CSE 15L:
Software Tools and Techniques Laboratory

Winter 2021 - http://ieng6.ucsd.edu/~cs15x

Instructors: Gary Gillespie Keith Muller

Class sessions will be recorded and made available to students asynchronously.

Schedule

Last Lecture

1. Introduction to vi

Today

1. Introduction to Unix

First/Last Resort Tools

- Lowest overhead requirement, works with connection issues

- Connecting to remote machine over a network
- man ssh
- ssh your_username@host_ip_address
- ssh cs15lxxx@ieng6.ucsd.edu

- to copy files between machines over a network
- man sftp
- sftp cs15lxxx@ieng6.ucsd.edu
  - put, get
  - cd, lcd,
  - ls,lls

Review

1. What is the command for editing the file foobar.java in the vi editor?

2. The vi editor has two common modes: command and insert mode. How do you switch between the two?

3. List two ways of exiting the vi editor.

Review

1. What is the command for editing the file foobar.java in the vi editor?

```
vi foobar.java
```

2. The vi editor has two common modes: command and insert mode. How do you switch between the two?

3. List two ways of exiting the vi editor.

Review

1. What is the command for editing the file foobar.java in the vi editor?

```
vi foobar.java
```

2. The vi editor has two common modes: command and insert mode. How do you switch between the two?

   command -> insert: **`i`** or **`a`** to insert around the cursor, **`I`** or **`A`** to enter insert mode at beginning or end, **`o`** or **`O`** to enter insert mode after/before a line.

   insert -> command: **`<esc>`**

3. List two ways of exiting the vi editor.

Review

1. What is the command for editing the file foobar.java in the vi editor?

   `vi foobar.java`

2. The vi editor has two common modes: command and insert mode. How do you switch between the two?

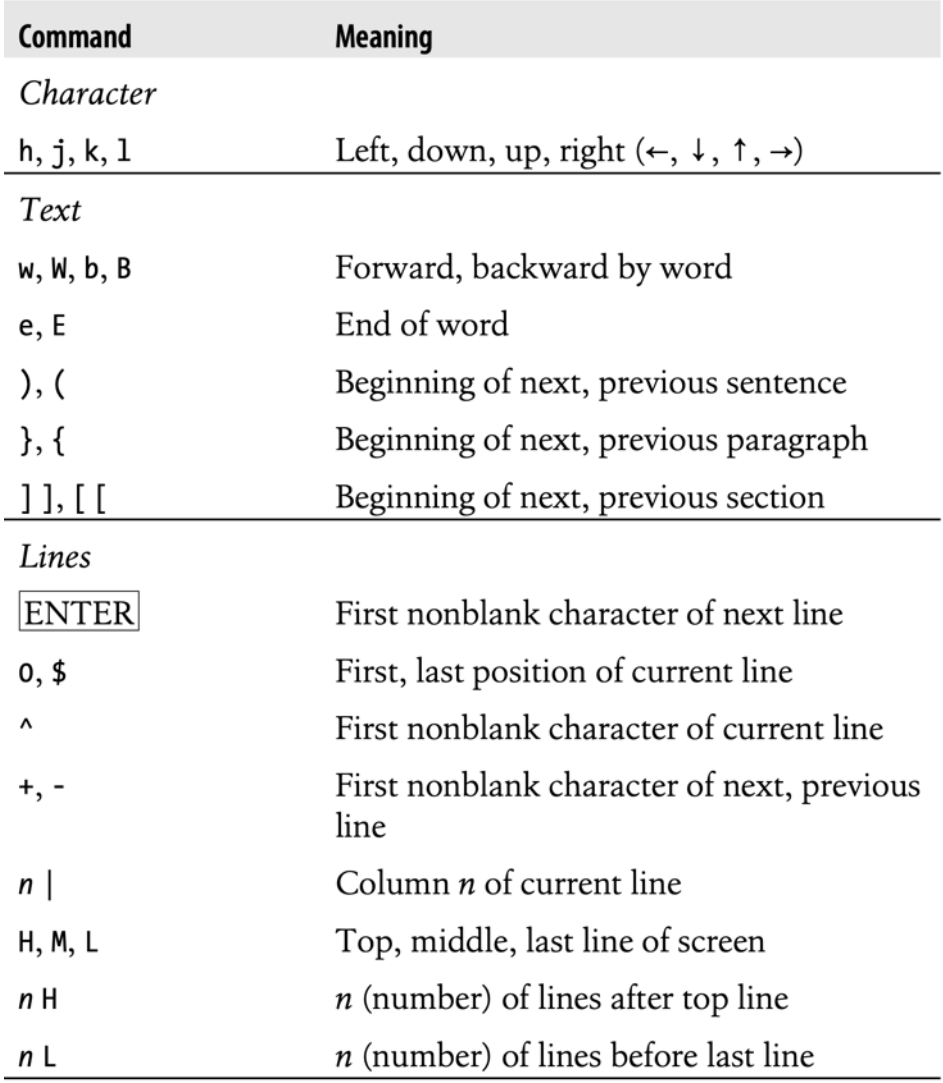
   command -> insert: **`i`** or **`a`** to insert around the cursor, **`I`** or **`A`** to enter insert mode at beginning or end, **`o`** or **`O`** to enter insert mode after/before a line.

   insert -> command: **`<esc>`**

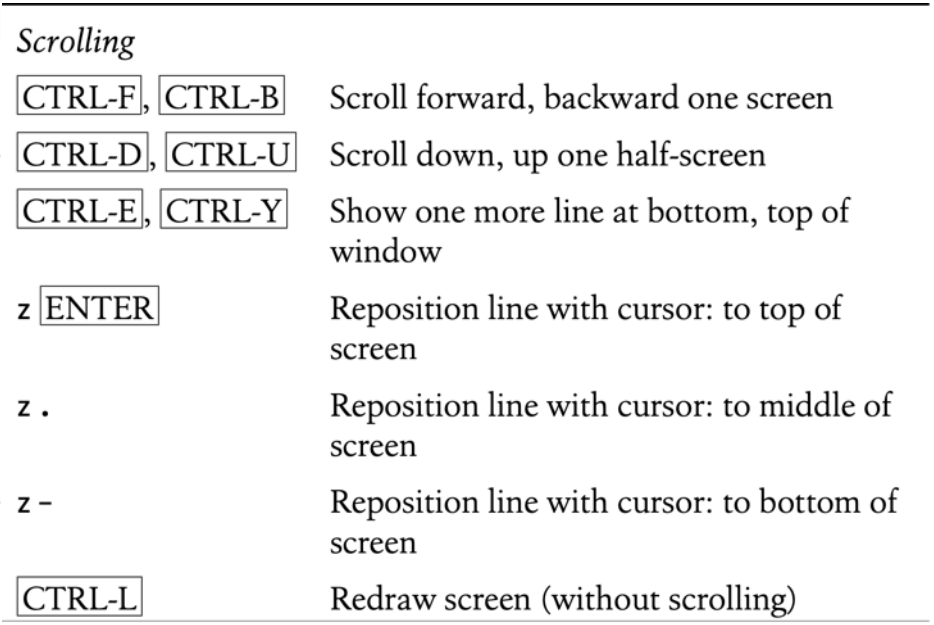
3. List two ways of exiting the vi editor.

