CodeBuilder User's Guide - 6.0 File Systems

Archived: 2026-04-05 22:01:56 UTC

In the traditional UNIX world, disks are devices that can be formatted into file systems and mounted for access by UNIX applications. In the Macintosh world, disks are called volumes. The Macintosh File Manager is the entity responsible for formatting disks into volumes and controlling access to the files on those volumes. CodeBuilder augments the MacOS HFS file system with two UNIX file system implementations, a native fast file system (FFS) and a UNIX file system that sits on top of the MacOS HFS (UFS).

The first, known as FFS, implements a traditional native UNIX fast file system either within a single Macintosh file or in a SCSI disk partition. There is no need to reformat or partition a volume to create a FFS within a file, but that will be necessary for FFS on a partition. For an FFS within a file, CodeBuilder continues to use the services of the Macintosh File Manager, but only for basic I/O operations. For an FFS on a partition, CodeBuilder interacts directly with the Macintosh SCSI Manager. Macintosh files may be stored on an FFS, but they are not available for access by Macintosh applications. A future extension of FFS will permit desktop mounting and use by Macintosh applications.

The second file system, known as UFS, implements the UNIX file system on top of the Macintosh hierarchical file system. CodeBuilder imposes UNIX file system semantics on the Macintosh by mapping the UNIX file system requests onto the appropriate Macintosh File Manager routines. UNIX files are stored alongside Macintosh files, so there is no requirement to reformat/partition existing HFS volumes.

6.1 UNIX Fast File System Overview

CodeBuilder's FFS is derived from the Berkeley BSD UNIX high-performance file system. It is a simple and elegant file system that has its roots in the original UNIX file system developed at Bell Labs. Detailed information may be found in two supplementary documents: (1) *A Fast File System for UNIX*, and (2) *Fsck - The UNIX File System Check Program*. These are located in the Documentation folder under */CodeBuilder UNIX Docs/Sys Admin's Docs (SMM)/fastfs.pdf* or *fsck.pdf*.

6.1.1 File System Organization

UNIX disks are divided into one or more partitions. Each partition may contain one file system, and a file system never spans multiple partitions. Critical file system sizing parameters are stored in a *super-block*, which is replicated to protect against catastrophic loss.

The file system stores files. Certain files are designated as directories which contain pointers to other files, some of which may be other directory files. Every file has an associated descriptor called an *inode*, which contains ownership, permissions, time stamps, and pointers to assigned data blocks.

A disk partition is divided into one or more areas called *cylinder groups*. A small percentage of each cylinder group is taken for some bookkeeping information, including a redundant copy of the super-block, inode slots, a bitmap of available data blocks, and data block usage summary information. A static number of inodes is allocated when the file system is created. The default policy is to create one inode per 2048 bytes of space in the cylinder group, which is expected to be far more than will be needed.

UNIX files are a single stream of bytes, with no operating system imposed format. Text files use the ASCII 'LF' character ('\n') to denote an end-of-line. Binary files often begin with a four byte code identifying the format of the data that follows.

6.1.2 File Names

UNIX file and directory names are limited to 255 characters. MacOS file names are limited to 31 characters. Names may not include the '/' character because it is used in pathnames to separate directory and file name components. An absolute pathname begins with a '/'; anything else is relative to the current working directory. Name matching is case sensitive. (MacOS file names are **NOT** case sensitive, however Tenon's extension enables case sensitivity in Tenon's UFS file system.)

6.1.3 Access Permissions

UNIX file and directory access rights are divided into read, write, and execute permissions. Read permission allows a user to examine the contents of a file. Write permission allows a user to change or append data to a file. Execute permission for binary files allows a user to execute the file as a program, and for directories allows a user to search the directory. Each file and directory is tagged with a user and group ID. Three levels of permissions are specified, one for the owner of a file, a second for the members of the designated group, and a third for "everyone else".

6.1.4 Time Stamps

UNIX saves three time stamps for each file: (1) the time the file's data was modified, (2) the time the file's attributes were last modified, and (3) the time the file was last accessed (i.e. the data was read).

