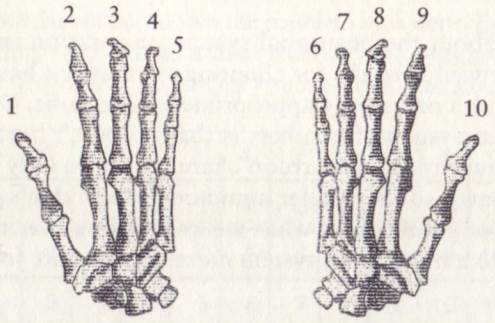
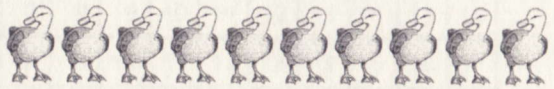
Chapter Eight

Alternatives to Ten

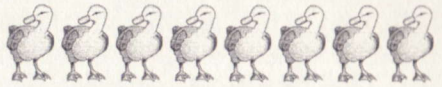
Ten is an exceptionally important number to us humans. Ten is the number of fingers and toes most of us have, and we certainly prefer to have all ten of each. Because our fingers are convenient for counting, we humans have adapted an entire number system that's based on the number 10.



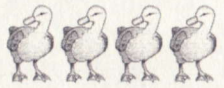
As I mentioned in the previous chapter, the number system that we use is called *base ten,* or *decimal.* The number system seems so natural to us that it's difficult at first to conceive of alternatives. Indeed, when we see the number *10* we can't help but think that this number refers to this many ducks:

10 = 

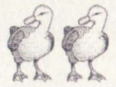
But the only reason that the number 10 refers to this many ducks is that this many ducks is the same as the number of fingers we have. If human beings had a different number of fingers, the way we counted would be different, and 10 would mean something else. That same number 10 could refer to this many ducks:

10 = 

or this many ducks:

10 = 

or even this many ducks:

10 = 

When we get to the point where 10 means just two ducks, we'll be ready to examine how switches, wires, lightbulbs, and relays (and by extension, computers) can represent numbers.

What if human beings had only four fingers on each hand, like cartoon characters? We probably never would have thought to develop a number system based on ten. Instead, we would have considered it normal and natural and sensible and inevitable and incontrovertible and undeniably proper to base our number system on eight. We wouldn't call this a *decimal* number system. We'd call it an *octal* number system, or *base eight.*

If our number system were organized around eight rather than ten, we wouldn't need the symbol that looks like this:

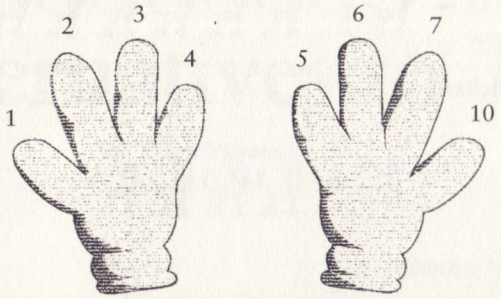
9

Show this symbol to any cartoon character and you'll get the response, "What's that? What's it for?" And if you think about it a moment, we also wouldn't need the symbol that looks like this:

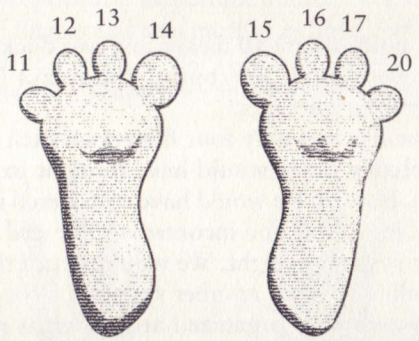
8

In the decimal number system, there's no special symbol for ten, so in the octal number system there's no special symbol for eight.

The way we count in the decimal number system is 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, and then 10. The way we count in the octal number system is 0, 1, 2, 3, 4, 5, 6, 7, and then what? We've run out of symbols. The only thing that makes sense is 10, and that's correct. In octal, the next number after 7 is 10. But this 10 doesn't mean the number of fingers that humans have. In octal, 10 refers to the number of fingers that cartoon characters have.



We can continue counting on our four-toed feet:



When you're working with number systems other than decimal, you can avoid some confusion if you pronounce a number like 10 as *one zero*. Similarly, 13 is pronounced *one three* and 20 is pronounced *two zero*. To *really* avoid confusion, we can say *two zero base eight* or *two zero octal*.

