Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'wordexp.3p' command

$ man wordexp.3p

WORDEXP(3P) POSIX Programmer's Manual WORDEXP(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

wordexp, wordfree ? perform word expansions

SYNOPSIS

```
#include <wordexp.h>
int wordexp(const char *restrict words, wordexp_t *restrict pwordexp,
    int flags);
void wordfree(wordexp_t *pwordexp);
```

DESCRIPTION

The wordexp() function shall perform word expansions as described in the Shell and Utilities volume of POSIX.1?2017, Section 2.6, Word Ex? pansions, subject to quoting as described in the Shell and Utilities volume of POSIX.1?2017, Section 2.2, Quoting, and place the list of ex? panded words into the structure pointed to by pwordexp.

The words argument is a pointer to a string containing one or more words to be expanded. The expansions shall be the same as would be per? formed by the command line interpreter if words were the part of a com? mand line representing the arguments to a utility. Therefore, the ap?

plication shall ensure that words does not contain an unquoted <new? line> character or any of the unquoted shell special characters '|', '&', ';', '<', '>' except in the context of command substitution as specified in the Shell and Utilities volume of POSIX.1?2017, Section 2.6.3, Command Substitution. It also shall not contain unquoted paren? theses or braces, except in the context of command or variable substi? tution. The application shall ensure that every member of words which it expects to have expanded by wordexp() does not contain an unquoted initial comment character. The application shall also ensure that any words which it intends to be ignored (because they begin or continue a comment) are deleted from words. If the argument words contains an un? quoted comment character (<number-sign>) that is the beginning of a to? ken, wordexp() shall either treat the comment character as a regular character, or interpret it as a comment indicator and ignore the re? mainder of words.

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

Member Type	Member Name	Description
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 shall store 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 split? ting (see the Shell and Utilities volume of POSIX.1?2017, Section 2.6.5, Field Splitting) or pathname expansion (see the Shell and Utili? ties volume of POSIX.1?2017, Section 2.6.6, Pathname Expansion) shall be a separate word in the pwordexp->we_wordv list. The words shall be in order as described in the Shell and Utilities volume of

POSIX.1?2017, Section 2.6, Word Expansions. The first pointer after the last word pointer shall be a null pointer. The expansion of spe? cial parameters described in the Shell and Utilities volume of POSIX.1?2017, Section 2.5.2, Special Parameters is unspecified.

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

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 fol? lowing 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 point? ers to add to the beginning of pwordexp->we_wordv. In other words, pwordexp->we_wordv shall point to pword? exp->we_offs null pointers, followed by pword? exp->we_wordc word pointers, followed by a null pointer.

WRDE_NOCMD If the implementation supports the utilities defined in the Shell and Utilities volume of POSIX.1?2017, fail if command substitution, as specified in the Shell and Util? ities volume of POSIX.1?2017, Section 2.6.3, Command Sub? stitution, 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 shall 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 to applications 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 shall not set WRDE_APPEND. All subsequent calls shall set it.
2. All of the calls shall set WRDE_DOOFFS, or all shall not set it.
3. After the second and each subsequent call, pwordexp->we_wordv shall 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 shall be the total number of words from all of the calls.
5. The application can change any of the fields after a call to word? exp(), but if it does it shall reset them to the original value be? fore a subsequent call, using the same pwordexp value, to word? free() or wordexp() with the WRDE_APPEND or WRDE_REUSE flag.

If the implementation supports the utilities defined in the Shell and Utilities volume of POSIX.1?2017, and words contains an unquoted char? acter?<newline>, '|', '&', ';', '<', '>', '(', ')', '{', '}'?in an in? appropriate context, wordexp() shall fail, and the number of expanded words shall be 0.

Unless WRDE_SHOWERR is set in flags, wordexp() shall redirect stderr to /dev/null for any utilities executed as a result of command substitu? tion while expanding words. If WRDE_SHOWERR is set, wordexp() may write messages to stderr if syntax errors are detected while expanding words, unless the stderr stream has wide orientation in which case the behavior is undefined. It is unspecified whether any write errors en? countered while outputting such messages will affect the stderr error

indicator or the value of errno.

The application shall ensure that if WRDE_DOOFFS is set, then pword?exp->we_offs has the same value for each wordexp() call and wordfree() call using a given pwordexp.

The results are unspecified if WRDE_APPEND and WRDE_REUSE are both specified.