   `:q` or **`:q!`** to exit without changes, **`ZZ`** or **`:x`** to save (if changes) and exit, or **:wq** to save and quit

Vi Movement

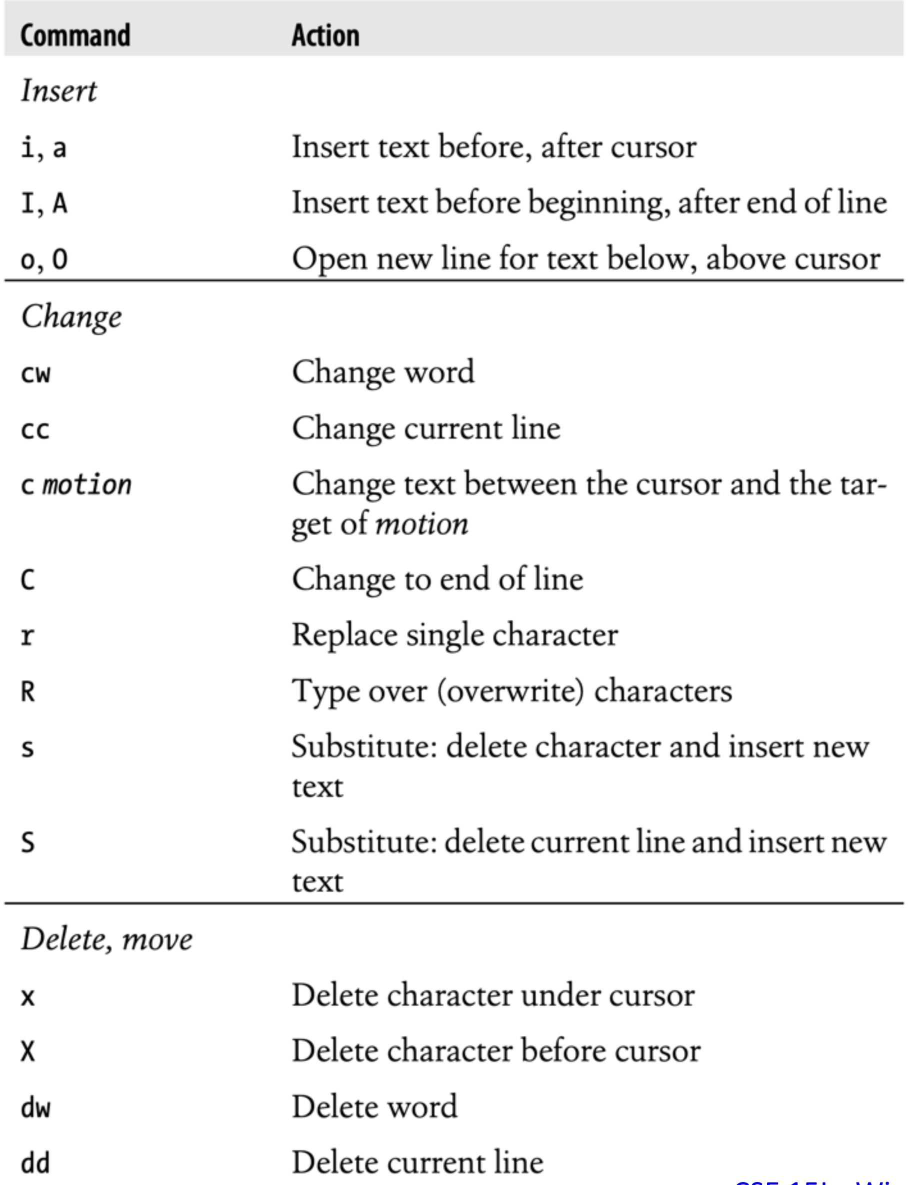


Command	Meaning
Character	
h, j, k, l	Left, down, up, right (←, ↓, ↑, →)
Text	
w, W, b, B	Forward, backward by word
e, E	End of word
), (	Beginning of next, previous sentence
}, {	Beginning of next, previous paragraph
]], [[	Beginning of next, previous section
Lines	
ENTER	First nonblank character of next line
0, $	First, last position of current line
^	First nonblank character of current line
+, -	First nonblank character of next, previous line
n |	Column *n* of current line
H, M, L	Top, middle, last line of screen
n H	*n* (number) of lines after top line
n L	*n* (number) of lines before last line

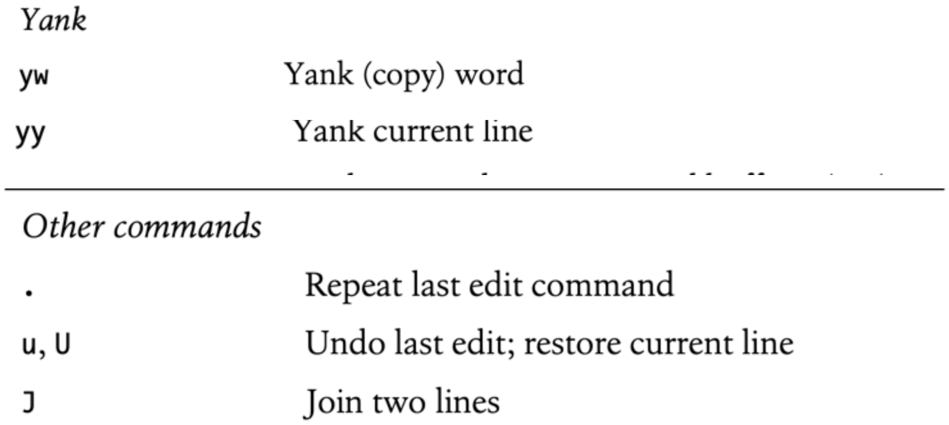


Scrolling	
CTRL-F, CTRL-B	Scroll forward, backward one screen
CTRL-D, CTRL-U	Scroll down, up one half-screen
CTRL-E, CTRL-Y	Show one more line at bottom, top of window
z ENTER	Reposition line with cursor: to top of screen
z .	Reposition line with cursor: to middle of screen
z -	Reposition line with cursor: to bottom of screen
CTRL-L	Redraw screen (without scrolling)

