HFS Utilities

Complete Technical Manual

HFS and HFS+ Filesystem Utilities for Unix Systems

hfsutils Project

December 17, 2025

Version 3.2.6

Copyright and License

Contents

Chapter 1

Introduction

1.1 Overview

The HFS Utilities (hfsutils) project provides a comprehensive set of tools for working with Apple's HFS (Hierarchical File System) and HFS+ (HFS Plus) filesystems on Unix-like operating systems including Linux, BSD, and macOS.

1.1.1 Purpose and Goals

HFS and HFS+ filesystems are commonly used on:

- Classic Mac OS systems (HFS)
- Mac OS X and macOS systems prior to APFS (HFS+)
- iPod devices
- External drives formatted for Mac compatibility
- Disk images and backups from Apple systems

This toolset enables Unix systems to:

- **Create** HFS and HFS+ filesystems with `mkfs.hfs` and `mkfs.hfs+`
- **Check and repair** filesystem integrity with `fsck.hfs` and `fsck.hfs+`
- **Mount** HFS and HFS+ volumes with `mount.hfs` and `mount.hfs+` (requires kernel support)
- **Manipulate files** on HFS volumes without mounting using `hfsutil` commands (useful on systems without HFS kernel drivers)

1.1.2 Supported Systems

The utilities work on any POSIX-compliant system with appropriate kernel support:

Note: On systems without kernel HFS support, the `hfsutil` commands can still be used to access HFS filesystem contents.

System	HFS Support	HFS+ Support
Linux	Kernel module `hfs`	Kernel module `hfsplus`
FreeBSD	Native support	Native support
macOS	Native support	Native support
OpenBSD	Via FUSE	Via FUSE
NetBSD	Via FUSE	Via FUSE

Table 1.1: Platform Support Matrix

1.1.3 Key Features

Full Specification Compliance

All utilities strictly adhere to:

- Inside Macintosh: Files (HFS specification)
- Apple Technical Note TN1150 (HFS+ specification)
- Unix/POSIX filesystem utility standards

Zero-Tolerance Validation

The test suite implements a "zero-tolerance" policy:

- ANY deviation from specification = test failure
- ALL filesystems must be 100% correct
- Complete validation before and after fsck operations

Journaling Support

HFS+ journaling is supported with appropriate warnings:

- Journal creation with `mkfs.hfs+ -j`
- Journal validation and replay in `fsck.hfs+`
- Linux kernel compatibility warnings (journaling not supported in Linux HFS+ driver)

Date Limit Awareness

The utilities handle filesystem date limits correctly:

- HFS: Maximum date February 6, 2028 (Y2K28 problem)
- HFS+: Maximum date February 6, 2040 (Y2K40 problem)
- Automatic date correction to safe values

1.2 Installation

1.2.1 Building from Source

Prerequisites

```
# Debian/Ubuntu
sudo apt-get install build-essential

# Fedora/RHEL
sudo dnf install gcc make

# macOS
xcode-select --install

# BSD
# Compiler included by default
```

Compilation

```
# Clone repository
git clone https://github.com/JotaRandom/hfsutils.git
cd hfsutils

# Build all utilities
make

# Build specific sets
make set-hfs         # mkfs.hfs, fsck.hfs, mount.hfs
make set-hfsplus     # mkfs.hfs+, fsck.hfs+, mount.hfs+

# Build with hfsutil
make all
```

Installation Options

Linux systems (with kernel HFS/HFS+ drivers):

```
sudo make install-linux PREFIX=/usr
```

Complete installation (filesystem utilities + hfsutil):

```
sudo make install-complete PREFIX=/usr/local
```

Individual utilities:

```
sudo make install-mkfs.hfs+ PREFIX=/usr
sudo make install-fsck.hfs PREFIX=/usr
sudo make install-mount.hfs+ PREFIX=/usr
```

1.2.2 Verifying Installation

After installation, verify the utilities are available:

```
mkfs.hfs --version
mkfs.hfs+ --version
fsck.hfs --version
```

```
fsck.hfs+ --version
mount.hfs --help
mount.hfs+ --help
hfsutil --version   # If installed
```

