



# CMPT 295

Unit - Data Representation

Lab 2 – Integral Representation and Impact of Fixed Size  
Memory in C

# Lab 2 - Objectives

- This lab relates to the demo we have seen in Lecture 3
- In this lab, we shall ...
  1. Set up our *working space* on our ***target machine***
  2. Get introduced to some Linux commands
  3. Investigate how the same bit pattern (series of 0's and 1') can be interpreted either as a signed value or an unsigned value in C
  4. Investigate the impact of fixed size memory when converting from one integral data type to another in C considering size, consider sign then considering both (size and sign)

# Lab 2 - Instructions

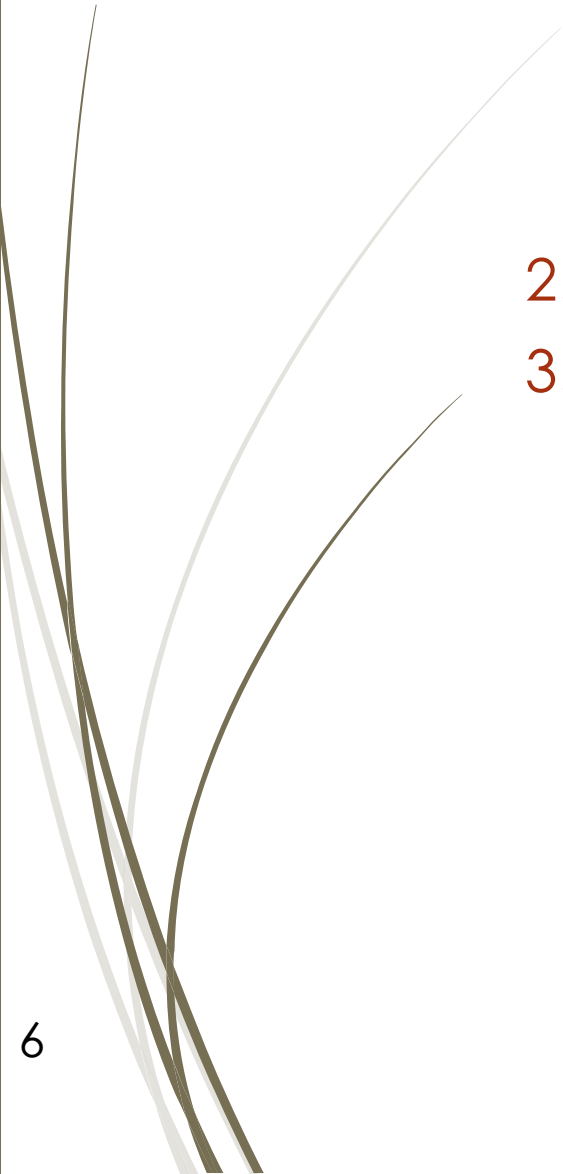
- ▶ Read and complete each slide, answering the questions
- ▶ Remember:
  - ▶ There is nothing to hand in in this lab
  - ▶ This lab is worth 0 marks
- ▶ If you have questions or are wondering about this lab's solution?
  - ▶ Attend our lab periods on Monday (starting May 26)
  - ▶ Come to the instructor's office hours (starting Week 2) and/or the TAs' office hours (starting Week 3)

# C Resources

- In this course, you need to be able to write basic C programs.
- For a review of C fundamentals, try the following introductory C tutorials:
  - [Essential C](#) - A concise introduction to C by Nick Parlante of Stanford
  - [C Tutorial For Beginners](#) with examples - from BeginnersBook
  - [C Tutorials from tutorialspoint](#)
  - [C Tutorials from Cornell](#)
  - [C Programming Boot Camp](#) by Paul Gribble at Western
- Many more C tutorials can be found online. Feel free to peruse.
- Also, have a look at the Resources web page on our course web site.

