Who am I?

- **My path here**
  - Born in Ames, IA → grew up in Chazy, NY → college at Univ. of New Hampshire → graduate school at Univ. of Chicago → taught math in Cape Town, SA → researcher at Univ. of Washington

- **What I do specifically with ATLAS data**
  - Working on searches for dark matter and exotic physics
  - Work on the calibration of things called “jets”

- **Am I a programming guru?**
  - One semester of c (not c++) in college
  - Caveat: my pedagogy comes from experience

- **I enjoy**
  - Teaching
  - Skiing/snowboarding
  - Hiking/climbing/running
  - Travelling
Hopes and Plans

● What are my sessions about? - “Programming” in c++
  ○ 1 - Going from knowing “cd, cp ...” to running a program
  ○ 2 - Storage and operations - “doing stuff”
  ○ 3 - Version Control (super saving)
  ○ 4 - Flow control - ifs and Loops
  ○ 5 - Efficient Storage Factorizing thinking - Functions
  ○ 6 - Designing useful black boxes - Classes
  ○ 7 - Advanced Compilation - makefiles
  ○ 8 - Python - the other big one
  ○ 9 - The other 90% of your time - Debugging

● What do I want at the end of my sessions?
  ○ I want to be non-essential ➔ You should be self-sufficient
  ○ Why?
    ■ We will only cover a fraction of programming
    ■ You must learn to self-educate to be a successful researcher
Introduction

● Abstract introduction about programming and paradigms
  ○ Colorful shapes that will eventually mean something
● Compiling - Building a program for the first time
● Learning to learn programming
  ○ (not from me!)
● Program layout
  ○ Beginning to read
● The language of programming
What is “programming”?

Three ways to answer this question

● **[Abstract]** A way to solve questions based on creating a well defined procedure to arrive at a numerical answer
  ○ Mathematician: How many prime numbers exist between 0 and 100?

● **[Abstract]** A way to take a set of data, perform operations and manipulations on that data, and produce another set of data
  ○ Asking a bunch of people who they want to with the election → determining on a state-by-state basis who will win each state → presenting that to an audience

● **[Concrete]** Creating a human readable (text) file, converting this to a machine readable set of processes, and executing this set of processes with some number of inputs
  ○ This is done in front of the computer and is what you will do most of your time
Programming: Abstract Concept

- LHC → We most often adhere to the second definition of programming.

“[0]”

[1] Event 1
Event 2
Event 3
... Event n

[2] Programming (note that this box is markedly not black)

[3]
**Programming Paradigms**

- **Functional**: Same input gives the same output
  - Pass data through “filters” that are independent of the data itself
  - Build up more complex manipulations via composition of simple operations \( f( g(x) ) \)
Programming Paradigms

- **Object-Oriented**: Same input *might give* different output
  - Store the data and methods in a single “structure” (objects) and manipulate these structures
  - Build more complex manipulations by linking objects together

**Simple Object**

**Object Oriented Program**
Compilation

- Basic concept: turn human readable text into a computer executable software
- What will you be using compilation for to begin with?
  - [1] Producing something you can run
  - [2] Getting hints at errors you have made while writing your program - “debugging”
Compilation

- In reality: there are two steps
  - Compiling: translation of human readable text (source code) into machine language
    - These look like nonsense and are not executable
  - Linking: bringing together many object files into a single executable
    - This produces the “a.out” executable - looks like nonsense but now its executable

Create object file from source code:
$ g++ -c hellocern.cxx
Compilation

• In reality: there are two steps
  ○ Compiling: translation of human readable text (source code) into machine language
    ■ These look like nonsense and are not executable
  ○ Linking: bringing together many object files into a single executable
    ■ This produces the “a.out” executable - looks like nonsense but now it's executable

• More correct term: “Build” = Compiling+Linking

Link object file(s) into executable:
$ g++ -o hellocern.exe hellocern.o

Run the executable command:
$ ./hellocern.exe
Exercise: Compiling Something

Get this file from lxplus and then compile and run it -
/afs/cern.ch/work/m/meehan/public/CSU/hellocern.cxx

● Reminder of some key steps (you don’t need to type the “$”)
  ○ Getting file from a remote computer
    ■ $ scp USERNAME@lxplus.cern.ch:/path/to/file/file.txt local_file.txt
  ○ Making a full build of a file and giving it a desired output name with the -o option
    ■ $ g++ -o <EXECUTABLENAME> <FILETOCOMPILE>
  ○ Running an executable
    ■ $ ./<EXECUTABLE>

