Hidden File Definition

A *hidden file* is a *file* that is not normally visible when examining the contents of the *directory* in which it resides. Likewise, a *hidden directory* is a directory that is normally invisible when examining the contents of the directory in which it resides.

A file is a named collection of related information that appears to the user as a single, contiguous block of data and that is retained in *storage*. Storage refers to computer devices or media that can retain data for relatively long periods of time (e.g., years or decades), such as hard disk drives (HDDs), CDROMs and magnetic tape; this contrasts with *memory*, which retains data only as long as the data is in use or the memory is connected to a power supply.

A directory (also sometimes referred to as a *folder*) can be conveniently viewed as a container for files and other directories. In Linux and other Unix-like operating systems, a directory is merely a special type of file that associates file names with a collection of *metadata* (i.e., data about about the files). Likewise, a *link* is a special type of file that points to another file (which can be a directory). Thus, it is somewhat redundant to use phrases such as *hidden files and directories*; however, they are descriptive and convenient, and thus they are frequently used. More precise terms are *hidden filesystem objects* and *hidden items*.

Hidden items on Unix-like operating systems are easily distinguishable from regular (i.e., non-hidden) items because their names are prefixed by a period (i.e., a dot). In Unix-like operating systems, periods can appear anywhere within the name of a file, directory or link, and they can appear as many times as desired. However, usually, the only time that they have special significance is when used to indicate a hidden file or directory[1].

In the Microsoft Windows operating systems, whether a filesystem object is hidden or not is an *attribute* of the item, along with such things as whether the file is read-only and a *system file* (i.e., a file that is critical to the operation of the operating system). Changing the visibility of such items is accomplished using a multi-step procedure.

Unix-like operating systems provide a larger set of attributes for filesystem objects than do the Microsoft Windows operating systems, including a system of *permissions*, which control which user(s) have access to each such object for reading, writing and executing. However, whether objects are hidden or not is not among the attributes. Rather, it is merely a superficial property that is easily changed by adding or removing a period from the beginning of the object name.

Many operating systems and application programs routinely hide objects in order to reduce the chances of users accidentally damaging or deleting critical system and configuration files. Hiding objects can also be useful for reducing visual clutter in directories, and thereby making it easier for users to locate desired files and subdirectories.

Another reason to hide filesystem objects is to make them invisible to casual snoopers. Although it is a very simple matter to make hidden files and directories visible, the great majority of computer users are not even aware that such files and directories exist (nor need they be).

Hidden filesystem objects are not listed by default when using the *ls* command (which is commonly employed to list the items in a directory). However, they can easily be made to appear by adding the *-a* option, which instructs ls to show *all* items in the designated directory. For example, the following command will display all items (inclusive of hidden items) in the *root directory* (i.e., the top level directory, which is represented by a forward slash):

```
ls -a /
```

As can be easily observed, every directory in a Unix-like operating system contains two *special files* whose names consist of dots only: one with a single dot and the other with two dots. They are not hidden items. The former represents the *current directory* (i.e., the directory in which the user is currently working) and the latter represents its *parent directory* (i.e., the directory in which the current directory resides).

The *-A* option can be used with the ls command in place of the -a option to tell ls to list all items in a directory except for these two *special files*. For example, the following would list all items exclusive of them in the current directory:

```
ls -A
```

Likewise, hidden items are invisible by default when viewing the contents of directories using a GUI (graphical user interface) file manager, such as *Nautilus*, which is the default file manager on the widely used GNOME desktop environment. However, they can be made to appear in Nautilus by opening the *Edit* menu, then selecting *Preferences* and finally checking the box next to *Show hidden and backup files*.

Hidden items are, of course, completely visible to the operating system. They can also be visible to application programs, but they are not usually visible to user interfaces of application programs. For example, they are not visible to the *gedit* text editor or to the *AbiWord* word processor, and thus they cannot be opened by these programs unless they are first changed to non-hidden items.

Hidden items on Unix-like operating systems are most commonly found in users' *home directories* (i.e., the directory that serves as a repository for the user's personal files). This includes the directory */root*, which is the home directory of the *root* (i.e., administrative) user (which should not be confused with the root directory). Configuration files that reside in users' home directories are hidden by default as a means of protecting them from accidental damage or deletion. This danger exists because each user by default has permission to modify or delete any file or directory in its home directory.

An example of a hidden file in a user's home directory on a typical Linux system is *.bash_history*, which is a *plain text* file (i.e., a file that consists entirely of human readable characters) listing commands that have been issued by that user in the *bash shell*. A shell is a program that provides the traditional, text-only user interface for Unix-like operating systems, and bash is the default shell on Linux.

Examples of hidden directories in each user's home directory on a typical Linux system are *.gnome* and *.kde*. The former contains the configuration data for that user's GNOME *desktop* and GNOME-based applications, and the latter contains the configuration data for that user's KDE desktop and KDE-based applications. A desktop is the background that is visible on a GUI when all windows are closed.

In contrast to configuration files for ordinary users, it is not necessary for system-wide configuration files to be hidden. This is because they are already inaccessible to ordinary users by means of their permissions, which generally allow them to be modified and executed only by the root user.

On Unix-like operating systems, ordinary items can be easily changed to hidden items and visa versa both in the GUI and at the *command line* (i.e., in text-only mode). As an example of the latter, the following uses the *mv* command, which is used to move and rename files, to convert a visible file named *yuriko* into a hidden file:

```
mv yuriko .yuriko
```

Similarly, the hidden file *.yuriko* can be converted into an ordinary file as follows:

```
mv .yuriko yuriko
```

Intruders sometimes insert hidden files or directories containing malicious code, such as password cracking programs, into users' home directories, as their hidden nature makes them less obvious and more difficult to detect by casual inspection. Thus, it is a good idea for system administrators to occasionally check their systems for unusual or suspicious hidden files and directories. A list of all hidden filesystem objects on a system can be generated by using the *find* command with its *-name* option (which tells it to search by name) as follows:

```
find / -name ".*"
```

The forward slash tells find to begin its search at the root directory (i.e., to search the entire filesystem). The asterisk (star-shaped character) is a *wildcard* that tells find to select every name that begins with a period regardless of the number of characters in the name after the period.

---

[1]The main exception is in the case of a relatively few types of filename extensions, which can be necessary for the proper functioning of some programs.

Created March 8, 2005. Updated July 21, 2006.