```
typedef unsigned short WORD;
```

We'll use the BYTE and WORD types in later examples.

Using Unions to Provide Multiple Views of Data

Although unions can be used in a portable way—see Section 16.4 for examples—they're often used in C for an entirely different purpose: viewing a block of memory in two or more different ways.

Here's a simple example based on the file_date structure described in Section 20.2. Since a file_date structure fits into two bytes, we can think of any two-byte value as a file_date structure. In particular, we could view an unsigned short value as a file_date structure (assuming that short integers are 16 bits long). The following union allows us to easily convert a short integer to a file date or vice versa:

```
union int_date {
  unsigned short i;
  struct file_date fd;
};
```

With the help of this union, we could fetch a file date from disk as two bytes, then extract its month, day, and year fields. Conversely, we could construct a date as a file_date structure, then write it to disk as a pair of bytes.

As an example of how we might use the int_date union, here's a function that, when passed an unsigned short argument, prints it as a file date:

```
void print_date(unsigned short n)
{
  union int_date u;

  u.i = n;
  printf("%d/%d/%d\n", u.fd.month, u.fd.day, u.fd.year + 1980);
}
```

Using unions to allow multiple views of data is especially useful when working with registers, which are often divided into smaller units. x86 processors, for example, have 16-bit registers named AX, BX, CX, and DX. Each of these registers can be treated as two 8-bit registers. AX, for example, is divided into registers named AH and AL. (The H and L stand for "high" and "low.")

When writing low-level applications for x86-based computers, we may need variables that represent the contents of the AX, BX, CX, and DX registers. We want access to both the 16- and 8-bit registers; at the same time, we need to take their relationships into account (a change to AX affects both AH and AL; changing AH or AL modifies AX). The solution is to set up two structures, one containing members that correspond to the 16-bit registers, and the other containing members that match the 8-bit registers. We then create a union that encloses the two structures: