Objectives

- Reviewing lab
- Introduction to
  - Problem solving
  - Algorithms
  - Programming languages

Typical Class Period Organization

1. Pearls of wisdom from Professor Sprenkle
2. Review course material in pods
   - Consult your notes, handouts, slides from recent classes (see course web site)
3. Review as a class
4. New stuff!
   - Some think-pair-share work

Course Logistics

- Handouts
  - Slide number won’t always line up with projected slides
  - Won’t always get to all
  - Don’t look ahead
- Office hours this week
  - Today: 2:45-4:45 p.m. (later than normal time)
  - Tomorrow: 10:30-11:30, 1:00-2:30 p.m.
  - Join on Zoom or stop by my office and I’ll join you in the advanced lab

Review: Lab

- Learned some UNIX commands
- Created a Web page
- Lessons learned:
  - Problems are fixable (often just typos!)
  - No need to say you’re “sorry”. You’re learning!
  - Learn from, adapt examples
  - Find a good solution
  - Honing your mental model

Review: UNIX

- UNIX is a bad coach
  - Doesn’t tell you when you’ve done something right
  - Only tells you when you’ve done something wrong

Terminal:

```
sprenkle@spartacus Desktop$ cp lab00.ppt.pdf lab00.pdf
sprenkle@spartacus Desktop$
```

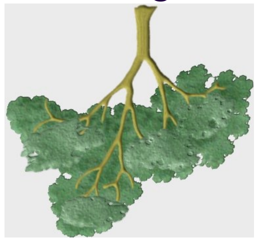
Did it work? Maybe.
While you’re learning, need to check/confirm it!