● If you finish, try to edit the file and recompile!
Program Layout

- Breaking down our simple program from last time: build up from here
Program Layout

- Breaking down our simple program from last time: build up from here

This is a comment.
They can (and should) go throughout the program.
Program Layout

- Breaking down our simple program from last time: build up from here

Bring in other functionality via the inclusion of headers and libraries (Homework 1 …)
Program Layout

● Breaking down our simple program from last time: build up from here

This says that your main function has access to the "std" (standard) namespace.
Program Layout

- Breaking down our simple program from last time: build up from here

Return value type. Must be an "int" for main()
Program Layout

• Breaking down our simple program from last time: build up from here

How to print stuff to the screen - more options to be learned on your own
Program Layout

- Breaking down our simple program from last time: build up from here

Whitespace - it actually means nothing in C++. Just used for structure/readability.
Program Layout

- Breaking down our simple program from last time: build up from here.

Return value for the function when it terminates here → can be accessed in terminal afterwards via "?"
Running this program

- Compiling and executing dumps the statement to the terminal as expected
- Able to access the return value of our program after running via “?” in terminal
  - Very useful when program becomes long and can have many endpoints

**QUESTION**: What is this thing called in the shell?
Factorizing: Headers

- If we kept developing everything in one file (session2.cxx) it would become long
  - My first analysis was like this and turned into ~10000 lines of code
- Factorize initially via use of headers - Why? (Description here)
  - Organization and interface vs. implementation: staying organized is essential
  - Re-compilation time: not a big issue now
- Headers are like “ingredient list” or “tool box”
Take away message: communication
Writing programs requires precise language
- Learn a new way to speak/think
I *hope* that you will be comfortable with all/most of these concepts
External Control

- We can get some information back from our program using return codes
  - We will see later how to output more verbose information (“data”)
- How do we put information *into* our program?
  - Arguments to the main function

```cpp
//this is my first c++ program.
#include <iostream>

using namespace std;

int main()
{
    cout << "Starting Program" << endl;
    return 34;
}
```
External Control

- `int argc`: How many arguments are supplied to your executable program?
- `char *argv[]`: An ordered list of “words” for each argument separated by a space
  - Disentangling the syntax (i.e. the various symbols in this term) mean will come in time

Executable program like before (looks like) We have four arguments to this program, separated by spaces

```
{Sam, Meehan, CERN, 2009}
```
External Control

- int argc: How many arguments are supplied to your executable program?
- char *argv[]: An ordered list of “words” for each argument separated by a space
  - Disentangling the syntax (i.e. the various symbols in this term) mean will come in time

We need to provide some input arguments if we want it to “see” those
External Control

- `int argc`: How many arguments are supplied to your executable program?
- `char **argv[]`: An ordered list of “words” for each argument separated by a space
  - Disentangling the syntax (i.e. the various symbols in this term) mean will come in time

We have now provided arguments, but nothing seems to happen...

... we didn’t actually use the input arguments anywhere in the program itself [Doh!]
External Control

- `int argc`: How many arguments are supplied to your executable program?
  - Can be used just like its a variable that we had declared within our program (more next session!)

It seems to be telling us that there are 3 arguments ... when we only gave it “first” and “argument”
External Control

- int argc: How many arguments are supplied to your executable program?
  - Can be used just like it's a variable that we had declared within our program (more next session!)
- The program name is always argument #0

```cpp
#include <iostream>

using namespace std;

int main(int argc, char *argv[]){
    cout << "Starting Program" << endl;
    cout << "Number of Args : " << argc << endl;
    return 34;
}
```
External Control

- `char *argv[]`: An ordered list of “words” for each argument separated by a space
  - Access this like a list of strings (more next session!)
  - Counting of arguments begins at 0
External Control

- `char *argv[]`: An ordered list of “words” for each argument separated by a space
  - Access this like a list of strings (more next session!)
  - Counting of arguments begins at 0
- These arguments can be stored for use further on in your program
  - But they are, by default, strings …
  - What if I want to use a number as input (“casting” - more next session)
This seems clunky/finicky

- Yes, it is!
  - The way your program runs depends intimately on the order of the arguments
  - There is no “default” behavior for your program (“I don’t want to enter arguments”)
- Example of a program that analyzes the average energy of electrons above some noise threshold (20 GeV)