Even though we've run out of fingers and toes, we can still continue counting in octal. It's basically the same as counting in decimal except that we skip every number that has an 8 or a 9 in it. And of course, the actual numbers refer to different quantities:

0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21, 22,
23, 24, 25, 26, 27, 30, 31, 32, 33, 34, 35, 36, 37, 40, 41, 42, 43,
44, 45, 46, 47, 50, 51, 52, 53, 54, 55, 56, 57, 60, 61, 62, 63, 64,
65, 66, 67, 70, 71, 72, 73, 74, 75, 76, 77, 100...

That last number is pronounced *one zero zero*. It's the number of fingers that cartoon characters have, multiplied by itself.

When writing decimal and octal numbers, we can avoid confusion and denote which is which by using a subscript to indicate the numbering system. The subscript TEN means base ten or decimal, and EIGHT means base eight or octal.

Thus, the number of dwarfs that Snow White meets is 7_{TEN} or 7_{EIGHT}

The number of fingers that cartoon characters have is 8_{TEN} or 10_{EIGHT}

The number of symphonies that Beethoven wrote is 9_{TEN} or 11_{EIGHT}

The number of fingers that humans have is 10_{TEN} or 12_{EIGHT}

The number of months in a year is 12_{TEN} or 14_{EIGHT}

The number of days in a fortnight is 14_{TEN} or 16_{EIGHT}

The "sweet" birthday celebration is 16_{TEN} or 20_{EIGHT}

The number of hours in a day is 24_{TEN} or 30_{EIGHT}

The number of letters in the Latin alphabet is 26_{TEN} or 32_{EIGHT}

The number of fluid ounces in a quart is 32_{TEN} or 40_{EIGHT}

The number of cards in a deck is 52_{TEN} or 64_{EIGHT}

The number of squares on a chessboard is 64_{TEN} or 100_{EIGHT}

The most famous address on Sunset Strip is 77_{TEN} or 115_{EIGHT}

The number of yards in an American football field is 100_{TEN} or 144_{EIGHT}

The number of starting women singles players at Wimbledon is 128_{TEN} or 200_{EIGHT}

The number of square miles in Memphis is 256_{TEN} or 400_{EIGHT}

Notice that there are a few nice round octal numbers in this list, such as 100_{EIGHT} and 200_{EIGHT} and 400_{EIGHT}. By the term *nice round number* we usually mean a number that has some zeros at the end. Two zeros on the end of a decimal number means that the number is a multiple of 100_{TEN}, which is 10_{TEN} times 10_{TEN}. With octal numbers, two zeros on the end means that the number is a multiple of 100_{EIGHT}, which is 10_{EIGHT} times 10_{EIGHT} (or 8_{TEN} times 8_{TEN}, which is 64_{TEN}).

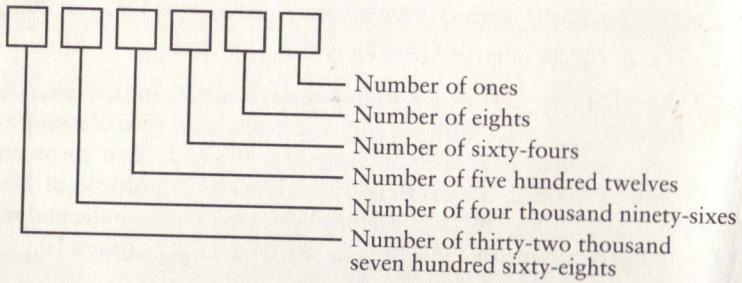
You might also notice that these nice round octal numbers 100_{EIGHT} and 200_{EIGHT} and 400_{EIGHT} have the decimal equivalents 64_{TEN}, 128_{TEN}, and 256_{TEN}, all of which are powers of two. This makes sense. The number 400_{EIGHT} (for example) is 4_{EIGHT} times 10_{EIGHT} times 10_{EIGHT}, all of which are powers of two. And anytime we multiply a power of two by a power of two, we get another power of two.

The following table shows some powers of two with the decimal and octal representations:

Power of Two	Decimal	Octal
2^0	1	1
2^1	2	2
2^2	4	4
2^3	8	10
2^4	16	20
2^5	32	40
2^6	64	100
2^7	128	200
2^8	256	400
2^9	512	1000
2^{10}	1024	2000
2^{11}	2048	4000
2^{12}	4096	10000

The nice round numbers in the rightmost column are a hint that number systems other than decimal might help in working with binary codes.