Check manpages:

```
man mkfs.hfs
man mkfs.hfs+
man fsck.hfs+
man mount.hfs+
```

1.3 Quick Start

1.3.1 Creating a Filesystem

Create an HFS filesystem:

```
# Create 10MB image file
dd if=/dev/zero of=test.img bs=1M count=10

# Format as HFS
mkfs.hfs -L "MyDisk" test.img
```

Create an HFS+ filesystem:

```
# Create 50MB image file
dd if=/dev/zero of=test.img bs=1M count=50

# Format as HFS+
mkfs.hfs+ -L "MyDisk" test.img

# Format with journaling (Linux warning will appear)
mkfs.hfs+ -j -L "MyDisk" test.img
```

1.3.2 Checking a Filesystem

Verify filesystem integrity:

```
# Check HFS filesystem (read-only)
fsck.hfs -n test.img

# Check and repair HFS+ filesystem
fsck.hfs+ -y test.img

# Verbose output
fsck.hfs+ -v -n test.img
```

1.3.3 Mounting a Filesystem

On systems with kernel HFS/HFS+ support:

```
# Create mount point
sudo mkdir /mnt/hfs

# Mount read-write
sudo mount.hfs+ test.img /mnt/hfs
```

```
6
7  # Mount read-only
8  sudo mount.hfs+ -r test.img /mnt/hfs
9
10 # Unmount
11 sudo umount /mnt/hfs
```

1.3.4 Using hfsutil Commands

On systems without kernel support or for direct access:

```
1  # Format volume
2  hfsutil hformat -L "MyDisk" test.img
3
4  # Mount in hfsutil
5  hfsutil hmount test.img
6
7  # List contents
8  hfsutil hls
9
10 # Copy file into volume
11 hfsutil hcopy myfile.txt :myfile.txt
12
13 # Copy file out of volume
14 hfsutil hcopy :myfile.txt retrieved.txt
15
16 # Unmount
17 hfsutil humount
```

1.4 Documentation Structure

This manual is organized as follows:

Chapter 2: HFS Specification Details of the classic HFS filesystem format

Chapter 3: HFS+ Specification Details of the HFS+ filesystem format with journaling

Chapter 4: mkfs Utilities Filesystem creation tools

Chapter 5: fsck Utilities Filesystem checking and repair tools

Chapter 6: mount Utilities Filesystem mounting tools

Chapter 7: hfsutil Commands File manipulation without mounting

Chapter 8: Implementation Details Internal architecture and algorithms

Chapter 9: Testing and Validation Test suite and quality assurance

Chapter 10: Appendix Structure definitions, error codes, glossary

Chapter 2

HFS Specification

2.1 HFS Filesystem Overview

The Hierarchical File System (HFS) is the filesystem used by Apple Computer for Mac OS systems from 1985 until Mac OS X. Also known as "Mac OS Standard" or "HFS Classic", it provides a hierarchical directory structure with support for file and folder metadata.

2.1.1 Key Characteristics

- **Maximum volume size**: 2 TB
- **Maximum file size**: 2 GB
- **Filename length**: 31 characters (Macintosh Roman encoding)
- **Date range**: January 1, 1904 to February 6, 2028 (Y2K28 limit)
- **Allocation block size**: 512 bytes to 64 KB
- **Case sensitivity**: Case-insensitive, case-preserving

2.1.2 Volume Structure

An HFS volume is divided into logical blocks (512 bytes each) and allocation blocks (multiples of logical blocks).

Block	Name	Description
0-1	Boot Blocks	System startup information (1024 bytes)
2	Master Directory Block	Volume metadata and B-tree locations
3+	Allocation Bitmap	Block allocation map
...	Extents B-tree	File extent records
...	Catalog B-tree	Directory and file records
...	Data Area	File contents
Last-1	Alternate MDB	Backup of Master Directory Block

Table 2.1: HFS Volume Layout

2.2 Master Directory Block

The Master Directory Block (MDB) is located at logical block 2 (offset 1024 bytes) and contains critical volume information.