VI Editing

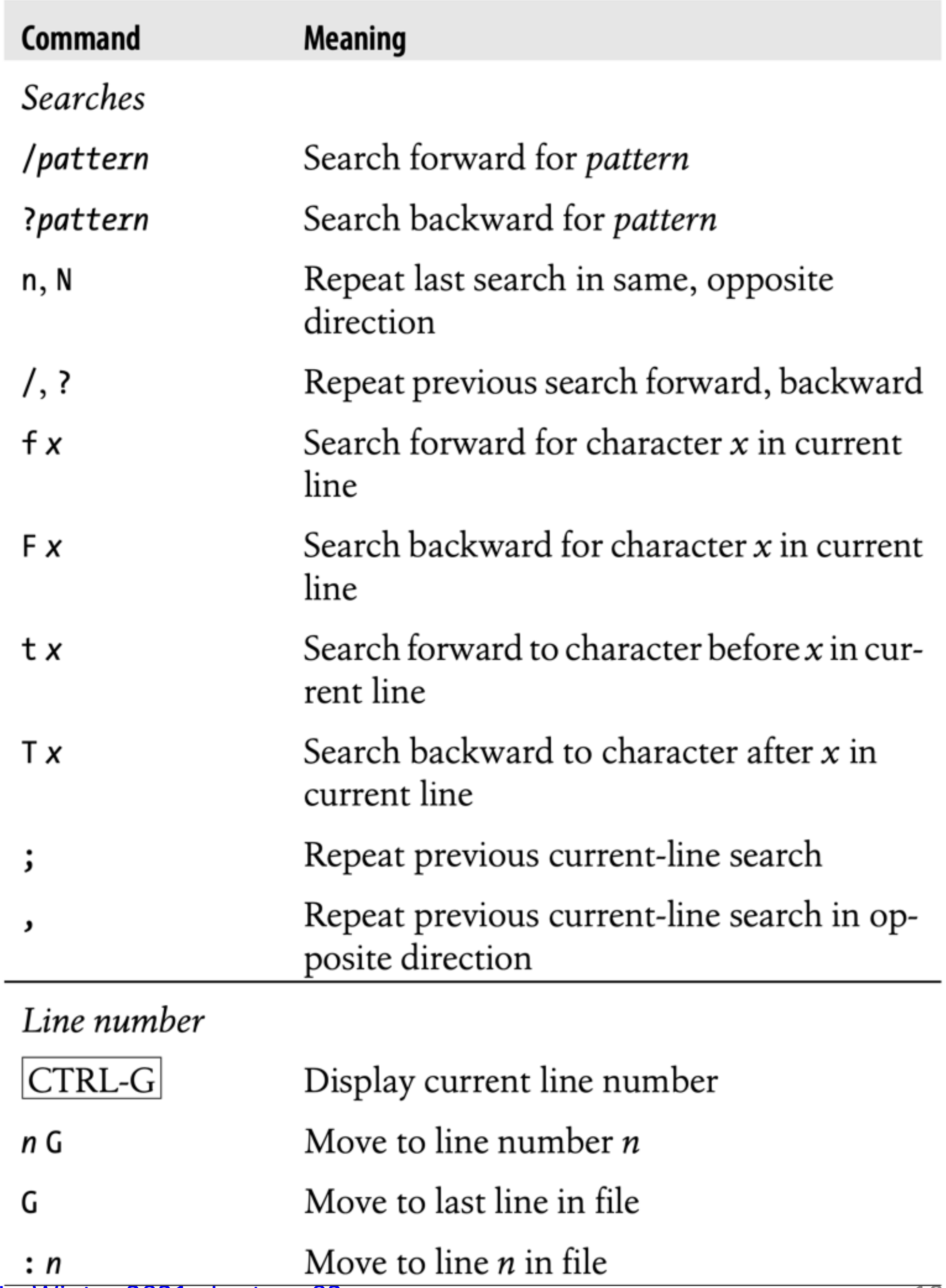


Command	Action
Insert	
`i, a`	Insert text before, after cursor
`I, A`	Insert text before beginning, after end of line
`o, O`	Open new line for text below, above cursor
Change	
`cw`	Change word
`cc`	Change current line
`c` *`motion`*	Change text between the cursor and the target of *motion*
`C`	Change to end of line
`r`	Replace single character
`R`	Type over (overwrite) characters
`s`	Substitute: delete character and insert new text
`S`	Substitute: delete current line and insert new text
Delete, move	
`x`	Delete character under cursor
`X`	Delete character before cursor
`dw`	Delete word
`dd`	Delete current line



Yank	
`yw`	Yank (copy) word
`yy`	Yank current line
Other commands	
`.`	Repeat last edit command
`u, U`	Undo last edit; restore current line
`J`	Join two lines

Vi Searching



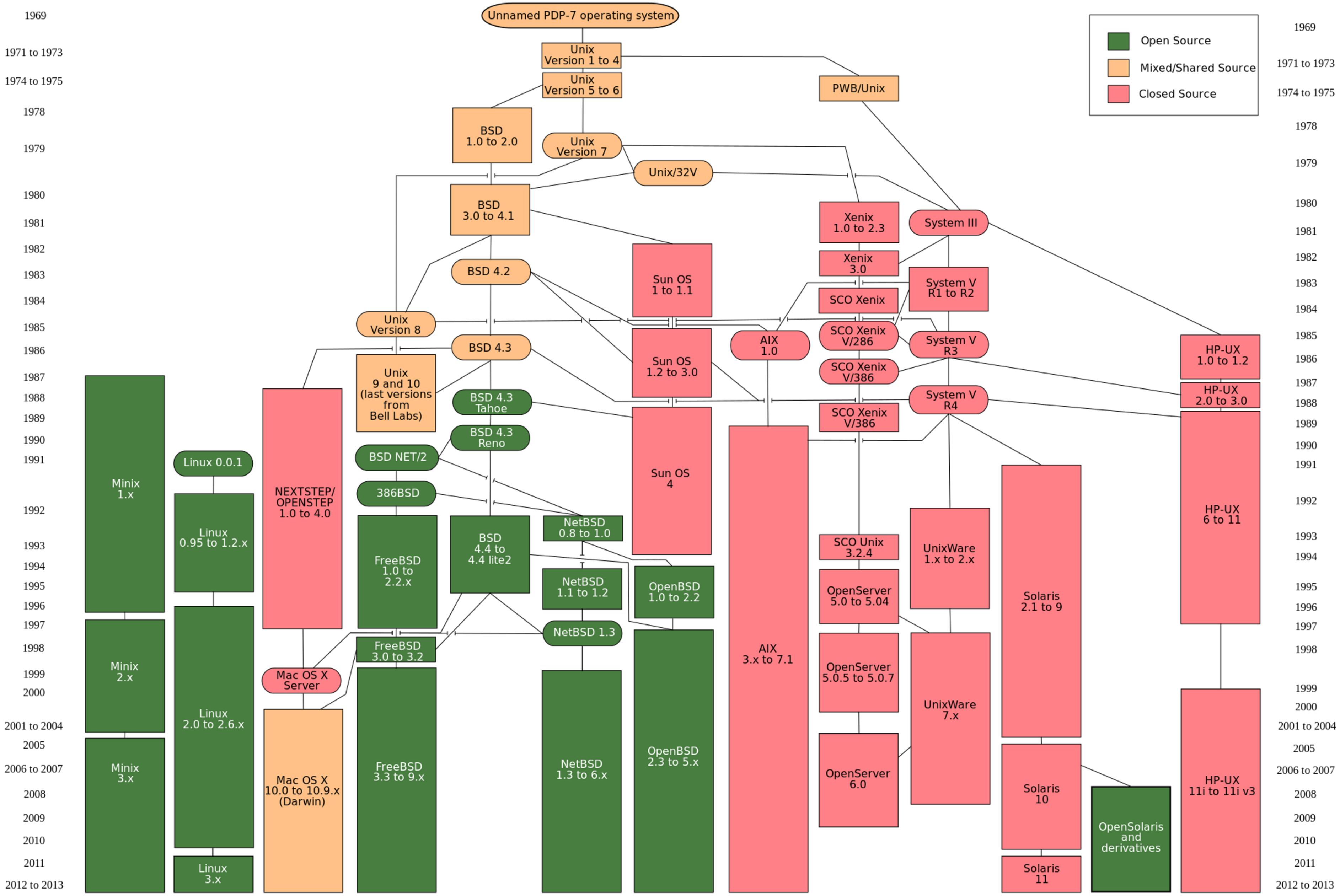
Command	Meaning
Searches	
/*pattern*	Search forward for *pattern*
?*pattern*	Search backward for *pattern*
n, N	Repeat last search in same, opposite direction
/, ?	Repeat previous search forward, backward
f *x*	Search forward for character *x* in current line
F *x*	Search backward for character *x* in current line
t *x*	Search forward to character before *x* in current line
T *x*	Search backward to character after *x* in current line
;	Repeat previous current-line search
,	Repeat previous current-line search in opposite direction
Line number	
CTRL-G	Display current line number
n G	Move to line number *n*
G	Move to last line in file
: *n*	Move to line *n* in file

Introduction to Unix

Relevance:
Why Do We Need to Know Unix?