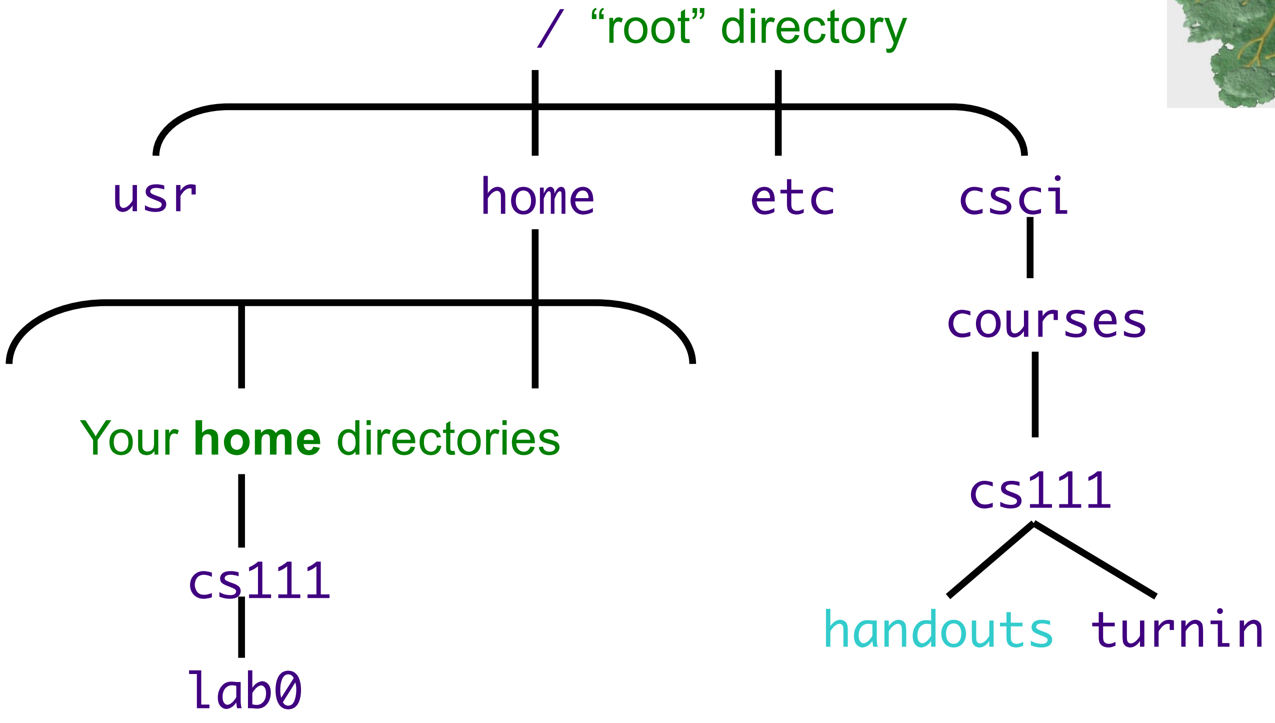
Review: Linux

- What is the syntax of the command to
  - List the files in a directory?
  - Change your current directory?
  - Make a directory?
  - Find out the current directory?
  - Make a copy of a file?
- What is the shortcut to refer to
  - The current directory?
  - The parent directory?
  - Your home directory?
- What is the difference between an *absolute* path and a *relative* path?
  - How do you know if a path is an *absolute* or *relative* path?
- What is the *absolute path* to your home directory?

(Partial) Linux File Structure

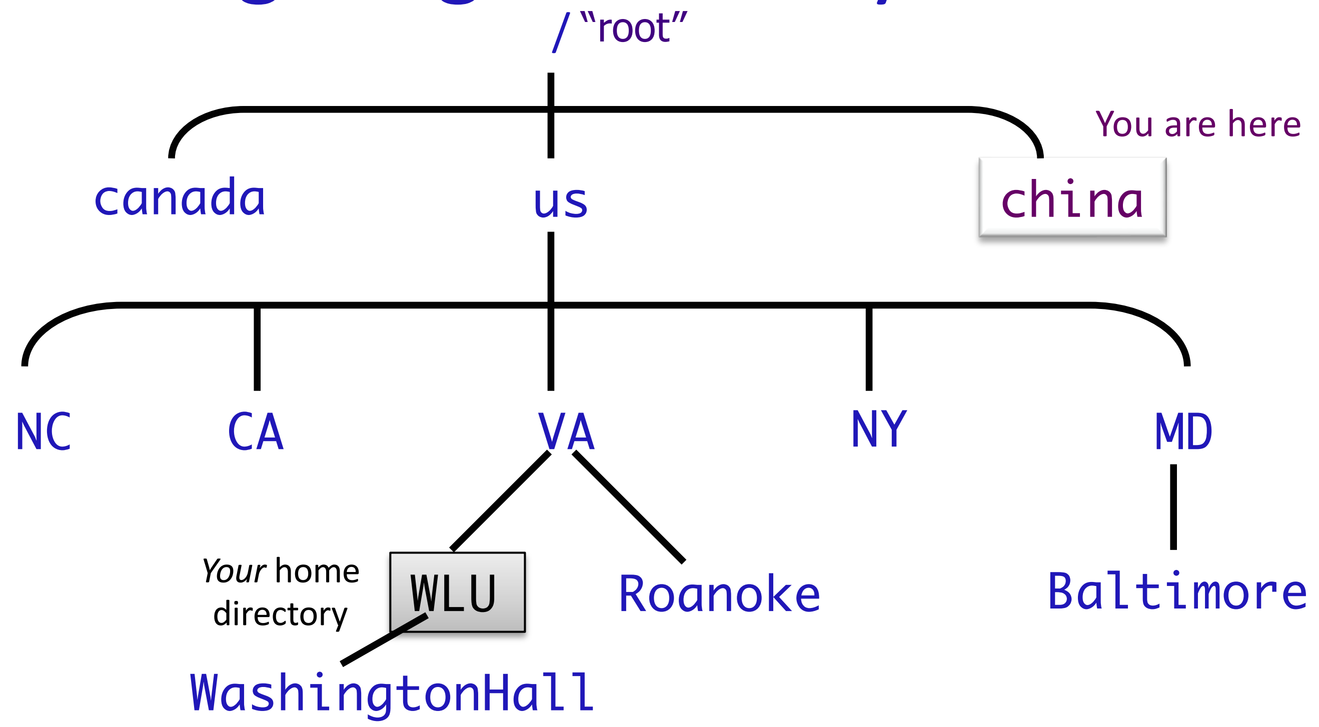
Paths through tree





What is the *absolute* path to the handouts directory?