6.1.5 Link Counts

Physical files are uniquely identified by their inode, but they may be identified by multiple names. The link count is simply the number of references to an inode.

6.1.6 Hard Links

The mapping of a file name in a directory to an inode is known as a hard link. Hard links are not allowed to cross file system boundaries.

6.1.7 Symbolic Links

Symbolic links are special files whose content is the pathname of the intended file or directory. The pathname may be absolute or relative, and may refer to other file systems. Symbolic links may point to other symbolic links.

6.2 Macintosh Hierarchical File System Overview

The MacOS method of organizing files is known as the hierarchical file system (HFS). Detailed information about the Macintosh file system and File Manager may be found in the Files manual which is part of the Inside Macintosh documentation series, published by Addison-Wesley.

6.2.1 File System Organization

Each Macintosh hard disk is formatted into a number of 512-byte addressable units known as *logical blocks*. A *volume* is a consecutive sequence of these blocks. Small disks are typically used whole as a single volume, while large disks are often partitioned into two or more volumes.

A volume consists of overhead, file storage, and free space. The overhead includes boot blocks, bitmap, master directory blocks, catalog and extents files. The catalog file maintains the hierarchy of directories and files on a volume, and the extents overflow file tracks assignment of allocation blocks to files which cannot be stored in the catalog file. File storage is assigned in units of allocation blocks, which are one or more consecutive logical blocks. A volume has at most 65,536 *allocation blocks*. This addressing constraint dictates the ratio of logical blocks to an allocation block. Files are stored in directories (also known as folders), which are themselves stored in directories. Each directory and file is assigned an integer ID which uniquely identifies it on a given volume. Every volume has a top directory known as the root directory, with a directory ID of 2. When a volume is mounted for use, the MacOS assigns a volume reference number which remains valid as long as the volume is mounted. Macintosh files are uniquely specified with a volume reference number, a parent directory ID, and a file name.

A Macintosh file has two forks, a data fork and a resource fork. File data resides in the data fork, while file resources are stored in the resource fork. An application would typically have resources such as menus, dialog boxes, icons, and even code segments. A document file could have resources such as preference settings, window locations, fonts and icons. Files are marked with a four byte creator and type, usually shown as alphanumeric tags, such as APPL, TEXT, BINA. The creator identifies the application that created the file, and the type generally indicates the file content. Text files use the ASCII 'CR' character ('\r') to denote an end-of-line.

6.2.2 File Names

Macintosh file and folder names are limited to 31 characters. Names may not include the '**:**' character because it is used in pathnames to separate volume, directory, and the file name components. An absolute pathname begins with a volume name. A pathname relative to the current working directory begins with a '**:**'. The File Manager is not case sensitive when matching file names, however, it does not ignore diacritical marks. (MacOS file names are **NOT** case sensitive, however Tenon's extension enables case sensitivity in Tenon's UFS file system.)

6.2.3 Access Permissions

Permissions for files on local volumes are based on the order and manner in which programs open them. Files can be opened for reading, writing, or exclusive read/write. Write access can be denied on an individual file basis by setting its lock via Finder's Get Info dialog box. The File Manager also prevents locked files in the Trash can from being deleted on an" Empty Trash" request. Setting a volume to read-only (by hardware or software) prevents any change to all files on that volume.

6.2.4 Time Stamps

The Macintosh File Manager saves three time stamps for each file: (1) the time the file was created, (2) the time the file was last modified, and (3) the time the file was last backed up (rarely used in the Macintosh world).

6.2.5 Aliases

An alias is a special kind of file that represents a file, folder, or volume.

6.3 CodeBuilder FFS

CodeBuilder's implementation of the UNIX Fast File System (FFS) allows it to be stored within a Macintosh HFS file, or on a SCSI disk partition. Using files allows existing volumes to be used as is, provided they have sufficient space to meet file system sizing requirements. Using partitions provides higher performance, but may require reformatting/partitioning disks.

