DONE**

To Convert Back from NTFS to FAT32

1. Backup
2. bootable disk for DOS
3. Run Fdisk.
4. Fdisk will prompt that whether you want to enable large disk support. **Select Yes 'Y'.**
5. Delete old fat32 partition.
6. Create a new partition.
7. exit fdisk and reboot.
8. Format the disk with format.com command.
9. Then run **fdisk /mbr enter and DONE.**

From Linux to FAT32

Most user wants to try linux, but for some reason or other want to move back to windows. Then follow this procedure. Although linux enables dual boot between windows and Linux please do read it below.

1. Run FDisk.
2. Delete NON DOS Partiton. Problems read below.
3. Create New Partition. Move to point number
4. Fdisk some times fails to delete and recognize linux partition. You have to use linux partition utility to convert it FAT.
5. to do that Run your linux setup from cd. In middle it will prompt to create partitions. Delete linux based and convert whole to FAT16 one. Don't leave empty space. Convert everything to fat16
6. Write changes to disk.
7. Boot to DOS
8. Run fdisk and create partitions.
9. Format the DISK.
10. Then run **fdisk /mbr** enter and DONE.

Creating Dual boot computer of Windows 95/98 and Linux.

It is very easy. But you will need some disk space. It is plenty now a days. I have whooping 12 GB free on my 20 GB disk. But minimum requirement is that you must have 1.5 GB Free. I assume that you use a 4 gb disk of which about 2.2 GB is for windows and 1.8 GB for linux. Windows require atleast 200 MB free. So think your space requirements accordingly. And the above 1.8 GB is not hard and fat. But linux is comfortable with 1 Gb and above as it also creates a swap partition for itself.

On your linux cd there in DOS folder there is a utility called **FIPS.** It could create partitions from existing ones without any Data damage and keeping previous ones intact. Newer versions (2.0 and later) could even recognize FAT32 systems. But first backup only for our own safety.

Defrag your disk. If you use any utility such as Norton Speeddisk or any other utility then check that you don't have data in end here is sample map of disk. Fips need free space at end to create partition. Some defrag utils place unused or less used data in end of disk. So Defrag with options that no data remains in end. Your disk map should look like this.

Boot to DOS Prompt. Run Fips. It will ask to backup. Well we don't need it. Proceed. If it terminates with error of no free space then defrag as mentioned above. It will do a long check and then you will be prompted to create a partition. Select the partiton from the free space. You can create multiple partitions but i will recommend to create one only. And yes leave atleast 200 MB free space for windows. Fips then create partition and wait it halts your system. That means everything completed successfully. Now reboot.
Now we have to Install linux on this new partition. Our old partition contains windows. And see there is no data damage whats so ever.
Now install Linux. In installation it will prompt for creating partitions. Don't touch the windows parition but create linux partition on our new partition. Linux setup will prompt whether you want to boot to Windows or Linux. Select default to Windows/DOS. This means every time you start your system it will prompt you which os to boot. If you don't select it will boot to windows. Install linux. and done.

Suggested reading :
Read FIPS DOCS Before proceeding
Read Linux Docs before proceedings
I will not be responsible for any data damge. If any thing goes wrong then use Powerquest Lost & Found or Stellar to recover data.

IN THE END - A little about me.

Hi my name is Shrishail Rana.

I am currently doing my post graduation in law and yes computers are my hobby, love or whatever you can say.

If you have any questions please send an email at srana4u@hotmail.com.

Bye for now. Wait and watch for more on my web site http://srana.cjb.net