2.2.1 MDB Structure

Field	Offset	Description
drSigWord	+0	Signature (0x4244 = 'BD')
drCrDate	+2	Volume creation date
drLsMod	+6	Last modification date
drAtrb	+10	Volume attributes (bit 8 = unmounted cleanly)
drNmFls	+12	Number of files in root directory
drVBMSt	+14	First allocation bitmap block
drAllocPtr	+16	Next unused allocation block
drNmAlBlks	+18	Number of allocation blocks
drAlBlkSiz	+20	Allocation block size (bytes)
drClpSiz	+24	Default clump size
drAlBlSt	+26	First allocation block
drNxtCNID	+28	Next unused Catalog Node ID
drFreeBks	+32	Number of free allocation blocks
drVN	+36	Volume name (1-27 characters, Pascal string)
drVolBkUp	+64	Last backup date
drVSeqNum	+68	Volume backup sequence number
drWrCnt	+70	Volume write count
drXTClpSiz	+72	Extents B-tree clump size
drCTClpSiz	+76	Catalog B-tree clump size
drNmRtDirs	+80	Number of directories in root
drFilCnt	+82	Number of files on volume
drDirCnt	+86	Number of directories on volume
drFndrInfo	+90	Finder information (32 bytes)
drVCSize	+122	Volume cache size
drVBMCSize	+124	Volume bitmap cache size
drCtlCSize	+126	Common volume cache size
drXTFlSize	+128	Extents B-tree file size
drXTExtRec	+132	Extents B-tree extents (3 extents)
drCTFlSize	+144	Catalog B-tree file size
drCTExtRec	+148	Catalog B-tree extents (3 extents)

Table 2.2: Master Directory Block Fields

2.2.2 Critical MDB Fields

drSigWord (Signature)

- Value: `0x4244` ('BD' in ASCII)
- Identifies the volume as HFS
- **Validation**: Must be exactly `0x4244`

drAtrb (Attributes)

Bit flags indicating volume state:

- Bit 8 (0x0100): Volume unmounted properly

- Bit 9 (0x0200): Spare blocks exist
- Bit 10 (0x0400): Volume needs consistency check
- Bit 15 (0x8000): Software lock

Important: mkfs.hfs sets bit 8 to indicate clean unmount.

drNxtCNID (Next Catalog Node ID)

- Minimum value: 16
- IDs 1-15 are reserved:
  - 1 = Parent of root directory
  - 2 = Root directory
  - 3 = Extents B-tree file
  - 4 = Catalog B-tree file
  - 5 = Bad block file
- **Validation**: mkfs.hfs initializes to 16

2.2.3 Alternate MDB

The alternate MDB is a complete copy of the MDB located at:

$$AlternateMDBoffset = volume\_size - 1024bytes \tag{2.1}$$

Purpose: Provides recovery if the primary MDB is corrupted.
Validation: fsck.hfs verifies both MDBs match.

2.3 B-Trees

HFS uses B-trees for efficient file and directory lookups.

2.3.1 Extents B-Tree

Stores file extent records (physical block ranges):

- Maps file portions to disk blocks
- Handles fragmented files
- Key: File CNID + fork type + start block

2.3.2 Catalog B-Tree

Stores directory and file records:

- Directory entries
- File metadata (type, creator, dates, etc.)
- Key: Parent directory CNID + filename

2.4 Date Representation

HFS dates are 32-bit unsigned integers representing seconds since:

January 1, 1904 00:00:00 GMT

2.4.1 Y2K28 Problem

Maximum representable date:

$$1904 + 2^{32}/(365.25 \times 24 \times 3600) \approx 2040 \tag{2.2}$$

However, **HFS specification limits dates to February 6, 2028**.

Implementation: mkfs.hfs and fsck.hfs use `hfs_get_safe_time()` to ensure dates are within valid range.

