C-4 Bit manipulation

Friday, February 22, 2013 6:38 PM

In computers, everything is eventually compressed down to a bunch of 1 and 0's known as binary language which the computer translates. Computers translate bytes which are made out of bits (there are 8 bits in a byte).

Binary is used to represent numbers by using a base power of 2. Every 0 or 1 indicates what the number can take form as in relation to the power of 2 (instead of the power of 10 which you are used to):

For example : 0101 = 5 because the rightmost 1 indicates a:
2^0x 1 = 1
2^1x 0 = 0
2^2x1 = 4
2^3x0 = 0
Now add up what you get to put your answer back in base 10 (5). Each 0 and 1 represents a bit so this is a 4 bit number. Note that I could expand this into a 16 bit number to allow for a bigger number such as:

	0	0	1	0	0	0	0	1	0	1	0	0	0	1
	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

If you put what each bit can indicate by starting at 1 at the rightmost bit and then doubling each bit to the left, you can treat the 1's and 0's as true or false values where you only add up the values indicated under the 1's to determine the binary number.(2048+64+16+1= 2129).

Numbers in C can also be thought as represented in binary and can have binary operations affect them. This is known as bit manipulation

The bit manipulations in C are as follows:

|-(or operation)-checks if that bit is true , the other bit is true , or both are true,
&-(and operation)-checks if both bits are true (each has 1) if not, puts down a 0
>> (shift to the right) shifts the leftmost 1 to the right depending on your indication.
<< (shift to the left) shifts the rightmost side left by the indicated number of 0's/
~ negates the entire number, turns all the 1 into 0's and 0's into 1's.