6.3.1 FFS Within a File

Macintosh files which contain an FFS are treated as block devices by CodeBuilder. They are set with creator 'MUMM' and type 'BLK '. These files are visible to Finder and may be moved or copied to other folders or volumes. They may also be dragged to the Trash can for removal. Be sure that this is intended before requesting "Empty Trash"! These files should never be selected for write access by other Macintosh applications. Even minor changes by other Macintosh applications could render the fast file system file unusable by CodeBuilder! See also section 6.9.1.1 Creating an FFS Within a File.

6.3.2 FFS Within a Partition

To establish an FFS in a partition, you must first create an A/UX partition a disk with a Macintosh formatting application. This partition cannot be mounted on the Macintosh desktop, so its content is not accessible by the

Finder or other Macintosh applications. See also section 6.9.1.2 Creating an FFS on a Partition.

6.4 CodeBuilder UFS

CodeBuilder's implementation of the local UNIX file system (UFS) uses the Macintosh File Manager (HFS), enabling CodeBuilder's UNIX files to be stored alongside Macintosh files. However, there are some constraints imposed by the Macintosh File Manager that the astute user should be aware of when porting other UNIX applications to CodeBuilder.

6.4.1 File Names

6.4.1.1 Maximum Number of Characters

The Macintosh File Manager allows a maximum of 31 characters in a file name; hence, CodeBuilder UFS file names are limited to 31 characters.

6.4.1.2 Case Sensitive File Names

CodeBuilder supports case sensitive file names by appending a 4 byte integer to the end of any case sensitive file name that collides with an existing case insensitive name. HFS stores the length of the file name, hence the names are unique within HFS. UNIX requests use a null terminated string comparison for identifying files, hence the appropriate file is matched as long as the first of the four appended bytes is NULL. CodeBuilder guarantees that this is the case. For subsequent collisions, a new integer is used.

Since the HFS limit on file name length is 31 characters, this method of resolving case sensitivity is limited to file names of length 27 characters or less. Attempting to create a new file that collides with an existing case insensitive file name of greater than 27 characters will fail.

6.4.1.3 Component Separators

The Macintosh File Manager uses the character ":" to separate names into volume, folder, and file name components. Thus, Macintosh file names cannot contain that character. UNIX uses the character "/" for the same purpose. CodeBuilder automatically maps all occurrences of ":" (which is valid in UNIX file names) into "/" (which is valid in Macintosh file names) before passing UNIX file names to the Macintosh File Manager.

6.4.1.4 Non-Printable Characters

Macintosh file names often include unprintable characters (the trademark symbol, for example). Although these characters are not invalid in UNIX file names, they are impossible to type or display on a UNIX command line. For this reason, non-printable ASCII characters in Macintosh file names are translated according to the

AppleSingle 7-bit ASCII naming convention[Ý]. Basically, non-printable characters are translated to a percent sign (%) followed by a 2 digit hexadecimal representation of the character's value. The following are some examples:

File Name as Viewed by MacOS	File Name as Viewed by CodeBuilder
Filename™	Filename%aa
Filename®	Filename%a8
Filename©	Filename%a9

Ý: For more information on Apple Single, refer to section 6.7.1 AppleSingle Encapsulation.

6.4.2 Linked Files

In UNIX, linking allows several file names to be associated with the same physical file. CodeBuilder properly supports UNIX hard and soft links via the *link(2)* and *symlink(2)* system calls.

6.4.2.1 Hard Links

CodeBuilder provides a file system paradigm that permits copying or moving either UNIX or Macintosh files with either UNIX or Macintosh tools with equal results. However, the UNIX implementation of hard links prohibits cross disk hard links. Using the Finder to copy CodeBuilder's hard links from one volume to another will not work. Therefore, you cannot use the Finder (or any other Macintosh tool) to copy or move CodeBuilder hard links from one Macintosh volume to another. This restriction is similar to the traditional UNIX restriction - no cross disk hard links.

CodeBuilder's implementation of hard links uses a hidden folder at the *root* level of a volume, and uses the Macintosh HFS equivalent of *inode* numbers to find targets of hard links in this folder. When the source of a hard link is moved to another volume, the link will be unresolvable. Even if the correct target of the link is copied from the source volume's hidden folder, it will be assigned a new *inode* number and the source link will still be unresolvable.

