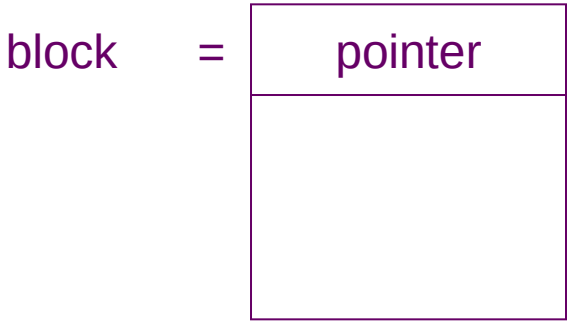
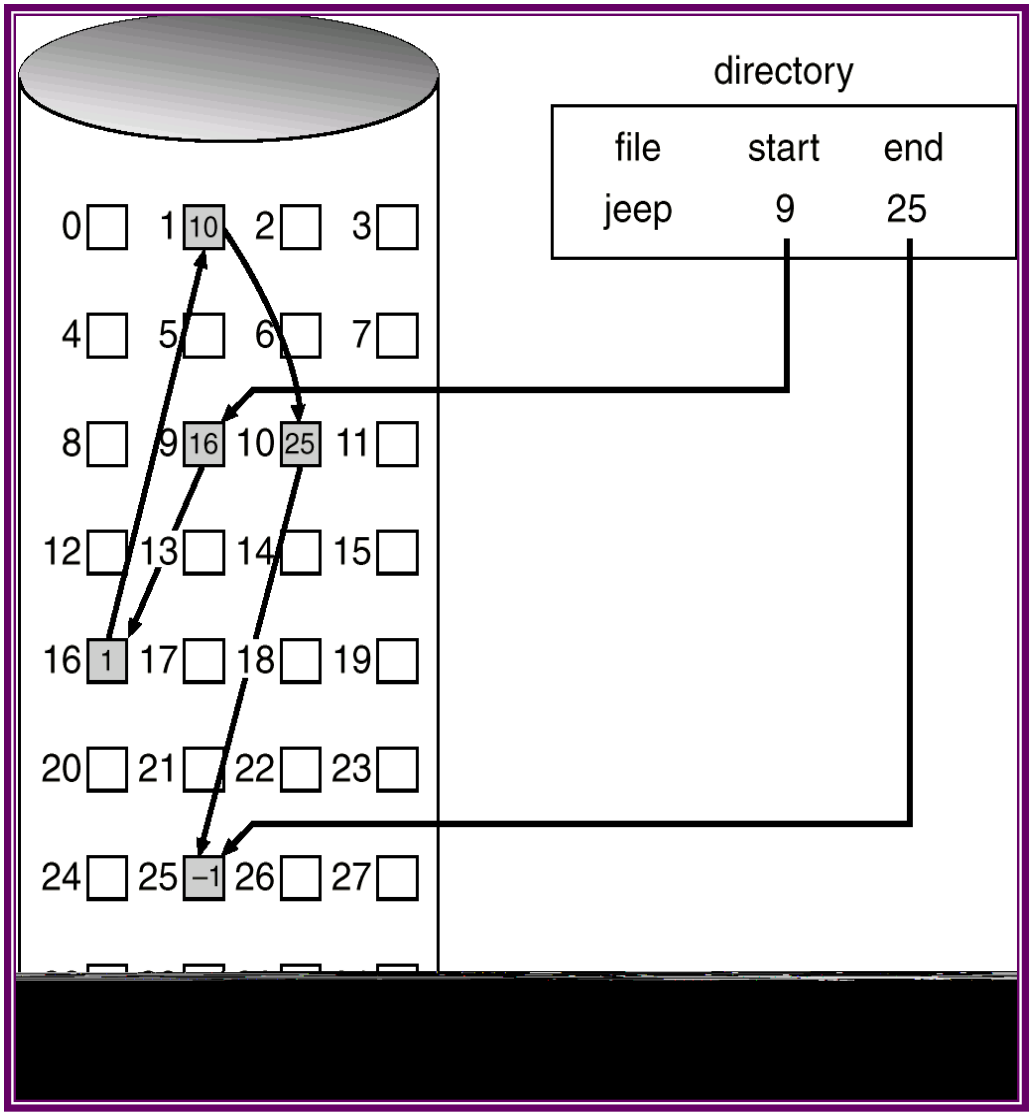
FAT ->Based on Linked Allocation

- Each file is a linked list of disk blocks:
- blocks may be **scattered anywhere** on the disk.



