**CSE 351 Section 1 – Number Bases and Working in C [Solutions]**

Hi there and welcome to section! ☺

**Numerals**

|  |  |  |
| --- | --- | --- |
| **Binary** | **Decimal** | **Hex** |
| 0000 | 0 | 0 |
| 0001 | 1 | 1 |
| 0010 | 2 | 2 |
| 0011 | 3 | 3 |
| 0100 | 4 | 4 |
| 0101 | 5 | 5 |
| 0110 | 6 | 6 |
| 0111 | 7 | 7 |
| 1000 | 8 | 8 |
| 1001 | 9 | 9 |
| 1010 | 10 | A |
| 1011 | 11 | B |
| 1100 | 12 | C |
| 1101 | 13 | D |
| 1110 | 14 | E |
| 1111 | 15 | F |

A *numeral* is a symbolic representation of a number. For the purposes of this class, we will define a numeral as a sequence of digits (symbols).

**Number Bases**

If we have an -digit numeral in base , then the value of that numeral is , which is just fancy notation to say that instead of a 10’s or 100’s place we have a ’s or ’s place.

The most common bases we will use in this class are 2, 10, and 16, which are called binary, decimal, and hexadecimal (or hex), respectively. In base , each digit can only be one of fixed symbols (0-1 for binary, 0-9 for decimal, etc.).

The table on the right shows the equivalent numerals for the numbers 0 through 15 in these three major number bases. We differentiate between these bases by using the prefix ‘0b’ for binary and ‘0x’ for hexadecimal.

**Exercises:**

1. Complete the table below by converting the numbers into the other two common bases. You may leave the “Decimal” column unsimplified.

|  |  |  |
| --- | --- | --- |
| **Binary** | **Decimal** | **Hexadecimal** |
| 0b10010011 | 27 + 24 + 21 + 20 = 147 | 0x93 |
| 0b10110 | 1161 + 6160 = 22 | 0x16 |
| 0b111111 | 63 | 0x3F |
| 0b100100 | 25 + 22 = 36 | 0x24 |
| 0b110000110000 | 12162 + 3161 = 3120 | 0xC30 |
| 0b0 | 0 | 0x0 |
| 0b101110101101 | 11162 + 10161 + 13160 = 2989 | 0xBAD |
| 0b110110101 | 437 | 0x1B5 |

**Setting Up Your System**

You have four options for your working environment:

1. CSE Labs: Log in locally to one of the *Linux* machines in CSE 002, 003, or 006 (must have a CSE account)
2. Remote access: Log in remotely to attu.cs.washington.edu (CSE account)
3. Install the CSE VM: <https://www.cs.washington.edu/lab/software/linuxhomevm>
4. Personal computer: Must be running a Linux distribution (e.g. Ubuntu, Fedora, CentOS)

You will need the following tools for the rest of the course, so make sure you know how to access/use them (already installed on attu and the VM) and start to get familiar with them:

* Text Editor (personal preference)
  + Try many, pick one! Some tutorials can be found on the course website.
  + Command-line: nano, vim, emacs
  + Graphical: gedit, emacs
* GNU Compiler Collection (gcc)
  + Example: gcc -Wall -g -std=c99 -o execName sourceCode.c
    - -W sets warnings
    - -g turns on debugging symbols
    - -std sets what version of C we are using
    - -o sets the name of the resulting executable
* GNU Project Debugger (gdb)
  + Command-line debugger that we will use heavily later in the course

**Code Examples:**

1. Download HelloWorld.c from the class webpage:

$ wget https://courses.cs.washington.edu/courses/cse351/19su/sections/01/code/HelloWorld.c

1. Open the file in your favorite text editor and read the comments
2. Compile the file to the executable hello: $ gcc -o hello HelloWorld.c
3. Run the program: $ ./hello
4. Download calculator.c from the class webpage:

$ wget https://courses.cs.washington.edu/courses/cse351/19su/sections/01/code/calculator.c

1. Read through the code in a text editor, then compile and run the program
2. Example usage: $ ./calculator 4 5 +

**printf**

Used to print to the console. Unfortunately, you can’t concatenate String variables like you can in Java.

You provide a format string as the first argument, which includes placeholders to print out variables:

* %d for signed int, %u for unsigned int, %f for float, %s for “string”, %x for hexadecimal, %p for pointer
* Examples:
  + printf("I am %d years old", 20) prints “I am 20 years old”
  + printf("My name is %s", "Alfian") prints “My name is Alfian”
  + printf("%d in hex is %x", 2827, 2827) prints “2827 in hex is 0xb0b”