W

wchar.h[C95] A header that declares machinery supporting multibyte and wide character processing. It contains definitions or declarations for the identifiers in the following table:

Name	*Purpose*
`btowc`	Test for valid character
`fgetwc`	Read wide character from file
`fgetws`	Read wide string from file
`fputwc`	Write wide character to file
`fputws`	Write wide string to file
`fwide`	Determine orientation of stream
`fwprintf`	Formatted write to file
`fwscanf`	Formatted read from file
`getwc`	Read wide character from `stdin`
`getwchar`	Read wide character from `stdin`
`mbrlen`	Find length of multibyte string
`mbrtowc`	Convert multibyte character to wide character
`mbsinit`	Check for initial conversion state
`mbsrtowcs`	Convert multibyte string to wide character
`mbstate_t`	Conversion state object type
`NULL`	Null pointer constant
`putwc`	Write wide character to `stdout`
`putwchar`	Write wide character to `stdout`
`size_t`	Size/count type
`swprintf`	Formatted write to wide string
`swscanf`	Formatted read from wide string
`tm`	A calendar time type
`ungetwc`	Pushback wide character to `stdin`
`vfwprintf`	Formatted write to a file
`vfwscanf`	Formatted read from a file
`vswprintf`	Formatted write to a wide string
`vswscanf`	Formatted read from a wide string
`vwprintf`	Formatted write to `stdout`
`vwscanf`	Formatted read from `stdin`
`WCHAR_MAX`	Max value of type `wchar_t`
`WCHAR_MIN`	Min value of type `wchar_t`
`wchar_t`	Wide character type
`wcrtomb`	Convert wide character to multibyte character
`wcsrtombs`	Convert wide string to multibyte string

Name	*Purpose*
`wcscat`	Concatenate wide strings
`wcschr`	Search wide string for wide character
`wcscmp`	Compare wide strings
`wcscoll`	Collate wide strings
`wcscpy`	Copy a wide string
`wcscspn`	Compute initial length match
`wcserror`	Produce error message
`wcsftime`	Format a time into a wide string
`wcslen`	Compute wide string length
`wcsncat`	Concatenate wide strings
`wcsncmp`	Compare leading part of wide strings
`wcsncpy`	Copy leading part of wide string
`wcspbrk`	Locate wide character in wide string
`wcsrchr`	Reverse search for wide character
`wcsspn`	Compute initial length match
`wcsstr`	Search wide string for wide string
`wcstod`	Convert wide string to `double`
`wcstof`	Convert wide string to `float`
`wcstok`	Break wide string into tokens
`wcstol`	Convert wide string to `long`
`wcstold`	Convert wide string to `long double`
`wcstoll`	Convert wide string to `long long`
`wcstoul`	Convert wide string to `unsigned long`
`wcstoull`	Convert wide string to `unsigned long long`
`wcsxfrm`	Transform wide string
`wctob`	Test if multibyte character
`WEOF`	Wide end-of-file
`wint_t`	Type to hold any character
`wmemchr`	Search memory for wide character
`wmemcmp`	Compare memory blocks
`wmemcpy`	Copy memory blocks
`wmemmove`	Safe copy of memory blocks
`wmemset`	Initialize memory block
`wprintf`	Formatted write to `stdout`
`wscanf`	Formatted read from `stdin`

See future library directions.

`WCHAR_MAX`[C95,C99] A macro, defined in stdint.h, that indicates the maximum value of the type `wchar_t`. It expands to an integer constant expression suitable for use with a `#if` directive. C95 defined it in wchar.h, and C99 added it to stdint.h as well. If that type is a signed integer, `WCHAR_MAX` shall be no less than 127; if the type is an unsigned integer, `WCHAR_MAX` shall be no less than 255.

`WCHAR_MIN`[C95,C99] A macro, defined in stdint.h, that indicates the minimum value of the type `wchar_t`. It expands to an integer constant expression suitable for use with a `#if` directive. C95 defined it in wchar.h, and C99 added it to stdint.h as well. If that type is an unsigned integer, `WCHAR_MIN` shall be no greater than -127; if the type is an unsigned integer, `WCHAR_MIN` shall be zero.