In general, unless an application requires specific semantics of hard links that are not also supplied by soft links, soft links should be the preferred method of linking UFS files.

6.4.3 Directory Link Counts

The *stat()* system call does not return the correct number of links for a directory. The Macintosh File Manager does not keep a directory link count for Macintosh folders, and CodeBuilder does not attempt to fabricate a correct directory link count for UNIX directories on local Macintosh File Systems[Ý].Ý: CodeBuilder always returns a link count of 2 for UFS directories.

6.4.4 Locked Files

The Macintosh File System has the ability to lock individual files, effectively making the file 'read only'. The UNIX file system has no notion of locking for individual files, although it does support the locking of entire file systems by mounting them as 'read only'. If a UNIX application attempts to modify or remove a locked file, CodeBuilder will return the error EROFS. Although this error implies that the entire file system is 'read only', it may be only the individual file that is locked. In this case it is necessary to unlock the file (use the Finder's Get Info entry in the File menu) before CodeBuilder can modify or remove the file.

6.4.5 File Types

The Macintosh File Manager saves a file type and file creator as part of the attributes for every file. CodeBuilder uses the identifier 'MUMM' for UNIX files. In CodeBuilder documentation the term "UNIX file" is used to refer to any file with a creator 'MUMM' and the term "Macintosh file" is used to refer to all other files. Note that Macintosh applications can see and access the UNIX files, since they are simply Macintosh files with a creator 'MUMM'. Only CodeBuilder makes the distinction between UNIX files and Macintosh files.

All CodeBuilder documents and files are owned by the application named *CodeBuilder*. CodeBuilder uses file types for the different classes of files in UNIX. Each file type has a unique icon, easing identification of UNIX files when using the Macintosh Finder.

BINA - Binary Files

Binary files are streams of data. They may be UNIX executable files, text files, database files, or others. No translation of the data is done on binary files when accessed via CodeBuilder. This is the simplest, fastest, and most common type of file. When CodeBuilder creates a new file, it always creates files of type BINA.

TEXT - Text Files

Files containing only ASCII printable characters are referred to as "text" files. The Macintosh and UNIX formats for text files have subtle differences. Macintosh uses the character 'cr' (0xD) to terminate a line of text and UNIX uses the character 'nl' (0xA). CodeBuilder files with a type 'TEXT' are stored on the disk in the native Macintosh text file format (the line terminator is 'cr').

When UNIX applications access type TEXT files, CodeBuilder automatically translates the 'cr' character to 'nl', so the UNIX applications see these files in the expected UNIX text file format. A result of this automatic translation is that both Macintosh word processors and UNIX editors can be used to edit 'MUMM'/'TEXT' files. Note that some Macintosh word processors may inadvertently add resource forks to the UNIX text files, but these resource forks will always be ignored by CodeBuilder. For more information on the handling of text files in CodeBuilder, refer to the chapter "Text File Manipulation".

LINK - Symbolic Links

This file type is used to identify UNIX symbolic links. The path to follow for the symbolic link is stored in the data fork of the file.

 **CHR - Character Devices**

This file type is used to identify UNIX character devices. UNIX assigns two numbers to each device. The 'major' number describes the device type, and the 'minor' number uniquely identifies the device. The major and minor numbers for this device are packed in four bytes and are stored in the data fork of the file.

 **BLK - Block Devices**

This file type is used to identify UNIX block devices. The major and minor numbers for this device are packed in four bytes and are stored in the data fork of the file.

 **SOCK - Sockets**

This file type is used to identify UNIX sockets. Sockets are communication endpoints used for sending and receiving data.

 **FIFO - Pipes**

This file type is used to identify named UNIX pipes. A pipe is a special type of file that is created by UNIX processes in order to pass information to other processes. Pipes enforce a first-in, first-out (FIFO) mechanism on data.

 **HLNK - Hard Links**

This file type is used to identify hard links. Hard linking allows several file names to be associated with the same physical file.

