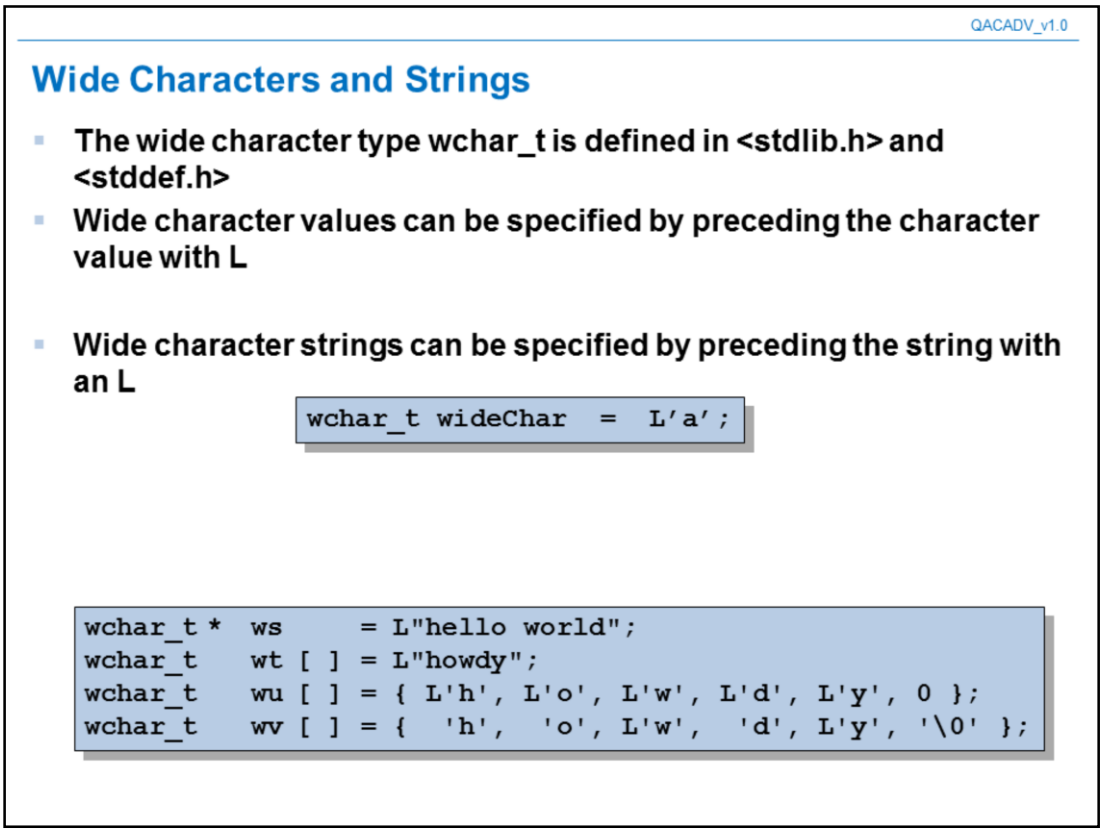

QACADV_v1.0
Appendix - Internationalisation
Advanced C
transforming performance
through learning

QACADV_v1.0

Internationalisation

- **K&R C was linked implicitly to American English and the Roman character-set, specifically the ASCII characters**
- **Now an implementation may support multiple locales, each with a different character set, date format, etc.**
- **There is always a default "C" locale which has generally accepted (USA/Latin) defaults**
- **Implementation may define others, but are not required by the standard**
- **Support for "wide" (multi-byte) characters for non-Latin character-sets**
  - L prefix to character constants and strings
  - Various functions in library



The standard specifies that the language provides support for the following:

