Week One Lecture

Welcome to:

CSCI 3308 Software Development
Methods and Tools

Instructor:

Chris Womack
christopher.womack@colorado.edu

Agenda

1. Introductions

2. Overview of Course (Syllabus)

3. Topic One: The UNIX environment
    - Overview of UNIX
    - Command Shell
    - Preview of Lab1

Introductions

Who am I?

How did I come to be here as an instructor?

Introductions

Teaching Assistant:

- None

Course Assistants:

- Bum Soo Kim – Located in the CSEL - TBA

Grader:

- Yawen Zhang - Yawen.Zhang@colorado.edu

Moodle Enrollment

- https://moodle.cs.colorado.edu/course/view.php?id=163
- CSCI 3308 Software Development Methods and Tools
- Enrollment key: **agile**
- Self Enrollment

A note on your development environment:

Your work for this course will be done on a VM that you install on your PC as part of Lab 1.

The VM is a “server” that runs under your PC’s operating system. Your VM will run a version of Linux. Your interface with your VM through a “command shell” Window.

The VM software allocates PC resources to your VM (memory, disk, CPU, network connectivity.)

For our database work, you will install MySQL DBMS software on your VM.

Concepts

Software Development: (methods and tools)

Project Management

Software is developed within the context of a PROJECT

- There are different methods of managing and executing projects
- Waterfall versus Agile

Version Control

Software is developed within the context of VERSIONS

- Code is kept in a repository and tracked by version
- Developers check out code modules, work on them, then check them back into the repository when ready
- We use software tools to manage repositories and module version control

Software Development: (methods and tools)

Code Analysis and Improvement

- Software must not only work right, but it must be efficient and play nicely
- We use tools to analyze code for performance
- How efficiently does the code use memory, CPU, disk I/O

DBMS (Database Management System)

- Application software processes data
- That data is stored, retrieved, updated via database software
- We use SQL as the programming language to interact with the DBMS

Concepts

Software Development: (methods and tools)

Presentation

- Application software must interface with users via a presentation layer
- What's the user's platform? Mobile versus PC versus browser

Testing

- Application software must be tested to ensure that it satisfies the customers' features/requirements
- There are many strategies for and levels of testing

Software Development: (methods and tools)

Debugging

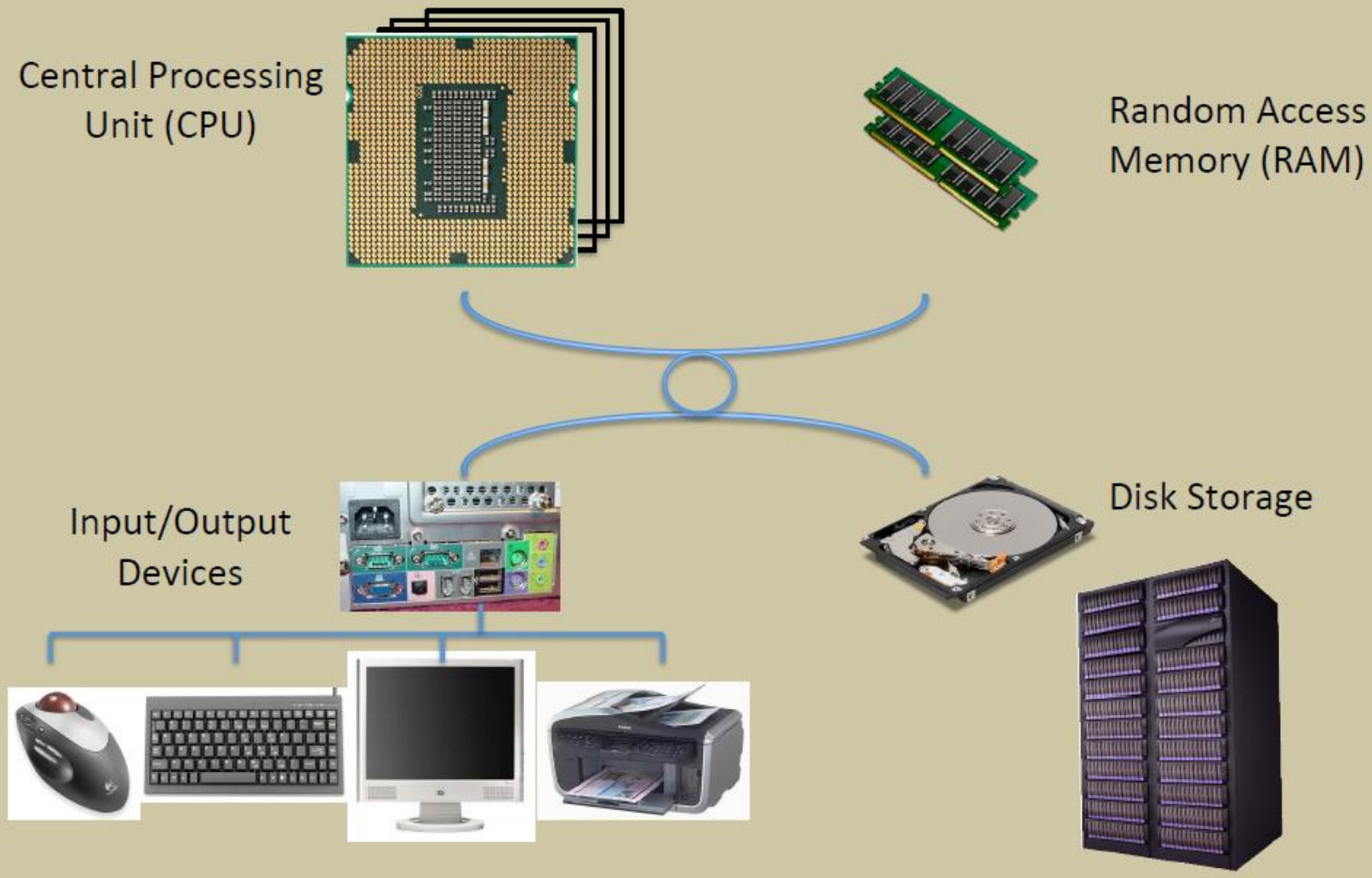
- During testing we identify bugs – that is, the software fails to work right
- It may stop, fail, corrupt the data, produce incorrect results, etc.
- We have tools to help us identify the source of those failures within the code

