Introduction and Lab 0

CS 0449: Introduction to System Software

SHINWOO KIM
TEACHING ASSISTANT

shinwookim@pitt.edu
https://pitt.edu/~shk148/

University of Pittsburgh | School of Computing and Information

Meta-Notes

- These slides were adapted heavily from recitation slides created by *Martha Dixon* who was a teaching assistant (TA) for this course in Fall of 2020. They contain materials which were obtained from various sources, including, but not limited to, the following:

[1] J. Misurda, CS 0449: Introduction to Systems Software, 3rd ed. Pittsburgh, PA: University of Pittsburgh, 2017.

[2] S. J. Matthews, T. Newhall, and K. C. Webb, Dive into Systems: A Gentle Introduction to Computer Systems. San Francisco, CA: No Starch Press, 2022.

[3] R. Bryant, D. R. O'Hallaron, and M. S., Computer Systems: A Programmer's Perspective. Princeton, NJ: Pearson, 2016.

[4] L. Oliveira, V. Petrucci, and J. Misurda, in Introduction to Systems Software, 2022

Agenda

- Introduction (*1 min*)
- Administrivia & Logistics (*5 min*)
- L0. Hello World! (*40 min*)

Shinwoo Kim

Undergraduate Teaching Assistant

E-mail: `shinwookim@pitt.edu`

Preferred method of communication

Discord: `shinwookim`

Office: TBA; By appt.

Preface subject line with "[CS 449]"

Resources/Technologies for this course

- **Course Website** → Course Materials
  - Syllabus, Lecture Slides, Lab/Project Handouts
  - https://cs0449.gitlab.io/sp2023
- **Recitation Website** → Recitation Materials
  - Recitation Slides, Code Examples
  - https://sites.pitt.edu/~shk148/teaching/CS0449-2241/
- **Discord** → Q&A
  - Assignment Help, Extra Resources, **Announcements**
- **GradeScope** → Assignment Submission and Deadlines

Welcome to CS449!

- Introduction to **Systems** Software.
  - Field of CS dealing with **hardware-software interaction**
- **C** programming language (and **x86 assembly**)
  - Provides **abstraction** from assembly/machine code (syntax)
  - Maintains the **low level access** to memory (pointers)
- **Memory** will be an important topic
  - Von Neumann Architecture

Why is this class (notoriously) hard?

- Lots of concepts

449 is a broad course

Computational Theory	CS 0441: Discrete Structures for CS CS 1502: Formal Methods in Computer Science CS 1510: Theory of Computation
Algorithm Design	CS 1511: Algorithm Design CS 1501: Algorithms and Data Structure I CS 0445: Algorithms and Data Structure II CMPINF 0401: Introduction to Programming
Applications	
Operating Systems	CS 1550: Introduction to Operating Systems
Instruction Set Architecture	CS 0447: Computer Organization & Assembly Language
Logic Design	Computer Engineering & Electrical Engineering
Electrical Design	
Physics	Physics

CS 0449: Introduction to System Software

Why is this class (notoriously) hard?

- Lots of concepts
  - Attend lectures and recitations
    - Watch the video recordings if you absolutely cannot attend in-person
  - Study often
  - Form study groups
- Projects are <u>relatively</u> hard and long
  - Develop good programming skills
    - *Measure twice, cut once!*
  - Comment, comment, and comment
    - Checkoff meetings
  - Start early and show up to office hours!

Recitation

- **Lectures** *present* high-level concepts
- **Recitations** *applies* concepts
  - Clarify lectures/review topics
  - Introduce tools and skills
  - Preparation for labs/projects
    - Labs apply knowledge from lecture
    - Projects provide deeper dive into new skills and concepts
- *Students who don't come to class typically do not do well*
  - Unannounced quizzes/labs to test your understanding

Teaching Philosophy

- A **guide** to help **you** explore concepts.
  - My job isn't to force you to learn.
- Here to help you succeed
  - Get a better understand of the material
    - But also, get a better grade
  - **Ask for help**!
    - You're not expected to know everything. You are learning.
    - No questions are dumb! Don't be afraid to ask them!
    - **DON'T STRUGGLE IN SILENCE.**

DON'T STRUGGLE IN SILENCE.

Poll Everywhere

- Classroom engagement tool
  - Similar to Top-hat
- Helps me (and instructors) gauge your understanding of the content
- Helps you prepare for the exams
  - Preview some exam-style questions
  - Often pulled from previous exams and quizzes

PEV: Binary Representation

```c
#include <stdio.h>
int main()
{
    int x = 0xAC;
    printf("%d", x);
    return 0;
}
```

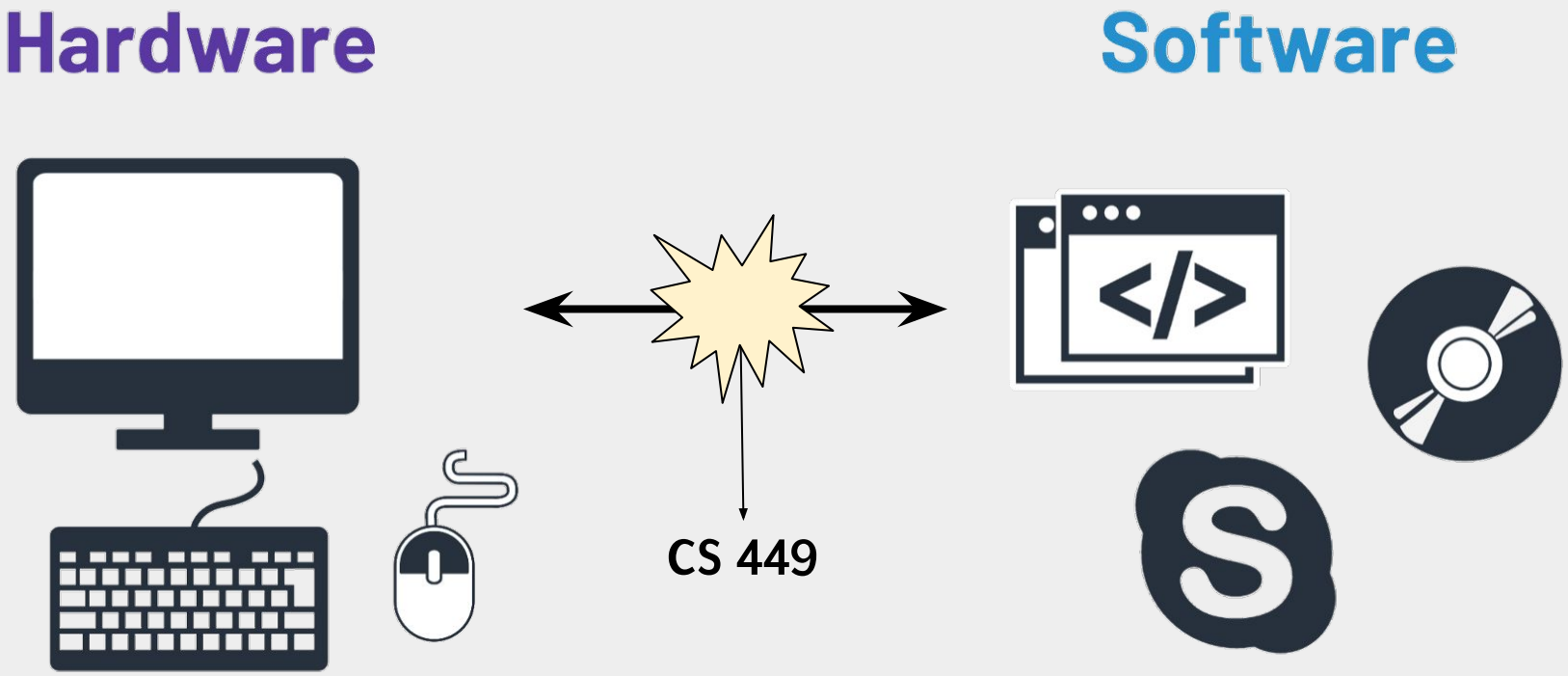
What does the program output? I.e., What is the value of x (in decimal)?

Lab handout on the course website!

