

# Development Environment



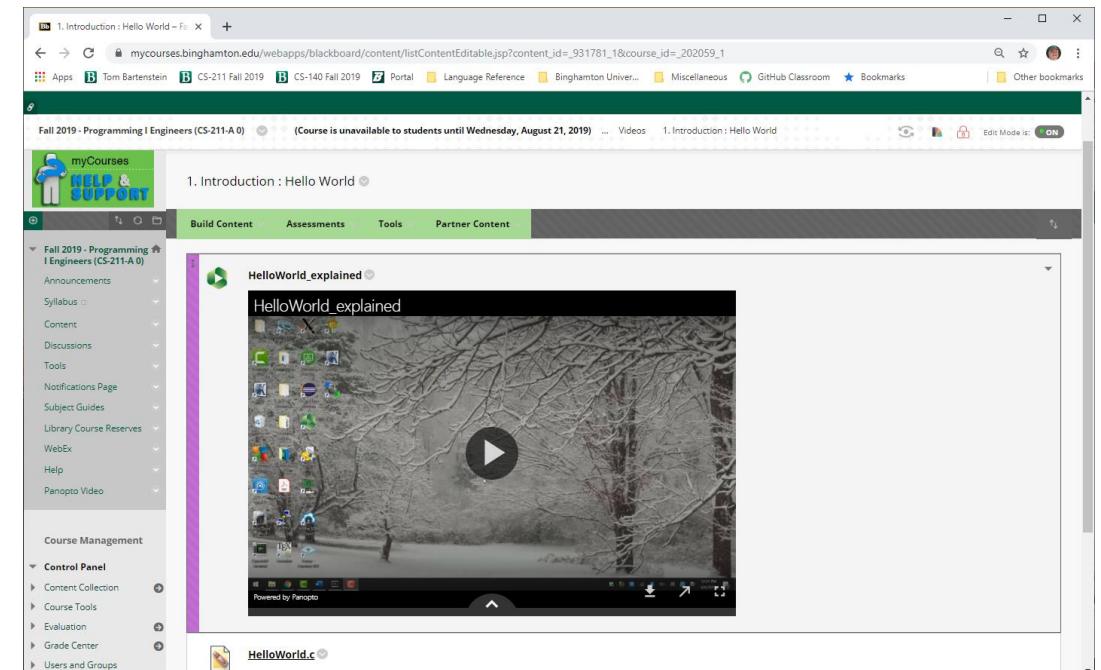
# iClicker Attendance

Please click on A if you are here:

A. I am here today.

# Video Review: Hello World

- Terminal Window
- Unix Directories
- `#include`
- The main function
- C instructions
- Running the compiler
- Running the compiled code (the command)
- Compiler errors

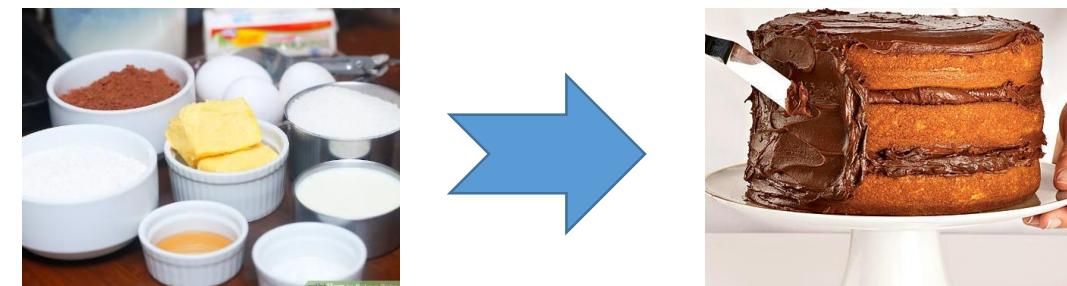


# Demonstration

- Creating and Running “Hello World” using SSH/Remote
- Showing edit:
  - On remote system
  - Using MobaXterm automated Upload/Download
  - Using WinSCP automated Upload/Download
  - Using mounted U-Drive

# Algorithm

- List of instructions that transform input into output
  - Finite set of steps
  - Executed in a definite order by a deterministic mechanism
  - Execution must terminate
- Analogous to a recipe
  - Finite set of steps
  - Executed in a definite order (but may not be a deterministic mechanism)
  - Execution terminates



# Computer

---

A programmable  
algorithm execution  
machine



# iClicker Question

Please click on the FIRST choice that is true:

- A. I have written and run a program in C.
- B. I have written and run a program in Python.
- C. I have written and run a program in some other language (e.g. Java)
- D. I have never written a computer program before

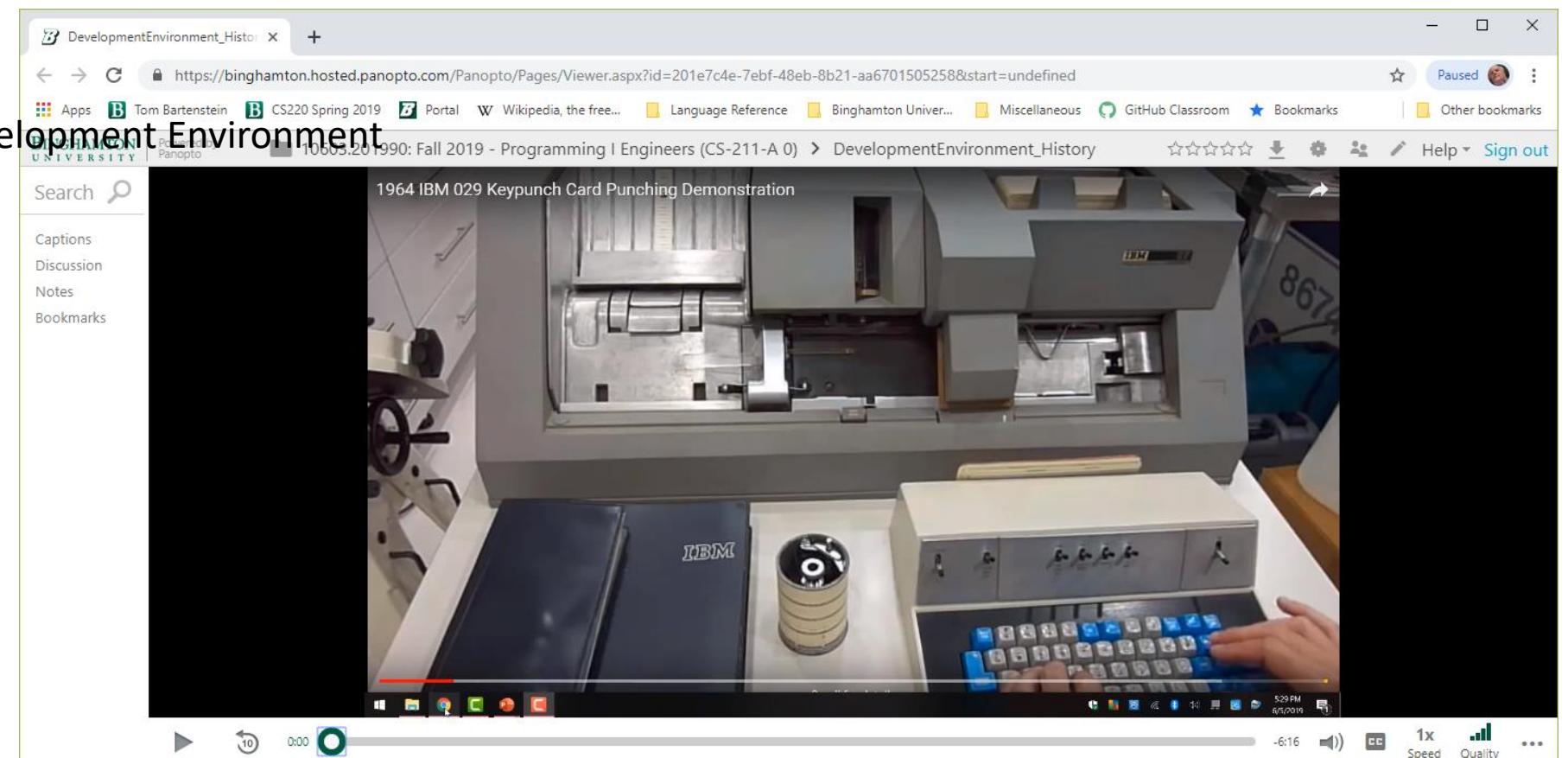
# Historical Background: Terminal Window