Documentation

- Someday someone will have to read your code in order to modify it or fix it
- We use tools to assist us in thoroughly documenting our code

Computer Architecture

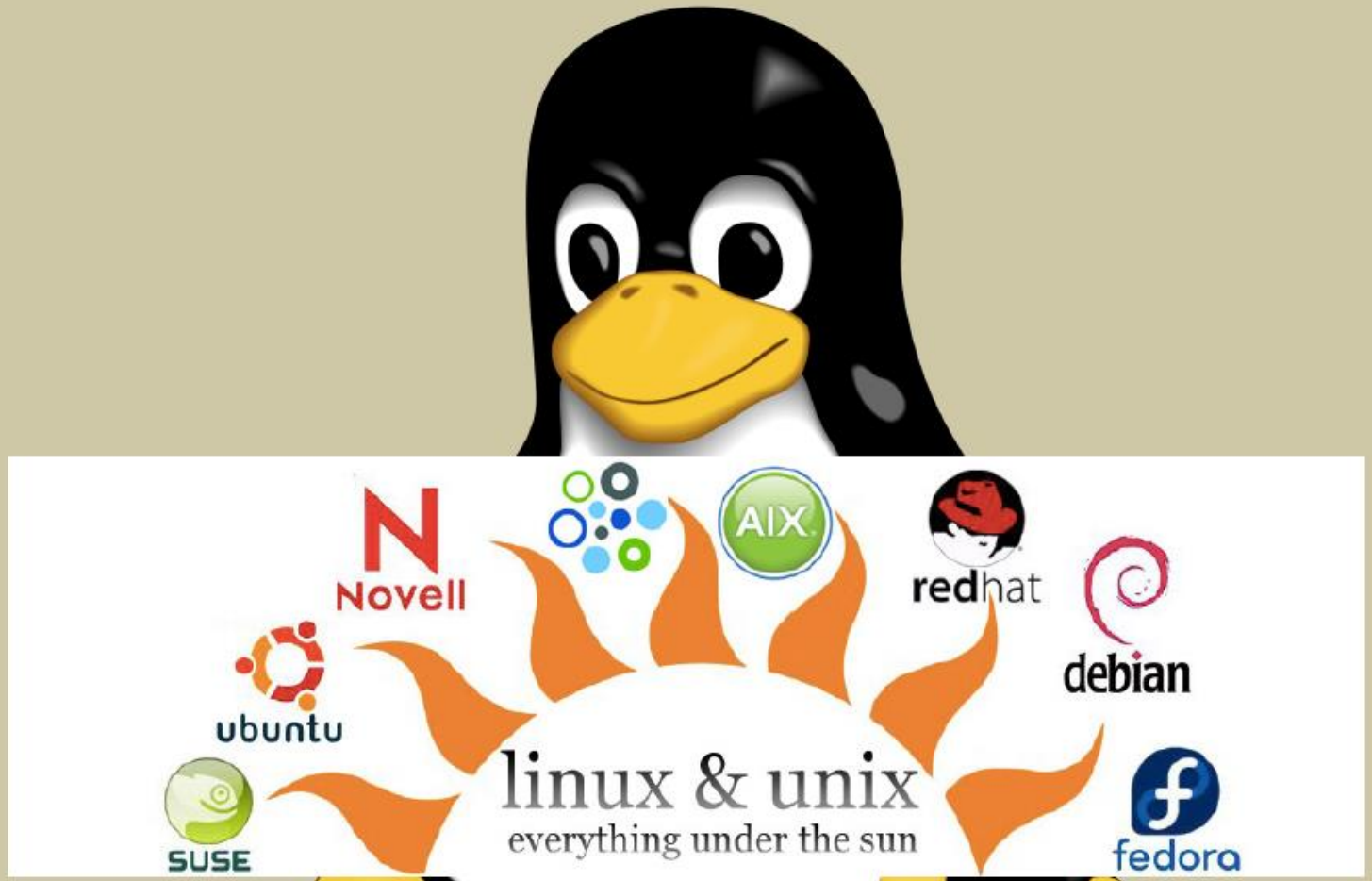
Unix



What is an Operating System?

- Hardware components are only able to control themselves (CPU, RAM, DISK)
- Components do not know how to interact with each other.
- How do we get components to operate together?