`wchar_t`[C89] An integer type that has a range of values capable of representing distinct codes for all members of the largest extended character set specified among the supported locales. `wchar_t` is needed when declaring or referencing wide characters and wide strings. It is defined in stddef.h, stdlib.h, and wchar.h. Note that in C++, this name is a keyword. *See also* Unicode; `WCHAR_MAX`; `WCHAR_MIN`

`wcrtomb`[C95] A function that is a restartable version of `wctomb`.

```
#include <wchar.h>
size_t wcrtomb(char * restrict s,
    wchar_t wc, mbstate_t * restrict ps);
```

`wcs` **prefix** *See* future library directions.

`wcscat`[C95] A function that copies the wide string pointed to by `s2` to the end of the wide string pointed to by `s1`. In the process, the trailing wide null of `s1` is overwritten by the first wide character of the wide string pointed to by `s2`. The destination wide string is terminated with a wide null.

```
#include <wchar.h>
wchar_t *wcscat(wchar_t * restrict s1,
    const wchar_t * restrict s2);
```

The value of `s1` is returned. If the wide strings located at `s1` and `s2` overlap, the behavior of `wcscat` is undefined. *See also* `strcat`.

`wcschr`[C95] A function that searches the wide string `s` for a wide character `c`.

```
#include <wchar.h>
wchar_t *wcschr(const wchar_t *s, wchar_t c);
```

If `c` is found, `wcschr` returns a pointer to that location within `s`; otherwise it returns `NULL`. The wide null that terminates `s` is included in the search; so `wcschr` can be used to locate the trailing null as well. *See also* `strchr`.

`wcscmp`[C95] A function that compares a wide string at the location pointed to by `s2` to a wide string at the location pointed to by `s1`.

```
#include <wchar.h>
int wcscmp(const wchar_t *s1, const wchar_t *s2);
```

An integer less than, equal to, or greater than zero is returned to indicate whether `s1` is less than, equal to, or greater than `s2`, respectively. *See also* `strcmp`.

`wcscoll`[C95] A function that compares a wide string at the location pointed to by `s2` to a wide string at the location pointed to by `s1`.

```
#include <wchar.h>
int wcscoll(const wchar_t *s1, const wchar_t *s2);
```

An integer less than, equal to, or greater than zero is returned to indicate whether `s1` is less than, equal to, or greater than `s2`, respectively. The comparison is locale specific (based on the value of `LC_COLLATE`). In the `"C"` locale, `wcscoll` should give the same result as `wcscmp`. *See also* `strcoll`.

`wcscpy`[C95] A function that copies the wide string pointed to by `s2` to the location pointed to by `s1`. The destination wide string is terminated with a wide null.

```
#include <wchar.h>
wchar_t *wcscpy(wchar_t * restrict s1,
    const wchar_t * restrict s2);
```

The value of `s1` is returned. If the wide strings located at `s1` and `s2` overlap, the behavior of `wcscpy` is undefined. To copy overlapping objects, use `wmemmove` instead. *See also* `strcpy`.

`wcscspn`[C95] A function that finds the longest wide string starting at the location pointed to by `s1`, such that it does not contain any of the wide characters in the wide string pointed to by `s2`. Alternatively, it could be viewed that `wcscspn` locates the first wide character in the string pointed to by `s1` that is present in the wide string pointed to by `s2`.

```
#include <wchar.h>
size_t wcscspn(const wchar_t *s1, const wchar_t *s2);
```

The return value indicates the length of the nonmatching wide string found at `s1` (which is the same as the subscript of the first matching wide character in `s1`). *See also* `strcspn`.

`wcsftime`[C95] A function that constructs a date and time wide string by putting wide characters into the array pointed to by `s` according to the controlling format specified by `format`. No more than `maxsize` wide characters

are written to the array. `timeptr` points to a structure whose contents are used to determine the appropriate wide characters for the array.

```
#include <time.h>
#include <wchar.h>
size_t wcsftime(wchar_t * restrict s, size_t maxsize,
    const wchar_t * restrict format,
    const struct tm * restrict timeptr);
```

This function provides a locale-specific way of generating date and time wide strings. Refer to your library documentation for specific details of the controlling format conversion specifiers. *See also* `strftime`.

`wcslen`[C95] A function that returns the number of wide characters in the wide string (excluding the trailing wide null) pointed to by `s`.

```
#include <wchar.h>
size_t wcslen(const wchar_t *s);
```

`wcsncat`[C95] A function that copies, at most, the first `n` wide characters from the wide string pointed to by `s2` to the end of the wide string pointed to by `s1`. The destination wide string is terminated with a wide null.

```
#include <wchar.h>
wchar_t *wcsncat(wchar_t * restrict s1,
    const wchar_t * restrict s2, size_t n);
```

The value of `s1` is returned. If a wide null is not found in the first `n` wide characters of `s2`, `n` wide characters will be copied followed by a wide null. If a wide null is found, it is copied to `s1` and the copy is terminated. If the wide strings located at `s1` and `s2` overlap, the behavior of `wcsncat` is undefined. *See also* `strncat`.