- Watch the video, available on myCourses

- Content

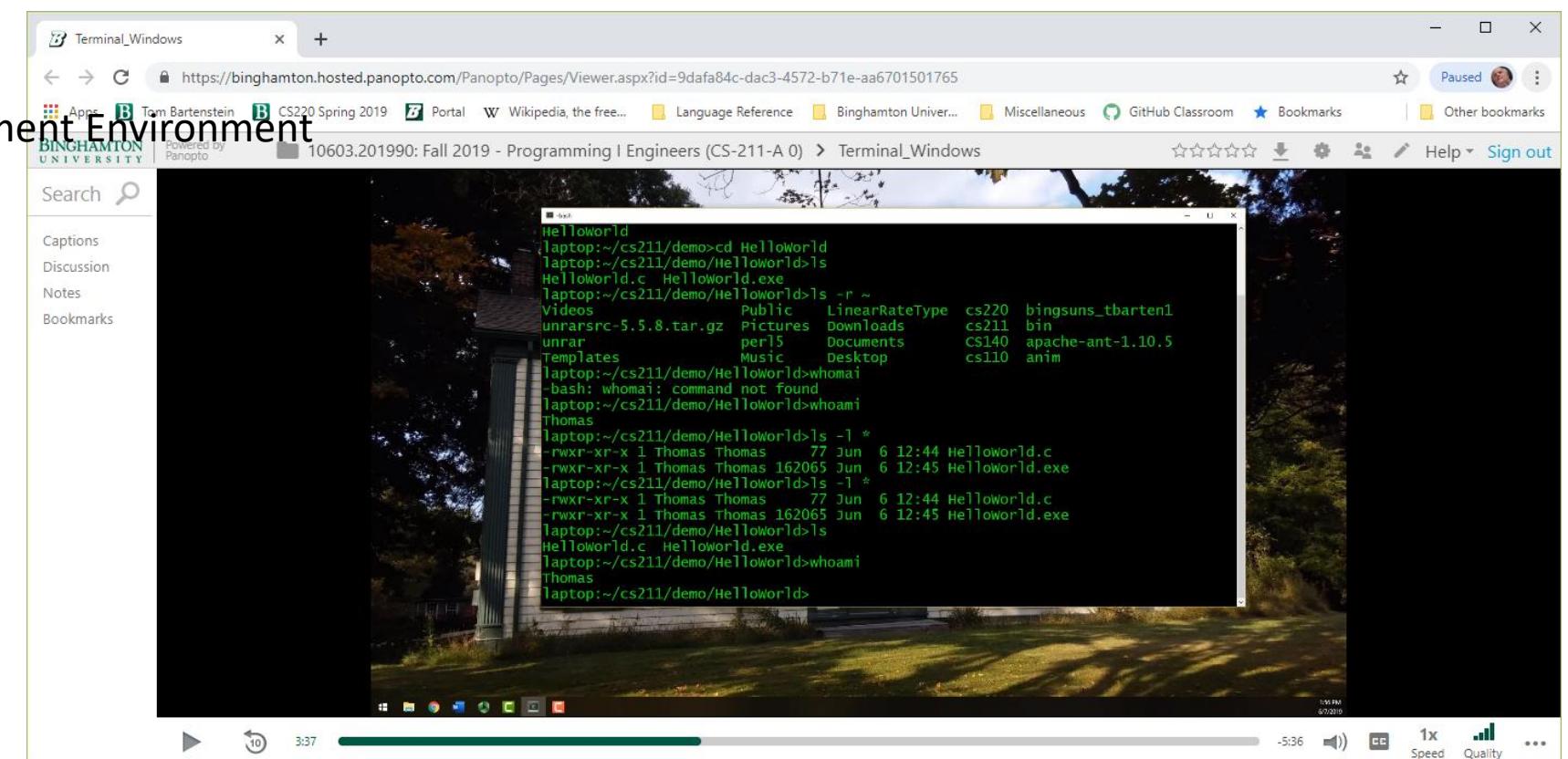
- Videos

- 2. Development Environment



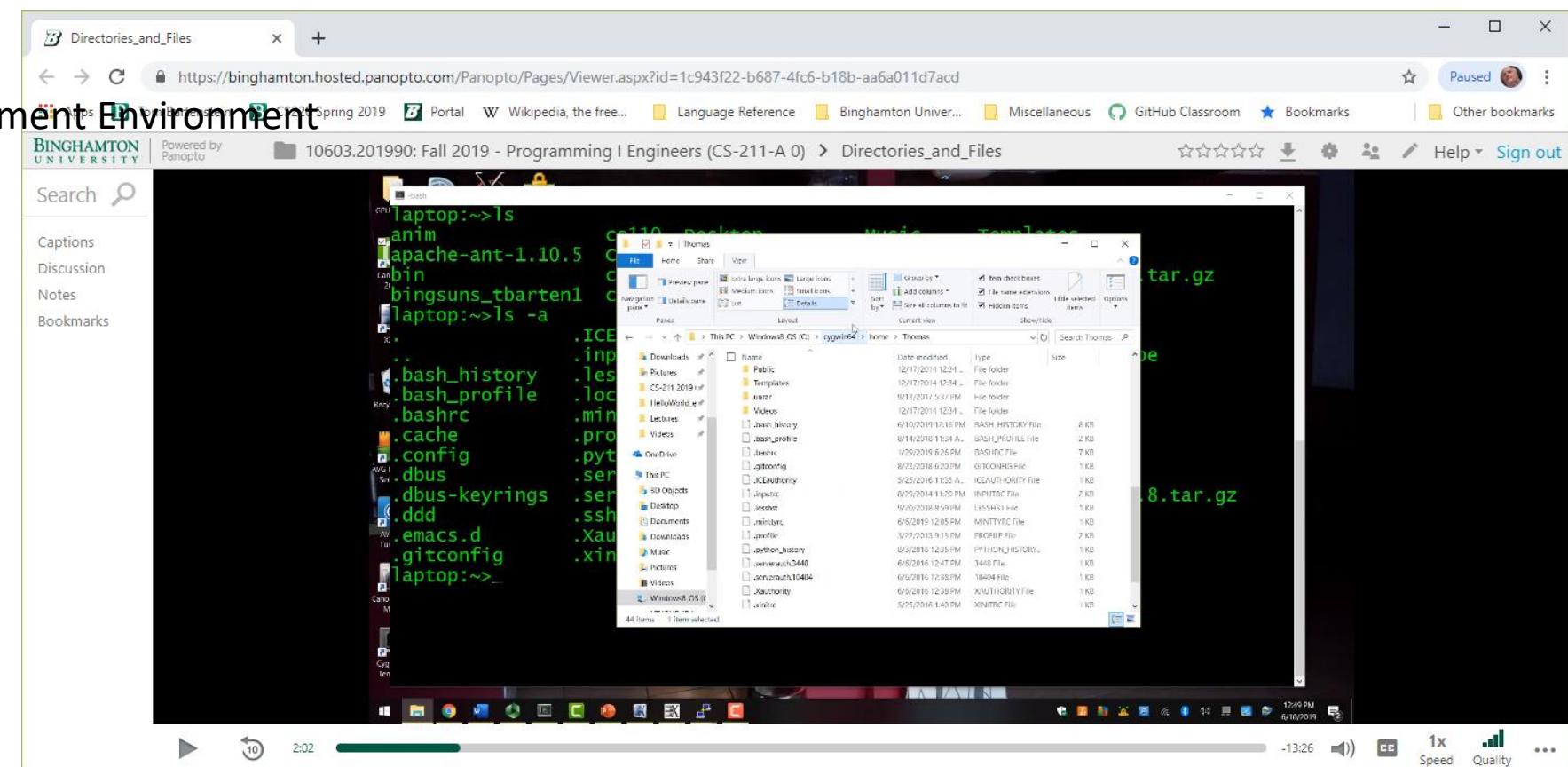
# Using Terminal Windows

- Watch the video, available on myCourses
  - Content
  - Videos
    - 2. Development Environment



# Files and Folders

- Watch the video, available on myCourses
  - Content
    - Videos
      - 2. Development Environment



