**Difference between & and &&**

in C (and other languages probably) a single | or & is a bitwise comparison.

The double || or && is a logical comparison.

http://stackoverflow.com/questions/7331686/why-and-not

logical operator (||,&&) vs bitwise operator (|,&)

The most crucial difference between logical operator and bitwise operator is that logical operator takes two booleans and produces a boolean while bitwise operator takes two integers and produces an integer (note: integers means any integral data type, not just int).

To be pedantic, bitwise operator takes a bit-pattern (e.g. 01101011) and does a bit-wise AND/OR on each bits. So, for example if you have two 8-bit integers:

a = 00110010 (in decimal: 32+16+2 = 50)

b = 01010011 (in decimal: 64+ 16+2+1 = 83)

----------------

a & b = 00010010 (in decimal: 16+2 = 18)

a | b = 01110011 (in decimal: 64+32+16+2+1 = 115)

while a logical operator only works in bool:

a = true

b = false

--------------

a && b = false

a || b = true

Second, it is often possible to use bitwise operator on bool since true and false is equivalent to 1 and 0 respectively, and it happens that if you translate true to 1 and false to 0, then do bitwise operation, then convert non-zero to true and zero to false; it happens that the result will be the same had you just used logical operator (check this for exercise).