Managing files

KT1901 Files and directories, naming conventions, and the File Allocation Table (FAT)

File Allocation Table (FAT)

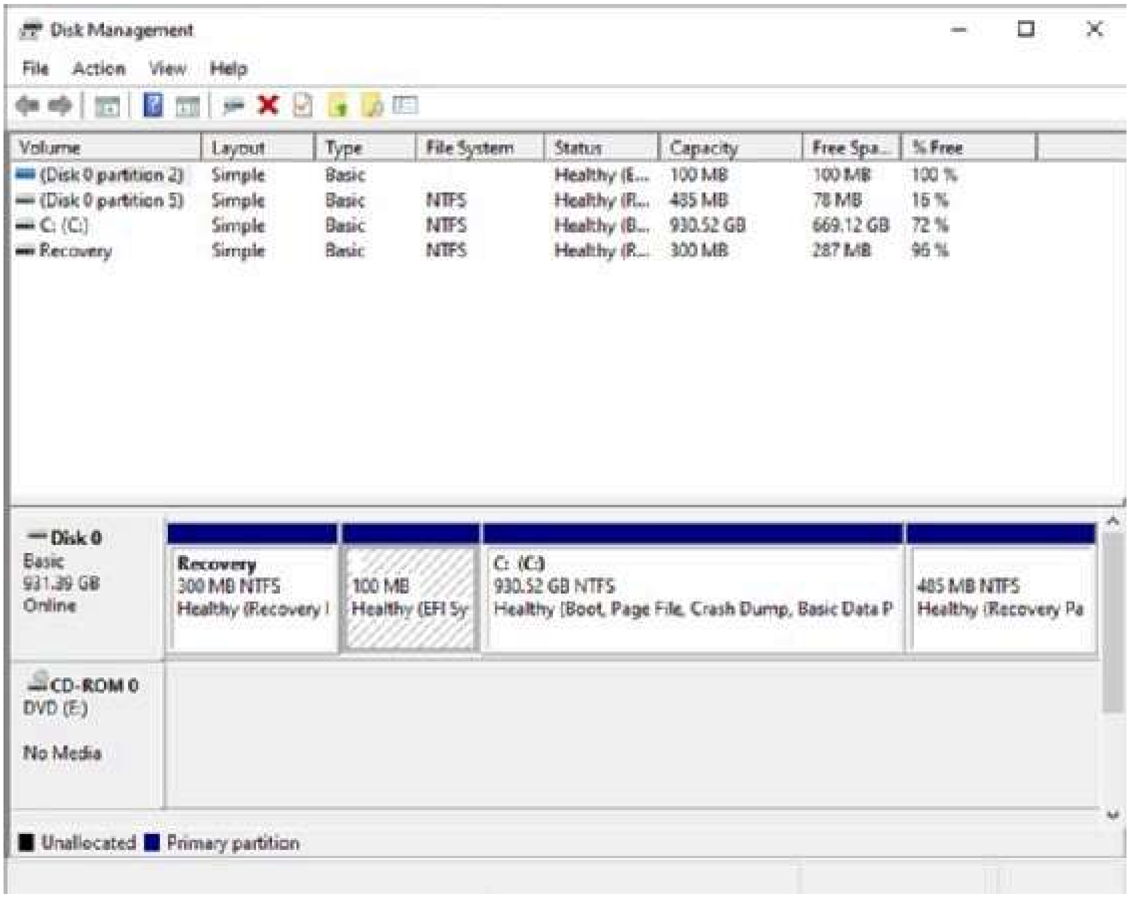
What is File Allocation Table (FAT)?

File Allocation Table (FAT) is a file system that was developed by Microsoft to support small disks and simple folder structures. The file system is named File Allocation Table because it uses a table to track the clusters on a storage volume, as well as how those clusters are linked together through their associated files and directories. Because of the table's important role, the acronym *FAT* is sometimes used to refer to the table itself, rather than the file system, although it's not uncommon to see it used both ways in a single resource.

The FAT file system got its start with the introduction of MS-DOS, or Microsoft Disk Operating System, and is still in use today, but it has evolved over the years to accommodate growing data volumes. During that time, there have been multiple versions of the file system. The original version was FAT8, followed by FAT12, then FAT16 and finally FAT32. Several variants have also been developed based on the last three versions -- FAT12, FAT16 and FAT32 -- further extending the file system.

The number associated with each FAT version (e.g., FAT16) refers to the number of bits that are used for each entry in the allocation table. For example, an allocation table entry for a FAT16 volume is 16 bits long. However, it's not just the number of entry bits that differentiates the FAT versions. Each version also supports larger volumes and file sizes than its predecessor. Even so, FAT is still limited to small-scale storage compared with more modern file systems. The largest file size FAT32 can support is 4 gigabytes, and the largest volume size is 2 terabytes.

At one time, the FAT file system was used extensively for Windows computers. However, today, most Windows systems use the New Technology File System (NTFS) -- or, to a lesser degree, the Resilient File System (ReFS), which Microsoft introduced in 2012 promising to support greater availability, resiliency and scalability.



Volume	Layout	Type	File System	Status	Capacity	Free Spa...	% Free
(Disk 0 partition 2)	Simple	Basic		Healthy (E...	100 MB	100 MB	100 %
(Disk 0 partition 5)	Simple	Basic	NTFS	Healthy (R...	485 MB	78 MB	16 %
C: (C:)	Simple	Basic	NTFS	Healthy (B...	930.52 GB	669.12 GB	72 %
Recovery	Simple	Basic	NTFS	Healthy (R...	300 MB	287 MB	96 %

Disk 0
Basic
931.39 GB
Online

Recovery
300 MB NTFS
Healthy (Recovery I

100 MB
Healthy (EFI Sy

C: (C:)
930.52 GB NTFS
Healthy (Boot, Page File, Crash Dump, Basic Data P

485 MB NTFS
Healthy (Recovery Pa

CD-ROM 0
DVD (E:)

No Media

■ Unallocated ■ Primary partition

Although Windows 10 and 11 still support FAT, they use NTFS by default, and ReFS to a lesser extent.

Although the FAT system has been superseded on most Windows computers, current Windows editions still support FAT, as do macOS, Linux and Unix. In addition, FAT is still used for floppy disks, USB flash drives and storage media in smaller, portable devices such as digital cameras.

What is the function of File Allocation Table?

FAT plays a pivotal role in facilitating file and directory access on a FAT volume, and is directly integrated into the volume's layout. A FAT volume is organized into multiple sections that follow a prescribed order, starting with the boot sector:

Boot sector. Stores the information needed by the file system to access the volume, including the volume's layout and file system structure.

Allocation table region. Stores the primary allocation table, as well as a duplicate of the table that serves as a backup in case the primary table cannot be accessed.

Root directory. Stores a directory table containing entries that describe the files and folders stored on the volume. Each entry includes information such as name, size, created date and last modified date. An entry also includes the starting cluster number, which corresponds to an entry in the file allocation table. For FAT12 and FAT16, the root directory exists at a specified location on the storage media -- right after the allocation tables -- and has a fixed number of maximum entries. The root directory for FAT32 is maintained in the data region and does not have a fixed number of maximum entries.

Data region. Stores the actual file and directory data. This area is divided into clusters that make up the bulk of the volume. The clusters are numbered sequentially and correspond to the listings in the allocation table.

Clusters are the basic units of logical storage on a FAT volume. A cluster can range in size from 512 bytes to 64 kilobytes, depending on volume size and type of FAT file system. Each file written to a FAT volume is stored in one or more clusters, with each cluster available to only one file at a time. A file's clusters might be adjacent to each other in consecutive order or be scattered across the volume, resulting in a fragmented file.

The allocation table on a FAT volume tracks each cluster in that volume, mapping the allocation chains associated with the volume's directories and files. Each table entry corresponds to a cluster and is typically in one of four states:

The cluster is unused.