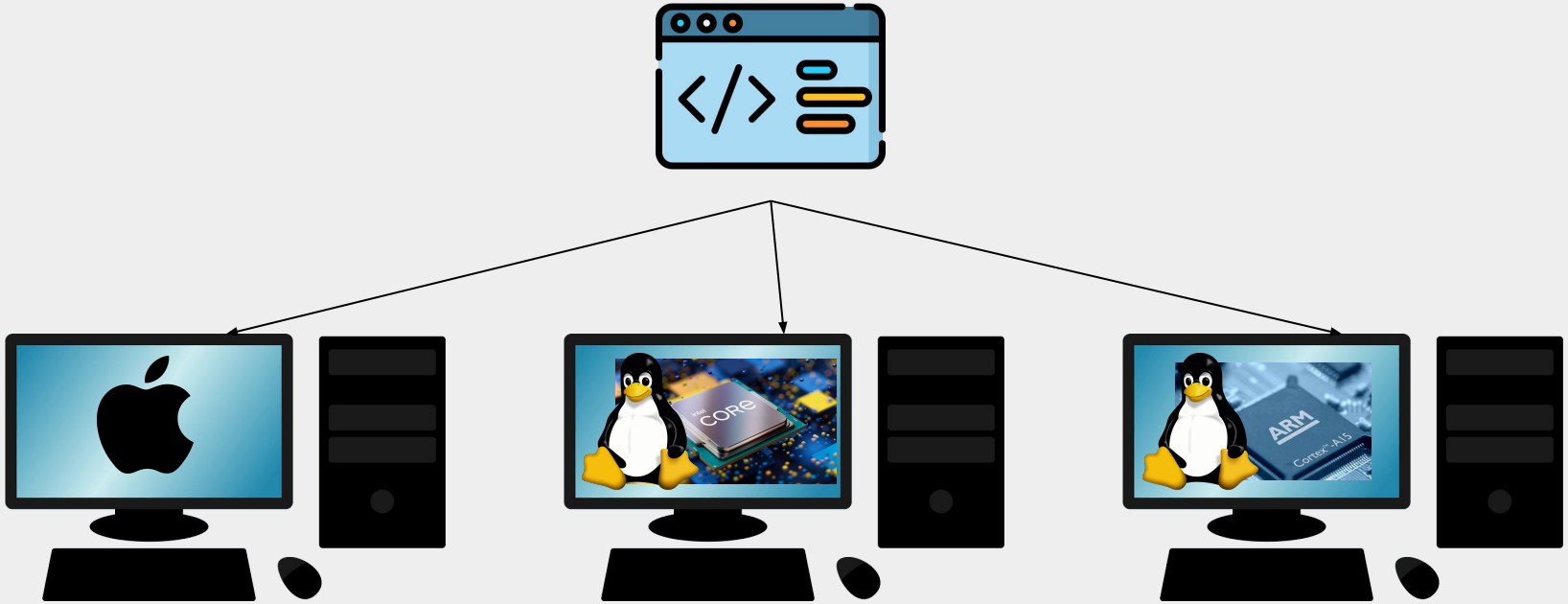
L0. Hello Lab

Getting up to speed with the environment

We will study how Hardware-Software Interact



But Programs can be weird on different machines…

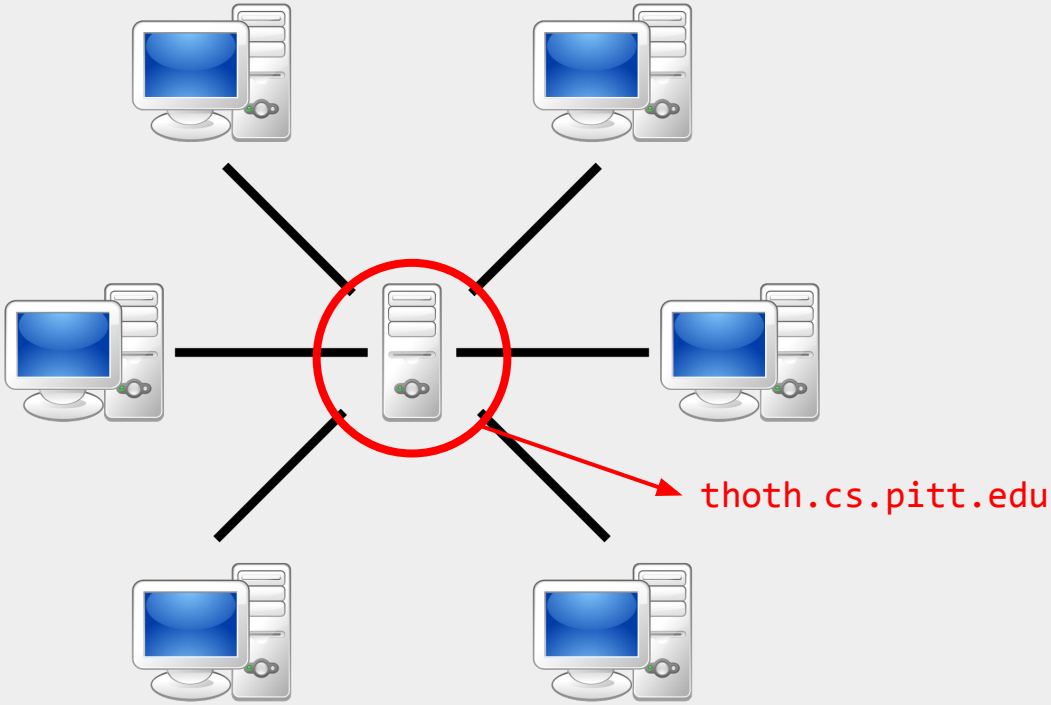



```
ld: library not found for -libfftw3
```

```
Hello, World!
```

```
Hellooooooooooooo
```