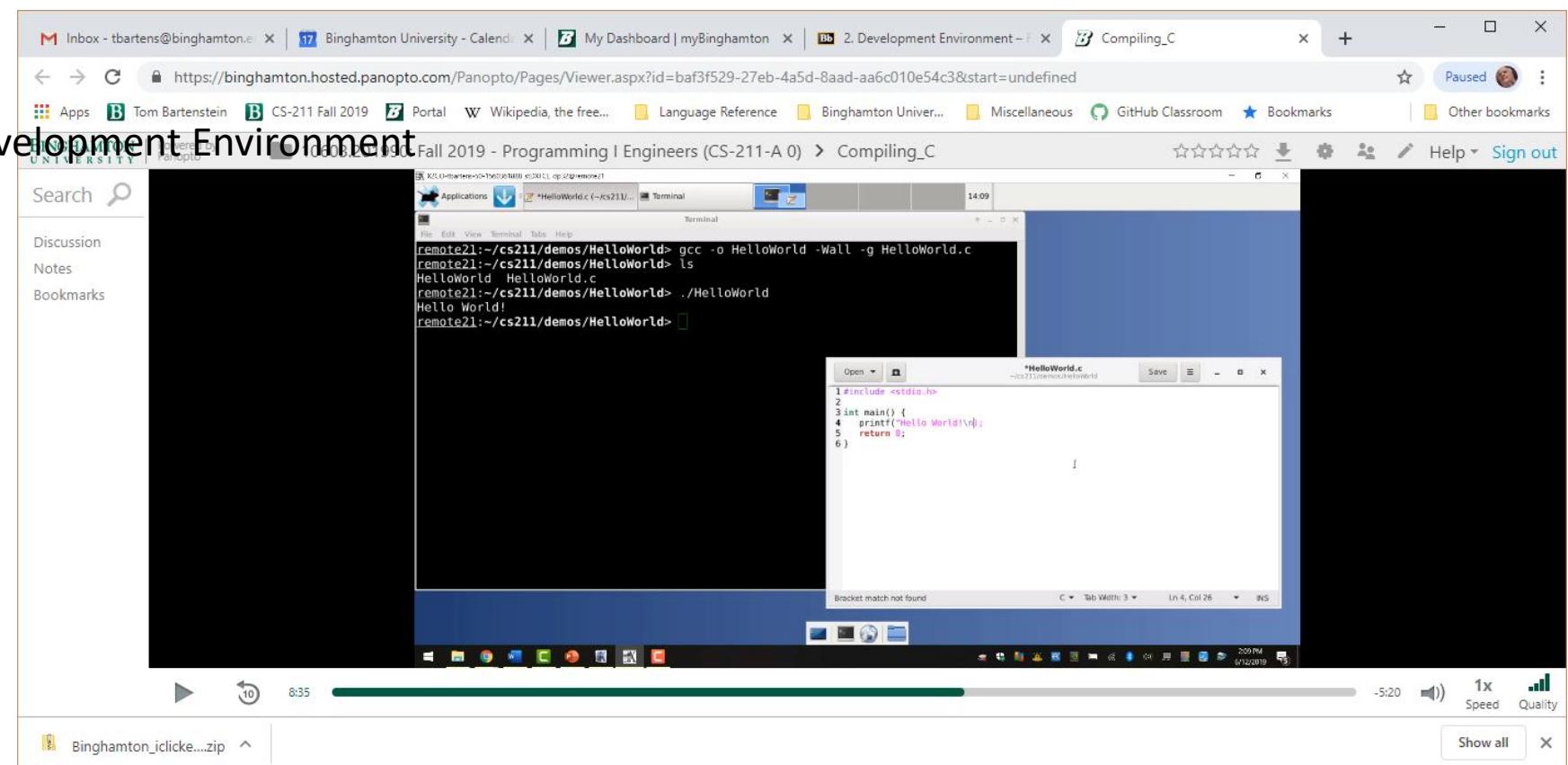
# Compiling C Code

- Watch the video, available on myCourses

- Content

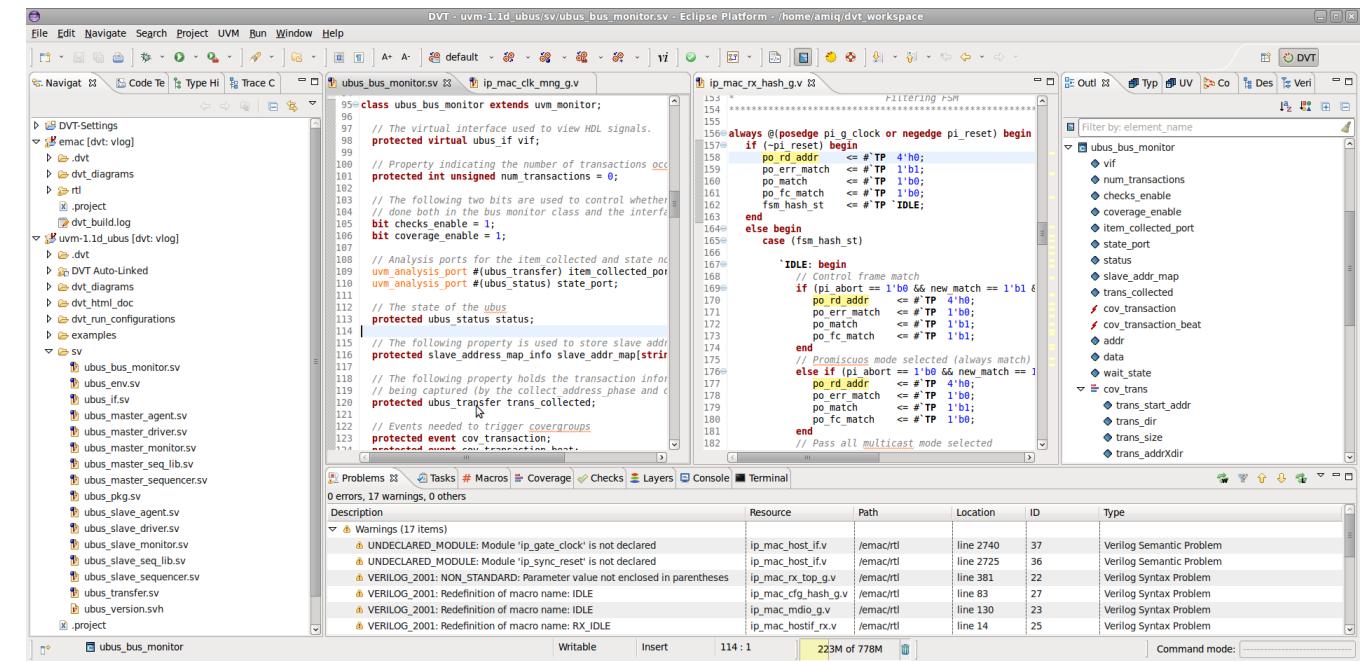
- Videos

- 2. Development Environment



# Integrated Development Environments

- Single interactive graphical user interface (GUI)
- Includes:
  - File/Project Management
  - Editor
  - Compiler
  - Builder
  - Debugger
  - Execution Environment



The screenshot shows the Eclipse Platform IDE interface with the following components:

- Code Editor:** Displays Verilog code for `ubus_bus_monitor.sv` and `ip_mac_rx_hash_g.v`. The code includes declarations for `ubus_if`, `num_transactions`, `checks_enable`, and `state_port`, along with logic for `idle` and `tx` states.
- Problems View:** Shows 0 errors, 17 warnings, and 0 others. A table lists the warnings, including undeclared modules and non-standard parameter values.

Resource	Path	Location	ID	Type
ip.mac_host_if.v	/emacrtl	line 2740	37	Verilog Semantic Problem
ip_mac_host_if.v	/emacrtl	line 2725	36	Verilog Semantic Problem
ip.mac_rx_top.g.v	/emacrtl	line 381	22	Verilog Syntax Problem
ip_mac_cfg_hash.g.v	/emacrtl	line 83	27	Verilog Syntax Problem
ip_mac_mdio_g.v	/emacrtl	line 130	23	Verilog Syntax Problem
ip_mac_hostif_rx.v	/emacrtl	line 14	25	Verilog Syntax Problem

# iClicker Question

Please click on the FIRST choice that is true:

- A. I have written code without using an IDE.
- B. I have written code using an IDE.
- C. I have seen a friend writing code using an IDE
- D. I have never written a computer program before

# Why not use an IDE for CS-211?

- There is no generally acceptable IDE for C
- IDE hides the basics of programming
  - Things you don't have to know the rest of your life, but