The cluster is in an allocation chain associated with a specific file or directory; the table entry references to the next cluster in the chain.

The cluster is the last cluster in an allocation chain associated with a specific file or directory.

The cluster is bad.

The first cluster entry in the allocation table is typically used for the volume's FAT ID, and the second entry indicates that the cluster is reserved. In a FAT32 volume, the third entry usually corresponds to the first cluster used for the root directory. The rest of the cluster entries in the allocation table are devoted to the directory and file allocation chains. Each chain begins with the cluster identified in the root directory listing for that directory or file. A chain might be made up of only one cluster, in which case it's marked as the last cluster in the chain.

KT1902 Common executable file extensions, such as batch files, command files, and power shell scripts and purpose

Executable File Formats

You must compile your program to an executable before it can be run or debugged. This topic summarizes the different types of executable file that the Micro Focus COBOL Compiler can produce. It also gives recommendations to help you choose which file format is most appropriate for your applications and summarizes how you can simply packaging by bundling intermediate and generate code files into Micro Focus library files.

.NET COBOL code

.NET COBOL compiles to Common Intermediate Language which can be executed under the Common Language Runtime (CLR) on Windows and Linux systems.

The Compiler supports the generation of procedural COBOL code to .NET assemblies and also offers object-oriented extensions to the COBOL language.

JVM COBOL code

JVM COBOL compiles to Java bytecode (.class files) which can run under the Java Virtual Machine.

The Compiler the supports generation of procedural COBOL code to Java .class files and also offers object-oriented extensions to the COBOL language.

Executable code

You produce executable code by compiling and linking in one step.

An executable file has a filename extension of .exe (Windows) or no filename extension (UNIX).

To create executable code, you need either a COBOL project or a remote COBOL project. Select **Project > Properties > Micro Focus > Build Configuration** and create or edit a configuration, choosing a **Target types** option of **All Executable Files** or **Single Executable File.**

Object code

Object code files are created when you build your project. The object code file is not executable. It requires linking with the run-time system to produce an executable file.

By default, object code files have the extension .obj (Windows) or .o (UNIX).

Object code files must be linked to create .exe or .dll files (Windows) or executable, callable shared object or shared library files (UNIX).

Intermediate code

Intermediate code files have the extension .int and are usually used for testing and debugging. The Compiler creates the intermediate code file during its first phase, when it checks the program syntax. Intermediate code files compile quickly.

.int files are dynamically loadable, and don't need to be linked into a system executable. You can ship them to your users as executable files, but we recommend that you use .exe files and .dll files for this.

Generated code

Generated code files have the extension .gnt and are created by the Compiler, on request, during its second phase. Generated code is slower to compile than intermediate code but the resulting code runs faster.

.gnt files are dynamically loadable, and don't need to be linked into a system executable. You can ship them to your users as executable files, but we recommend that you use .exe files and .dll files for this.

Dynamic Link Libraries (Windows) or Callable shared objects (UNIX)

Dynamic link libraries (Windows) or callable shared objects (UNIX) are dynamically loaded at run time when required; that is, when referenced as a main entry point (for example, by run (Windows) or cobrun (UNIX)) or by the COBOL CALL syntax. When all the entry points in a dynamic link library (Windows) or a callable shared object (UNIX) have been canceled then the file is unloaded, releasing any memory used.

This behavior is similar to .int and .gnt code but is different to linked shared libraries and system executables, which are always loaded at process start-up, whether they are used or not. Further, the code and memory used by shared libraries and system executables are only unloaded when the process terminates.

A dynamic link library (Windows) or a callable shared object (UNIX) can contain more than one COBOL program and can also contain other language programs, such as C and C++. Dynamic link libraries (Windows) or callable shared objects (UNIX) can also be linked with third party object files or shared libraries. This behavior is similar to system executables and shared libraries but differs from .int and .gnt files, where each file corresponds to a single COBOL program.

A dynamic link library (Windows) has a filename extension of .dll.

A callable shared object file (UNIX) has a filename extension of .so.