 **SHLB - Shared Libraries**

This file type is used to identify a shared library file. Many of the traditional UNIX shared libraries (*libc*, *libm*, etc.) will have two versions in CodeBuilder - one for the compile time Header Call definitions and the other for the run time dynamic linking.

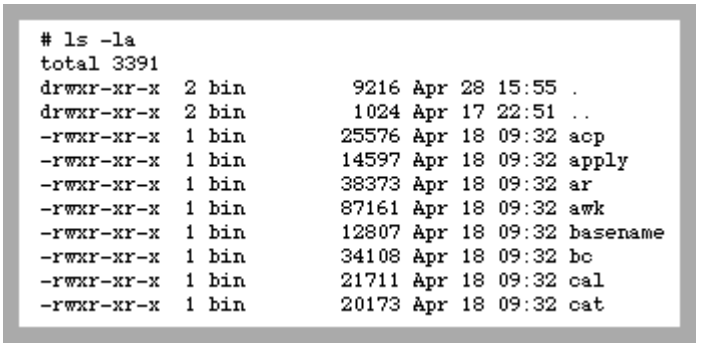
6.4.6 File Permissions

CodeBuilder stores the user id, group id, and mode bits in a reserved area of the resource fork for each UNIX file. CodeBuilder can also set these attributes for Macintosh files[Ý]. Note that this protection only applies to file access

from CodeBuilder; it does not apply to the Finder or other Macintosh applications.

CodeBuilder stores the protection attributes for a directory in a special file within the folder. This file has the name '

/' and is invisible to the Finder. However, other Macintosh applications may see this file. It should not be deleted or changed.



```
# ls -la
total 3391
drwxr-xr-x  2 bin         9216 Apr 28 15:55 .
drwxr-xr-x  2 bin         1024 Apr 17 22:51 ..
-rwxr-xr-x  1 bin        25576 Apr 18 09:32 acp
-rwxr-xr-x  1 bin        14597 Apr 18 09:32 apply
-rwxr-xr-x  1 bin        38373 Apr 18 09:32 ar
-rwxr-xr-x  1 bin        87161 Apr 18 09:32 awk
-rwxr-xr-x  1 bin        12807 Apr 18 09:32 basename
-rwxr-xr-x  1 bin        34108 Apr 18 09:32 bc
-rwxr-xr-x  1 bin        21711 Apr 18 09:32 cal
-rwxr-xr-x  1 bin        20173 Apr 18 09:32 cat
```

Figure 20. Example of File Permissions

Ý: The CodeBuilder application has resources which allow the user to configure the default UNIX access privilege modes for Macintosh files and folders; see the CodeBuilder Resources Appendix B.

6.4.7 Time Stamps

When CodeBuilder examines the time stamps of a Macintosh file, the file's Macintosh creation time is used for the UNIX data modification time. The file's Macintosh modification time is used for the UNIX attribute modification times. The file's Macintosh last-backed-up time is used for the UNIX last-accessed-time. When CodeBuilder modifies the time stamps of a Macintosh file, the file's Macintosh creation time is set to the UNIX data modification time. The file's Macintosh last-backed-up time is set to the UNIX last-accessed-time. The file's Macintosh modification time is set to the UNIX attribute modification time.

File Time Stamps

Macintosh	**UNIX**
file created	data last modified
file modified	attributes last modified
file backed up	data last accessed

Since most of the interaction between UNIX and Macintosh programs depends on only the "modified" timestamp, the CodeBuilder interpretation of the Macintosh timestamps provides Mac/UNIX interoperability. For example, changing a source file with a UNIX tool (like *vi*) results in a change to the file's "modified" timestamp. A Macintosh application which subsequently accesses this file will notice the change in the modified time, and will behave accordingly (e.g., a Macintosh development tool will rebuild a project if the project depends on the modified source file). In the reverse situation, a Macintosh editor will also change the file's "created" and

"modified" timestamp, which will be interpreted by UNIX programs in the proper, "the-data-(or attribute)-has-been-modified" sense. In some cases, the Macintosh editor may only change the UNIX attribute modification time. In this case, it may be necessary to *touch(8)* the file from within CodeBuilder to rebuild the project correctly.

