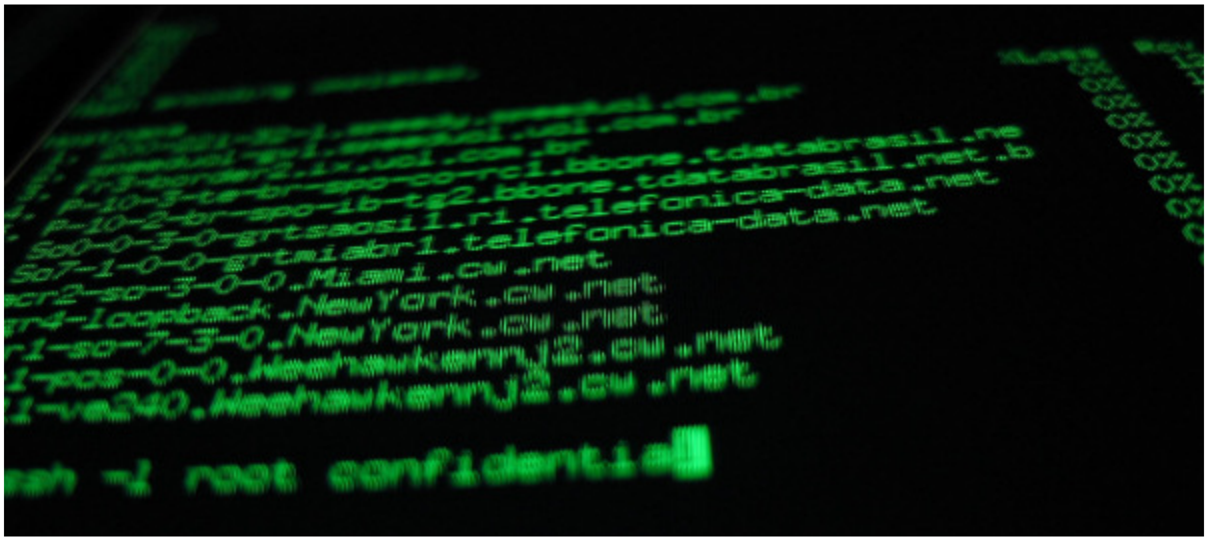
Lab 1. Introduction to Linux and C++.

Making sure that all have Gradescope invite (and resend to the ones that do not using their EMPLID and email that they actually use), and have everyone create their Gradescope account.

Review of some Unix terminal commands from CSCI127.



You need to know just a few commands to work comfortably in a Unix terminal: **`ls`** , **`cd`** , **`pwd`** , **`mkdir`** , **`cp`** , **`mv`** , **`rm`** .

A brief summary:

`pwd`	print the current working directory
`ls`	list files in the current directory
`ls path/to/a/directory`	list files in the directory
`cd path/to/a/directory`	change directory

`cd ..`	go to the parent directory (one level up)
`mkdir newdirectoryname`	create new directory
`cp file1 file2`	copy `file1` and call the copy `file2`
`mv file1 file2`	rename (move) `file1` to `file2`
`rmdir directoryname`	remove empty directory
`rm file`	remove `file`

These are some useful directory shortcuts:

`.`	the current directory
`..`	the parent directory of the current
`~`	the home directory

And these additional unix commands:

`touch newfilename`	create new file
`chmod <options> file`	change file permissions (read `+r` , write `+w` , execute `+x`)
`man command`	documentation about the `command`

Also read this article and watch its videos:

Become a Command Line Ninja With These Time-Saving Shortcuts

Try this Unix tutorial for more information.

Compiling C++ code with `g++` .

First, get acquainted with the `gedit` Text Editor, which you will be using to write your C++ programs.

Now, lets set the default tab width on your Text Editor to `3` . Once you open it, click on `Text Editor` in the upper left corner of the screen and select `Preferences` from the drop down menu. Click the `Editor` tab and set `Tab width` to `3` . Done!

Regarding coding style: remember that it is of paramount importance that all of your code is correctly indented. You must always use tabs to indent lines of code. Never use spaces to indent! In addition to automatic grading by Gradescope, we will spot check your code manually and will reduce your grade, if the indentation is not correct or if spaces are used instead of tabs.

Consider the following program that asks for your current age and prints it on the screen:

```
#include <iostream>
using namespace std;

int main() {
	cout << "Enter your age: " << endl;
	int age = 0;
	cin >> age;

	cout << "Your age is " << age << "." << endl;
}
```

To compile program `code.cpp` and call the executable `myprog`, type in the terminal:

```
g++ -o myprog code.cpp
```

To run the produced compiled program:

```
./myprog
```

Alternatively, if you compile the program without giving the output file name (such as `-o myprog`), the executable file will be called `a.out`, which you can execute the same way:

```
./a.out
```

How to compile and run programs on your own computer?