  - You should understand how they work at least once.
  - Know what an IDE is doing for you!
- Learn the alternatives to an IDE
  - Sometimes, the old ways are faster / more efficient
  - Sometimes you are in an environment where the IDE doesn't exist



# Exercise

What is  $x$  times  $x$ ?

$$x \cdot x = ?$$

- Write a computer program that reads a number, and computes the square of that number.

# Infrastructure

- Use the same basic structure of a program that we used for “HelloWorld.c”

```
#include <stdio.h>
```

```
int main() {
    printf("Hello World!\n");
    return 0;
}
```

- (But use a different sub-directory and a different file name, e.g. “square.c”)

# What kind of a number is $x$ ?

- Symbols: '2' '5' is that 2,5 or 25 or 2.5???
- Integer?  $3 \times 3 = 9$ ,  $5 \times 5 = 25$ , ...
- Real Number:  $3.1 \times 3.1 = 9.61$ ,  $5.5 \times 5.5 = 30.25$ , ...
  - How big or small?
  - How much precision?
- In a C program, we create a variable and assign a type to tell C what kind of number we are using:

`float x;`

read as “ $x$  is a floating point number” (floating point is “real”)

# How do we get a value for “x”?

- Ask our “user” (person running the program) for a value.
- First, tell the user we are looking for a value to square:

```
printf("Enter a number to square :>");
```

- Then, invoke a C library function to get input from the user

```
scanf("%d ",&x);
```

# How do we calculate and print the result?

- Calculate  $x^2$  by using the C expression:  $x*x$ 
  - Multiply the current value of  $x$  by itself
- Print the result using the C library function, `printf`:

```
printf("The square of %f is %f\n", x, x*x);
```

