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 'wchar.h.0p' command

$ man wchar.h.0p

wchar.h(0P) POSIX Programmer's Manual wchar.h(0P)

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

wchar.h ? wide-character handling

SYNOPSIS

#include <wchar.h>

DESCRIPTION

Some of the functionality described on this reference page extends the ISO C standard. Applications shall define the appropriate feature test macro (see the System Interfaces volume of POSIX.1?2017, Section 2.2, The Compilation Environment) to enable the visibility of these symbols in this header.

The <wchar.h> header shall define the following types:

FILE As described in <stdio.h>.

locale_t As described in <locale.h>.

mbstate_t An object type other than an array type that can hold the conversion state information necessary to convert between sequences of (possibly multi-byte) characters and wide characters. If a codeset is being used such that an mb?

state_t needs to preserve more than two levels of reserved state, the results are unspecified.

size_t As described in <stddef.h>.

va_list As described in <stdarg.h>.

wchar_t As described in <stddef.h>.

wctype_t A scalar type of a data object that can hold values which represent locale-specific character classification.

wint_t An integer type capable of storing any valid value of wchar_t or WEOF.

The tag tm shall be declared as naming an incomplete structure type, the contents of which are described in the <time.h> header.

The implementation shall support one or more programming environments in which the width of wint_t is no greater than the width of type long.

The names of these programming environments can be obtained using the confstr() function or the getconf utility.

The <wchar.h> header shall define the following macros:

WCHAR_MAX As described in <stdint.h>.

WCHAR_MIN As described in <stdint.h>.

WEOF Constant expression of type wint_t that is returned by several WP functions to indicate end-of-file.

NULL As described in <stddef.h>.

Inclusion of the <wchar.h> header may make visible all symbols from the headers <ctype.h>, <string.h>, <stdarg.h>, <stddef.h>, <stdio.h>, <stdlib.h>, and <time.h>.

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided for use with ISO C standard compilers. Arguments to functions in this list can point to arrays containing wchar_t values that do not correspond to members of the character set of the current locale. Such values shall be processed according to the specified semantics, unless otherwise stated.

```
wint_t      btowc(int);
wint_t      fgetwc(FILE *);
wchar_t     *fgetws(wchar_t *restrict, int, FILE *restrict);
```

```
wint_t      fputwc(wchar_t, FILE *);
int         fputws(const wchar_t *restrict, FILE *restrict);
int         fwide(FILE *, int);
int         fwprintf(FILE *restrict, const wchar_t *restrict, ...);
int         fwscanf(FILE *restrict, const wchar_t *restrict, ...);
wint_t      getwc(FILE *);
wint_t      getwchar(void);
int         iswalnum(wint_t);
int         iswalpha(wint_t);
int         iswcntrl(wint_t);
int         iswctype(wint_t, wctype_t);
int         iswdigit(wint_t);
int         iswgraph(wint_t);
int         iswlower(wint_t);
int         iswprint(wint_t);
int         iswpunct(wint_t);
int         iswspace(wint_t);
int         iswupper(wint_t);
int         iswxdigit(wint_t);
size_t      mbrlen(const char *restrict, size_t, mbstate_t *restrict);
size_t      mbrtowc(wchar_t *restrict, const char *restrict, size_t,
              mbstate_t *restrict);
int         mbsinit(const mbstate_t *);
size_t      mbsnrtowcs(wchar_t *restrict, const char **restrict,
              size_t, size_t, mbstate_t *restrict);
size_t      mbsrtowcs(wchar_t *restrict, const char **restrict, size_t,
              mbstate_t *restrict);
FILE        *open_wmemstream(wchar_t **, size_t *);
wint_t      putwc(wchar_t, FILE *);
wint_t      putwchar(wchar_t);
int         swprintf(wchar_t *restrict, size_t,
              const wchar_t *restrict, ...);
int         swscanf(const wchar_t *restrict,
```

```
                const wchar_t *restrict, ...);
wint_t      towlower(wint_t);
wint_t      towupper(wint_t);
wint_t      ungetwc(wint_t, FILE *);
int         vfwprintf(FILE *restrict, const wchar_t *restrict, va_list);
int         vfwscanf(FILE *restrict, const wchar_t *restrict, va_list);
int         vswprintf(wchar_t *restrict, size_t,
                const wchar_t *restrict, va_list);
int         vswscanf(const wchar_t *restrict, const wchar_t *restrict,
                va_list);
int         vwprintf(const wchar_t *restrict, va_list);
int         vwscanf(const wchar_t *restrict, va_list);
wchar_t     *wcpcpy(wchar_t *restrict, const wchar_t *restrict);
wchar_t     *wcpncpy(wchar_t *restrict, const wchar_t *restrict, size_t);
size_t      wcrtomb(char *restrict, wchar_t, mbstate_t *restrict);
int         wcscasecmp(const wchar_t *, const wchar_t *);
int         wcscasecmp_l(const wchar_t *, const wchar_t *, locale_t);
wchar_t     *wcscat(wchar_t *restrict, const wchar_t *restrict);
wchar_t     *wcschr(const wchar_t *, wchar_t);
int         wcscmp(const wchar_t *, const wchar_t *);
int         wcscoll(const wchar_t *, const wchar_t *);
int         wcscoll_l(const wchar_t *, const wchar_t *, locale_t);
wchar_t     *wcscpy(wchar_t *restrict, const wchar_t *restrict);
size_t      wcscspn(const wchar_t *, const wchar_t *);
wchar_t     *wcsdup(const wchar_t *);
size_t      wcsftime(wchar_t *restrict, size_t,
                const wchar_t *restrict, const struct tm *restrict);
size_t      wcslen(const wchar_t *);
int         wcsncasecmp(const wchar_t *, const wchar_t *, size_t);
int         wcsncasecmp_l(const wchar_t *, const wchar_t *, size_t,
                locale_t);
wchar_t     *wcsncat(wchar_t *restrict, const wchar_t *restrict, size_t);
int         wcsncmp(const wchar_t *, const wchar_t *, size_t);
```

```
wchar_t      *wcsncpy(wchar_t *restrict, const wchar_t *restrict, size_t);
size_t       wcsnlen(const wchar_t *, size_t);
size_t       wcsnrtombs(char *restrict, const wchar_t **restrict, size_t,
                size_t, mbstate_t *restrict);
wchar_t      *wcspbrk(const wchar_t *, const wchar_t *);
wchar_t      *wcsrchr(const wchar_t *, wchar_t);
size_t       wcsrtombs(char *restrict, const wchar_t **restrict,
                size_t, mbstate_t *restrict);
size_t       wcsspn(const wchar_t *, const wchar_t *);
wchar_t      *wcsstr(const wchar_t *restrict, const wchar_t *restrict);
double       wcstod(const wchar_t *restrict, wchar_t **restrict);
float        wcstof(const wchar_t *restrict, wchar_t **restrict);
wchar_t      *wcstok(wchar_t *restrict, const wchar_t *restrict,
                wchar_t **restrict);
long         wcstol(const wchar_t *restrict, wchar_t **restrict, int);
long double  wcstold(const wchar_t *restrict, wchar_t **restrict);
long long    wcstoll(const wchar_t *restrict, wchar_t **restrict, int);
unsigned long wcstoul(const wchar_t *restrict, wchar_t **restrict, int);
unsigned long long
             wcstoull(const wchar_t *restrict, wchar_t **restrict, int);
int          wcswidth(const wchar_t *, size_t);
size_t       wcsxfrm(wchar_t *restrict, const wchar_t *restrict, size_t);
size_t       wcsxfrm_l(wchar_t *restrict, const wchar_t *restrict,
                size_t, locale_t);
int          wctob(wint_t);
wctype_t     wctype(const char *);
int          wcwidth(wchar_t);
wchar_t      *wmemchr(const wchar_t *, wchar_t, size_t);
int          wmemcmp(const wchar_t *, const wchar_t *, size_t);
wchar_t      *wmemcpy(wchar_t *restrict, const wchar_t *restrict, size_t);
wchar_t      *wmemmove(wchar_t *, const wchar_t *, size_t);
wchar_t      *wmemset(wchar_t *, wchar_t, size_t);
int          wprintf(const wchar_t *restrict, ...);
```

