Chapter 21

The Standard Library

Using the Library

- The C89 standard library is divided into 15 parts, with each part described by a header.
- C99 has an additional nine headers.

<assert.h>	<inttypes.h>†	<signal.h>	<stdlib.h>
<complex.h>†	<iso646.h>†	<stdarg.h>	<string.h>
<ctype.h>	<limits.h>	<stdbool.h>†	<tgmath.h>†
<errno.h>	<locale.h>	<stddef.h>	<time.h>
<fenv.h>†	<math.h>	<stdint.h>†	<wchar.h>†
<float.h>	<setjmp.h>	<stdio.h>	<wctype.h>†

†C99 only

Using the Library

- Most compilers come with a more extensive library that has additional (nonstandard) headers.
- Nonstandard headers often provide:
  - Functions that are specific to a particular computer or operating system
  - Functions that allow more control over the screen and keyboard
  - Support for graphics or a window-based user interface

Using the Library

- The standard headers consist primarily of function prototypes, type definitions, and macro definitions.
- When a file includes several standard headers, the order of `#include` directives doesn't matter.
- It's also legal to include a standard header more than once.

Restrictions on Names Used in the Library

- Any file that includes a standard header must obey two rules:
  - The names of macros defined in that header can't be used for any other purpose.
  - Library names with file scope (`typedef` names, in particular) can't be redefined at the file level.

Restrictions on Names Used in the Library

- Other restrictions are less obvious:
  - ***Identifiers that begin with an underscore followed by an upper-case letter or a second underscore*** are reserved for use within the library.
  - ***Identifiers that begin with an underscore*** are reserved for use as identifiers and tags with file scope.
  - ***Every identifier with external linkage in the standard library*** is reserved for use as an identifier with external linkage. In particular, the names of all standard library functions are reserved.

Restrictions on Names Used in the Library

- These rules apply to *every* file in a program, regardless of which headers the file includes.
- Moreover, they apply not just to names that are currently used in the library, but also to names that are set aside for future use.
- For example, C reserves identifiers that begin with `str` followed by a lower-case letter.

Functions Hidden by Macros

- The C standard allows headers to define macros that have the same names as library functions, but requires that a true function be available as well.
- It's not unusual for a library header to declare a function *and* define a macro with the same name.

Functions Hidden by Macros

- `getchar` is a library function declared in the `<stdio.h>` header:

```
int getchar(void);
```

- `<stdio.h>` usually defines `getchar` as a macro as well:

```
#define getchar() getc(stdin)
```

- By default, a call of `getchar` will be treated as a macro invocation.

Functions Hidden by Macros

- A macro is usually preferable to a true function, because it will probably improve the speed of a program.
- Occasionally, a genuine function is needed, perhaps to minimize the size of the executable code.

Functions Hidden by Macros

- A macro definition can be removed (thus gaining access to the true function) by using `#undef`:

```
#include <stdio.h>
#undef getchar
```

- `#undef` has no effect when given a name that's not defined as a macro.

Functions Hidden by Macros

- Individual uses of a macro can be disabled by putting parentheses around its name:

```
ch = (getchar)();
  /* instead of ch = getchar(); */
```

- The preprocessor can't spot a parameterized macro unless its name is followed by a left parenthesis.
- However, the compiler can still recognize `getchar` as a function.

C89 Library Overview

`<assert.h>` ***Diagnostics***

Contains only the `assert` macro, which can be used to insert self-checks into a program. If any check fails, the program terminates.

`<ctype.h>` ***Character Handling***

Provides functions for classifying characters and for converting letters from lower to upper case or vice versa.

C89 Library Overview

`<errno.h>` ***Errors***

Provides `errno` ("error number"), an lvalue that can be tested after a call of certain library functions to see if an error occurred.

`<float.h>` ***Characteristics of Floating Types***

Provides macros that describe the characteristics of floating types, including their range and accuracy.

C89 Library Overview

`<limits.h>` ***Sizes of Integer Types***

Provides macros that describe the characteristics of integer types (including character types), including their maximum and minimum values.

`<locale.h>` ***Localization***

Provides functions to help a program adapt its behavior to a country or other geographic region.

C89 Library Overview

<math.h> ***Mathematics***

Provides common mathematical functions.

<setjmp.h> ***Nonlocal Jumps***

Provides the `setjmp` and `longjmp` functions. `setjmp` "marks" a place in a program; `longjmp` can then be used to return to that place later.