The following constants are defined as error return values:

WRDE_BADCHAR One of the unquoted characters?<newline>, '|', '&', ';', '<', '>', '(', ')', '{', '}'?appears in words in an inap?propriate 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 un?terminated string.

RETURN VALUE

Upon successful completion, wordexp() shall return 0. Otherwise, a non-zero value, as described in <wordexp.h>, shall be returned to indicate an error. If wordexp() returns the value WRDE_NOSPACE, then pword?exp->we_wordc and pwordexp->we_wordv shall be updated to reflect any words that were successfully expanded. In other error cases, if the WRDE_APPEND flag was specified, pwordexp->we_wordc and pword?exp->we_wordv shall not be modified.

The wordfree() function shall not return a value.

ERRORS

No errors are defined.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

The wordexp() 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 pathname (or list of pathnames) 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.

POSIX.1?2008 does not require the wordexp() function to be thread-safe if passed an expression referencing an environment variable while any other thread is concurrently modifying any environment variable; see exec.

Even though the WRDE_SHOWERR flag allows the implementation to write messages to stderr during command substitution or syntax errors, this standard does not provide any way to detect write failures during the output of such messages.

Applications which use wide-character output functions with stderr should ensure that any calls to wordexp() do not write to stderr, by avoiding use of the WRDE_SHOWERR flag.

RATIONALE

This function was included as an alternative to glob(). There had been continuing controversy over exactly what features should be included in glob(). It is hoped that by providing wordexp() (which provides all of the shell word expansions, but which may be slow to execute) and glob() (which is faster, but which only performs pathname expansion, without tilde or parameter expansion) this will satisfy the majority of appli? cations.

While wordexp() could be implemented entirely as a library routine, it is expected that most implementations run a shell in a subprocess to do the expansion.

Two different approaches have been proposed for how the required infor? mation might be presented to the shell and the results returned. They

are presented here as examples.

One proposal is to extend the echo utility by adding a -q option. This option would cause echo to add a <backslash> before each <backslash> and <blank> that occurs within an argument. The wordexp() function could then invoke the shell as follows:

```
(void) strcpy(buffer, "echo -q");
(void) strcat(buffer, words);
if ((flags & WRDE_SHOWERR) == 0)
    (void) strcat(buffer, "2>/dev/null");
f = popen(buffer, "r");
```

The wordexp() function would read the resulting output, remove unquoted <backslash> characters, and break into words at unquoted <blank> char? acters. If the WRDE_NOCMD flag was set, wordexp() would have to scan words before starting the subshell to make sure that there would be no command substitution. In any case, it would have to scan words for un? quoted special characters.

Another proposal is to add the following options to sh:

-w wordlist

This option provides a wordlist expansion service to applica? tions. The words in wordlist shall be expanded and the following written to standard output:

1. The count of the number of words after expansion, in decimal, followed by a null byte
2. The number of bytes needed to represent the expanded words (not including null separators), in decimal, followed by a null byte
3. The expanded words, each terminated by a null byte

If an error is encountered during word expansion, sh exits with a non-zero status after writing the former to report any words suc? cessfully expanded

-P Run in ``protected'' mode. If specified with the -w option, no command substitution shall be performed.

With these options, wordexp() could be implemented fairly simply by

creating a subprocess using fork() and executing sh using the line:

execl(<shell path>, "sh", "-P", "-w", words, (char *)0);

after directing standard error to /dev/null.

It seemed objectionable for a library routine to write messages to standard error, unless explicitly requested, so wordexp() is required to redirect standard error to /dev/null to ensure that no messages are generated, even for commands executed for command substitution. The WRDE_SHOWERR flag can be specified to request that error messages be written.

The WRDE_REUSE flag allows the implementation to avoid the expense of freeing and reallocating memory, if that is possible. A minimal imple? mentation can call wordfree() when WRDE_REUSE is set.

FUTURE DIRECTIONS

None.

SEE ALSO

exec, fnmatch(), glob()

The Base Definitions volume of POSIX.1?2017, <wordexp.h>

The Shell and Utilities volume of POSIX.1?2017, Chapter 2, Shell Com? mand Language

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1-2017, Standard for Information Technology -- Por? table Operating System Interface (POSIX), The Open Group Base Specifi? cations Issue 7, 2018 Edition, Copyright (C) 2018 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.ker? nel.org/doc/man-pages/reporting_bugs.html .