Add an Operating System



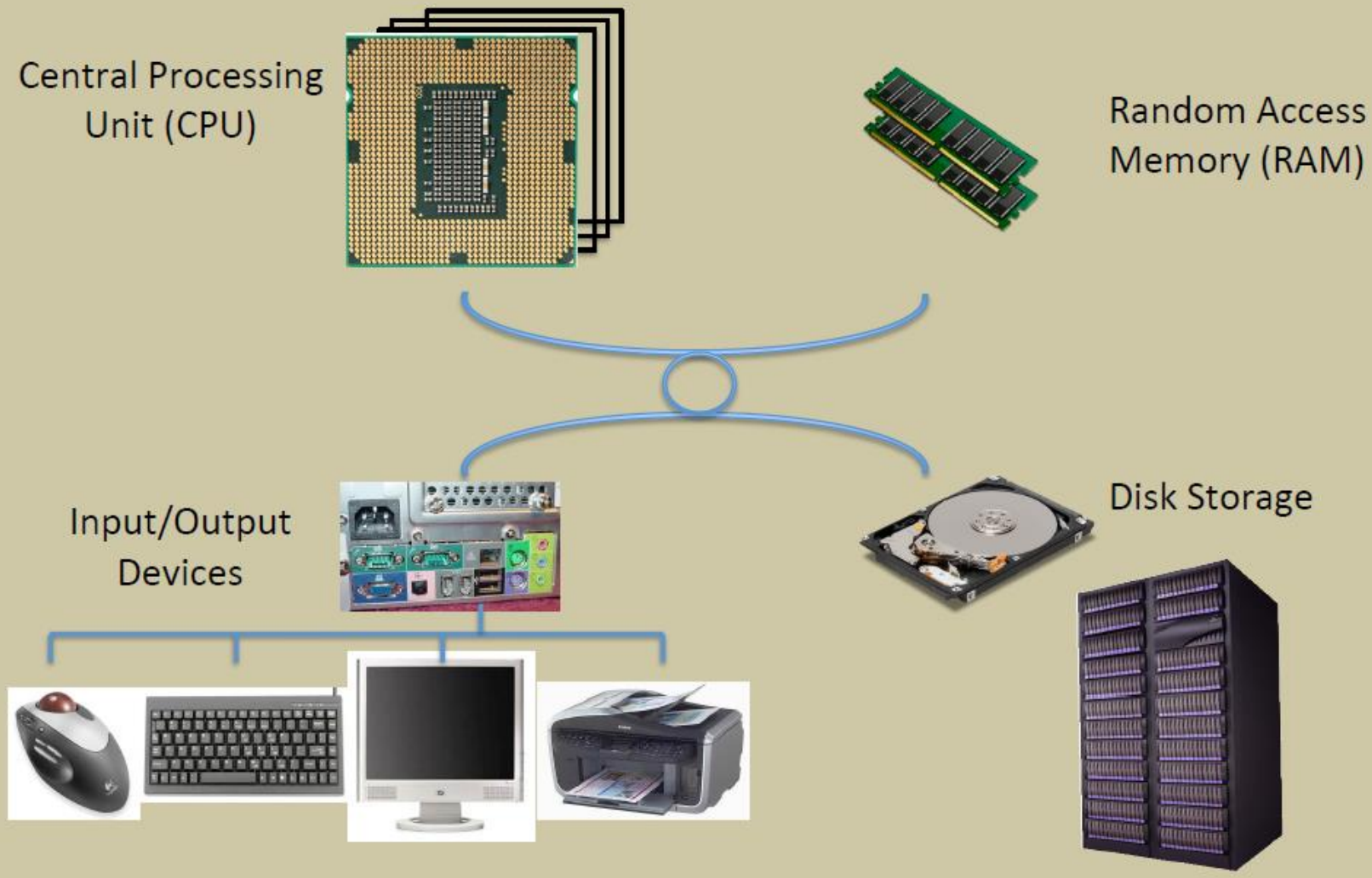
Is it Unix or is it Linux?

- Unix is a trademarked word
- The operating system originally designed in the late 60's was reborn again as open source
- Linux contains many of the original features and quirks of the Unix operating system