Recommendations

This section contains general guidance and recommendations about when to use each of the different file formats.

.int files have the advantage of being portable across different hardware platforms. They have the disadvantage of not being optimized for any target architecture and will typically execute more slowly than code that has been generated for a specific platform.

.gnt files are optimized for a specific chipset. They typically execute faster than .int code. Micro Focus recommends the use of generated executables for best application performance.

.gnt files are not locked by the operating system while executing. This means that new files can be overlaid, but this is not considered good practice.

Standard executable formats, such as .dll or .exe, are also optimized for the target platform and can be digitally signed. These formats can also contain more than one program.

Micro Focus recommends the use of standard format executables.

Bundling intermediate or generated code files

You can bundle .int files and .gnt files in Micro Focus library files. The advantages of this are:

It simplifies the packaging of files for users, as it reduces the number of files to be shipped.

When your program calls a subprogram stored in a library file, the library file and all the programs loaded in it are loaded into memory. This can make program execution faster.

A library file has the file extension .lbr.

For example, given the following programs in our application:

Figure 1. Programs in application

ProgC and ProgD could be compiled as generated code files, and then placed in a library file, mylib.lbr. When ProgA calls ProgC, mylib.lbr would be loaded into memory, also loading progc.gnt and progd.gnt. You would need to ship proga.gnt, progb.gnt, and mylib.lbr, as well as a trigger program.

KT1903 Copying, moving, deleting and archiving (ZIP) files and folders

Copy Files or Folders

Copy a file or folder (Windows 10)

Method 1: Right-click

Click **File Explorer** icon.

Go to the location where stores your file or folder (hard drive, USB, etc.).

Click the name of the file or folder you wish to copy.

Right-click the highlighted file or folder and click **Copy**.

Go to the destination folder, right-click the destination folder and click **Paste**.

Method 2: Ribbon toolbar commands

Click **File Explorer** icon.

Go to the location where stores your file or folder (hard drive, USB, etc.).

Click the name of the file or folder you wish to copy.

Click the **Home** tab at the top (on the ribbon).

Click the **Copy** button.

Go to the destination location, click the **Home** tab.

Click the **Paste** button.

Move Files or Folders

Move a file or folder (Windows 10)

Method 1

Click **File Explorer** icon.

Go to the location where stores your file or folder (hard drive, USB, etc.).

Click the name of the file or folder you wish to move.

Click the **Home** tab at the top (on the ribbon).

Click the **Move to** button.

Click **Choose Location** (if you don’t find the right spot on the drop-down menu).

Go to the location you want to move for this folder.

Click **Move**.

Method 2

Click **File Explorer** icon.

Go to the location where stores your file or folder.

Click the name of the file or folder you wish to move.

Hold down the right mouse button and drag the file or folder to the location you wish to move.

Difference between copying and moving files / folders

Copying – make a duplicate of the selected file or folder and place it in another location.

Moving – move the original files or folder from one place to another (change the destination).

The move deletes the original file or folder, while copy creates a duplicate.

Delete and Rename Files or Folders

Delete a file or folder (Windows 10)

Click **File Explorer** icon.

Go to the location where stores your file or folder.

Click the name of the file or folder you wish to delete.

Press the **delete** key (on the keyboard) or right click the file or folder and click **Delete**.

Recover a deleted file or folder

Go to Desktop and click the **Recycle Bin.**

Right-click the name of the file or folder (that you wish to recover). ...

Click **Restore**. (The file or the folder will be restored to its original location.)

Rename files or folders

Click **File Explorer** icon.

Go to the location where stores your file or folder.

Right click the name of the file or folder you wish to rename.

Click **Rename** (on the menu that opens up).

Type a new name for the file and press Enter.

Note:

If a file is in use, Windows will not allow you to rename the file.

Do not change the file extension of a file when renaming the file (file extensions often indicate the file type, file format, etc.).

Internal Assessment Criteria and Weight

- IAC1901 Types of files and the respective purposes are identified

(Weight 2%)

Previous Next

Content List:

note

Computers And Computing Systems