Linked Allocation



FAT

- Modified linked allocation + Continuous allocation

- Modifications:
  - 1. Continuous allocation on the cluster basis
  - 2. Put file pointers into table

FAT

- **First modification:**
  - **extract all file pointers**
  - **create a special table from these pointers**
  - **table called FAT (File Allocation Table)**

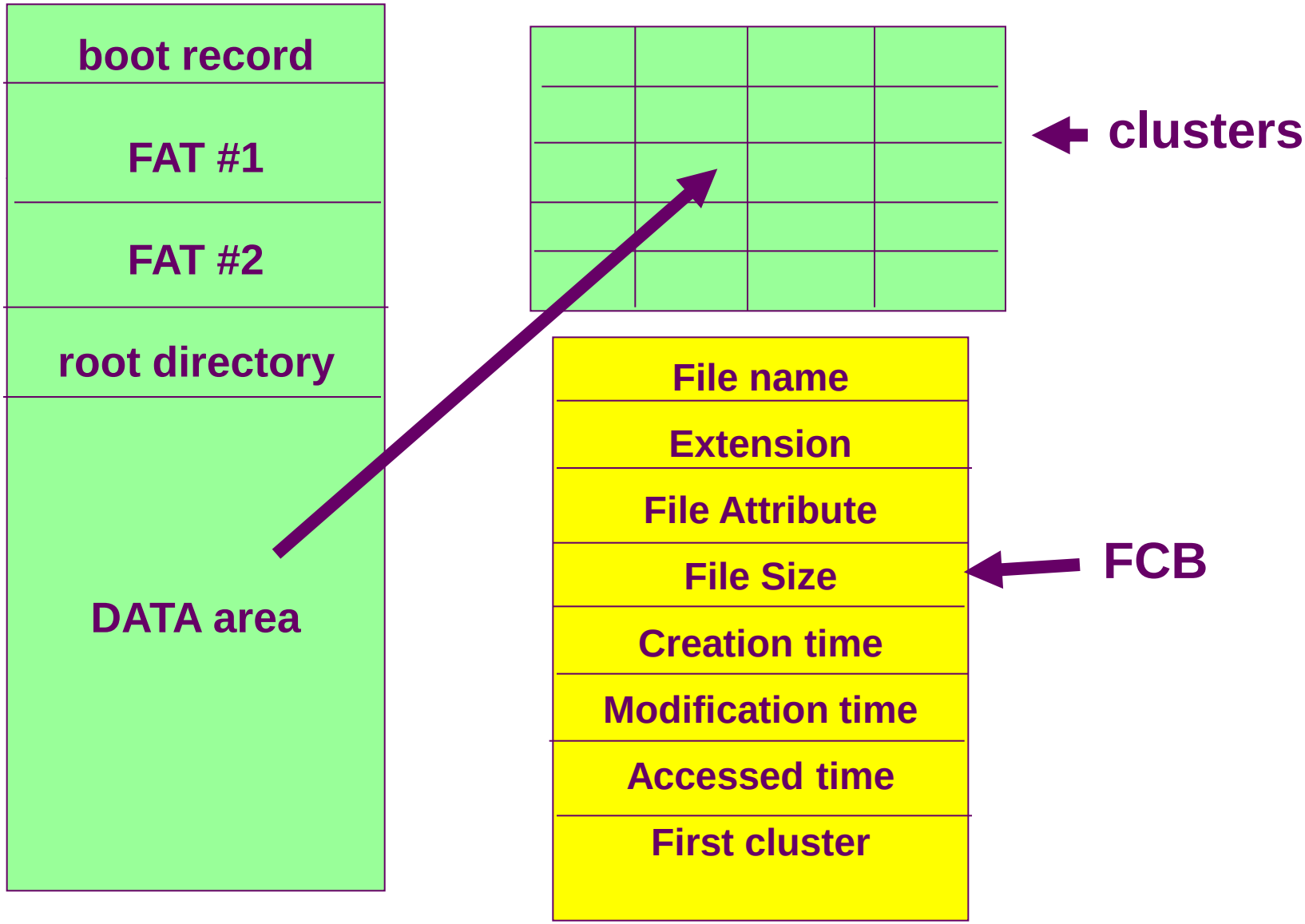
- **Second modification:**
  - **Extent-base allocation unit for file**
  - **Instead one block basis,**
  - **take multiply-block allocation unit, called cluster**

FAT File System Disk Volume Structures

- Keywords = file allocation table
- Minimal disk unit = disk block (512bytes)
- Minimal file unit = cluster
- Cluster = base unit of file = Multiple block size (2k-32K)

- Boot volume block
- FAT #1
- FAT #2
- Data area

FAT example


boot record
FAT #1
FAT #2
root directory
DATA area
clusters
File name
Extension
File Attribute
File Size
Creation time
Modification time
Accessed time
First cluster
FCB

