NAME

glob, globfree − find pathnames matching a pattern, free memory from glob()

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

#include <glob.h>

int glob(const char *restrict *pattern***, int** *flags***,**
int (**errfunc***)(const char ****epath***, int** *eerrno***),**
glob_t *restrict *pglob***);**
void globfree(glob_t ****pglob*);**

DESCRIPTION

The **glob**() function searches for all the pathnames matching *pattern* according to the rules used by the shell (see **glob**(7)). No tilde expansion or parameter substitution is done; if you want these, use **wordexp**(3).

The **globfree**() function frees the dynamically allocated storage from an earlier call to **glob**().

The results of a **glob**() call are stored in the structure pointed to by *pglob*. This structure is of type*glob_t* (declared in*<glob .h>*) and includes the following elements defined by POSIX.2 (more may be present as an extension):

```
typedef struct {
    size_t   gl_pathc;    /* Count of paths matched so far  */
    char   **gl_pathv;    /* List of matched pathnames.  */
    size_t   gl_offs;     /* Slots to reserve ingl_pathv.  */
} glob_t;
```

Results are stored in dynamically allocated storage.

The argument *flags* is made up of the bitwise OR of zero or more the following symbolic constants, which modify the behavior of **glob**():

GLOB_ERR
Return upon a read error (because a directory does not have read permission, for example). By default, **glob**() attempts carry on despite errors, reading all of the directories that it can.

GLOB_MARK
Append a slash to each path which corresponds to a directory.

GLOB_NOSORT
Don’t sort the returned pathnames. The only reason to do this is to save processing time. By default, the returned pathnames are sorted.

GLOB_DOOFFS
Reserve *pglob−>gl_offs* slots at the beginning of the list of strings in *pglob−>pathv*. The reserved slots contain null pointers.

GLOB_NOCHECK
If no pattern matches, return the original pattern. By default, **glob**() returns **GLOB_NOMATCH** if there are no matches.

GLOB_APPEND
Append the results of this call to the vector of results returned by a previous call to **glob**(). Do not set this flag on the first invocation of **glob**().

GLOB_NOESCAPE
Don’t allow backslash ('\') to be used as an escape character. Normally, a backslash can be used to quote the following character, providing a mechanism to turn off the special meaning metacharacters.

flags may also include any of the following, which are GNU extensions and not defined by POSIX.2:

GLOB_PERIOD
Allow a leading period to be matched by metacharacters. By default, metacharacters can't match a leading period.

GLOB_ALTDIRFUNC
Use alternative functions *pglob->gl_closedir*, *pglob->gl_readdir*, *pglob->gl_opendir*, *pglob->gl_lstat*, and *pglob->gl_stat* for filesystem access instead of the normal library functions.

GLOB_BRACE
Expand **csh**(1) style brace expressions of the form **{a,b}**. Brace expressions can be nested. Thus, for example, specifying the pattern "{foo/{,cat,dog},bar}" would return the same results as four separate **glob**() calls using the strings: "foo/", "foo/cat", "foo/dog", and "bar".

GLOB_NOMAGIC
If the pattern contains no metacharacters, then it should be returned as the sole matching word, even if there is no file with that name.

GLOB_TILDE
Carry out tilde expansion. If a tilde ('~') is the only character in the pattern, or an initial tilde is followed immediately by a slash ('/'), then the home directory of the caller is substituted for the tilde. If an initial tilde is followed by a username (e.g., "~andrea/bin"), then the tilde and username are substituted by the home directory of that user. If the username is invalid, or the home directory cannot be determined, then no substitution is performed.

GLOB_TILDE_CHECK
This provides behavior similar to that of **GLOB_TILDE**. The difference is that if the username is invalid, or the home directory cannot be determined, then instead of using the pattern itself as the name, **glob**() returns **GLOB_NOMATCH** to indicate an error.

GLOB_ONLYDIR
This is a *hint* to **glob**() that the caller is interested only in directories that match the pattern. If the implementation can easily determine file-type information, then nondirectory files are not returned to the caller. However, the caller must still check that returned files are directories. (The purpose of this flag is merely to optimize performance when the caller is interested only in directories.)

If *errfunc* is not NULL, it will be called in case of an error with the arguments *epath*, a pointer to the path which failed, and *eerrno*, the value of *errno* as returned from one of the calls to **opendir**(3), **readdir**(3), or **stat**(2). If*errfunc* returns nonzero, or if **GLOB_ERR** is set, **glob**() will terminate after the call to *errfunc*.

Upon successful return, *pglob->gl_pathc* contains the number of matched pathnames and *pglob->gl_pathv* contains a pointer to the list of pointers to matched pathnames. The list of pointers is terminated by a null pointer.

It is possible to call **glob**() several times. In that case, the **GLOB_APPEND** flag has to be set in *flags* on the second and later invocations.

As a GNU extension, *pglob->gl_flags* is set to the flags specified, **or**ed with **GLOB_MAGCHAR** if any metacharacters were found.

RETURN VALUE

On successful completion, **glob**() returns zero. Other possible returns are:

GLOB_NOSPACE
for running out of memory,

GLOB_ABORTED
for a read error, and

GLOB_NOMATCH
for no found matches.

ATTRIBUTES

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

Interface	Attribute	Value
glob()	Thread safety	MT-Unsafe race:utent env sig:ALRM timer locale
globfree()	Thread safety	MT-Safe

In the above table, *utent* in *race:utent* signifies that if any of the functions **setutent**(3), **getutent**(3), or **endutent**(3) are used in parallel in different threads of a program, then data races could occur. **glob**() calls those functions, so we use race:utent to remind users.

STANDARDS

POSIX.1-2001, POSIX.1-2008, POSIX.2.

NOTES

The structure elements *gl_pathc* and *gl_offs* are declared as *size_t* in glibc 2.1, as they should be according to POSIX.2, but are declared as *int* in glibc 2.0.

BUGS

The **glob**() function may fail due to failure of underlying function calls, such as **malloc**(3) or **opendir**(3). These will store their error code in *errno*.

EXAMPLES

One example of use is the following code, which simulates typing

```
ls -l *.c ../*.c
```

in the shell:

```
glob_t globbuf;

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

SEE ALSO

ls(1), **sh**(1), **stat**(2), **exec**(3), **fnmatch**(3), **malloc**(3), **opendir**(3), **readdir**(3), **wordexp**(3), **glob**(7)