File System Basics

A *file system* handles the persistent storage of data files, apps, and the files associated with the operating system itself. Therefore, the file system is one of the fundamental resources used by all processes. APFS is the default file system in macOS, iOS, watchOS, and tvOS. APFS replaces HFS+ as the default file system for iOS 10.3 and later, and macOS High Sierra and later. macOS additionally supports a variety of other formats.Regardless of the underlying format, all of the disks attached to the device—whether they are physically plugged in or are connected indirectly through the network—contribute space to create a single collection of files. Because the number of files can easily be many millions, the file system uses directories to create a hierarchical organization.

The macOS file system is designed for Mac computers, where both users and software have access to the file system. Users access the file system directly through the Finder, which presents a user-oriented view of the file system by hiding or renaming some files and directories. Apps access the file system using the system interfaces, which show the complete file system precisely as it appears on disk.

Domains Determine the Placement of Files

In macOS, the file system is divided into multiple domains, which separate files and resources based on their intended usage. This separation provides simplicity for the user, who only needs to worry about a specific subset of files. Arranging files by domain also lets the system apply blanket access privileges to files in that domain, preventing unauthorized users from changing files intentionally or inadvertently.

- The *user domain* contains resources specific to the users who log in to the system. Although it technically encompasses all users, this domain reflects only the home directory of the current user at runtime. User home directories can reside on the computer's boot volume (in the `/Users` directory) or on a network volume. Each user (regardless of privileges) has access to and control over the files in his or her own home directory.
- The *local domain* contains resources such as apps that are local to the current computer and shared among all users of that computer. The local domain does not correspond to a single physical directory, but instead consists of several directories on the local boot (and root) volume. This domain is typically managed by the system, but users with administrative privileges may add, remove, or modify items in this domain.
- The *network domain* contains resources such as apps and documents that are shared among all users of a local area network. Items in this domain are typically located on network file servers and are under the control of a network administrator.
- The *system domain* contains the system software installed by Apple. The resources in the system domain are required by the system to run. Users cannot add, remove, or alter items in this domain.

Figure 1 shows how the local, system, and user domains map to the local file system of a macOS installation. (The network domain is not shown but is similar in many ways to the local domain.) This figure shows the visible directories that the user might see. Depending on the user's system, other directories may be visible or some of the ones shown here may be hidden.

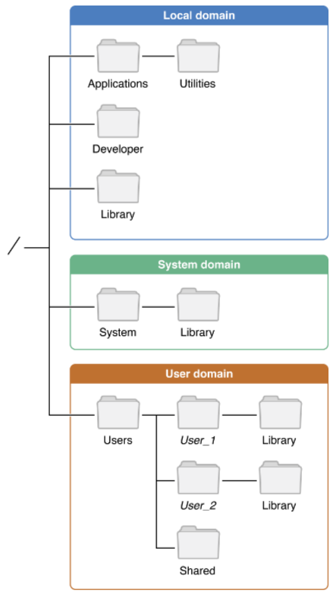

Figure 1 The local macOS file system

macOS Standard Directories: Where Files Reside

Whether provided by the system or created by your app, every file has its place in macOS. Table 1 lists some of the top-level directories in a macOS installation and the types of content that each one contains.

Table 1 Commonly used directories in macOS

Directory	Usage
`/Applications`	This directory is where you install apps intended for use by all users of a computer. The App Store installs apps purchased by the user in this directory automatically. The `Utilities` subdirectory contains a subset of apps that are intended for use in managing the local system. This directory is part of the local domain.
`Library`	There are multiple `Library` directories on the system, each one associated with a different domain or specific user. Apps should use the `Library` directory to store app-specific (or system-specific) resources.

