

# Data Storage Concepts and Terminology

## Volumes:

A **volume** is any physical mass of stored data. A disk is a volume. A CD is a volume. A [USB memory stick](#) is also a volume. A **volume label** is simply a user supplied label that can be recorded with a volume to identify it. Volume labels can be recorded onto storage **media** (physical material) during the process of formatting them or at any later time. Volume labels are typically displayed in the heading of all disk directory listings and can be displayed or changed easily by using the correct commands; for example, the `vol` command in Windows.

## Filenames:

Computer data normally is stored as named objects called **files**. When files are **saved** (recorded on storage media), they must be identified by labels called **filenames** that are given to each file by the user. When naming files, you must avoid choosing any label that is reserved for use by your operating system for the labeling of commands or devices that are connected to the computer. For example, you should not name a file "erase" or "exit". Such labels are referred to as **reserved words**. Specific grammatical rules for naming files can be found in the documentation that came with your operating system. But a common rule of thumb that applies to most operating systems is to limit the characters in a filename to only letters, numerals, or the underscore (`_`) symbol. Doing so will ensure that files can be copied between different computer systems without the complications that often result by using characters that are allowed by only one operating system, such a blank spaces or punctuation. Each operating system (OS) behaves differently with regard to its reaction to the use of uppercase and lowercase letters in filenames. Windows is **case-insensitive**; UNIX/Linux is **case-sensitive**; and modern Mac's offer either option. All modern operating systems are **case-aware**, meaning that they will preserve the case of letters in a label.

## Extensions:

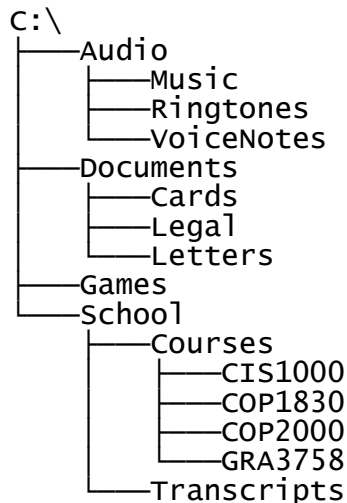
Files can be further identified by an optional **extension** (suffix) often added to the end of a filename, following a period. For example, a file containing a message for your mother might be labeled MOTHER.TXT. The filename MOTHER identifies the contents, and the extension TXT identifies the [data type](#) of file as a **text file**. The choice of both of these labels is usually entirely up to you, although many programs record their own extensions when saving a file on a disk to serve as a reminder of which program authored the file or what data language was used to store it. For more insight into the purpose and use of extensions, read the [web page about Windows® File Association](#).

## Directories (Folders), Sub-Directories, and Paths:

Regardless of how files are physically stored, each OS provides an indexing system known as a **file system** for organizing files into groups and sub-groups. These are typically based on hierarchical lists of files known as **directories** (or **folders**). Each storage volume (disk, CD, DVD, memory stick, etc.) has a primary directory recorded on it known as its **root directory**. This is typically identified by a single slash (forward leaning "/" on Mac's and UNIX/Linux, backward leaning "\" on Windows). An empty root directory is often recorded on a device when it is manufactured, although each OS has a utility program for preparing or renewing a fresh file system on a storage device. This action is known as **formatting**.

As files are recorded on a device, the user provides a filename that is recorded (along with other file

properties such as creation date and size) in the device's directory. Directories frequently get rather large; so each OS allows us to also create **sub-directories** (or **sub-folders**) to help us to group large quantities of files into a more manageable structure. These sub-directories are actually only data files that are used by the OS to supplement the root directory in organizing or indexing the other files on the device. Any directory can contain sub-directories, often resulting in massive hierarchical file systems many levels deep. Users often mentally visualize such structures as folder trees, with an initial root folder that branches into sub-folders, which themselves branch into more sub-folders, and so on. Most modern operating systems display such tree structures as upside-down tree diagrams, with the root at the top and the branches extending downward and to the right as in the generic illustration below for a fictional device named "C:" (a common label for the fixed disk drive on a Windows computer).



Sub-directories are always considered to be **beneath** (subordinate to, or listed within) the Root Directory. They are said to be **children** of the root. Or in other words, the root directory "\" is the **parent** of the sub-directory called **Audio**. Likewise, **Audio** is the parent folder of: **Music**, **Ringtones**, and **VoiceNotes**. People use this genealogical terminology to discuss the position that a folder holds in the file system tree because it is already familiar to them. For example, the folder named **MUSIC** is the **sibling** of **Ringtones** and the **grandchild** of the root folder "\". The folder named **MUSIC** is the **cousin** of the folder named **CARDS** (child of **Documents**); and so on.

In order to **access** (read or write) a file on a disk, your OS must know which directory (folder) to use. Each OS has its own habits about which folder it uses by default. The default folder is referred to as the **current working folder** (or **present working directory**). The Windows operating system often selects the folder named **Documents** by default. Other systems might select the most recently used folder. However, users also can indicate a specific folder to use by typing a **path** to that folder through the file system tree. For example, using Windows, if you wanted to specify an audio file named **BIRDSONG.MP3** in the folder named **Ringtones**, you would typically have to include the names of that folder's ancestors when specifying the file, as in:  
**C:\Audio\Ringtones\BIRDSONG.MP3**

The path to the file above is "**C:\Audio\Ringtones\**". Note that the slashes must lean the other way if you are using a Mac or UNIX/Linux based computer. Also notice that the symbol **\** often is used simply as a **delimiter** (separating character) between folder labels when typing a path description. It only signifies the root directory when it appears at the front of the path, ahead of any directory label.

