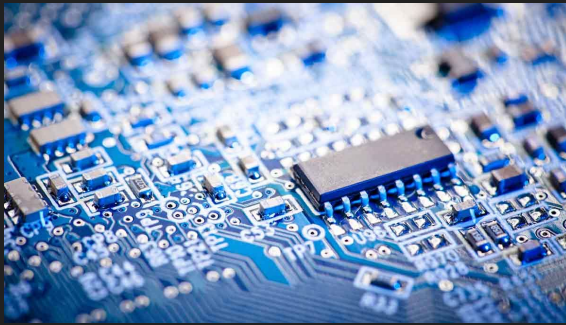


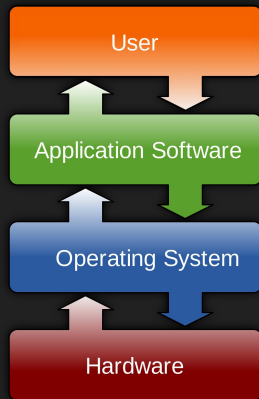
Please do not redistribute these slides
without prior written permission



CS3650

Computer Systems

Ferdinand Vesely



Intro	Virtualization	Concurrency	Persistence	Appendices
Preface	3 Dialogue	12 Dialogue	25 Dialogue	35 Dialogue
TOC	4 Processes	13 Address Spaces	26 Concurrency and Threads	36 I/O Devices
1 Dialogue	5 Process API	14 Memory API	27 Thread API	37 Hard Disk Drives
2 Introduction	6 Direct Execution	15 Address Translation	28 Locks	38 Redundant Disk Arrays (RAID)
	7 CPU Scheduling	16 Segmentation	29 Locked Data Structures	39 Files and Directories
	8 Multi-level Feedback	17 Free Space Management	30 Condition Variables	40 File System Implementation
	9 Lottery Scheduling	18 Introduction to Paging	31 Semaphores	41 Fast File System (FFS)
	10 Multi-CPU Scheduling	19 Translation Lookaside Buffers	32 Concurrency Bugs	42 FFSCK and Journaling
	11 Summary	20 Advanced Page Tables	33 Event-based Concurrency	43 Log-structured File System (LFS)
		21 Swapping Mechanisms	34 Summary	44 Flash-based SSDs
		22 Swapping Policies		45 Data Integrity and Protection
		23 Case Study: VAX/VMS		46 Summary
		24 Summary		47 Dialogue
				48 Distributed Systems
				49 Network File System (NFS)
				50 Andrew File System (AFS)
				51 Summary

Pre-Class Warmup

- Take a moment, and introduce yourself to someone next to you. They are going to be your colleagues for the next 14 weeks!
 - “e.g. What is your name? What is the worst bug you have ever encountered? Favourite PL? OS?”
 - Will your classmate(s) and you be the next:
 - Jobs-Woz
 - Gates-Allen
 - Frances Allen
 - Turing-Church
 - Radhia and Patrick Cousot



Lecture 1 - Overview

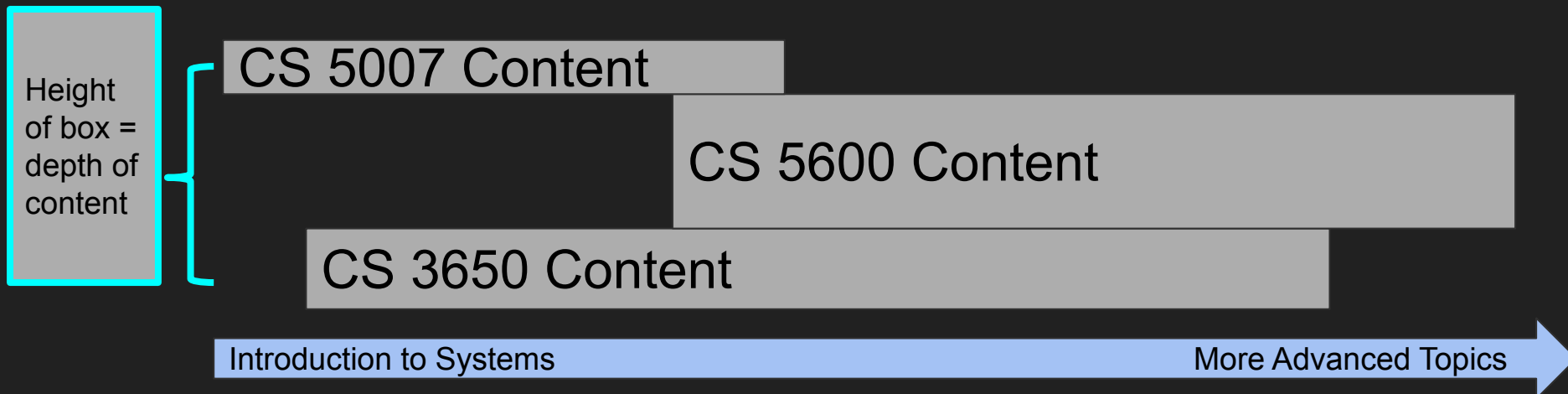
About your Instructor

- I grew up in (Czecho-)Slovakia
- Studied philosophy, worked as a web developer
- Then moved to Swansea, Wales, UK to study CS
- Wanted to study CS because of operating systems, but my undergrad OS class was a letdown...
- Did my PhD there in Programming Language Semantics and Implementation

So what is this course?

Computer Systems course in Computer Science

- A rough visualization of where the course is in the curriculum



Masters level course in Computer Science

- A rough

Our goal is to get everyone
through--not to be intimidated!

You will then be ready to take on
CS5600!

Height
of box =
depth of
content

In

Advanced Topics

Roughly Speaking this course has a few ‘modules’

1. Computer Systems Fundamentals
 - a. Terminal, C, Assembly, toolchain
2. Virtualization
 - a. Processes
3. Computer Architecture
 - a. Memory/Cache/Virtual Memory
4. Concurrency
 - a. Threads/Locks/Semaphores
 - b. Parallelism
5. Persistence
 - a. File Systems
 - b. Storage Devices
6. Other Selected Topics Throughout The Semester
 - a. Debugging/Instrumentation

Roughly Speaking this course has a few 'modules'

1. Computer Systems Fundamentals

- a. Terminal, C, Assembly, toolchain

2. Virtualization

- a. Processes

3. Computer Architecture

- a. Memory/Cache/etc

4. Concurrency


- a. Threads/Locks/Semaphores
- b. Parallelism

5. Persistence

- a. File Systems
- b. Storage Devices

6. Other Selected Topics

- a. Debugging/Instrumentation/Final



Note Operating Systems is the biggest chunk. Most things we do in the course you should view through the lens of an operating system.

Computer Systems = Magic?

- I hate to break it to you, but there is no magic in computers.
- Computers are just 1's and 0's In this course, we are going to look at 1's and 0's, and how to combine them to create different abstractions.
- That is where the magic comes in however—through the creativity and the art of computer science.
- Computer Science is an art!



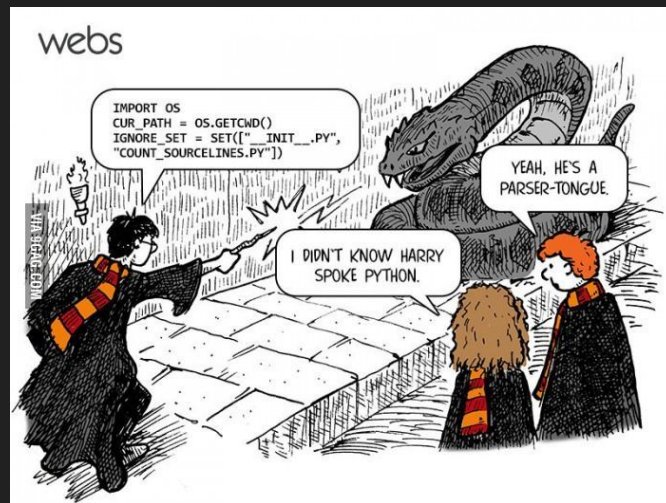
“No more magic”

- This is my mantra for all computer systems courses
- We do not have to look at machines any more and think there is magic going on.



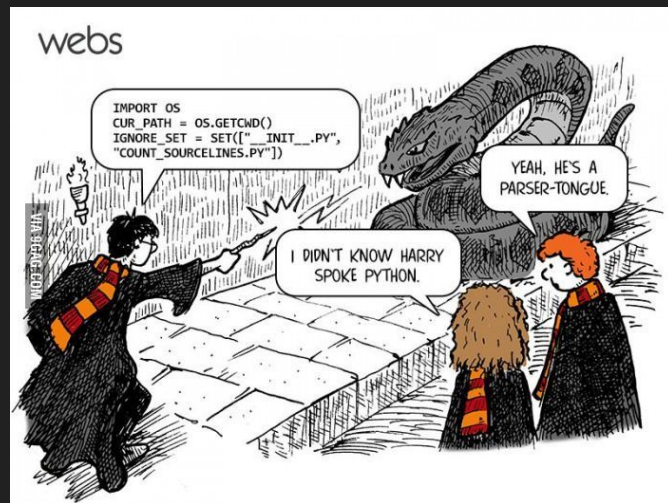
“No more magic”

- This is my mantra for all computer systems courses
- We do not have to look at machines any more and think there is magic going on.
- Someone programmed our operating systems, devices, and software
 - And they started off where you are!



