straggling semicolon.

**Just a Moment**

Sometimes it's useful to build a time delay into a program. For example, you might have encountered programs that flash a message onscreen and then go on to something else before you can read it. You're left with the fear that you've missed irretrievable information of vital importance. It would be so much nicer if the program paused five seconds before moving on. The **while** loop is handy for producing this effect. One of the earlier techniques was to make the computer count for a while to use up time:

```c
long wait = 0;
while (wait < 10000)
    wait++;            // counting silently
```

The problem with this approach is that you have to change the counting limit when you change computer processor speed. Several games written for the original IBM PC, for example, became unmanageably fast when run on its faster successors. A better approach is to let the system clock do the timing for you.

The ANSI C and the C++ libraries have a function to help you do this. The function is called `clock()`, and it returns the system time elapsed since a program started execution. There are a couple of complications. First, `clock()` doesn't necessarily return the time in seconds. Second, the function's return type might be `long` on some systems, `unsigned long` on others, or perhaps some other type.

But the `ctime` header file (time.h on less current implementations) provides solutions to these problems. First, it defines a symbolic constant, `CLOCKS_PER_SEC`, that equals the number of system time units per second. So dividing the system time by this value yields seconds. Or, you can multiply seconds by `CLOCKS_PER_SEC` to get time in the system units. Second, `ctime` establishes `clock_t` as an alias for the `clock()` return type. (See the note about **Type Aliases**.) This means you can declare a variable as type `clock_t` and the compiler then converts that to `long` or `unsigned int` or whatever the proper type is for your system.

**Compatibility Notes**