# How do we compile the result?

- Use the same concepts as “HelloWorld”

```
> gcc -g -Wall -o square square.c
```

# How do we run the result?

- Use the same concepts as in HelloWorld
  - But now we have to enter a number at the prompt and hit “enter”!

```
>./square
```

```
Enter a number to square :> 3.1
```

```
The square of 3.100000 is 9.610000
```

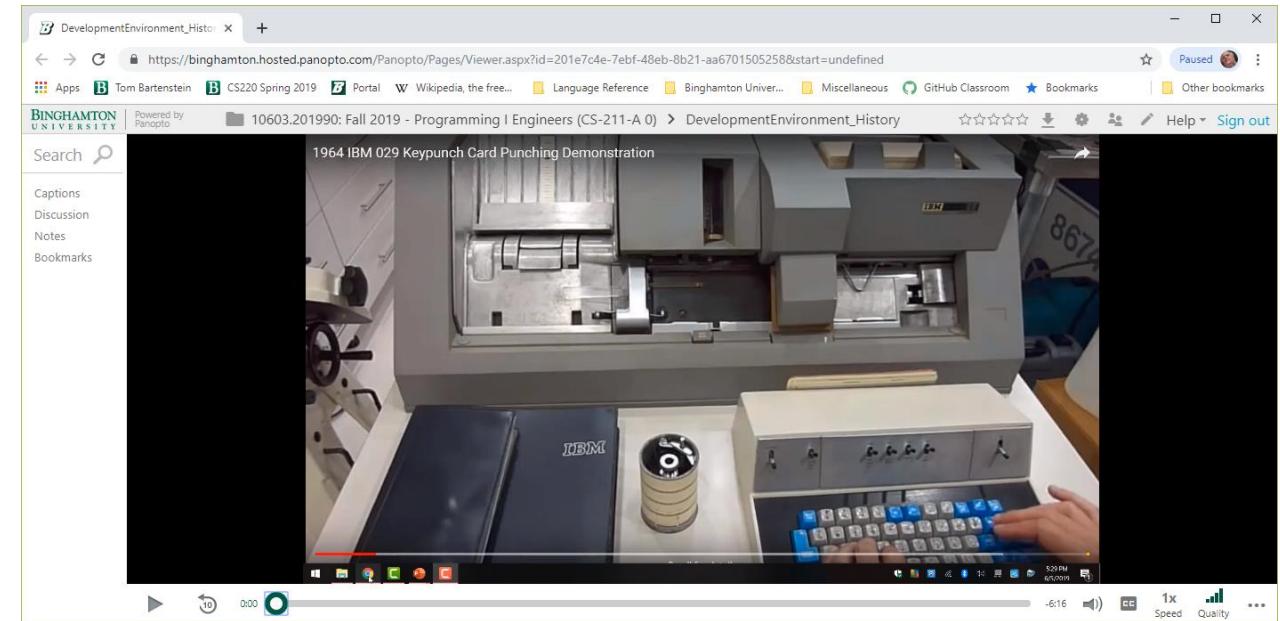
```
>
```

# How do we test our code?

- Think about what might break the program
  - Simple tests first: 3,5
  - More complicated: 3.1, 5.5
  - Negative numbers? : -7, -9.3
  - How big can I get?: 100, 999999999
  - How little can I get?: 0.0001, 0.00000000000001
  - What if user doesn't enter a number? : hello

# Resources

- Programming in C Chapters 1&2
- UNIX tutorial <http://www.ee.surrey.ac.uk/Teaching/Unix/>
- Wikipedia: List of Unix Commands [https://en.wikipedia.org/wiki/List\\_of\\_Unix\\_commands](https://en.wikipedia.org/wiki/List_of_Unix_commands)
- gedit wiki <https://wiki.gnome.org/action/show/Apps/Gedit?action=show&redirect=Gedit>
- gedit on-line manual <https://help.gnome.org/users/gedit/stable/>
- gcc on-line manual <https://gcc.gnu.org/onlinedocs/gcc-4.7.4/gcc/>
- Wikipedia: IDE [https://en.wikipedia.org/wiki/Integrated\\_development\\_environment](https://en.wikipedia.org/wiki/Integrated_development_environment)

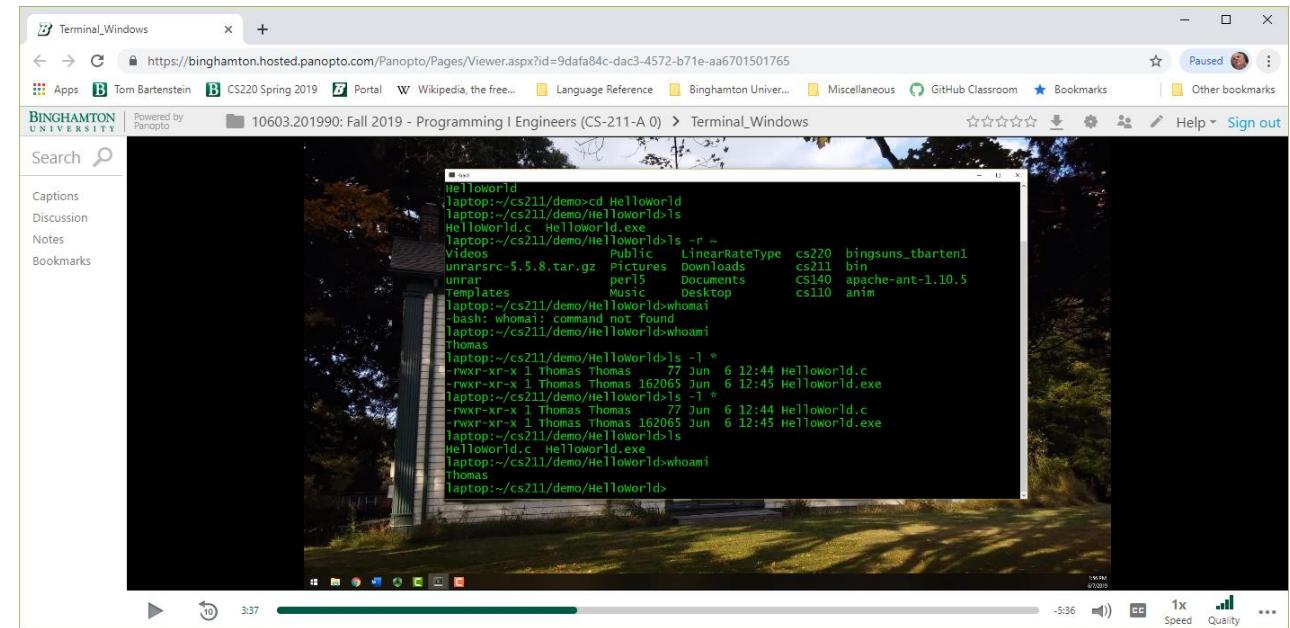


# Historical Background

Summary Notes

# Historical Background

- IBM Punch Card: <https://www.youtube.com/watch?v=YnnGbcM-H8c>
- Reading punch cards:  
<https://www.youtube.com/watch?v=w62NC1R6WLs>  
<https://www.youtube.com/watch?v=w62NC1R6WLs>
- Printing Results: <https://www.youtube.com/watch?v=FiEGoVzmyvs>
- Playing Music on a printer:  
<https://www.youtube.com/watch?v=jufHGUp3xQw>
- Linux Teletype: <https://www.youtube.com/watch?v=-Ul-f3hPJQM#t=67.618982>