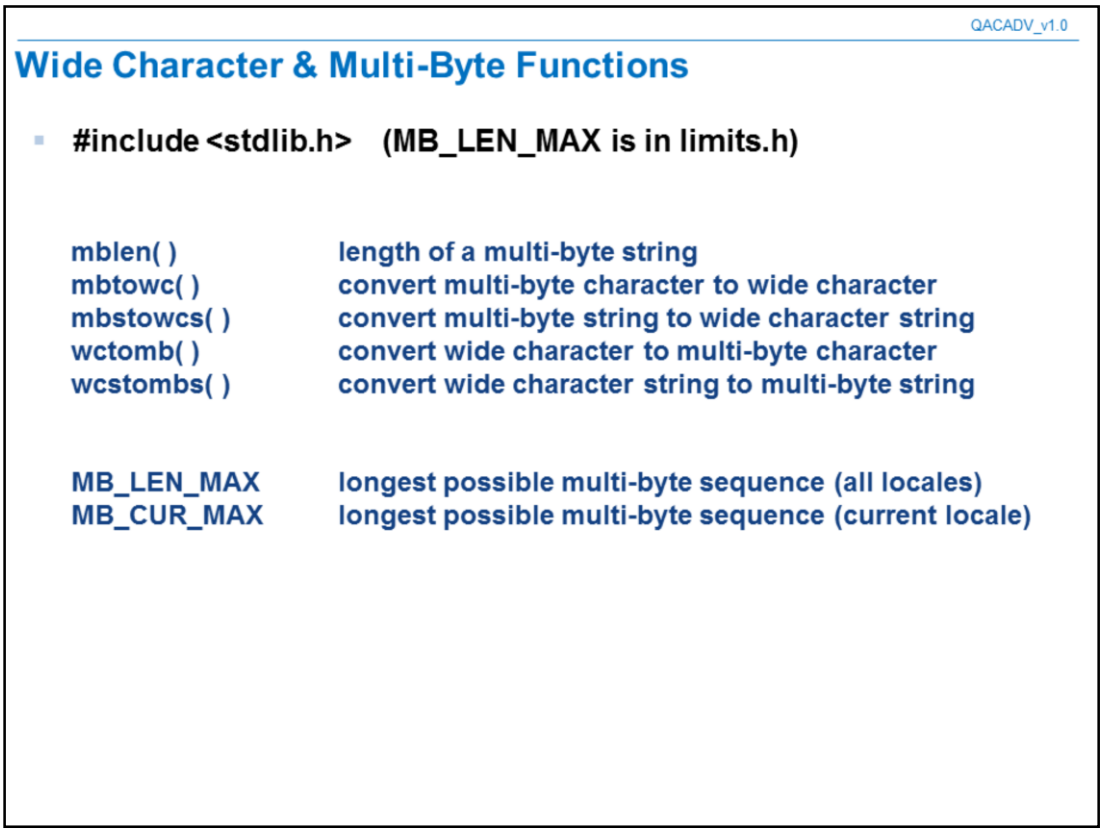
A single-byte set representing the minimal C character-set.

An additional single-byte set that is implementation specific.

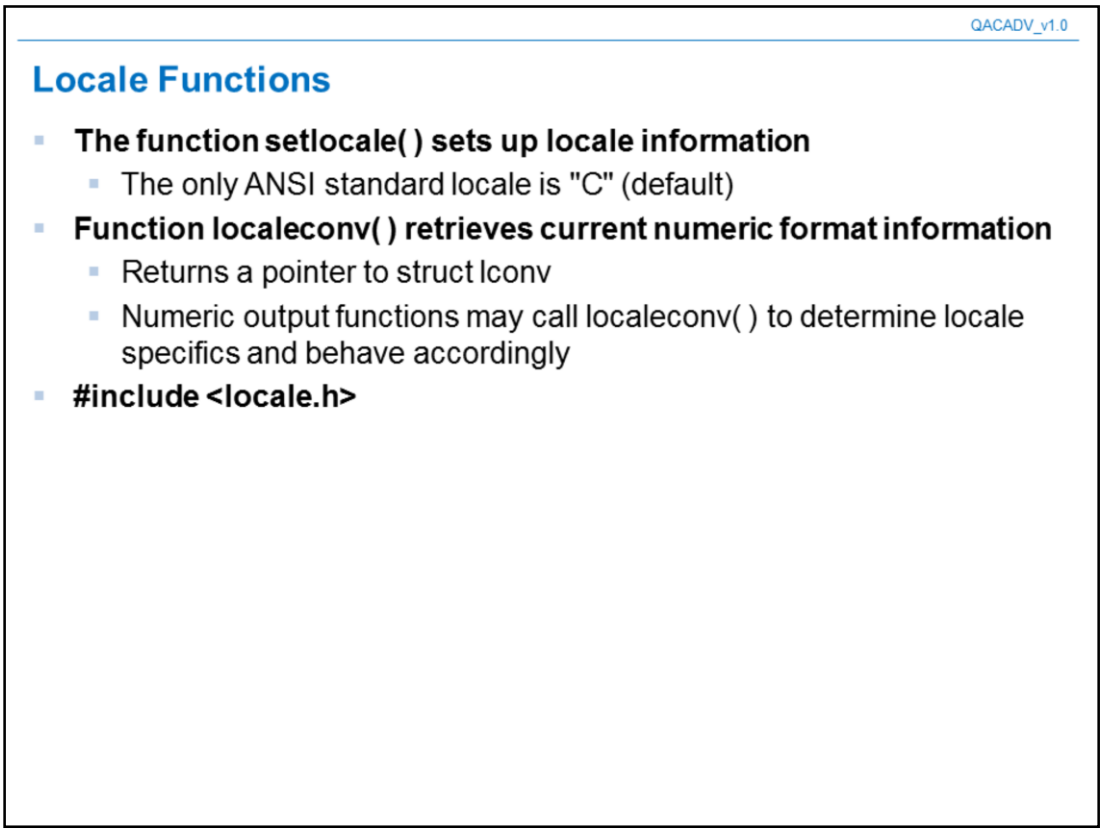
A further multi-byte set that is implementation specific.