- Operating system of all CSE classes as a CS major at UCSD
- Well adopted in industry
- Powerful development environment
- Expected competence for most industry jobs
- Back end of many data and compute systems including industry giants like Facebook and Google

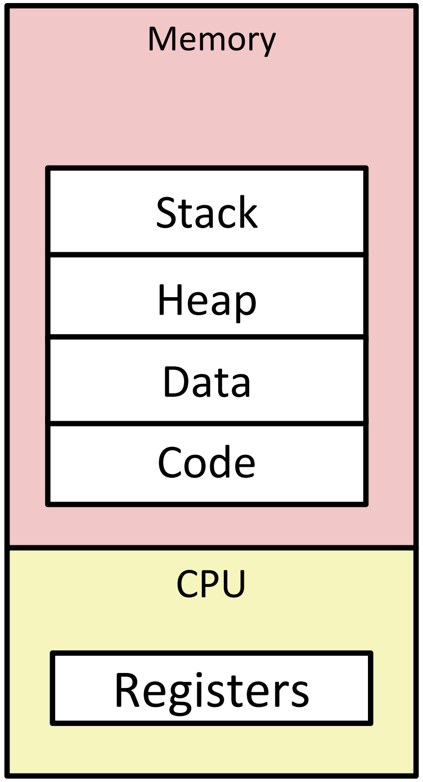
Everything is a file or a process!



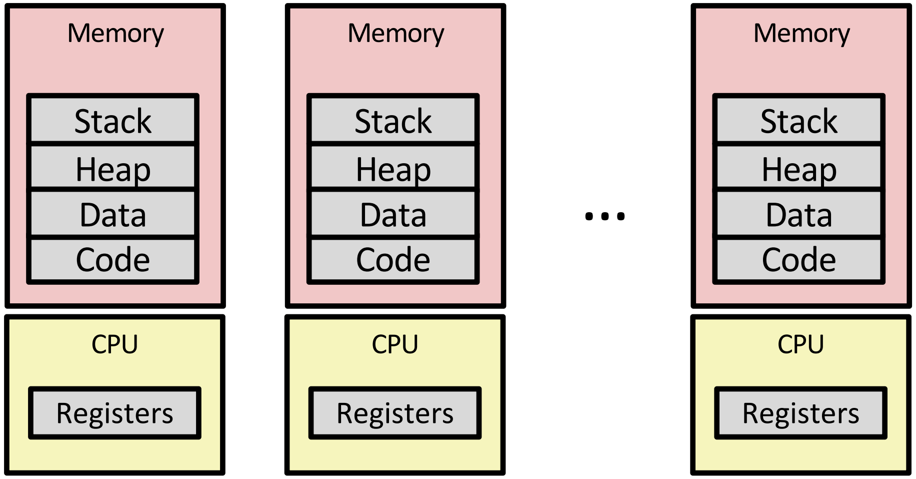
Sobell, “A Practical Guide to Linux Commands, Editors, and Shell Programming,” pg. 2-6

What is a Process?

- A ***process*** is an instance of a running program
  - One of the most profound ideas in computer science
  - Not the same as “program” or “processor”
- Process provides each program with two key abstractions:
  - *Logical control flow*
    - Each program seems to have exclusive use of the CPU (current process)
  - *Private address space*
    - Each program appears to have exclusive use of main memory

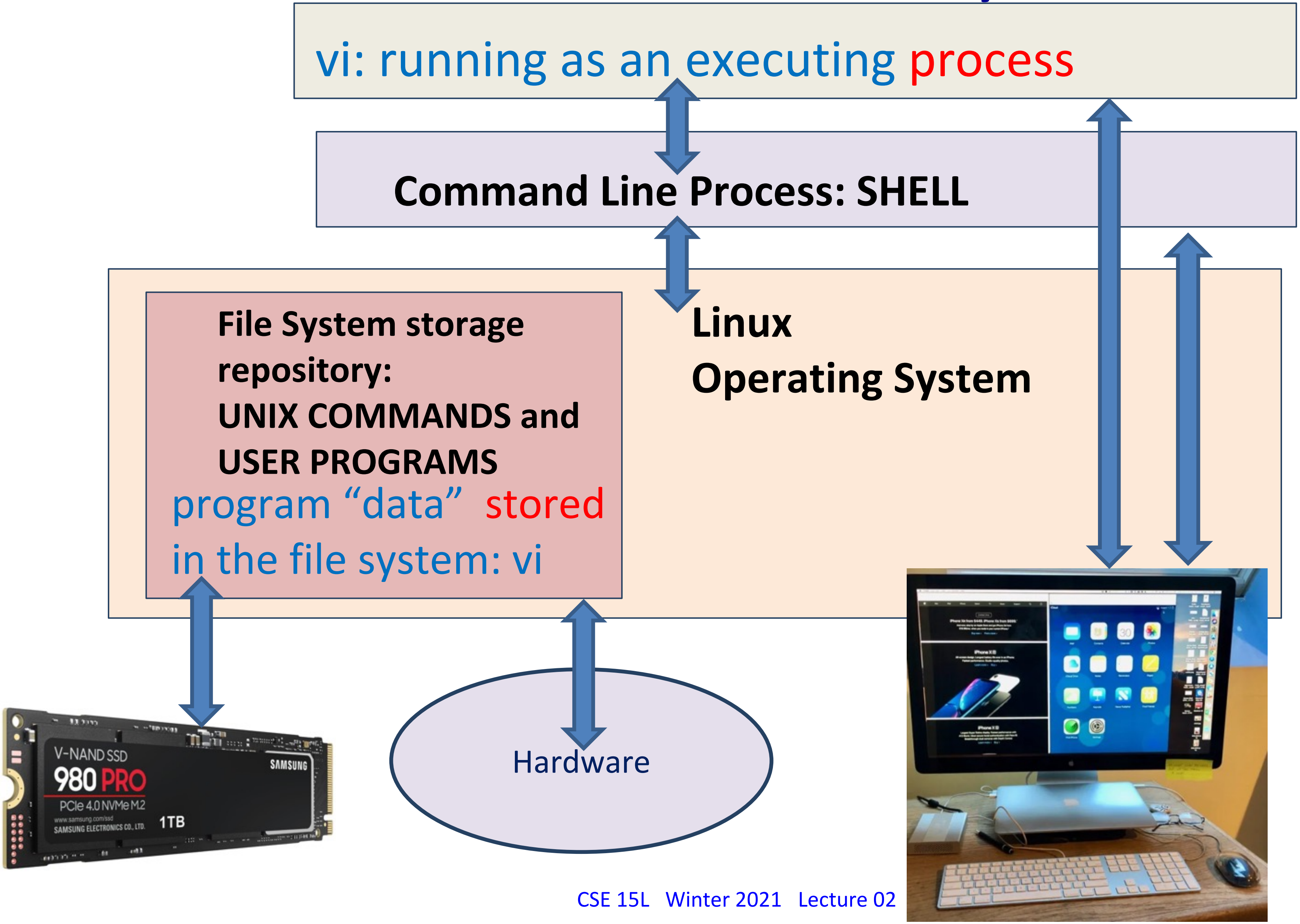


Multiprocessing: The Illusion (Single CPU case)



