Binary Operators

Along with the pointer, another low-level aspect of C is the binary operator. Binary operators work to manipulate the bits inside any C language variable except for a `float` or `double`. Table 27-1 lists the lot of them.

Table 27-1 Bitwise Operators

Operator	*Function*
&	AND
^	Exclusive OR (EOR or XOR)
|	Inclusive OR
~	One's complement
<<	Shift bits left
>>	Shift bits right

The only place I have seen these types of operators used are when programs interact with the operating system or PC hardware. For example, if some port on the PC has its low bit set, you can use one or more of these operators to determine whether that's true.

A special feature of the >> and << operators is that they perform superfast binary division and multiplication. For example:

```
x = y >> 1;
```

This function divides the value of `y` by 2, which is what happens when you shift the bits in the value of `y` one step to the left. It's much faster than using this statement:

```
x = y/2;
```

To divide `y` by 4, you can use this function:

```
x = y >> 2;
```

Now, you have to *think* in binary to make this work, but it's possible. Likewise, you can double a value by using the << operator. But save that one for another day.