int wscanf(const wchar_t *restrict, ...);

The following sections are informative.

APPLICATION USAGE

The iswblank() function was a late addition to the ISO C standard and was introduced at the same time as the ISO C standard introduced <wc?type.h>, which contains all of the isw*() functions. The Open Group Base Specifications had previously aligned with the MSE working draft and had introduced the rest of the isw*() functions into <wchar.h>. For backwards-compatibility, the original set of isw*() functions, without iswblank(), are permitted (as part of the XSI option) in <wchar.h>. For maximum portability, applications should include <wc?type.h> in order to obtain declarations for the isw*() functions. This compatibility has been made obsolescent.

RATIONALE

In the ISO C standard, the symbols referenced as XSI extensions are in <wctype.h>. Their presence here is thus an extension.

FUTURE DIRECTIONS

None.

SEE ALSO

<ctype.h>, <locale.h>, <stdarg.h>, <stddef.h>, <stdint.h>, <stdio.h>, <stdlib.h>, <string.h>, <time.h>, <wctype.h>

The System Interfaces volume of POSIX.1?2017, Section 2.2, The Compila?tion Environment, btowc(), confstr(), fgetwc(), fgetws(), fputwc(), fputws(), fwide(), fwprintf(), fwscanf(), getwc(), getwchar(), iswal?num(), iswalpha(), iswcntrl(), iswctype(), iswdigit(), iswgraph(), iswlower(), iswprint(), iswpunct(), iswspace(), iswupper(), iswxdigit(), mbrlen(), mbrtowc(), mbsinit(), mbsrtowcs(), open_mem?stream(), putwc(), putwchar(), towlower(), towupper(), ungetwc(), vfw?printf(), vfwscanf(), wcrtomb(), wcscasecmp(), wcscat(), wcschr(), wc?scmp(), wcscoll(), wcscpy(), wcscspn(), wcsdup(), wcsftime(), wcslen(), wcsncat(), wcsncmp(), wcsncpy(), wcspbrk(), wcsrchr(), wcsrtombs(), wc?sspn(), wcsstr(), wcstod(), wcstok(), wcstol(), wcstoul(), wcswidth(), wcsxfrm(), wctob(), wctype(), wcwidth(), wmemchr(), wmemcmp(), wmem?

cpy(), wmemmove(), wmemset()

The Shell and Utilities volume of POSIX.1?2017, getconf

COPYRIGHT