- Computer has multiple processes ready to run at any time
  - All runnable processes are allocated memory
- A CPU runs just one program at a time (current process)
- The OS switches instruction execution among the processes rapidly to create the illusion of concurrent execution

Linux Abstraction Layers



vi: running as an executing process

Command Line Process: SHELL

Linux

Operating System

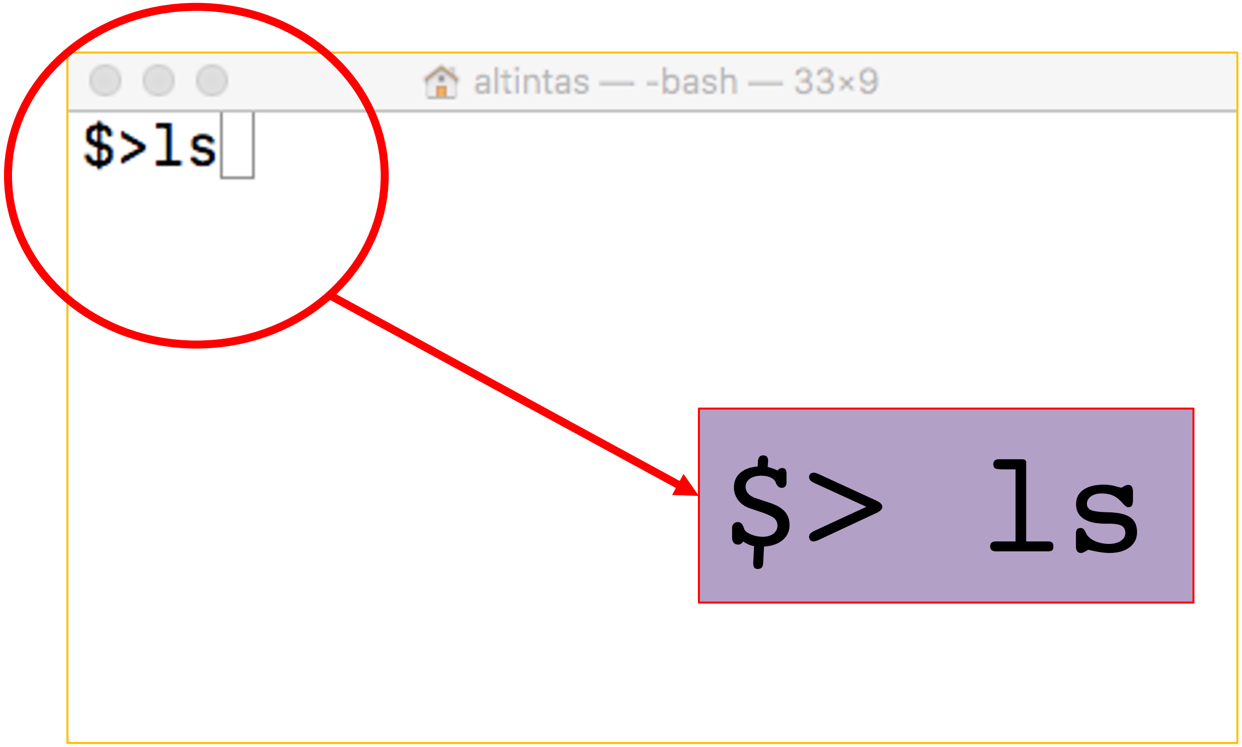
File System storage repository:

UNIX COMMANDS and USER PROGRAMS

program “data” stored in the file system: vi

Hardware

The UNIX Shell



```
$>ls
```

```
$> ls
```

What is a shell?

- Interface between the user and the kernel
- A command line interpreter
- Starts automatically when you login
  - Allows programming (shell scripting) within the shell environment
  - Accepts commands and often makes *system calls* to carry them out

In Unix…

- Commands are programs
- Processes are executing programs
  - They have unique PIDs (process identifiers)
- Files are collections of data
  - Text, binary, etc…
  - Organized in directory structure

Sobell, "A Practical Guide to Linux Commands, Editors, and Shell Programming," pg. 117

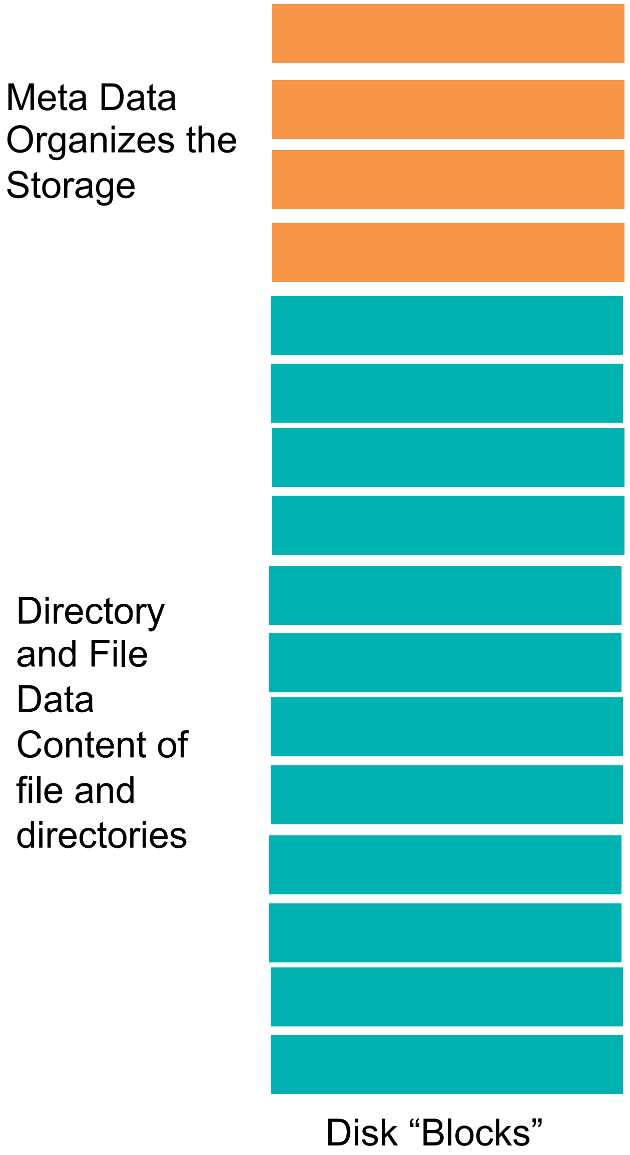
Common Shells

- Bourne Shell (**`/bin/sh`**)
- Bash (**`/bin/bash`**) “Bourne-Again Shell”
- C Shell (**`/bin/csh`**)
- Turbo C Shell (**`/bin/tcsh`**)
- Korn Shell (**`/bin/ksh`**)
- Z Shell (**`/bin/zsh`**)

Note: Typing **`ps`** in a terminal will show which shell you are using. Try it out!

