CS 3650 Computer Systems – Summer 1 2025

Introduction to Computer Systems

Unit 1

Northeastern University

* Acknowledgements: created based on Christo Wilson, Ferdinand Vesely, and Alden Jackson's lecture slides for the same course.
Based on Ji-Yong Shin's slides

Course Instructor

- Dr. Ben Weintraub
- Email: b.weintraub@northeastern.edu

- I grew up in Chicago
- Undergrad at the University of Iowa
- Worked as a software engineer before my PhD
    - IBM
    - Silicon Valley startup
- PhD here at NU in the Cybersecurity and Privacy Institute
- Research on network security and blockchains
    - Measuring the prevalence of predatory trading
    - Detecting scammy smart contracts
    - Evaluating correctness of payment protocols
- I also coach the NU Women's Ultimate B team: The Valkyries

Who are you?

My Thoughts About Learning

- Screens impede learning in lecture settings
    - Distracting for yourself and others.
    - Paper notes recommended
- This course is designed for undergrads
    - There's nothing you shouldn't be able to figure out
- Half summer course move quickly
    - Plan your time deliberately
- Focus is the biggest challenge to most learners
    - When you work, hide your phone—it actually helps

AI Policy

1. Generative AI may be used to aid programming (unless assignment says otherwise), but may **not** be used for writing or documentation
2. Any use of Generative AI must come with attribution including the specific model, date/time of use, and exact prompts. Failure to do so will be considered plagiarism.
3. Generative AI should be used **not** to get assignments done as quickly as possible but rather to facilitate learning.

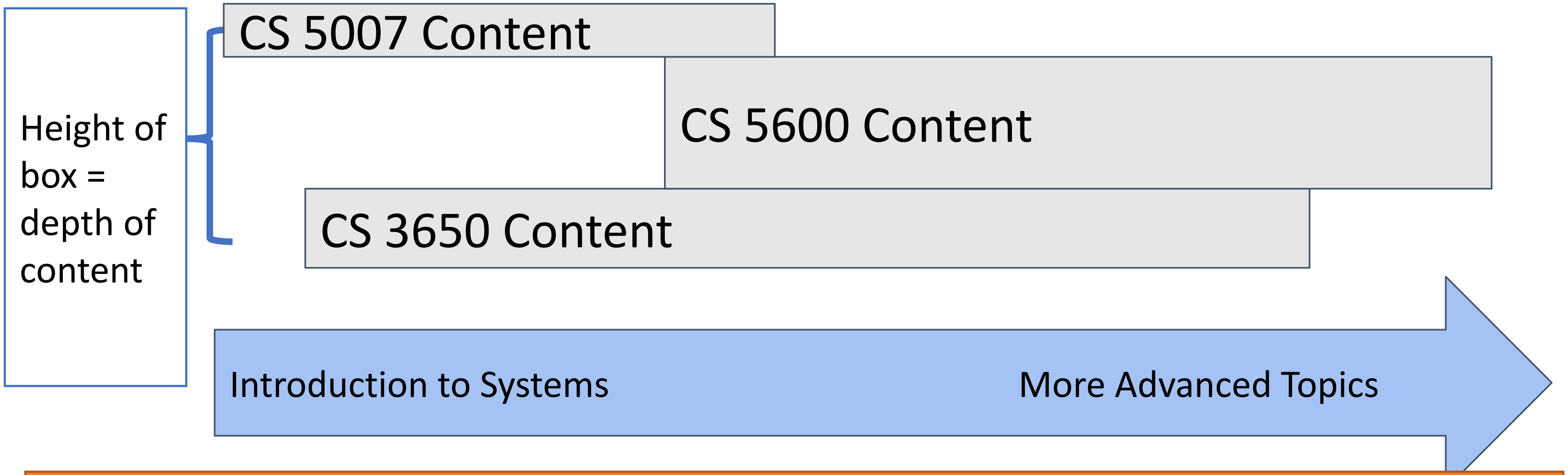
- AI has limitations including biases.
- It is not always right.

Recommendation: don't use AI for anything that will take longer to verify than if you were to just do it yourself.

What is this course about?

Computer Systems courses at Khoury College

- Three courses with the same name!
- A rough visualization of where the course is in the curriculum



My goal is to get everyone through & not be intimidated!
You will then be ready to take on CS5600!

Roughly Speaking this course has a few ‘modules’

- Computer Systems Fundamentals
  - Terminal, C, Assembly, and Compilers
- Virtualization
  - Processes
- Computer Architecture
  - Memory/Cache/etc
- Concurrency
  - Threads/Locks/Semaphores
  - Parallelism
- Persistence
  - File Systems
  - Storage Devices
- Other Selected Topics
  - 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.
- Someone programmed our operating systems, devices, and software
  - And they started off where you are!

```
IMPORT OS
CUR_PATH = OS.GETCWD()
IGNORE_SET = SET(["__INIT__.PY",
"COUNT_SOURCELINES.PY"])
```

webs
I DIDN'T KNOW HARRY SPOKE PYTHON.
YEAH, HE'S A PARSER-TONGUE.
VIA 9GAG.COM

Overview

- Lectures
  - Monday–Thursday 3:20 pm – 5:00 pm
- Office Hours
  - Thursday 1:00 pm – 3:00 pm (location TBD)
- Course website: https://ben-weintraub.com/classes/cs3650-summer25/
  - General Info
  - Lecture materials: notes but no slides
  - Assignments
- Assignment submission
  - Canvas > Gradescope
- Discussions and questions
  - Canvas > Piazza
  - See sign-up code on Canvas home page

Course Goals

- let us review the syllabus
  - https://ben-weintraub.com/classes/cs3650-summer25/

- All course related information is on the webpage

Course Materials

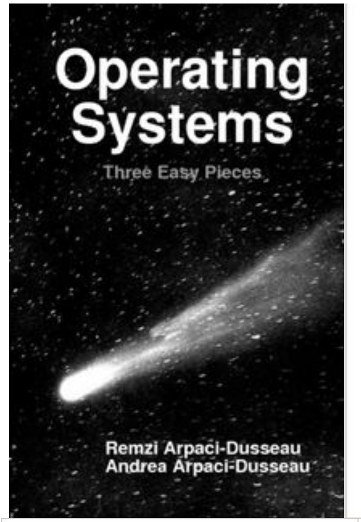
- A laptop is highly recommended

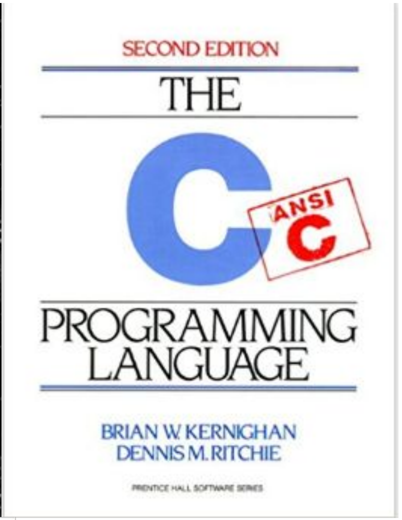
- I do not care what operating system you use on your computer
  - Mac (even with an M1) , Linux (Ubuntu, Debian, etc.), Windows

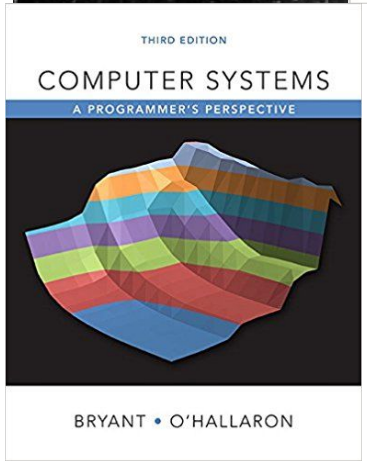
- We will have an online virtualized Linux environment set up for you

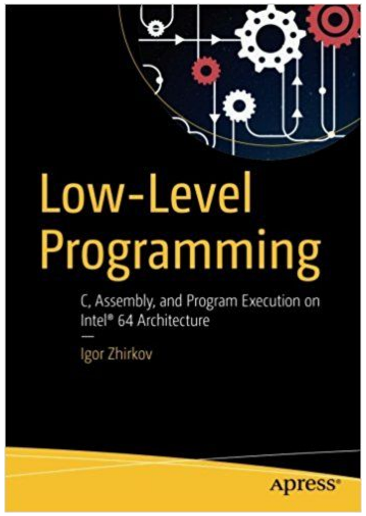
Course Text

- Free main textbooks
  - Dive into Systems
  - Operating Systems: Three Easy Pieces (aka OSTEP)
- Recommended
  - Low-Level Programming: C, Assembly, and Program Execution on Intel® 64 Architecture
  - C Programming Language Book
  - Computer Systems: A Programmer's Perspective
- Inspiration drawn from both of these texts.
- Labs and lectures will have several web resources to check out!








Low-Level Programming
C, Assembly, and Program Execution on Intel® 64 Architecture
Igor Zhirkov
Apress®

Teaching Style

- Learning from multiple sources is more effective
  - There will be lectures
  - Visuals on slides
  - Labs and assignments
- 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
- Please ask questions!
  - I try to avoid randomly calling on students--but do participate!
- Come to office hours!

E-mail: don’t use it!

- Post on Piazza general questions to minimize e-mail
    - If not already a member, register through the link on Canvas

- Come to office hours to minimize e-mail

How to ask questions

- Ask specific questions
  - My code doesn’t work/compile (bad)

- Proper question structure (good)
  - I did X
  - I expected Y to happen
  - Instead, Z happened
  - I tried A, B, and C
  - I here’s debug info from E, D, and F

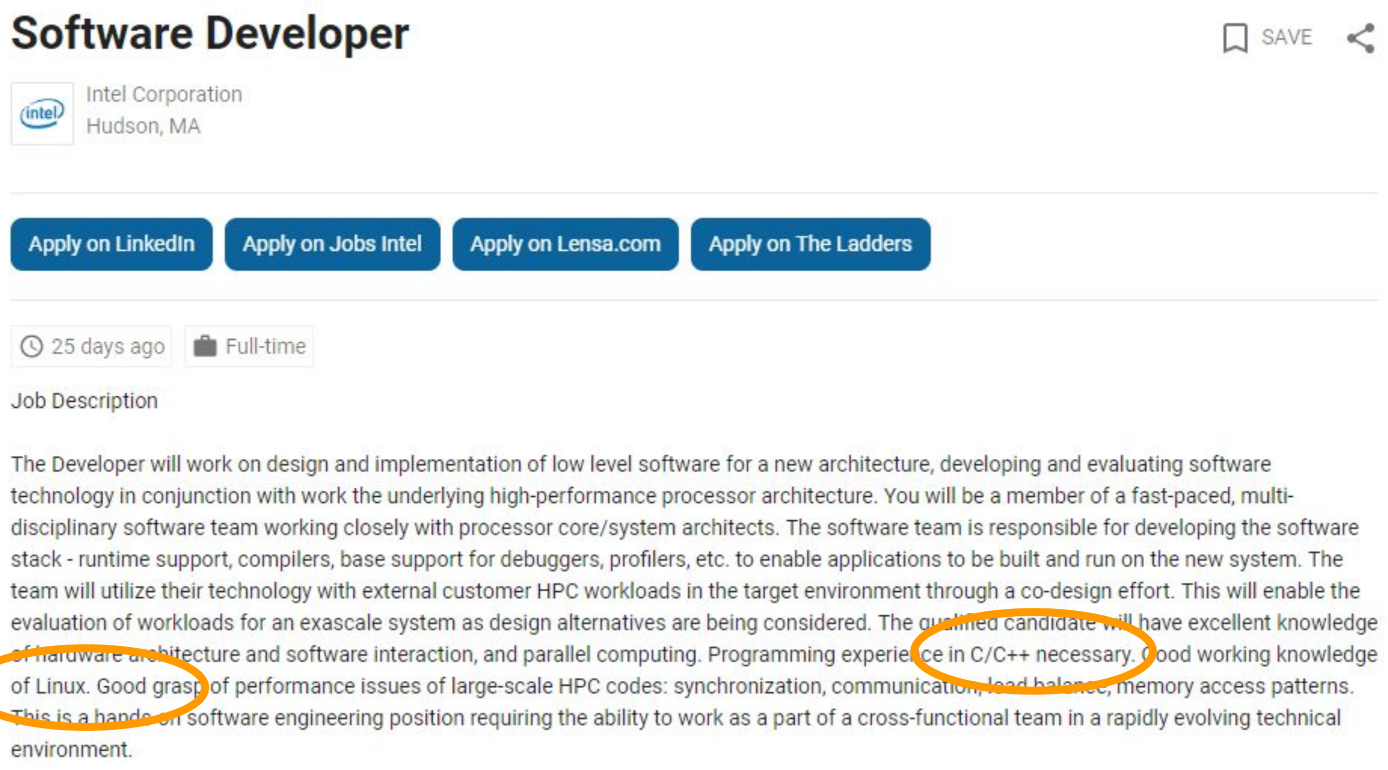
- But do not reveal solutions in public Piazza posts

Expectations

- You have taken some ‘programming’ related class.
  - 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)

Yes I know there is GO, Erlang, Rust, etc.

Why C?



