

# Lab 0

Professor Sprenkle  
sprenkles@wlu.edu  
January 8, 2008

## Objectives

- Intro to Labs
- Intro to Operating Systems
- Why programming languages?
- Start Lab #0
  - UNIX/Linux intro, worksheet
  - Portal account (Blog for "Broader CS Issues")
  - Create Web page
  - Use jEdit (Text Editor), IDLE
  - Use Python interpreter in interactive mode
  - Write Python programs

A lot of different things but doable!

Jan 8, 2008

Sprenkle - CS111

2

## Intro to Labs

- Introduce our Student Assistants
- 3 hours to get started on labs
  - Often will need to finish lab after lab period
  - Use this lab (P405), preferably, or P413

Jan 8, 2008

Sprenkle - CS111

## Operating Systems

- Manage hardware resources
- Three popular operating system variations:

Mac	PC/Windows	UNIX
-----	------------	------

  - Compare in terms of cost, popularity, available software, security
- Learn Linux (a UNIX variation) in this class

Note: "PC" for Windows is a misnomer because all of these OSs are for "personal computers".

Jan 8, 2008

Sprenkle - CS111

## P405 Machines

- Run both Linux and Windows
  - Linux natively
  - Windows virtually
- To switch between OSs use
  - Ctl-Alt-F7 for Linux
  - Ctl-Alt-F9 for Windows
  - (May be backwards sometimes)
- P413 - Linux-only

Jan 8, 2008

Sprenkle - CS111

## Pause While You Log In

- Follow handout's instructions
- Open browser
- Navigate to Lab 0, from course's "Schedule" page
  - We're starting on the first objective "Learning to Use the Linux Machines" on paper
  - Return to Web page for rest of lab
- How different is the UI than Windows or Mac?

Jan 8, 2008

Sprenkle - CS111

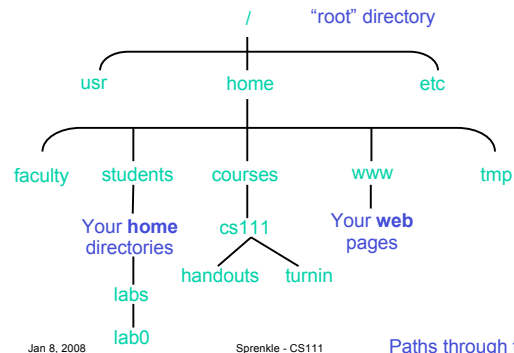
## Intro to UNIX

- Can execute operations by typing commands in terminals or using GUIs
  - We will use terminals most of the time
  - Today: learn essential UNIX commands
- File structure
  - Organize our files
  - Hierarchy of directories ("folders" in Windows world)

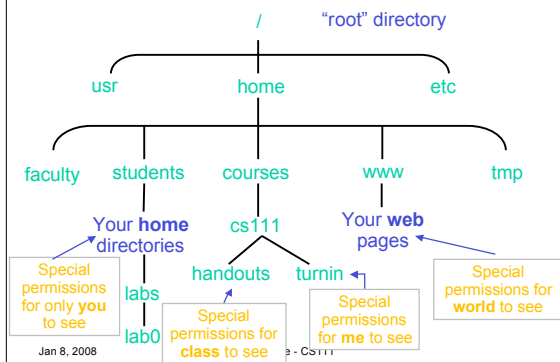
Jan 8, 2008

Sprenkle - CS111

## (Partial) Linux File Structure



## (Partial) Linux File Structure



## Intro to UNIX: Essential Commands

- Manipulating Files
  - **ls** - list the files, directories in a directory
  - **mkdir** - make a directory
  - **cp** - copy a file/directory
  - **mv** - move a file/directory
  - **rm** - remove (delete) a file/directory
- Navigating Directories
  - **pwd** - "print working directory"
  - **cd** - "change directory"

Jan 8, 2008

Sprenkle - CS111

## Intro to UNIX: Shortcuts

- **.**
    - Current directory
  - **..**
    - Parent directory
- Often used with **cp**, **mv**, **cd** commands
- ```
graph TD
    root["/"] --> home
    home --> courses
```
- **cd** or **cd ~**
    - Change to your HOME directory

Jan 8, 2008

Sprenkle - CS111

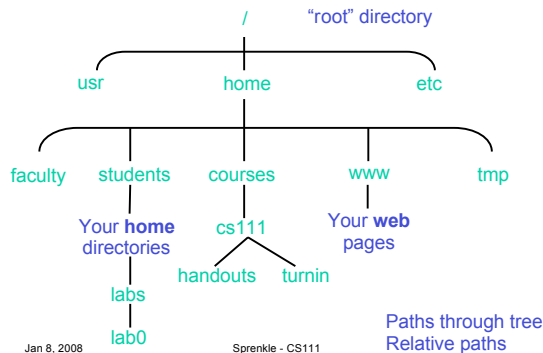
## Synchronizing ...