Running code in same environment would be nice



L0 Goals

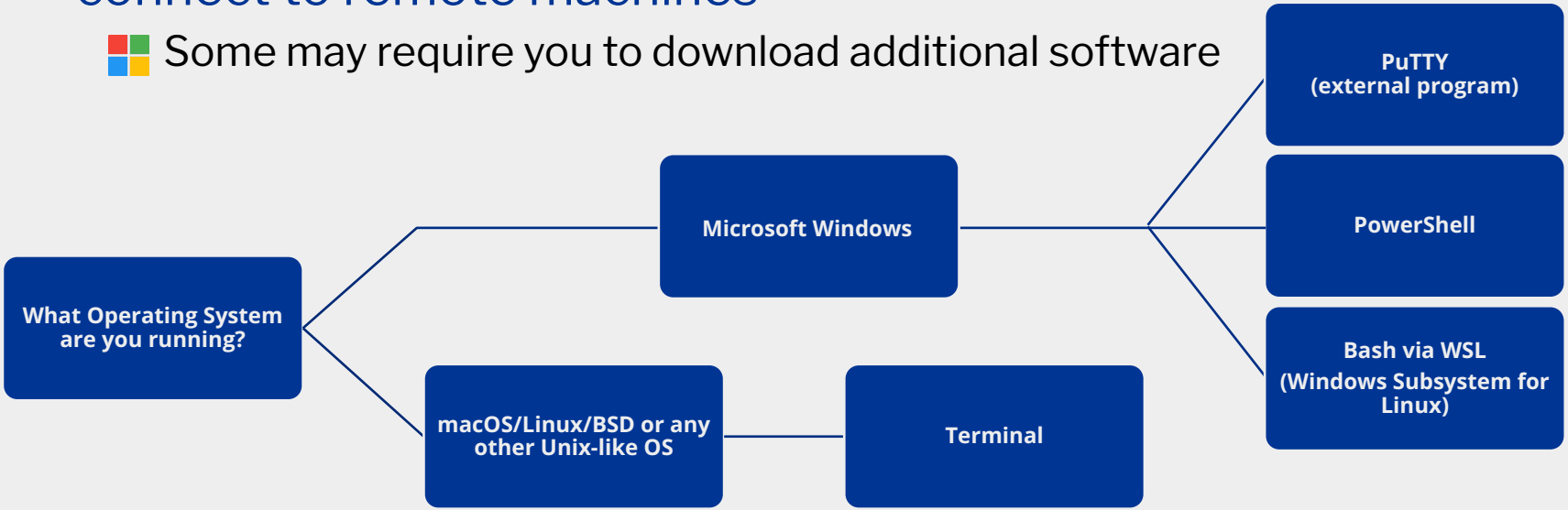
1. Set up Computing Environment
   a. Accessing `thoth.cs.pitt.edu`
   b. Getting familiar with the Shell (command line)
2. Learn to **compile** and run C code on Linux
3. Learn to **debug** C programs using GDB

Thoth

- Server maintained by CS department for students enrolled in *systems* courses CS 449 & CS 1550
  - Run all labs/projects/assignments on it
  - Ensures everyone's code compiles and runs in the same way
    - 'shared environment'
  - Code **must** work on Thoth to receive full credit
    - ~~*"But it works on my computer…"*~~
- **Secure shell** (ssh) to access Thoth

SSH client

- Most operating systems have a built-in ssh client which lets you connect to remote machines
  - Some may require you to download additional software



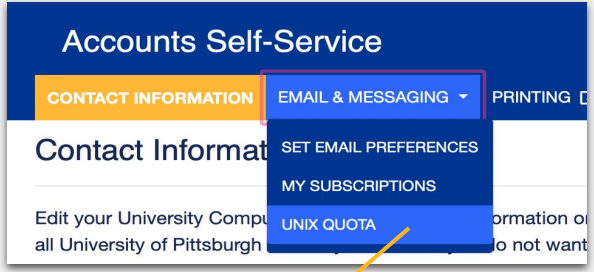
ssh username@thoth.cs.pitt.edu

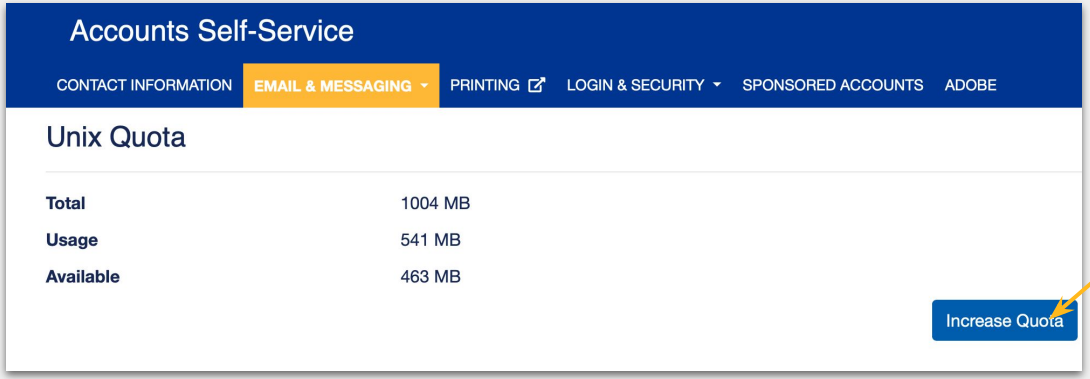
- In your terminal/command-line
  - `ssh <pittid>@thoth.cs.pitt.edu`
    - **E.g.,** `ssh shk148@thoth.cs.pitt.edu`
  - Then use your Pitt university computing account password (same as `my.pitt.edu`)