Software Developer

SAVE

Intel Corporation
Hudson, MA

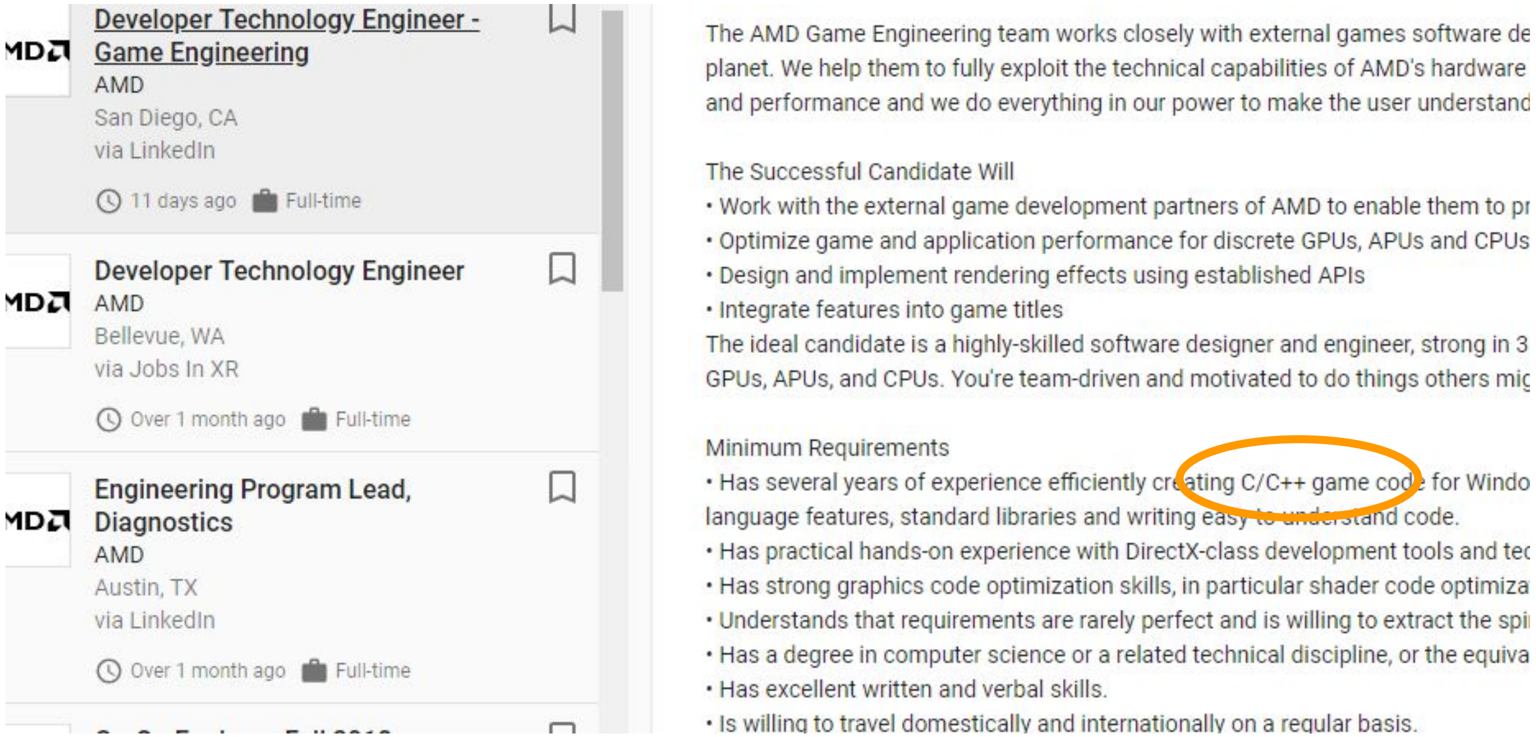
Apply on LinkedIn | Apply on Jobs Intel | Apply on Lensa.com | Apply on The Ladders

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

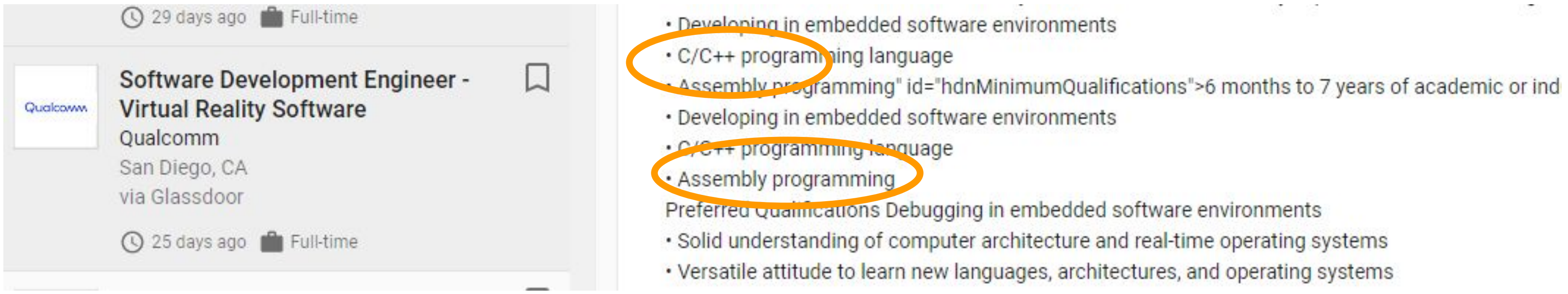
Engineering Program Lead, Diagnostics
AMD
Austin, TX
via LinkedIn
Over 1 month ago Full-time

The AMD Game Engineering team works closely with external games software de
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 pr
• 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 3
GPUs, APUs, and CPUs. You're team-driven and motivated to do things others mig

Minimum Requirements
• Has several years of experience efficiently creating C/C++ game code for Windo
language features, standard libraries and writing easy to understand code.
• Has practical hands-on experience with DirectX-class development tools and tec
• Has strong graphics code optimization skills, in particular shader code optimiza
• Understands that requirements are rarely perfect and is willing to extract the spi
• Has a degree in computer science or a related technical discipline, or the equiva
• 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

How to be successful in CS 3650

- Read the assigned reading before class

- Attend the class
    - Ask questions
    - Answer questions
    - Take notes

- You need theoretical backgrounds from class to succeed in labs/assignments/projects

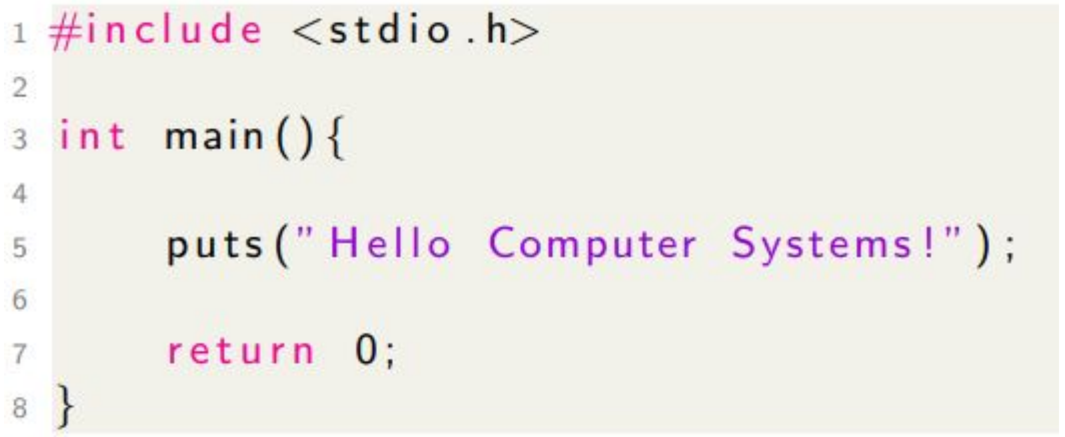
How to be successful in CS 3650

- Labs/Assignments/Projects
  - Plan ahead and start early
  - DO NOT START AT THE LAST MINUTE
  - Ask questions early
    - Setting up the environment itself could take a long time
    - Coding always takes longer than your expectation
    - Debugging can take longer than you think
    - If your stuck on one thing for <u>more than an hour</u>: ask for help

Questions?

So what exactly is C?

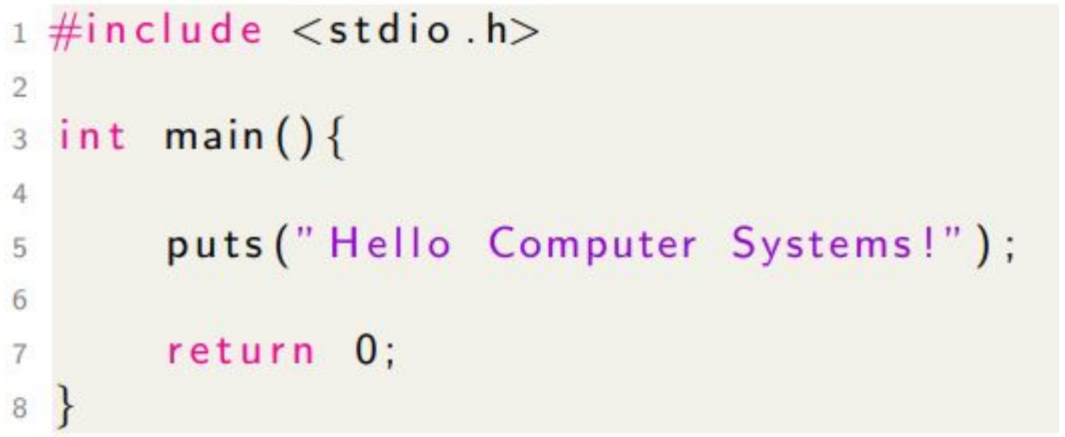
Here is what 'C' looks like



```c
#include <stdio.h>

int main(){

    puts("Hello Computer Systems!");

    return 0;
}
```

Here is what ‘C’ looks like

- compile with: clang hello.c -o hello



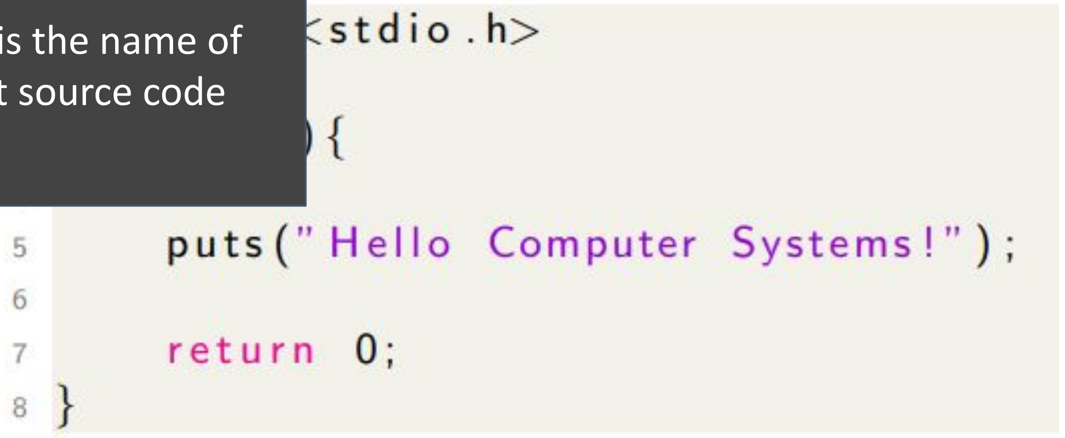
```c
#include <stdio.h>

int main(){

    puts("Hello Computer Systems!");

    return 0;
}
```

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

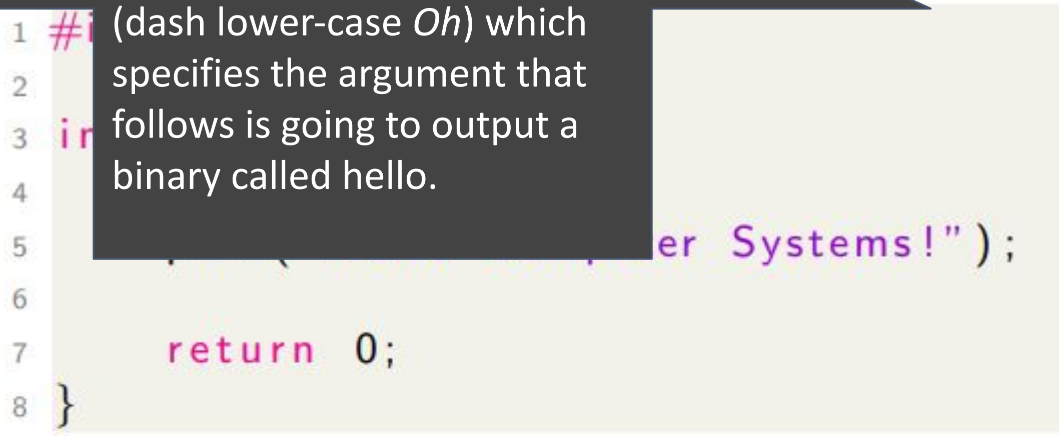


```
<stdio.h>

){

    puts("Hello Computer Systems!");

    return 0;
}
```

Here is what ‘C’ looks like

- compile with: clang hello.c -o hello

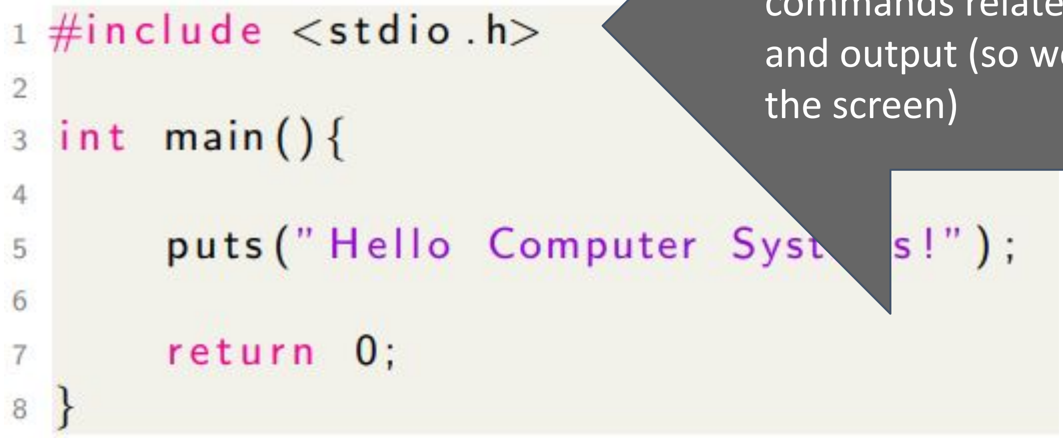
And we are using a flag ‘-o’ (dash lower-case *Oh*) which specifies the argument that follows is going to output a binary called hello.



```
#i

ir

                                  er Systems!");

    return 0;
}
```

Here is what ‘C’ looks like

- compile with: clang hello.c -o hello

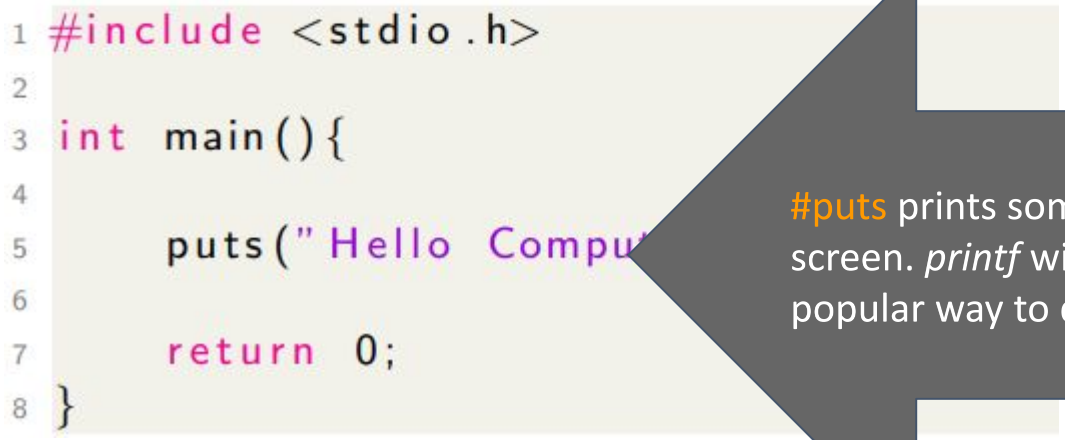


```
#include <stdio.h>

int main(){

    puts("Hello Computer Syst    s!");

    return 0;
}
```

