W

wc — Command

Count words, lines, and characters in files
wc [-clw] [*file…***]**

wc counts words, lines, and characters in each *file* named. If no *file* is given, **wc** uses the standard input. If more than one *file* is given, **wc** also prints a total for all of the files.

A *word* is a string of characters surrounded by white space (blanks, tabs, or newlines).

Options control the printing of various counts:

-c Print a count of character.

-l Print a count of lines.

-w Print a count of words.

The default action is to print all counts.

See Also

commands

wcstombs() — General utility (libc)

Convert sequence of wide characters to multibyte characters
#include <stdlib.h>
size_t wcstombs(wchar_t ******multibyte***, const char** ******widechar***, size_t** *number***);**

The function **wcstombs** converts a sequence of wide characters to their corresponding multibyte characters. It is the same as a series of calls of the type:

```
wctomb(multibyte, *widechar);
```

except that the call to **wcstombs** does not affect the internal state of **wctomb**.

widechars points to the base of the sequence of wide characters to be converted to multibyte characters. *multibyte* points to the area into which the characters will be written. The sequence begins and ends in an initial shift state. *number* is the number of characters to be converted. **wcstombs** converts characters either until it reads and converts the null character that ends the sequence, or until it has converted *number* characters. In the latter case, no null character is written at the end of the sequence of multibyte characters.

wcstombs returns -1 cast to **size_t** if it encounters an invalid wide character before it has converted *number* characters. Otherwise, it returns the number of characters converted, excluding the null character that ends the sequence.

Cross-reference

Standard, §4.10.7.4

See Also

general utilities, mbstowcs

Notes

The operation of this function is affected by the program's locale, as set by the function **setlocale**. See **localization** for more information.

LEXICON

wctomb() — General utility (libc)

Convert a wide character to a multibyte character
#include <stdlib.h>
int wctomb(char **string*, **wchar_t** *widecharacter*);

wctomb converts *widecharacter* to its corresponding multibyte character and stores the result in the area pointed to by *string*.

If *string* is set to NULL, then **wctomb** merely checks to see if the current set of multibyte characters include state-dependent encodings. It returns zero if the set does not include state-dependent codings, and a number other than zero if it does.

If *string* is set to a value other than NULL, then **wctomb** does the following:

1. It returns zero if *widecharacter* is zero.
2. It returns -1 if the value of *widecharacter* does not correspond to a legitimate multibyte character for the present locale.
3. If the value of *widecharacter* does correspond to a legitimate multibyte character, then it returns the number of bytes that comprise that character.

wctomb never returns a value greater than that of the macro **MB_CUR_MAX**.

Cross-reference

Standard, §4.10.7.5

See Also

general utilities, MB_CUR_MAX, mblen, mbtowc, wchar_t

Notes

The operation of this function is affected by the program's locale, as set by the function **setlocale**. See **localization** for more information.

The address pointed to by *string* should have **MB_CUR_MAX** bytes of storage allocated to it. If not, you may overwrite memory currently in use.

while — C keyword

Loop construct
while(*condition***)** *statement*

while introduces a conditional loop. Unlike a **do** loop, a **while** loop tests *condition* before execution of *statement*. The loop ends when *condition* is no longer satisfied. Hence, the loop may not execute at all, if *condition* is initially false.

For example,

```
while (variable < 10)
```

introduces a loop whose statements will continue to execute until **variable** is equivalent to ten or greater. The statement

```
while (1)
```

will loop until interrupted by **break**, **goto**, or **return**.

Example

For an example of this statement, see **sscanf**.

Cross-references

Standard, §3.6.5.1
The C Programming Language, ed. 2, pp. 60*ff*

See Also

C keywords, do, for, statements, while

wildcards — Definition

Wildcards are characters that, under special circumstances, represent a range of ASCII characters. Another name for them is "metacharacters". The wildcards available under MS-DOS are as follows:

? Match any one character.

***** Match any number of characters, or no characters at all.

See Also

Definitions, egrep, patterns, pnmatch

write() — Extended function (libc)

Write into a file
short write(short *fd***, char** ******buffer***, short** *n***);**

write writes *n* bytes of data, beginning at address *buffer*, into the file *fd*. Writing begins at the current write position, as set by the last call to either **write** or **lseek**. **write** advances the position of the file pointer by the number of characters written.

write returns -1 if an error occurred before the **write** operation commenced, such as if *fd* is bad or *buffer* contains an invalid address. Otherwise, it returns the number of bytes actually written. It should be considered an error if this number is not the same as *n*.

Example

For an example of how to use this function, see the entry for **open**.

See Also

extended miscellaneous

Notes

write is a low-level call that passes data directly to MS-DOS. It should not be intermixed with high-level calls, such as **fread**, **fwrite**, or **fopen** without care.

write is not described in the ANSI Standard. Programs that use it do not conform strictly to the Standard, and may not be portable to other compilers or to other environments.