“No more magic”

- On the other hand: a modern OS is a lot about creating and maintaining **illusions**



Course Goals

- Let us review the syllabus (which is on the website)
- <https://course.ccs.neu.edu/cs3650sp22/>



A note on assignments

- First 4-5 Assignments are individual
- Then, you get to pick a partner and can work in pairs (optional)
- You can partner up across sections



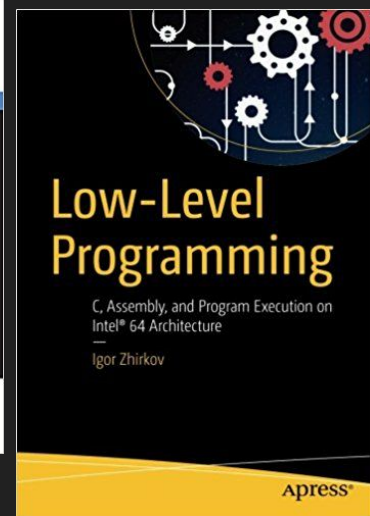
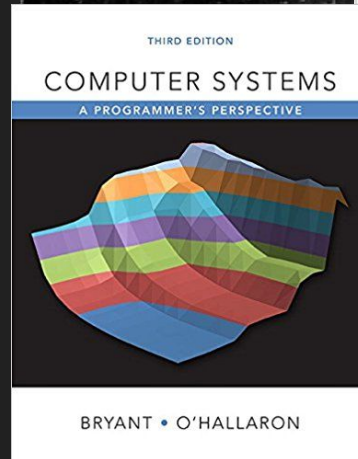
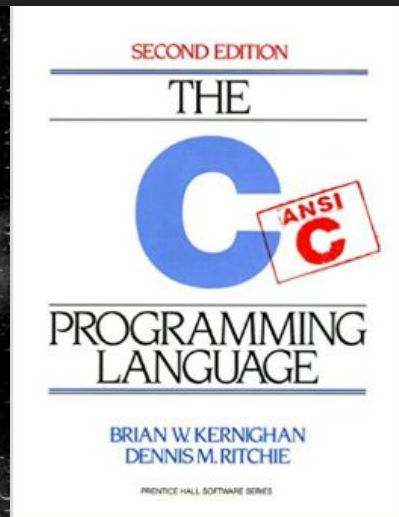
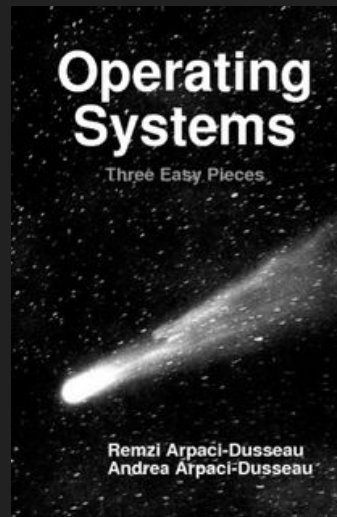
Course Materials

- A laptop is highly recommended
- I do not care what operating system you use on your computer
 - Mac, Linux (Ubuntu, Debian, etc.), Windows
 - In the case that you do not have a laptop, there are Khoury has VDI systems that are available
 - Reach out to me about labs, where we going to try to work together in class in parallel
- However, we will use a Linux system for much of the course



Course Text

- (free)
 - <http://pages.cs.wisc.edu/~remzi/OSTEP/>
 - <https://diveintosystems.org/>
 - <https://www.amazon.com/Low-Level-Programming-Assembly-Execution-Architecture/dp/1484224027>
- (Recommended)
 - C Programming Language Book
- (Recommended)
 - Computer Systems: A Programmer's Perspective
- Labs and lectures will have several web resources to check out!



Teaching Style

- Everyone learns differently--optimize as needed along the way
 - There will be lectures
 - Sometimes, there will be slides
 - In-class labs
- This is a very hands on class – we will build things!
- There will be plenty of opportunity to make mistakes
Do not be afraid to be wrong
 - The worst case scenario is we review
- Do ask questions!
 - Occasionally I may even pause to write down the question
 - I try to avoid randomly calling on students--but do participate!
- Come to office hours! Mine or the TAs or both!

Teaching Assistants

- Listed on the General tab on the webpage
 - Welcome them!
 - Currently 10 TAs
- TA Office Hours: tbd
 - Via Khoury Office Hours App

E-mail: try to avoid it

- Post general questions on Piazza to minimize e-mail
 - You should be registered here:
<https://piazza.com/northeastern/spring2022/cs3650>
- Come to office hours to minimize e-mail
- If all else fails, shoot me an e-mail
- ...and then remind me you've sent me an e-mail



Expectations

- You have taken some 'programming' related class.
 - Today you will notice I am calibrating a bit! :)
 - In the instance that you have not--you can still perform well.
 - i.e. Make sure you do the readings
- You know at least one programming language well
 - In this course we will use C and get exposed to x86-64 assembly
 - C is (still) the industry standard
 - (You can pick up whatever other fancy systems language later once you learn one)

Expectations

- You have taken some 'programming' related class.
 - Today you will notice I am calibrating a bit! :)
 - In the instance that you have not--you can still perform well.
 - i.e. Make sure you do the readings
- You know at least one programming language well
 - In this course we will use C and get exposed to x86-64 assembly
 - C is (still) the industry standard
 - (You can pick up whatever other fancy system language you want) Yes I know there is GO, Erlang, Rust, etc.

Why C?

Software Developer

SAVE



Intel Corporation
Hudson, MA

Apply on LinkedIn

Apply on Jobs Intel

Apply on Lensa.com

Apply on The Ladders


25 days ago


Full-time


Job Description

The Developer will work on design and implementation of low level software for a new architecture, developing and evaluating software technology in conjunction with work the underlying high-performance processor architecture. You will be a member of a fast-paced, multi-disciplinary software team working closely with processor core/system architects. The software team is responsible for developing the software stack - runtime support, compilers, base support for debuggers, profilers, etc. to enable applications to be built and run on the new system. The team will utilize their technology with external customer HPC workloads in the target environment through a co-design effort. This will enable the evaluation of workloads for an exascale system as design alternatives are being considered. The qualified candidate will have excellent knowledge of hardware architecture and software interaction, and parallel computing. Programming experience in C/C++ necessary. Good working knowledge of Linux. Good grasp of performance issues of large-scale HPC codes: synchronization, communication, load balance, memory access patterns. This is a hands-on software engineering position requiring the ability to work as a part of a cross-functional team in a rapidly evolving technical environment.

Why C?

**Developer Technology Engineer - Game Engineering**
AMD
San Diego, CA
via LinkedIn
11 days ago Full-time

**Developer Technology Engineer**
AMD
Bellevue, WA
via Jobs In XR
Over 1 month ago Full-time

**Engineering Program Lead, Diagnostics**
AMD
Austin, TX
via LinkedIn
Over 1 month ago Full-time

The AMD Game Engineering team works closely with external games software developers on the planet. We help them to fully exploit the technical capabilities of AMD's hardware and performance and we do everything in our power to make the user understand

The Successful Candidate Will

- Work with the external game development partners of AMD to enable them to produce high-quality games
- Optimize game and application performance for discrete GPUs, APUs and CPUs
- Design and implement rendering effects using established APIs
- Integrate features into game titles


The ideal candidate is a highly-skilled software designer and engineer, strong in 3D graphics, GPUs, APUs, and CPUs. You're team-driven and motivated to do things others might not

Minimum Requirements

- Has several years of experience efficiently creating C/C++ game code for Windows and Linux, understanding of C/C++ language features, standard libraries and writing easy to understand code.
- Has practical hands-on experience with DirectX-class development tools and technologies
- Has strong graphics code optimization skills, in particular shader code optimization
- Understands that requirements are rarely perfect and is willing to extract the spirit of the requirements
- Has a degree in computer science or a related technical discipline, or the equivalent
- Has excellent written and verbal skills.
- Is willing to travel domestically and internationally on a regular basis.

Why C? (You get the idea)

🕒 29 days ago 📁 Full-time



Software Development Engineer - Virtual Reality Software
Qualcomm
San Diego, CA
via Glassdoor

🕒 25 days ago 📁 Full-time

- Developing in embedded software environments
- C/C++ programming language
- Assembly programming" id="hdnMinimumQualifications">6 months to 7 years of academic or ind
- Developing in embedded software environments
- C/C++ programming language
- Assembly programming

Preferred Qualifications Debugging in embedded software environments

- Solid understanding of computer architecture and real-time operating systems
- Versatile attitude to learn new languages, architectures, and operating systems

Course Questions, Comments, Concerns?

So what exactly is C?

Here is what 'C' looks like

```
1 #include <stdio.h>
2
3 int main(){
4
5     puts("Hello Computer Systems!");
6
7     return 0;
8 }
```

Here is what 'C' looks like

compile with: `clang hello.c -o hello`

```
1 #include <stdio.h>
2
3 int main(){
4
5     puts("Hello Computer Systems!");
6
7     return 0;
8 }
```

Here is what 'C' looks like

compile with: `clang hello.c -o hello`

'clang' is the compiler

hello.c is the name of
our text source code
file

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

```
    ("Hello Computer Systems!");
```

```
    return 0;
```

```
8 }
```

Here is what 'C' looks like

compile with: `clang hello.c -o hello`

```
1 #i
```

And we are using a flag '-o' (dash lower-case Oh) which specifies the argument that follows is going to output a binary called hello.

```
4  
5  
6  
7  
8 }
```

```
ter Systems!");
```


Here is what 'C' looks like

compile with: `clang hello.c -o hello`

```
1 #include <stdio.h>
2
3 int main(){
4
5     puts("Hello Computer Systems!");
6
7     return 0;
8 }
```

`#include` brings in a library of commands related to standard input and output (so we can print text to the screen)

Here is what 'C' looks like

compile with: `clang hello.c -o hello`

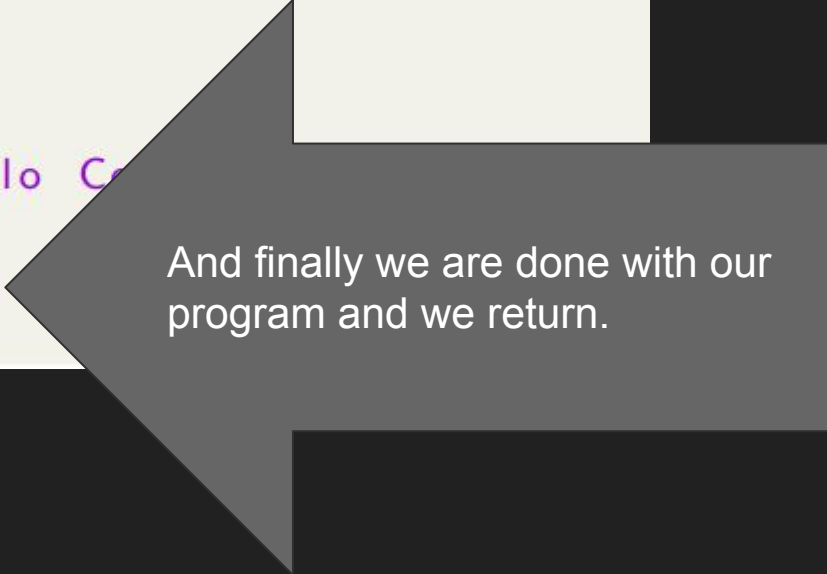
```
1 #include <stdio.h>
2
3 int main(){
4
5     puts("Hello Computer");
6
7     return 0;
8 }
```

`#puts` prints something to the screen. *printf* will be another popular way to do this.

Here is what 'C' looks like

compile with: `clang hello.c -o hello`

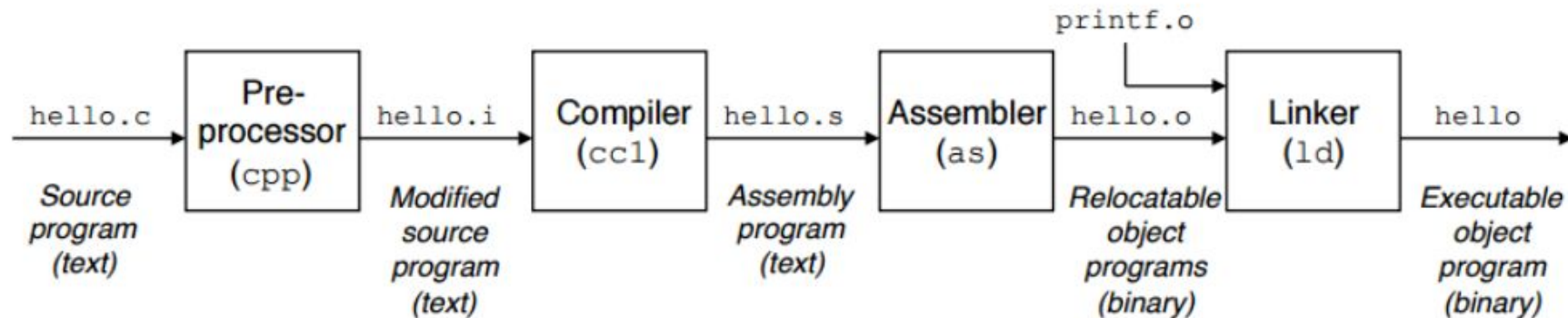
```
1 #include <stdio.h>
2
3 int main(){
4
5     puts("Hello C");
6
7     return 0;
8 }
```



And finally we are done with our program and we return.

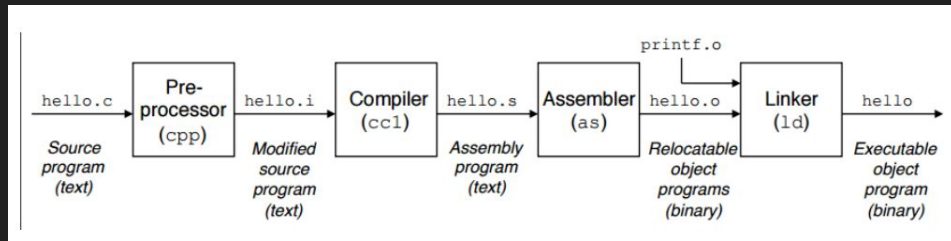
C and the compilation process

- In a picture, this is the compilation process from start to finish
- (Note in this class we'll use clang, but gcc is also fine)



Little exercise to see what compiler is doing

- Generate assembly code
 - `clang -S hello.c`
- Investigate assembly
- Compile assembly to executable file
 - `clang hello.s -o hello`
- Generate Object file
 - `clang -c hello.s`
- View Object File
 - `nl hello.o` (unreadable)
- Investigate Object File
 - `objdump -d hello.o` (disassembly)
 - `objdump -t hello.o` (symbol table)



Quick view of the assembly

- How many folks have **not** written assembly before?

Raise hands on Zoom or in classroom

```
indigo.ccs.neu.edu - PuTTY
1      .file      "hello.c"
2      .text
3      .globl  main
4      .align   16, 0x90
5      .type    main,@function
6 main:
7      .cfi_startproc
8 # BB#0:
9      pushq   %rbp
10     .Ltmp2:
11     .cfi_def_cfa_offset 16
12     .Ltmp3:
13     .cfi_offset %rbp, -16
14     movq    %rsp, %rbp
15     .Ltmp4:
16     .cfi_def_cfa_register %rbp
17     subq    $16, %rsp
18     leaq    .L.str, %rdi
19     movl    $0, -4(%rbp)
20     callq   puts
21     movl    $0, %ecx
22     movl    %eax, -8(%rbp)      # 4-byte Spill
23     movl    %ecx, %eax
24     addq    $16, %rsp
25     popq    %rbp
26     ret
27     .Ltmp5:
28     .size    main, .Ltmp5-main
29     .cfi_endproc
30
31     .type    .L.str,@object      # @.str
32     .section .rodata.str1.1,"aMS",@progbits,1
33 .L.str:
34     .asciz   "Hello Computer Systems!"
35     .size    .L.str, 24
36
37
38     .ident   "clang version 3.4.2 (tags/RELEASE_34/dot2-final)"
39     section  ".note.GNU-stack","",@progbits
```

Quick view of the assembly

- How many folks have **not** written assembly before?

It's not too bad, you can pull out various functions to orient yourself

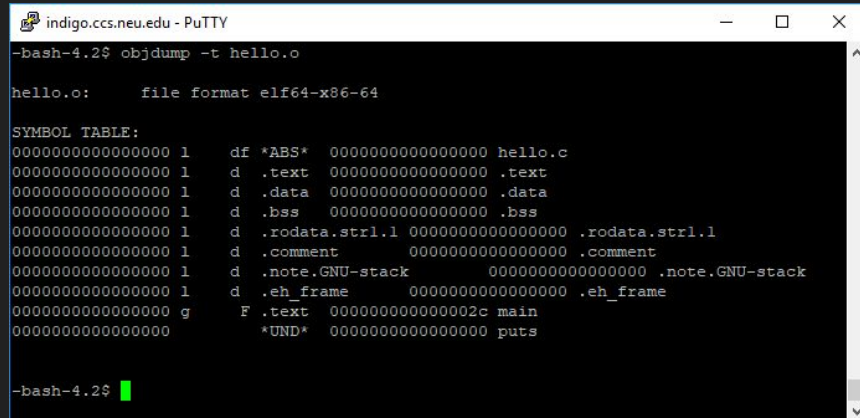
Our string

```
indigo.ccs.neu.edu - PuTTY
1      .file      "hello.c"
2      .text
3      .globl  main
4      .align   16, 0x90
5      .type    main,@function
6 main:
7      .cfi_startproc
8 # BB#0:
9      pushq   %rbp
10     .Ltmp2:
11     .cfi_def_cfa_offset 16
12     .Ltmp3:
13     .cfi_offset %rbp, -16
14     movq    %rsp, %rbp
15     .Ltmp4:
16     .cfi_def_cfa_register %rbp
17     subq    $16, %rsp
18     leaq    .L.str, %rdi
19     movl    0, %ecx
20     callq   puts
21     movl    0, %ecx
22     movl    %eax, -8(%rbp)      # 4-byte Spill
23     movl    %ecx, %eax
24     addq    $16, %rsp
25     popq    %rbp
26     ret
27     .Ltmp5:
28     .size    main, .Ltmp5-main
29     .cfi_endproc
30
31     .type    .L.str,@object      # @.str
32     .section  .rodata.str1.1,"aMS",@progbits,1
33     .L.str:
34     .asciz   "Hello Computer Systems!"
35     .size    .L.str, 31
36
37     .ident   "clang version 3.4.2 (tags/RELEASE_34/dot2-final)"
38     .section  ".note.GNU-stack","",@progbits
39
```

Quick view of objdump

- How many folks have **not** used objdump before?

Raise hands again...



```
indigo.ccs.neu.edu - PuTTY
-bash-4.2$ objdump -t hello.o

hello.o:      file format elf64-x86-64

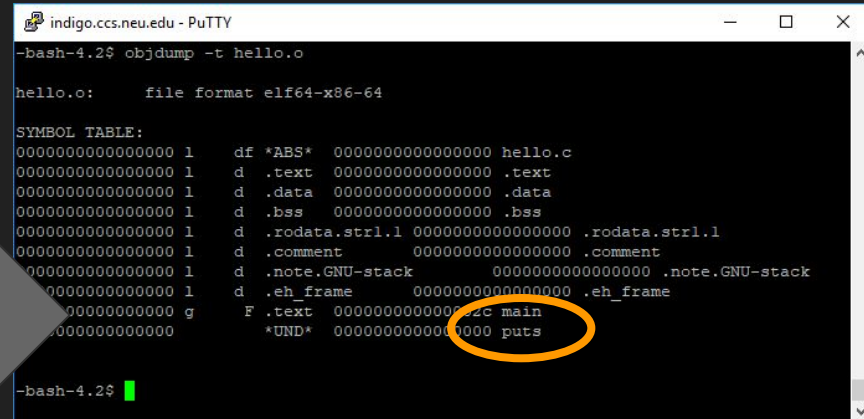
SYMBOL TABLE:
0000000000000000 1      df *ABS*  0000000000000000 hello.c
0000000000000000 1      d  .text  0000000000000000 .text
0000000000000000 1      d  .data  0000000000000000 .data
0000000000000000 1      d  .bss   0000000000000000 .bss
0000000000000000 1      d  .rodata.strl.1 0000000000000000 .rodata.strl.1
0000000000000000 1      d  .comment 0000000000000000 .comment
0000000000000000 1      d  .note.GNU-stack 0000000000000000 .note.GNU-stack
0000000000000000 1      d  .eh_frame 0000000000000000 .eh_frame
0000000000000000 g      F  .text  000000000000002c main
0000000000000000      *UND*  0000000000000000 puts

-bash-4.2$
```


Quick view of objdump

- How many folks have **not** used objdump before?

Powerful tool to pull out some information
(Can see functions/libraries used)



```
indigo.ccs.neu.edu - PuTTY
-bash-4.2$ objdump -t hello.o

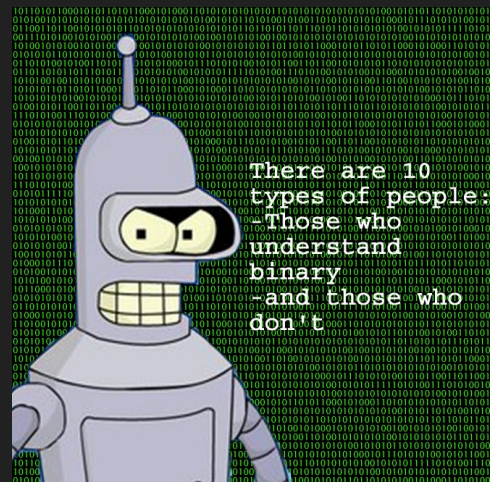
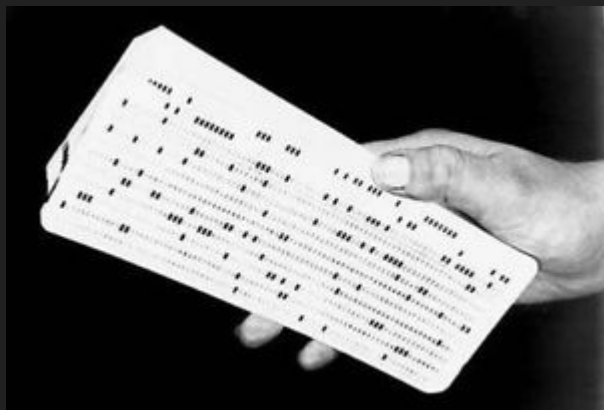
hello.o:      file format elf64-x86-64

SYMBOL TABLE:
0000000000000000 1      df *ABS* 0000000000000000 hello.c
0000000000000000 1      d  .text 0000000000000000 .text
0000000000000000 1      d  .data 0000000000000000 .data
0000000000000000 1      d  .bss  0000000000000000 .bss
0000000000000000 1      d  .rodata.str1.1 0000000000000000 .rodata.str1.1
0000000000000000 1      d  .comment 0000000000000000 .comment
0000000000000000 1      d  .note.GNU-stack 0000000000000000 .note.GNU-stack
0000000000000000 1      d  .eh_frame 0000000000000000 .eh_frame
0000000000000000 g      F .text 0000000000000000 _ZC main
0000000000000000      *UND* 0000000000000000 puts

-bash-4.2$
```

So Compilers are pretty neat

- When we start looking at some of the information taken in, we appreciate the job they do.
 - i.e. transform high level language to binary
- All of a sudden, writing some C code is not so bad!
 - (And it of course is better than pure binary!)



<http://www.learn-c.org/>

- Part of your first assignment will be performing some C Programming exercises.
- Here you will run examples on the web through some nice interactive tutorials
 - (We will revisit C from the command line shortly)

Code

▶ Run

Reset

Solution

Output

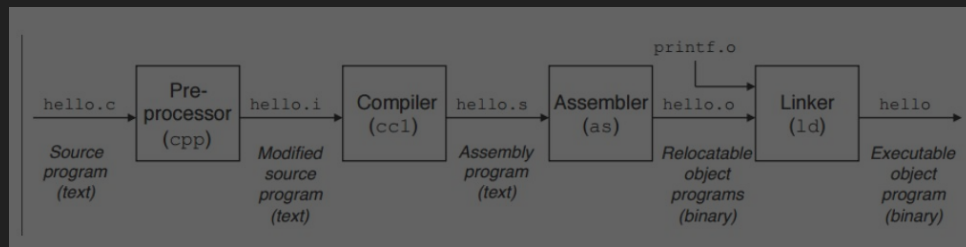
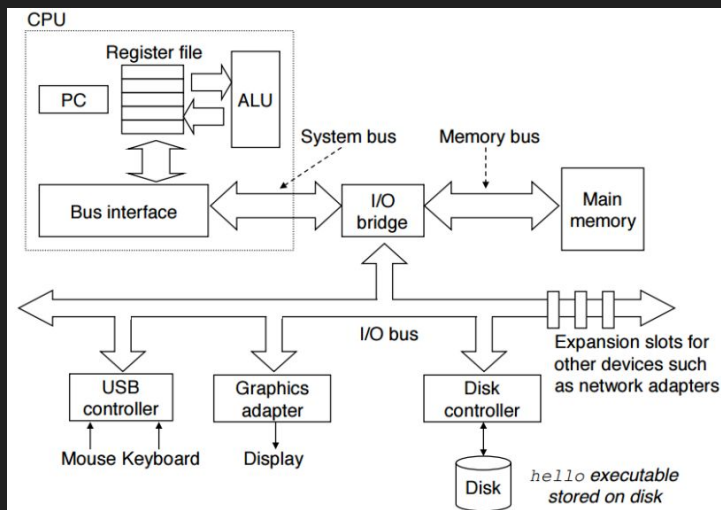
Expected Output

Minimize Window

```
1 /* Welcome to the Interactive C Tutorial.
2 Start by choosing a chapter and
3 write your code in this window. */
4
5 #include <stdio.h>
6
7 int main() {
8     printf("Hello, World!");
9     return 0;
10 }
11
```

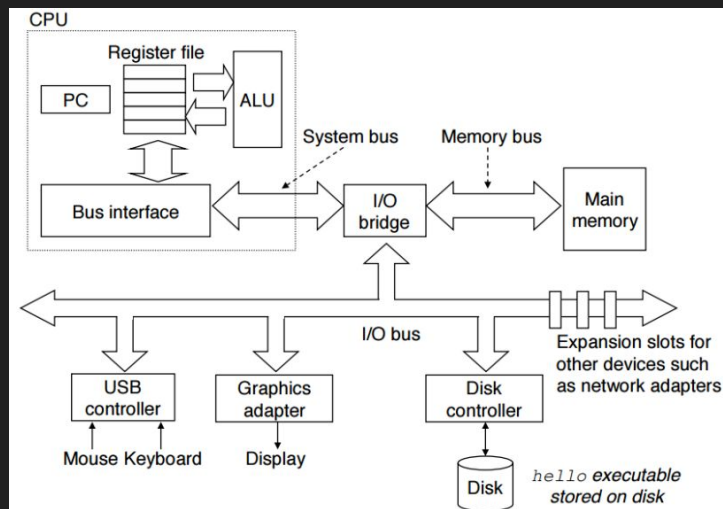
So compilers are a core element of this class

- The other core pieces are the hardware(left) and operating system (right)



So compilers are a core of this class

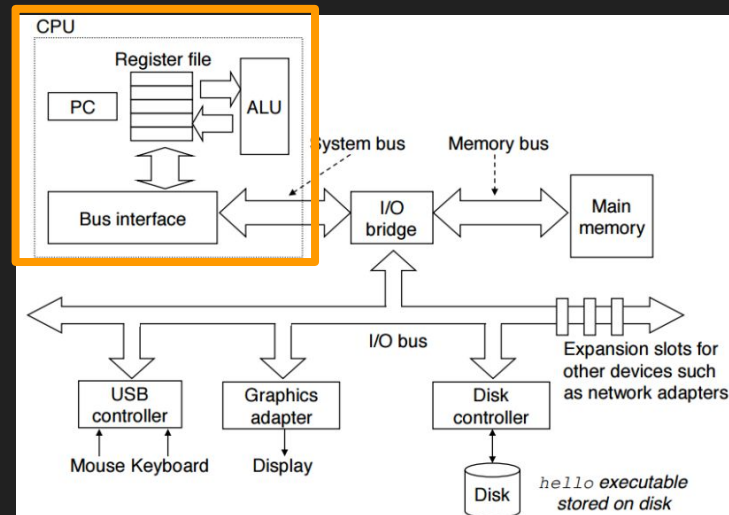
- The other core pieces are the hardware(left) and operating system (right)



Let's take a few minutes to think about the hardware

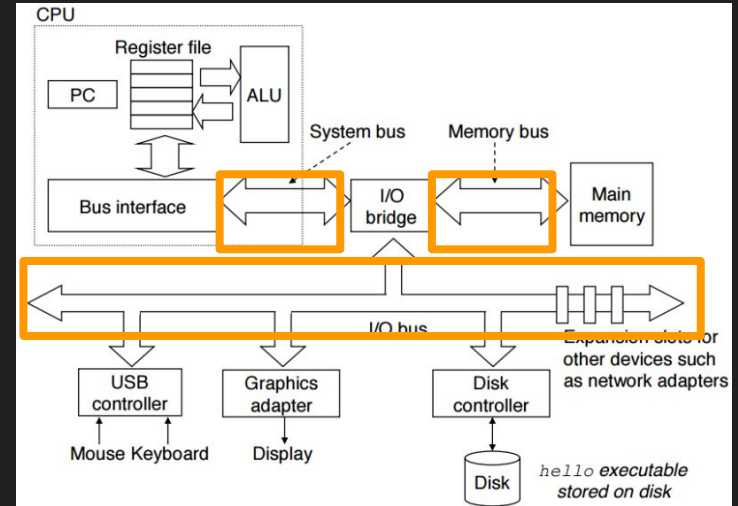
Modern Hardware Visual Abstraction

- The “brain” of modern hardware is the CPU
 - That’s where 1 instruction is executed at a time
 - Only 1!
 - (Note: Modern computers have multiple cores)
- We generally measure the speed at which a CPU executes in Megahertz or Gigahertz
 - This is a metric for how ‘fast’ a CPU performs, and how complex of software can be run.



Modern Hardware Visual Abstraction

- Beyond the CPU, a number of devices may also be connected.
- Buses transfer information from devices and memory into the CPU.
- There is a lot going on, and this needs to be managed
- Note: Busses can be thought of as simple networks, with many things hardcoded

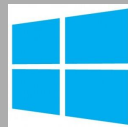


So compilers are a core of this class

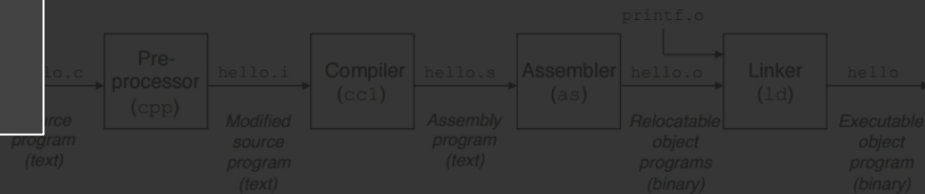
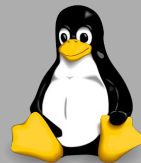
- The other core pieces are the hardware(left) and operating system (right)

Let's take a moment to think about operating systems

macOS



iOS



What is an Operating System?