#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

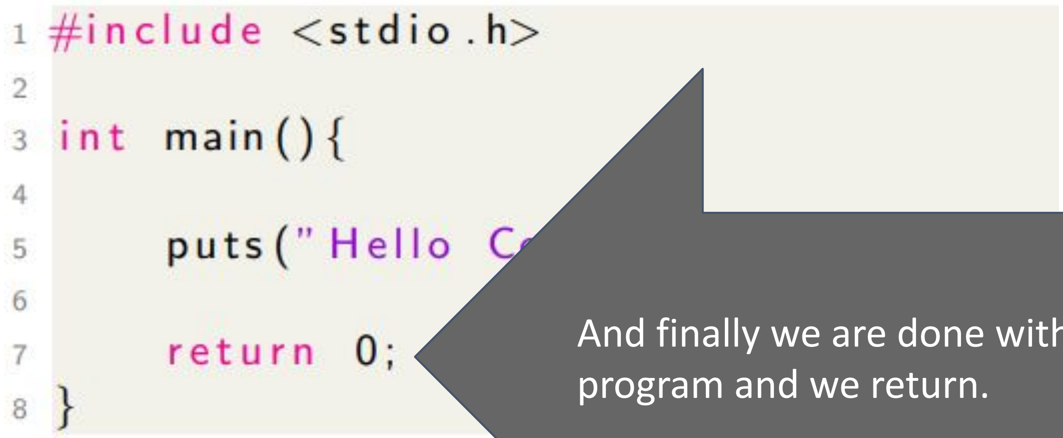


```
#include <stdio.h>

int main(){

    puts("Hello Compu

    return 0;
}
```

#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

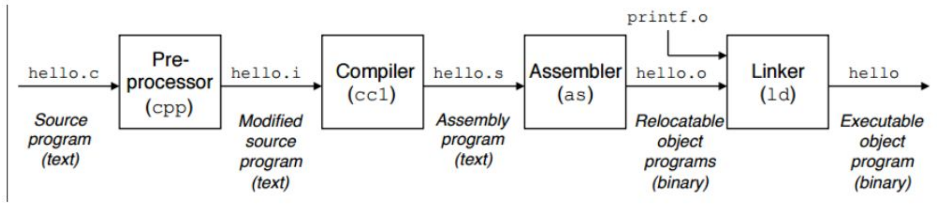


```
#include <stdio.h>

int main(){

    puts("Hello C

    return 0;
}
```

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

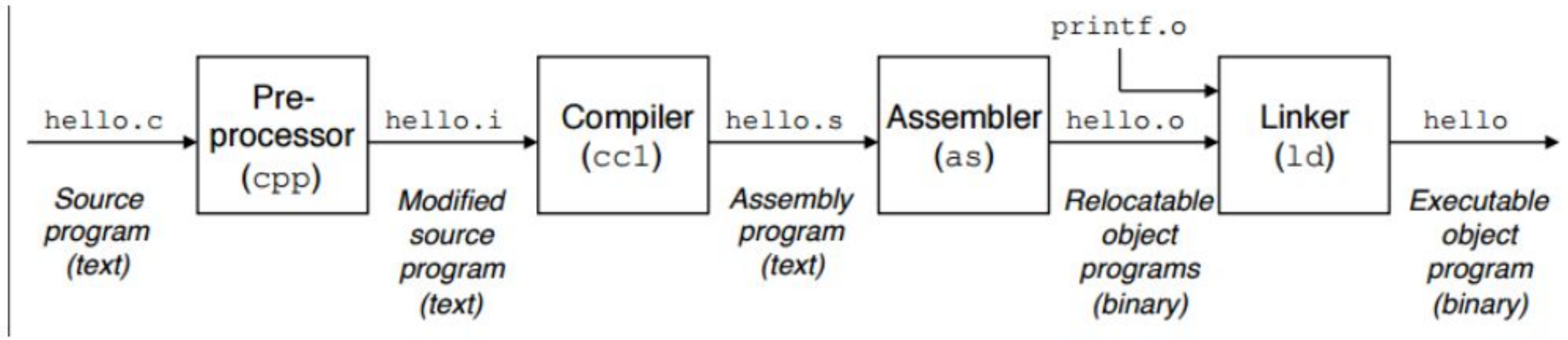
Little exercise to see what compiler is doing

- Generate assembly code
  - clang -S hello.c
- Investigate assembly
- Compile assembly to executable
  - 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 – shows assembly of machine instructions)
  - objdump -t hello.o (shows symbol table)



C and the compilation process

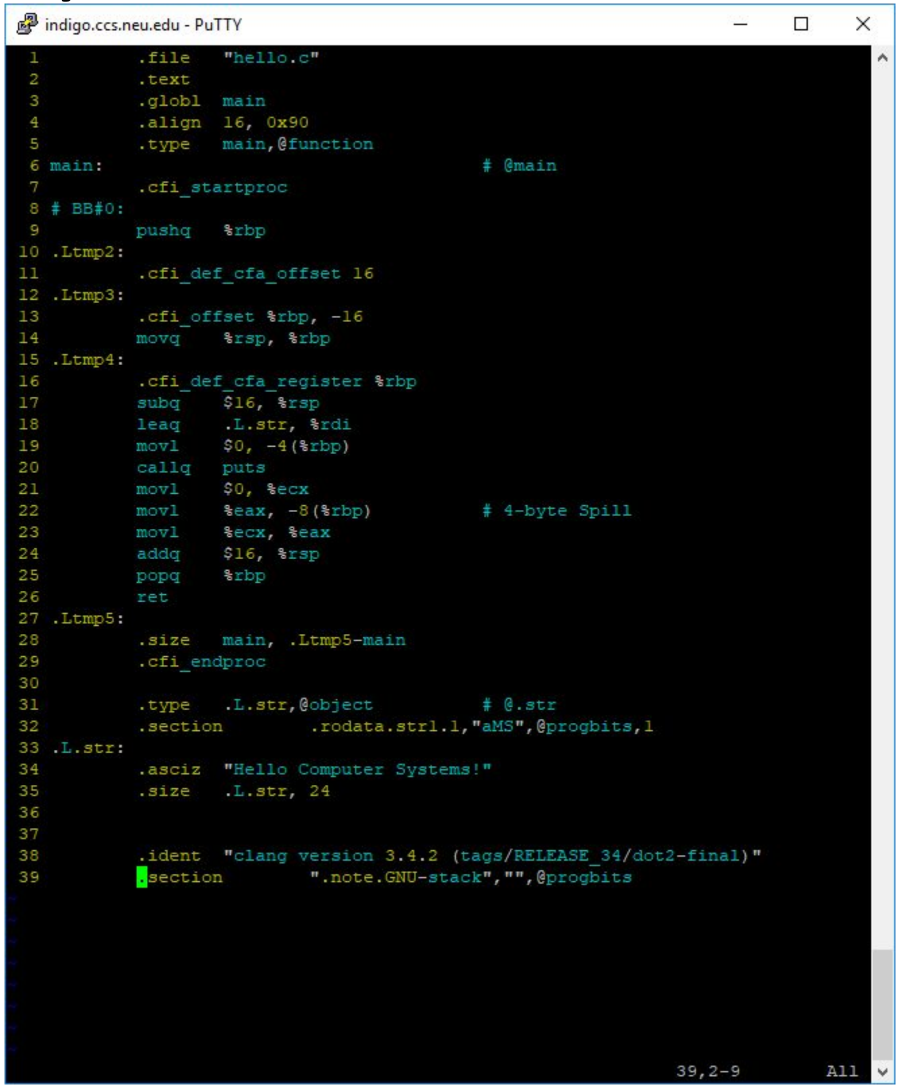
- In a picture, this is the compilation process from start to finish
- (Note in this class we'll use clang, but gcc is also fine)



Quick view of the assembly

- How many folks have not written assembly before?

```c
#include <stdio.h>

int main(){

    puts("Hello Computer Systems!");

    return 0;
}
```



```
        .file   "hello.c"
        .text
        .globl  main
        .align  16, 0x90
        .type   main,@function
main:                                   # @main
        .cfi_startproc
# BB#0:
        pushq   %rbp
.Ltmp2:
        .cfi_def_cfa_offset 16
.Ltmp3:
        .cfi_offset %rbp, -16
        movq    %rsp, %rbp
.Ltmp4:
        .cfi_def_cfa_register %rbp
        subq    $16, %rsp
        leaq    .L.str, %rdi
        movl    $0, -4(%rbp)
        callq   puts
        movl    $0, %ecx
        movl    %eax, -8(%rbp)          # 4-byte Spill
        movl    %ecx, %eax
        addq    $16, %rsp
        popq    %rbp
        ret
.Ltmp5:
        .size   main, .Ltmp5-main
        .cfi_endproc

        .type   .L.str,@object          # @.str
        .section        .rodata.str1.1,"aMS",@progbits,1
.L.str:
        .asciz  "Hello Computer Systems!"
        .size   .L.str, 24


        .ident  "clang version 3.4.2 (tags/RELEASE_34/dot2-final)"
        .section        ".note.GNU-stack","",@progbits
                                                            39,2-9          All
```

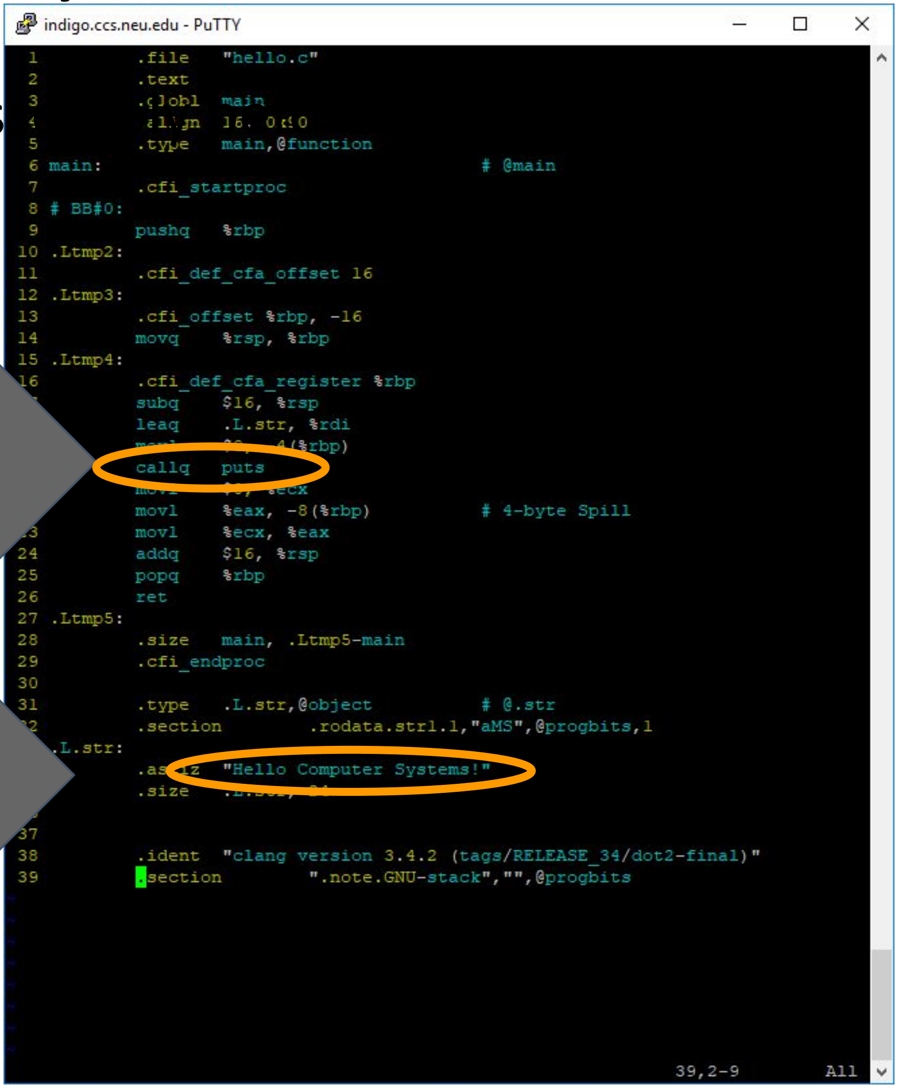
Quick view of the assembly

- How many folks have not written as

```c
#includ
int ma
    puts("Hello Computer Systems!");
    return 0;
}
```

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

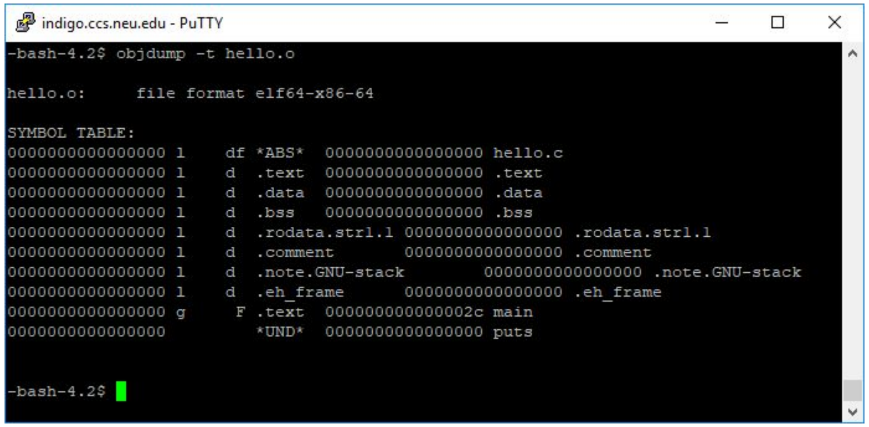
Our string



```
        .file   "hello.c"
        .text
        .globl  main
        .align  16, 0x90
        .type   main,@function
main:                                   # @main
        .cfi_startproc
# BB#0:
        pushq   %rbp
.Ltmp2:
        .cfi_def_cfa_offset 16
.Ltmp3:
        .cfi_offset %rbp, -16
        movq    %rsp, %rbp
.Ltmp4:
        .cfi_def_cfa_register %rbp
        subq    $16, %rsp
        leaq    .L.str, %rdi
        movl    $0, -4(%rbp)
        callq   puts
        movl    $0, %ecx
        movl    %eax, -8(%rbp)          # 4-byte Spill
        movl    %ecx, %eax
        addq    $16, %rsp
        popq    %rbp
        ret
.Ltmp5:
        .size   main, .Ltmp5-main
        .cfi_endproc

        .type   .L.str,@object          # @.str
        .section        .rodata.str1.1,"aMS",@progbits,1
.L.str:
        .asciz  "Hello Computer Systems!"
        .size   .L.str, 24


        .ident  "clang version 3.4.2 (tags/RELEASE_34/dot2-final)"
        .section        ".note.GNU-stack","",@progbits
                                                            39,2-9        All
```

Quick view of objdump

- How many folks have not used objdump before?