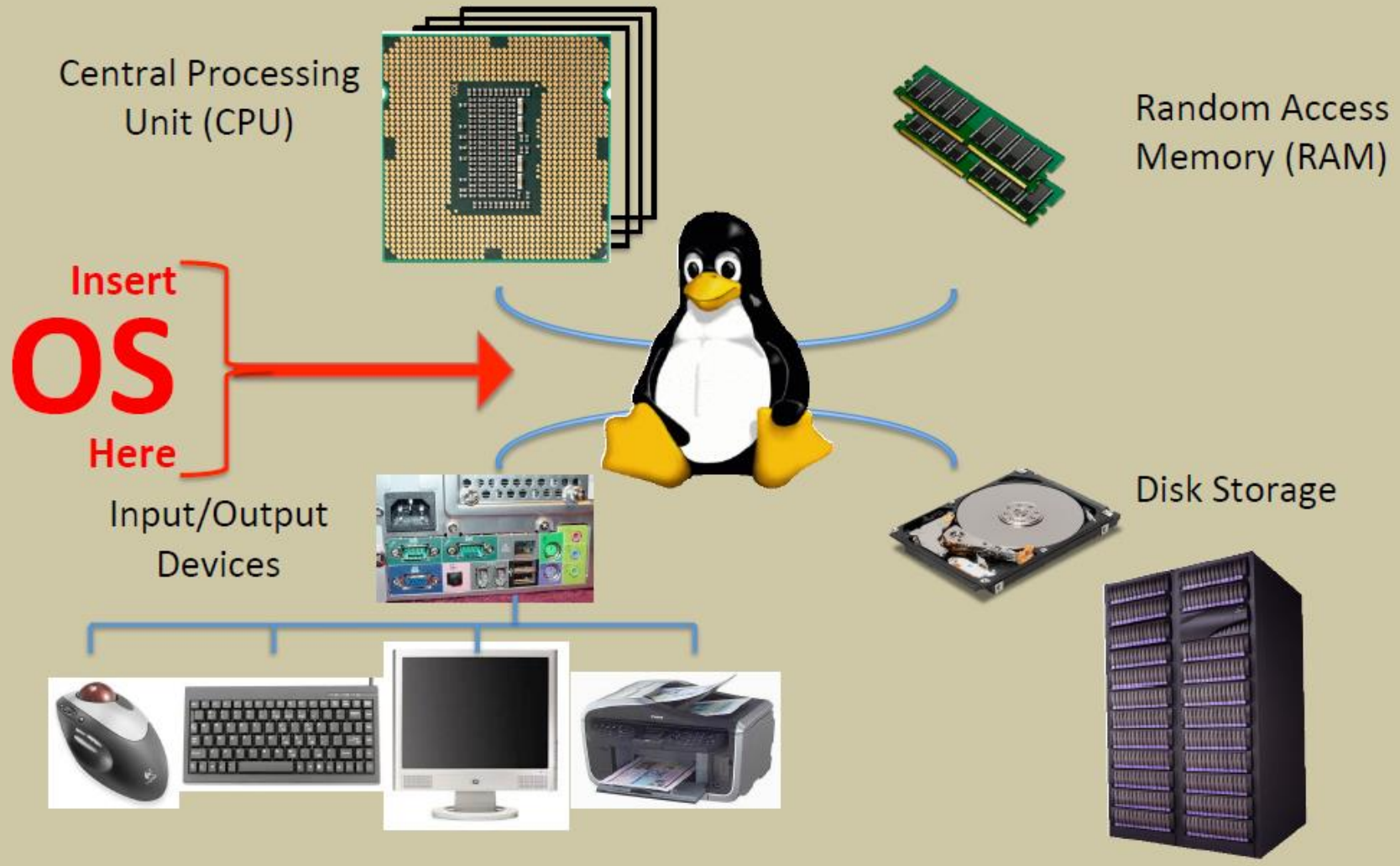
UNIX®

Computer Architecture

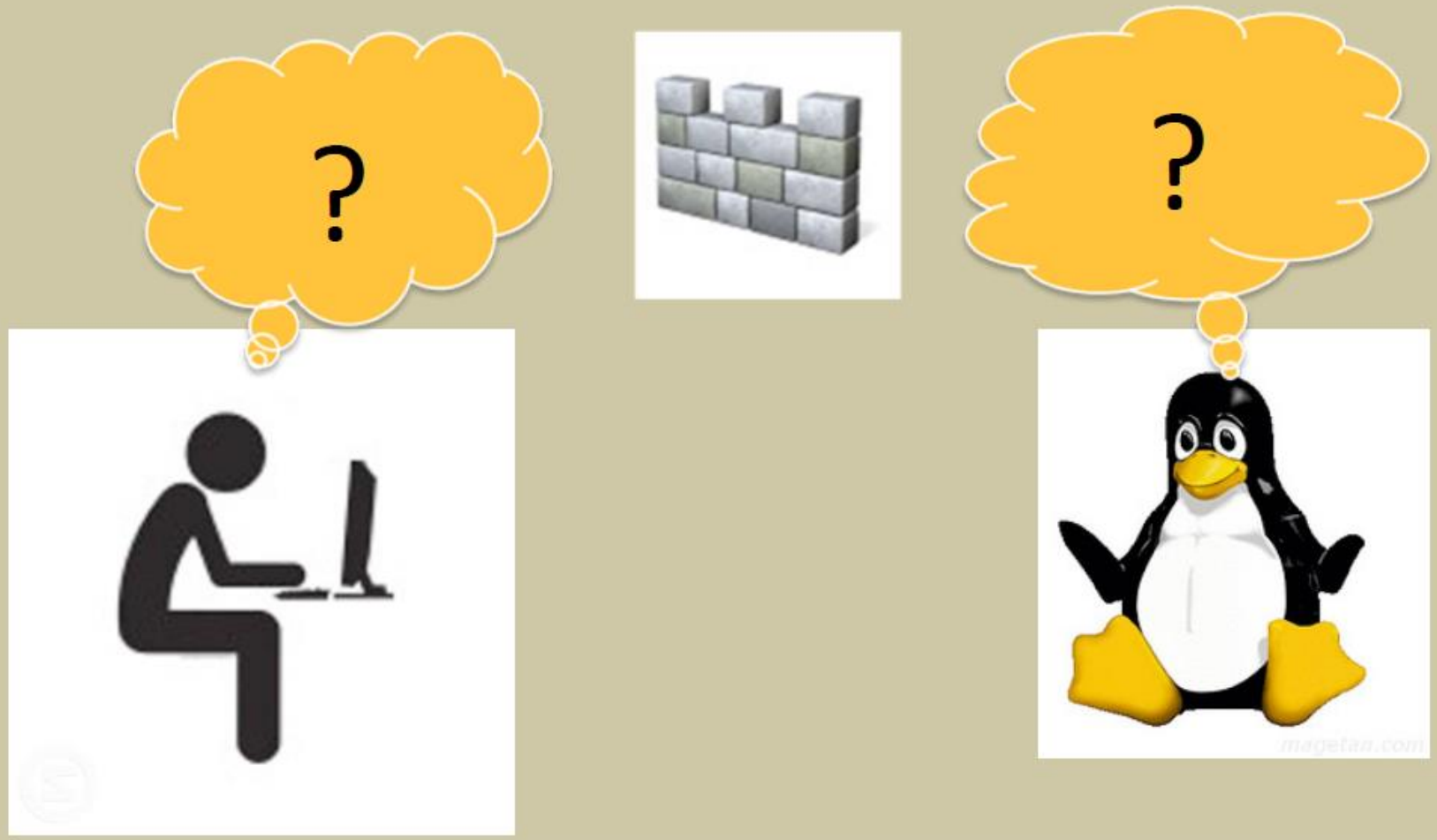
Unix



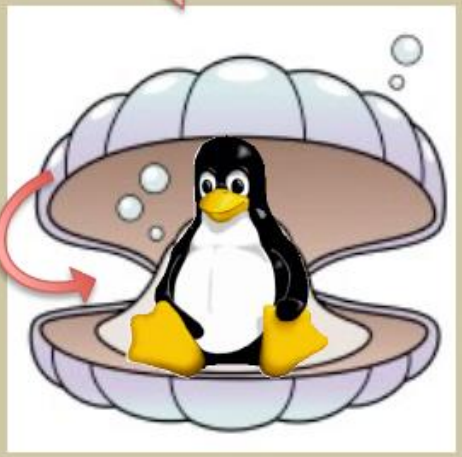
Computer Architecture



How can a user communicate with the Operating System?