2.5 Allocation Strategy

2.5.1 Allocation Blocks

Files are allocated in multiples of allocation blocks:

- Allocation block size determined by volume size
- Typical sizes: 512 B, 1 KB, 2 KB, 4 KB
- Larger blocks = less overhead, more waste for small files

2.5.2 Clump Size

Default allocation size when extending files:

- Reduces fragmentation
- Calculated as multiple of allocation blocks
- Stored in drClpSiz (MDB)

2.6 Compatibility Notes

2.6.1 Character Encoding

HFS uses Macintosh Roman encoding:

- 8-bit encoding
- Incompatible with UTF-8
- Characters >127 map differently than ISO-8859-1

Limitation: Filenames with non-ASCII characters may not display correctly on non-Mac systems.

2.6.2 Resource Forks

HFS supports dual-fork files:

- **Data fork**: Regular file contents
- **Resource fork**: Application-specific data

Note: Most Unix systems only support data forks. Resource forks are preserved but not directly accessible.

2.6.3 Type and Creator Codes

HFS files have 4-character type and creator codes:

- Type: File type (e.g., 'TEXT', 'PICT')
- Creator: Creating application (e.g., 'MSWD' for Microsoft Word)

Modern usage: Largely replaced by filename extensions.

Chapter 3

HFS+ Specification

3.1 HFS+ Filesystem Overview

HFS Plus (HFS+), also known as Mac OS Extended, is Apple's modern filesystem designed to replace HFS. Introduced in Mac OS 8.1 (1998), it addresses HFS limitations while maintaining backward compatibility.

3.1.1 Key Improvements over HFS

- **Unicode filenames**: Full Unicode support (up to 255 UTF-16 characters)
- **Larger volumes**: Up to 8 EB (exabytes) theoretical
- **Larger files**: Up to 8 EB per file
- **Smaller allocation blocks**: More efficient space usage
- **Journaling support**: Optional transaction journal for crash recovery
- **Hard links**: Support for hard links (Mac OS X 10.2+)
- **Symbolic links**: Support for symbolic links
- **Extended attributes**: Arbitrary metadata on files/folders
- **Case sensitivity option**: HFSX variant supports case-sensitive names

3.1.2 HFS+ Characteristics

Feature	Specification
Maximum volume size	8 EB (2^{63} bytes)
Maximum file size	8 EB (2^{63} bytes)
Filename length	255 Unicode characters (UTF-16)
Date range	January 1, 1904 to February 6, 2040 (Y2K40)
Minimum allocation block	512 bytes
Block addressing	32-bit allocation block numbers
Case sensitivity	Case-insensitive (HFS+) or case-sensitive (HFSX)

Table 3.1: HFS+ Specifications

3.2 Volume Structure

HFS+ volumes are structured similarly to HFS but with enhanced metadata structures.

3.2.1 Volume Layout

Location	Name	Description
Byte 0	Reserved	Boot sector (512 bytes)
Byte 512	Reserved	Additional boot area (512 bytes)
Byte 1024	Volume Header	Primary volume metadata
...	Allocation Bitmap	Volume space allocation map
...	Allocation File	B-tree of allocation bitmap extents
...	Extents Overflow File	B-tree of file extent records
...	Catalog File	B-tree of all files and folders
...	Attributes File	B-tree of extended attributes
...	Startup File	Boot loader for non-Mac systems
...	Data Area	File contents and special files
-1024	Alternate VH	Backup copy of Volume Header

Table 3.2: HFS+ Volume Layout

3.3 Volume Header

The Volume Header is the primary metadata structure, located at byte offset 1024 from the start of the volume.

3.3.1 Volume Header Structure

Field	Offset	Description
signature	+0	Volume signature (0x482B = 'H+' for HFS+, 0x4858 = 'HX' for HFSX)
version	+2	Version number (4 for HFS+, 5 for HFSX)
attributes	+4	Volume attributes (see below)
lastMountedVersion	+8	Signature of OS that last mounted volume
journalInfoBlock	+12	Starting block of journal info
createDate	+16	Volume creation date (HFS+ time)
modifyDate	+20	Last modification date
backupDate	+24	Last backup date
checkedDate	+28	Last fsck date
fileCount	+32	Number of files on volume
folderCount	+36	Number of folders on volume
blockSize	+40	Allocation block size (bytes)
totalBlocks	+44	Total allocation blocks
freeBlocks	+48	Free allocation blocks
nextAllocation	+52	Next unused allocation block