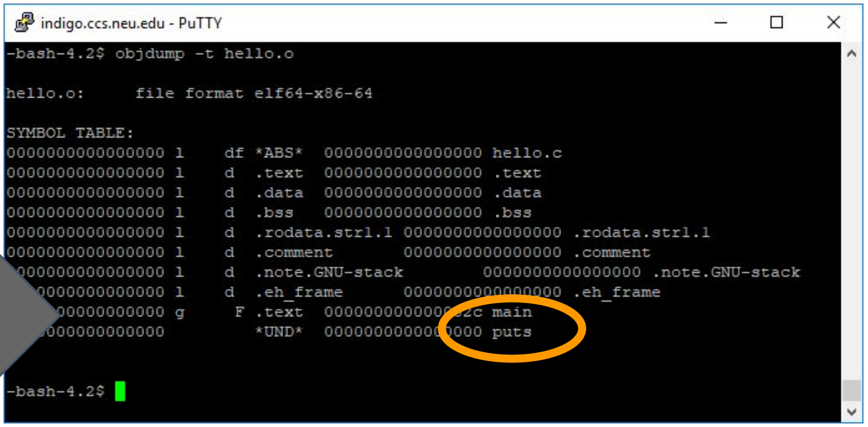


```
-bash-4.2$ objdump -t hello.o

hello.o:     file format elf64-x86-64

SYMBOL TABLE:
0000000000000000 l    df *ABS*  0000000000000000 hello.c
0000000000000000 l    d  .text  0000000000000000 .text
0000000000000000 l    d  .data  0000000000000000 .data
0000000000000000 l    d  .bss   0000000000000000 .bss
0000000000000000 l    d  .rodata.str1.1 0000000000000000 .rodata.str1.1
0000000000000000 l    d  .comment       0000000000000000 .comment
0000000000000000 l    d  .note.GNU-stack        0000000000000000 .note.GNU-stack
0000000000000000 l    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?

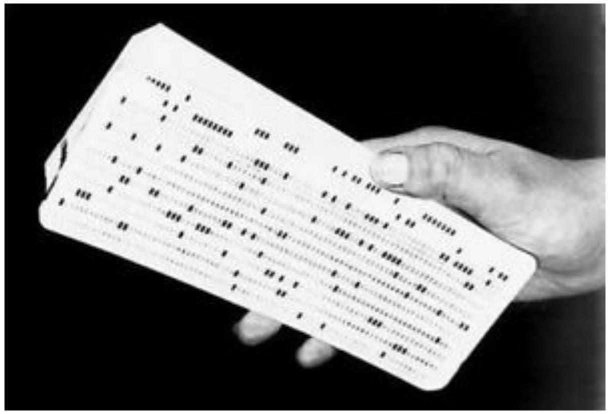
Powerful tool to pull out some information
(Can see functions/libraries used)

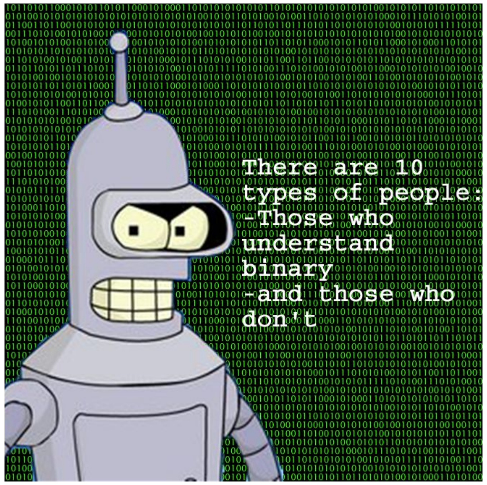


```
-bash-4.2$ objdump -t hello.o

hello.o:     file format elf64-x86-64

SYMBOL TABLE:
0000000000000000 l    df *ABS*  0000000000000000 hello.c
0000000000000000 l    d  .text  0000000000000000 .text
0000000000000000 l    d  .data  0000000000000000 .data
0000000000000000 l    d  .bss   0000000000000000 .bss
0000000000000000 l    d  .rodata.str1.1 0000000000000000 .rodata.str1.1
0000000000000000 l    d  .comment       0000000000000000 .comment
 000000000000000 l    d  .note.GNU-stack        0000000000000000 .note.GNU-stack
   0000000000000 l    d  .eh_frame      0000000000000000 .eh_frame
     00000000000 g     F .text  000000000000002c main
    000000000000         *UND*  0000000000000000 puts