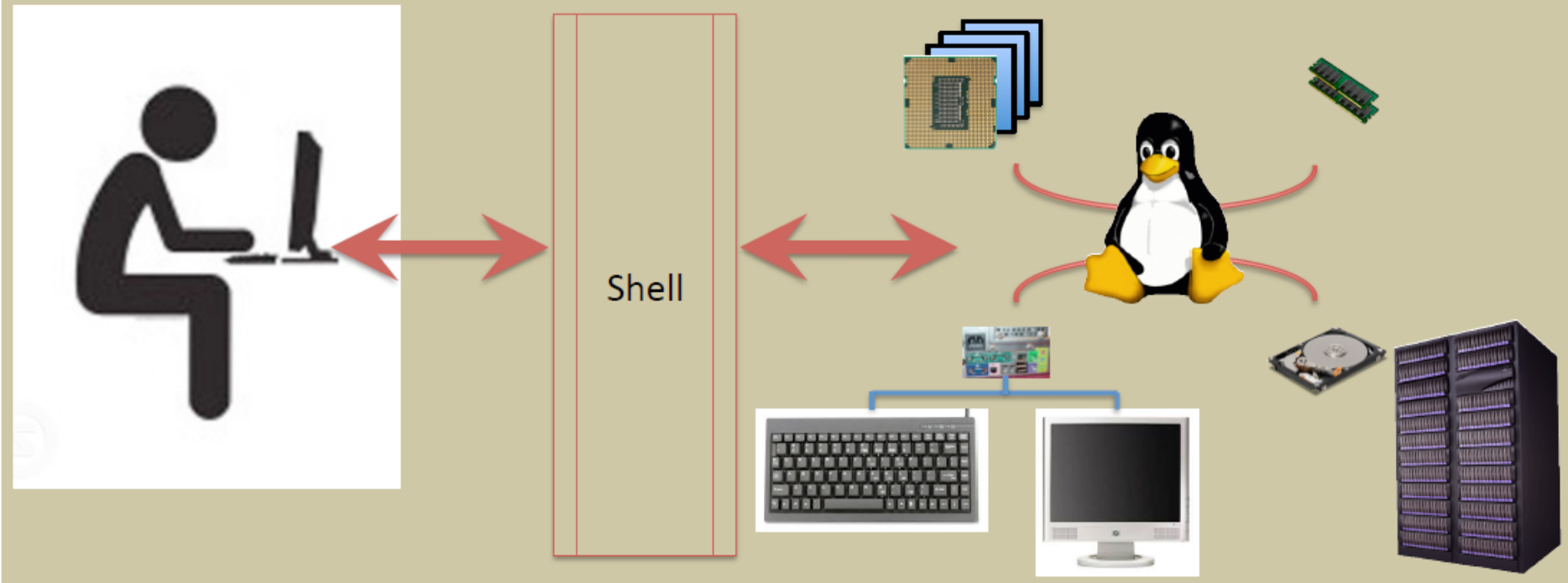


How can a user communicate with the Operating System?



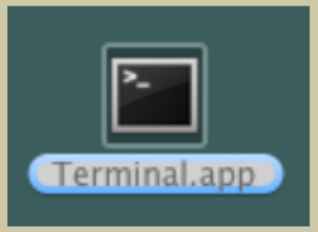
- User talks to the shell
- The shell interprets the commands and talks to the operating system

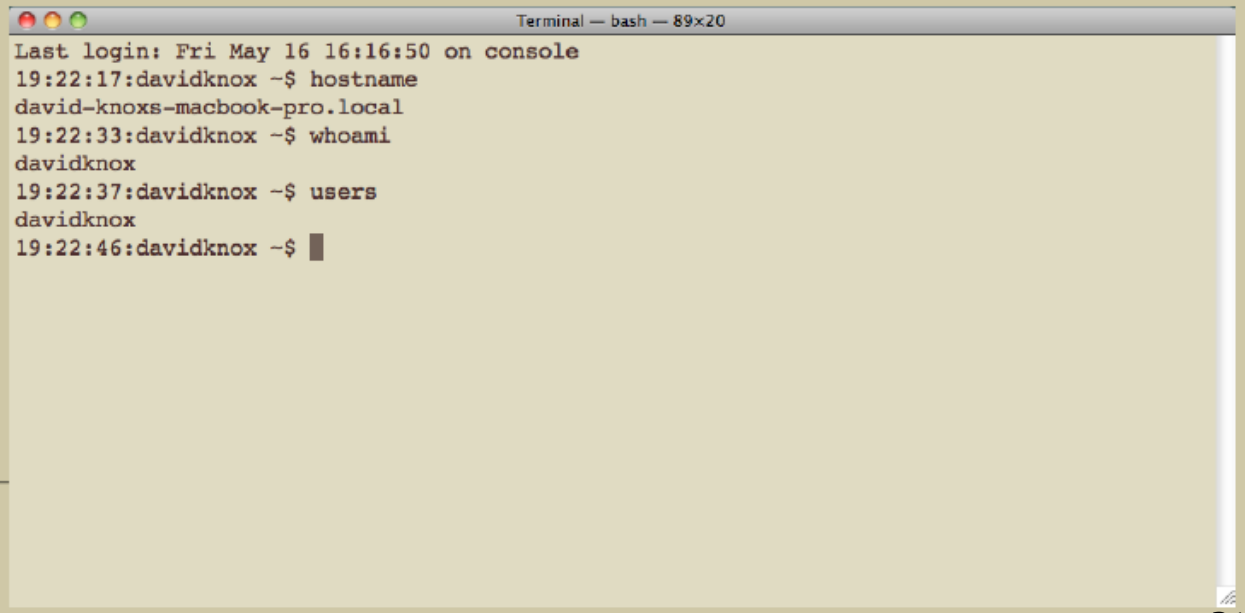
We can communicate with the Operating System by using the Shell



Using the Terminal Window

- When you select the terminal icon from your desktop, you create an interface with a Shell
- Every keystroke is sent to the shell, every character sent back is displayed in the window





Format of Commands

<cmd name> <options> <parameters>

- Most commands are programs
- Options tell the program how we want to process the data
- Parameters are the data the command will process

Getting help with Commands

- Most programs will understand the "--help" option and print information about the programs usage
- The shell can also lookup the usage by using the "man" command. To get help on connecting to a server,