Field	Offset	Description
rsrcClumpSize	+56	Default resource fork clump size
dataClumpSize	+60	Default data fork clump size
nextCatalogID	+64	Next unused Catalog Node ID
writeCount	+68	Volume write count
encodingsBitmap	+72	Text encodings used (64 bits)
finderInfo	+80	Finder information (32 bytes)
allocationFile	+112	Fork data for allocation file (80 bytes)
extentsFile	+192	Fork data for extents file (80 bytes)
catalogFile	+272	Fork data for catalog file (80 bytes)
attributesFile	+352	Fork data for attributes file (80 bytes)
startupFile	+432	Fork data for startup file (80 bytes)

Table 3.3: Volume Header Fields (total size: 512 bytes)

3.3.2 Critical Volume Header Fields

signature

- **HFS+**: `0x482B` ('H+' in ASCII)
- **HFSX**: `0x4858` ('HX' in ASCII)
- Identifies volume type
- **Validation**: mkfs.hfs+ sets to `0x482B`

version

- **HFS+**: 4
- **HFSX**: 5
- Indicates filesystem variant
- **Validation**: mkfs.hfs+ sets to 4

attributes (Volume Attributes)

32-bit flags field:

- Bit 0-6: Reserved
- Bit 7 (0x0080): Volume is locked by hardware
- Bit 8 (0x0100): Volume unmounted cleanly
- Bit 9 (0x0200): Volume has bad blocks
- Bit 10 (0x0400): Volume needs consistency check
- Bit 11 (0x0800): Catalog IDs wrapped around
- Bit 12 (0x1000): Unused node fix required

- Bit 13 (0x2000): Volume is journaled
- Bit 14 (0x4000): Volume is software locked
- Bit 15-31: Reserved

Important: mkfs.hfs+ sets bit 8 (0x0100) to indicate clean unmount.

nextCatalogID

- Minimum value: 16
- Reserved IDs 1-15:
  - 1 = Root folder's parent
  - 2 = Root folder
  - 3 = Extents overflow file
  - 4 = Catalog file
  - 5 = Bad block file
  - 6 = Allocation bitmap file
  - 7 = Startup file
  - 8 = Attributes file
  - 14 = Journal file
  - 15 = Journal info block
- **Validation**: Must be ¿= 16

rsrcClumpSize and dataClumpSize

- Default allocation sizes for extending forks
- **Must be non-zero**
- Typically set to `blockSize * 4`
- Reduces fragmentation
- **Validation**: mkfs.hfs+ initializes both correctly

3.3.3 Alternate Volume Header

Located at the second-to-last sector of the volume:

$$AlternateVHoffset = volume\_size - 1024bytes \tag{3.1}$$

Purpose: Recovery if primary Volume Header is corrupted.
Maintenance: Updated simultaneously with primary VH.

3.4 HFS+ B-Trees

HFS+ uses B-trees extensively for efficient data organization. All B-trees follow the same structure but contain different record types.

3.4.1 B-Tree Structure

Each B-tree consists of:

- **Header node**: B-tree metadata (first node, index 0)
- **Map nodes**: Allocation bitmap for B-tree nodes
- **Index nodes**: Internal nodes with pointers to child nodes
- **Leaf nodes**: Contain actual data records

3.4.2 B-Tree Header Record

Located in the first node of each B-tree:

Field	Offset	Description
treeDepth	+0	Current depth of tree (1-based)
rootNode	+2	Node number of root node
leafRecords	+4	Number of leaf records
firstLeafNode	+8	Node number of first leaf
lastLeafNode	+12	Node number of last leaf
nodeSize	+16	Size of each node (bytes)
maxKeyLength	+18	Maximum key length
totalNodes	+20	Total nodes in tree
freeNodes	+24	Number of free nodes
reserved1	+28	Reserved
clumpSize	+32	Clump size for file
btreeType	+36	Type of B-tree (0=HFS, 128=HFS+)
keyCompareType	+37	Key comparison method
attributes	+38	B-tree attributes
reserved3	+42	Reserved (76 bytes)

Table 3.4: B-Tree Header Record

3.4.3 Catalog File

The Catalog File is a B-tree containing all files and folders on the volume.

