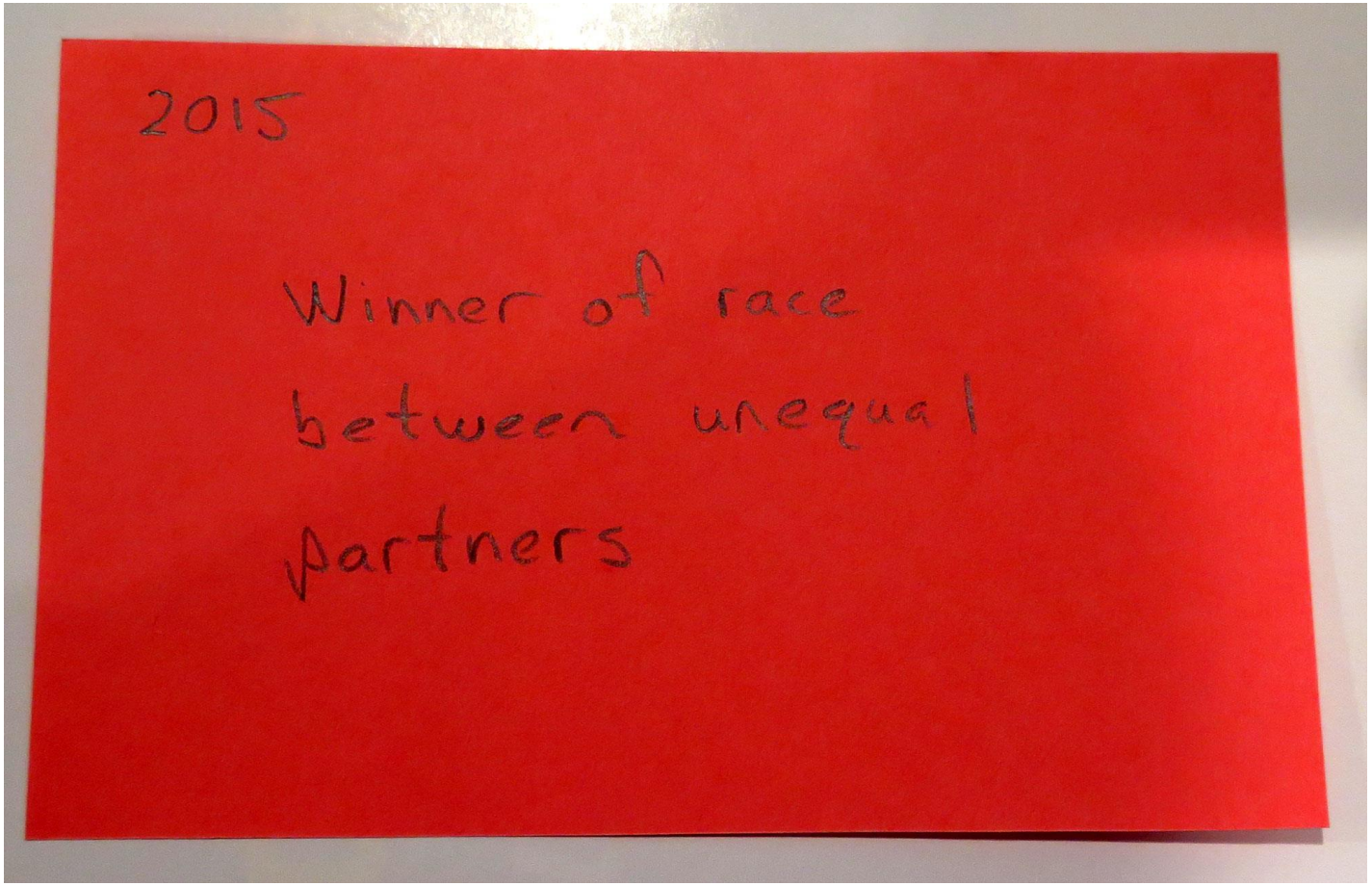
Lesson Module Checklist

- Slides
- WB converted

- Flash cards
- Page numbers
- 1st minute quiz
- Web Calendar summary
- Web book pages
- Commands

- Lock turnin directory at midnight
- Opus - hide script tested
- Practice test ready on Blackboard for 11 AM
- P2 Test system online and unlocked

- 9V backup battery for microphone
- Backup slides, CCC info, handouts on flash drive


2015
Winner of race
between unequal
partners

Student Learner Outcomes

1. Navigate and manage the UNIX/Linux file system by viewing, copying, moving, renaming, creating, and removing files and directories.
2. Use the UNIX features of file redirection and pipelines to control the flow of data to and from various commands.
3. With the aid of online manual pages, execute UNIX system commands from either a keyboard or a shell script using correct command syntax.