<table>
<thead>
<tr>
<th>Executable program</th>
<th>Number of events to analyze overall</th>
<th>Location of input file with data</th>
<th>Location where to write the output data</th>
</tr>
</thead>
<tbody>
<tr>
<td>./elecAnalyze</td>
<td>eScan 500</td>
<td>data2017.root</td>
<td>myAnalysis.root</td>
</tr>
</tbody>
</table>

prompt$> ./elecAnalyze eScan 500 20 data2017.root myAnalysis.root
This seems clunky/finicky

- Yes, it is!
  - The way your program runs depends intimately on the order of the arguments
  - There is no “default” behavior for your program (“I don’t want to enter arguments”)
- Example of a program that analyzes the average energy of electrons above some noise threshold (20 GeV)
- Will these two program executions give the same results? - NO!

<table>
<thead>
<tr>
<th>Command 1</th>
<th>Command 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>prompt$&gt; ./elecAnalyze eScan 500 20 data2017.root myAnalysis.root</code></td>
<td><code>prompt$&gt; ./elecAnalyze eScan 20 500 myAnalysis.root data2017.root</code></td>
</tr>
</tbody>
</table>

You are now only looking at 20 events and the energy threshold is 25 times higher (20 GeV → 500 GeV)
This seems clunky/finicky

- Yes, it is!
  - The way your program runs depends intimately on the order of the arguments
  - There is no “default” behavior for your program (“I don’t want to enter arguments”)

- Example of a program that analyzes the average energy of electrons above some noise threshold (20 GeV)

- Will these two program executions give the same results? - NO!

```
prompt$> ./elecAnalyze eScan 500 20 data2017.root myAnalysis.root

prompt$> ./elecAnalyze eScan 20 500 myAnalysis.root data2017.root

OH NO!!!! You are overwriting the input data file that you collected from the LHC!!!! (Hopefully you have it backed up.)
```
This seems clunky/finicky

- Yes, it is!
  - The way your program runs depends intimately on the order of the arguments
  - There is no “default” behavior for your program (“I don’t want to enter arguments”)
- Example of a program that analyzes the average energy of electrons above some noise threshold (20 GeV)
- Will these two program executions give the same results? - NO!

```
prompt$> ./elecAnalyze eScan 500 20 data2017.root myAnalysis.root
```

```
prompt$> ./elecAnalyze eScan 20 500 myAnalysis.root data2017.root
```

Your program probably will crash in the very beginning when it tries to get the input data because its not in the correct format in your output. Or non-existent even!
This seems clunky/finicky

- Luckily there are alternatives to hardcoded command line arguments
  - getopt(): Native c++ command line parser
    - Native = This code comes for “free” with g++
  - jamolnngg/argparse: An external package but a bit more user friendly
    - ... later on we will learn what GitHub is (that’s where this code is stored)
    - ... later on we will learn how to integrate external packages like this into your code
- Arguments get “tagged” and read in according to that tag
  - And arguments can be made “optional”
  - or given default values so that you can shorten the command line execution call

```
prompt$> ./elecAnalyze --type eScan --nev 500 --min 20 --in data2017.root --out myAnalysis.root

prompt$> ./elecAnalyze --in data2017.root --out myAnalysis.root --type eScan --nev 500 --min 20

prompt$> ./elecAnalyze --nev 500 --min 20 --in data2017.root --out myAnalysis.root
```

(Make the default behavior of elecAnalyze to be the average energy program)
Two Most Important Parts

- [1] Compile/Run your program
  - Write your program little by little and run it between each step
  - Rarely does a program run when you first try writing it
Two Most Important Parts

1. [1] Print things out!!
   - The only way you know what the computer is doing is to get information from it
   - This is done by *printing things to the terminal*
Helping Yourself ("the hoo ra ra")

- Reminder: I am not a programming guru / I’m a physicist
  - Requires me to “parameterize my ignorance” to make scientific progress
- To go from 0 to 60 you must take the lead
  - student → self-educator → researcher
- Copying code? : “I approximately don’t care”
  - Use your academic judgement about using the work of others
  - Anything meaningful in research will not be found on the internet
- You will get out of this what you put into it
  - The more preparation you do NOW
  - ... the more enjoyable and fruitful your summer
Helping Yourself