  Type: **man login**

Terminal — less — 89×20

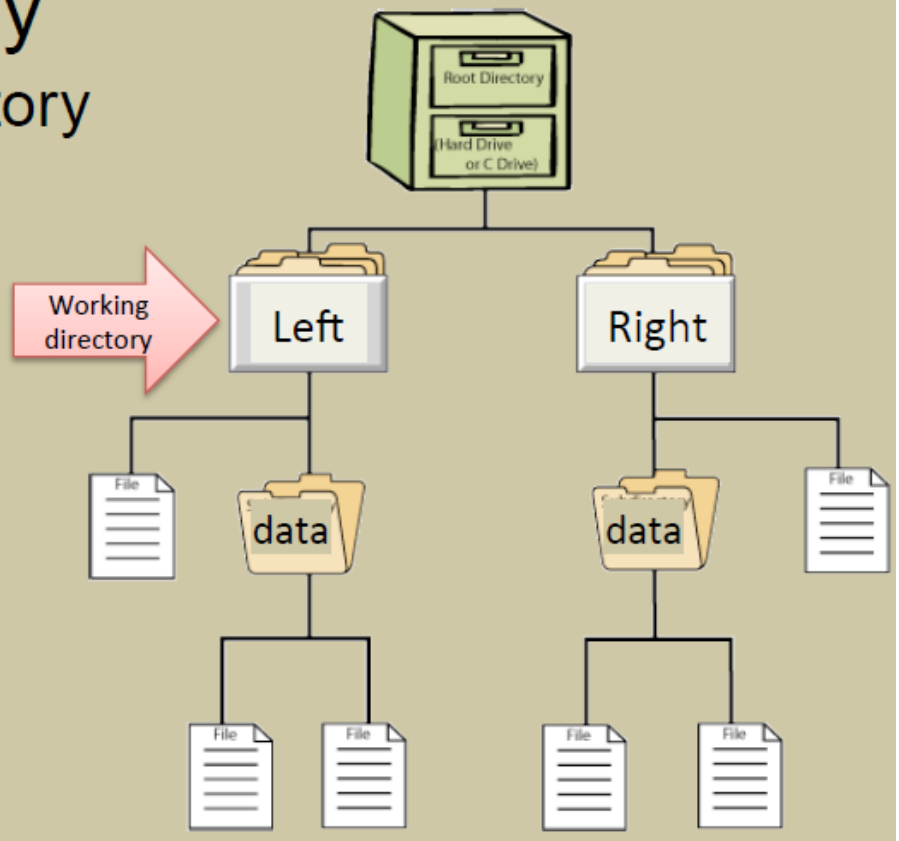
```
LOGIN(1)                    BSD General Commands Manual                    LOGIN(1)

NAME
     login -- log into the computer

SYNOPSIS
     login [-pq] [-h hostname] [user]
     login -f [-lpq] [-h hostname] [user [prog [args...]]]

DESCRIPTION
     The login utility logs users (and pseudo-users) into the computer system.

     If no user is specified, or if a user is specified and authentication of
     the user fails, login prompts for a user name.  Authentication of users is
     configurable via pam(8).  Password authentication is the default.

     The following options are available:

     -f      When a user name is specified, this option indicates that proper
:
```

"Everything is a file"

- In Unix, everything looks like a file:
  - documents stored on disk
  - directories
  - inter-process communication
  - network connections
  - devices (printers, graphics cards, interactive terminals, …)
  - programs
  - scripts
- They are accessed in a uniform way:
  - consistent API (e.g., read, write, open, close, …)
  - consistent naming scheme (e.g., /home/debray, /dev/cdrom)

Navigating the File System

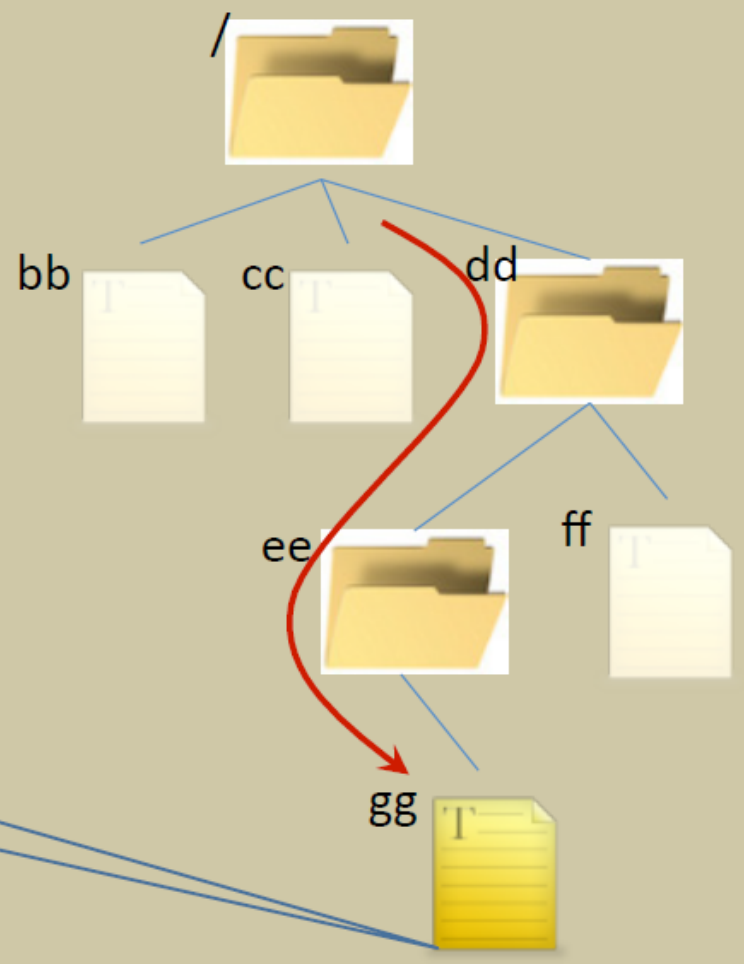
- Current working directory
  - pwd – print working directory
  - cd – change directory
  - Relative path vs full path
- File System commands:
  - ls – list directory contents
  - cp – copy files
  - rm – remove files
  - mv – move files
  - mkdir – make directory
  - rmdir – remove directory



cd /Right
cd data

Referring to files: Absolute Paths

- *Absolute path*
  - list the directories on the path from the root ("/")
  - separated by "/"



