NAME

locate - find filenames quickly

SYNOPSIS

locate [**-0Scims**] [**-l** *limit*] [**-d** *database*] *pattern* ...

DESCRIPTION

The **locate** program searches a database for all pathnames which match the specified *pattern*. The database is recomputed periodically (usually weekly or daily), and contains the pathnames of all files which are publicly accessible.

Shell globbing and quoting characters ("*", "?", "\", "[" and "]") may be used in *pattern*, although they will have to be escaped from the shell. Preceding any character with a backslash ("\") eliminates any special meaning which it may have. The matching differs in that no characters must be matched explicitly, including slashes ("/").

As a special case, a pattern containing no globbing characters ("foo") is matched as though it were "*foo*".

Historically, locate only stored characters between 32 and 127. The current implementation stores any character except newline (‘\n’) and NUL (‘\0’). The 8-bit character support does not waste extra space for plain ASCII file names. Characters less than 32 or greater than 127 are stored in 2 bytes.

The following options are available:

-0 Print pathnames separated by an ASCII NUL character (character code 0) instead of default NL (newline, character code 10).

-S Print some statistics about the database and exit.

-c Suppress normal output; instead print a count of matching file names.

-d *database* Search in *database* instead of the default file name database. Multiple **-d** options are allowed. Each additional **-d** option adds the specified database to the list of databases to be searched.

The option *database* may be a colon-separated list of databases. A single colon is a reference to the default database.

$ locate -d $HOME/lib/mydb: foo

will first search string "foo" in *$HOME/lib/mydb* and then in */var/db/locate.database*.

$ locate -d $HOME/lib/mydb::/cdrom/locate.database foo

will first search string "foo" in *$HOME/lib/mydb* and then in */var/db/locate.database* and then in */cdrom/locate.database*.

$ locate -d db1 -d db2 -d db3 pattern

is the same as

$ locate -d db1:db2:db3 pattern

or

$ locate -d db1:db2 -d db3 pattern

If **-** is given as the database name, standard input will be read instead. For example, you can compress your database and use:

$ zcat database.gz | locate -d - pattern

This might be useful on machines with a fast CPU and little RAM and slow I/O. Note: you can only use *one* pattern for stdin.

-i Ignore case distinctions in both the pattern and the database.

-l *number* Limit output to *number* of file names and exit.

-m Use mmap(2) instead of the stdio(3) library. This is the default behavior and is faster in most cases.

-s Use the stdio(3) library instead of mmap(2).

ENVIRONMENT

LOCATE_PATH path to the locate database if set and not empty, ignored if the **-d** option was specified.

FILES

/var/db/locate.database	locate database
/usr/libexec/locate.updatedb	Script to update the locate database

/etc/periodic/weekly/310.locate Script that starts the database rebuild

SEE ALSO

find(1), whereis(1), which(1), fnmatch(3), locate.updatedb(8)

Woods, James A., "Finding Files Fast", *;login*, 8:1, pp. 8-10, 1983.

HISTORY

The **locate** command first appeared in 4.4BSD. Many new features were added in FreeBSD 2.2.

BUGS

The **locate** program may fail to list some files that are present, or may list files that have been removed from the system. This is because locate only reports files that are present in the database, which is typically only regenerated once a week by the */etc/periodic/weekly/310.locate* script. Use find(1) to locate files that are of a more transitory nature.

The **locate** database is typically built by user "nobody" and the locate.updatedb(8) utility skips directories which are not readable for user "nobody", group "nobody", or world. For example, if your HOME directory is not world-readable, *none* of your files are in the database.

The **locate** database is not byte order independent. It is not possible to share the databases between machines with different byte order. The current **locate** implementation understands databases in host byte order or network byte order if both architectures use the same integer size. So on a FreeBSD/i386 machine (little endian), you can read a locate database which was built on SunOS/sparc machine (big endian, net).

The **locate** utility does not recognize multibyte characters.