Review: Navigating the File System



Given that you're in /china,
how would you go to canada? WLU? Washington Hall?

Hilary Mason

- Founder of Fast Forward Labs
  - a machine intelligence research company
- Formerly Chief Scientist at bitly
- "Teaching someone to program is like giving them a **superpower**."

What This Course Is About

Problem Solving!

From *30 Rock*

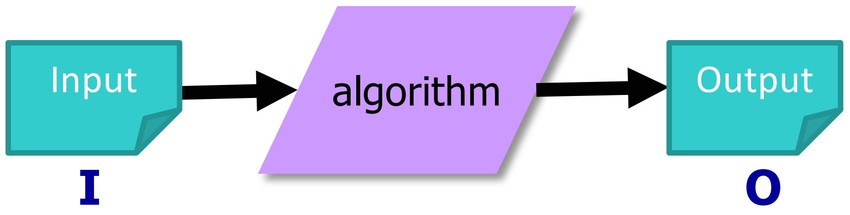
Computational Problem Solving 101

- **Computational Problem**:
  A problem that can be solved by logic
- To solve the problem:
  1. Create a **model** of the problem
  2. Design an **algorithm** for solving the problem using the model
  3. Write a **program** that *implements* the algorithm

Computational Problem Solving 101

- **Algorithm**: a well-defined, step-by-step process for solving a problem
  - Has a *finite* number of steps
  - Completes in a *finite* amount of time
- Program
  - An algorithm written in a **programming language**
  - Also called *code*
  - As code base grows, becomes an *application*

Algorithms: Input and Output

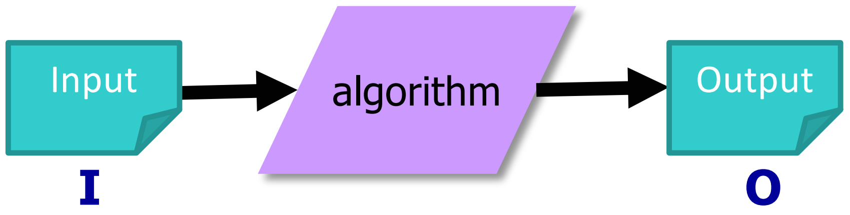


- Algorithms often have a defined **input** and **output**
- Correct algorithms give the intended output for a set of input
- Example: Multiply by 10
  - I/O for a correct algorithm:

Input	Output
5	50
.32	
x	

- More examples
  - averaging numbers, recipes

Algorithms: Input and Output



- Algorithms often have a defined **input** and **output**
- Correct algorithms give the intended output for a set of input
- Example: Multiply by 10
  - I/O for a correct algorithm:
- More examples
  - averaging numbers, recipes

Input	Output
5	50
.32	3.2
x	10x

Making a Peanut Butter & Jelly Sandwich

- How do you make a peanut butter and jelly sandwich?
- Write down the steps so that someone else can follow your instructions
  - Make no assumptions about the person's knowledge of PB&J sandwiches
  - The person has the following materials:
    - Loaf of bread, Jar of PB, Jar of Jelly
    - 2 knives, a paper plate, napkins
- Algorithm: What is the input? What is the output?

Discussion of PB&J

- The computer: a blessing and a curse
  - Recognize and meet the challenge!
- Be unambiguous, descriptive
  - Must be clear for the computer to understand
  - "Do what I **meant**! Not what I said!"
    - Motivates programming languages
- Creating/Implementing an algorithm
  - Break down pieces
  - Try it out
  - Revise

Discussion of PB&J

- Steps need to be done in a particular order
- Be prepared for special cases
    - Any other special cases we didn’t discuss?
- Aren’t necessarily spares in real life
    - Need to write correct algorithms!
- Reusing similar techniques
    - Do the same thing with a little twist
- Looping
    - For repeating the same action

Looking Ahead

- Lab 0 due Friday