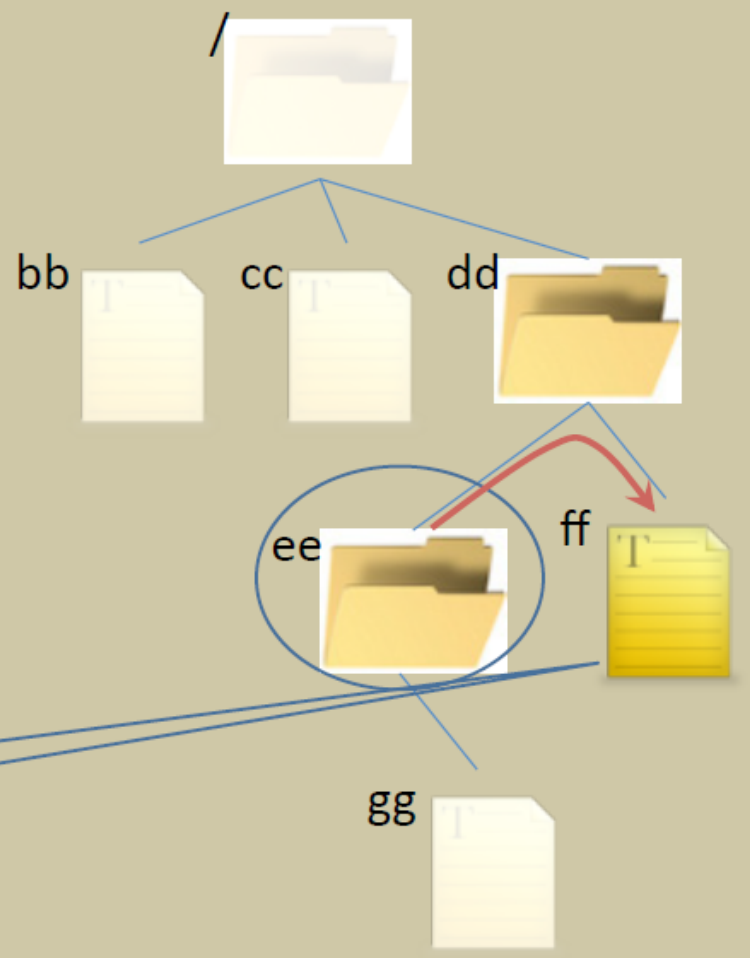
What is the absolute path of gg?

Referring to files: Relative Paths

- *Current directory*
- *Relative path*

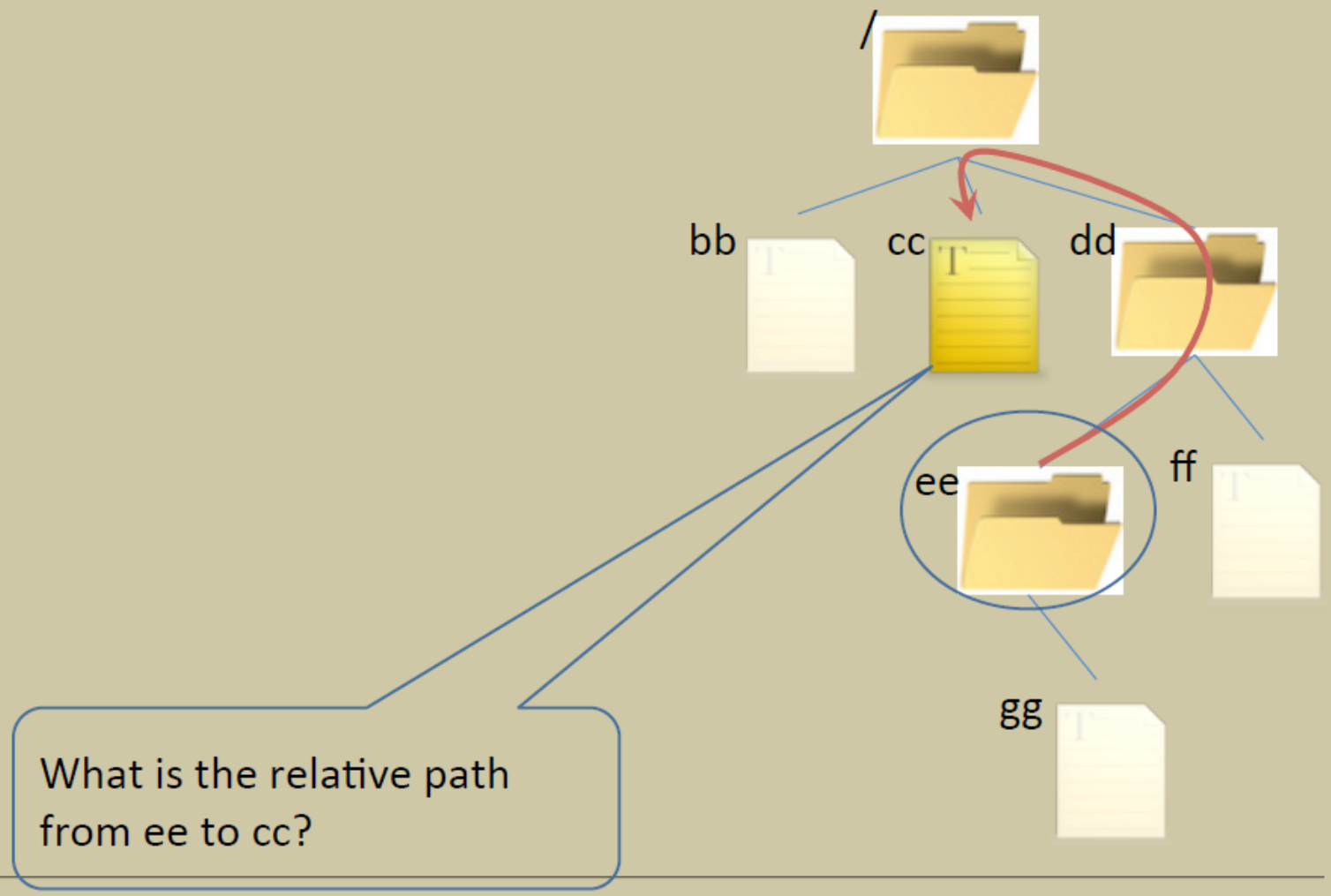
..