When a path starts by referring to the name of the device (such as disk drive C:) or to the root directory on that device, then the path is described as an **absolute path**. The path in the example above is an absolute path. Alternatively, if your current working folder is the **Audio** folder, then you could have specified the **BIRDSONG.MP3** file with a shorter **relative path** containing only enough information to get from the current working folder to the file's location. The relative path to the **BIRDSONG.MP3** file from the **Audio** folder would be just:  
`Ringtones\BIRDSONG.MP3`

In addition to the labels provided by the user when saving files and making folders, each sub-directory contains entries recording information about itself and its parent folder. These appear in the directory listings as files named `.` (dot) and `..` (double dot) respectively. Users and programs employ these entries to navigate the directory tree. For example, if the current working folder was the **Ringtones** folder and you wanted to refer to a file named **NOTE1.WAV** in the **VoicENotes** folder, you could specify the path as either:

Absolute notation: `C:\Audio\VoicENotes\NOTE1.WAV`

or

Relative notation: `..\VoicENotes\NOTE1.WAV`

The double dot (`..`) refers to the parent of the *current* folder (**Audio**). Your choice of which path notation you use will typically depend on which is shorter to type, although there are reasons to sometimes prefer one notation over another.

### Changing the Current Working Directory:

Every OS provides a set of commands to define or report the current working directory. If you have a graphic user interface (GUI), then you typically just click on the name of a folder in the displayed tree to select it as the current working directory. However, if you are using a command line interface (CLI), then you often need to learn a few simple command words for this purpose. Fortunately, most operating systems use the same command word (**cd**) to change the default directory. Many CLI's display the name of (or the complete absolute path to) the current working directory as part of the message urging the user to type the next command. For example, the Windows Command Prompt utility often requests its next command with text similar to:

```
C:\Documents\Letters>
```

The **prompt** (message urging the user to act) above indicates that the current working directory on that system is "`C:\Documents\Letters`". If a user wanted to change the current working folder to be `C:\Games` instead, the command would be simply:  
`cd \Games`

The absolute path above lacks mention of the C: disk drive, because that drive was already the default; so mention of it would be redundant. Alternatively, the working directory could have been set using the relative notation, but it would have involved more typing to navigate up the tree (through two generations) to reach the target:  
`cd ..\..\Games`

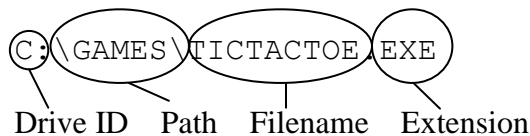
Normally, if you change the current working directory, Windows will only search that directory to find a file. So, if you were to change the current working directory to be the **Audio** sub-directory, Windows would not normally seek files recorded in other directories. Regardless of the directory that is currently

defined as the default, you can always access a file in any directory by defining a complete path to it when specifying the file. For more information on using Windows Commands, read the page about the [Windows "Command Prompt" Utility](#).

## File Specifications:

Every file must be given a filename. The filename implies the contents of the file and serves as an easy way to identify and locate it later for retrieval. Many filenames are also enhanced by adding a short suffix known as an **extension** following the filename which serves to identify the file's [data type](#) or to indicate which program recorded it. For example, many *text* files use an extension of TXT to indicate that data type. The programming language named C++ records the extension CPP after all of its filenames as an identifying signature.

On Windows-based computers, storage devices are identified by their **Drive ID** (a single identifying letter followed by a colon, such as "C:"). When combined with a path and a filename (with extension), a user can specify a single unique file in the file system. A proper **file specification** for an executable machine language program file named TICTACTOE that was recorded on a disk labeled C: in a sub-directory named GAMES would be:



A file specification must always be entered in the following order: (1) Device ID, (2) Path, (3) Filename, and (4) Extension (optional). The term "file specification" is frequently abbreviated as just **file-spec**.

## Wildcards:

We sometimes need to execute a single command that can affect more than one file at a time. To do this, you must have a grammatical way of referring to a group of files. The most common method for doing this involves the use of special characters called **wildcards** when typing file specifications. These characters allow us to type something called an **ambiguous file specification** that can refer to more than one file. For example, the filename `PROJ?` indicates *any* of the following files: `PROJ1`, `PROJ2`, `PROJA`, `PROJB`, etc. The question mark (?) wildcard allows the OS to accept any *single* valid character in its position. Note that the ambiguous reference of `PROJ?` would *not* properly refer to a file named `PROJECT` because the "?" only allows a *single* character to be matched. We could specify *all* 8 character long filenames which started with the letters "PROJ" by typing `PROJ????`. An easier way to do this is to use the other wildcard, the asterisk (\*). A specification of `PROJ*` tells the OS to match all remaining characters in the filename, starting with the position of the asterisk. Because the word asterisk is somewhat difficult to say, many people simply call that symbol a **star**. Both wildcards can be used in either the filename or the extension. A file-spec of `C*.EXE` refers to all *executable* files with filenames which start with the letter "C". The file-spec `MINE.*` indicates all files named MINE, regardless of their extension. You can refer to all files listed in a directory by using the fully ambiguous file-spec of `*.*` (usually spoken as "star, dot, star").

For more information about using your computer's file system, refer to the documentation for your operating system.