Master Boot Record (MBR)

- hard disk must have a consistent "**starting point**" where key information is stored about the disk, such as how many partitions it has, what sort of partitions they are, etc. There also needs to be somewhere that the BIOS can load the initial boot program that starts the process of loading the operating system. The place where this information is stored is called the master boot record (MBR). It is also sometimes called the master boot sector or even just the boot sector. (Though the master boot sector should not be confused with volume boot sectors, which are different.)
- The **master boot record** is always located at cylinder 0, head 0, and sector 1, the first sector on the disk (see here for more on these disk geometry terms). This is the consistent "starting point" that the disk always uses. When the BIOS boots the machine, it will look here for instructions and information on how to boot the disk and load the operating system. The master boot record contains the following structures:
- **Master Partition Table**: This small table contains the descriptions of the partitions that are contained on the hard disk. There is only room in the master partition table for the information describing four partitions. Therefore, a hard disk can have only four true partitions, also called primary partitions. Any additional partitions are logical partitions that are linked to one of the primary partitions. Partitions are discussed here. One of the partitions is marked as active, indicating that it is the one that the computer should use for booting up.
- **Master Boot Code**: The master boot record contains the small initial boot program that the BIOS loads and executes to start the boot process. This program eventually transfers control to the boot program stored on whichever partition is used for booting the PC.
- Due to the great importance of the information stored in the master boot record, if it ever becomes damaged or corrupted in some way, serious data loss can be--in fact, often will be--the result. Since the master boot code is the first program executed when you turn on your PC, this is a favorite place for virus writers to target.

Volume Boot Sectors

- Each partition (also sometimes called a "logical DOS volume" in the DOS/Windows world) has its own ***volume boot sector***. This is distinct from the *master* boot sector (or record) that controls the entire disk, but is similar in concept. It is also sometimes called the *volume boot record* or *partition boot sector*. **Each volume boot sector contains** the following:
- **1. Disk Parameter Block:** Also sometimes called the ***media parameter block***, this is a data table that contains specific information about the volume, such as its specifications (size, number of sectors it contains, etc.), label name, and number of sectors per cluster used on the partition.
- **2. Volume Boot Code:** This is code that is specific to the operating system that is using this volume and is used to start the load of the operating system. This code is called by the master boot code that is stored in the master boot record, but only for the primary partition that is set as active. For other partitions, this code sits unused.
- The volume boot sector is created when you do a high-level format of a hard disk partition. The boot sector's code is executed directly when the disk is booted, making it a favorite target for virus writers. The information contained in the disk parameter block is used by the operating system to determine where other internal structures of the partition are located, such as the file allocation tables.

FAT copies

- In the conventional FAT system, however, the backup FAT system **doesn't** work too well. The problem is that the two copies are kept right next to each other on the disk, so that if, for example, bad sectors develop on the disk where the first copy of the FAT is stored, chances are pretty good that the **second copy will be affected as well.**

- **Another problem** is that **disk utilities frequently duplicate** the primary FAT to the backup FAT location.
  - This means that **any corruption** that arises in the primary FAT
  - **may be**
  - **duplicated** to the backup copy **before it is noticed**.
- **Under FAT32, some improvements were made to the FAT backup scheme.**
  - **First**, **either copy of the FAT can be designated the "primary" and either the "backup".**
  - **Second**, the method by which the FAT is copied from the primary to the backup location **can be disabled**. The combination of these features allows the second FAT to be protected and used in the event of problems with the first.

Files, Directories, Paths and Directory Tree

- File
- Directory
- root directory
- Path name
- Tree

Internal Directory Structures

- FCB = **Each entry in a directory** is **32 bytes** in length, and **stores the following information**:

File name

Extension

File Attribute

File Size

Creation time

Modification time

Accessed time

First cluster

Root Directory and Regular Directories

- can only be one root directory for any disk volume
  - Conventional FAT
  - Root directory = fixed position + fixed size

Volume Type	maximum number of root directory entries
360 kB 5.25" Floppy Disk	112
720 kB 3.5" Floppy Disk	112
1.2 MB 5.25" Floppy Disk	224
1.44 MB 3.5" Floppy Disk	224
2.88 MB 3.5" Floppy Disk	448
Hard Disk	512

Regular directory & FAT32 root directory

- **Regular directories can have an arbitrary size**;
  - they use space on the disk much the way files do,
  - and when more space is needed to hold more entries,
  - the directory can be expanded the same way a file can

- **Under FAT32,**
- **the root directory is treated much more like a regular directory, and**
  - **can be relocated and expanded in size like any other**.
  - **It's still** a **good idea not to load up** the **root directory**
  - **with too many files**

Special consideration of root

- There are a couple of other special things you should know about the root directory.
- One is that it **cannot be deleted**; the reason for this I would think to be obvious. :^)
- Also, the **root directory has no parent**, since it is at the top of the tree structure.
- The **root directory still contains** a **".." entry**,
  - **but instead** of **pointing** to the **cluster number of the parent directory**
  - like a regular directory's parent entry,
  - it contains a **null value (zero**).