- Everyone started Linux worksheet
- Open a new terminal using your shortcut in the top bar

Jan 8, 2008

Sprenkle - CS111

## (Partial) Linux File Structure



## Why Do We Need Programming Languages?

- Computers can't understand English
  - Too ambiguous
- Humans can't easily write machine code

Problem Statement (English)

Machine code/Central Processing Unit (CPU)

000000 00001 00010 00110 00000 100000

Jan 8, 2008

Sprenkle - CS111

## Why Do We Need Programming Languages?

- Computers can't understand English
  - Too ambiguous
- Humans can't easily write machine code

Programmer (YOU!)  
translates from  
problem to algorithm  
(solution) to program

Python interpreter  
translates into  
bytecode

Problem Statement (English)

Algorithm/Pseudocode

High-level Programming Language (Python)

Bytecode

Machine code/Central Processing Unit (CPU)

Jan 8, 2008

Sprenkle - CS111

## Why Do We Need Programming Languages?

- Computers can't understand English
  - Too ambiguous
- Humans can't easily write machine code

Problem Statement (English)

Algorithm/Pseudocode

High-level Programming Language (Python)

Bytecode

Machine code/Central Processing Unit (CPU)

Python interpreter  
executes the  
bytecode in a "virtual  
machine"

Jan 8, 2008

Sprenkle - CS111

## Python Is ...

- A programming language
- An interpreter (which is a program) that executes Python code

Jan 8, 2008

Sprenkle - CS111

## Python

- A common *interpreted* programming language
  - Runs on many operating systems
- First released by Guido van Rossum in 1991
- Named after *Monty Python's Flying Circus*
- Minimalist syntax, emphasizes *readability*
- Flexible, fast, useful language
- Used by scientists, engineers, systems programmers

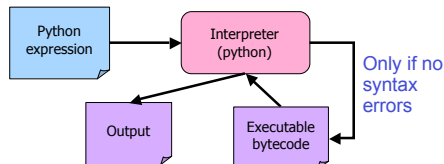
Jan 8, 2008

Sprenkle - CS111

18

## Python Interpreter

1. Validates Python programming language expression(s)
  - Enforces Python syntax rules
  - Reports syntax errors *Have a lot of these early on!*
2. Executes expression(s)



Jan 8, 2008

Sprengle - CS111

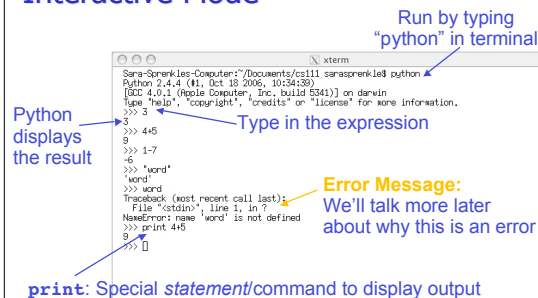
## Two Modes to Execute Python Code

- **Interactive:** using the interpreter
- **Batch:** execute *scripts* (i.e., files containing Python code)

Jan 8, 2008

Sprengle - CS111

## Interactive Mode



Jan 8, 2008

Sprengle - CS111

## Your Turn...

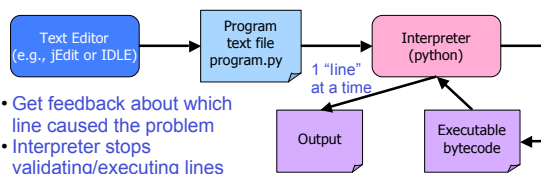
- Run the Python interpreter
- Enter the following expressions and see what Python displays:
  - > 3
  - > 4 \* -2
  - > -1+5
  - > 2 +
  - > print "Hello!"
- To quit the shell, use Control-D

Jan 8, 2008

Sprengle - CS111

## Batch Mode

1. Programmer types a *program/script* into a **text editor** (jEdit or IDLE).
2. An **interpreter** turns each expression into *bytecode* and then executes each expression



Jan 8, 2008

Sprengle - CS111

23

## Example Python Script

- What does this program do?

*Text file named: hello.py*

```

# Program that prints out "Hello, world!"
# by Sara Sprengle
# Last modified: 01/08/2008

print "Hello, world!"
    
```

*Print statement*

- > Validate your guess by executing the program
  - Go into labs/lab0 directory
  - **python hello.py**

Jan 8, 2008

Sprengle - CS111

## Example Python Script

```
# Program that prints out "Hello, world!"
# by Sara Sprenkle
# Last modified: 01/08/2008
print "Hello, world!"
```

} Documentation  
-- good style

- Only "Hello, world!" is printed out
- Python ignores everything after the "#"
  - Known as "comments" or, collectively, as **documentation**
- Your program should always start with a high-level description of what the program does, your name, and the date the program was written

Jan 8, 2008

Sprenkle - CS111

## Practice in Interactive, Batch Modes

- Open the IDLE development environment
  - Command: **idle &**
  - **&** Runs command in "background" so you can continue to use the terminal
  - Given that our programming language is named after Monty Python, any ideas what the development environment is named after?

Jan 8, 2008

Sprenkle - CS111

## Lab 0 Checklist

- Linux Worksheet
- Portal access
- Web Page
- Python practice, programs
- Print lab

Jan 8, 2008

Sprenkle - CS111