The octal system isn't different from the decimal system in any structural way. It just differs in details. For example, each position in an octal number is a digit that's multiplied by a power of eight:



Thus, an octal number such as 3725_{EIGHT} can be broken down like so:

$$3725_{EIGHT} = 3000_{EIGHT} + 700_{EIGHT} + 20_{EIGHT} + 5_{EIGHT}$$

This can be rewritten in any of several ways. Here's one way, using the powers of eight in their decimal forms:

$$\begin{aligned} 3725_{EIGHT} = {} & 3 \times 512_{TEN} + \\ & 7 \times 64_{TEN} + \\ & 2 \times 8_{TEN} + \\ & 5 \times 1 \end{aligned}$$

This is the same thing with the powers of eight shown in their octal form:

$$\begin{aligned} 3725_{EIGHT} = {} & 3 \times 1000_{EIGHT} + \\ & 7 \times 100_{EIGHT} + \\ & 2 \times 10_{EIGHT} + \\ & 5 \times 1 \end{aligned}$$

Here's another way of doing it:

$$\begin{aligned} 3725_{EIGHT} = {} & 3 \times 8^3 + \\ & 7 \times 8^2 + \\ & 2 \times 8^1 + \\ & 5 \times 8^0 \end{aligned}$$

If you work out this calculation in decimal, you'll get 2005_{TEN}. This is how you can convert octal numbers to decimal numbers.

We can add and multiply octal numbers the same way we add and multiply decimal numbers. The only real difference is that we use different tables for adding and multiplying the individual digits. Here's the addition table for octal numbers:

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

For example, $5_{EIGHT} + 7_{EIGHT} = 14_{EIGHT}$. So we can add two longer octal numbers the same way we add decimal numbers:

$$\begin{array}{r} 135 \\ +\ 643 \\ \hline 1000 \end{array}$$

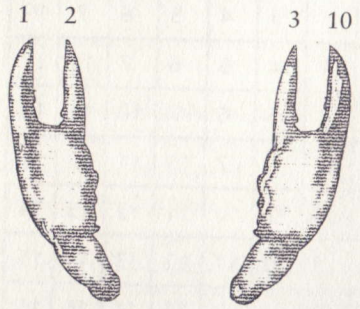
To begin with the right column, 5 plus 3 equals 10. Put down the 0, carry the 1. One plus 3 plus 4 equals 10. Put down the 0, carry the 1. One plus 1 plus 6 equals 10.

Similarly, 2 times 2 is still 4 in octal. But 3 times 3 isn't 9. How could it be? Instead 3 times 3 is 11_{EIGHT}, which is the same amount as 9_{TEN}. You can see the entire octal multiplication table at the top of the following page.

x	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

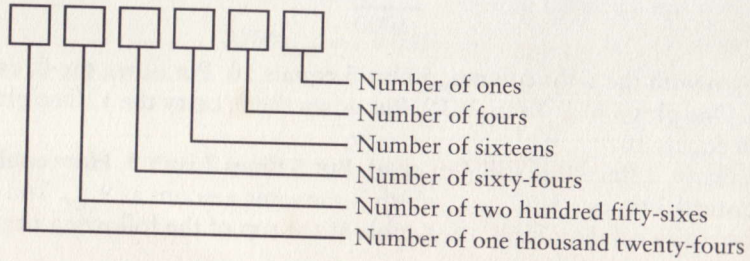
Here we have 4 × 6 equaling 30_{EIGHT}, but 30_{EIGHT} is the same as 24_{TEN}, which is what 4 × 6 equals in decimal.

Octal is as valid a number system as decimal. But let's go further. Now that we've developed a numbering system for cartoon characters, let's develop something that's appropriate for lobsters. Lobsters don't have fingers exactly, but they do have pincers at the ends of their two front legs. An appropriate number system for lobsters is the *quaternary* system, or base four:



Counting in quaternary goes like this: 0, 1, 2, 3, 10, 11, 12, 13, 20, 21, 22, 23, 30, 31, 32, 33, 100, 101, 102, 103, 110, and so forth.