A multi-byte character is one which requires a 2 or more byte sequence. This is a natural requirement for non-Latin alphabets such as Kanji. The byte sequence representation relies on s*tate dependent encoding*, i.e. an interpretation which depends on bytes early on in the sequence. With this method, characters from both the conventional C character-set or from alternative single-byte sets can be easily represented. Each character is also associated with an integer representation, referred to as *wide character encoding* and the integer type used is `typedef`'d in `stddef.h` and named `wchar_t`.

The prefix `L` is used to specify a wide char constant, a string of wide chars and an array of wide chars.



Multi-byte sequences are used for general processing such as input and output. The wide char is merely a 16-bit integer used for convenience within the program. It is important to be able to convert between the two representations. A conversion between the multi-byte and wide character representation is provided by the standard functions `mblen`, `mbstowcs`, `mbtowc`, `wcstombs` and `wctomb` (found in `stdlib.h`) and by two macros `MB_LEN_MAX` and `MB_CUR_MAX` (which are found in `limits.h` and `stdlib.h` respectively).

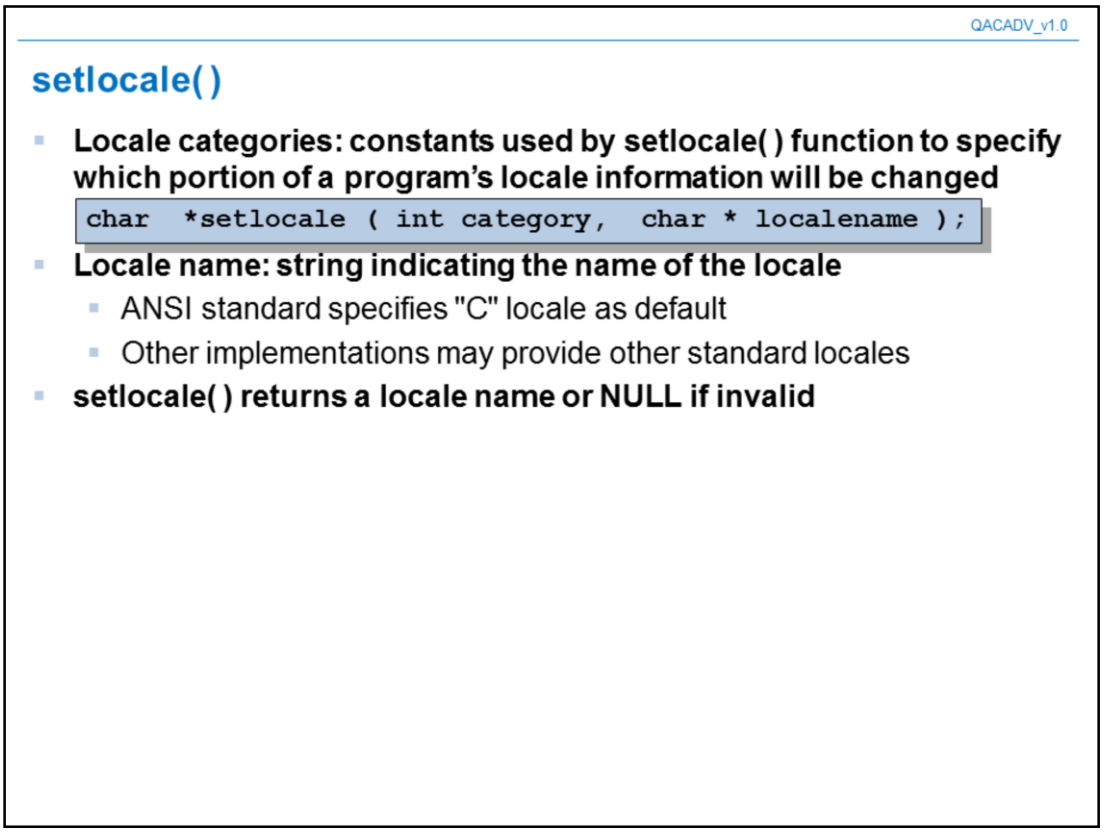


A "locale" summarises the conventions used by a particular spoken language, country, set of nations, and so on. Conventions include, for example, the way in which the time and date are formatted and monetary values displayed. The conventions are stored as members in a `struct` of type `lconv`. The locale is set to a particular convention at start up and can be changed during program execution.

The locale mechanism is supported through the header `locale.h` that defines `NULL`, several macros which all begin with "`LC_`", and `struct lconv`, which is to hold the conventions and the prototypes for the two functions `setlocale()` and `localeconv()`.

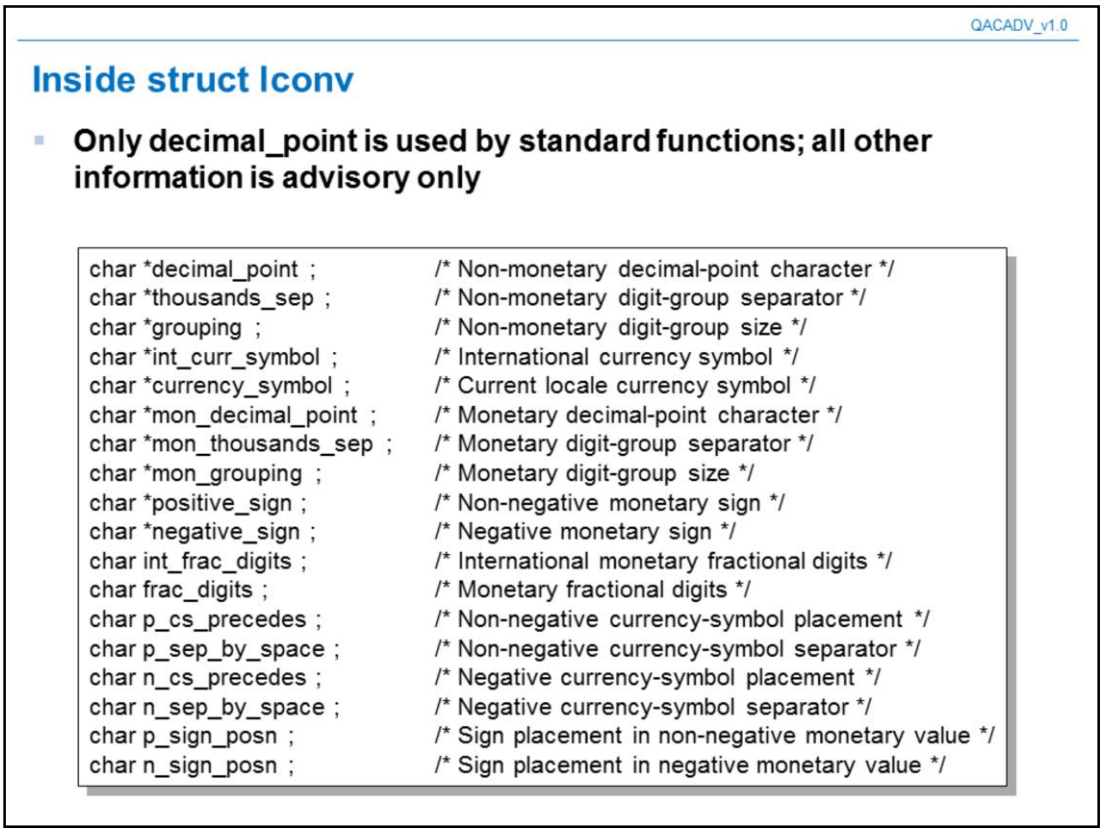
The name of the locale is held in a string which is used as a handle into the `setlocale` function. The default locale is named "`C`" - the only one defined in the standard.

The `localeconv` function returns a pointer to the appropriate `lconv` structure. The conventional settings for the current locale stored in this `struct` can then be accessed directly through this pointer. Many library functions make use of the `localeconv` for formatting information.