- Many wonderful resources exist for self-education in programming
  - **Google**: “how to (what is) <FILL IN THE BLANK> in c++”
    - <FILL IN THE BLANK> requires the correct language
    - The first hit will commonly be cplusplus.com, StackOverflow
  - **StackOverflow**: Forum online to answer questions
    - If you think you are the first one to see an error, you probably aren’t
  - **cplusplus.com**: The OFFICIAL reference
    - Hugely helpful number of examples
    - Much of what I will say comes from here
  - **YouTube**: Nice people make amazing video lectures on this stuff
    - E.g. I learned the concept of “inheritance” from [saurabhschool](http://saurabhschool) just one year ago
Exercise

Modify hellocern.cxx to define an integer variable to store the value five and then print this out just after “Hello CERN”

- Who doesn’t know how to do this?
  - Be proud of your ignorance! :)
- What are you going to Google to find out?
  - Part one: define an integer
  - Part two: print to the screen
Final Announcements

- Lectures/slides are posted for all future lectures
  - Feel free to go ahead and read them if you like

- Assignments
  - Five (5) in total for my section of the course
  - They will be due every Tuesday before the lecture starts
    - I will send out an email where you will reply (*NOT* Reply All) to send your assignment
    - I will check the timestamp on your email submission
  - DO NOT PLAGIARIZE!
    - This will only hurt your experience this summer
    - If you plagiarize from your peers, I will take it very seriously in terms of your mark for my section of the course
Harinder++

- Harinder covered many of the essential Linux basics in lectures 1 and 2
- A few additional commands/utilities will help with my sessions and further
  - Changing terminal configurations to make it “easier to work with” (in my opinion)

Open your terminal’s preferences

Set shortcuts for “New Tabs” and “New Terminals”
Harinder++

- Harinder covered many of the essential Linux basics in lectures 1 and 2
- A few additional commands/utilities will help with my sessions and further
  - Changing terminal configurations to make it “easier to work with” (in my opinion)

  Make your default startup size as you wish (e.g. taller)

  Change the “scroll capture” (useful when inspecting program runtime output)
Harinder++

- Harinder covered many of the essential Linux basics in lectures 1 and 2
- A few additional commands/utilities will help with my sessions and further
- Working with remote computers - lxplus at CERN
  - `ssh`: transform current terminal (shell) into a remote session on that computer

```
$ ssh USERNAME@lxplus.cern.ch
```

![Map showing remote access between Your Computer and lxplus.cern.ch](image_url)
Harinder++

- Harinder covered many of the essential Linux basics in lectures 1 and 2
- A few additional commands/utilities will help with my sessions and further
- Working with remote computers - lxplus at CERN
  - `ssh`: transform current terminal (shell) into a remote session on that computer

Working on my local machine  
Same terminal that you opened, but working on a remote machine (i.e. lxplus)
Harinder++

- Harinder covered many of the essential Linux basics in lectures 1 and 2
- A few additional commands/utilities will help with my sessions and further
- Working with remote computers - \texttt{lxplus at CERN}
  - \texttt{scp} : transfer files to/from a remote computer

$ scp \texttt{path/to/local\_file.txt} \texttt{USERNAME@lxplus.cern.ch:/afs/cern.ch/user/m/meehan/transferred\_file.txt}$
Harinder++

- Harinder covered many of the essential Linux basics in lectures 1 and 2
- A few additional commands/utilities will help with my sessions and further
- Working with remote computers - lxplus at CERN
  - scp: transfer files to/from a remote computer

```
$ scp USERNAME@lxplus.cern.ch:/afs/cern.ch/user/m/meehan/transferred_file.txt path/to/local_file.txt
```
Harinder++

- Harinder covered many of the essential Linux basics in lectures 1 and 2
- A few additional commands/utilities will help with my sessions and further
- Working with remote computers - lxplus at CERN
  - `scp` : transfer files to/from a remote computer

Working on remote machine, the file exists that I want to get

Working on my local machine, I use `scp` to transfer that file
Harinder++

- Harinder covered many of the essential Linux basics in lectures 1 and 2
- A few additional commands/utilities will help with my sessions and further
- A few additional commands: remember to use “man”
  - `cat`: dumps contents of a file to the screen
    - Use it to inspect text files quickly
  - `CTRL+A (^A)`: jump to the beginning of a continuous line
    - Lines/paths will get long and this will save time
  - `CTRL+E (^E)`: jump to the end of a continuous line
    - Lines/paths will get long and this will save time
  - `Tab Completion`: Your computer “knows” what you want to type
    - Our memory often fails us
  - `Up/Down Buttons`: Recall former commands

- If you have the thought “You know, I wish there was a command that did `<FILL_IN_THE_BLANK>`.” there probably is and you should google it

A nice cheat sheet of shell/bash commands: [link here]
Editors - Not MS Word

If you use Microsoft Word (or something like it) then you must abandon this in the land of programming ... why?