< … > = Placeholder (Don't include the < >
Backus–Naur form

Increasing Storage Space

- Projects in this course can get pretty big…
- `accounts.pitt.edu`
  1. "Email & Messaging"
  2. "Unix Quota"
  3. "Increase Quota"





Linux Shell

```
shinwoo@ubuntu:~$ bash --version
GNU bash, version 4.3.46(1)-release
```

- A **shell** is just a different interface to your computer!
  - You can do most things you could with a GUI (keyboard + mouse)
    - It's old, but still extremely useful
  - Good for automating tasks
    - (especially if we don't need a graphical environment)
    - E.g.
      - » Renaming 1000 files
      - » For interacting with remote computer (Like Thoth!)

Basic Shell Commands

- / is used as path separators

Directory Shortcuts

/	Root directory
.	Current directory
..	Parent directory (one above)
~	Home directory

```
shinwoo@ubuntu:~$ pwd            # prints the working directory
shinwoo@ubuntu:~$ man <command> # displays the manual for a command
shinwoo@ubuntu:~$ cd             # change directory
shinwoo@ubuntu:~$ ls             # lists directory content
... and so much more
```

Basic Shell Commands

- / is used as path separators

Directory Shortcuts

/	Root directory
.	Current directory
..	Parent directory (one above)
	ectory

https://linuxjourney.com/lesson/the-shell

```
shinwoo@ubuntu:~$ pwd            # prints the working directory
shinwoo@ubuntu:~$ man <command> # displays the manual for a command
shinwoo@ubuntu:~$ cd             # change directory
shinwoo@ubuntu:~$ ls             # lists directory content
... and so much more
```

Editing text files

- To create and modify a file, need to use text editors
- Thoth has several installed
  - **nano (basic, easiest to use)**
  - Vim (popular, steep learning curve)
  - ~~Emacs (popular, steep learning curve, heavily customizable)~~
- Can also use GUI text editors
  - Atom, VS Code, Notepad ++, …
    - *Later…*

Hello World! C edition

```c
#include <stdio.h>
int main (int argc, char* argv[]) {
   //Declare a variable
   int x;
   //Assign a variable
   x = 2;
   //Print a string and a variable
   printf("Hello world! x is currently %d \n", x);
   return 0;
}
```

Hello World! C edition

```c
#include <stdio.h>
int main (int argc, char* argv[]) {
    //Declare a variable
    int x;
    //Assign a variable
    x = 2;
    //Print a string and a variable
    printf("Hello world! x is currently %d \n", x);
    return 0;
}
```

Preprocessor command: tells compiler to include contents of the standard input and output file

Hello World! C edition

```c
#include <stdio.h>
int main (int argc, char* argv[]) {
   //Declare a variable
   int x;
   //Assign a variable
   x = 2;
   //Print a string and a variable
   printf("Hello world! x is currently %d \n", x);
   return 0;
}
```

Standard input/output file:
Contains functions like `scanf()` (take input) and `printf()` (display output)

Hello World! C edition

```c
#include <stdio.h>
int main (int argc, char* argv[]) {
   //Declare a variable
   int x;
   //Assign a variable
   x = 2;
   //Print a string and a variable
   printf("Hello world! x is currently %d \n", x);
   return 0;
}
```

Execution of C files starts from `main()`

Hello World! C edition

```c
#include <stdio.h>
int main (int argc, char* argv[]) {
    //Declare a variab
    int x;
    //Assign a variabl
    x = 2;
    //Print a string and a variable
    printf("Hello world! x is currently %d \n", x);
    r
}
```

Using `printf()` without using `#include <stdio.h>` results in compilation error

Placeholders:
- `%d` int
- `%u` unsigned int
- `%f` float
- `%s` "string"
- `%x` hexadecimal
- `%p` pointer

