NAME

wordexp, wordfree – perform word expansions

SYNOPSIS

#include <wordexp.h>

int wordexp(const char **words*, **wordexp_t** **pwordexp*, **int** *flags***);**

void wordfree(wordexp_t **pwordexp***);**

MT-LEVEL

MT-Safe

DESCRIPTION

The **wordexp( )** function performs word expansions, subject to quoting, and places the list of expanded words into the structure pointed to by *pwordexp*.

The **wordfree( )** function frees any memory allocated by **wordexp( )** associated with *pwordexp*.

words **Argument**

The *words* argument is a pointer to a string containing one or more words to be expanded. The expansions will be the same as would be performed by the shell if *words* were the part of a command line representing the arguments to a utility. Therefore, *words* must not contain an unquoted NEWLINE or any of the unquoted shell special characters:

| & ; < >

except in the context of command substitution. It also must not contain unquoted parentheses or braces, except in the context of command or variable substitution. If the argument *words* contains an unquoted comment character (number sign) that is the beginning of a token, **wordexp( )** may treat the comment character as a regular character, or may interpret it as a comment indicator and ignore the remainder of *words*.

pwordexp **Argument**

The structure type **wordexp_t** is defined in the header **<wordexp.h>** and includes at least the following members:

size_t we_wordc	Count of words matched by *words*.
char **we_wordv	Pointer to list of expanded words.
size_t we_offs	Slots to reserve at the beginning of *pwordexp*–>**we_wordv**.

The **wordexp( )** function stores the number of generated words into *pwordexp*–>**we_wordc** and a pointer to a list of pointers to words in *pwordexp*–>**we_wordv**. Each individual field created during field splitting is a separate word in the *pwordexp*–>**we_wordv** list. The words are in order. The first pointer after the last word pointer will be a **NULL** pointer.

It is the caller's responsibility to allocate the storage pointed to by *pwordexp*. The **wordexp( )** function allocates other space as needed, including memory pointed to by *pwordexp*–>**we_wordv**. The **wordfree( )** function frees any memory associated with *pwordexp* from a previous call to **wordexp( )**.

flags **Argument**

The *flags* argument is used to control the behavior of **wordexp( )**. The value of *flags* is the bitwise inclusive OR of zero or more of the following constants, which are defined in **<wordexp.h>**:

WRDE_APPEND   Append words generated to the ones from a previous call to

wordexp( ).

WRDE_DOOFFS Make use of *pwordexp*–>**we_offs**. If this flag is set, *pwordexp*–>**we_offs** is used to specify how many **NULL** pointers to add to the beginning of *pwordexp*–>**we_wordv**. In other words, *pwordexp*–>**we_wordv** will point to *pwordexp*–>**we_offs** **NULL** pointers, followed by *pwordexp*–>**we_wordc** word pointers, followed by a **NULL** pointer.

WRDE_NOCMD Fail if command substitution is requested.

WRDE_REUSE The *pwordexp* argument was passed to a previous successful call to **wordexp( )**, and has not been passed to **wordfree( )**. The result will be the same as if the application had called **wordfree( )** and then called **wordexp( )** without **WRDE_REUSE**.

WRDE_SHOWERR Do not redirect **stderr** to **/dev/null**.

WRDE_UNDEF Report error on an attempt to expand an undefined shell variable.

The **WRDE_APPEND** flag can be used to append a new set of words to those generated by a previous call to **wordexp( )**. The following rules apply when two or more calls to **wordexp( )** are made with the same value of *pwordexp* and without intervening calls to **wordfree( )**:

1. The first such call must not set **WRDE_APPEND**. All subsequent calls must set it.
2. All of the calls must set **WRDE_DOOFFS**, or all must not set it.
3. After the second and each subsequent call, *pwordexp*–>**we_wordv** will point to a list containing the following:
    a. zero or more **NULL** pointers, as specified by **WRDE_DOOFFS** and *pwordexp*–>**we_offs**.
    b. pointers to the words that were in the *pwordexp*–>**we_wordv** list before the call, in the same order as before.
    c. pointers to the new words generated by the latest call, in the specified order.
4. The count returned in *pwordexp*–>**we_wordc** will be the total number of words from all of the calls.
5. The application can change any of the fields after a call to **wordexp( )**, but if it does it must reset them to the original value before a subsequent call, using the same *pwordexp* value, to **wordfree( )** or **wordexp( )** with the **WRDE_APPEND** or **WRDE_REUSE** flag.

If *words* contains an unquoted:

NEWLINE **| & ; < > ( ) { }**

in an inappropriate context, **wordexp( )** will fail, and the number of expanded words will be zero.

Unless **WRDE_SHOWERR** is set in *flags*, **wordexp( )** will redirect **stderr** to **/dev/null** for any utilities executed as a result of command substitution while expanding *words*. If **WRDE_SHOWERR** is set, **wordexp( )** may write messages to *stderr* if syntax errors are detected while expanding *words*.

If **WRDE_DOOFFS** is set, then *pwordexp*–>**we_offs** must have the same value for each **wordexp( )** call and **wordfree( )** call using a given *pwordexp*.

The following constants are defined as error return values:

WRDE_BADCHAR	One of the unquoted characters: NEWLINE **|** **&** **;** **<** **>** **(** **)** **{** **}** appears in *words* in an inappropriate context.
WRDE_BADVAL	Reference to undefined shell variable when **WRDE_UNDEF** is set in *flags*.
WRDE_CMDSUB	Command substitution requested when **WRDE_NOCMD** was set in flags.
WRDE_NOSPACE	Attempt to allocate memory failed.
WRDE_SYNTAX	Shell syntax error, such as unbalanced parentheses or unterminated string.

RETURN VALUES

The following values are returned by **wordexp( )**:

0	successful completion.
non-zero	an error has occurred.
WRDE_NOSPACE	*pwordexp*–>**we_wordc** and *pwordexp*–>**we_wordv** will be updated to reflect any words that were successfully expanded. In other cases, they will not be modified.

The **wordfree( )** function returns no value.

USAGE

This function is intended to be used by an application that wants to do all of the shell's expansions on a word or words obtained from a user. For example, if the application prompts for a filename (or list of filenames) and then uses **wordexp( )** to process the input, the user could respond with anything that would be valid as input to the shell.

The **WRDE_NOCMD** flag is provided for applications that, for security or other reasons, want to prevent a user from executing shell commands. Disallowing unquoted shell special characters also prevents unwanted side effects such as executing a command or writing a file.

SEE ALSO

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