`wcsncmp`[C95] A function that compares no more than `n` wide characters from a wide string at the location pointed to by `s2` to a wide string at the location pointed to by `s1`. If a wide null is seen in the first `n` wide characters, the comparison is terminated.

```
#include <wchar.h>
int wcsncmp(const wchar_t *s1, const wchar_t *s2, size_t n);
```

An integer less than, equal to, or greater than zero is returned to indicate whether `s1` is less than, equal to, or greater than `s2`, respectively. *See also* `strncmp`.

`wcsncpy`[C95] A function that copies, at most, the first `n` wide characters from the wide string pointed to by `s2` to the location pointed to by `s1`. The destination string is terminated with a wide null character.

```
#include <wchar.h>
wchar_t *wcsncpy(wchar_t * restrict s1,
    const wchar_t * restrict s2, size_t n);
```

The value of `s1` is returned. If a wide null is not found in the first `n` wide characters of `s2`, the wide string pointed to by `s1` will not be null terminated. If a wide null is found, it is copied to `s1`. If there are less than `n` wide characters in the string pointed to by `s1`, extra null wide characters are appended to the end of the string pointed to by `s1` such that `n` wide characters are actually copied there. If the wide strings located at `s1` and `s2` overlap, the behavior of `wcsncpy` is undefined. To copy overlapping objects, use `wmemmove` instead. *See also* `strncpy`.

`wcspbrk`[C95] A function that searches the wide string pointed to by `s1` for the first of any one of the wide characters in the wide string pointed to by `s2`.

```
#include <wchar.h>
wchar_t *wcspbrk(const wchar_t *s1, const wchar_t *s2);
```

The return value is a pointer to the wide character found in `s1` or `NULL` if no match was found. *See also* `strpbrk`.

`wcsrchr`[C95] A function that searches the wide string `s` for the last occurrence of the wide character `c`.

```
#include <wchar.h>
wchar_t *wcsrchr(const wchar_t *s, wchar_t c);
```

If `c` is found, `wcsrchr` returns a pointer to that location within `s`; otherwise it returns `NULL`. The wide null terminating `s` is included in the search, so `wcsrchr` can be used to locate the trailing wide null as well. *See also* `strrchr`.

`wcsrtombs`[C95] A function that is a restartable version of `wcstomb`.

```
#include <wchar.h>
size_t wcsrtombs(char * restrict dst,
    const wchar_t ** restrict src,
    size_t len, mbstate_t * restrict ps);
```

`wcsspn`[C95] A function that finds the longest wide string starting at the location pointed to by `s1` such that it contains only those wide characters in the

wide string at the location pointed to by `s2`. Alternatively, it could be viewed that `wcsspn` locates the first wide character in the string pointed to by `s1` that is not present in the wide string pointed to by `s2`.

```
#include <wchar.h>
size_t wcsspn(const wchar_t *s1, const wchar_t *s2);
```

The return value indicates the length of the matching wide string found at `s1` (which is the same as the subscript of the first nonmatching wide character in `s1`). *See also* `strspn`.

`wcsstr`[C99] A function that searches the wide string pointed to by `s1` for the wide substring pointed to by `s2`.

```
#include <wchar.h>
wchar_t *wcsstr(const wchar_t *s1, const wchar_t *s2);
```

The return value represents the location of the wide substring `s2` within the wide string `s1`. If the substring is not found, `NULL` is returned. By definition, the null wide string is always found at the beginning of any wide string, including a null string. *See also* `strstr`.

`wcstod`[C95] A function that is the wide character equivalent to `strtod`.

```
#include <wchar.h>
double wcstod(const wchar_t * restrict nptr,
    wchar_t ** restrict endptr);
```

`wcstof`[C99] A function that is the wide character equivalent to `strtof`.

```
#include <wchar.h>
float wcstof(const wchar_t * restrict nptr,
    wchar_t ** restrict endptr);
```

`wcstoimax`[C99] A function that is equivalent to the `wcstol` and `wcstoll` functions, except that the initial portion of the string is converted to type `intmax_t`.

```
#include <stddef.h>
#include <inttypes.h>
intmax_t wcstoimax(const wchar_t * restrict nptr,
    wchar_t ** restrict endptr, int base);
```

If no conversion could be performed, zero is returned. If the correct value is outside the range of representable values, `INTMAX_MAX` or `INTMAX_MIN` is returned (depending on the sign of the value), and `errno` is set to `ERANGE`.