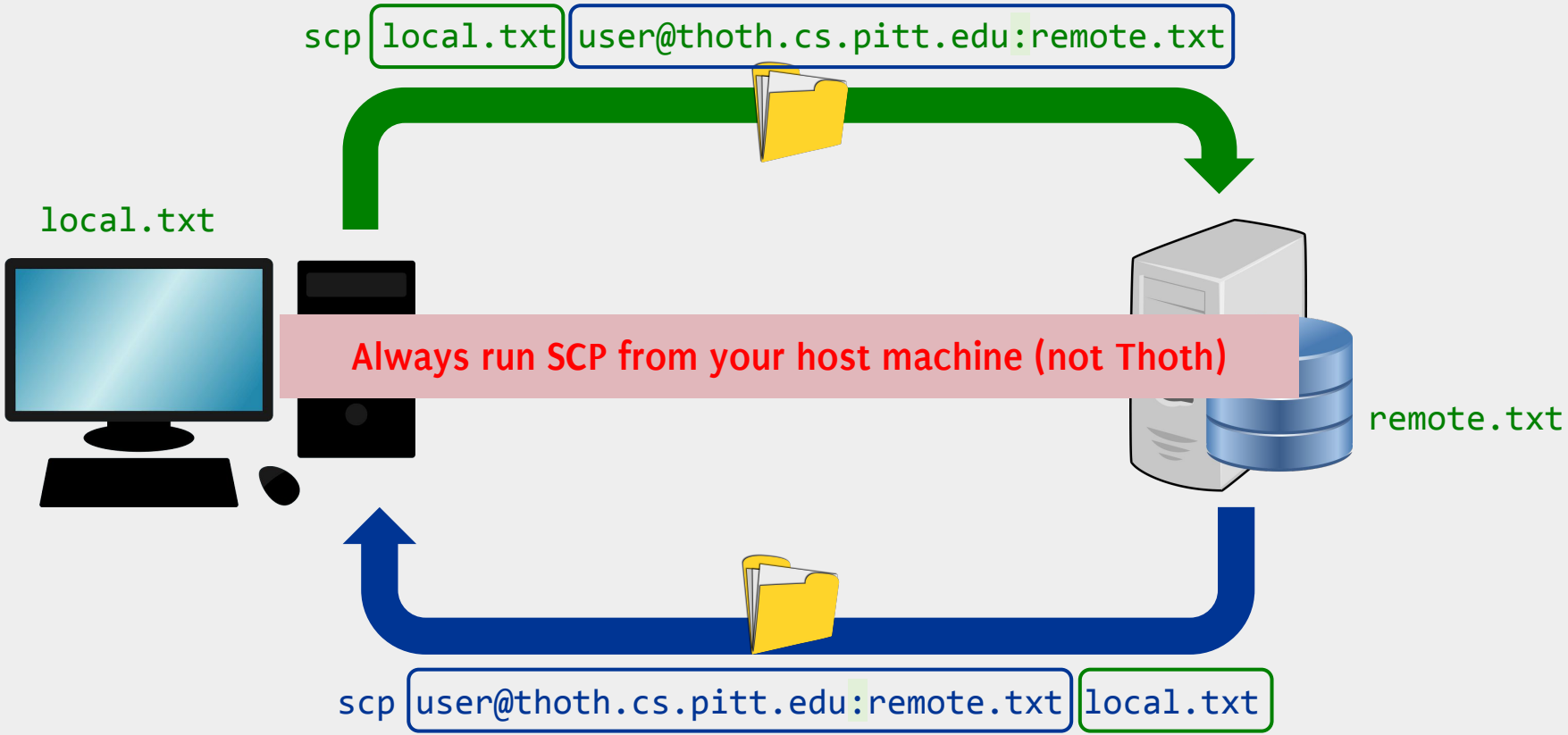
`printf()`: standard library function to send formatted output to screen

Hello World! C edition

```c
#include <stdio.h>
int main (int argc, char* argv[]) {
    //Declare a variable
    int x;
    //Assign a variable
    x = 2;
    //Print a string and a variable
    printf("Hello world! x is currently %d \n", x);
    return 0;
}
```

Exit status: Returning `0` means we exit without error

scp <source> <destination>



Moving files to and from Thoth

- To move files in-and-out of Thoth, we can use Secure Copy (SCP)
  - To copy `local.txt` to Thoth (saved as `remote.txt` in home directory of Thoth)
    - `scp local.txt user@thoth.cs.pitt.edu:remote.txt`
  - To copy `remote.txt` from Thoth (saved as `local.txt` on your device)
    - `scp user@thoth.cs.pitt.edu:remote.txt local.txt`
  - **Always run SCP from your host machine (not Thoth)**
- Let's check the contents of `main.c`
  - `cat main.c`
- We can make adjustments using nano
  - `nano main`

