Uses of Conditional Compilation

Conditional compilation is certainly handy for debugging, but its uses don't stop there. Here are a few other common applications:

- ***Writing programs that are portable to several machines or operating systems.*** The following example includes one of three groups of lines depending on whether `WIN32`, `MAC_OS`, or `LINUX` is defined as a macro:

  ```
  #if defined(WIN32)
  ...
  #elif defined(MAC_OS)
  ...
  #elif defined(LINUX)
  ...
  #endif
  ```

  A program might contain many of these `#if` blocks. At the beginning of the program, one (and only one) of the macros will be defined, thereby selecting a particular operating system. For example, defining the `LINUX` macro might indicate that the program is to run under the Linux operating system.

- ***Writing programs that can be compiled with different compilers.*** Different compilers often recognize somewhat different versions of C. Some accept a standard version of C, some don't. Some provide machine-specific language extensions; some don't, or provide a different set of extensions. Conditional compilation can allow a program to adjust to different compilers. Consider the problem of writing a program that might have to be compiled using an older, nonstandard compiler. The `__STDC__` macro allows the preprocessor to detect whether a compiler conforms to the standard (either C89 or C99); if it doesn't, we may need to change certain aspects of the program. In particular, we may have to use old-style function declarations (discussed in the Q&A at the end of Chapter 9) instead of function prototypes. At each point where functions are declared, we can put the following lines:

  ```
  #if __STDC__
  Function prototypes
  #else
  Old-style function declarations
  #endif
  ```

- ***Providing a default definition for a macro.*** Conditional compilation allows us to check whether a macro is currently defined and, if not, give it a default definition. For example, the following lines will define the macro `BUFFER_SIZE` if it wasn't previously defined:

  ```
  #ifndef BUFFER_SIZE
  #define BUFFER_SIZE 256
  #endif
  ```