# 1. Set up our *working space* on our ***target machine***

1. If you are in CSIL, then login to one of the CSIL Linux workstations. If you are not in CSIL, then remotely login to one of the CSIL Linux CPU servers (or workstations) by following the instructions found at [Remote Login to our \*\*\*target machine\*\*\* in CSIL](#)
  - **VM:** You can also work locally using a Linux Ubuntu VM on your computer. If you do so, you will need to adjust some of the instructions below. For example, you will not have the **sfuhome** directory on your local drive.
    - If you work locally using a Linux Ubuntu VM, I **strongly** suggest that, once you have compiled and tested your code on your VM, you compile it on the ***target machine*** (CSIL machine) since this is the environment (O.S. and compiler versions) we will be using in this course (and used by TAs when marking).



1. Set up our *working space* on our ***target machine*** (cont'd)

2. Open **Lab 2** (posted on our course web site) in a browser tab.
3. Start a **Terminal** window.
  - At this point, your current directory path should be **/home/userid** where **userid** is your *SFU username*.
  - To check, use the Linux command **pwd**.

# 1. Set up our working space on our **target machine** (cont'd)

4. Make **sfuhome** your working directory by using the command **cd sfuhome**.
  - This **cd** command is the **c**hange **d**irectory command.
  - Note: It is important that you complete all of your coursework within the **sfuhome** directory so that your work is saved on the network server instead of being on the hard disk of one of the local workstations. This way, your work can be accessed from any SFU and CSIL computer. Also, your work is backed up.
5. Within your **sfuhome** directory, create a **CMPT295** directory.
  - To create new directories (or folders), use the command **mkdir**.
  - Alternatively, you can use the **Files** application to navigate through your **sfuhome** directory and create the **CMPT295** directory.

## 2. Introducing Linux commands

- As you can see from having done Part 1 of this lab, in this course, you need to be familiar with some common Linux commands.
- For a summary of basic Linux commands:
  - [Beginners guide to using the Linux terminal](#)
  - [Cheatsheet of Linux commands](#)
  - [Linux commands we must know](#)
- Many more Linux command tutorials can be found online. Feel free to peruse.



### 3. Investigate how bits can be interpreted as integral values in C

1. Download **Lab2.zip** from our course web site, save and unpack it (using the command **unzip**) in your **CMPT295** directory.
  - **Tip:** If this command failed, verify that you have saved the file in the correct directory. You may either use the **Files** application or use the Linux command **mv** to move it to the correct location.
  - Alternatively, you can unpack compressed files using the **Extract Here** option of the **Files** application.



3. Investigate how bits can be interpreted as integral values in C

2. There should now be a subdirectory named **Lab2** inside your **CMPT295** directory.
- Change directory using **cd Lab2**.
  - Alternatively, you can use the **Files** application to navigate to the **Lab2** directory.

### 3. Investigate how bits can be interpreted as integral values in C (cont'd)

3. Open **Lab2\_Part3.c**, read its instructions and complete it such that your C program produces the results below.
  - **Hint:** In order to do so, make use of the functions you find in **Lab2\_Part3.c** and make use of the **C format specifiers**.
4. Once you have completed and executed your **Lab2\_Part3.c**, it must print the following results on the monitor screen:

Results:

```
1
2 Binary representation of -2004318072 is '0b10001000100010001000100010001000'
3
4 Hexadecimal representation of -2004318072 is '0x88888888'
5
6 Signed integral representation of -2004318072 is '-2004318072'
7
8 Unsigned integral representation of -2004318072 is '2290649224'
9
```

## 4. Investigate the impact of fixed size memory in C

1. Open **Lab2\_Part4.c** and hand trace the program. Make sure you understand what the program does, especially its main function. Without compiling and executing the program, can you guess which words (“Cafe”, “Decaf” and/or “Latte”) will the program print?
2. Now, confirm your guess by compiling and executing the program.
3. Modify the conditions of the three conditional statements of the main function such that only “Cafe” is printed when the program executes.
  - **Careful:** When modifying the code, you can only use data type casting and you can only add three casts, one in each of the conditions. No other modifications are permitted!