I'm not going to spend much time with the quaternary system because we'll be moving on shortly to something much more important. But we can see how each position in a quaternary number corresponds this time to a power of *four*:



The quaternary number 31232 can be written like this:

$$31232_{FOUR} = 3 \times 256_{TEN} + 1 \times 64_{TEN} + 2 \times 16_{TEN} + 3 \times 4_{TEN} + 2 \times 1_{TEN}$$

which is the same as

$$31232_{FOUR} = 3 \times 10000_{FOUR} + 1 \times 1000_{FOUR} + 2 \times 100_{FOUR} + 3 \times 10_{FOUR} + 2 \times 1_{FOUR}$$

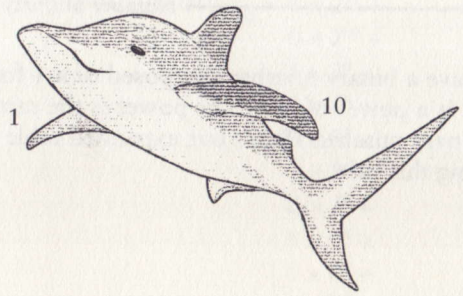
And it's also the same as

$$31232_{FOUR} = 3 \times 4^4 + 1 \times 4^3 + 2 \times 4^2 + 3 \times 4^1 + 2 \times 4^0$$

If we do the calculations in decimal, we'll find that 31232_{FOUR} equals 878_{TEN}.

Now we're going to make another leap, and this one is extreme. Suppose we were dolphins and must resort to using our two flippers for counting. This is the number system known as base two, or *binary* (from the Latin for *two by two*). It seems likely that we'd have only two digits, and these two digits would be 0 and 1.

Now, 0 and 1 isn't a whole lot to work with, and it takes some practice to get accustomed to binary numbers. The big problem is that you run out of digits very quickly. For example, here's how a dolphin counts using its flippers:



Yes, in binary the next number after 1 is 10. This is startling, but it shouldn't really be a surprise. No matter what number system we use, whenever we run out of single digits, the first two-digit number is always 10. In binary we count like this:

0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, 10000, 10001...

These numbers might look large, but they're really not. It's more accurate to say that binary numbers get *long* very quickly rather than large:

The number of heads that humans have is 1_{TEN} or 1_{TWO}

The number of flippers that dolphins have is 2_{TEN} or 10_{TWO}

The number of teaspoons in a tablespoon is 3_{TEN} or 11_{TWO}

The number of sides to a square is 4_{TEN} or 100_{TWO}

The number of fingers on one human hand is 5_{TEN} or 101_{TWO}

The number of legs on an insect is 6_{TEN} or 110_{TWO}

The number of days in a week is 7_{TEN} or 111_{TWO}

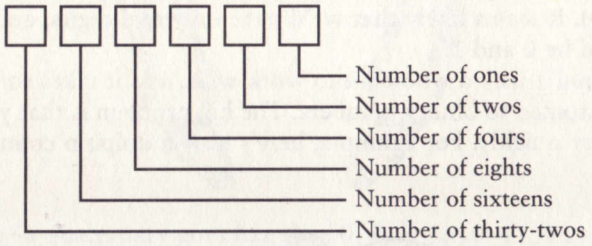
The number of musicians in an octet is 8_{TEN} or 1000_{TWO}

The number of planets in our solar system (including Pluto) is 9_{TEN} or 1001_{TWO}

The number of gallons in a cowboy hat is 10_{TEN} or 1010_{TWO}

and so forth.

In a multidigit binary number, the positions of the digits correspond to powers of two:



So anytime we have a binary number composed of a 1 followed by all zeros, that number is a power of two. The power is the same as the number of zeros in the binary number. Here's our expanded table of the powers of two demonstrating this rule:

Power of Two	Decimal	Octal	Quaternary	Binary
2^0	1	1	1	1
2^1	2	2	2	10
2^2	4	4	10	100
2^3	8	10	20	1000
2^4	16	20	100	10000
2^5	32	40	200	100000
2^6	64	100	1000	1000000
2^7	128	200	2000	10000000
2^8	256	400	10000	100000000
2^9	512	1000	20000	1000000000
2^{10}	1024	2000	100000	10000000000
2^{11}	2048	4000	200000	100000000000
2^{12}	4096	10000	1000000	1000000000000

Let's say we have the binary number 101101011010. This can be written as

$$\begin{aligned} 101101011010_{TWO} = {} & 1 \times 2048_{TEN} + \\ & 0 \times 1024_{TEN} + \\ & 1 \times 512_{TEN} + \\ & 1 \times 256_{TEN} + \\ & 0 \times 128_{TEN} + \\ & 1 \times 64_{TEN} + \\ & 0 \times 32_{TEN} + \\ & 1 \times 16_{TEN} + \\ & 1 \times 8_{TEN} + \\ & 0 \times 4_{TEN} + \\ & 1 \times 2_{TEN} + \\ & 0 \times 1_{TEN} \end{aligned}$$

