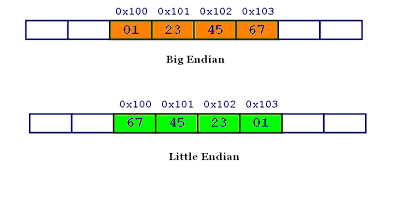
# Little and Big Endian Mystery

What are these?  
Little and big endian are two ways of storing multibyte data-types ( int, float, etc). In little endian machines, last byte of binary representation of the multibyte data-type is stored first. On the other hand, in big endian machines, first byte of binary representation of the multibyte data-type is stored first.  
  
Suppose integer is stored as 4 bytes (For those who are using DOS based compilers such as C++ 3.0 , integer is 2 bytes) then a variable x with value 0x01234567 will be stored as following.



Memory representation of integer ox01234567 inside Big and little endian machines

#include <stdio.h>

int main()

{

unsigned int i = 1;

char \*c = (char\*)&i;

if (\*c)

printf("Little endian");

else

printf("Big endian");

getchar();

return 0;

}

n the above program, a character pointer c is pointing to an integer i. Since size of character is 1 byte when the character pointer is de-referenced it will contain only first byte of integer. If machine is little endian then \*c will be 1 (because last byte is stored first) and if machine is big endian then \*c will be 0.

Does endianness matter for programmers?  
Most of the times compiler takes care of endianness, however, endianness becomes an issue in following cases.

It matters in network programming: Suppose you write integers to file on a little endian machine and you transfer this file to a big endian machine. Unless there is little endian to big endian transformation, big endian machine will read the file in reverse order. You can find such a practical example here.

Standard byte order for networks is big endian, also known as network byte order. Before transferring data on network, data is first converted to network byte order (big endian).

Sometimes it matters when you are using type casting, below program is an example.

What are bi-endians?  
Bi-endian processors can run in both modes little and big endian.

What are the examples of little, big endian and bi-endian machines ?  
Intel based processors are little endians. ARM processors were little endians. Current generation ARM processors are bi-endian.

Motorola 68K processors are big endians. PowerPC (by Motorola) and SPARK (by Sun) processors were big endian. Current version of these processors are bi-endians.  
  
Does endianness affects file formats?  
File formats which have 1 byte as a basic unit are independent of endianness e.g., ASCII files . Other file formats use some fixed endianness forrmat e.g, JPEG files are stored in big endian format.

Which one is better — little endian or big endian?  
The term little and big endian came from Gulliver’s Travels by Jonathan Swift. Two groups could not agree by which end an egg should be opened -a- the little or the big. Just like the egg issue, there is no technological reason to choose one byte ordering convention over the other, hence the arguments degenerate into bickering about sociopolitical issues. As long as one of the conventions is selected and adhered to consistently, the choice is arbitrary.