Making sure the file transferred correctly

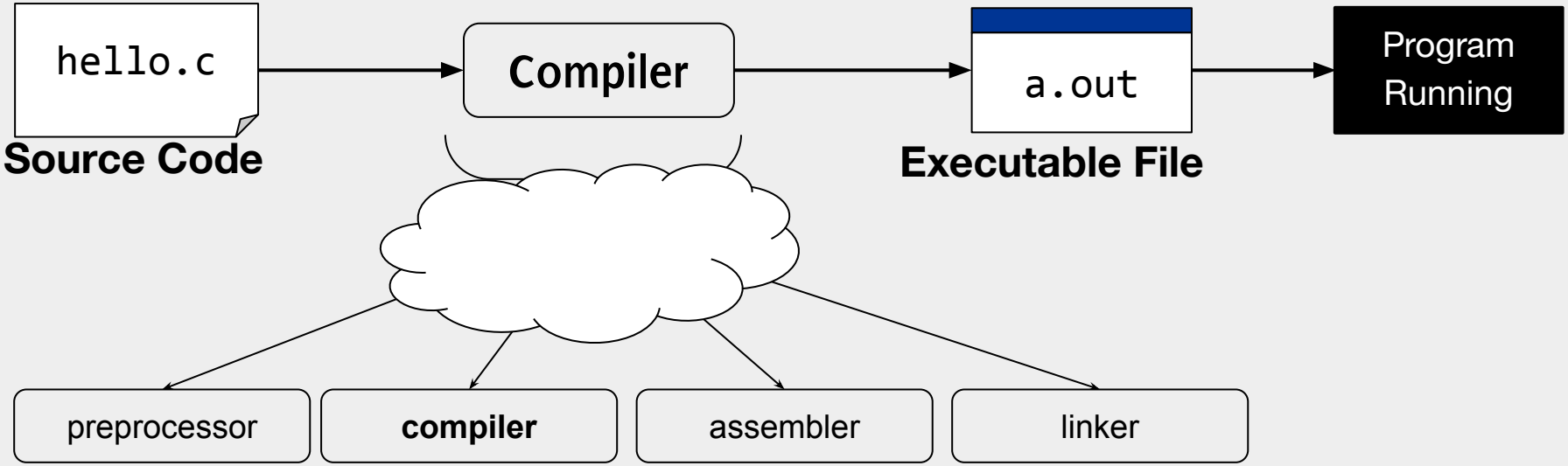
- Check contents of main.c
  - `cat main.c`
- Make adjustments using nano
  - `nano main`