Open question?

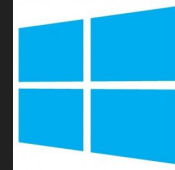
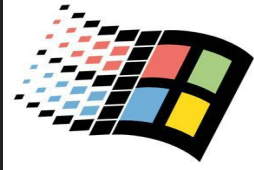
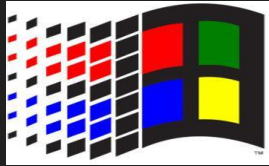
What is an Operating System?

Open question?

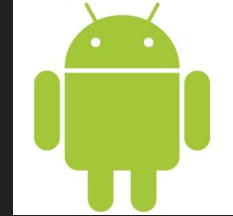
Because typically when I boot up a machine, I see windows/Linux/Mac booting up.

Many Different OSes

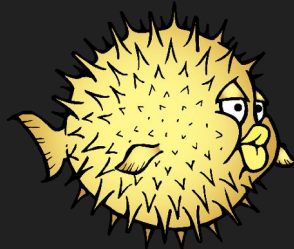
Windows



Linux



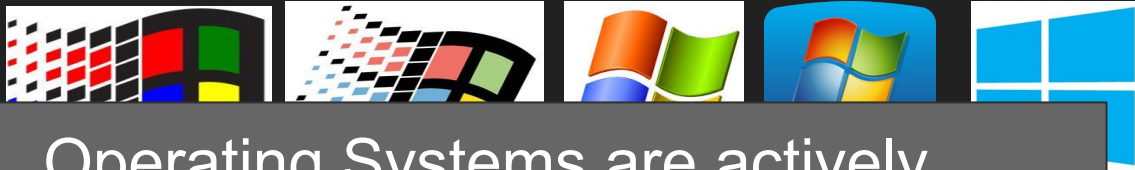
BSD



iOS

Many Different OSes

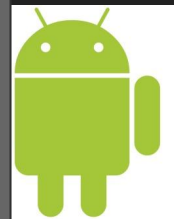
Windows



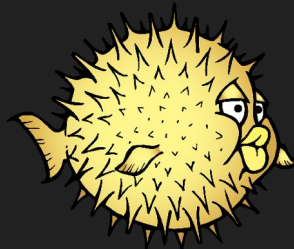
Operating Systems are actively developed! (read: co-ops/jobs)

Linux

You can actively contribute to the open source ones now!



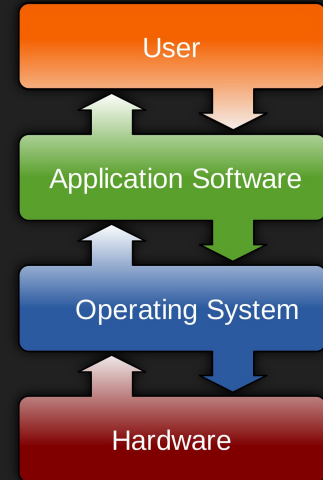
BSD



iOS

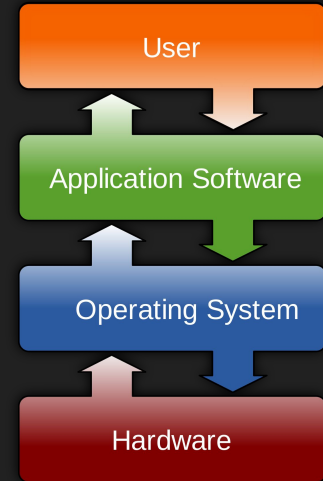
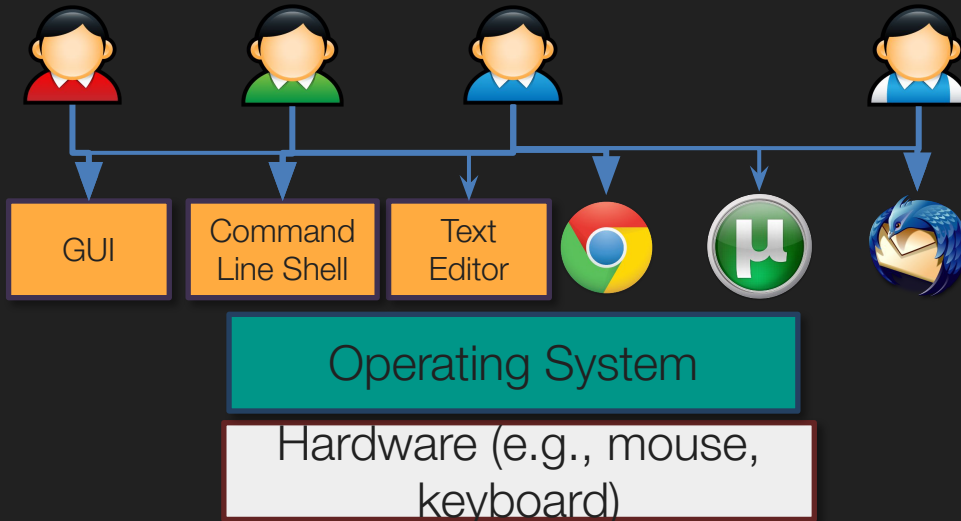
What is an Operating System?

- An OS is any and all software that sits between a user program and the hardware
- OS is a resource manager and allocator
 - Decides between conflicting requests for hardware access
 - Attempts to be efficient and fair
- OS is a control program
 - Controls execution of user programs
 - Prevents errors and improper use



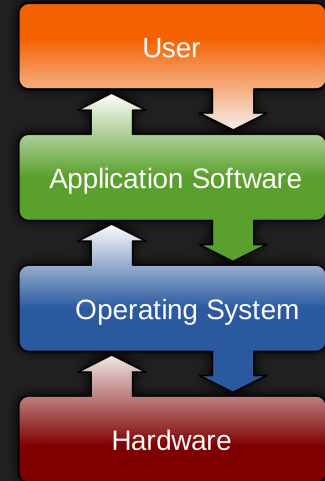
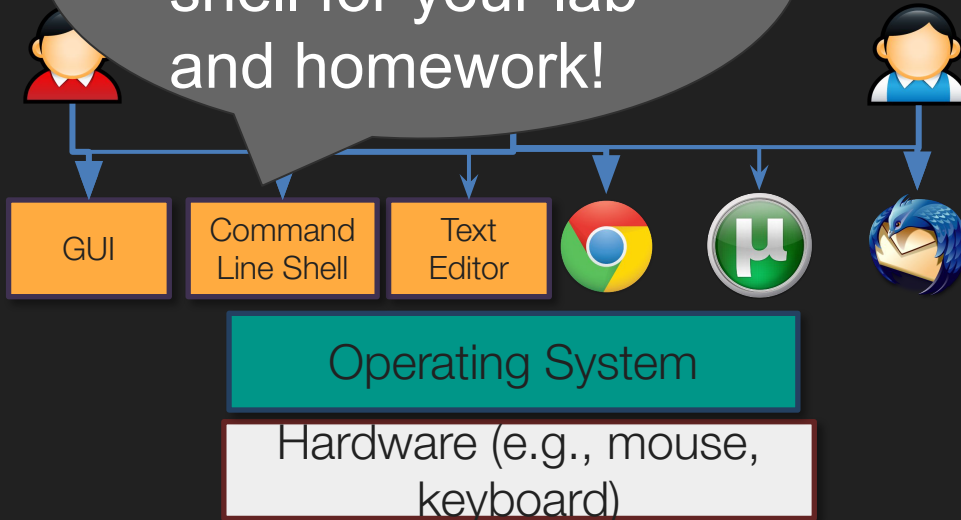
What is an Operating System?

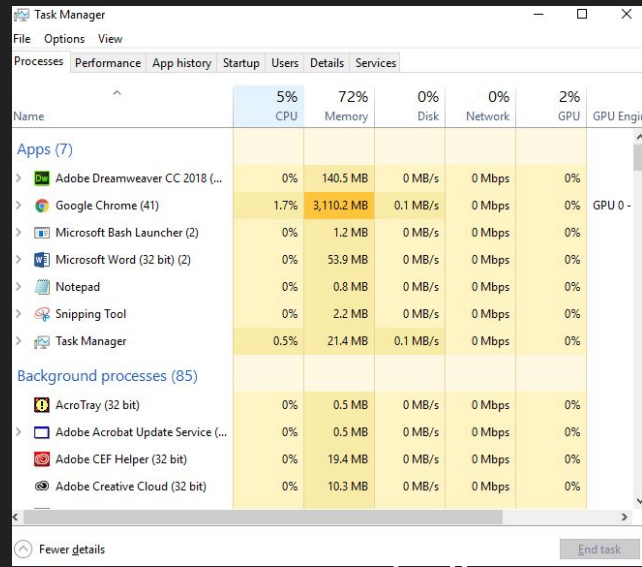
- An OS is any and all software that sits between a user program and the hardware



What is an Operating System?

- An OS is the interface between a user and the computer hardware. Shortly you will be working in the shell for your lab and homework!

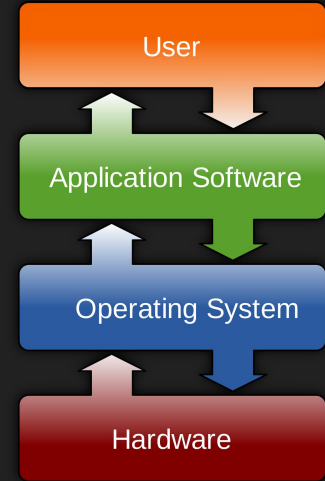




The screenshot shows the Windows Task Manager Performance tab. At the top, it displays overall system usage: CPU at 5%, Memory at 72%, Disk at 0%, Network at 0%, and GPU at 2%. Below this, a table lists running applications and their resource usage. The table is divided into 'Apps (7)' and 'Background processes (85)'.

Name	CPU	Memory	Disk	Network	GPU	GPU Engine
Apps (7)						
Adobe Dreamweaver CC 2018 (...)	0%	140.5 MB	0 MB/s	0 Mbps	0%	
Google Chrome (41)	1.7%	3,110.2 MB	0.1 MB/s	0 Mbps	0%	GPU 0 -
Microsoft Bash Launcher (2)	0%	1.2 MB	0 MB/s	0 Mbps	0%	
Microsoft Word (32 bit) (2)	0%	53.9 MB	0 MB/s	0 Mbps	0%	
Notepad	0%	0.8 MB	0 MB/s	0 Mbps	0%	
Snipping Tool	0%	2.2 MB	0 MB/s	0 Mbps	0%	
Task Manager	0.5%	21.4 MB	0.1 MB/s	0 Mbps	0%	
Background processes (85)						
AcroTray (32 bit)	0%	0.5 MB	0 MB/s	0 Mbps	0%	
Adobe Acrobat Update Service (...)	0%	0.5 MB	0 MB/s	0 Mbps	0%	
Adobe CEF Helper (32 bit)	0%	19.4 MB	0 MB/s	0 Mbps	0%	
Adobe Creative Cloud (32 bit)	0%	10.3 MB	0 MB/s	0 Mbps	0%	

- OS is a resource manager and allocator
 - Decides between conflicting requests for hardware access
 - Attempts to be efficient and fair



Two Common OS Families

- POSIX
 - Anything Unix-ish
 - e.g. Linux, BSDs, Mac, Android, iOS, QNX
- Windows
 - Stuff shipped by Microsoft

Many other operating systems may exist specific to a domain (e.g. an operating system for a car, handheld gaming device, or smart refrigerator)

Two Common OS Families

- POSIX
 - Anything Unix-ish
 - e.g. Linux, BSDs, Mac, Android, iOS
- Windows
 - Stuff shipped by Microsoft

In this course, we will work in a POSIX Environment. Our Khoury machines are Unix based.

Many other operating systems may exist specific to a domain (e.g. an operating system for a car or handheld gaming device)

Unix/Linux



What is xv6?

A teaching operating system! (i.e. small version of Unix)

<https://pdos.csail.mit.edu/6.828/2012/xv6.html>

Xv6, a simple Unix-like teaching operating system

The latest version of xv6 is at: [xv6](#)

Introduction

Xv6 is a teaching operating system developed in the summer of 2006 for MIT's operating systems course, [6.828: Operating System Engineering](#).

History and Background

For many years, MIT had no operating systems course. In the fall of 2002, one was created to teach operating systems engineering. In the course students to multiple systems—V6 and Jos—helped develop a sense of the spectrum of operating system designs.

V6 presented pedagogic challenges from the start. Students doubted the relevance of an obsolete 30-year-old operating system written in an obscure language. In 2006, we had decided to replace V6 with a new operating system, xv6, modeled on V6 but written in ANSI C and running on multiprocessor Intel processors (instead of using special-case solutions for uniprocessors such as enabling/disabling interrupts) and helps relevance. Finally, writing a new

A ~~teaching~~ small & manageable operating system!

<https://pdos.csail.mit.edu/6.828/2012/xv6.html>

Xv6, a simple Unix-like teaching operating system

The latest version of xv6 is at: [xv6](#)

Introduction

Xv6 is a teaching operating system developed in the summer of 2006 for MIT's operating systems course, [6.828: Operating System Engineering](#).

History and Background

For many years, MIT had no operating systems course. In the fall of 2002, one was created to teach operating systems engineering. In the course students to multiple systems—V6 and Jos—helped develop a sense of the spectrum of operating system designs.

V6 presented pedagogic challenges from the start. Students doubted the relevance of an obsolete 30-year-old operating system written in an obscure language. In 2006, we had decided to replace V6 with a new operating system, xv6, modeled on V6 but written in ANSI C and running on multiprocessor Intel processors (instead of using special-case solutions for uniprocessors such as enabling/disabling interrupts) and helps relevance. Finally, writing a new

xv6

- We will be using xv6 to build and implement some Operating Systems features
- This will give you experience adding features to a large piece of software.

Who, what, why, Linux? <https://www.linuxfoundation.org/>



- Linux is a family of free open source operating systems
 - That means the code is freely available, and you can contribute to the project!
- It was created by [Linus Torvalds](#)
 - Variants of Linux are: Ubuntu, Debian, Fedora, Gentoo Linux, Arch Linux, CentOS, ...
 - They all operate under *roughly* the same core code, which is called the kernel.
 - Often they differ by the software, user interface, and configuration settings.
 - So very often Linux software for one flavor of Linux will run on the other with few or no changes.
- Generally we (as systems programmers) like Linux, because it is a clean and hackable operating system.
- When many folks think of Unix-like operating systems, they may think of a hacker using a 'command-line interface' to program.

Over 30 years ago...

On Monday, August 26, 1991 at 2:12:08 AM UTC-4, Linus Benedict Torvalds wrote:

```
> Hello everybody out there using minix -  
>  
> I'm doing a (free) operating system (just a hobby, won't be big and  
> professional like gnu) for 386(486) AT clones. This has been brewing  
> since april, and is starting to get ready. I'd like any feedback on  
> things people like/dislike in minix, as my OS resembles it somewhat  
> (same physical layout of the file-system (due to practical reasons)  
> among other things).  
>  
> I've currently ported bash(1.08) and gcc(1.40), and things seem to work.  
> This implies that I'll get something practical within a few months, and  
> I'd like to know what features most people would want. Any suggestions  
> are welcome, but I won't promise I'll implement them :-)  
>  
>      Linus (torv...@kruuna.helsinki.fi)  
>  
> PS. Yes - it's free of any minix code, and it has a multi-threaded fs.  
> It is NOT protable (uses 386 task switching etc), and it probably never  
> will support anything other than AT-harddisks, as that's all I have :-).
```

Over 30 years ago...

On Monday, August 26, 1991 at 2:12:08 AM UTC-4, Linus Benedict Torvalds wrote:

> Hello everybody out there using minix -

>

> I'm doing a (free) operating system (**just a hobby, won't be big and**

> **professional like gnu**) for 386(486) AT clones. This has been brewing

> since april, and is starting to

> things people like/dislike in

> (same physical layout of the f

> among other things).

>

> I've currently ported bash(1.0

> This implies that I'll get something practical in a few months, and

> I'd like to know what features most people want. **Any suggestions**

> **are welcome, but I won't promise I'll implement them :-)**

>

> **Linus** (torvalds@kruuna.sinkki.fi)

>

> PS. Yes - it's free of any minix code, and it has a multi-threaded fs.

> **It is NOT portable (uses 386 task switching etc), and it probably never**

> **will support anything other than AT-harddisks, as that's all I have :-).**

Linux platforms: Alpha, ARC, ARM, ARM64, Apple M1 C6x, H8/300, Hexagon, Itanium, m68k, Microblaze, MIPS, NDS32, Nios II, OpenRISC, PA-RISC, PowerPC, RISC-V, s390, SuperH, SPARC, Unicore32, x86, x86-64, XBurst, Xtensa

The command line interface

- The command line interface is at the highest level just another program.
- Linux and Mac have terminals built-in, and Windows as well (cmd and powershell).
- From it, we can type in the names of programs to perform work for us
- (Next slide for examples)

Starting MS-DOS...

C:\>_

```
[root@localhost ~]# ping -q fa.wikipedia.org
PING text.pmtpa.wikimedia.org (208.80.152.2) 56(84) bytes of data.
64
--- text.pmtpa.wikimedia.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 540.528/540.528/540.528/0.000 ms
[root@localhost ~]# pwd
/root
[root@localhost ~]# cd /var
[root@localhost var]# ls -la
total 72
dwxr-xr-x. 18 root root 4096 Jul 30 22:43 .
dwxr-xr-x. 23 root root 4096 Sep 14 20:42 ..
dwxr-xr-x. 2 root root 4096 May 14 00:15 account
dwxr-xr-x. 11 root root 4096 Jul 31 22:26 cache
dwxr-xr-x. 3 root root 4096 May 18 16:03 db
dwxr-xr-x. 3 root root 4096 May 18 16:03 empty
dwxr-xr-x. 2 root root 4096 May 18 16:03 games
dwxr-xr-x. 2 root gdm 4096 Jun 2 18:39 gdm
dwxr-xr-x. 38 root root 4096 May 18 16:03 lib
dwxr-xr-x. 2 root root 4096 May 18 16:03 local
lrwxrwxrwx. 1 root root 11 May 14 00:12 lock -> ../run/lock
dwxr-xr-x. 14 root root 4096 Sep 14 20:42 log
lrwxrwxrwx. 1 root root 10 Jul 30 22:43 mail -> spool/mail
dwxr-xr-x. 2 root root 4096 May 18 16:03 nix
dwxr-xr-x. 2 root root 4096 May 18 16:03 opt
dwxr-xr-x. 2 root root 4096 May 18 16:03 preserve
dwxr-xr-x. 2 root root 4096 Jul 1 22:11 report
lrwxrwxrwx. 1 root root 6 May 14 00:12 run -> ../run
dwxr-xr-x. 14 root root 4096 May 18 16:03 spool
dwxr-xr-x. 4 root root 4096 Sep 12 23:50 tmp
dwxr-xr-x. 2 root root 4096 May 18 16:03 yp
[root@localhost var]# yum search wiki
Loaded plugins: langpacks, presto, refresh-packagekit, remove-with-leaves
rpmfusion-free-updates                                | 2.7 kB    00:00
rpmfusion-free-updates/primary_db                     | 296 kB    00:04
rpmfusion-nonfree-updates                             | 2.7 kB    00:00
updates/metalink                                      | 5.9 kB    00:00
updates                                                | 4.7 kB    00:00
updates/primary_db                                     73% [=====] | 62 kB/s | 2.6 MB 00:15 ETA
```

Why the command line?

- “I love GUI interfaces, so simple and sleek looking”
- Well, I will argue the command line is a lot faster than moving your mouse
- It is also very convenient for ‘scripting’ behavior that you could not so easily do in a GUI environment.
 - Executing a few commands in a row in a script is a piece of cake!
- And if you are working remotely, you often will not have any GUI environment at all!
 - (Often machines you need to access do not have a monitor attached)

Example shell script



mikeshah@DESKTOP-DDNGQVA: /mnt/c/Windows/System32

```
1 # Lines that start with a 'hashmark' or 'pound sign'
2 # are comments that are ignored.
3 # You should use them liberally!
4
5 # This line is special and tells us we have an executable script.
6 #!/bin/bash
7
8 # Output hello and two items read in as command-line arguments
9 echo "Hello $1 $2"
10 echo "What is your age?"
11 # Read in a value
12 read myAge
13 echo "That is great you are $myAge years old!"
```

Example shell script

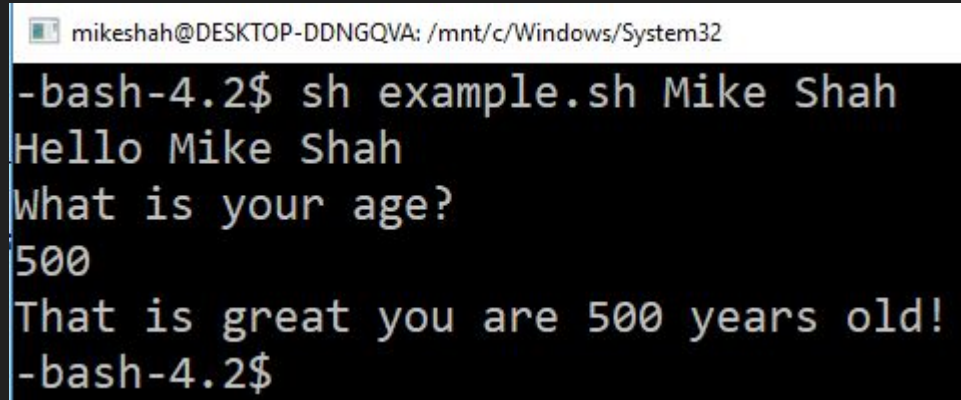
- I wrote this script in a text editor called 'vim'
- You will have to learn VIM (or emacs) in this course.
 - It's a great skill to have.

mikeshah@DESKTOP-DDNGQVA: /mnt/c/Windows/System32

```
1 # Lines that start with a 'hashmark' or 'pound sign'
2 # are comments that are ignored.
3 # You should use them liberally!
4
5 # This line is special and tells us we have an executable script.
6 #!/bin/bash
7
8 # Output hello and two items read in as command-line arguments
9 echo "Hello $1 $2"
10 echo "What is your age?"
11 # Read in a value
12 read myAge
13 echo "That is great you are $myAge years old!"
```

Example shell script Executing

- Note “Mike Shah” are the first and second arguments passed into this program

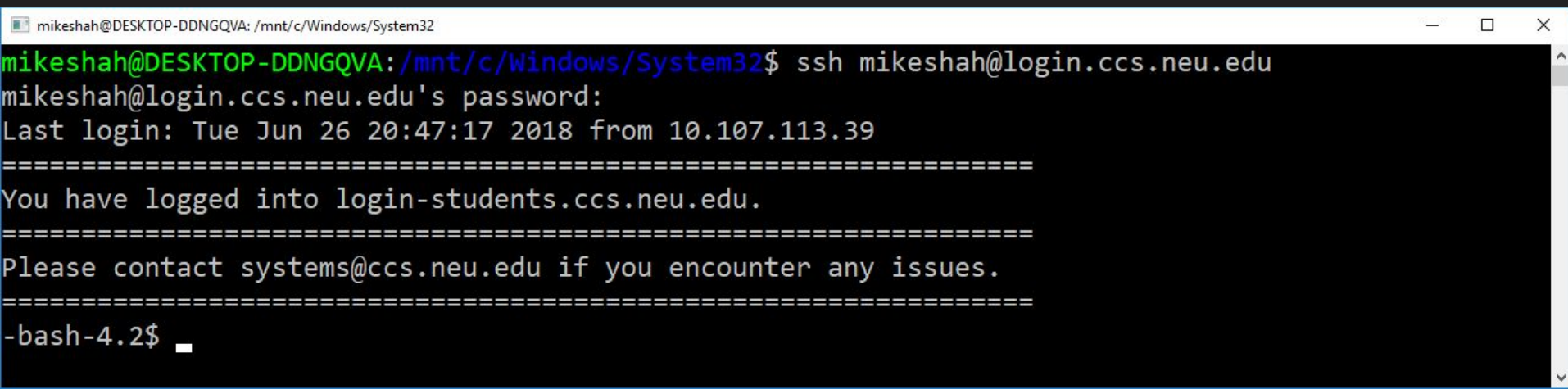
A terminal window with a black background and white text. The title bar at the top reads 'mikeshah@DESKTOP-DDNGQVA: /mnt/c/Windows/System32'. The terminal shows the command '-bash-4.2\$ sh example.sh Mike Shah' being entered. The script outputs 'Hello Mike Shah' and 'What is your age?'. The user enters '500'. The script then outputs 'That is great you are 500 years old!' and returns to the prompt '-bash-4.2\$'.

```
mikeshah@DESKTOP-DDNGQVA: /mnt/c/Windows/System32
-bash-4.2$ sh example.sh Mike Shah
Hello Mike Shah
What is your age?
500
That is great you are 500 years old!
-bash-4.2$
```

(Am I really 500 years old? Time flies when you are having fun!)

ssh - secure shell

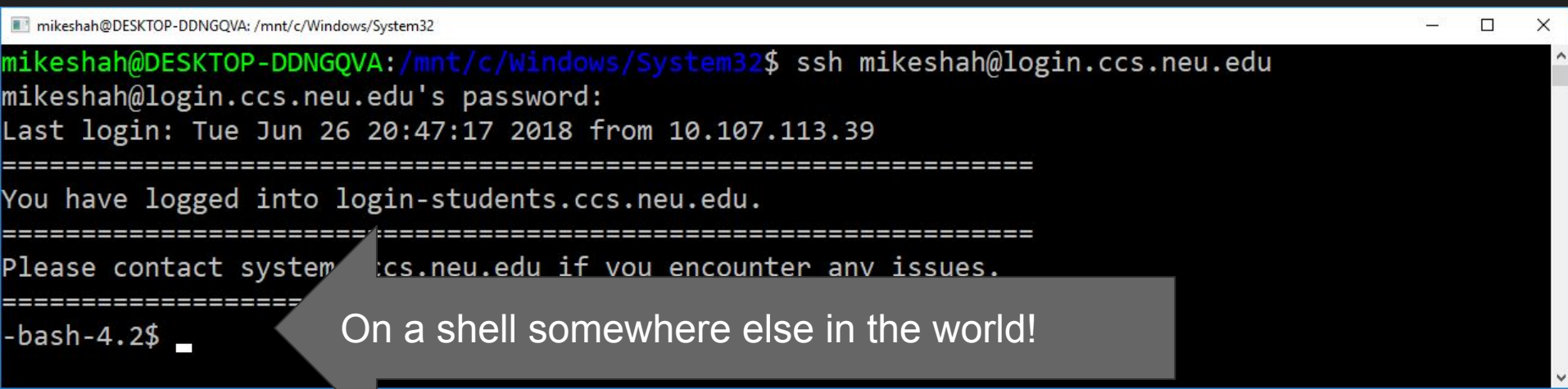
- Our tool for remote access--which we will do for all of our work!
- `ssh some_user_name@login.ccs.neu.edu`
- After typing in my password successfully, I am now executing commands on a machine somewhere on Northeastern's campus



```
mikeshah@DESKTOP-DDNGQVA: /mnt/c/Windows/System32
mikeshah@DESKTOP-DDNGQVA:/mnt/c/Windows/System32$ ssh mikeshah@login.ccs.neu.edu
mikeshah@login.ccs.neu.edu's password:
Last login: Tue Jun 26 20:47:17 2018 from 10.107.113.39
=====
You have logged into login-students.ccs.neu.edu.
=====
Please contact systems@ccs.neu.edu if you encounter any issues.
=====
-bash-4.2$
```


ssh - secure shell

- Our tool for remote access--which we will do for all of our work!
- `ssh some_user_name@login.ccs.neu.edu`
- After typing in my password successfully, I am now executing commands on a machine somewhere on Northeastern's campus



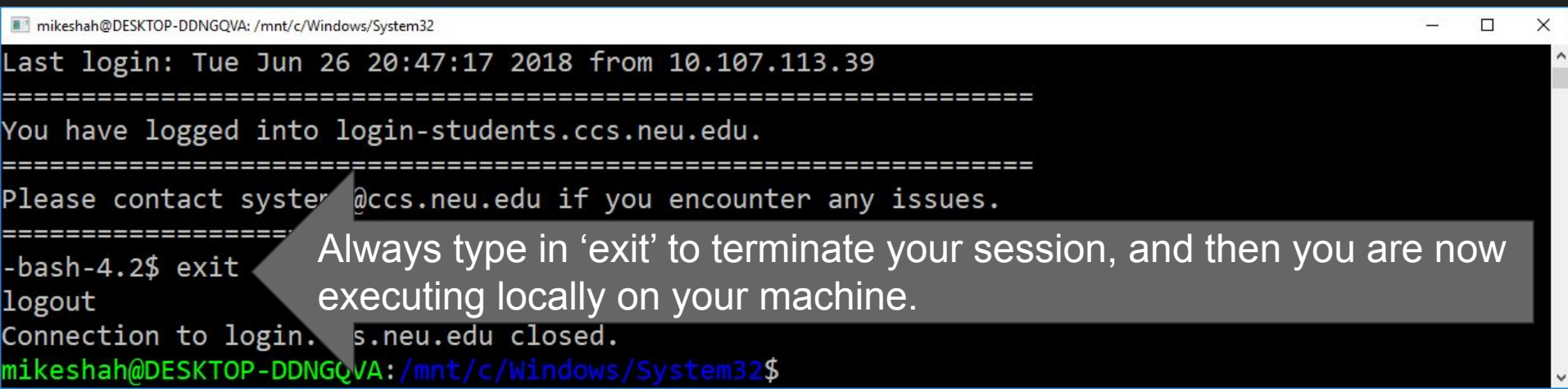
A terminal window titled "mikeshah@DESKTOP-DDNGQVA: /mnt/c/Windows/System32" showing an SSH session. The user enters the command `ssh mikeshah@login.ccs.neu.edu`. The terminal displays the password prompt, the last login time, and a welcome message. A large grey arrow points from the text "On a shell somewhere else in the world!" to the prompt `-bash-4.2$`.

```
mikeshah@DESKTOP-DDNGQVA: /mnt/c/Windows/System32$ ssh mikeshah@login.ccs.neu.edu
mikeshah@login.ccs.neu.edu's password:
Last login: Tue Jun 26 20:47:17 2018 from 10.107.113.39
=====
You have logged into login-students.ccs.neu.edu.
=====
Please contact system@login.ccs.neu.edu if you encounter any issues.
=====
-bash-4.2$
```

