NAME

fnmatch – match filename or pathname

SYNOPSIS

#include <fnmatch.h>

int fnmatch(const char ****pattern*, const char ****string***, int** *flags***);**

DESCRIPTION

The **fnmatch**() function checks whether the *string* argument matches the *pattern* argument, which is a shell wildcard pattern.

The *flags* argument modifies the behavior; it is the bitwise OR of zero or more of the following flags:

FNM_NOESCAPE
If this flag is set, treat backslash as an ordinary character, instead of an escape character.

FNM_PATHNAME
If this flag is set, match a slash in *string* only with a slash in *pattern* and not by an asterisk (*) or a question mark (?) metacharacter, nor by a bracket expression ([]) containing a slash.

FNM_PERIOD
If this flag is set, a leading period in *string* has to be matched exactly by a period in *pattern*. A period is considered to be leading if it is the first character in *string*, or if both **FNM_PATHNAME** is set and the period immediately follows a slash.

FNM_FILE_NAME
This is a GNU synonym for **FNM_PATHNAME**.

FNM_LEADING_DIR
If this flag (a GNU extension) is set, the pattern is considered to be matched if it matches an initial segment of *string* which is followed by a slash. This flag is mainly for the internal use of glibc and is only implemented in certain cases.

FNM_CASEFOLD
If this flag (a GNU extension) is set, the pattern is matched case-insensitively.

RETURN VALUE

Zero if *string* matches *pattern*, **FNM_NOMATCH** if there is no match or another nonzero value if there is an error.

CONFORMING TO

POSIX.2. The **FNM_FILE_NAME**, **FNM_LEADING_DIR**, and **FNM_CASEFOLD** flags are GNU extensions.

NAME

fgetc, fgets, getc, getchar, fputc, fputs, putc, putchar – input and output of characters and strings

SYNOPSIS

#include <stdio.h>

int fgetc(FILE ****stream*);**
char *fgets(char ****s*, int** *size***, FILE ****stream***);**
int getc(FILE ****stream*);**
int getchar(void);
int fputc(int *c***, FILE ****stream***);**
int fputs(const char ****s*, FILE ****stream***);**
int putc(int *c***, FILE ****stream***);**
int putchar(int *c***);**

DESCRIPTION

fgetc() reads the next character from *stream* and returns it as an *unsigned char* cast to an *int*, or **EOF** on end of file or error.

getc() is equivalent to **fgetc**() except that it may be implemented as a macro which evaluates *stream* more than once.

getchar() is equivalent to **getc**(*stdin*).

fgets() reads in at most one less than *size* characters from *stream* and stores them into the buffer pointed to by *s*. Reading stops after an **EOF** or a newline. If a newline is read, it is stored into the buffer. A **'\0'** is stored after the last character in the buffer.

fputc() writes the character *c*, cast to an *unsigned char*, to *stream*.

fputs() writes the string *s* to *stream*, without its terminating null byte ('\0').

putc() is equivalent to **fputc**() except that it may be implemented as a macro which evaluates *stream* more than once.

putchar(*c*); is equivalent to **putc**(*c*, *stdout*).

Calls to the functions described here can be mixed with each other and with calls to other output functions from the *stdio* library for the same output stream.

RETURN VALUE

fgetc(), **getc**() and **getchar**() return the character read as an *unsigned char* cast to an *int* or **EOF** on end of file or error.

fgets() returns *s* on success, and NULL on error or when end of file occurs while no characters have been read. **fputc**(), **putc**() and **putchar**() return the character written as an *unsigned char* cast to an *int* or **EOF** on error.

fputs() returns a nonnegative number on success, or **EOF** on error.

