(You may be wondering how it's possible to store the seconds—a number between 0 and 59—in a field with only 5 bits. Well, DOS cheats: it divides the number of seconds by 2, so the `seconds` member is actually between 0 and 29.) If we're not interested in the `seconds` field, we can leave out its name:

```
struct file_time {
  unsigned int : 5;          /* not used */
  unsigned int minutes: 6;
  unsigned int hours: 5;
};
```

The remaining bit-fields will be aligned as if the `seconds` field were still present.

Another trick that we can use to control the storage of bit-fields is to specify 0 as the length of an unnamed bit-field:

```
struct s {
  unsigned int a: 4;
  unsigned int : 0;     /* 0-length bit-field */
  unsigned int b: 8;
};
```

A 0-length bit-field is a signal to the compiler to align the following bit-field at the beginning of a storage unit. If storage units are 8 bits long, the compiler will allocate 4 bits for the `a` member, skip 4 bits to the next storage unit, and then allocate 8 bits for `b`. If storage units are 16 bits long, the compiler will allocate 4 bits for `a`, skip 12 bits, and then allocate 8 bits for `b`.

20.3 Other Low-Level Techniques

Some of the language features that we've covered in previous chapters are used often in low-level programming. To wrap up this chapter, we'll take a look at several important examples: defining types that represent units of storage, using unions to bypass normal type-checking, and using pointers as addresses. We'll also cover the `volatile` type qualifier, which we avoided discussing in Section 18.3 because of its low-level nature.

Defining Machine-Dependent Types

Since the `char` type—by definition—occupies one byte, we'll sometimes treat characters as bytes, using them to store data that's not necessarily in character form. When we do so, it's a good idea to define a `BYTE` type:

```
typedef unsigned char BYTE;
```

Depending on the machine, we may want to define additional types. The x86 architecture makes extensive use of 16-bit words, so the following definition would be useful for that platform: