NAME

wordexp, wordfree − perform word expansion like a posix-shell

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

#include <wordexp.h>

int wordexp(const char *restrict *s*, **wordexp_t *restrict** *p*, **int** *flags***);**
void wordfree(wordexp_t ***p*);**

Feature Test Macro Requirements for glibc (see **feature_test_macros**(7)):

wordexp(), **wordfree**():
  _XOPEN_SOURCE

DESCRIPTION

The function **wordexp**() performs a shell-like expansion of the string *s* and returns the result in the structure pointed to by *p*. The data type*wor dexp_t* is a structure that at least has the fields *we_wordc*, *we_wordv*, and *we_offs*. The field*we_wor dc* is a *size_t* that gives the number of words in the expansion of *s*. The field*we_wor dv* is a *char *** that points to the array of words found. The field *we_offs* of type *size_t* is sometimes (depending on *flags*, see below) used to indicate the number of initial elements in the *we_wordv* array that should be filled with NULLs.

The function **wordfree**() frees the allocated memory again. More precisely, it does not free its argument, but it frees the array *we_wordv* and the strings that points to.

The string argument

Since the expansion is the same as the expansion by the shell (see **sh**(1)) of the parameters to a command, the string *s* must not contain characters that would be illegal in shell command parameters. In particular, there must not be any unescaped newline or |, &, ;, <, >, (,), {, } characters outside a command substitution or parameter substitution context.

If the argument *s* contains a word that starts with an unquoted comment character #, then it is unspecified whether that word and all following words are ignored, or the # is treated as a non-comment character.

The expansion

The expansion done consists of the following stages: tilde expansion (replacing ~user by user's home directory), variable substitution (replacing $FOO by the value of the environment variable FOO), command substitution (replacing $(command) or `command` by the output of command), arithmetic expansion, field splitting, wildcard expansion, quote removal.

The result of expansion of special parameters ($@, $*, $#, $?, $−, $$, $!, $0) is unspecified.

Field splitting is done using the environment variable $IFS. If it is not set, the field separators are space, tab, and newline.

The output array

The array *we_wordv* contains the words found, followed by a NULL.

The flags argument

The *flag* argument is a bitwise inclusive OR of the following values:

WRDE_APPEND
: Append the words found to the array resulting from a previous call.

WRDE_DOOFFS
: Insert *we_offs* initial NULLs in the array *we_wordv*. (These are not counted in the returned *we_wordc*.)

WRDE_NOCMD
: Don't do command substitution.

WRDE_REUSE
The argument *p* resulted from a previous call to **wordexp**(), and **wordfree**() was not called. Reuse the allocated storage.

WRDE_SHOWERR
Normally during command substitution *stderr* is redirected to */dev/null*. This flag specifies that *stderr* is not to be redirected.

WRDE_UNDEF
Consider it an error if an undefined shell variable is expanded.

RETURN VALUE

On success, **wordexp**() returns 0. On failure, **wordexp**() returns one of the following nonzero values:

WRDE_BADCHAR
Illegal occurrence of newline or one of |, &, ;, <, >, (,), {, }.

WRDE_BADVAL
An undefined shell variable was referenced, and the **WRDE_UNDEF** flag told us to consider this an error.

WRDE_CMDSUB
Command substitution requested, but the **WRDE_NOCMD** flag told us to consider this an error.

WRDE_NOSPACE
Out of memory.

WRDE_SYNTAX
Shell syntax error, such as unbalanced parentheses or unmatched quotes.

VERSIONS

wordexp() and **wordfree**() are provided since glibc 2.1.

ATTRIBUTES

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

Interface	**Attribute**	**Value**
wordexp()	Thread safety	MT-Unsafe race:utent const:env env sig:ALRM timer locale
wordfree()	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. **wordexp**() calls those functions, so we use race:utent to remind users.

STANDARDS

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

EXAMPLES

The output of the following example program is approximately that of "ls [a-c]*.c".

```
#include <stdio.h>
#include <stdlib.h>
#include <wordexp.h>

int
main(void)
{
    wordexp_t p;
    char **w;

    wordexp("[a−c]*.c", &p, 0);
    w = p.we_wordv;
```

```
        for (size_t i = 0; i < p.we_wordc; i++)
            printf("%s\n", w[i]);
        wordfree(&p);
        exit(EXIT_SUCCESS);
    }
```

SEE ALSO

fnmatch(3), **glob**(3)