Sobell, “A Practical Guide to Linux Commands, Editors, and Shell Programming,” pg. 28-29

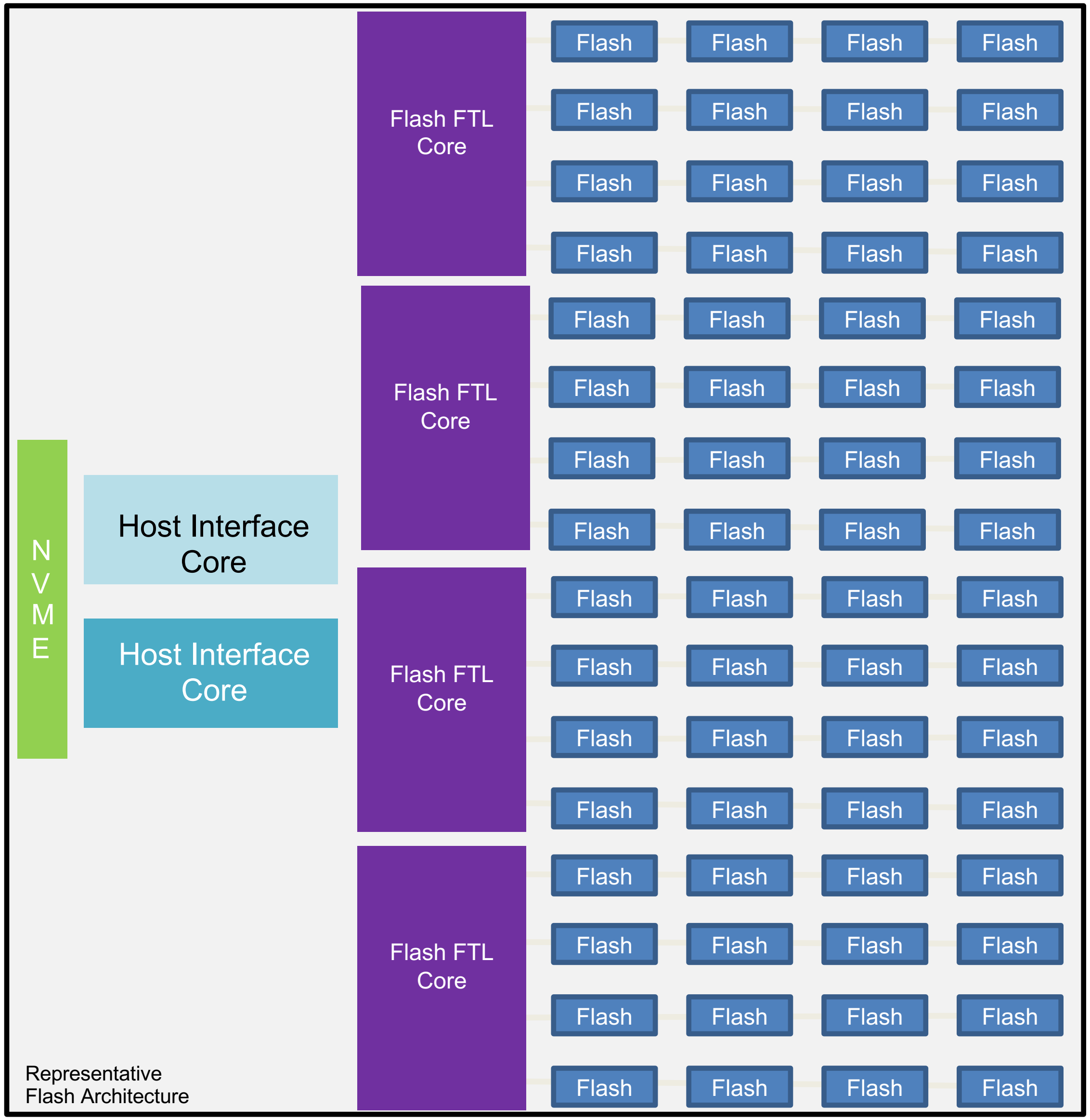
SSD and File Systems
Linux OS Subsystem



Meta Data Organizes the Storage

Directory and File Data Content of file and directories

Disk “Blocks”



Flash FTL Core

Flash FTL Core

Flash FTL Core

Flash FTL Core

Flash Flash Flash Flash
Flash Flash Flash Flash
Flash Flash Flash Flash
Flash Flash Flash Flash
Flash Flash Flash Flash
Flash Flash Flash Flash
Flash Flash Flash Flash
Flash Flash Flash Flash
Flash Flash Flash Flash
Flash Flash Flash Flash
Flash Flash Flash Flash
Flash Flash Flash Flash
Flash Flash Flash Flash
Flash Flash Flash Flash
Flash Flash Flash Flash
Flash Flash Flash Flash