On a shell somewhere else in the world!

ssh - secure shell

- Our tool for remote access--which we will do for all of our work!
- `ssh some_user_name@login.ccs.neu.edu`
- After typing in my password successfully, I am now executing commands on a machine somewhere on Northeastern's campus



```
mikeshah@DESKTOP-DDNGQVA: /mnt/c/Windows/System32
Last login: Tue Jun 26 20:47:17 2018 from 10.107.113.39
=====
You have logged into login-students.ccs.neu.edu.
=====
Please contact system@ccs.neu.edu if you encounter any issues.
=====
-bash-4.2$ exit
logout
Connection to login-students.ccs.neu.edu closed.
mikeshah@DESKTOP-DDNGQVA: /mnt/c/Windows/System32$
```

Always type in 'exit' to terminate your session, and then you are now executing locally on your machine.

Feeling overwhelmed or forgetting a command?

- Luckily there are built-in 'manual pages'
- Called the 'man pages' for short.
- Simply type 'man command_name' for help
 - (Hit 'q' to quit the page when you are done)

```
mikeshah@DESKTOP-DDNGQVA: /mnt/c/Windows/System32
-bash-4.2$ man ls
```

```
mikeshah@DESKTOP-DDNGQVA: /mnt/c/Windows/System32
LS(1)      User Commands      LS(1)

NAME
    ls - list directory contents

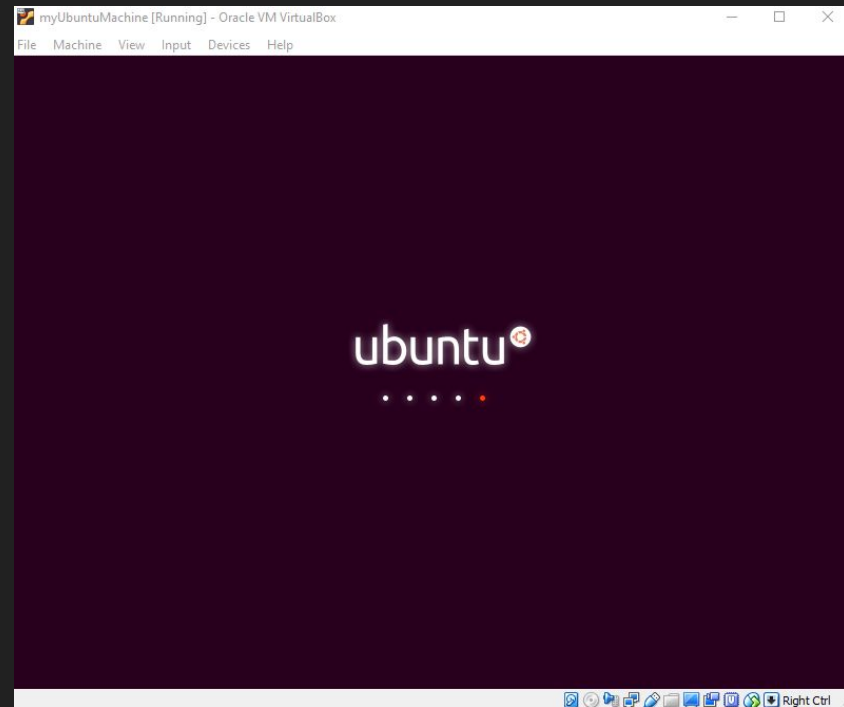
SYNOPSIS
    ls [OPTION]... [FILE]...

DESCRIPTION
    List information about the
    FILES (the current directory
    by default).  Sort entries
    alphabetically if none of
    -cftuvSUX nor --sort is
    specified.

    Mandatory arguments to long
    options are mandatory for
    short options too.
```

SSH and Virtual Machine

- Part of your upcoming labs/assignments will involve setting up a Linux environment on your desktop.
- Another part will involve working in a remote linux environment through ssh.
 - ssh is a way to remotely access a machine somewhere else in the world through a command-line terminal



Lab (Logistics)

- Motivation: Practice with tools and techniques useful for an upcoming assignment
- Typically on Fridays, possibly combined with a shorter lecture
- Submitted individually, but you can pair up with your neighbor if classroom layout permits
- I (& TA if available) will walk around and help folks
- When you are finished you may leave or work on extending the lab further.
- The intent is that labs take the duration allocated in class, but maybe an additional 1-2 hours.
- The lab is due the following week (See the very bottom “Deliverable” section), typically on Tuesdays
- More about this on Friday (first Lab)

In-Class Activities (Logistics)

- Typically a little quiz or some problems on lecture material
 - We want to know that you're paying attention :-)
 - Typically we will try to go over answers together (depending on the tool I use)
- Each is 1.5% of your grade

Join our GitHub Classroom

Go to <https://tinyurl.com/2p8zwytb>

You should see a screen similar to this.

Use command-F to open a search window
Search for your myNortheastern username
Click on it to register.

Only pick your myNortheastern username.

Join the classroom:

Khoury-CS3650Sp22

To join the GitHub Classroom for this course, please select yourself from the list below to associate your GitHub account with your school's identifier (i.e., your name, ID, or email).

Can't find your name? [Skip to the next step](#) →

Identifiers	
abbott.st	>
adams.ely	>
agatep.o	>
alessio.j	>
ali.ama	>
ali.she	>
an.co	>
angulo.an	>
astorina.a	>

Accepting a project

If you are successful, you should see a new page like the upper figure.

Click on “Accept this assignment”.

After accepting the assignment you may see a screen like the lower one.

Refresh the screen...

Khoury-CS3650Sp22

Accept the assignment — Git & Github Fundamentals

Once you accept this assignment, you will be granted access to the `git-github-fundamentals-awjacks` repository in the [Khoury-CS3650](#) organization on GitHub.

Accept this assignment



You accepted the assignment, **Git & Github Fundamentals** . We're configuring your repository now. This may take a few minutes to complete. Refresh this page to see updates.

Note: You may receive an email invitation to join [Khoury-CS3650](#) on your behalf. No further action is necessary.

Classroom repo

Click on the link to go to the repo...


You may need to use
Northeastern's
SSO to authenticate yourself ...



You're ready to go!

You accepted the assignment, **Git & Github Fundamentals**.

Your assignment repository has been created:

 <https://github.com/Khoury-CS3650/git-github-fundamentals-awjacks>

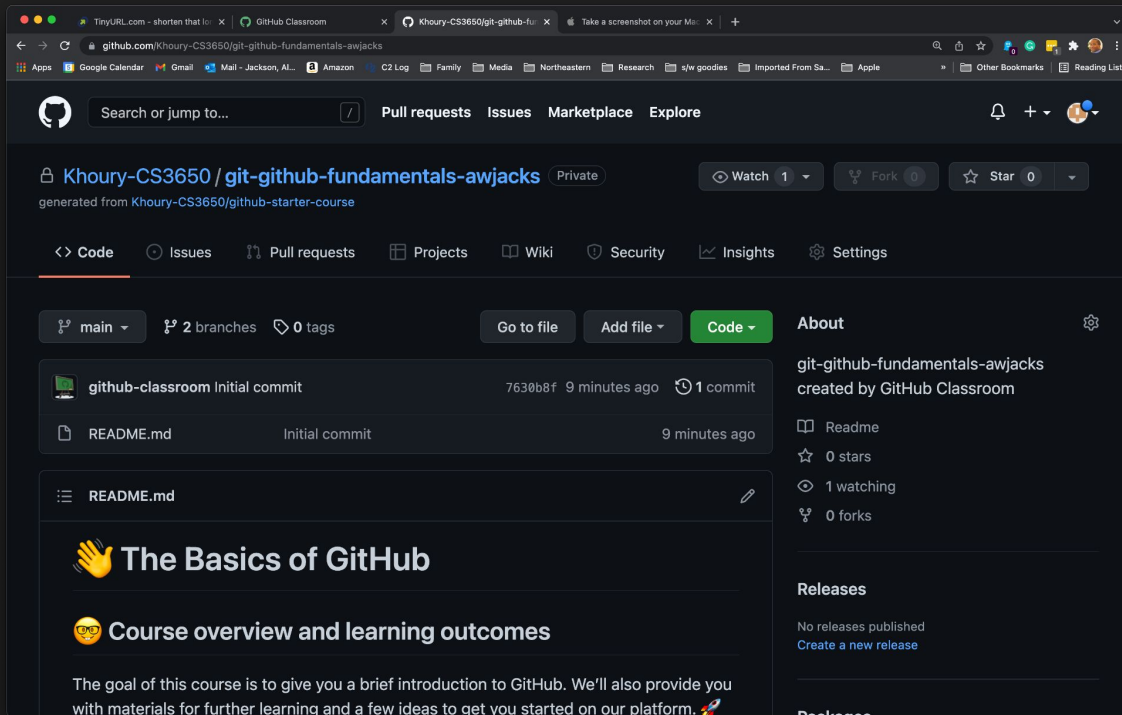
We've configured the repository associated with this assignment ([update](#)).

Note: You may receive an email invitation to join [Khoury-CS3650](#) on your behalf. No further action is necessary.

Git and GitHub review

Read the overview of GitHub basics.

If you are new to git or need a refresher, browse the resources at the bottom of the page.



This lecture in summary

- We are going to learn about computer systems
 - This includes software (e.g. compilers), hardware, and some operating system concepts.
- We are going to work in a Unix environment
 - This work will be performed on a command-line
 - In this course we can access a command-line either:
 - Through SSH or a Virtual Machine