`/Network`	This directory contains the list of computers in the local area network. There is no guarantee that files located on network file servers will have the `/Network` directory at the beginning of their path. Path names vary depending on several factors, including how the network volume was mounted. For example, if the user uses the Connect to Server command to mount a volume, paths begin with the `/Volumes` directory. When writing code, assume that files on any volume other than the boot volume could be located on a network-based server.
`/System`	This directory contains the system resources required by macOS to run. These resources are provided by Apple and must not be modified. This directory comprises the contents of the system domain.
`/Users`	This directory contains one or more user home directories. The user home directory is where user-related files are stored. A typical user's home directory includes the following subdirectories: • `Applications`—Contains user-specific apps. • `Desktop`—Contains the items on the user's desktop. • `Documents`—Contains user documents and files. • `Downloads`—Contains files downloaded from the Internet. • `Library`—Contains user-specific app files (hidden in macOS 10.7 and later). • `Movies`—Contains the user's video files. • `Music`—Contains the user's music files. • `Pictures`—Contains the user's photos. • `Public`—Contains content the user wants to share. • `Sites`—Contains web pages used by the user's personal site. (Web Sharing must be enabled to display these pages.) The preceding directories are for storing user documents and media only. Apps must not write files to the preceding directories unless explicitly directed to do so by the user. The sole exception to this rule is the `Library` directory, which apps may use to store data files needed to support the current user. Of the subdirectories, only the `Public` directory is accessible by other users on the system. Access to the other directories is restricted by default.

Important: The files in the user's `Documents` and `Desktop` directories should reflect only the documents that the user created and works with directly. Similarly, the media directories should contain only the user's media files. Those directories must never be used to store data files that your app creates and manages automatically. If you need a place to store automatically generated files, use the `Library` directory, which is designated specifically for that purpose.

Hidden Files and Directories: Simplifying the User Experience

To simplify the experience for users, the Finder, and some specific user-facing interfaces (such as the Open and Save panels), hide many files and directories that the user should never have to use. Many of the hidden items are system- or app-specific resources that users cannot (or should not) access directly. Among the files and directories that are hidden are the following:

- **Dot directories and files.** Any file or directory whose name starts with a period (`.`) character is hidden automatically. This convention is taken from UNIX, which used it to hide system scripts and other special types of files and directories. Two special directories in this category are the `.` and `..` directories, which are references to the current and parent directories respectively.
- **UNIX-specific directories.** The directories in this category are inherited from traditional UNIX installations. They are an important part of the system's BSD layer but are more useful to software developers than end users. Some of the more important directories that are hidden include:
    - `/bin`—Contains essential command-line binaries. Typically, you execute these binaries from command-line scripts.
    - `/dev`—Contains essential device files, such as mount points for attached hardware.
    - `/etc`—Contains host-specific configuration files.
    - `/sbin`—Contains essential system binaries.
    - `/tmp`—Contains temporary files created by apps and the system.
    - `/usr`—Contains non-essential command-line binaries, libraries, header files, and other data.
    - `/var`—Contains log files and other files whose content is variable. (Log files are typically viewed using the Console app.)
- **Explicitly hidden files and directories.** The Finder may hide specific files or directories that should not be accessed directly by the user. The most notable example of this is the `/Volumes` directory, which contains a subdirectory for each mounted disk in the local file system from the command line. (The Finder provides a different user interface for accessing local disks.) In macOS 10.7 and later, the Finder also hides the `~/Library` directory—that is, the `Library` directory located in the user's home directory.
- **Packages and bundles.** Packages and bundles are directories that the Finder presents to the user as if they were files. Bundles hide the internal workings of executables such as apps and just present a single entity that can be moved around the file system easily. Similarly, packages allow apps to implement complex document formats consisting of multiple individual files while still presenting what appears to be a single document to the user.

The Library Directory Stores App-Specific Files

The `Library` directory is where apps and other code modules store their custom data files. Understanding the structure of the `Library` directory is important. You use this directory to store data files, caches, resources, preferences, and even user data in some specific situations.