`wcstok`[C95] Successive calls to this function can be used to break a wide string, pointed to by `s1`, into a series of wide null-terminated tokens using the token terminator wide characters specified by the wide string pointed to by `s2`.

```
#include <wchar.h>
wchar_t *wcstok(wchar_t * restrict s1,
    const wchar_t * restrict s2, wchar_t ** restrict ptr);
```

The value returned is either a pointer to the token found or `NULL` if no token was found.

Unlike `strtok`, `wcstok` has a third argument, which points to a user-provided `wchar_t` pointer into which the `wcstok` function stores information necessary for it to continue scanning the same wide string in a future call.

`wcstol`[C95] A function that is the wide character equivalent to `strtol`.

```
long int wcstol(const wchar_t * restrict nptr,
    wchar_t ** restrict endptr, int base);
```

`wcstold`[C99] A function that is the wide character equivalent to `strtold`.

```
#include <wchar.h>
long double wcstold(const wchar_t * restrict nptr,
    wchar_t ** restrict endptr);
```

`wcstoll`[C99] A function that is the wide character equivalent to `strtoll`.

```
long long int wcstoll(const wchar_t * restrict nptr,
    wchar_t ** restrict endptr, int base);
```

`wcstombs`[C89] A function that converts a sequence of wide characters in `pwcs` to a sequence of multibyte characters pointed to by `s`.

```
#include <stdlib.h>
int wcstombs(char * restrict s, wchar_t * restrict pwcs,
    size_t n);
```

See also `wcsrtombs`.

`wctomb`[C89] A function that determines the number of bytes needed to represent the multibyte character corresponding to the code whose value is `wchar`. It also converts that wide character `wchar` to a multibyte character which it stores at `s`.

```
#include <stdlib.h>
int wctomb(char *s, wchar_t wchar);
```

See also `wcrtomb`.

`wcstoul`[C95] A function that is the wide character equivalent to `strtoul`.

```
unsigned long int wcstoul(const wchar_t * restrict nptr,
    wchar_t ** restrict endptr, int base);
```

`wcstoull`[C99] A function that is the wide character equivalent to `strtoull`.

```
unsigned long long int wcstoull(const wchar_t * restrict
    nptr, wchar_t ** restrict endptr, int base);
```

`wcstoumax`[C99] A function that is equivalent to the `wcstoul` and `wcstoull` functions, except that the initial portion of the string is converted to type `uintmax_t`.

```
#include <stddef.h>
#include <inttypes.h>
uintmax_t wcstoumax(const wchar_t * restrict nptr,
    wchar_t ** restrict endptr, int base);
```

If no conversion could be performed, zero is returned. If the correct value is outside the range of representable values, `UINTMAX_MAX` is returned, and `errno` is set to `ERANGE`.

`wcsxfrm`[C95] A function that transforms the wide string pointed to by `s2` into another wide string pointed to by `s1`. The transformation is locale specific.

```
#include <wchar.h>
size_t wcsxfrm(wchar_t * restrict s1,
    const wchar_t * restrict s2, size_t n);
```

If two wide strings are transformed, and the resultant wide strings are compared using `wcscmp`, the same result should be obtained as if the original two strings had been compared using `wcscoll`. No more than `n` transformed wide characters are to be written into `s1`. If the transformed wide string is bigger than `n`, only `n` wide characters are copied to `s1`, and the function returns the total number needed to hold the whole transformed string. Normally, `n` is large enough and the return value is the number of wide characters actually used in `s1` (excluding the wide null). If `s1` and `s2` designate wide strings that overlap, the result is undefined. *See also* `strxfrm`.

`wctob`[C95] A function that indicates whether its argument `c` corresponds to a member of the extended character set whose multibyte character representation is a single byte when in the initial shift state. If there is such a correspondence, the wide character is converted.

```
#include <stdio.h>
#include <wchar.h>
int wctob(wint_t c);
```

If `c` does not correspond to a multibyte character with length one in the initial shift state, `EOF` is returned; otherwise, the single-byte representation of that character as an `unsigned char` converted to an `int` is returned.