Introductions and Credits

Jim Griffin
- Created this Linux course
- Created Opus and the CIS VLab
- Jim's site: http://cabrillo.edu/~jgriffin/

Rich Simms
- HP Alumnus
- Started teaching this course in 2008 when Jim went on sabbatical
- Rich's site: http://simms-teach.com

And thanks to:
- John Govsky for many teaching best practices: e.g. the First Minute quizzes, the online forum, and the point grading system (http://teacherjohn.com/)

Student checklist
(How to attend from home or in the classroom)

1) Browse to the CIS 90 website Calendar page
   - http://simms-teach.com
   - Click CIS 90 link on left panel
   - Click Calendar link near top of content area
   - Locate today's lesson on the Calendar

2) Download the presentation slides for today's lesson for easier viewing

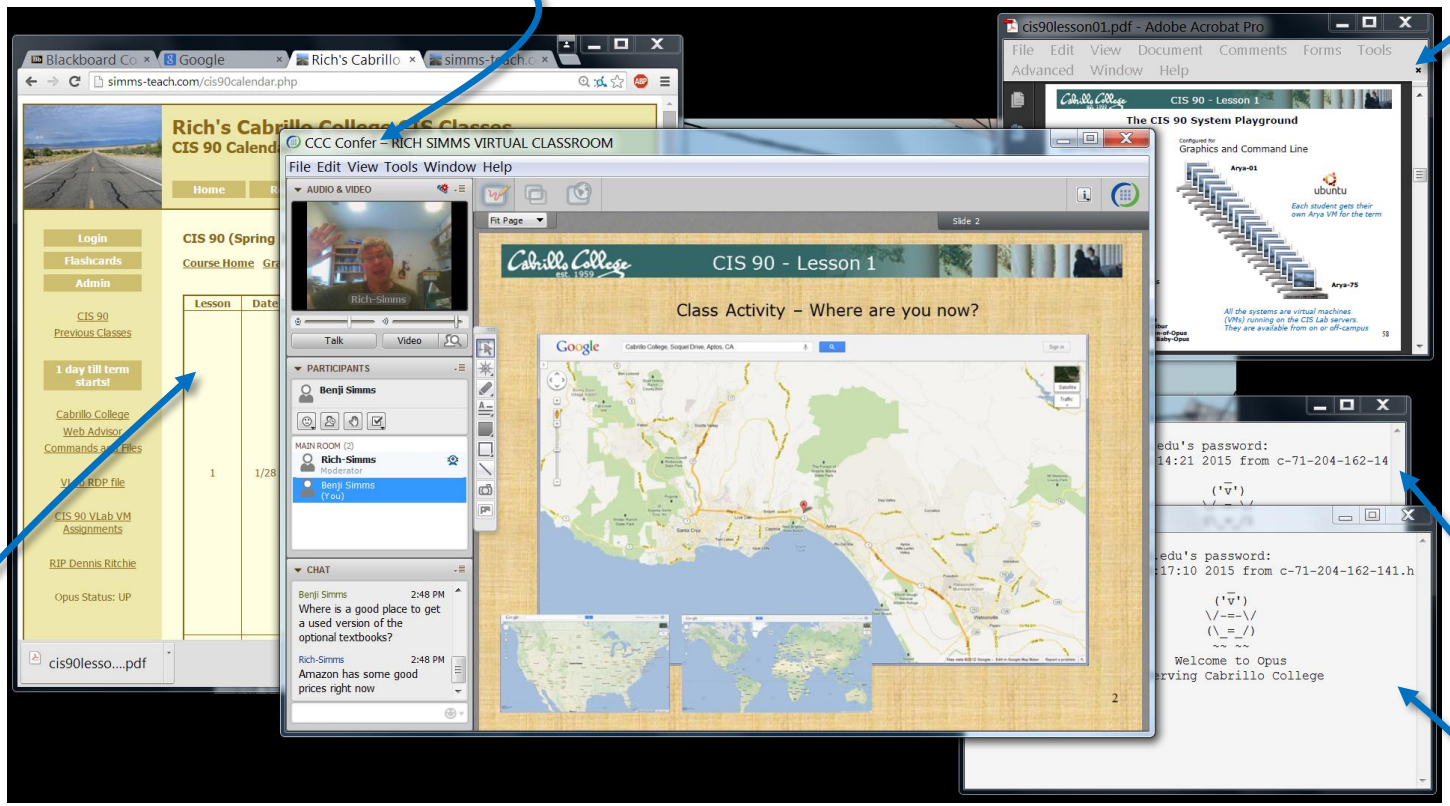
3) Click Enter virtual classroom to join CCC Confer session