Catalog Keys

- **keyLength**: 2 bytes
- **parentID**: 4 bytes (Parent folder CNID)
- **nodeName**: Variable-length Unicode string

Catalog Record Types

1. **Folder Record (0x0001)**: Directory metadata
2. **File Record (0x0002)**: File metadata and fork information
3. **Folder Thread (0x0003)**: Maps CNID to parent folder
4. **File Thread (0x0004)**: Maps file CNID to parent folder

File Record Structure

- recordType: 0x0002
- flags: File flags
- fileID: Catalog Node ID
- createDate, modifyDate: Timestamps
- permissions: Unix permissions
- userInfo, finderInfo: Finder metadata
- textEncoding: Filename encoding hint
- dataFork: Fork data for data fork (80 bytes)
- resourceFork: Fork data for resource fork (80 bytes)

3.4.4 Extents Overflow File

B-tree storing additional extent records when a file's fork exceeds the 8 extents stored in the catalog.

Extent Key

- keyLength: 2 bytes
- forkType: 0x00 (data) or 0xFF (resource)
- fileID: Catalog Node ID
- startBlock: Starting allocation block

Extent Descriptor

- startBlock: Starting allocation block (4 bytes)
- blockCount: Number of contiguous blocks (4 bytes)

3.4.5 Attributes File

B-tree storing extended attributes (metadata) for files and folders.

Supported Attributes

- Extended attributes (xattrs)
- Access Control Lists (ACLs)
- Resource fork data (if not inline)
- Compressed data (HFS+ compression)

3.5 Journaling

HFS+ supports optional journaling for filesystem consistency after crashes.

3.5.1 Journal Structure

- **Journal Info Block**: Located at block specified in Volume Header
- **Journal Buffer**: Circular buffer of transactions
- **Transaction Records**: Logged filesystem changes

3.5.2 Journal Operation

1. **Begin Transaction**: Start logging changes
2. **Log Metadata Changes**: Write changes to journal buffer
3. **Commit Transaction**: Mark transaction complete
4. **Apply Changes**: Write changes to filesystem
5. **Release Transaction**: Free journal space

3.5.3 Journal Replay

After unclean unmount:

1. fsck.hfs+ detects journal
2. Scans journal for uncommitted transactions
3. Replays committed but unapplied transactions
4. Marks volume clean

3.5.4 Linux Compatibility Warning

CRITICAL: The Linux HFS+ kernel driver does NOT support journaling.

- Journaled volumes may mount read-only automatically
- Journal changes are ignored
- Risk of data corruption on unclean shutdown
- fsck.hfs+ can replay journal, but Linux won't maintain it

Recommendation: For Linux systems, create HFS+ volumes without journaling (omit `-j` option in mkfs.hfs+).

3.6 Date Representation

HFS+ uses 32-bit unsigned integers for dates:

Seconds since January 1, 1904 00:00:00 GMT

3.6.1 Y2K40 Problem

Maximum date with 32-bit unsigned:

$$1904 + \frac{2^{32}}{365.25 \times 24 \times 3600} \approx February 6, 2040 \tag{3.2}$$

Implementation: hfsutils uses `hfs_get_safe_time()` to ensure dates stay within valid range.

3.7 Unicode Filenames

HFS+ stores filenames as UTF-16 (fully decomposed).

3.7.1 Normalization

HFS+ uses a special Unicode normalization similar to NFD:

- Fully decomposed (e.g., é → e + combining acute)
- Case-insensitive comparison (HFS+) or case-sensitive (HFSX)
- Maximum 255 UTF-16 code units

3.7.2 Character Restrictions

Filenames cannot contain:

- Colon (:) - path separator
- NULL character

3.8 HFS+ vs HFSX

HFSX is a variant of HFS+ with case-sensitive filename comparison.