`wctrans`[C95] A function that constructs a value of type `wctrans_t` which describes a mapping between wide characters identified by `property`.

```
#include <wctype.h>
wctrans_t wctrans(const char *property);
```

The strings recognized by this function and their corresponding meaning are as follows:

String	*Class*
`"tolower"`	`towlower`
`"toupper"`	`towupper`

These strings are valid in all locales.

If `property` represents a valid mapping of wide characters according to the `LC_CTYPE` category of the current locale, the nonzero value returned is valid as the second argument to `towctrans`; otherwise, zero is returned.

`wctrans_t`[C95] A scalar type, defined in wctype.h, that is capable of holding values that represent locale-specific character mappings. *See also* `towctrans`; `wctrans`.

`wctype`[C95] A function that constructs a value of type `wctype_t` which describes a class of wide characters identified by the string argument `property`.

```
#include <wctype.h>
wctype_t wctype(const char *property);
```

The strings recognized by this function and their corresponding meaning are as follows:

String	*Class*
"alnum"	iswalnum
"alpha"	iswalpha
"blank"	iswblank
"cntrl"	iswcntrl
"digit"	iswdigit
"graph"	iswgraph
"lower"	iswlower
"print"	iswprint
"punct"	iswpunct
"space"	iswspace
"upper"	iswupper
"xdigit"	iswxdigit

These strings are valid in all locales.

If `property` identifies a valid class of wide characters according to the `LC_CTYPE` category of the current locale, this function returns a nonzero value that is valid as the second argument to the function `iswctype`; otherwise, it returns zero.

wctype.h[C95] A header that contains various character testing and conversion functions, some type definitions, and a macro. The `isw*` family members return a zero or nonzero value based on the truth of their operation while the `tow*` family members return a possibly case-converted value of their character argument.

All functions take one `wint_t` argument. However, the `wint_t` argument must be either representable in a `wchar_t` or it must be the macro `WEOF`. If the argument has any other value, the behavior is undefined.

The behavior of some functions is locale specific.

This header contains definitions or declarations for the following identifiers:

Name	*Purpose*
`iswalnum`	Test if wide character is alphanumeric
`iswalpha`	Test if wide character is alphabetic
`iswblank`[C99]	Test if wide character is blank
`iswcntrl`	Test if wide character is control
`iswctype`	Test if some character classification
`iswdigit`	Test if wide character is digit (0–9)
`iswgraph`	Test if wide character is graphic
`iswlower`	Test if wide character is lowercase
`iswprint`	Test if wide character is printable
`iswpunct`	Test if wide character is punctuation

Name	*Purpose*
`iswspace`	Test if wide character is space
`iswupper`	Test if wide character is uppercase
`iswxdigit`	Test if wide character is hex digit
`towctrans`	Map a character
`towlower`	Produce lowercase version
`towupper`	Produce uppercase version
`wint_t`	Type to hold any character
`wctrans`	Convert mapping name to value
`wctrans_t`	Character mapping value
`wctype`	Convert classification name to value
`wctype_t`	Character classification value
`WEOF`	Wide character end-of-file indicator

See also ctype.h; future library directions.

`wctype_t`[C95] A scalar type, defined in wctype.h, that is capable of holding values that represent locale-specific character classifications. *See also* `iswctype`; `wctype`.

`WEOF`[C95] A macro, defined in wchar.h and wctype.h, that is a constant expression of type `wint_t` which is returned by numerous functions to indicate an end-of-file condition. *See also* `EOF`.

WG14 Known formally as ISO/IEC JTC 1 SC22/WG14, this standards committee is responsible for the production and maintenance of what is generally known as the ISO C standard. Its U.S. counterpart is J11.

WG15 Known formally as ISO/IEC JTC 1 SC22/WG15, this standards committee is responsible for the production and maintenance of POSIX, the Portable Operating System Interface for Computer Environments. Its U.S. counterpart is IEEE P1003.

WG21 Known formally as ISO/IEC JTC 1 SC22/WG21, this standards committee is responsible for the production and maintenance of what is generally known as the ISO C++ standard. Its U.S. counterpart is J16.