-bash-4.2$
```

So compilers are pretty neat

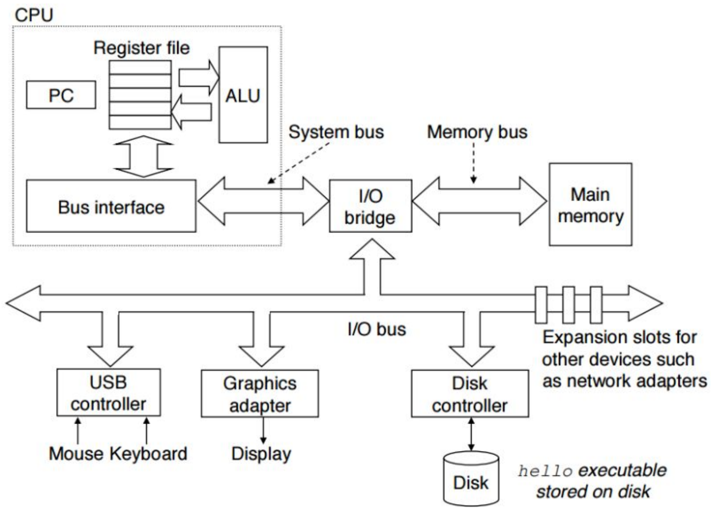
- When we start looking at some of the information taken in, we appreciate the job they do.
  - i.e. transform high level language to binary
- All of a sudden, writing some C code is not so bad!
  - (And it of course is better than pure binary!)

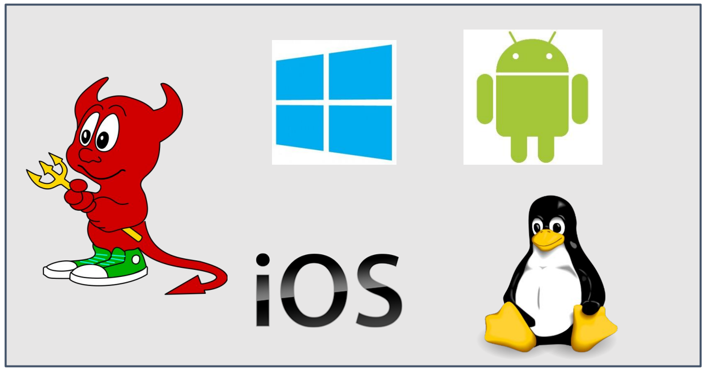




C and compilers allow us to control the system

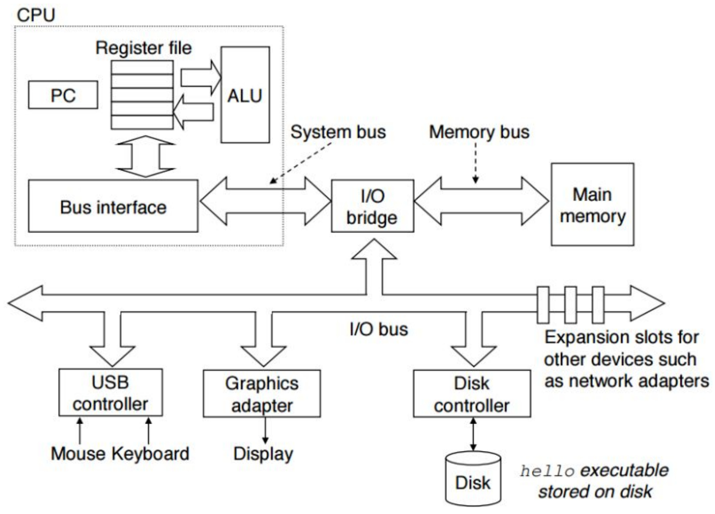
- Core pieces of systems include hardware(left) and operating system (right)





C and compilers allow us to control the system

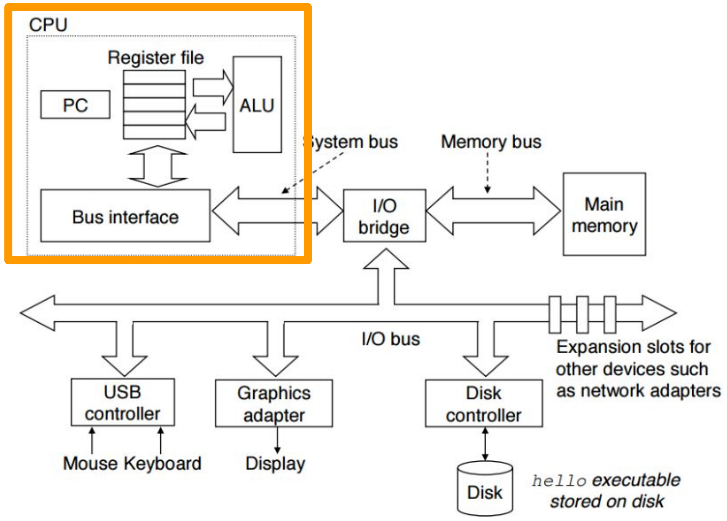
- Core pieces of systems include hardware(left) and operating system (right)



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

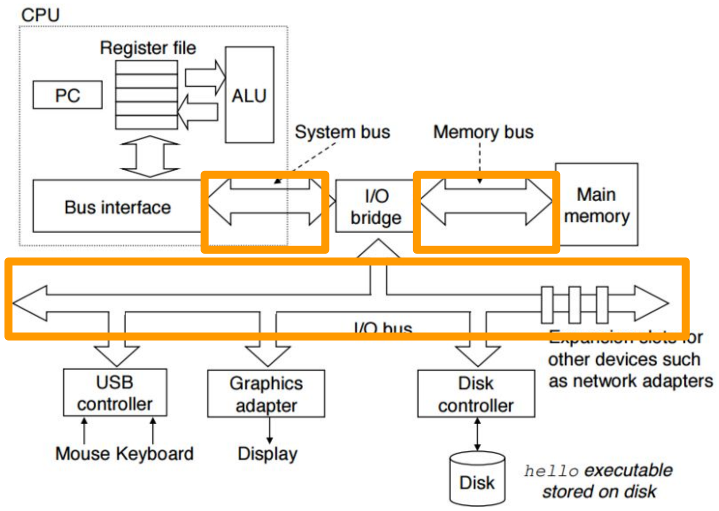
Modern Hardware Visual Abstraction

- CPU is the “brain” of modern hardware
  - 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.

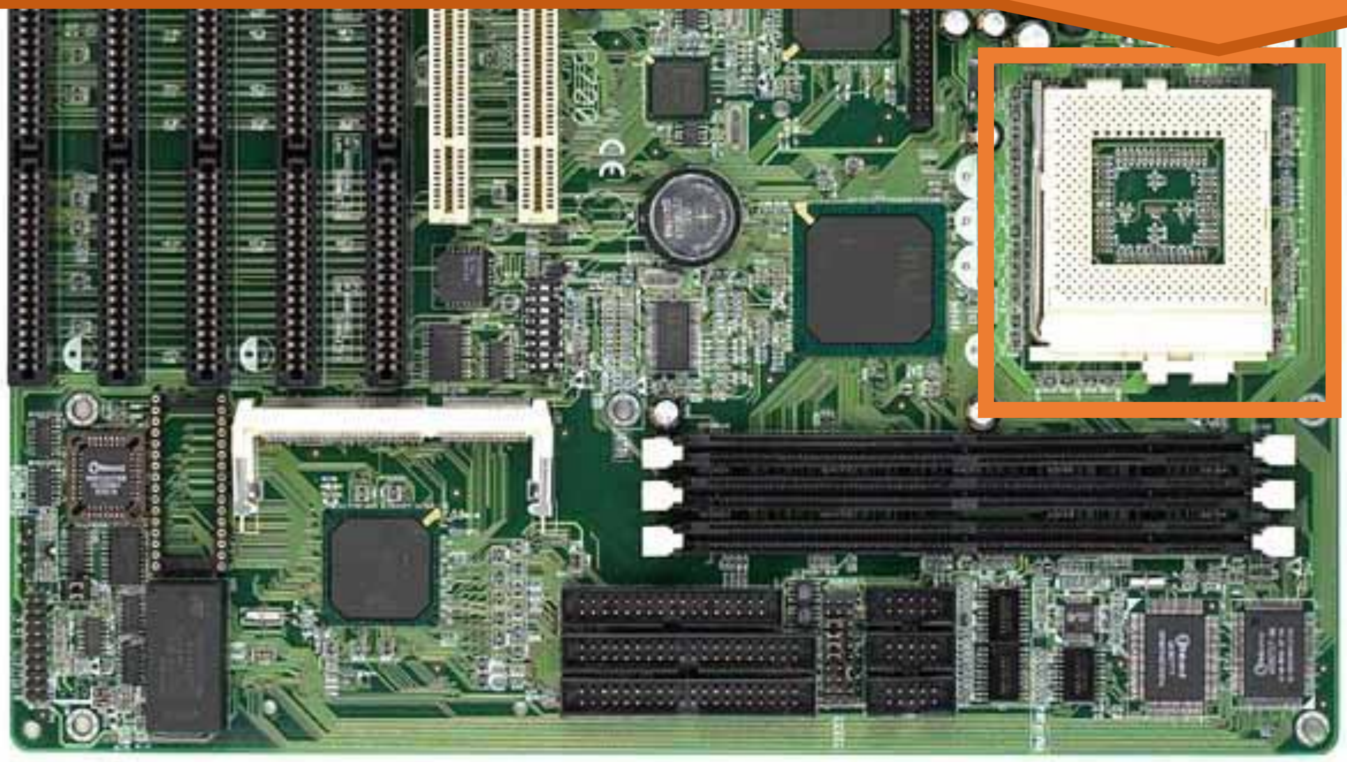


Modern Hardware Visual Abstraction

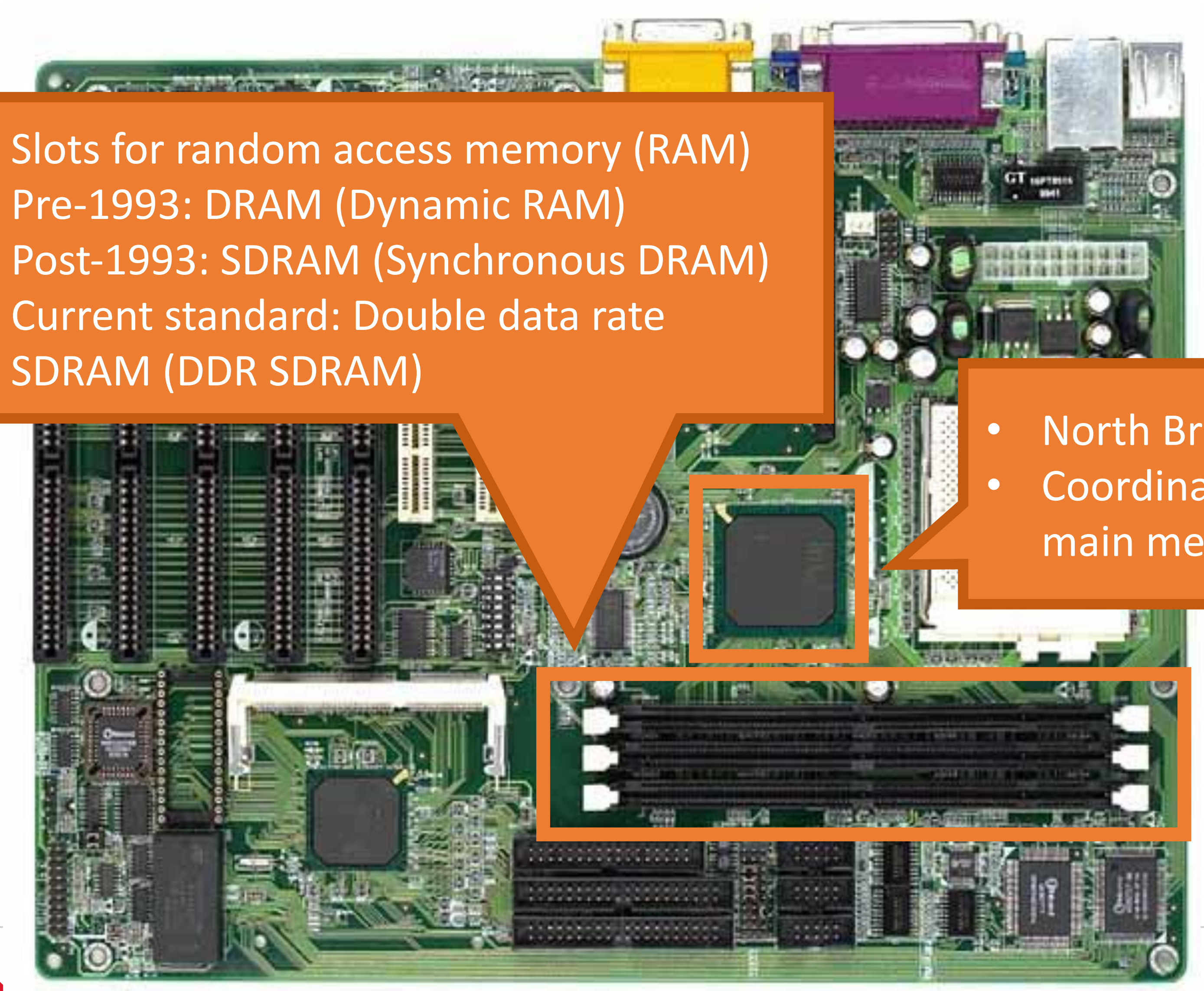
- Beyond the CPU, a number of devices may also be connected.
- Buses transfer information from devices and memory into the CPU.
- There is a lot going on, and this needs to be managed
- Note: Busses can be thought of as simple networks, with many things hardcoded



- CPU Socket
- Many different physical socket standards
  - This a Pentium 1 socket
- Physical standard is less important than Instruction Set Architecture (ISA)
  - IBM PCs are Intel 80386 compatible
  - Original x86 design
  - Intel, AMD, VIA
- Today's dominant ISA: x86-64, developed by AMD



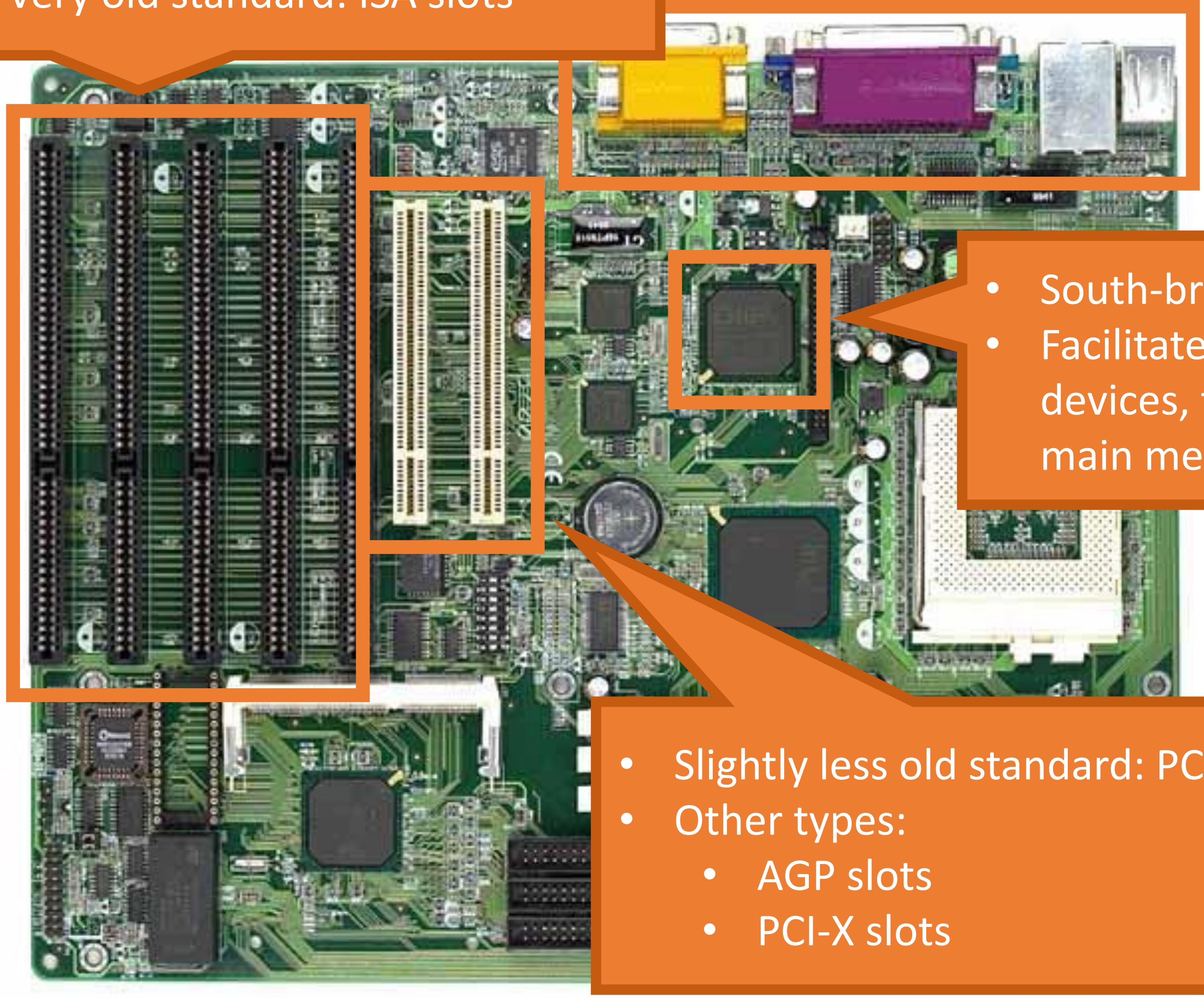
University



- Slots for random access memory (RAM)
- Pre-1993: DRAM (Dynamic RAM)
- Post-1993: SDRAM (Synchronous DRAM)
- Current standard: Double data rate SDRAM (DDR SDRAM)

- North Bridge
- Coordinates access to main memory

University



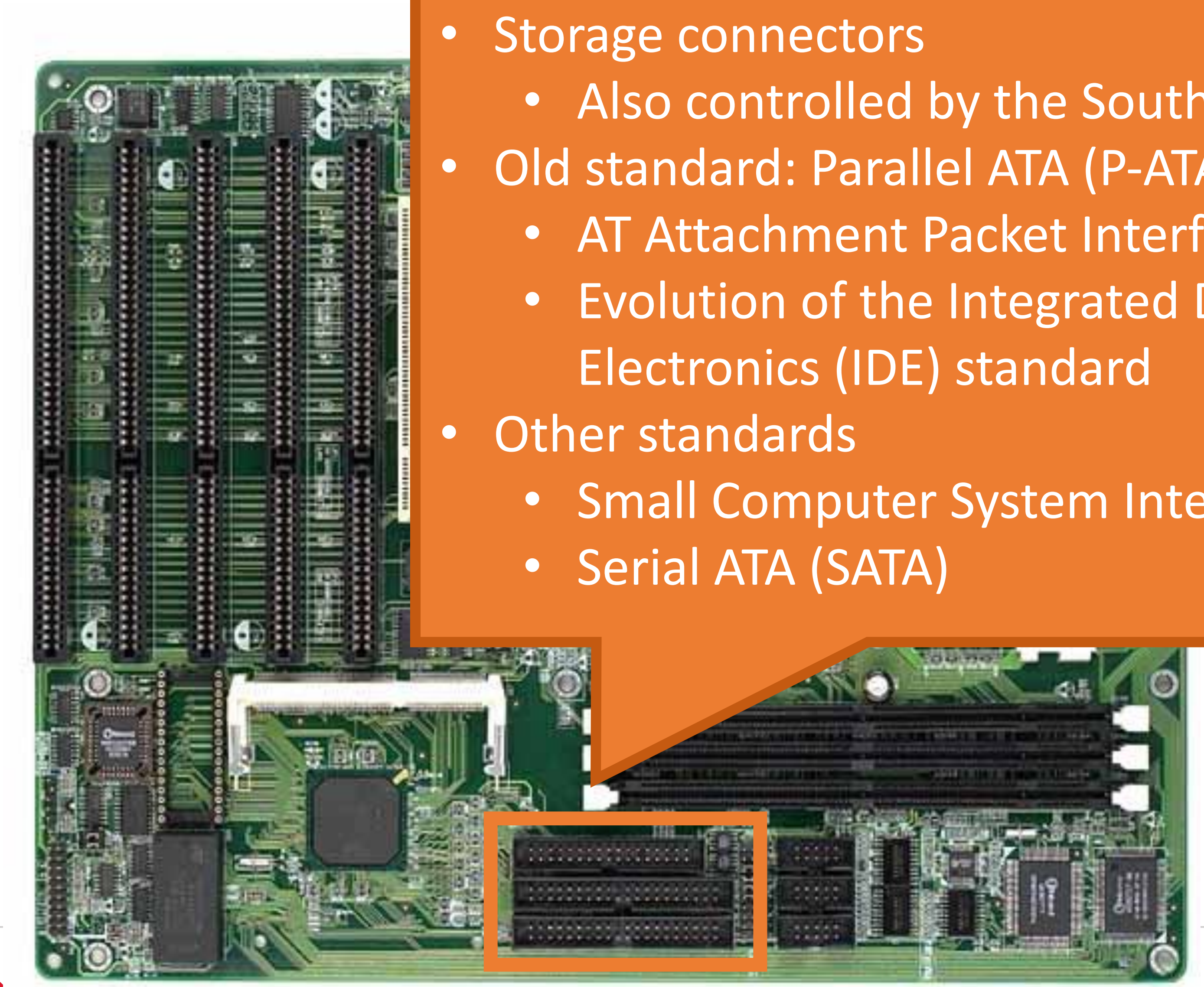
- I/O device slots
- Attached to the south-bridge bus
- Very old standard: ISA slots

- Built in I/O also on the PCI/ISA bus

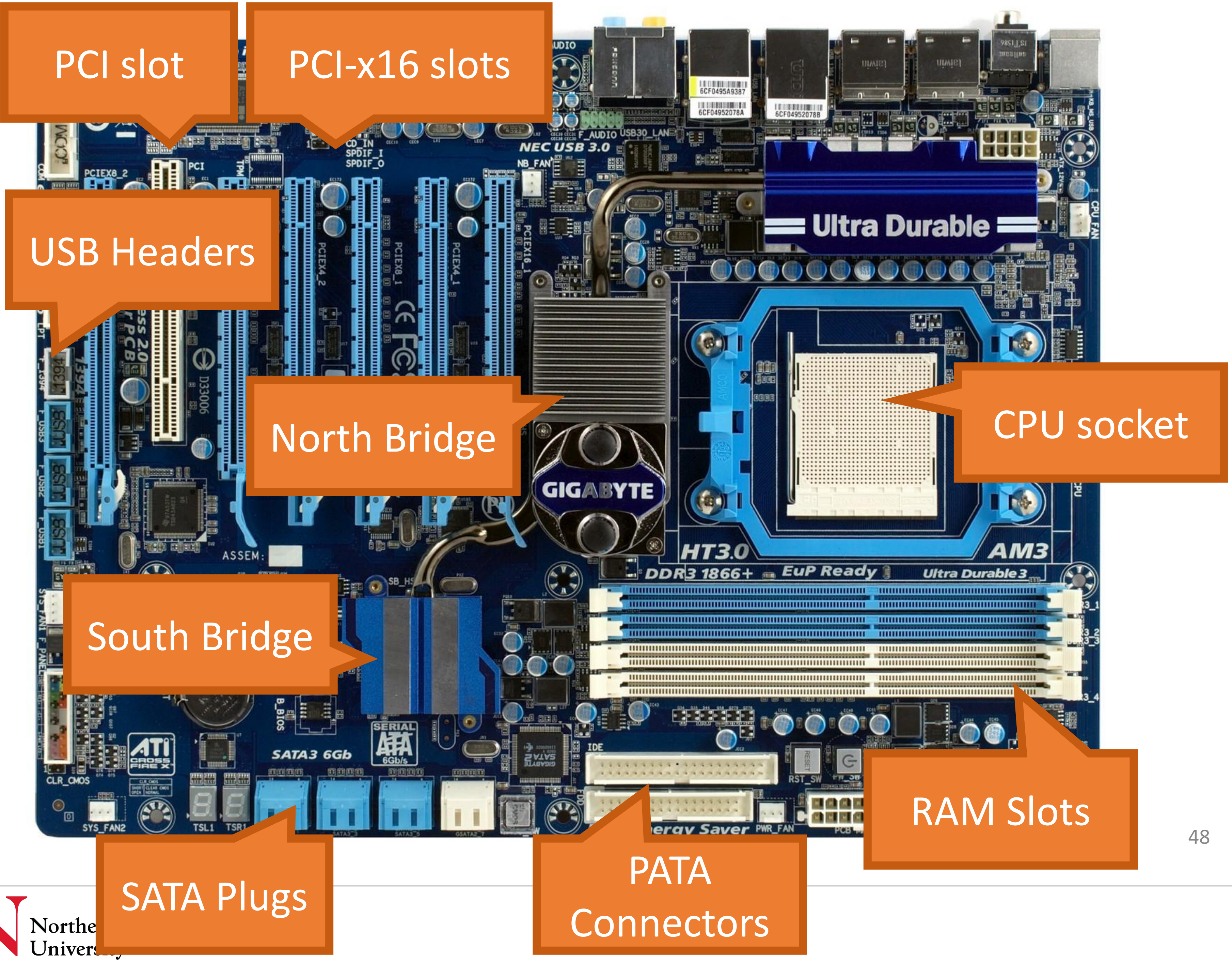
- South-bridge
- Facilitates I/O between devices, the CPU, and main memory

- Slightly less old standard: PCI slots
- Other types:
  - AGP slots
  - PCI-X slots

University



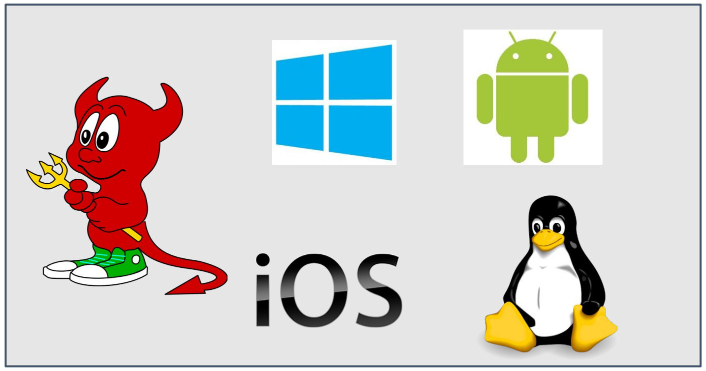
- Storage connectors
  - Also controlled by the South Bridge
- Old standard: Parallel ATA (P-ATA)
  - AT Attachment Packet Interface (ATAPI)
  - Evolution of the Integrated Drive Electronics (IDE) standard
- Other standards
  - Small Computer System Interface (SCSI)
  - Serial ATA (SATA)


PCI slot
PCI-x16 slots
USB Headers
North Bridge
CPU socket
South Bridge
RAM Slots
SATA Plugs
PATA Connectors
Ultra Durable
GIGABYTE
HT3.0
AM3
DDR3 1866+
EuP Ready
NEC USB 3.0
SATA3 6Gb

C and compilers allow us to control the system

- Core pieces of systems include hardware(left) and operating system (right)

Let’s take a moment to think about operating systems



What is an Operating System?

Many Different OSes

Windows

Windows®7

Linux

redhat

BSD

iOS

Many Different OSes

Windows

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

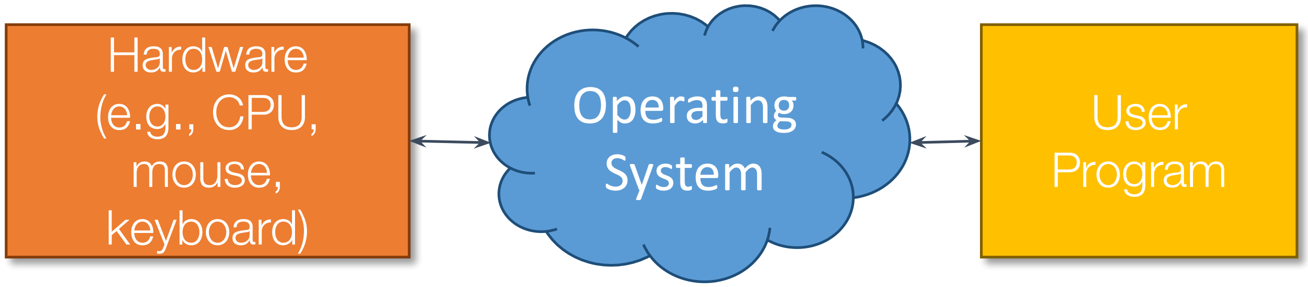
You can actively contribute to the open source ones now!

Linux

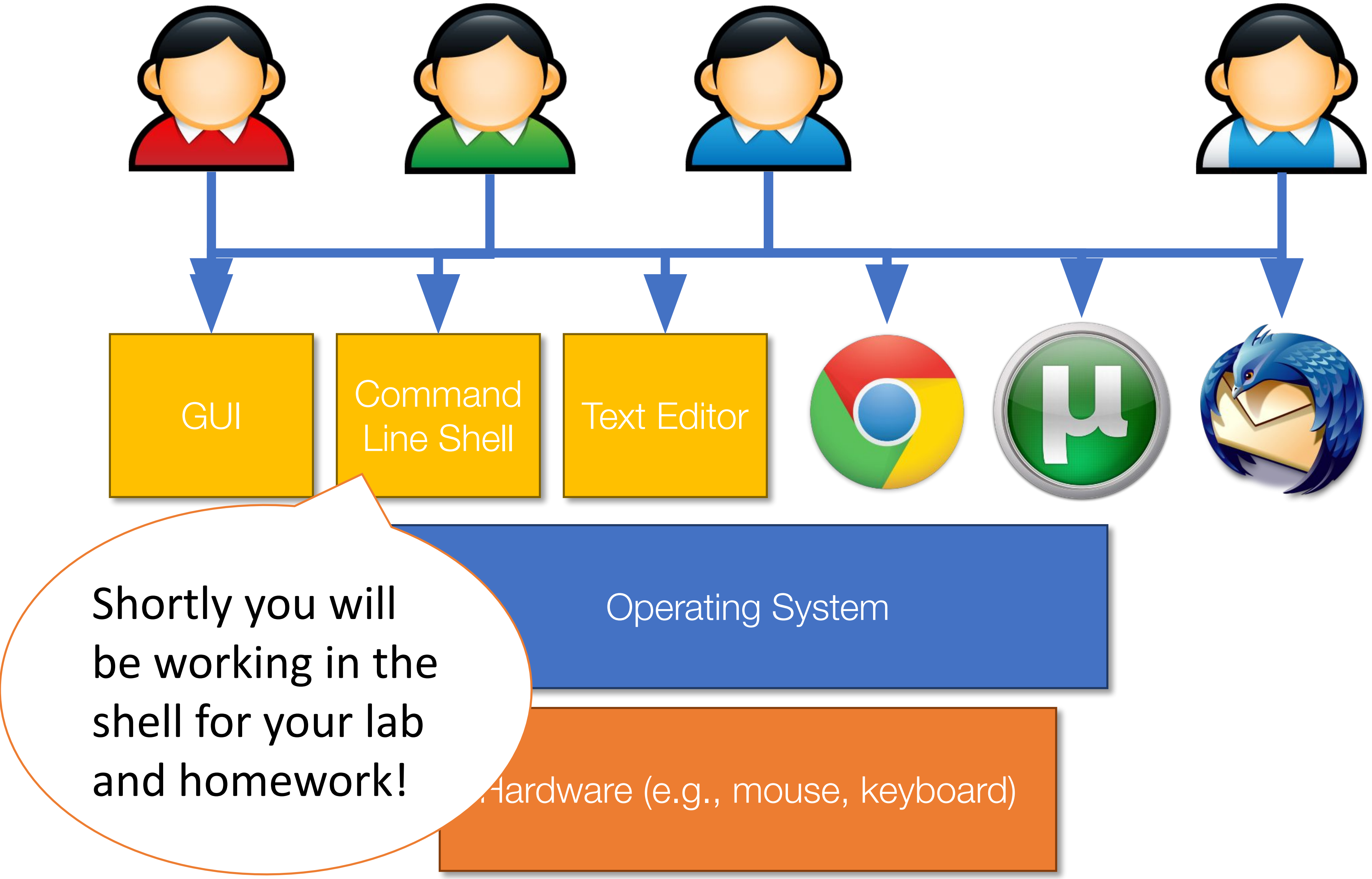
BSD

What is an Operating System?

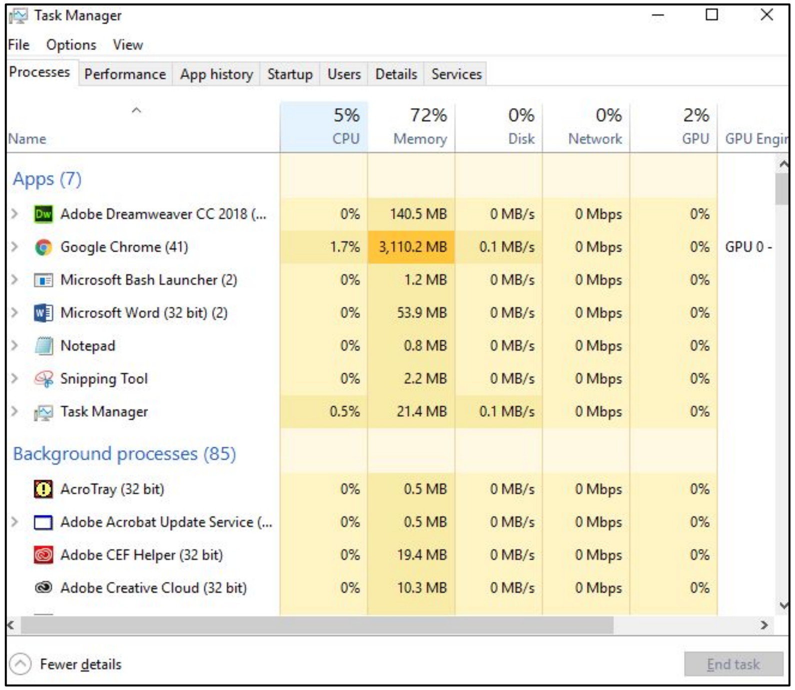
- OS is software that sits between user programs and hardware



- OS provides interfaces to computer hardware
  - User programs do not have to worry about details



Shortly you will be working in the shell for your lab and homework!


Task Manager
File Options View
Processes Performance App history Startup Users Details Services
Fewer details
End task

Name	5% CPU	72% Memory	0% Disk	0% Network	2% GPU	GPU Engi
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%	

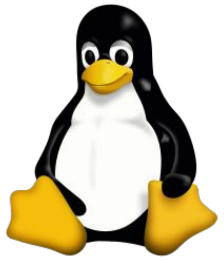
- OS is a resource manager and control program
  - Controls execution of user programs
  - Decides between conflicting requests for hardware access
  - Attempts to be efficient and fair
  - Prevents errors and improper use

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)

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

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, etc.
  - 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 :-(.
```

