Introduction

When you learn to program in a high-level language like C (although C is fairly low-level, as high-level languages go), the idea is to avoid worrying too much about the hardware. You want the ability to represent mathematical abstractions, such as sets, etc. and have high level language features like threads, higher-order functions, exceptions.

High-level languages, for the most part, try to make you as unaware of the hardware as possible. Clearly, this isn't entirely true, because efficiency is still a major consideration for some programming languages.

C, in particular, was created to make it easier to write operating systems. Rather than write UNIX in assembly, which is slow process (because assembly code is tedious to write), and not very portable (because assembly is specific to an ISA), the goal was to have a language that provided good control-flow, some abstractions (structures, function calls), and could be efficiently compiled and run quickly.

Writing operating systems requires the manipulation of data at addresses, and this requires manipulating individual bits or groups of bits.

That's where two sets of operators are useful: *bitwise* operators and *bitshift* operators.

You can find these operators in C, C++, and Java (and presumably C#, since it's basically Java). Bitwise operators allow you to read and manipulate bits in variables of certain types.

Even though such features are available in C, they aren't often taught in an introductory level programming course. That's because intro level courses prefer to emphasize abstraction. With many departments using Java, there's a trend to increase what's abstract, and not get into the representation.

For example, some languages have support for stacks, queues, hashtables, and so forth. These "canned" data structures are meant to provide you, the programmer, with objects that perform certain tasks, while relieving you of the tedium and detail of understanding how the data is represented.

Nevertheless, if you intend to do some work in systems programming, or other forms of low-level coding (operating systems, device drivers, socket programming, network programming), knowing how to access and manipulate bits is important.