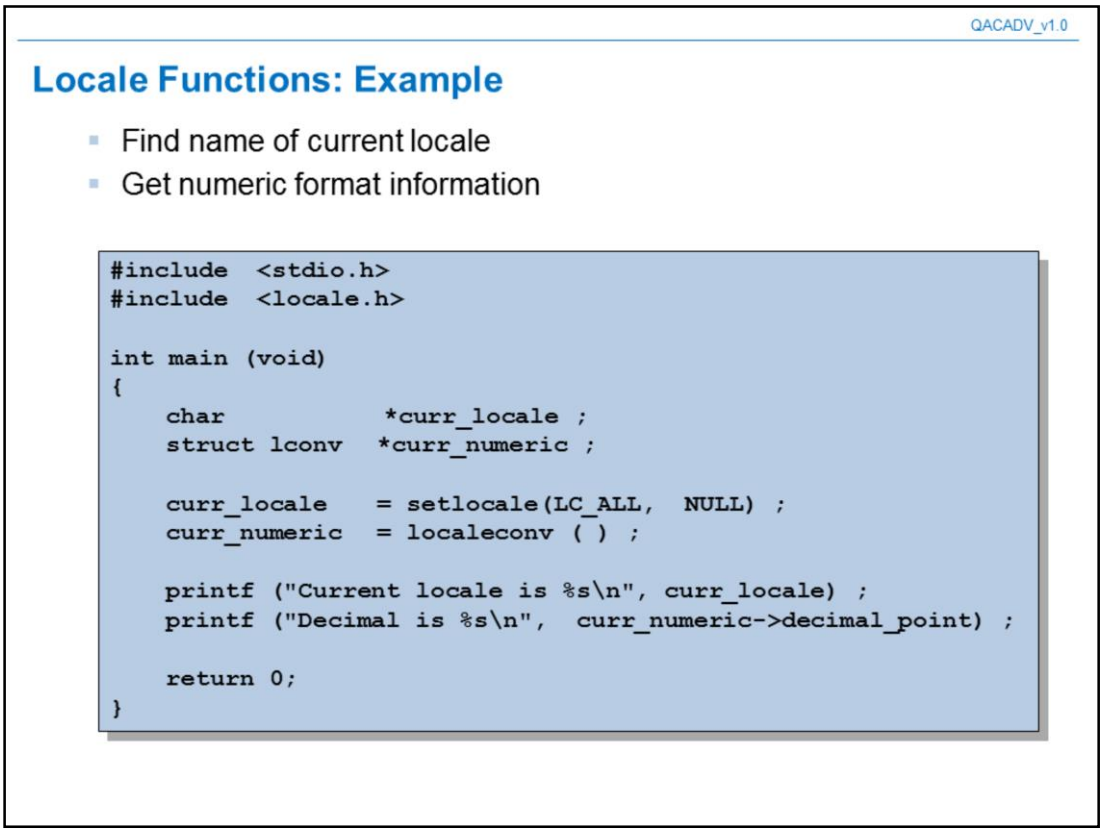
The category refers to one of the `LC_` macros defined in `locale.h`. These include some or all of the following:

`LC_COLLATE`	Affects collating used in `strcoll` and `strxfrm`.
`LC_CTYPE`	Affects character handling and multi-byte functions.
`LC_MONETARY`	Affects monetary information returned by `localeconv`.
`LC_NUMERIC`	Affects decimal point information returned by `localeconv` and decimal point processing used in numeric conversion and I/O functions.
`LC_TIME`	Affects time conversion in the `strftime` function.
`LC_ALL`	Affects all categories.



The above list of members are not in any recognised order and can be added to or replaced by others - there is no standard set.

The `char*` members all point to character strings, e.g. `"DM"`. The `char` members contain integer values or `CHAR_MAX` when not appropriate for a particular locale convention. Some of the `char` members represent intended integer values, e.g. `frac_digits` has the value 2, some are codes and others are booleans.



This example shows how to determine the name of the current locale via the `setlocale()` function, and then determine the symbol representing a decimal point in the current locale using the `localeconv()` function. The name of the default locale is "`C`" and the character "." is used in the United Kingdom for a decimal point.

Other locale settings can be obtained in a similar way by inspecting the various members of the `struct lconv` variable.

Note that the header file `locale.h` must be included for the locale prototypes, structure template and `#define` values.