.



If you are in the ee directory, what is the relative path to ff?

Referring to files: Relative Paths



Commands for Processing Files

- cat – copy the contents to the screen
- more (less) – display file contents in user friendly manner
- head – copy the first lines of a file to the screen
- tail – copy the last lines of a file to the screen
- wc – count the number of lines, words, characters in a file
- grep - globally search a regular expression and print

Commands for checking the System Status

- who – list of current users
- whoami – what is my user name
- top – list the processes using the most resources
- ps – process status
- uptime – how long has the system been running
- date – current date and time

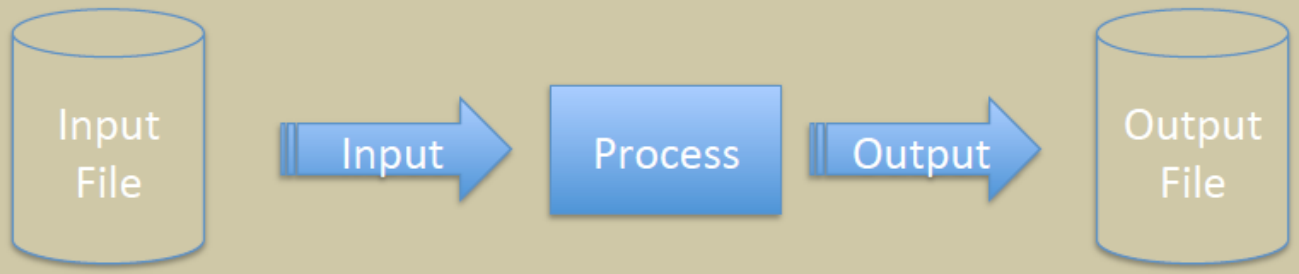
How can a user communicate with the Operating System?

- Shell maintains information
  - PATH : used for finding programs/scripts to be executed
  - HOME : full path to users home directory
  - …

Input and Output for a Process

- Most programs can accept input from the keyboard
- Most programs produce output
- Normally the output is given to the shell to be displayed on the screen

Redirecting Input and Output



- We can tell the shell to pass the characters from a file to a process as input
- We can ask the shell to place the output into a file instead of displaying it on the screen

Redirecting Output

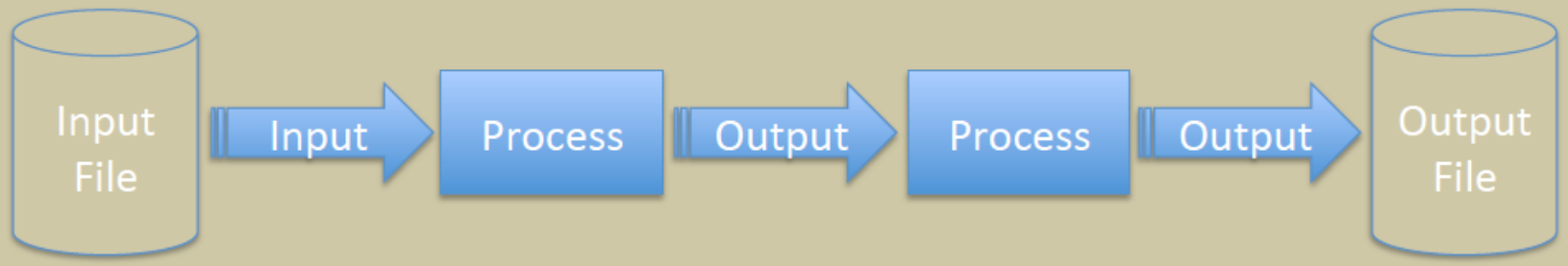
- Shell can redirect the output from any command into a file

history > my_history.out

- Name assigned for default input: stdin
- Name assigned to default output: stdout
- Some programs write output to one file and error information to another file: stderr

history >1 my_history.out >2my_history.err

Piping output of one process as input to another process



- Piping can connect multiple processes into a single command
- "|" is used to indicate the connection

```
ps -ax | wc
```

This command will print the process information to *stdout*. The shell will then take that output and use it as *stdin* to the *wc* command. The output from *wc* is not redirected or piped to another process, therefore it will appear on the screen.

Useful Commands in Piping

- wc - word, line, character, and byte count
- grep - output lines matching a pattern
- sort - sort lines of text files
- uniq - filter out repeated lines in a file
- cut - cut out selected portions of each line of a file
- tee – sends the data to both to a file and *stdout*

File completion

From command line,

- Type part of name of a command or filename, press <tab> and will auto-complete the name for you.

- Arrow keys to go back through history of commands you typed in.

history

Display a history of recently used commands

- `history`
- `history 10`
- `history -r 10`
- `!!`
- `!n`
  - repeat command **n** in the history where n is a #
- `!-1`
- `!-2`
- `!ca`

Some other useful commands

- **wc**

```
wc file
wc –l file
wc –w file
```

- **grep** *pattern* [*file*]

```
grep public *java
grep include controller.cpp
grep TODO src/*
ls | grep –i main
```

diff

Compare two files

- `diff file1 file2`
- `sdiff file1 file2`

sort

- sort data.txt
- sort –n data.txt *(on numerical data file)*
  Why is the –n parameter necessary?
- Sort by column
  sort –k2 data.txt
  sort –k2,3 data.txt
- sort –u data.txt

find

- Find a file in a directory tree.
  (period means to start in the current directory)
- `find . -name filename -print`

End of Lecture One