The same number can be written this way:

$$\begin{aligned} 101101011010_{TWO} = {} & 1 \times 2^{11} + \\ & 0 \times 2^{10} + \\ & 1 \times 2^{9} + \\ & 1 \times 2^{8} + \\ & 0 \times 2^{7} + \\ & 1 \times 2^{6} + \\ & 0 \times 2^{5} + \\ & 1 \times 2^{4} + \\ & 1 \times 2^{3} + \\ & 0 \times 2^{2} + \\ & 1 \times 2^{1} + \\ & 0 \times 2^{0} \end{aligned}$$

If we just add up the parts in decimal, we get 2048 + 512 + 256 + 64 + 16 + 8 + 2, which is $2{,}906_{TEN}$.

To convert binary numbers to decimal more concisely, you might prefer a method that uses a template I've prepared:

x128	x64	x32	x16	x8	x4	x2	x1

__ + __ + __ + __ + __ + __ + __ + __ = ____

This template allows you to convert numbers up to eight binary digits in length, but it could easily be extended. To use it, put up to eight binary digits in the 8 boxes at the top, one digit to a box. Do the eight multiplications and put the products in the 8 lower boxes. Add these eight boxes for the final result. This example shows how to find the decimal equivalent of 10010110:

1	0	0	1	0	1	1	0
x128	x64	x32	x16	x8	x4	x2	x1

128 + 0 + 0 + 16 + 0 + 4 + 2 + 0 = 150

Converting from decimal to binary isn't quite as straightforward, but here's a template that let's you convert decimal numbers from 0 through 255 to binary:

÷128	÷64	÷32	÷16	÷8	÷4	÷2	÷1

The conversion is actually trickier than it appears, so follow the directions carefully. Put the entire decimal number (less than or equal to 255) in the box in the upper left corner. Divide that number (the dividend) by the first divisor (128), as indicated. Put the quotient in the box below (the box at the lower left corner), and the remainder in the box to the right (the second box on the top row). That first remainder is the dividend for the next calculation, which uses a divisor of 64. Continue in the same manner through the template.

Keep in mind that each quotient will be either 0 or 1. If the dividend is less than the divisor, the quotient is 0 and the remainder is simply the dividend. If the dividend is greater than or equal to the divisor, the quotient is 1 and the remainder is the dividend minus the divisor. Here's how it's done with 150:

150	22	22	22	6	6	2	0
÷128	÷64	÷32	÷16	÷8	÷4	÷2	÷1
1	0	0	1	0	1	1	0

If you need to add or multiply two binary numbers, it's probably easier to do the calculation in binary rather than convert to decimal. This is the part you're *really* going to like. Imagine how quickly you could have mastered addition if the only thing you had to memorize was this:

+	0	1
0	0	1
1	1	10

Let's use this table to add two binary numbers:

```
  1100101
+ 0110110
----------
 10011011
```

Starting at the right column: 1 plus 0 equals 1. Second column from right: 0 plus 1 equals 1. Third column: 1 plus 1 equals 0, carry the 1. Fourth column: 1 (carried) plus 0 plus 0 equals 1. Fifth column: 0 plus 1 equals 1. Sixth column: 1 plus 1 equals 0, carry the 1. Seventh column: 1 (carried) plus 1 plus 0 equals 10.

The multiplication table is even simpler than the addition table because it can be entirely derived by using two of the very basic rules of multiplication: Multiplying anything by 0 gets you 0, and multiplying any number by 1 has no effect on the number.

x	0	1
0	0	0
1	0	1

Here's a multiplication of 13_{TEN} by 11_{TEN} in binary:

```
    1101
  x 1011
----------
    1101
   1101
  0000
 1101
----------
10001111
```

The result is 143_{TEN}.

People who work with binary numbers often write them with leading zeros (that is, zeros to the left of the first 1)—for example, 0011 rather than just 11. This doesn't change the value of the number at all; it's just for cosmetic purposes. For example, here are the first sixteen binary numbers with their decimal equivalents:

Binary	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Let's take a look at this list of binary numbers for a moment. Consider each of the four vertical columns of zeros and ones, and notice how the digits alternate going down the column:

- The rightmost digit alternates between 0 and 1.
- The next digit from the right alternates between two 0s and two 1s.
- The next digit alternates between four 0s and four 1s.
- The next digit alternates between eight 0s and eight 1s.