6.5 CodeBuilder Root File System Layout

The CodeBuilder root file system is delivered as an FFS within a file. The FFS file is named *CodeBuilder_FFS*, and resides in the same folder as the CodeBuilder application. A companion folder named *CodeBuilder_HFS* provides for UFS oriented storage.

At the top of the CodeBuilder File System hierarchy is the UNIX root directory, also referred to as "

/". The path "//" is a special case in CodeBuilder that allows UNIX processes to refer to files and folders outside of the CodeBuilder root. "//" is interpreted to mean "the real root of the HFS volume that contains the CodeBuilder file system hierarchy". The corresponding routines in the CodeBuilder libraries (*getwd(3)*) have been modified to understand a pathname that begins with "//".

6.5.1 The *root* Directory Tree

The *root* directory is represented by a "/" and contains the following directories:

Name	**Description**
CDROM	Mount point for CodeBuilder CD-ROM.
base	Base of source code and place for building binaries in CodeBuilder distribution.
bin	Basic user utilities.
bootvol	Symbolic link or mount point for the root of the MacOS boot volume.
dev	Block, character and other special device files.
etc	System configuration files and scripts..
hfs	Symbolic link to CodeBuilder_HFS
lost+found	Storage for files lost or corrupted by file system damage.
bin	Basic user utilities.
mnt	Temporary mount point.
sbin	Basic system administration utilities.
tmp	Directory for temporary files.
usr	Contains majority of system utilities and files.

var	Multi-purpose log, temporary, transient and spool files.
volume	*macmntd* creates subdirectory mount points for removable volumes.

6.5.2 The *usr* Directory Tree

Name	Description
X11	Symbolic link to CodeBuilder X Window software.
X11R6	CodeBuilder X11R6 software.
bin	Binaries.
doc	Miscellaneous documentation.
include	*include* files.
info	Documentation for GNU programs.
lib	Libraries.
libexec	Catchall for system daemons.
local	Mount point for file system containing local applications and support files.
macppc	PowerPC-specific files go here.
man	*troff* source for man pages.
sbin	System administrator binary files.
share	Text and database files readily shared among 4.4BSD systems.
src	*rpm* support tree.

6.5.3 The *var* Directory Tree

Name	Description
adm	Administrative files.
at	Timed command scheduling files.
backups	Miscellaneous backup files.
cron	cron data files.
db	Database files.

games	Miscellaneous game status and log files.
lib	Dynamic, machine specific application support files.
lock	Storage for emacs file locking.
log	Miscellaneous system log files.
msgs	System messages.
obj	Object files.
preserve	Temporary home of files preserved when editors fail.
run	System information files, rebuilt after each reboot.
spool	Miscellaneous printer spooling directories.
tmp	Temporary files that are not discarded between system reboots.

6.5.4 Major System Administration Files

For details about these files, see on-line man pages.

Name	**Description**
/etc/fstab	Contains file system mount information [*fstab(5)*].
/etc/group	Defines mapping of group numbers to names and authorized users [*group(5)*].
/etc/hosts	Defines host names, aliases and IP address.
/etc/master.passwd	Defines the set of users authorized to log in; provides mapping between user id and name, default group, home directory, and shell [*passwd(5)*].
/etc/printcap	Contains the list of available printers and the parameters used by the line printer daemon [*printcap(5)*].
/etc/rc	Contains the script of commands *init(8)* is to execute to establish the desired environment [*rc(8)*] .
/etc/rc.conf	Contains variables derived from information entered in the CodeBuilder Control Panel for use by */etc/rc* when starting CodeBuilder.
/etc/syslog.conf	*/usr/sbin/syslogd* configuration file [*syslog(8)*].
/etc/ttys	Specifies terminal support information for *init(8)* [*ttys(5)*].
/usr/share/misc/termcap	Defines the capabilities for numerous terminal types [*termcap(5)*]. Being phased out by *ncurses*.

[Top of Page][6.6 Mounting Macintosh Volumes][Table of Contents]