`while` A looping construct that evaluates its controlling expression before each iteration of the loop, like `for` and unlike `do`/`while`, which always executes at least once. A `while` construct can always be rewritten as a `for` construct and vice versa. It is used as follows:

`while (` *expression* `)`
statement

expression is evaluated. If it tests false, the body of the `while` statement is bypassed. If it tests true, *statement* is executed. The process is then repeated. The equivalent `for` construct is

```
for ( ; expression ; )
          statement
```

expression is a full expression.

white space One or more adjacent space, horizontal tab, vertical tab, formfeed, and new-line characters that separate adjacent source tokens. A comment may occur any place white space is permitted and is replaced by a single space character.

wide character[C89] *See* character, wide.

wide character constant[C89] *See* constant, character, wide.

wide string[C89] An array of `wchar_t` terminated by a null wide character. *See also* string; string literal; string literal, wide.

wide string literal[C89] *See* string literal, wide.

wide type *See* type, wide.

widening The conversion of an expression from a narrower to a wider type. For example, `char` and `short` expressions typically are promoted to type `int`, and `float` is promoted to `double`, when used in expressions such as function call arguments. (Standard C permits narrow types to be kept as such via the appropriate use of prototypes, although an implementation is not obliged to do so.)

`WINT_MAX`[C99] A macro, defined in stdint.h, that indicates the maximum value of the type `wint_t`. It expands to an integer constant expression suitable for use with a `#if` directive. If that type is a signed integer, `WINT_MAX` shall be no less than 32767; if the type is an unsigned integer, `WINT_MAX` shall be no less than 65535.

`WINT_MIN`[C99] A macro, defined in stdint.h, that indicates the minimum value of the type `wint_t`. It expands to an integer constant expression suitable for use with a `#if` directive. If that type is an unsigned integer, `WINT_MIN` shall be no greater than -32767; if the type is an unsigned integer, `WINT_MIN` shall be zero.

`wint_t`[C95] An integer type, defined in wchar.h and wctype.h, that is capable of holding any character from the extended character set, as well as at least one value that does not correspond to any member of the extended character set (where one such value is reserved for `WEOF`). *See also* `WINT_MAX`; `WINT_MIN`.

`wmemchr`[C95] A function that searches the first `n` wide characters of a wide string `s` for a wide character `c`.

```
#include <wchar.h>
wchar_t *wmemchr(const wchar_t *s, wchar_t c, size_t n);
```

If `c` is found, `wmemchr` returns a pointer to that location within `s`, otherwise, it returns `NULL`. *See also* `memchr`.

`wmemcmp`[C95] A function that compares `n` wide characters at the location pointed to by `s2` to the wide characters at the location pointed to by `s1`.

```
#include <wchar.h>
int wmemcmp(const wchar_t * s1, const wchar_t * s2,
    size_t n);
```

An integer less than, equal to, or greater than zero is returned to indicate whether the first `n` characters starting at `s1` have binary values less than, equal to, or greater than, respectively, those starting at `s2`. *See also* `memcmp`.

`wmemcpy`[C95] A function that copies `n` wide characters from the location pointed to by `s2` to the wide characters at the location pointed to by `s1`.

```
#include <wchar.h>
wchar_t *wmemcpy(wchar_t * restrict s1,
    const wchar_t * restrict s2, size_t n);
```

The value of `s1` is returned. If the objects located at `s1` and `s2` overlap, the behavior of `wmemcpy` is undefined. To copy overlapping objects, use `wmemmove` instead. *See also* `memcpy`.

`wmemmove`[C95] A function that copies `n` wide characters from the location pointed to by `s2` to the wide characters at the location pointed to by `s1`.

```
#include <wchar.h>
wchar_t *wmemmove(wchar_t *s1, const wchar_t *s2, size_t n);
```

The value of `s1` is returned. `wmemmove` works correctly even if the objects located at `s1` and `s2` overlap. If the objects are known to not overlap, it may be more efficient to use `wmemcpy` instead. *See also* `memmove`.

`wmemset`[C95] A function that sets the first `n` wide characters of the object pointed to by `s` to the value `c`.

```
#include <wchar.h>
wchar_t *wmemset(wchar_t *s, wchar_t c, size_t n);
```

The value returned is `s`. *See also* `memset`.

`wprintf`[C95] The wide character analog of `printf`.

```
#include <wchar.h>
int wprintf(const wchar_t * restrict format, ...);
```

`wscanf`[C95] The wide character analog of `scanf`.

```
#include <wchar.h>
int wscanf(const wchar_t * restrict format, ...);
```

♢ ♢ ♢