Computer Data

Babbo

Bits and Bytes

In this document, we are going to disect the C statement

```
int my_int = 5;
```

All computer data is composed of bytes, which themselves are composed of 8 bits each. A byte is the smallest piece of data which a computer manipulates: the individual bits are manipulated by manipulating the byte they belong to. A single bit is like an on/off switch: it can take only the values 0, which might mean the number 0, False, and No, and 1, which might mean the number 1, True, and Yes. It is up to programmers and programming languages to determine what these values represent: for example, a particular "on" bit might indicate that your system is out of usable memory and that an attempt to save a file has failed. A bit could also mean that a number is a negative number instead of a positive one. However, when the initialization of `my_int` to 5 is executed, your computer sets the 4 bytes, and thus 32 bites, which comprise an int in such a way that they represent the number 5. To do this, your computer treats each bit as a **binary** digit. Recall that in base 10, the nth digit from the right represents $d * 10^{n-1}$, where d is the digit's value (0 to 9). So for example, $139 = 9*10^0+3*10^1+1*10^2 = 9+30+100$. In binary, numbers are represented in powers of 2, using only the digits 0 and 1. A 32-bit representation of 5 is therefore 00000000000000000000000000000101.

That's a lot of zeroes! Note how only the first and third digits are 1, or "on". This is because $5 = 2^0 + 2^2 = 1 + 4$.

In base 10, the maximum integer we may represent with n digits is $10^n - 1$. For example, with 3 digits, we may represent the number $999 = 10^3 - 1 = 1000 - 1$. Likewise, since ints have 32 bits (again, binary digits), we may represent a number as large as $2^32 - 1 = 4,294,967,295$. But wait! That's only if we omit negative numbers. ints in C represent both positive AND negative numbers. One way to do this is by using the last bit to indicate the sign. For example, the following sequence of 8 bits (for readability) could represent -5 : 10000101. Notice that it is identical to the previous representation, but with the leading bit on. While this is a simple and intuitive way to include negative numbers, most computers use what is called "2's complement" representation, which I will not expand on here.