There are several `Library` directories throughout the system but only a few that your code should ever need to access:

- `Library` in the current home directory—This is the version of the directory you use the most because it is the one that contains all user-specific files. In iOS, `Library` is placed inside the apps data bundle. In macOS, it is the app's sandbox directory or the current user's home directory (if the app is not in a sandbox).
- `/Library` (macOS only)—Apps that share resources between users store those resources in this version of the `Library` directory. Sandboxed apps are not permitted to use this directory.
- `/System/Library` (macOS only)—This directory is reserved for use by Apple.

Table 2 Key subdirectories of the `Library` directory

Directory	Usage
`Application Support`	Use this directory to store all app data files except those associated with the user's documents. For example, you might use this directory to store app-created data files, configuration files, templates, or other fixed or modifiable resources that are managed by the app. An app might use this directory to store a modifiable copy of resources contained initially in the app's bundle. A game might use this directory to store new levels purchased by the user and downloaded from a server. All content in this directory should be placed in a custom subdirectory whose name is that of your app's bundle identifier or your company.
`Caches`	Use this directory to write any app-specific support files that your app can re-create easily. Your app is generally responsible for managing the contents of this directory and for adding and deleting files as needed.
`Frameworks`	In macOS, frameworks that must be shared by multiple apps can be installed in either the local or user domain. The Frameworks directory in the system domain stores the frameworks you use to create your macOS apps.
`Preferences`	This directory contains app-specific preference files. You should not create files in this directory yourself. Instead, use the NSUserDefaults class or CFPreferences API to get and set preference values for your app.

How the System Identifies the Type of Content in a File

There are two primary techniques for identifying the type of content in a file:

- Uniform Type Identifiers (UTIs)
- Filename extensions

A *uniform type identifier* is a string that uniquely identifies a class of entities considered to have a "type." UTIs provide consistent identifiers for data that all apps and services can recognize and rely upon. They are also more flexible than most other techniques because you can use them to represent any type of data, not just files and directories. Examples of UTIs include:

- `public.text`—A public type that identifies text data.
- `public.jpeg`—A public type that identifies JPEG image data.
- `com.apple.bundle`—An Apple type that identifies a bundle directory.
- `com.apple.application-bundle`—An Apple type that identifies a bundled app.

Whenever a UTI-based interface is available for specifying file types, you should prefer that interface over any others. Many macOS interfaces allow you to specify UTIs corresponding to the files or directories you want to work with. For example, in the Open panel, you can use UTIs as file filters and limit the types of files the user selects to ones your app can handle. Several AppKit classes, including `NSDocument`, `NSPasteboard`, and `NSImage`, support UTIs.

Security: Protect the Files You Create

Because all user data and system code are stored on disk somewhere, protecting the integrity of files and the file system is an important job. For that reason, there are several ways to secure content and prevent it from being stolen or damaged by other processes.

Permissions and Access Control Lists Govern All Access to Files

Access to files and directories is governed by a mixture of access control lists (ACLs) and BSD permissions. Access control lists are a set of fine-grained controls that define exactly what can and cannot be done to a file or directory and by whom. With access control lists, you can grant individual users different levels of access to a given file or directory. By contrast, BSD permissions only allow you to give access to three classes of users: the file's owner, a single group of users that you specify, and all users.

Files Can Be Encrypted On Disk

Users can encrypt the contents of a volume using the Disk Utility app. (They can also encrypt just the boot volume from the Security & Privacy system preference.) The contents of an encrypted disk are available to apps only while the computer is running. When the user puts the computer to sleep or shuts it down, the decryption keys are destroyed to prevent unauthorized access to the disk's contents.

Files, Concurrency, and Thread Safety

Because file-related operations involve interacting with the hard disk and are therefore slow compared to most other operations, most of the file-related interfaces in macOS are designed with concurrency in mind. Several technologies incorporate asynchronous operation into their design and most others can execute safely from a dispatch queue or secondary thread.