File Names and Extensions

- The file name is **comprised of two parts**:
- **1. File Name:** The base name of the file itself. This part of the file name must be between one and eight characters in length.
- A special code is used as the first character of the file name to indicate **deleted files**.
- **2. File Extension:** The extension of the file, which is optional and hence can be from zero to three characters.
- Since the file name is limited to eight characters and the extension to three, the **conventional DOS naming scheme** is sometimes called ***8.3 naming***.

Long File Names (255 chars)

- "**Mega Corporation - fourth quarter results.DOC**"
  - would be stored as shown,
  - but also under the **alias**
- "**MEGACO~1.DOC**".
- Long file names are stored in regular directories
  - **using the standard directory entries**,
  - but using a **couple of tricks**.
- The Windows 95 file system creates a **standard directory entry** for the file, in which it puts the **short file name alias**.
- Then, it uses **several additional directory** entries to hold the rest of the **long file name**.
- A **single long file name can use many directory entries**
  - (since each entry is only 32 bytes in length),

File Attribute

Attribute	Bit Code
Read-Only	00000001
Hidden	00000010
System	00000100
Volume Label	00001000
Directory	00010000
Archive	00100000

File Attribute

- **Read-Only:** Most software, when seeing a file marked read-only, **will refuse to delete or modify it.** This is pretty straight-forward. For example, DOS will **say "Access denied" if you try to delete a read-only file**. On the other hand, **Windows Explorer** will happily munch it. Some will choose the middle ground: they will let you modify or delete the file, but **only after asking for confirmation.**

- **Hidden:** This one is pretty self-explanatory as well; if the file is marked hidden then under normal circumstances **it is hidden from view**. DOS will not display the file when you type "DIR" **unless a special flag is used**, as shown in the earlier example.

- **System:** This flag is used to tag important files that are used by the system and **should not be altered or removed from the disk**. In essence, this is like a "more serious" read-only flag and is for the most part treated in this manner.

- **Volume Label:** Every disk volume can be assigned an identifying label, either when it is formatted, or later through various tools such as the DOS command "LABEL". **The volume label is stored** in the **root directory** as a file **entry with** the **label attribute set**.

File Attribute

- **Directory:** This is the bit that differentiates between entries that describe files and those that describe subdirectories within the current directory. In theory you can convert a file to a directory by changing this bit. Of course in practice, trying to do this would result in a mess--the entry for a directory has to be in a specific format.

- **Archive:** This is a **special bit** that is used as a "**communications link**" **between software applications that modify files**, and **those that are used for backup**.
  - Most backup software allows the user to do an **incremental backup**, which only selects for backup any files that have changed since the last backup. This bit is used for this purpose.
  - When the **backup software backs up** ("archives") the file, **it clears the archive bit** (makes it zero). **Any software that modifies the file subsequently**, is supposed to **set the archive** bit.
  - Then, the next time that the backup software is run, it knows by looking at the archive bits which files have been modified, and **therefore which need to be backed up**. Again, this use of the bit is "voluntary"; the backup software relies on other software to use the archive bit properly; some programs could modify the file without setting the archive attribute, but fortunately most software is "well-behaved" and uses the bit properly. Still, you should not rely on this mechanism absolutely to ensure that **your critical files are backed up**.

Clusters and File Allocation

- technique for
  - **dividing the storage** on a **disk volume**
  - into discrete chunks,
  - to balance the needs of efficient disk use and performance.

- **These chunks** are called **clusters**.

- The **process** **by** **which files are assigned to clusters**
- **is called allocation,**
- **so clusters** are also **sometimes called allocation units**.