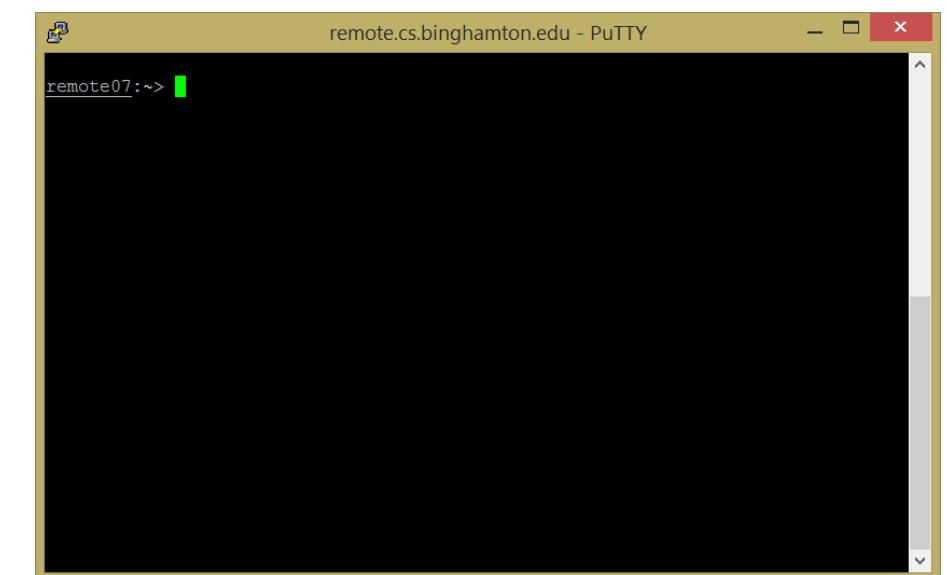


# Terminal Windows

## Summary Notes

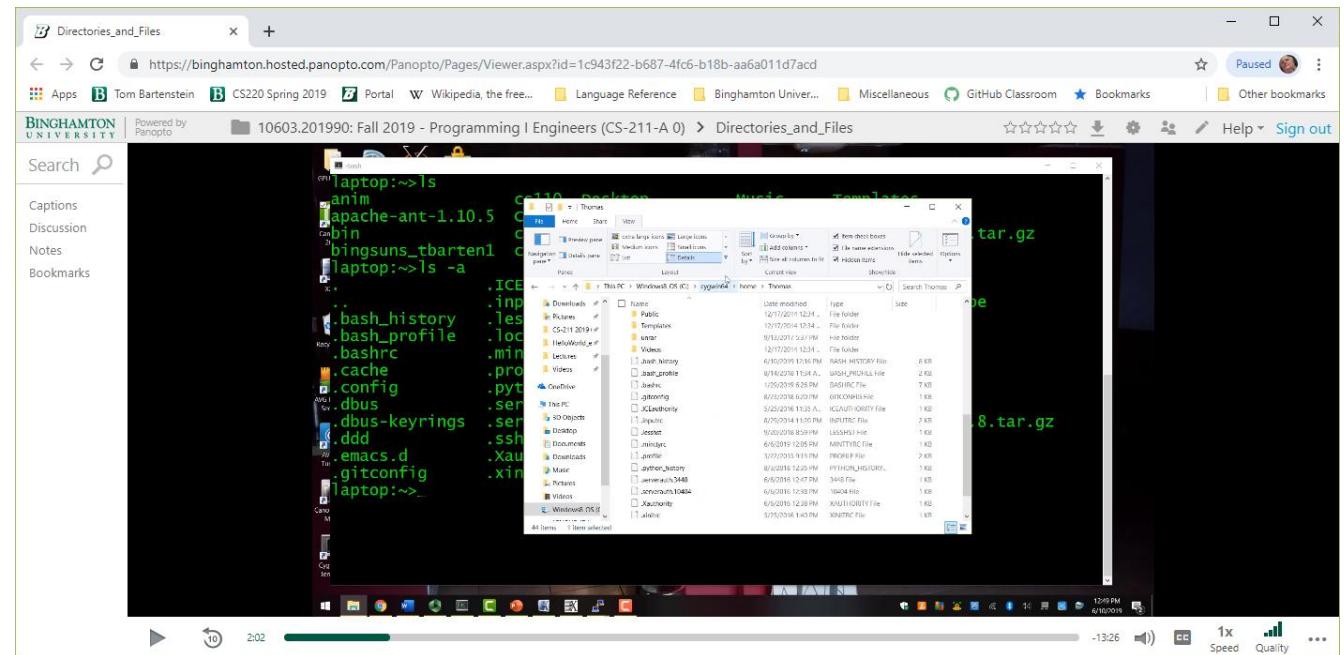
# UNIX “terminal”

- Command line window
- “Prompt”
  - Machine name
  - Current Directory
  - “lab01:~/cs211>”
- Type commands at prompt
  - Nothing happens until you hit “<enter>”
- When you hit enter...
  - Command is executed
  - Response appears
  - Another prompt appears



