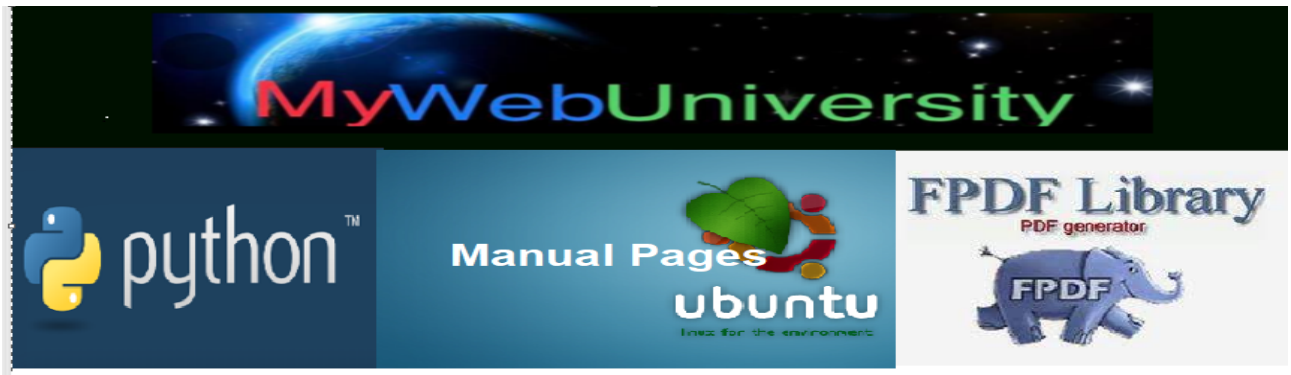

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'mbstowcs.3'

$ man mbstowcs.3

mbstowcs(3) Library Functions Manual mbstowcs(3)

NAME

mbstowcs - convert a multibyte string to a wide-character string

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

#include <stdlib.h>

size_t mbstowcs(size_t dsize;

wchar_t dest[restrict dsize], const char *restrict src,

size_t dsize);

DESCRIPTION

If dest is not NULL, convert the multibyte string src to a wide-character string starting at dest. At most dsize wide characters are written to dest. The sequence of characters in the string src shall begin in the initial shift state. The conversion can stop for three rea? sons:

? An invalid multibyte sequence has been encountered. In this case, (size_t) -1 is re? turned.

? dsize non-L'\0' wide characters have been stored at dest. In this case, the number of wide characters written to dest is returned, but the shift state at this point is lost.

? The multibyte string has been completely converted, including the terminating null char? acter ('\0'). In this case, the number of wide characters written to dest, excluding the terminating null wide character, is returned.

If dest is NULL, dsize is ignored, and the conversion proceeds as above, except that the

converted wide characters are not written out to memory, and that no length limit exists.

In order to avoid the case 2 above, the programmer should make sure dsize is greater than or equal to mbstowcs(NULL,src,0)+1.

The programmer must ensure that there is room for at least dsize wide characters at dest.

RETURN VALUE

The number of wide characters that make up the converted part of the wide-character string, not including the terminating null wide character. If an invalid multibyte sequence was en? countered, (size_t) -1 is returned.

ATTRIBUTES

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

Interface	Attribute	Value
mbstowcs()	Thread safety	MT-Safe

VERSIONS

The function mbsrtowcs(3) provides a better interface to the same functionality.

STANDARDS

C11, POSIX.1-2008.

HISTORY

POSIX.1-2001, C99.

NOTES

The behavior of mbstowcs() depends on the LC_CTYPE category of the current locale.

EXAMPLES

The program below illustrates the use of mbstowcs(), as well as some of the wide character classification functions. An example run is the following:

```
$ ./t_mbstowcs de_DE.UTF-8 Gr??e!
Length of source string (excluding terminator):
    8 bytes
    6 multibyte characters
Wide character string is: Gr??e! (6 characters)
    G alpha upper
    r alpha lower
```

```
        ? alpha lower
        ? alpha lower
        e alpha lower
        ! !alpha
Program source
    #include <locale.h>
    #include <stdio.h>
    #include <stdlib.h>
    #include <string.h>
    #include <wchar.h>
    #include <wctype.h>
    int
    main(int argc, char *argv[])
    {
        size_t mbslen;      /* Number of multibyte characters in source */
        wchar_t *wcs;       /* Pointer to converted wide character string */
        if (argc < 3) {
            fprintf(stderr, "Usage: %s <locale> <string>\n", argv[0]);
            exit(EXIT_FAILURE);
        }
        /* Apply the specified locale.  */
        if (setlocale(LC_ALL, argv[1]) == NULL) {
            perror("setlocale");
            exit(EXIT_FAILURE);
        }
        /* Calculate the length required to hold argv[2] converted to
           a wide character string.  */
        mbslen = mbstowcs(NULL, argv[2], 0);
        if (mbslen == (size_t) -1) {
            perror("mbstowcs");
            exit(EXIT_FAILURE);
        }
        /* Describe the source string to the user.  */
```

```
printf("Length of source string (excluding terminator):\n");
printf("    %zu bytes\n", strlen(argv[2]));
printf("    %zu multibyte characters\n\n", mbslen);
/* Allocate wide character string of the desired size.  Add 1
   to allow for terminating null wide character (L'\0').  */
wcs = calloc(mbslen + 1, sizeof(*wcs));
if (wcs == NULL) {
    perror("calloc");
    exit(EXIT_FAILURE);
}
/* Convert the multibyte character string in argv[2] to a
   wide character string.  */
if (mbstowcs(wcs, argv[2], mbslen + 1) == (size_t) -1) {
    perror("mbstowcs");
    exit(EXIT_FAILURE);
}
printf("Wide character string is: %ls (%zu characters)\n",
       wcs, mbslen);
/* Now do some inspection of the classes of the characters in
   the wide character string.  */
for (wchar_t *wp = wcs; *wp != 0; wp++) {
    printf("    %lc ", (wint_t) *wp);
    if (!iswalpha(*wp))
        printf("!");
    printf("alpha ");
    if (iswalpha(*wp)) {
        if (iswupper(*wp))
            printf("upper ");
        if (iswlower(*wp))
            printf("lower ");
    }
    putchar('\n');
}
```

```
        exit(EXIT_SUCCESS);
    }
```

SEE ALSO

mblen(3), mbsrtowcs(3), mbtowc(3), wcstombs(3), wctomb(3)