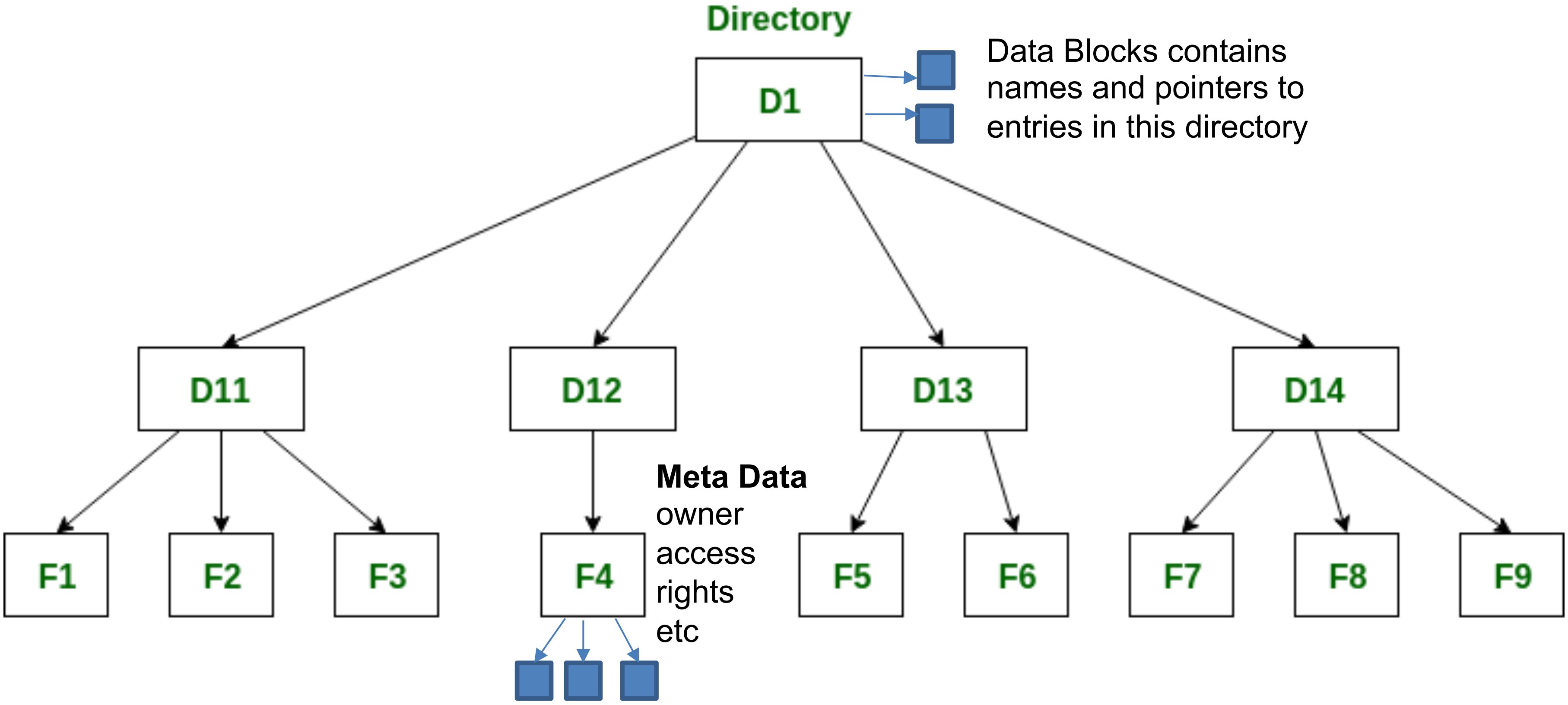
NVME

Host Interface Core

Host Interface Core

Representative
Flash Architecture

Storage Organization File “Tree”



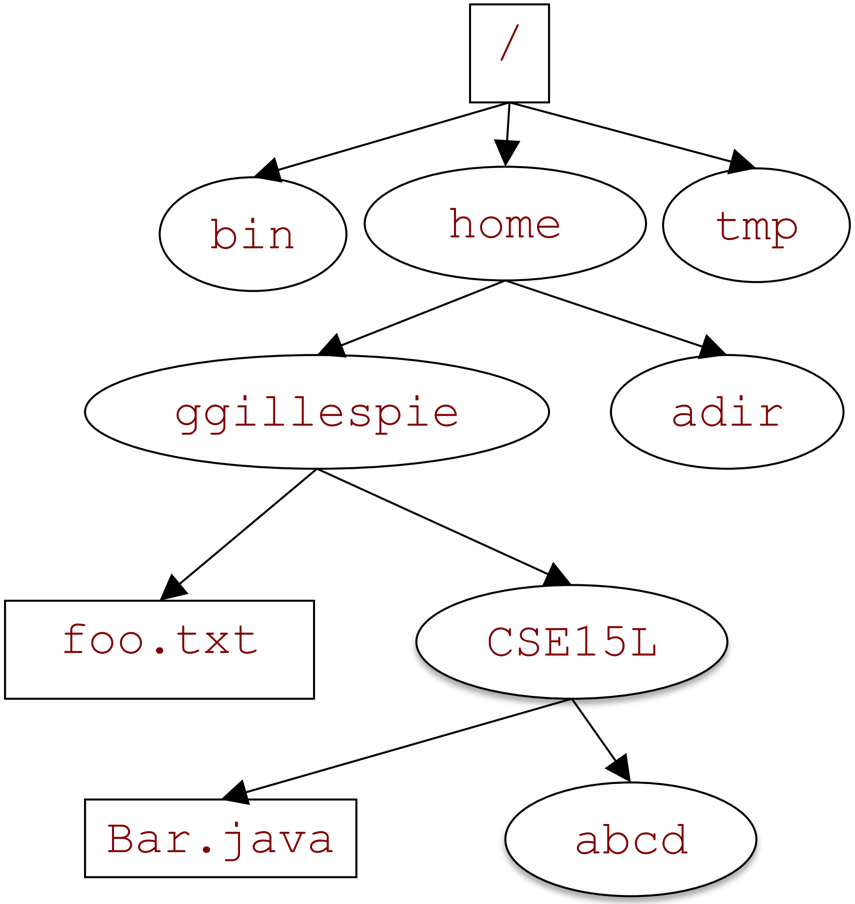
Data Blocks contains names and pointers to entries in this directory

Meta Data
owner
access
rights
etc

Data Blocks store contents of a file

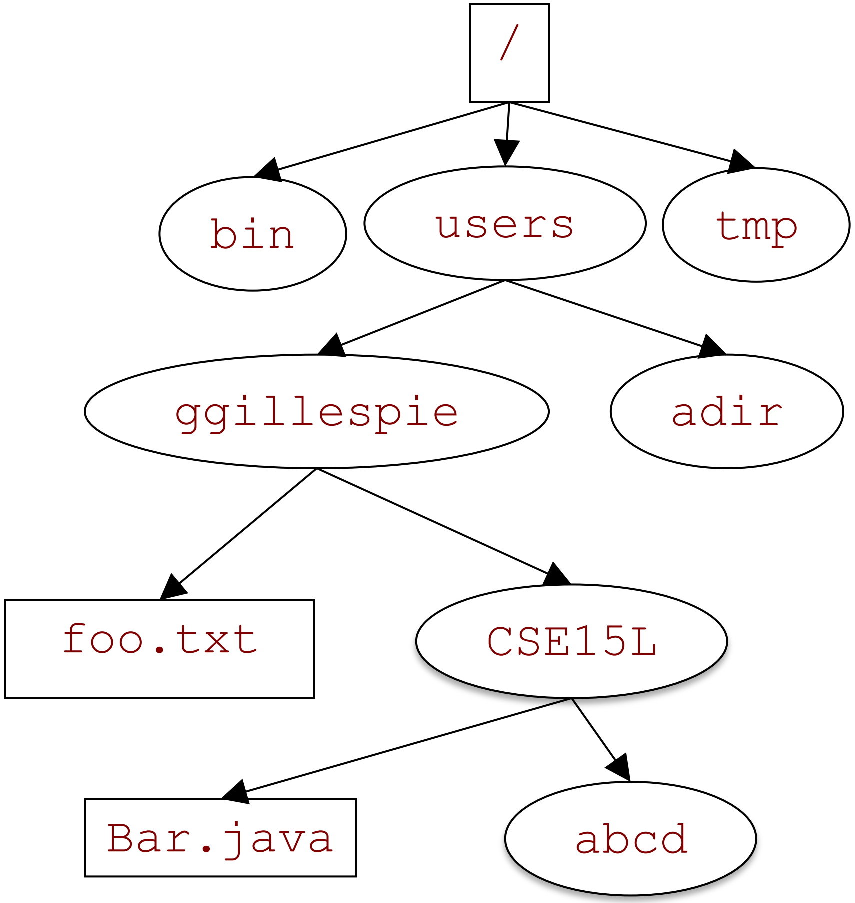
UNIX file hierarchy

- Organizes disk storage
- Maps name to file system objects (directory or a file)
- Directories may contain files or other directories
- Leads to a tree structure for the filesystem
- Root directory: /



Sobell, “A Practical Guide to Linux Commands, Editors, and Shell Programming,” pg. 78-85

Path names



- Separate directories by **/**
- Absolute path names
  - start at root and follow the tree

  /users/ggillespie/foo.txt

- Relative path
  - start at working directory
  - **..** refers to level above; **.** refers to working dir.
  - If **/users/ggillespie/CSE15L** is working dir, the path to the file can be written as:

    ../foo.txt

- ~ (TILDE) in path names
  - **~/foo.txt** or **~ggillespie/foo.txt**

Caution when working with relative paths:

Make sure you check which directory you are in.