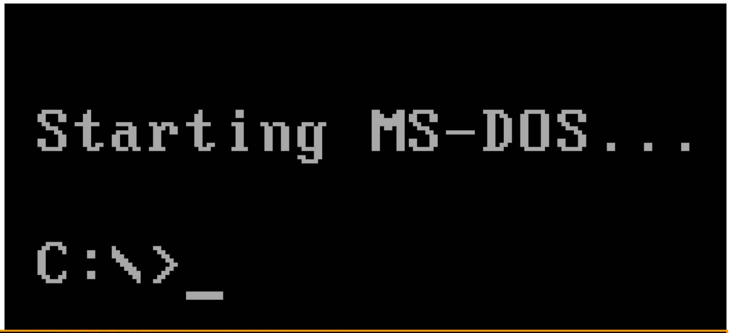
Over 30 years ago…

```
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) ope
> professional like gnu
> since april, and is s
> things people like/di
> (same physical layout
> among other things).
>
> I've currently ported                                           k.
> This implies that I'll get s          actical within a few months, and
> I'd like to know what feat          people would want. Any suggestions
> are welcome, but I won't         e I'll implement them :-)
>
>       Linus (torv..      una.helsinki.fi)
>
> PS. Yes - it's  ree 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 :-(.
```

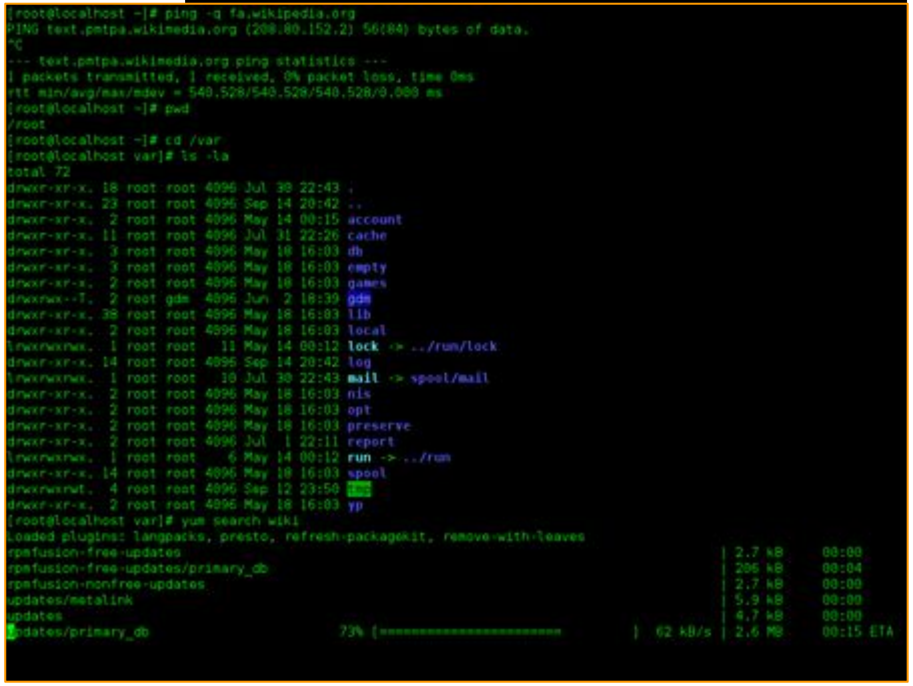
Linux platforms: Alpha, ARC, ARM, ARM64, Apple M1 C6x, H8/300, Hexagon, Itanium, m68k, Microblaze, MIPS, NDS32, Nios II, OpenRISC, PA-RISC, PowerPC, RISC-V, s390, SuperH, SPARC, Unicore32, x86, x86-64, XBurst, Xtensa