Feature	**HFS+**	**HFSX**
Signature	0x482B ('H+')	0x4858 ('HX')
Version	4	5
Case sensitivity	No	Yes
Filename comparison	Case-insensitive	Case-sensitive
"File.txt" = "file.txt"	Yes	No

Table 3.5: HFS+ vs HFSX

Note: hfsutils mkfs.hfs+ creates standard HFS+ volumes (case-insensitive). HFSX support is not currently implemented.

3.9 Compatibility Considerations

3.9.1 macOS

- Native support for HFS+ and HFSX
- Full journaling support
- All extended attributes supported
- APFS is now default (macOS 10.13+)

3.9.2 Linux

- Kernel module `hfsplus` required
- **No journaling support in kernel driver**
- Basic read/write support
- Some extended attributes supported
- May mount journaled volumes read-only for safety

3.9.3 FreeBSD/OpenBSD/NetBSD

- Native HFS+ read support
- Write support via FUSE
- No journaling support

3.9.4 Windows

- No native HFS+ support
- Third-party drivers available (Paragon, MacDrive)
- Boot Camp drivers provide read-only access

Chapter 4

mkfs Utilities

4.1 mkfs.hfs - HFS Filesystem Creation

This chapter will include complete mkfs.hfs and mkfs.hfs+ documentation from manpages.

4.1.1 Command Synopsis

```
mkfs.hfs [-f] [-L label] device
mkfs.hfs+ [-f] [-j] [-L label] [-s size] device
```

Note: Full content will be converted from doc/man/mkfs.hfs.8 and mkfs.hfplus.8

Chapter 5

fsck Utilities

5.1 fsck.hfs - HFS Filesystem Check

This chapter will include complete fsck.hfs and fsck.hfs+ documentation from manpages.

5.1.1 Command Synopsis

```
fsck.hfs [-dfnpvy] device
fsck.hfs+ [-dfnpvy] device
```

Note: Full content will be converted from doc/man/fsck.hfs.8 and fsck.hfs+.8

Chapter 6

mount Utilities

6.1 mount.hfs - HFS Filesystem Mounting

This chapter will include complete mount.hfs and mount.hfs+ documentation from manpages.

6.1.1 Command Synopsis

```
mount.hfs [-o options] [-r] [-w] [-v] device mountpoint
mount.hfs+ [-o options] [-r] [-w] [-v] device mountpoint
```

Note: Full content will be converted from doc/man/mount.hfs.8 and mount.hfs+.8

Chapter 7

hfsutil Commands

7.1 hfsutil Commands

This chapter will include hfsutil command documentation.

7.1.1 Commands

- hformat - Format HFS volumes
- hmount - Mount HFS volumes
- humount - Unmount HFS volumes
- hls - List directory contents
- hcopy - Copy files
- hmkdir - Create directories
- And more...

Note: Full content will be converted from doc/man/hfsutils.1 and related manpages

Chapter 8

Implementation Details

8.1 Implementation Details

This chapter will include implementation details from HFS_IMPLEMENTATION_NOTES.md

8.1.1 Topics to Cover

- Code structure
- B-tree algorithms
- Validation strategies
- Repair mechanisms

Chapter 9

Testing and Validation

9.1 Testing and Validation

This chapter will include test suite documentation.

9.1.1 Zero-Tolerance Policy

All filesystems must be 100% correct. Any deviation from specification results in test failure.

9.1.2 Test Suite

- test/test_mkfs.sh - Filesystem creation tests
- test/test_fsck.sh - Validation tests
- test/test_hfsutils.sh - Command tests

Appendix A

Appendix

A.1 Structure Definitions

This appendix will include complete structure definitions for HFS and HFS+.

A.2 Error Codes

This section will list all error codes used by the utilities.

A.3 Glossary

Allocation Block Unit of disk space allocation

B-tree Balanced tree data structure

Catalog Directory of all files and folders

CNID Catalog Node ID

Extent Contiguous range of allocation blocks

Fork Part of a file (data or resource)

MDB Master Directory Block (HFS)

Volume Header Main metadata structure (HFS+)

A.4 References

- Inside Macintosh: Files
- Apple Technical Note TN1150
- Linux HFS/HFS+ kernel documentation
- BSD filesystem documentation