Clusters (Allocation Units)

- **smallest unit** of space on the hard disk is the *sector (***512bytes***)*,
- **individual sectors are not used**

- There are **several performance reasons** for this.
- **keeping track of this many pieces of information** is
  - **time-consuming**
  - **resource-consuming**

- What FAT does instead is to group sectors into larger blocks that are called clusters, or **allocation units**.
  - **cluster size is determined primarily by the size** of the **disk volume**:
  - generally speaking, **larger volumes use larger cluster sizes**.
  - **cluster ranges** in size from **4 sectors (2K) to 64 sectors (32K**).
  - In some situations **128-sector clusters may be used (65K).**

File Chaining and FAT Cluster Allocation

- **There is an entry in the file allocation** table **for each cluster** used on the disk.
- Each entry contains a value that represents how the cluster is being used.
- There are **different codes used** to represent the **different possible statuses** that a cluster can have.
  - **0 = free**
  - **any + = next**
  - **special = last**
  - **special = bad**
- Every cluster that is in use by a file has in its entry in the FAT a cluster number that links the current cluster to the next cluster that the file is using.
- Then that cluster has in **its entry the number of the cluster after it**.
- The **last cluster** used by the file is marked with a special code that tells the system that it is the last cluster of the file; for the FAT16 file system this may be a number like 65,535 (16 ones in binary format).
- **Since the clusters are linked one to** the **next in this manner**, they **are said to be chained.**

File Deletion and Undeletion

- **Delete file under FAT = file is not destroyed**
- **Deletion =system places** the **hex byte code E5h**
  - ☞ into the **first letter of the file name of the file**
- **Do not use its clusters for a while**, as **long free exist**
- **Undelete can be done, easily**
- **But fragmentation is possible**
- **New Method = Recycle bin**

Fragmentation and Defragmentation

- file is stored as a linked list of clusters
- data that is contained in a file **can be located anywhere** on the disk.
- **(clusters can be anywhere)**

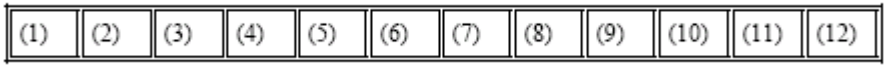
- In the **ideal case**, every file would in fact be **completely contiguous**
- utilities that can optimize the disk by rearranging the files so that they are **contiguous**.
- This process is called **defragmentation or defragmenting**.

- The utilities that do this are, called **defragmenters**.

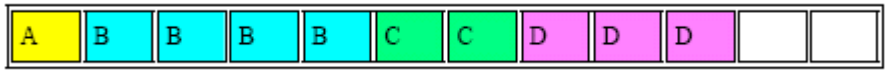
- The **most famous** one is
  - **Norton's SpeedDisk**, and
  - **Microsoft** now includes a DEFRAG program for DOS and a
  - **built-in defragmenter** for **most versions of Windows as well**.

Fragmentation example

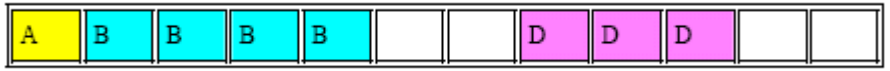
The table below represents the usage of the **12 clusters**. Initially, the table is empty:



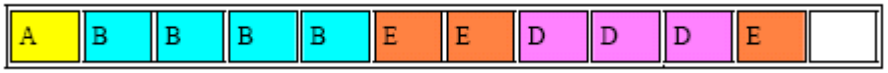
now let's suppose that **we create 4 files**



so we delete C. This leaves the disk looking like this

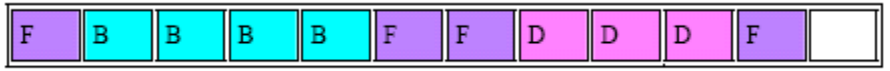


create E (3 clusters)

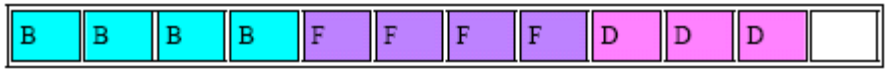


delete A and E

create F (4 clusters)



defragmetation



FAT File System Errors

- **Lost Clusters**
  - **Lost clusters are simply ones** that are
    - **marked in the FAT as being in use**,
    - but that the **system cannot link to any file**
- **Cross-Linked Files**
  - **two files** can end up **pointing to the same data** on the disk **(same cluster)**
- **Invalid Files or Directories (FCBs):**
  - **some entries** are **no longer following the "rules" for how a file or directory is supposed to be laid out**
- **Allocation or FAT Errors:**
  - **entries in the FAT can become corrupted** or **set to invalid values**

FAT Sizes: FAT12, FAT16 and FAT32

- **FAT12:** The **oldest type of FAT**
- **uses a 12-bit binary number** to **hold the cluster number**.

- A volume formatted using FAT12
- can hold a maximum of 2^12= **4,086 clusters**,
- which is **2^12 minus a few values**
- (to allow for reserved values to be used in the FAT).

