

[Skip site navigation](#) (1)[Skip section navigation](#) (2)

Header And Logo



Peripheral Links

[Donate to FreeBSD](#)

Search

Site Navigation

- [Home](#)
- [About](#)
 - [Introduction](#)
 - [Features](#)
 - [Advocacy](#)
 - [Marketing](#)
- [Get FreeBSD](#)
 - [Release Information](#)
 - [Release Engineering](#)
- [Documentation](#)
 - [FAQ](#)
 - [Handbook](#)
 - [Porter's Handbook](#)
 - [Developer's Handbook](#)
 - [Manual Pages](#)
 - [Documentation Project Primer](#)
 - [All Books and Articles](#)
- [Community](#)
 - [Mailing Lists](#)
 - [Forums](#)
 - [User Groups](#)
 - [Events](#)
- [Developers](#)
 - [Project Ideas](#)
 - [SVN Repository](#)

- [Support](#)
 - [Vendors](#)
 - [Security Information](#)
 - [Bug Reports](#)
 - [Submit Bug-report](#)
- [Foundation](#)
 - [Donate](#)

FreeBSD Man Pages

Man Page or Keyword Search:

[Man](#) 1 - General Commands FreeBSD 4.3-RELEASE default Architecture

[Apropos](#) Keyword Search (all sections)

[home](#) | [help](#)

FILE(1) FreeBSD General Commands Manual FILE(1)

NAME

file - determine file type

SYNOPSIS

file [-bciknsvzL] [-f *namefile*] [-m *magicfiles*] *file* ...

DESCRIPTION

This manual page documents version 3.33 of the **file** command. **File** tests each argument in an attempt to classify it. There are three sets of tests, performed in this order: filesystem tests, magic number tests, and language tests. The *first* test that succeeds causes the file type to be printed.

The type printed will usually contain one of the words *text* (the file contains only printing characters and a few common control characters and is probably safe to read on an ASCII terminal), *executable* (the file contains the result of compiling a program in a form understandable to some UNIX kernel or another), or *data* meaning anything else (data is usually 'binary' or non-printable). Exceptions are well-known file formats (core files, tar archives) that are known to contain binary data. When modifying the file */usr/share/misc/magic* or the program itself, *preserve these keywords*. People depend on knowing that all the readable files in a directory have the word 'text' printed. Don't do as Berkeley did and change 'shell commands text' to 'shell script'. Note that the file */usr/share/misc/magic* is built mechanically from a large number of small files in the subdirectory *Magdir* in the source distribution of this program.

The filesystem tests are based on examining the return from a [stat\(2\)](#) system call. The program checks to see if the file is empty, or if it's some sort of special file. Any known file types appropriate to the system you are running on (sockets, symbolic links, or named pipes (FIFOs) on those systems that implement them) are intuited if they are defined in the system header file *<sys/stat.h>*.

The magic number tests are used to check for files with data in particular fixed formats. The canonical example of this is a binary executable (compiled program) *a.out* file, whose format is defined in *a.out.h* and possibly *exec.h* in the standard include directory. These files have a

cutable, and which of several types thereof. The concept of 'magic number' has been applied by extension to data files. Any file with some invariant identifier at a small fixed offset into the file can usually be described in this way. The information identifying these files is read from the magic file `/usr/share/misc/magic`.

If a file does not match any of the entries in the magic file, it is examined to see if it seems to be a text file. ASCII, ISO-8859-x, non-ISO 8-bit extended-ASCII character sets (such as those used on Macintosh and IBM PC systems), UTF-8-encoded Unicode, UTF-16-encoded Unicode, and EBCDIC character sets can be distinguished by the different ranges and sequences of bytes that constitute printable text in each set. If a file passes any of these tests, its character set is reported. ASCII, ISO-8859-x, UTF-8, and extended-ASCII files are identified as 'text' because they will be mostly readable on nearly any terminal; UTF-16 and EBCDIC are only 'character data' because, while they contain text, it is text that will require translation before it can be read. In addition, file will attempt to determine other characteristics of text-type files. If the lines of a file are terminated by CR, CRLF, or NEL, instead of the Unix-standard LF, this will be reported. Files that contain embedded escape sequences or overstriking will also be identified.

Once file has determined the character set used in a text-type file, it will attempt to determine in what language the file is written. The language tests look for particular strings (cf `names.h`) that can appear anywhere in the first few blocks of a file. For example, the keyword `.br` indicates that the file is most likely a `troff(1)` input file, just as the keyword `struct` indicates a C program. These tests are less reliable than the previous two groups, so they are performed last. The language test routines also test for some miscellany (such as `tar(1)` archives).

Any file that cannot be identified as having been written in any of the character sets listed above is simply said to be 'data'.