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).
  - For Windows: wsl2 is highly recommended
- From it, we can type in the the names of programs to perform work for us



```
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.
^C
--- 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
drwxr-xr-x. 18 root root 4096 Jul 30 22:43 .
drwxr-xr-x. 23 root root 4096 Sep 14 20:42 ..
drwxr-xr-x.  2 root root 4096 May 14 00:15 account
drwxr-xr-x. 11 root root 4096 Jul 31 22:26 cache
drwxr-xr-x.  3 root root 4096 May 18 16:03 db
drwxr-xr-x.  3 root root 4096 May 18 16:03 empty
drwxr-xr-x.  2 root root 4096 May 18 16:03 games
drwxrwx--T.  2 root gdm  4096 Jun  2 18:39 gdm
drwxr-xr-x. 38 root root 4096 May 18 16:03 lib
drwxr-xr-x.  2 root root 4096 May 18 16:03 local
lrwxrwxrwx.  1 root root   11 May 14 00:12 lock -> ../run/lock
drwxr-xr-x. 14 root root 4096 Sep 14 20:42 log
lrwxrwxrwx.  1 root root   10 Jul 30 22:43 mail -> spool/mail
drwxr-xr-x.  2 root root 4096 May 18 16:03 nis
drwxr-xr-x.  2 root root 4096 May 18 16:03 opt
drwxr-xr-x.  2 root root 4096 May 18 16:03 preserve
drwxr-xr-x.  2 root root 4096 Jul  1 22:11 report
lrwxrwxrwx.  1 root root    6 May 14 00:12 run -> ../run
drwxr-xr-x. 14 root root 4096 May 18 16:03 spool
drwxrwxrwt.  4 root root 4096 Sep 12 23:50 tmp
drwxr-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                                   | 206 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?

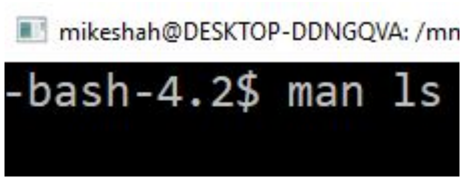
- You might argue “I love GUI interfaces, so simple and sleek looking”
- The command line is a lot faster than moving your mouse
- It is also very convenient for ‘scripting’ behavior that you could not so easily do in a GUI environment.
    - Executing a few commands in a row in a script is a piece of 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)

Shell demo

- ls
- cd (cd ~, /, ..)
- pwd
- tree
- tab
- up/down arrow
- history

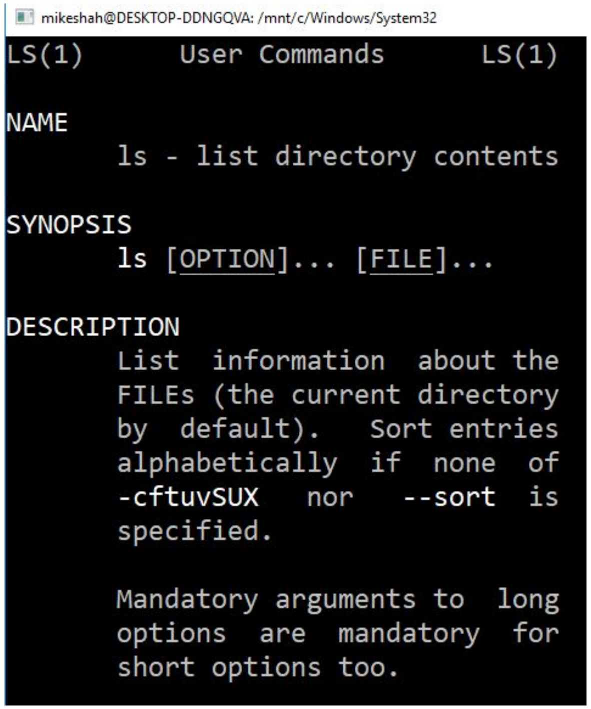
Feeling overwhelmed or forgetting a command?

- 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: /mn

```
-bash-4.2$ man ls
```

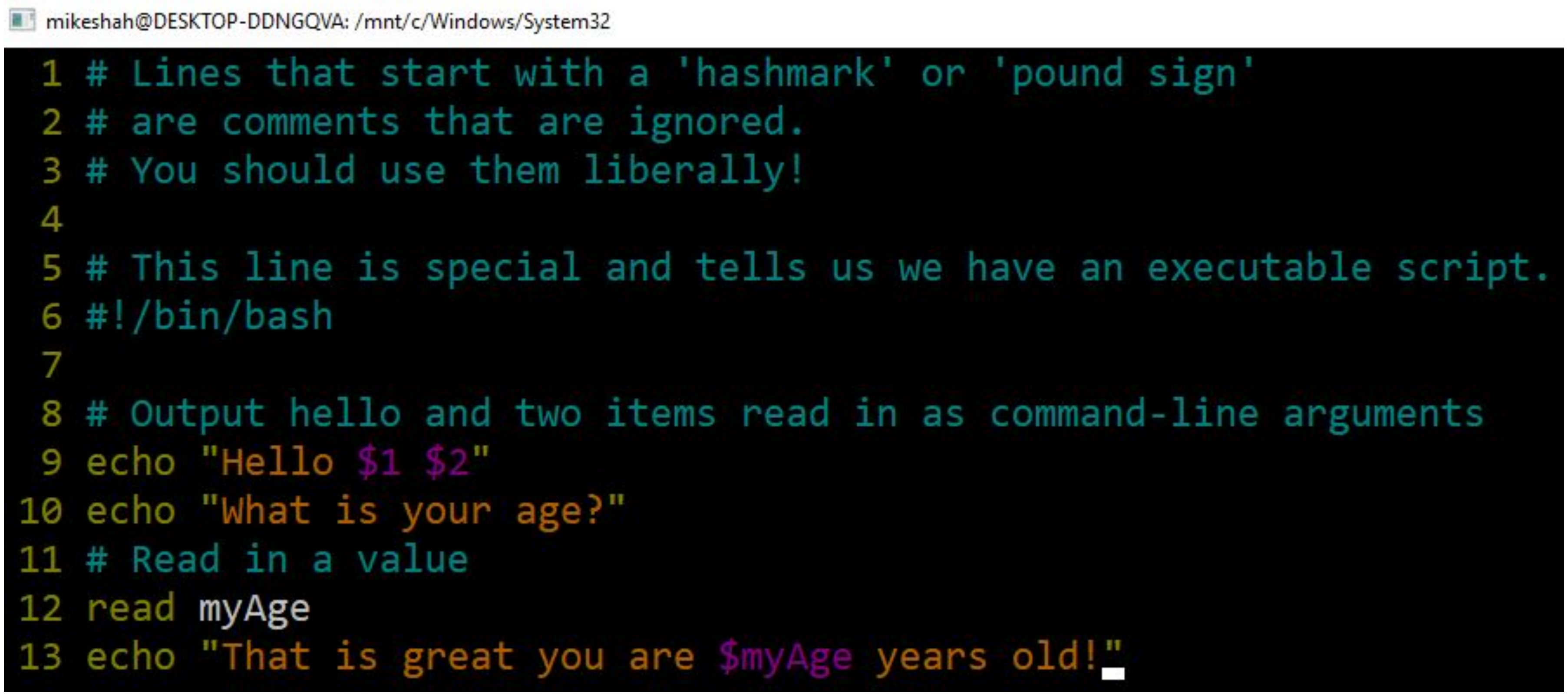


mikeshah@DESKTOP-DDNGQVA: /mnt/c/Windows/System32

```
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.
```

Bash Script Demo

Example shell script

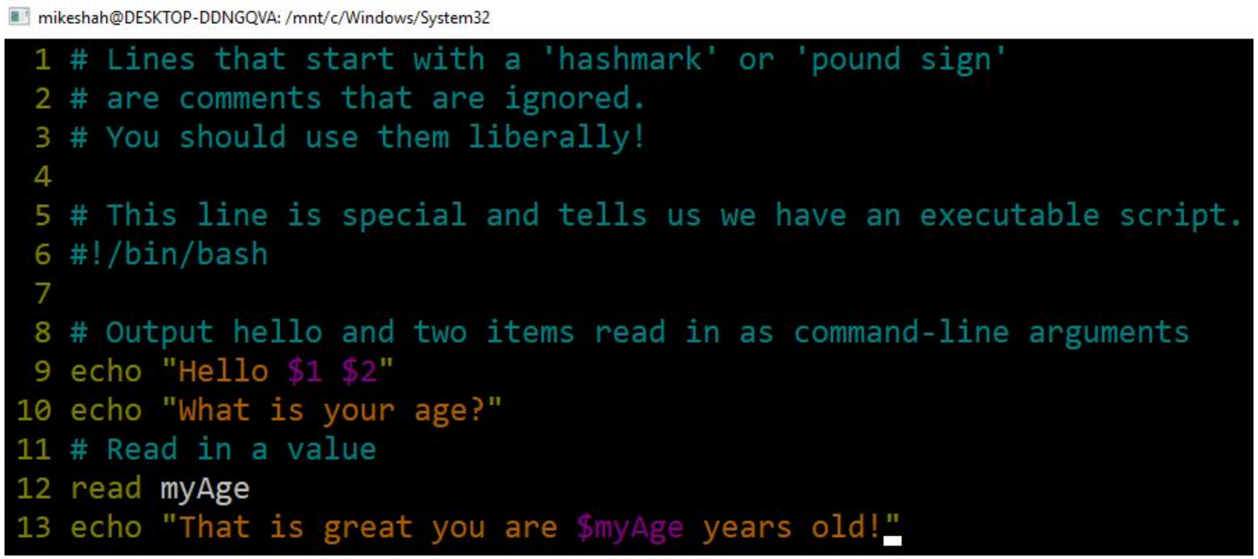


mikeshah@DESKTOP-DDNGQVA: /mnt/c/Windows/System32

```
# Lines that start with a 'hashmark' or 'pound sign'
# are comments that are ignored.
# You should use them liberally!

# This line is special and tells us we have an executable script.
#!/bin/bash

# Output hello and two items read in as command-line arguments
echo "Hello $1 $2"
echo "What is your age?"
# Read in a value
read myAge
echo "That is great you are $myAge years old!"
```

Example shell script

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

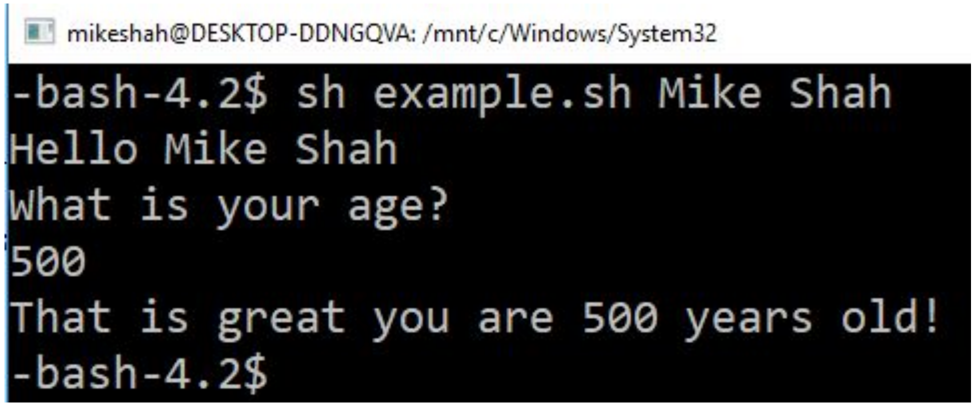


mikeshah@DESKTOP-DDNGQVA: /mnt/c/Windows/System32

```
# Lines that start with a 'hashmark' or 'pound sign'
# are comments that are ignored.
# You should use them liberally!

# This line is special and tells us we have an executable script.
#!/bin/bash

# Output hello and two items read in as command-line arguments
echo "Hello $1 $2"
echo "What is your age?"
# Read in a value
read myAge
echo "That is great you are $myAge years old!"
```

Example shell script Executing

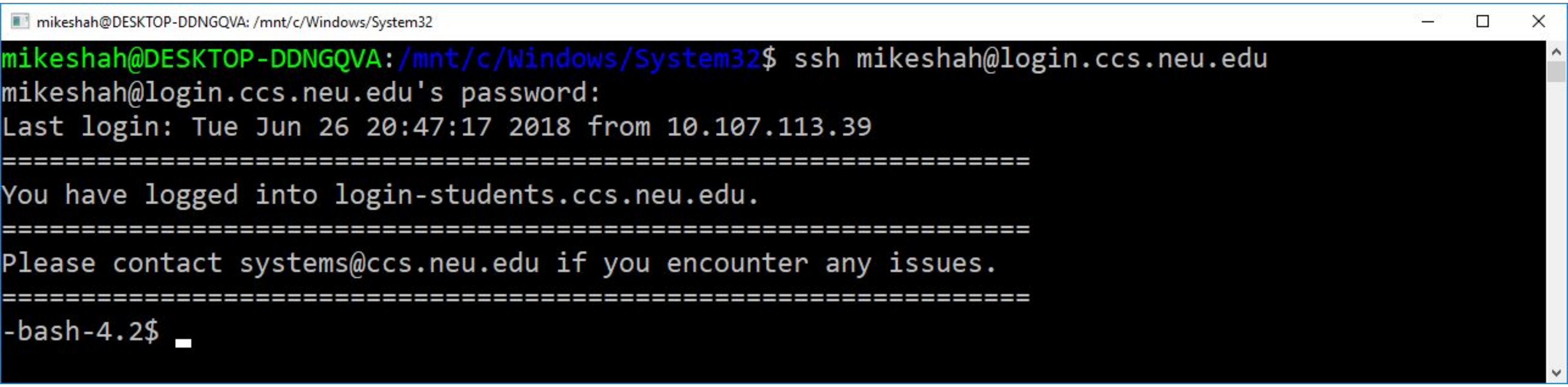
- (Am I really 500 years old? Time flies when you are having fun!)
  - Note "Mike Shah" are the first and second arguments passed into this program


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$
```

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

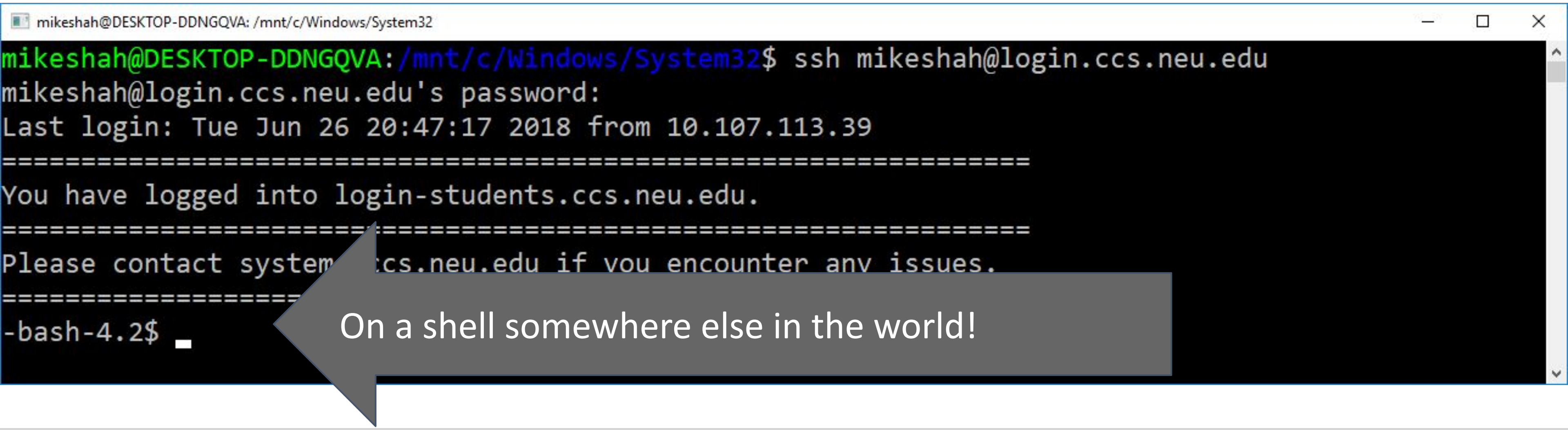


```
mikeshah@DESKTOP-DDNGQVA:/mnt/c/Windows/System32$ ssh mikeshah@login.ccs.neu.edu
mikeshah@login.ccs.neu.edu's password:
Last login: Tue Jun 26 20:47:17 2018 from 10.107.113.39
=================================================================
You have logged into login-students.ccs.neu.edu.
=================================================================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

SSH Demo

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

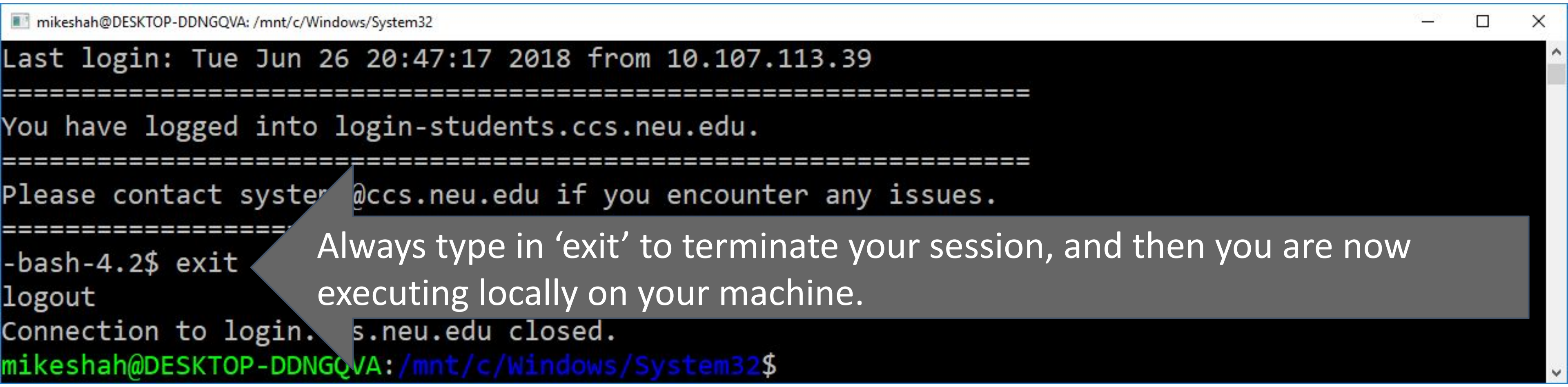


mikeshah@DESKTOP-DDNGQVA: /mnt/c/Windows/System32

