NAME

glob, **globfree** - generate pathnames matching a pattern

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <glob.h>

int
glob(*const char * restrict pattern*, *int flags*, *int (*errfunc)(const char *, int)*, *glob_t * restrict pglob*);

void
globfree(*glob_t *pglob*);

DESCRIPTION

The **glob**() function is a pathname generator that implements the rules for file name pattern matching used by the shell.

The include file <*glob.h*> defines the structure type *glob_t*, which contains at least the following fields:

```
typedef struct {
        size_t gl_pathc;      /* count of total paths so far */
        size_t gl_matchc;     /* count of paths matching pattern */
        size_t gl_offs;                   /* reserved at beginning of gl_pathv */
        int gl_flags;                     /* returned flags */
        char **gl_pathv;      /* list of paths matching pattern */
} glob_t;
```

The argument *pattern* is a pointer to a pathname pattern to be expanded. The **glob**() argument matches all accessible pathnames against the pattern and creates a list of the pathnames that match. In order to have access to a pathname, **glob**() requires search permission on every component of a path except the last and read permission on each directory of any filename component of *pattern* that contains any of the special characters ‘*’, ‘?’ or ‘[’.

The **glob**() argument stores the number of matched pathnames into the *gl_pathc* field, and a pointer to a list of pointers to pathnames into the *gl_pathv* field. The first pointer after the last pathname is NULL. If the pattern does not match any pathnames, the returned number of matched paths is set to zero.

It is the caller’s responsibility to create the structure pointed to by *pglob*. The **glob**() function allocates other space as needed, including the memory pointed to by *gl_pathv*.

The argument *flags* is used to modify the behavior of **glob**(). The value of *flags* is the bitwise inclusive OR of any of the following values defined in <*glob.h*>:

GLOB_APPEND	Append pathnames generated to the ones from a previous call (or calls) to **glob**(). The value of *gl_pathc* will be the total matches found by this call and the previous call(s). The pathnames are appended to, not merged with the pathnames returned by the previous call(s). Between calls, the caller must not change the setting of the GLOB_DOOFFS flag, nor change the value of *gl_offs* when GLOB_DOOFFS is set, nor (obviously) call **globfree**() for *pglob*.
GLOB_DOOFFS	Make use of the *gl_offs* field. If this flag is set, *gl_offs* is used to specify how many NULL pointers to prepend to the beginning of the *gl_pathv* field. In other words, *gl_pathv* will point to *gl_offs* NULL pointers, followed by *gl_pathc* pathname pointers, followed by a NULL pointer.
GLOB_ERR	Causes **glob**() to return when it encounters a directory that it cannot open or read. Ordinarily, **glob**() continues to find matches.
GLOB_MARK	Each pathname that is a directory that matches *pattern* has a slash appended.
GLOB_NOCHECK	If *pattern* does not match any pathname, then **glob**() returns a list consisting of only *pattern*, with the number of total pathnames set to 1, and the number of matched pathnames set to 0. The effect of backslash escaping is present in the pattern returned.
GLOB_NOESCAPE	By default, a backslash (‘\’) character is used to escape the following character in the pattern, avoiding any special interpretation of the character. If GLOB_NOESCAPE is set, backslash escaping is disabled.
GLOB_NOSORT	By default, the pathnames are sorted in ascending collation order; this flag prevents that sorting (speeding up **glob**()).

The following values may also be included in *flags*, however, they are non-standard extensions to IEEE Std 1003.2 ("POSIX.2").

GLOB_ALTDIRFUNC	The following additional fields in the pglob structure have been initialized with alternate functions for glob to use to open, read, and close directories and to get stat information on names found in those directories.

```
void *(*gl_opendir)(const char * name);
struct dirent *(*gl_readdir)(void *);
void (*gl_closedir)(void *);
int (*gl_lstat)(const char *name, struct stat *st);
int (*gl_stat)(const char *name, struct stat *st);
```

This extension is provided to allow programs such as restore(8) to provide globbing from directories stored on tape.

GLOB_BRACE
: Pre-process the pattern string to expand ‘{pat,pat,...}’ strings like csh(1). The pattern ‘{}’ is left unexpanded for historical reasons (and csh(1) does the same thing to ease typing of find(1) patterns).