**GOOD**: You can point and click on stuff which is what you are used to

**BAD**: Lots of formatting information needs to be stored along with the text ("WYSIWYG")

**BAD**: Header and footer need to be stored in file

**BAD**: Not all space that looks white is actually white ...
Editors - Not MS Word

If you use Microsoft Word (or something like it) then you must abandon this in the land of programming ... why?

**GOOD**: No header and footer

**GOOD**: No margin information stored

**GOOD**: You can point and click on stuff which is what you are used to

**BAD**: Still have formatting information ...
Editors - Not MS Word

If you use Microsoft Word (or something like it) then you must abandon this in the land of programming ... why?

GOOD: Text is actually text. It looks boring and monochrome.

GOOD: You can actually configure all of these colors so that it's not *so* monochrome (open source)

GOOD: Whitespace is whitespace

GOOD: No margin information stored

GOOD: No header and footer

BAD: You don’t get to click (always). Requires some getting used to
Editors - Not MS Word

If you use Microsoft Word (or something like it) then you must abandon this in the land of programming ... why?

User Friendly
Fluffy

Acquired Taste
Technically Oriented
Editors - nano

- Nano is a very basic editor that is good for beginners
  - I started with it when I first learned programming
- It gives you a built in cheat sheet at the bottom - example: \(^C = \text{control+C}\)

[1] Open a new file
[2] Typing directly enters content at the location of the cursor
[3] Type \(^O\) to start the write-out and then it will ask you to confirm → press enter

Go here for more: https://www.nano-editor.org/
Editors - nano

- Nano is a very basic editor that is good for beginners
  - I started with it when I first learned programming
- It gives you a built in cheat sheet at the bottom - example: \(^C = \text{control}+C\)

[4] Go to the place in the line where you want to start cutting from → press ^K
The text from that point to the end of the line is now saved in memory
Go to where you want to paste it → press ^U

Go here for more: [https://www.nano-editor.org/](https://www.nano-editor.org/)
Editors - nano

- Nano is a very basic editor that is good for beginners
  - I started with it when I first learned programming
- It gives you a built in cheat sheet at the bottom - example: $^C = \text{control}+C$

Exercise: How do you think you search for something?

[6] If you try to exit ($^X$) without saving, it will ask you to confirm this

Go here for more: https://www.nano-editor.org/
Editors - emacs

- Emacs is a more flexible editor that is a good next step
  - This is what I use when I want to access lxplus from “any random computer”
- No cheat sheet → so it takes some learning of the key combinations
  - Many are formatted like “C-x C-s” = Type Control+x and then Control+s

[1] Make a new file just like in nano (-nw suppresses graphics porting)
[2] Editing is just like in nano (start typing)
[3] To save → type C-x and then C-s

The command sequence is shown as you type

It gives you some feedback once you finish

Go here for more: https://www.gnu.org/software/emacs/manual/
Editors - emacs

- Emacs is a more flexible editor that is a good next step
  - This is what I use when I want to access lxplus from “any random computer”
- No cheat sheet → so it takes some learning of the key combinations
  - Many are formatted like “C-x C-s” = Type Control+x and then Control+s (not always true)

[4] Go to the place in the line where you want to start cutting from → press C-k (Kill)

Go to where you want to paste it and type C-y (Yank)

[5] Type C-s → type string you want to search for

Go here for more: https://www.gnu.org/software/emacs/manual/
Editors - emacs

- Emacs is a more flexible editor that is a good next step
  - This is what I use when I want to access lxplus from “any random computer”
- No cheat sheet → so it takes some learning of the key combinations
  - Many are formatted like “C-x C-s” = Type Control+x and then Control+s

[6] To exit, execute the command sequence “C-x C-c”

Asks you to confirm the exit (just like nano)