- FAT12 is therefore most suitable for **very small volumes**, and
- is used on **floppy disks** and **hard disk partitions**
- **smaller than about 16 MB**
- (the latter being rare today.)

FAT Sizes: FAT12, FAT16 and FAT32

- **FAT16:**
- The **FAT used for most older systems**,
- and for **small partitions** on modern **systems**,
- **uses** a **16-bit binary number** to **hold cluster numbers**.

- When you see someone refer to a "FAT" volume generically,
  - they are usually referring to FAT16,
  - because it is the de facto standard for hard disks,
  - even with FAT32 now more popular than FAT16.
- A volume using **FAT16**
- **can hold a maximum of 65,526 clusters**,
- which is **2^16** less a few values
- (again for reserved values in the FAT).

- FAT16 is used for hard disk volumes ranging in **size from 16MB to 2048 MB**.
- **VFAT** is a variant of FAT16.

FAT Sizes: FAT12, FAT16 and FAT32

- **FAT32:**
- The newest FAT type,
- **FAT32** is supported by newer versions of Windows,
- including Windows 95's OEM SR2 release, as well as Windows 98, Windows ME and Windows 2000.
- **FAT32 uses a 28-bit binary cluster number**
- **--*not* 32, because 4 of the 32 bits are "reserved".**
- **28 bits is still enough to permit ridiculously huge volumes—**

- FAT32 can theoretically handle
- volumes with over **268 million clusters**,
- and will support (theoretically) drives up to **2 TB in size**.

- **However to do this the size** of the **FAT grows very large**; see here for details on FAT32's limitations.

FAT Sizes: summary tables

Attribute	FAT12	FAT16	FAT32
Used For	Floppies and very small hard disk volumes	Small to moderate-sized hard disk volumes	Medium-sized to very large hard disk volumes
Size of Each FAT Entry	12 bits	16 bits	28 bits
Maximum Number of Clusters	4,086	65,526	~268,435,456
Cluster Size Used	0.5 KB to 4 KB	2 KB to 32 KB	4 KB to 32 KB
Maximum Volume Size	16,736,256	2,147,123,200	about 2^41

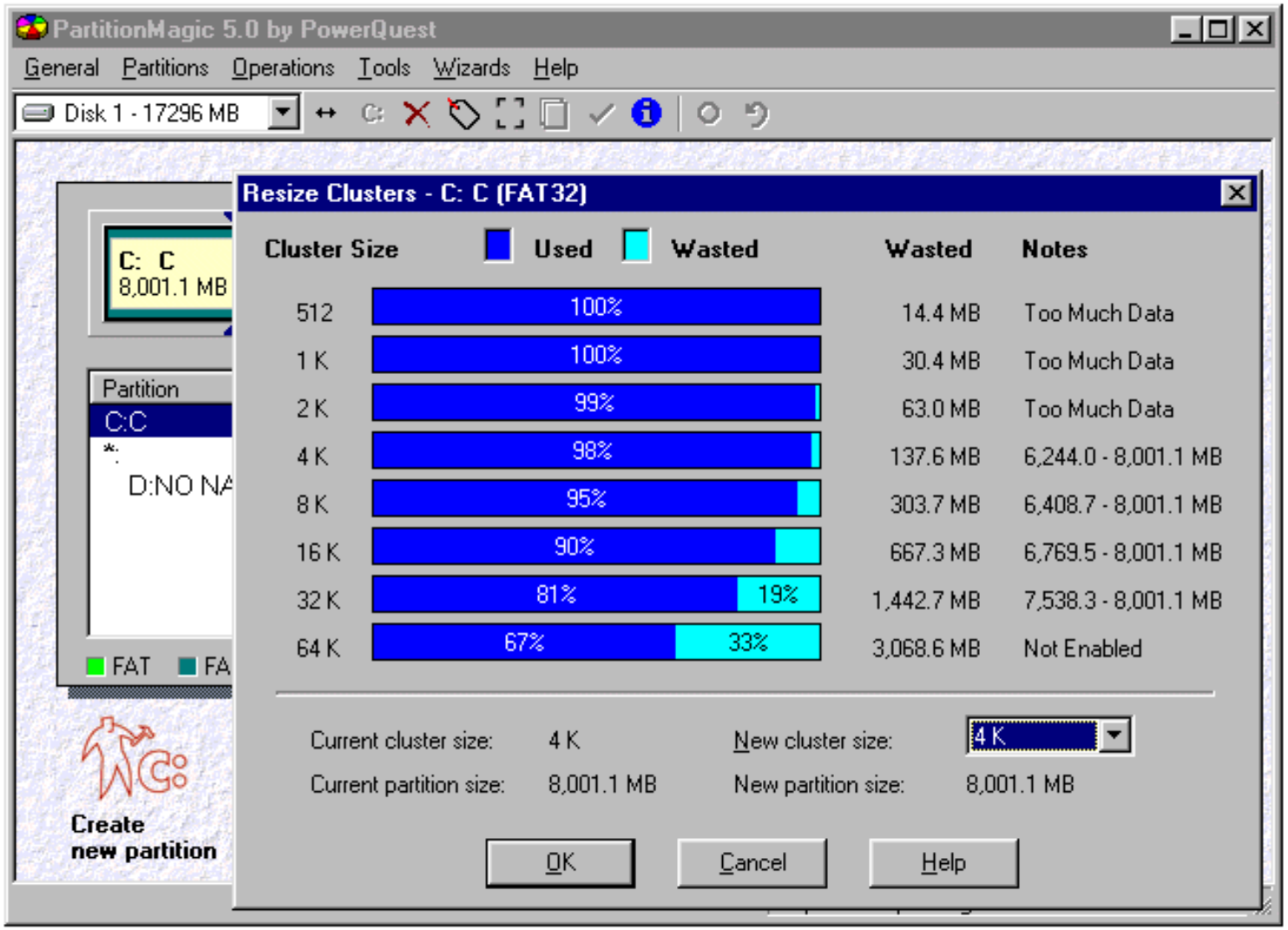
16MB 2GB 2TB

FAT Partition Efficiency: Slack

- **slack**,
- which is the **colloquial term**
- **used** to **refer to wasted space**
- **due to** the **use of clusters** **for storing files**.

- **space is simply wasted as a result of the cluster system**
- that **FAT uses**

Partition Magic: Slack Analysis



FAT16
Relationship of Partition Size and Cluster Size

Cluster Size	Maximum Partition Size (MiB)	Maximum Partition Size (bytes)	
2 kiB	127.98	134,197,248	127MB
4 kiB	255.96	268,394,496	255MB
8 kiB	511.92	536,788,992	
16 kiB	1,023.84	1,073,577,984	1GB
32 kiB	2,047.69	2,147,155,968	2GB

FAT32 Performance Tradeoff: FAT32 Cluster Sizes and FAT Sizes

- **Cluster size**
- **Smaller is better for slack**
- **But FAT tables are larges**, **performances will drop** (metadata caching)
- **FAT16 v FAT32 example**

FAT Type	FAT16	FAT32
Cluster Size	32 kiB	4 kiB
Number of FAT Entries	65,526	524,208
Size of FAT	~ 128 kiB	~ 2 MiB

FAT32: FAT size

Partition Size	4 kiB Clusters	8 kiB Clusters	16 kiB Clusters	32 kiB Clusters
8 GiB	8 MiB	4 MiB	2 MiB	1 MiB
16 GiB	16 MiB	8 MiB	4 MiB	2 MiB
32 GiB	32 MiB	16 MiB	8 MiB	4 MiB
64 GiB	64 MiB	32 MiB	16 MiB	8 MiB
2 TiB (2,048 GiB)	--	*1,024 MiB*	*512 MiB*	*256 MiB*

very large

Using Partitioning to Reduce Slack

- FAT16 example (2GB disk)

FAT16 Cluster Size	Size of Each Partition	Number of Partitions	Typical Total Slack (All Partitions)	Typical Total Slack (% of Disk Space)
2 kiB	128 MiB	16	28 MiB	1.4%
4 kiB	256 MiB	8	56 MiB	2.8%
8 kiB	512 MiB	4	112 MiB	5.6%
16 kiB	1 GiB	2	225 MiB	11.2%
32 kiB	2 GiB	1	450 MiB	22.5%

Using Partitioning to Reduce Slack

- FAT32 example (64GB disk)

FAT32 Cluster Size	Size of Each Partition	Number of Partitions	Typical Total Slack (All Partitions)	Typical Total Slack (% of Disk Space)
4 kiB	8 GiB	8	225 MiB	0.35%
8 kiB	16 GiB	4	450 MB	0.7%
16 kiB	32 GiB	2	900 MB	1.4%
32 kiB	64 GiB	1	1,800 MB	2.8%

This is 1.8GB

Small, but?

Disk Partitioning and Formatting Programs

- fdisk
- format
- sys
- Partition magic

Copying Utilities (Drive Copy, etc.)