GLOB_MAGCHAR
: Set by the **glob**() function if the pattern included globbing characters. See the description of the usage of the *gl_matchc* structure member for more details.

GLOB_NOMAGIC
: Is the same as GLOB_NOCHECK but it only appends the *pattern* if it does not contain any of the special characters ‘‘*’’, ‘‘?’’ or ‘‘[’’. GLOB_NOMAGIC is provided to simplify implementing the historic csh(1) globbing behavior and should probably not be used anywhere else.

GLOB_TILDE
: Expand patterns that start with ‘~’ to user name home directories.

GLOB_LIMIT
: Limit the total number of returned pathnames to the value provided in *gl_matchc* (default ARG_MAX). This option should be set for programs that can be coerced into a denial of service attack via patterns that expand to a very large number of matches, such as a long string of ‘*/../*/..’.

If, during the search, a directory is encountered that cannot be opened or read and *errfunc* is non-NULL, **glob**() calls *(*errfunc)*(*path*, *errno*), however, the GLOB_ERR flag will cause an immediate return when this happens.

If *errfunc* returns non-zero, **glob**() stops the scan and returns GLOB_ABORTED after setting *gl_pathc* and *gl_pathv* to reflect any paths already matched. This also happens if an error is encountered and GLOB_ERR is set in *flags*, regardless of the return value of *errfunc*, if called. If GLOB_ERR is not set and either *errfunc* is NULL or *errfunc* returns zero, the error is ignored.

The **globfree**() function frees any space associated with *pglob* from a previous call(s) to **glob**().

RETURN VALUES

On successful completion, **glob**() returns zero. In addition the fields of *pglob* contain the values described below:

gl_pathc	contains the total number of matched pathnames so far. This includes other matches from previous invocations of **glob**() if GLOB_APPEND was specified.
gl_matchc	contains the number of matched pathnames in the current invocation of **glob**().
gl_flags	contains a copy of the *flags* argument with the bit GLOB_MAGCHAR set if *pattern* contained any of the special characters ‘‘*’’, ‘‘?’’ or ‘‘[’’, cleared if not.
gl_pathv	contains a pointer to a NULL-terminated list of matched pathnames. However, if *gl_pathc* is zero, the contents of *gl_pathv* are undefined.

If **glob**() terminates due to an error, it sets errno and returns one of the following non-zero constants, which are defined in the include file <*glob.h*>:

GLOB_NOSPACE	An attempt to allocate memory failed, or if *errno* was E2BIG, GLOB_LIMIT was specified in the flags and *pglob->gl_matchc* or more patterns were matched.
GLOB_ABORTED	The scan was stopped because an error was encountered and either GLOB_ERR was set or *(*errfunc)()* returned non-zero.
GLOB_NOMATCH	The pattern did not match a pathname and GLOB_NOCHECK was not set.

The arguments *pglob->gl_pathc* and *pglob->gl_pathv* are still set as specified above.

EXAMPLES

A rough equivalent of ‘ls -l *.c *.h’ can be obtained with the following code:

```
glob_t g;

g.gl_offs = 2;
glob("*.c", GLOB_DOOFFS, NULL, &g);
glob("*.h", GLOB_DOOFFS | GLOB_APPEND, NULL, &g);
g.gl_pathv[0] = "ls";
g.gl_pathv[1] = "-l";
execvp("ls", g.gl_pathv);
```

SEE ALSO

sh(1), fnmatch(3), regex(3)

STANDARDS

The current implementation of the **glob**() function *does not* conform to IEEE Std 1003.2 ("POSIX.2"). Collating symbol expressions, equivalence class expressions and character class expressions are not supported.

The flags GLOB_ALTDIRFUNC, GLOB_BRACE, GLOB_LIMIT, GLOB_MAGCHAR, GLOB_NOMAGIC, and GLOB_TILDE, and the fields *gl_matchc* and *gl_flags* are extensions to the POSIX standard and should not be used by applications striving for strict conformance.

HISTORY

The **glob**() and **globfree**() functions first appeared in 4.4BSD.

BUGS

Patterns longer than MAXPATHLEN may cause unchecked errors.

The **glob**() argument may fail and set errno for any of the errors specified for the library routines stat(2), closedir(3), opendir(3), readdir(3), malloc(3), and free(3).