Which of the following is an absolute path?

a) `/Users/ggillespie/temp/in.txt`

b) `temp/in.txt`

c) `../in.txt`

d) None of the above

e) More than one of the above

pwd

Caution when working with relative paths:

Make sure you check which directory you are in.

Easy to remember!

pwd stands for print working directory

Common Unix Commands

- **`pwd`** (print working directory)
- **`ls`** (list files)
- **`mkdir`** (make directory)
- **`cd`** (change directory)
- **`cp`** (copy)
- **`mv`** (move or rename)
- **`rm`** (remove)
- **`cat`** (catenate or read and output a file)
- **`man`** (display manual for Unix command)

Command Line Syntax

- Some commands can be run as is (**`ls`**, **`pwd`**) but others require arguments
- Arguments are given to the shell as space separated strings (1 or more spaces!)
- Entire line is processed when you press RETURN

```
$ command arg1 arg2 arg3
```

Examples:

```
mkdir hw1
cp TemplateGUI.java NewGUI.java
rm bin/*.class
```

Note: **`*`** is the wildcard character and represents 0 or more characters

stdin, stdout, and stderr

- Each shell (and in fact all programs) automatically have open three io streams when they start up
  - Standard input (stdin): Usually from the keyboard
  - Standard output (stdout): Usually to the terminal
  - Standard error (stderr): Usually to the terminal

- Programs use these three iostreams when reading (e.g. `cin` in C++), writing (e.g. `cout` in C++), or reporting errors/diagnostics (e.g. `cerr` in C++)

- Java: `System.in`, `System.out`, `System.err`

Sobell, "A Practical Guide to Linux Commands, Editors, and Shell Programming," pg. 131

IO Redirection To a File

>	creates or overwrites file if it exists
>>	creates or appends to file if it exists
<	input from file

stdout:

```
$ ls > MyFiles.text
$ cat > temp1.txt
$ cat >> temp1.txt
```

stdin:

```
$ mail user@domain.com < message
```

IO Redirection to another Process: Pipe |

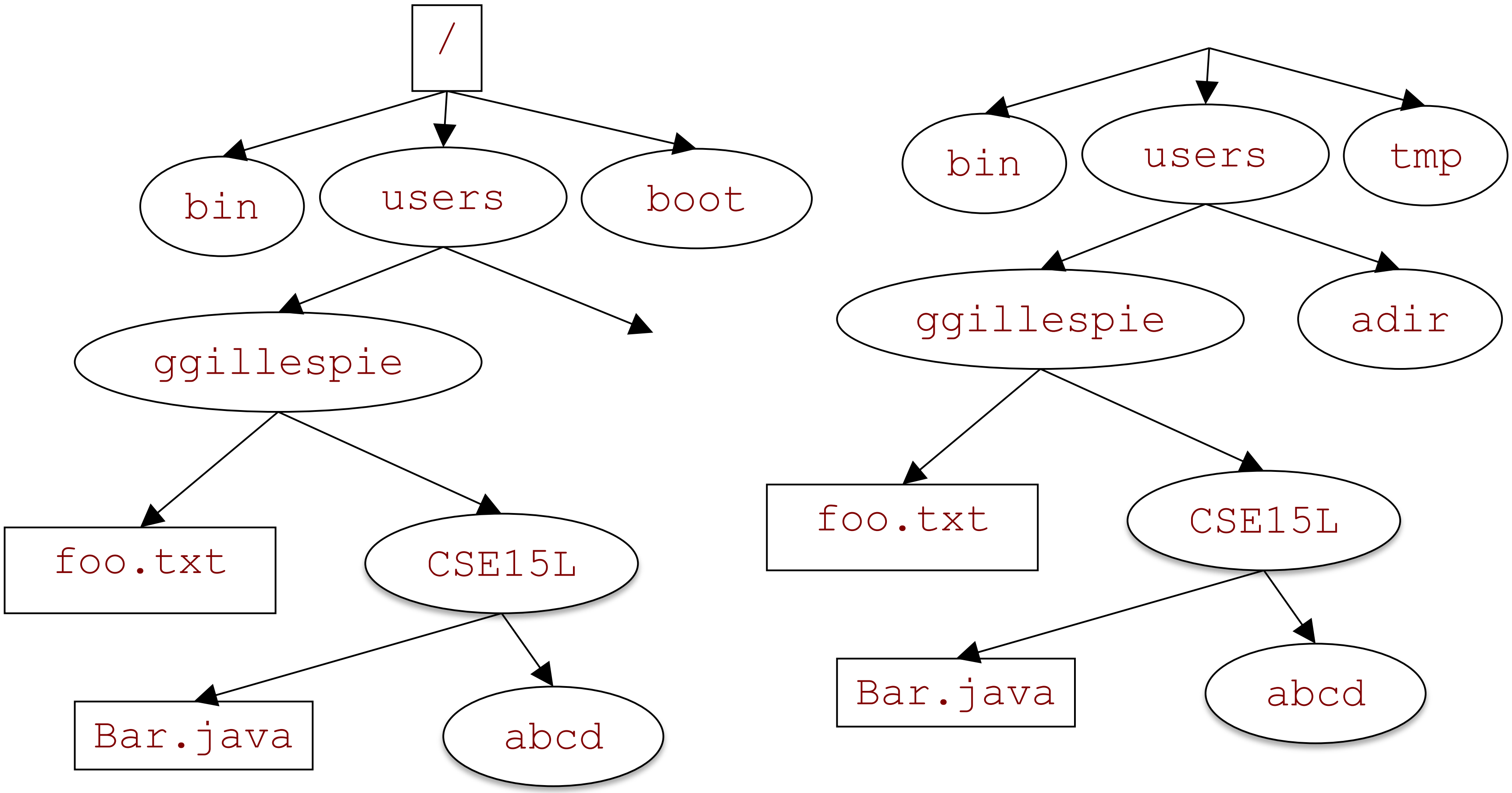
- ps augx | grep cs15l
  - Creates two currently executing processes
  - where the output of the ps is sent to the input of grep
- $ ps augx | grep cs15l

```
root      11229  0.0  0.0 176248  5488 ?        Ss   12:46   0:00 sshd: cs15lwi21
cs15lwi+ 11313  0.0  0.0 176248  2308 ?        S    12:47   0:00 sshd: cs15lwi21@pts/16
cs15lwi+ 11314  0.0  0.0 127632  3504 pts/16   Ss   12:47   0:00 -bash
root      11846  0.0  0.0 180612  5864 ?        Ss   12:48   0:00 sshd: cs15lwi21atb [priv]
cs15lwi+ 11964  0.0  0.0 180612  2360 ?        S    12:48   0:00 sshd: cs15lwi21atb@pts/17
```

Let’s start a UNIX Shell and review what we learned…

Next Lecture

- Introduction to debugging
- Scientific method


/
bin
users
boot
ggillespie
foo.txt
CSE15L
Bar.java
abcd
bin
users
tmp
ggillespie
adir
foo.txt
CSE15L
Bar.java
abcd