# UNIX terminal hints

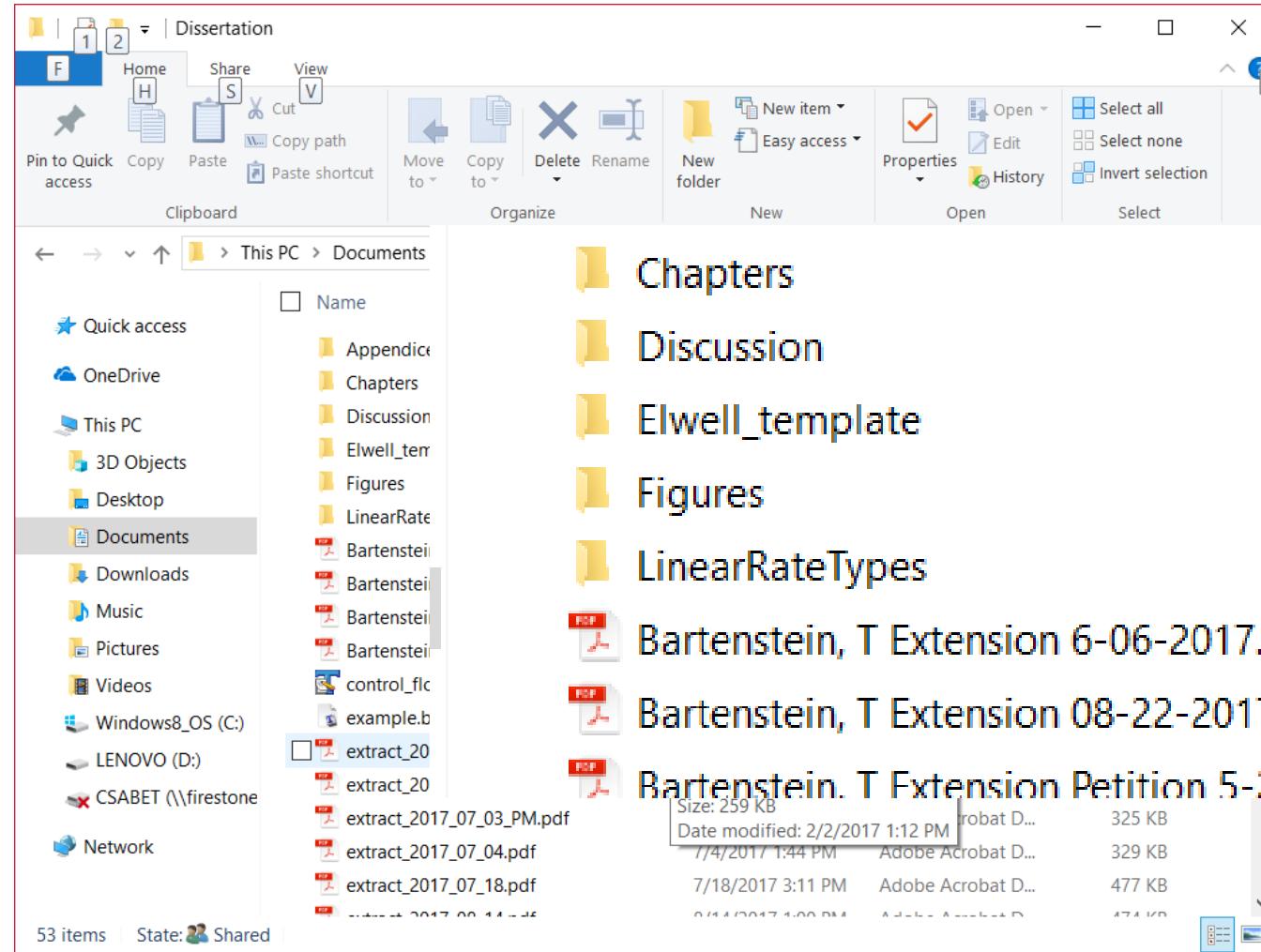
- Output is scrollable... scroll up to see what happened before
  - Scroll amount is finite... you have lost what happened too many lines ago
- Command Editing
  - You can use the left and right arrows to move in a command before you hit enter
  - Typing new characters inserts, also delete or backspace
- Command History
  - There is a command history
  - Use up and down arrows to retrieve earlier commands



# Files and Folders

## Summary Notes

# Files and Folders



The screenshot shows a Windows File Explorer window with a red border. The title bar says 'Dissertation'. The ribbon menu includes 'Clipboard' (Pin to Quick access, Copy, Paste, Move to, Copy to, Delete, Rename, New folder, New item, Open, Select all, Select none, Select none, Invert selection), 'Organize' (New item, Open, Select), and 'Select' (Select all, Select none, Select none, Invert selection). The left sidebar shows 'This PC > Documents' with a list of drives and network locations. The main pane lists files and folders with their names, modification dates, and types. A context menu is open over the bottom-most PDF file, showing options like 'Size: 259 KB', 'Date modified: 2/2/2017 1:12 PM', 'Adobe Acrobat D...', '7/4/2017 1:44 PM', 'Adobe Acrobat D...', '325 KB', '329 KB', '7/18/2017 3:11 PM', 'Adobe Acrobat D...', '477 KB', '0/14/2017 1:00 PM', 'Adobe Acrobat D...', '474 KB', and 'Add to archive'.

Name	Modified	Type
Chapters	8/7/2017 12:53 PM	File folder
Discussion	12/27/2016 12:51	File folder
Elwell_template	10/28/2016 4:09 PM	File folder
Figures	1/29/2018 2:17 PM	File folder
LinearRateTypes	8/20/2016 11:37 A	File folder
Bartenstein, T Extension 6-06-2017.pdf	6/7/2017 11:58 AM	Adobe Acrobat D...
Bartenstein, T Extension 08-22-2017.pdf	8/22/2017 4:40 PM	Adobe Acrobat D...
Rartenstein, T Extension Petition 5-25-...	5/28/2017 2:51 PM	Adobe Acrobat D...

# “Home Directory”

- When you log in to UNIX, you are in your “Home” directory
  - .e.g /home/tbarten1
- Shorthand: “~”
- You own your home directory
  - You can create or remove files
  - You can create or remove sub-directories
  - You control attributes of files in your home directory
- You may be able to read outside your home directory
- You probably cannot write outside your home directory



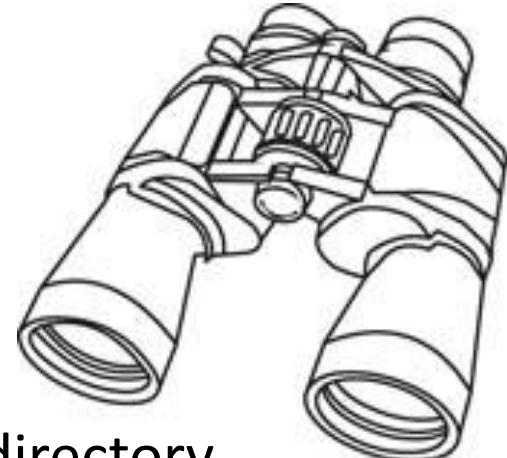
# Current Directory



- “Current Directory” is the directory you are working in
  - Starts at your home directory
- Abbreviated as “.”
- Change the current directory with the command “cd <argument>”
  - “cd /<fully-qualified directory>” move to the fully qualified directory
  - “cd <label> moves to the <label> sub-directory of the current directory
  - “cd ..” moves to the parent of the current directory
  - “cd” with no argument returns to your home directory

# Listing what's in a directory

- “ls <arguments>”
  - “ls” with no arguments lists what’s in the current directory
  - “ls <label>” lists what’s in the label sub-directory of the current directory
  - ...
- “ls -l <arguments>” lists file names and attributes



