NAME

mtree — map a directory hierarchy

SYNOPSIS

mtree [**−LPUcdeinqruxw**] [**−f** *spec*] [**−f** *spec*] [**−K** *keywords*] [**−k** *keywords*]
[**−p** *path*] [**−s** *seed*] [**−X** *exclude-list*]

DESCRIPTION

The **mtree** utility compares the file hierarchy rooted in the current directory against a specification read from the standard input. Messages are written to the standard output for any files whose characteristics do not match the specifications, or which are missing from either the file hierarchy or the specification.

The options are as follows:

−c Print a specification for the file hierarchy to the standard output.

−d Ignore everything except directory type files.

−e Do not complain about files that are in the file hierarchy, but not in the specification.

−f *file*
Read the specification from *file*, instead of from the standard input.

If this option is specified twice, the two specifications are compared with each other, rather than to the file hierarchy. The specifications be sorted like output generated using **−c**. The output format in this case is somewhat remniscent of comm(1), having "in first spec only", "in second spec only", and "different" columns, prefixed by zero, one and two TAB characters respectively. Each entry in the "different" column occupies two lines, one from each specification.

−i Indent the output 4 spaces each time a directory level is descended when create a specification with the **−c** option. This does not affect either the /set statements or the comment before each directory. It does however affect the comment before the close of each directory.

−K *keywords*
Add the specified (whitespace or comma separated) *keywords* to the current set of keywords.

−k *keywords*
Use the “type” keyword plus the specified (whitespace or comma separated) *keywords* instead of the current set of keywords.

−L Follow all symbolic links in the file hierarchy.

−n Do not emit pathname comments when creating a specification. Normally a comment is emitted before each directory and before the close of that directory when using the **−c** option.

−P Do not follow symbolic links in the file hierarchy, instead consider the symbolic link itself in any comparisons. This is the default.

−p *path*
Use the file hierarchy rooted in *path*, instead of the current directory.

−q Quiet mode. Do not complain when a “missing” directory cannot be created because it already exists. This occurs when the directory is a symbolic link.

−r Remove any files in the file hierarchy that are not described in the specification.

−S Skip calculating the digest of the extended attributes of the file.

−s *seed*
Display a single checksum to the standard error output that represents all of the files for which the keyword **cksum** was specified. The checksum is seeded with the specified value.

−U Modify the owner, group, permissions, and modification time of existing files to match the specification and create any missing directories or symbolic links. User, group and permissions must all be specified for missing directories to be created. Corrected mismatches are not considered errors.

−u Same as **−U** except a status of 2 is returned if the file hierarchy did not match the specification.

−w Make some error conditions non-fatal warnings.

−X *exclude-list*
The specified file contains `fnmatch`(3) patterns matching files to be excluded from the specification, one to a line. If the pattern contains a '`/`' character, it will be matched against entire pathnames (relative to the starting directory); otherwise, it will be matched against basenames only. No comments are allowed in the *exclude-list* file.

−x Do not descend below mount points in the file hierarchy.

Specifications are mostly composed of "keywords", i.e., strings that specify values relating to files. No keywords have default values, and if a keyword has no value set, no checks based on it are performed.

Currently supported keywords are as follows:

cksum	The checksum of the file using the default algorithm specified by the `cksum`(1) utility.
flags	The file flags as a symbolic name. See `chflags`(1) for information on these names. If no flags are to be set the string "none" may be used to override the current default.
ignore	Ignore any file hierarchy below this file.
gid	The file group as a numeric value.
gname	The file group as a symbolic name.
md5digest	The MD5 message digest of the file.
sha1digest	The FIPS 160-1 ("SHA-1") message digest of the file.
ripemd160digest	The RIPEMD160 message digest of the file.
mode	The current file's permissions as a numeric (octal) or symbolic value.
nlink	The number of hard links the file is expected to have.
nochange	Make sure this file or directory exists but otherwise ignore all attributes.
uid	The file owner as a numeric value.
uname	The file owner as a symbolic name.
size	The size, in bytes, of the file.
link	The file the symbolic link is expected to reference.
time	The last modification time of the file.
btime	The creation (birth) time of the file.
atime	The last access time of the file.
ctime	The last metadata modification time of the file.
ptime	The time the file was added to its parent folder.
inode	The inode number of the file.

xattrsdigest
Digest of the extended attributes of the file.

acldigest
Digest of the access control list of the file.

type The type of the file; may be set to any one of the following:

block	block special device
char	character special device
dir	directory
fifo	fifo
file	regular file
link	symbolic link
socket	socket

The default set of keywords are **flags**, **gid**, **mode**, **nlink**, **size**, **link**, **time**, and **uid**.

There are four types of lines in a specification.

The first type of line sets a global value for a keyword, and consists of the string "/set" followed by whitespace, followed by sets of keyword/value pairs, separated by whitespace. Keyword/value pairs consist of a keyword, followed by an equals sign ("="), followed by a value, without whitespace characters. Once a keyword has been set, its value remains unchanged until either reset or unset.

The second type of line unsets keywords and consists of the string "/unset", followed by whitespace, followed by one or more keywords, separated by whitespace.

The third type of line is a file specification and consists of a file name, followed by whitespace, followed by zero or more whitespace separated keyword/value pairs. The file name may be preceded by whitespace characters. The file name may contain any of the standard file name matching characters ("[", "]", "?" or "*"), in which case files in the hierarchy will be associated with the first pattern that they match.

Each of the keyword/value pairs consist of a keyword, followed by an equals sign ("="), followed by the keyword's value, without whitespace characters. These values override, without changing, the global value of the corresponding keyword.

All paths are relative. Specifying a directory will cause subsequent files to be searched for in that directory hierarchy. Which brings us to the last type of line in a specification: a line containing only the string "`..`" causes the current directory path to ascend one level.

Empty lines and lines whose first non-whitespace character is a hash mark ("#") are ignored.

The **mtree** utility exits with a status of 0 on success, 1 if any error occurred, and 2 if the file hierarchy did not match the specification. A status of 2 is converted to a status of 0 if the **−U** option is used.

FILES

`/etc/mtree` system specification directory

EXIT STATUS

The **mtree** utility exits 0 on success, and >0 if an error occurs.

EXAMPLES

The **−d** and **−u** options can be used in combination to create directory hierarchies for distributions and other such things; the files in `/etc/mtree` were used to create almost all directories in this FreeBSD distribution.

SEE ALSO

`chflags`(1), `chgrp`(1), `chmod`(1), `cksum`(1), `md5`(1), `stat`(2), `fts`(3), `md5`(3), `chown`(8)

HISTORY

The **`mtree`** utility appeared in 4.3BSD–Reno. The MD5 digest capability was added in FreeBSD 2.1, in response to the widespread use of programs which can spoof `cksum`(1). The SHA-1 and RIPEMD160 digests were added in FreeBSD 4.0, as new attacks have demonstrated weaknesses in MD5. Support for file flags was added in FreeBSD 4.0, and mostly comes from NetBSD.