```
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 system    cs.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



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 syster @ccs.neu.edu if you encounter any issues.
==================
-bash-4.2$ exit
logout
Connection to login.  s.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.

Version Control

- Shared access to codebase
- Every change is logged
- Allows parallel development of multiple features
- Git, Subversion, Mercurial

Git Demo

CS 3650 Computer Systems – Summer 1 2025

Introduction to Computer Systems (cont.)

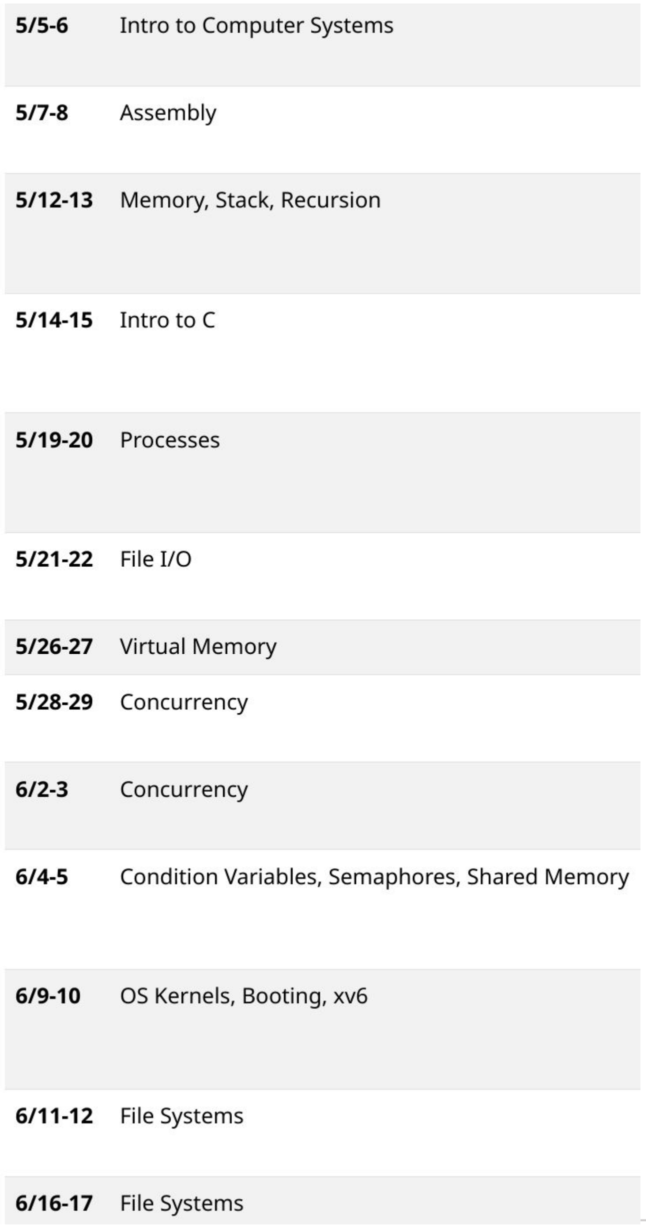
Unit 1

Northeastern University

* Acknowledgements: created based on Christo Wilson, Ferdinand Vesely, and Alden Jackson's lecture slides for the same course.
Based on Ji-Yong Shin's slides

Course Progression

- 11 units
- Most units will last two lectures
- Some units will last three or four lectures



5/5-6	Intro to Computer Systems
5/7-8	Assembly
5/12-13	Memory, Stack, Recursion
5/14-15	Intro to C
5/19-20	Processes
5/21-22	File I/O
5/26-27	Virtual Memory
5/28-29	Concurrency
6/2-3	Concurrency
6/4-5	Condition Variables, Semaphores, Shared Memory
6/9-10	OS Kernels, Booting, xv6
6/11-12	File Systems
6/16-17	File Systems

Quizzes

- Most units, there will include a quiz on the topics from class
- The intention is to make you engage with the material
- Questions will be from lectures and readings

Testing is a great way to learn, but don’t wait for me to quiz you:

Quiz yourself!

Labs

- Practice implementation techniques and tools
- Exercises related to the unit topic
    - Preparation for current or upcoming assignment/project
- Graded mostly on effort – the intention is to encourage you to do the exercises as preparation for assignments
- Ideally, provided class time (30-60 minutes) to work on the labs
    - May become homework if more lecture time is needed
-

Assignments and Projects

- Most of your grade will come from these
- There will be about **eight assignments** and **two projects**
- Out of the eight assignments, the <u>first four are strictly individual</u>, the <u>remainder and the projects can be completed in pairs</u>

Projects

- There will be two projects
- These are longer (two weeks), more substantial programming exercises
  - More planning and/or experimenting required
- Description will be intentionally vague – you are expected to do more reading, thinking, and asking

Grading

Class Activities (around 10)	15%
Labs (around 10)	15%
Assignments (around 8)	48%
Projects (2)	22%

Regrading: Coach's Challenge

- Two (2) challenges each semester.
  a. Must come to office hours and make a formal challenge specifying the problem or problems you want to be regraded, and
  b. For each of these problems, why you think the problem was misgraded. Be specific or the same mistake might happen again.
- **If error** in grading, grade will be corrected: keep your challenge.
- **If original grade stands**, permanently lose your challenge.
- If two challenges are exhausted, cannot request regrades.
- You may not challenge any points lost due to lateness.
- Group projects
  a. All members must have an available challenge to contest a grade.
  b. Challenge results apply to each member of the group

Meet someone new

- Turn to a person near you
- Introduce yourself
  - Where are you from?
  - What are your hobbies?

Containers and Docker

Why containerize applications?

-- build once, run anywhere --

- A container packs together an <u>application and all its dependencies</u> and <u>isolates the application</u> from the rest of the machine it runs on.
- **Running multiple instances:** Because the dependencies are isolated from each other you can run multiple containers on the same machine without them interfering with each other.
- **Automated installation on clusters:** Orchestrators (such as Kubernetes) automatically distribute containerized applications across a cluster of servers so you do not have to manually install applications.

Containers vs VMs

- **Hypervisors:** pure virtual machine environment, a dedicated kernel-level VMM program runs instead of the OS kernel.
- **Hosted VM:** VMs are hosted by the host OS, a VMM runs on the host OS and the guest OS runs on the VMM – e.g. VirtualBox on your laptop
- **Containers:** share the kernel with the OS on the machine they are running on
- VM – fixed resources, overhead of running a whole kernel.
- Faster to start a container than a kernel
- VMs offer better isolation

Linux features that make containers work

- **Control groups (Cgroups):** limits the resources, such as memory, CPU, and network input/output, that a group of Linux processes can use
  - By limiting the resources a process can use, containers provide protection against attacks that consume excessive resources
- **Linux Namespaces:** restricts visibility of resources to a process
  - By putting a process in a namespace, you can restrict the resources that are visible to that process
- **Changing the Root Directory:** limits the set of files and directories that a process can see
  - By changing the root directory when the container is created, a container cannot see the host's entire filesystem

What is Docker?

- Docker is an application that allows you to create and build containers
    - Set of commands to manipulate images and containers

- **Image:** complete and executable version of an application
- **Container:** is the instantiation of an image

- Docker maintains a repository of images

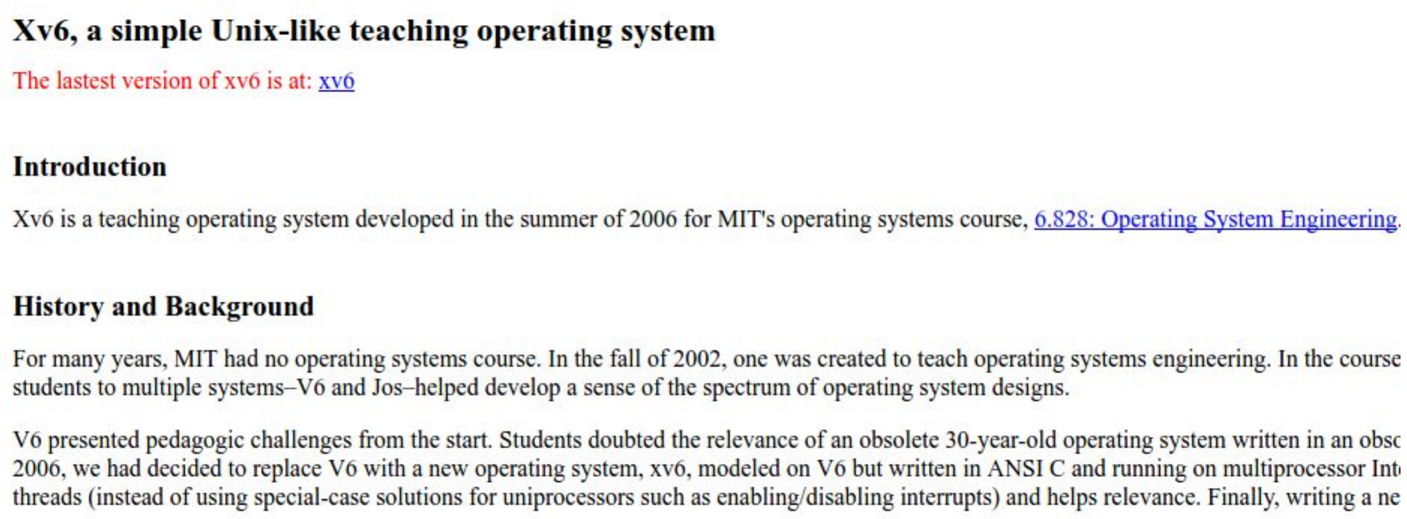
Docker Demo

Northeastern University

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 lastest 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.

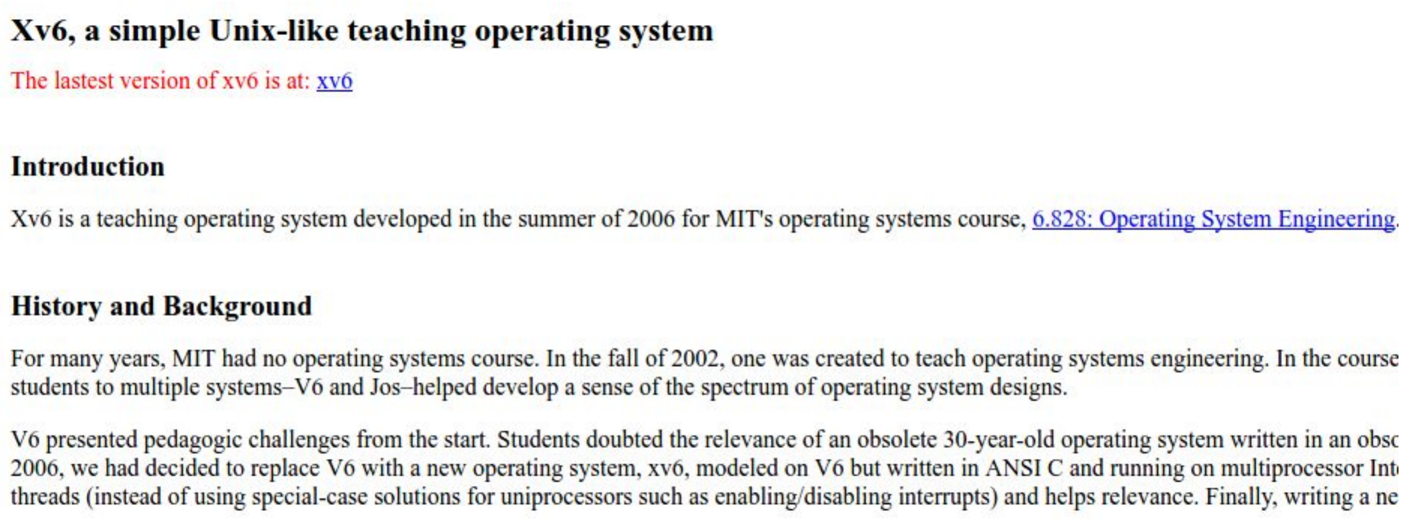
History and Background

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 obsc
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 Int
threads (instead of using special-case solutions for uniprocessors such as enabling/disabling interrupts) and helps relevance. Finally, writing a ne

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 lastest 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 obsc
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 Int
threads (instead of using special-case solutions for uniprocessors such as enabling/disabling interrupts) and helps relevance. Finally, writing a ne

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.

Meet someone new (again)

- Turn to a person near you
- Introduce yourself
  - Where are you from?
  - What are your hobbies?

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
- One final thing
  - Even with the best planning…
  - Some things may change this semester that are beyond our control
  - Everyone (including us) needs to be flexible
  - If you have an issue, it is better to tell us early than at the last minute
- I'm looking forward to being your guide to Computer Systems

Labs

- See the Assignment menu on our Canvas page
- The deliverables from this lab will be part of assignment 1 (released today)
- You need a Khoury ID to get access to the class resources
- Demo of logging in to the class VM