# Unix Directory Commands

- Make a new directory using “`mkdir <label>`”
  - makes a sub-directory of your current directory
  - e.g. “`mkdir examples`” will create directory “`/home/tbarten1/examples`”
    - `<label>` becomes a sub-directory of the current directory
    - The “examples” directory will be empty
- Remove an empty directory using “`rmdir <label>`”
  - e.g. “`rmdir examples`” removes the “examples” subdirectory of the current directory



# Using Directories

- In the Linux Lab, your home directory is space provided by BU
- Your home directory is shared between Linux and Windows on PODS
- Recommend, make a subdirectory for each class

~> mkdir cs211

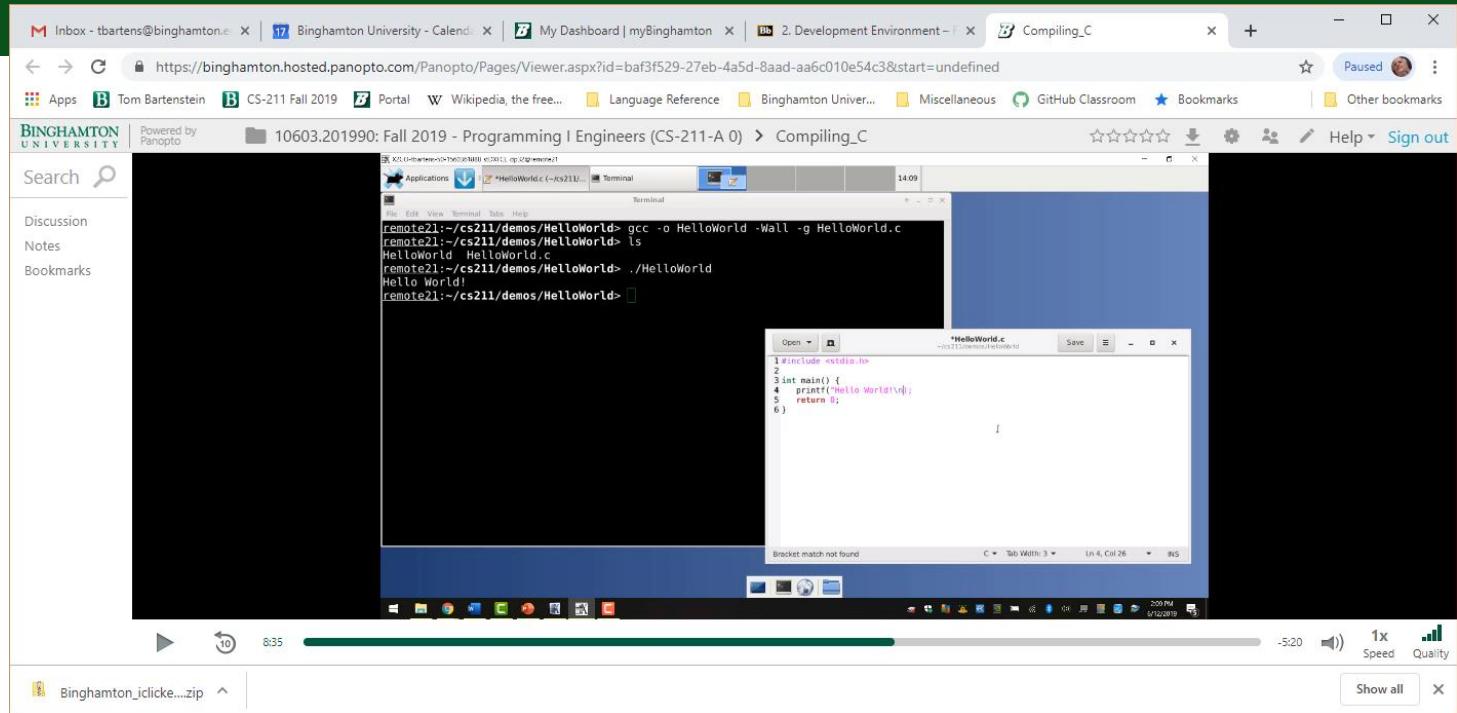
~> cd cs211

- Typically, make a subdirectory for each activity in the class

~/cs211> mkdir lab01

~/cs211> cd lab01

~/cs211/lab01>

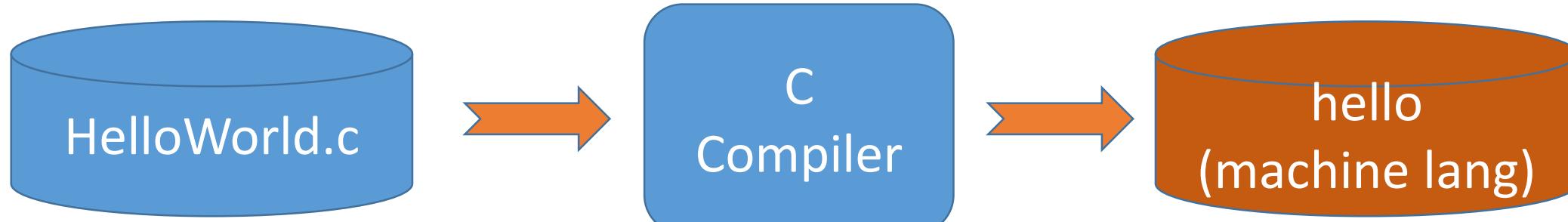


# Compiling C Code

Summary Notes

# Computer Instructions

- Computers “understand” a very low level binary set of instructions called ***machine language***
- Machine language is not easily read or understood by humans
- A ***programming language*** is a well defined “language” with a precise syntax and meaning (semantics)
  - Designed to be understood by humans
  - Easily translated into machine language by a compiler
  - An algorithm expressed in a computer language is a computer program



# Compiling C Code

- Compile command: gcc (Followed by parameters)
  - “-o <filename>” to create executable file named <filename>
  - “-g” to enable debugging
  - “-Wall” to turn on all warning messages

**gcc –o HelloWorld -g -Wall HelloWorld.c**



- Compiles HelloWorld.c from the current directory
- If there are no errors, writes executable file “HelloWorld”

# Compiler Messages

- When the compiler gets confused it generates a message
  - Warning: Something seems wrong, but compiler can live with it
  - Error: Something is wrong that stops the compile – No output generated!
- Messages contain the <line>.<column> that the compiler was working on when it got confused.
- Message contains the compiler's description of the problem

```
floatx.c:32:2: warning: statement with no effect [-Wunused-value]
result << 1;
^
```

```
floatx.c:71:9: error: invalid operands to binary << (have 'double' and 'int')
result << 1;
^
```



# Running C Code

- You need to tell the operating system what executable file to run
- Shorthand is to use “.” to represent the current directory
- e.g. “./mycmd” runs the executable file “mycmd” in the current directory