nano: A GNU text editor

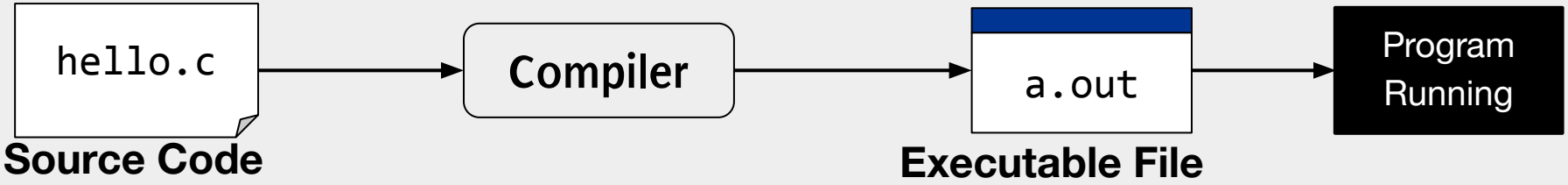
```
  GNU nano 2.0.9            File: txt_files/testfile                    Modified

Learn how to use nano to boost your terminal confidence!
Edit config files like a pro!
Make easy to-do lists and notes in a text-only format!
Do it via SSH from a smartphone or other computer!

# /etc/fstab: static file system information.
#
# Use 'blkid -o value -s UUID' to print the universally unique identifier
# for a dev                                                          name
# devices t                                                          (5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
# / was on /dev/sdb1 during installation
                                [ Read 17 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

[^ = ctrl] ⇒ [Exit = ctrl + X]

Life of a C Program

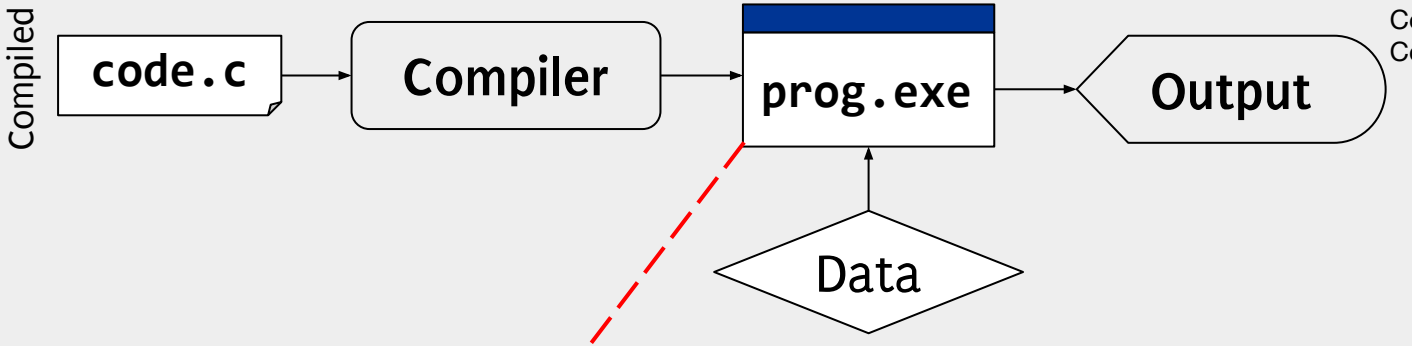


Life of a C Program

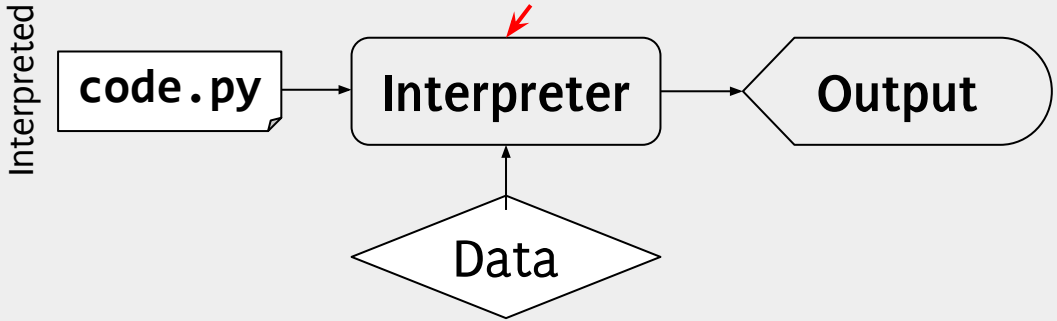


- Before code can execute, must first be ***compiled***
- Compilation: *translating HLL source code → machine language*
- The executable file is a binary file that contains the machine code (CPU instructions) of the program
  - + data, …

Interpreted vs Compiled



Fast Execution
Exploit architectural features
Complex
Compile *before* execution



Machine independent
Easy to debug, modify
Space & Time overhead

Let's *compile* our first C program

- Compile using GCC (Gnu Compiler Collection)
  - `$ gcc -Wall -g -std=c99 -o hello main.c`
- Run our Hello World program
  - `$ ./hello`

ile o first C m

-W: sets warnings

-Wall: enable all warnings

-g: turns on debugging symbols

-o: used for renaming executable

What you want the executable to be named

e using gcc (Gnu Compiler Collection)

- `gcc -Wall -g -std=c99 -o hello main.c`

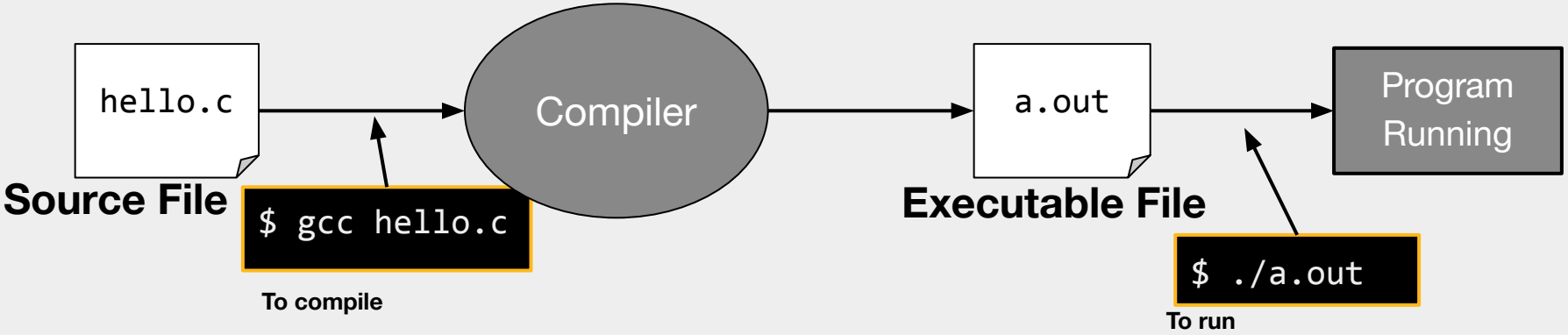
Source file's name

-std: sets what version of C is being used

-std=c99: use C99

- At last, we ca o World program
  - `./hello`

Run the executable



Submitting Lab 0A

- Rename `main.c` to `username_lab0.c`
  - `mv main.c shk148_lab0.c`
    - `mv` command is also used for moving files

- Copy `username_lab0.c` to Dr. Oliveira's folder
  - `cp shk148_lab0.c /afs/pitt.edu/home/l/u/lun8/public/lab0/submissions`

- To check if lab was submitted correctly, run:
  - `/afs/pitt.edu/home/l/u/lun8/public/lab0/materials/check_submission.sh shk148`

Summary

1. Slides and Materials on Recitation Website: https://sites.pitt.edu/~shk148/teaching/
2. Join Discord before the link expires
3. Lab0B ← *Do this!*
4. Office hours will be posted (...soon™)

Shinwoo Kim

shinwookim@pitt.edu