Go here for more: https://www.gnu.org/software/emacs/manual/
Editors - vim

- vim is the gold standard in that its very standard (found on most machines)
  - I never use this because I think its “too complicated”
  - Some people love it (e.g. Harinder) and swear by it
- Main learning hurdle: **INSERT** vs. **EDIT** mode
  - **INSERT**: Adding new/original text that is not already there
  - **EDIT**: Everything is “hotkey-combinations” → requires learning (for me, via Youtube)

Main page here: [http://www.vim.org/](http://www.vim.org/)
Another cheat sheet: [https://vim.rtorr.com/](https://vim.rtorr.com/)
I learned also from YouTube: [video here](#)
Editors - vim

- vim is the gold standard in that its very standard (found on most machines)
  - I never use this because I think its “too complicated”
  - Some people love it (e.g. Harinder) and swear by it
- No cheat sheet at bottom → so it takes some learning of the key combinations

[1] Can you guess how to create a new file in vim

[2] Typing is not a given → need to type “i”

Now you can start typing stuff and it will appear as modifications

Main page here: [http://www.vim.org/](http://www.vim.org/)
Cheat sheet: [https://vim.rtorr.com/](https://vim.rtorr.com/)
Editors - vim

- vim is the gold standard in that its very standard (found on most machines)
  - I never use this because I think its “too complicated”
  - Some people love it (e.g. Harinder) and swear by it
- No cheat sheet at bottom → so it takes some learning of the key combinations

Main page here: http://www.vim.org/
Cheat sheet: https://vim.rtorr.com/

[2b] When you are ready to no longer insert → type “Esc” to go into edit mode

[3] Save files in edit mode → type “.:” and then “w” and then Enter

The command sequence is shown as you type (like emacs)

It tells you once executed (like emacs)
Editors - vim

- vim is the gold standard in that its very standard (found on most machines)
  - I never use this because I think its “too complicated”
  - Some people love it (e.g. Harinder) and swear by it
- No cheat sheet at bottom → so it takes some learning of the key combinations

[4] To remove a line you will “delete” it in edit mode → type “dd” (delete)

[5] To search for something in edit mode→ start with hotkey “/” and type pattern afterwards (e.g. “/this”)
Editors - vim

- vim is the gold standard in that its very standard (found on most machines)
  - I never use this because I think its “too complicated”
  - Some people love it (e.g. Harinder) and swear by it
- No cheat sheet at bottom → so it takes some learning of the key combinations

[6] To quit from edit mode → start with hotkey “:” and type “quit”

Main page here: [http://www.vim.org/](http://www.vim.org/)
Cheat sheet: [https://vim.rtorr.com/](https://vim.rtorr.com/)

If this joke isn’t funny, then you should review Harinder’s slides
Editors: GUI applications

- Can we have the best of both worlds? → YES!
  - Industry commonly uses tools that make your life easier
- GUI editors and IDE’s (Integrated Development Environment) exist to make coding and project management more manageable

**GOOD**: Easily see which files are openned

**GOOD**: Access files on a remote machine via “scp”

**GOOD**: Can customize colors much more easily than in emacs like we saw before

**GOOD**: Get to click on stuff (“GUI”) so you don’t have to remember so many hotkeys

**GOOD**: The list goes on ...

**BAD**: Not on Ubuntu
Editors: GUI applications

- Can we have the best of both worlds? → YES!
  - Industry commonly uses tools that make your life easier
- GUI editors and IDE’s (Integrated Development Environment) exist to make coding and project management more manageable

Tabs let you open multiple files like a web browser

**Sublime Text**
- Useful thing for when code gets long
- See exactly where you are on your computer
- Pretty colors for text - more useful than you may think

**Atom Editor**
- Use useful thing for when code gets long
Exercise: .bashrc greeting

Try to add a greeting (e.g. “Hey Sam, now get to work!”) to your .bashrc file. This should be printed each time you open a new terminal.

- Try following along with these steps
  - Open a terminal
  - Go to your home directory → $ cd
  - Open for editing (e.g. in nano) the “.bashrc” file) → $ nano .bashrc
    - There’s a lot of stuff in here already
  - Add the following line at the very top of the file → [echo “Hey Sam, now get to work!”]
    - Don’t forget the quotes
  - Save the file however you must in your editor of choice
  - Exit the editor
  - Close the terminal and open a new one

- Do you see your nice message at the top?