SEE ALSO

read(2), **write**(2), **ferror**(3), **fgetwc**(3), **fgetws**(3), **fopen**(3), **fread**(3), **fseek**(3), **getline**(3), **getwchar**(3), **scanf**(3), **ungetwc**(3), **write**(2), **ferror**(3), **fopen**(3), **fputwc**(3), **fputws**(3), **fseek**(3), **fwrite**(3), **gets**(3), **putwchar**(3), **scanf**(3), **unlocked_stdio**(3)

NAME

isalnum, isalpha, isascii, isblank, iscntrl, isdigit, isgraph, islower, isprint, ispunct, isspace, isupper, isxdigit, isalnum_l, isalpha_l, isascii_l, isblank_l, iscntrl_l, isdigit_l, isgraph_l, islower_l, isprint_l, ispunct_l, isspace_l, isupper_l, isxdigit_l – character classification functions

SYNOPSIS

```
#include <ctype.h>

int isalnum(int c);
int isalpha(int c);
int iscntrl(int c);
int isdigit(int c);
int isgraph(int c);
int islower(int c);
int isprint(int c);
int ispunct(int c);
int isspace(int c);
int isupper(int c);
int isxdigit(int c);

int isascii(int c);
int isblank(int c);
```

DESCRIPTION

These functions check whether *c*, which must have the value of an *unsigned char* or **EOF**, falls into a certain character class according to the specified locale. The functions without the "_l" suffix perform the check based on the current locale.

The functions with the "_l" suffix perform the check based on the locale specified by the locale object *locale*. The behavior of these functions is undefined if *locale* is the special locale object **LC_GLOBAL_LOCALE** (see **duplocale**(3)) or is not a valid locale object handle.

The list below explains the operation of the functions without the "_l" suffix; the functions with the "_l" suffix differ only in using the locale object *locale* instead of the current locale.

isalnum()
: checks for an alphanumeric character; it is equivalent to **(isalpha**(*c*) **|| isdigit**(*c*)).

isalpha()
: checks for an alphabetic character; in the standard **"C"** locale, it is equivalent to **(isupper**(*c*) **|| islower**(*c*)). In some locales, there may be additional characters for which **isalpha**() is true—letters which are neither uppercase nor lowercase.

isascii()
: checks whether *c* is a 7-bit *unsigned char* value that fits into the ASCII character set.

isblank()
: checks for a blank character; that is, a space or a tab.

iscntrl()
: checks for a control character.

isdigit()
: checks for a digit (0 through 9).

isgraph()
: checks for any printable character except space.

islower()
: checks for a lowercase character.

isprint()
: checks for any printable character including space.

ispunct()
: checks for any printable character which is not a space or an alphanumeric character.

isspace()
: checks for white-space characters. In the **"C"** and **"POSIX"** locales, these are: space, form-feed (**'\f'**), newline (**'\n'**), carriage return (**'\r'**), horizontal tab (**'\t'**), and vertical tab (**'\v'**).

isupper()
: checks for an uppercase letter.

isxdigit()
: checks for hexadecimal digits, that is, one of
0 1 2 3 4 5 6 7 8 9 a b c d e f A B C D E F.

RETURN VALUE

The values returned are nonzero if the character *c* falls into the tested class, and zero if not.

ATTRIBUTES

For an explanation of the terms used in this section, see **attributes**(7).

Interface	Attribute	Value
isalnum(), **isalpha**(), **isascii**(), **isblank**(), **iscntrl**(), **isdigit**(), **isgraph**(), **islower**(), **isprint**(), **ispunct**(), **isspace**(), **isupper**(), **isxdigit**()	Thread safety	MT-Safe

NOTES

The standards require that the argument *c* for these functions is either **EOF** or a value that is representable in the type *unsigned char*. If the argument *c* is of type *char*, it must be cast to *unsigned char*, as in the following example:

```
char c; ... res = toupper((unsigned char) c);
```

This is necessary because *char* may be the equivalent of *signed char*, in which case a byte where the top bit is set would be sign extended when converting to *int*, yielding a value that is outside the range of *unsigned char*.

The details of what characters belong to which class depend on the locale. For example, **isupper**() will not recognize an A-umlaut (Ä) as an uppercase letter in the default **C** locale.

SEE ALSO

iswalnum(3), **iswalpha**(3), **iswblank**(3), **iswcntrl**(3), **iswdigit**(3), **iswgraph**(3), **iswlower**(3), **iswprint**(3), **iswpunct**(3), **iswspace**(3), **iswupper**(3), **iswxdigit**(3), **newlocale**(3), **setlocale**(3), **toascii**(3), **tolower**(3), **toupper**(3), **uselocale**(3), **ascii**(7), **locale**(7)

COLOPHON

This page is part of release 5.10 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man−pages/.