OPTIONS

- `-b` Do not prepend filenames to output lines (brief mode).
- `-c` Cause a checking printout of the parsed form of the magic file. This is usually used in conjunction with `-m` to debug a new magic file before installing it.
- `-f namefile`
Read the names of the files to be examined from `namefile` (one per line) before the argument list. Either `namefile` or at least one filename argument must be present; to test the standard input, use `-'` as a filename argument.
- `-i` Causes the file command to output mime type strings rather than the more traditional human readable ones. Thus it may say `'text/plain; charset=us-ascii'` rather than `'ASCII text'`. In order for this option to work, file changes the way it handles files recognised by the command itself (such as many of the text file types, directories etc), and makes use of an alternative `'magic'` file. (See `FILES` section, below).
- `-k` Don't stop at the first match, keep going.
- `-m list`
Specify an alternate `list` of files containing magic numbers. This can be a single file, or a colon-separated list of files.
- `-n` Force stdout to be flushed after checking each file. This is only useful if checking a list of files. It is intended to be

- s Normally, file only attempts to read and determine the type of argument files which [stat\(2\)](#) reports are ordinary files. This prevents problems, because reading special files may have peculiar consequences. Specifying the -s option causes file to also read argument files which are block or character special files. This is useful for determining the filesystem types of the data in raw disk partitions, which are block special files. This option also causes file to disregard the file size as reported by [stat\(2\)](#) since on some systems it reports a zero size for raw disk partitions.
- v Print the version of the program and exit.
- z Try to look inside compressed files.
- L Cause symlinks to be followed, as the like-named option in [ls\(1\)](#). (on systems that support symbolic links).

FILES

/usr/share/misc/magic default list of magic numbers
 /usr/share/misc/magic.mime default list of magic numbers, used to output mime types when the -i option is specified.

ENVIRONMENT

The environment variable MAGIC can be used to set the default magic number files.

SEE ALSO

[od\(1\)](#), [strings\(1\)](#), [magic\(5\)](#)

STANDARDS CONFORMANCE

This program is believed to exceed the System V Interface Definition of FILE(CMD), as near as one can determine from the vague language contained therein. Its behaviour is mostly compatible with the System V program of the same name. This version knows more magic, however, so it will produce different (albeit more accurate) output in many cases.

The one significant difference between this version and System V is that this version treats any white space as a delimiter, so that spaces in pattern strings must be escaped. For example,

```
>10 string language impress (imPRESS data)
in an existing magic file would have to be changed to
>10 string language\ impress (imPRESS data)
```

In addition, in this version, if a pattern string contains a backslash, it must be escaped. For example

```
0 string \begindata Andrew Toolkit document
in an existing magic file would have to be changed to
0 string \\begindata Andrew Toolkit document
```

SunOS releases 3.2 and later from Sun Microsystems include a [file\(1\)](#) command derived from the System V one, but with some extensions. My version differs from Sun's only in minor ways. It includes the extension of the '&' operator, used as, for example,

```
>16 long&0x7fffffff >0 not stripped
```

MAGIC DIRECTORY

The magic file entries have been collected from various sources, mainly USENET, and contributed by various authors. Christos Zoulas (address below) will collect additional or corrected magic file entries. A consolidation of magic file entries will be distributed periodically.

The order of entries in the magic file is significant. Depending on what

file around for comparison purposes (rename it to [/usr/share/misc/magic.orig](#)).

EXAMPLES

```
$ file file.c file /dev/hda
file.c:  C program text
file:    ELF 32-bit LSB executable, Intel 80386, version 1,
        dynamically linked, not stripped
/dev/hda: block special

$ file -s /dev/hda{,1,2,3,4,5,6,7,8,9,10}
/dev/hda:  x86 boot sector
/dev/hda1: Linux/i386 ext2 filesystem
/dev/hda2: x86 boot sector
/dev/hda3: x86 boot sector, extended partition table
/dev/hda4: Linux/i386 ext2 filesystem
/dev/hda5: Linux/i386 swap file
/dev/hda6: Linux/i386 swap file
/dev/hda7: Linux/i386 swap file
/dev/hda8: Linux/i386 swap file
/dev/hda9: empty
/dev/hda10: empty

$ file -i file.c file /dev/hda
file.c:    text/x-c
file:     application/x-executable, dynamically linked (uses shared libs), not stripped
/dev/hda: application/x-not-regular-file
```

HISTORY

There has been a **file** command in every UNIX since at least Research Version 6 (man page dated January, 1975). The System V version introduced one significant major change: the external list of magic number types. This slowed the program down slightly but made it a lot more flexible.

This program, based on the System V version, was written by Ian Darwin <ian@darwinsys.com> without looking at anybody else's source code.

John Gilmore revised the code extensively, making it better than the first version. Geoff Collyer found several inadequacies and provided some magic file entries. Contributions by the ``&'` operator by Rob McMahon <cudcv@warwick.ac.uk>, 1989.

Guy Harris <guy@netapp.com>, made many changes from 1993 to the present.

Primary development and maintenance from 1990 to the present by Christos Zoulas <christos@astron.com>.

Altered by Chris Lowth <chris@lowth.com>, 2000: Handle the `-i` option to output mime type strings and using an alternative magic file and internal logic.

Altered by Eric Fischer <enf@pobox.com>, July, 2000, to identify character codes and attempt to identify the languages of non-ASCII files.

The list of contributors to the *Magdir* directory (source for the [/usr/share/misc/magic](#) file) is too long to include here. You know who you are; thank you.

LEGAL NOTICE

Copyright (c) Ian F. Darwin, Toronto, Canada, 1986-1999. Covered by the standard Berkeley Software Distribution copyright; see the file LEGAL.NOTICE in the source distribution.

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts



Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research



With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips



Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.