There are three components you should make sure you have installed:

- `g++` compiler,
- a functioning terminal application, and
- a good plain text editor (Gedit, Emacs, Vim, or Sublime Text) to write programs.

If your computer OS is Linux or Mac OS, you already have everything! If it is Windows, please follow this excellent tutorial: Linux on Windows Tutorial . OSX. You already have everything. It may come with g++ pre-installed. If you have it setup, this is all you need for most of the course.

How to submit your programs.

Each program should be submitted through Gradescope.

Write separate programs for each part of the assignment.
Submit only the source code `.cpp` files, not the compiled executables.
Each program should start with a comment that contains your name and a short program description, for example:

```
/*
Author: your name
Course: CSCI-136
Instructor: their name
Assignment: title, e.g., Lab1A

Here, briefly, at least in one or a few sentences
describe what the program does.
*/
```

Task A. Find the smaller of two integers.

Write a program `smaller.cpp` that asks the user to input **two integer numbers** and prints out the smaller of the two.

Example

```
$ ./smaller
Enter the first number: 15
Enter the second number: -24

The smaller of the two is -24
```

Task B. Find the smaller of three integers.

Write a program `smaller3.cpp` that asks the user to input **three integer numbers**, and prints out the smallest of the three.

(Hint: There are many possible solutions here. One possible strategy: Given numbers `x`, `y`, and `z`, you can first compare `x` and `y`, take whichever is smaller and compare it with `z`.)

Example

```
$ ./smaller3
Enter the first number: 23
Enter the second number: 76
Enter the third number: 37

The smaller of the three is 23
```

Task C. A leap year or a common year?

Introducing the modulo operator `%`

In C++, operator `%` computes the remainder of the division of `x` by `y`. For example, `37 % 10` returns `7`, because this is the remainder of `37` when divided by `10`.

The task

Write a program `leap.cpp` that asks the user to input an integer representing a year number (1999, 2016, etc.). If the input year is a *leap year* according to the modern Gregorian calendar, it should print `Leap year`, otherwise, print `Common year`.

In the **modern Gregorian calendar**, a year is a *leap year* if it is divisible by 4, but century years are not leap years unless they are divisible by 400. Here is the pseudocode:

```
if (year is not divisible by 4) then (it is a common year)
else if (year is not divisible by 100) then (it is a leap year)
else if (year is not divisible by 400) then (it is a common year)
else (it is a leap year)
```

This means that 2012, 2016, 2020, and 2040 are all leap years.
However, the century years 1800, 1900, 2100, 2200, 2300 and 2500 are NOT.
Yet, 2000, 2400, 2800 are still leap years.

Example 1

```
$ ./leap
Enter year: 2016
```

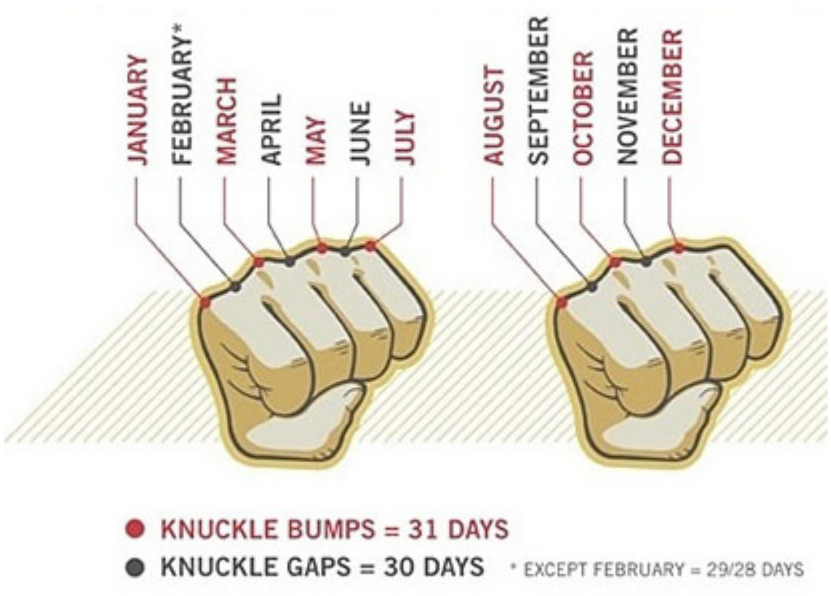
```
Leap year
```

Example 2

```
$ ./leap
Enter year: 2017

Common year
```

Task D. Number of days in a given month



Write a program `month.cpp` that asks the user to input the year and the month (1-12) and prints the number of days in that month (taking into account leap years). You may not use `switch case` or arrays even if you know these language constructs.

Example

```
$ ./month
Enter year: 2017
Enter month: 5
```

31 days