This is *very* methodical, wouldn't you say? Indeed, you can easily write the next sixteen binary numbers by just repeating the first sixteen and putting a 1 in front:

Binary	Decimal
10000	16
10001	17
10010	18
10011	19
10100	20
10101	21
10110	22
10111	23
11000	24
11001	25
11010	26
11011	27
11100	28
11101	29
11110	30
11111	31

Here's another way of looking at it: When you count in binary, the rightmost digit (also called the least significant digit), alternates between 0 and 1. Every time it changes from a 1 to a 0, the digit second to right (that is, the next most significant digit) also changes, either from 0 to 1 or from 1 to 0. So every time a binary digit changes from a 1 to a 0, the next most significant digit also changes, either from a 0 to a 1 or from a 1 to a 0.

When we're writing large decimal numbers, we use commas every three places so that we can more easily know what the number means at a glance. For example, if you see 12000000, you probably have to count digits, but if you see 12,000,000, you know that means twelve million.

Binary numbers can get very long very quickly. For example, twelve million in binary is 101101110001101100000000. To make this a *little* more readable, it's customary to separate every four binary digits with a dash, for example 1011-0111-0001-1011-0000-0000 or with spaces: 1011 0111 0001 1011 0000 0000. Later on in this book, we'll look at a more concise way of expressing binary numbers.

By reducing our number system to just the binary digits 0 and 1, we've gone as far as we can go. We can't get any simpler. Moreover, the binary number system bridges the gap between arithmetic and electricity. In previous chapters, we've been looking at switches and wires and lightbulbs and relays, and any of these objects can represent the binary digits 0 and 1:

A wire can be a binary digit. If current is flowing through the wire, the binary digit is 1. If not, the binary digit is 0.

A switch can be a binary digit. If the switch is on, or closed, the binary digit is 1. If the switch is off, or open, the binary digit is 0.

A lightbulb can be a binary digit. If the lightbulb is lit, the binary digit is 1. If the lightbulb is not lit, the binary digit is 0.

A telegraph relay can be a binary digit. If the relay is closed, the binary digit is 1. If the relay is at rest, the binary digit is 0.

Binary numbers have a whole *lot* to do with computers.

Sometime around 1948, the American mathematician John Wilder Tukey (born 1915) realized that the words *binary digit* were likely to assume a much greater importance in the years ahead as computers became more prevalent. He decided to coin a new, shorter word to replace the unwieldy five syllables of *binary digit*. He considered *bigit* and *binit* but settled instead on the short, simple, elegant, and perfectly lovely word *bit*.

Chapter Nine

Bit by Bit by Bit

When Tony Orlando requested in a 1973 song that his beloved "Tie a Yellow Ribbon Round the Ole Oak Tree," he wasn't asking for elaborate explanations or extended discussion. He didn't want any ifs, ands, or buts. Despite the complex feelings and emotional histories that would have been at play in the real-life situation the song was based on, all the man really wanted was a simple yes or no. He wanted a yellow ribbon tied around the tree to mean "Yes, even though you messed up big time and you've been in prison for three years, I still want you back with me under my roof." And he wanted the absence of a yellow ribbon to mean "Don't even *think* about stopping here."

These are two clear-cut, mutually exclusive alternatives. Tony Orlando did *not* sing, "Tie half of a yellow ribbon if you want to think about it for a while" or "Tie a blue ribbon if you don't love me anymore but you'd still like to be friends." Instead, he made it very, very simple.

Equally effective as the absence or presence of a yellow ribbon (but perhaps more awkward to put into verse) would be a choice of traffic signs in the front yard: Perhaps "Merge" or "Wrong Way."

Or a sign hung on the door: "Closed" or "Open."

Or a flashlight in the window, turned on or off.

You can choose from lots of ways to say yes or no if that's all you need to say. You don't need a sentence to say yes or no; you don't need a word, and you don't even need a letter. All you need is a *bit*, and by that I mean all you need is a 0 or a 1.

As we discovered in the previous chapters, there's nothing really all that special about the decimal number system that we normally use for counting. It's pretty clear that we base our number system on ten because that's