- copy
- xcopy
- Drive imaging utility (drive image, Norton ghosts)

Disk Compression

- There are actually many **different types of disk compression**.
- **It is possible to compress:**
  - **individual files**
  - **directories on a hard disk**
  - **compress an entire disk volume**

Compression Types

- The most common compression methods used on PCs are as follows:
- **Utility-Based File Compression:**
- **1. Operating System File Compression**
- **2. Volume Compression**

- **1. Operating System File Compression:** While not supported by the FAT file system used by DOS and most versions of Windows, some operating systems **support the compression of files on an individual basis within** the **operating system itself**.
- For example, Windows NT and Windows 2000 support **this feature when you use the NTFS file system**.
- This is in many ways the best type of compression, because it is both automatic (decompression is done by the operating system when the file is needed by any program) and it allows full control over which types of files are compressed.

Compression Types

- **2. Volume Compression:**
- This option is distinctly different from compressing individual files.
- Using the appropriate operating system, it is also possible to **create entire disk volumes that are compressed**.
- This has traditionally been done either through utilities provided with the operating system, or through third-party packages.
- Volume compression allows you to save disk space without having to individually compress files to use them.
- Every file that is copied to the compressed volume is automatically compressed, and each file is automatically decompressed when any software program needs it.
- Volume compression is transparent to the use and generally works well on most PCs.
- As mentioned in the introduction to this section, it is not used much on newer machines any more, because disks today are so large and cheap.

File Compression Products

- pkzip
- zipmagic
- winzip
- winrar

Volume Compression Products

- **Double-space driver**
- **Drive-space driver**

- **CVF**
- **estimated compression v actual compression**
- Example
- **2:1 v 1.6**

- **Confusion with free space**

- **Slack drops, but performances drop also (require power CPU)**

Estimated free space (500MB)

assume 2:1

500MB is true free space

take **100**MB-file after compress **80**MB

So we have 500 MB of "true" space on the host disk, in the CVF, and 1000 MB of space in the compressed disk, assuming our 2:1 ratio. Now suppose we copy to this empty disk a 100 MB file that cannot be compressed very much; let's suppose it can only be compressed at a ratio of 1.25 to 1. This means we will use up not 50 MB of real CVF space as we would expect from a file compressible at 2:1, but rather 80 MB (100 divided by 1.25). Here's what the disk will look like now:

Storage	Used Space	Free Space	Total
Uncompressed Total (Inside the CVF)	80 MB	420 MB	500 MB
Compression Ratio	1.25:1	2:1	1.88:1
Compressed Total	100 MB	840 MB	940 MB

100/80=1.25

Free
420*2=840

Total
100+840=940
940/500=1.88

80 used + 45 new

Take a new 180MB, after 4:1 compress, 45MB

Now let's copy another file to the compressed disk, let's say a 180 MB database file that will compress at a ratio of 4 to 1. (This is quite common with large database files, believe it or not.) This file will take only 45 MB of storage in the CVF, instead of the 90 MB that is "par" for a 2:1 volume. Here's what will happen:

Storage	Used Space	Free Space	Total
Uncompressed Total (Inside the CVF)	125 MB	375 MB	500 MB
Compression Ratio	2.24:1	2:1	2.06:1
Compressed Total	280 MB	750 MB	1030 MB

280/125=2.24

Be carefully

FATx exFAT TFAT

- **FATX**
- FATX is a slightly modified version of the FAT filesystem, and is designed for Microsoft's Xbox video game console hard disk drive and memory cards. FATX is not to be confused with exFAT, described below.

- **exFAT**
- exFAT (also sometimes incorrectly and inappropriately known as FAT64) is an incompatible replacement for FAT file systems that was introduced with Windows Embedded CE 6.0. MBR partition type is 0x7 (the same as **NTFS**). exFAT is intended to be used on **SDXC** and **flash drives**, where FAT is used today. Microsoft has provided a hotfix to add support for exFAT to Windows XP, while **Windows Vista Service Pack 1** added exFAT support to Windows Vista.
- exFAT introduces a free space bitmap allowing faster space allocation and faster deletes, support for files up to 18,446,744,073,709,551,615 (2^{64}-1) bytes, larger cluster sizes (up to 32 MB in the first implementation), an extensible directory structure and name hashes for filenames for faster comparisons. No short 8.3 filenames are stored. It does not have security **ACLs** or **file system journaling** like **NTFS**, though device manufacturers can choose to implement simplified support for transactions (backup file allocation table used for the write operations, primary FAT for storing last known good allocation table, which is essential for writeable removable media to mitigate corruption).

- **TFAT/TexFAT** Main article: **Transaction-Safe FAT File System**
- TFAT and TexFAT are layers over the FAT and exFAT file systems respectively that provide a level of transaction safety to reduce the risk of data loss in the event of a power outage or unexpected removal of the drive.