C89 Library Overview

<signal.h> ***Signal Handling***

Provides functions that deal with exceptional conditions (signals).

- The `signal` function installs a function to be called if a given signal should occur later.
- The `raise` function causes a signal to occur.

<stdarg.h> ***Variable Arguments***

Provides tools for writing functions that can have a variable number of arguments.

C89 Library Overview

`<stddef.h>` ***Common Definitions***

Provides definitions of frequently used types and macros.

`<stdio.h>` ***Input/Output***

Provides a large assortment of input/output functions, including operations on both sequential and random-access files.

C89 Library Overview

`<stdlib.h>` ***General Utilities***

Provides functions that perform the following operations:

- Converting strings to numbers
- Generating pseudo-random numbers
- Performing memory management tasks
- Communicating with the operating system
- Searching and sorting
- Performing conversions between multibyte characters and wide characters

C89 Library Overview

`<string.h>` ***String Handling***

Provides functions that perform string operations, as well as functions that operate on arbitrary blocks of memory.

`<time.h>` ***Date and Time***

Provides functions for determining the time (and date), manipulating times, and formatting times for display.

C99 Library Changes

- Some of the biggest changes in C99 affect the standard library:
  - ***Additional headers.*** The C99 standard library has nine headers that don't exist in C89.
  - ***Additional macros and functions.*** C99 adds macros and functions to several existing headers (especially `<math.h>`).
  - ***Enhanced versions of existing functions.*** Some existing functions, including `printf` and `scanf`, have additional capabilities in C99.

C99 Library Changes

`<complex.h>` ***Complex Arithmetic***

Defines the `complex` and `I` macros.

Provides functions for performing mathematical operations on complex numbers.

`<fenv.h>` ***Floating-Point Environment***

Provides access to floating-point status flags and control modes.

C99 Library Changes

<inttypes.h> ***Format Conversion of Integer Types***

Defines macros that can be used in format strings for input/output of the integer types declared in <stdint.h>.

Provides functions for working with greatest-width integers.

<iso646.h> ***Alternative Spellings***

Defines macros representing the operators whose symbols contain the characters &, |, ~, !, and ^.

C99 Library Changes

`<stdbool.h>` ***Boolean Type and Values***

Defines the `bool`, `true`, and `false` macros, as well as a macro that can be used to test whether these macros have been defined.

`<stdint.h>` ***Integer Types***

Declares integer types with specified widths and defines related macros.

Defines parameterized macros that construct integer constants with specific types.

C99 Library Changes

<tgmath.h> ***Type-Generic Math***

Provides "type-generic" macros that can detect argument types and substitute a call of a <math.h> or <complex.h> function.

<wchar.h> ***Extended Multibyte and Wide-Character Utilities***

Provides functions for wide-character input/output and wide string manipulation.

C99 Library Changes

`<wctype.h>` ***Wide-Character Classification and Mapping Utilities***

The wide-character version of `<ctype.h>`.

Provides functions for classifying and changing the case of wide characters.

The <stddef.h> Header: Common Definitions

- Types defined in <stddef.h>:
  - ptrdiff_t. The type of the result when two pointers are subtracted.
  - size_t. The type returned by the sizeof operator.
  - wchar_t. A type large enough to represent all possible characters in all supported locales.
- Macros defined in <stddef.h>:
  - NULL. Represents the null pointer.
  - offsetof. Computes the number of bytes between the beginning of a structure and one of its members.

The <stddef.h> Header: Common Definitions

- An example structure:

```
struct s {
  char a;
  int b[2];
  float c;
};
```

- The value of `offsetof(struct s, a)` must be 0, but the offsets of `b` and `c` depend on the compiler.
- One possibility is that `offsetof(struct s, b)` is 1, and `offsetof(struct s, c)` is 9.
- If a compiler should leave a three-byte hole after `a`, the offsets of `b` and `c` would be 4 and 12.

The <stdbool.h> Header (C99): Boolean Type and Values

- Macros defined in <stdbool.h>:

  `bool` (defined to be `_Bool`)

  `true` (defined to be `1`)

  `false` (defined to be `0`)

  `__bool_true_false_are_defined` (defined to be `1`)

- A program could use a preprocessing directive to test the last of these before attempting to define its own version of `bool`, `true`, or `false`.