4) Connect to Opus using Putty or ssh command

Student checklist
(How to layout your screen when attending class)

CCC Confer

Downloaded PDF of Lesson Slides

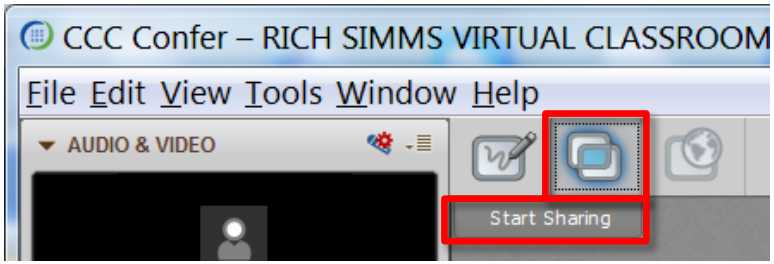


CIS 90 website Calendar page

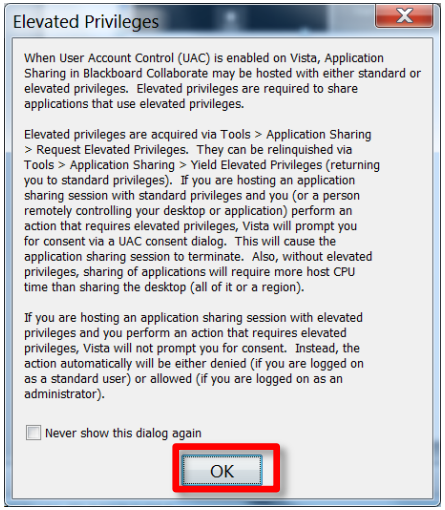
One or more login sessions to Opus

Student checklist
(To share your desktop with the class)

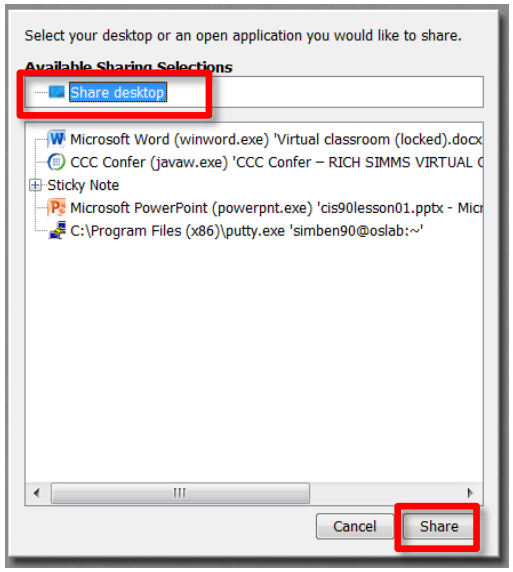
1) Instructor gives you sharing privileges



2) Click overlapping rectangles icon. If white "Start Sharing" text is present then click it as well.



3) Click OK button.



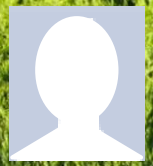
4) Select "Share desktop" and click Share button.

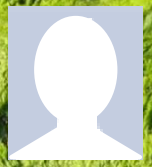
Instructor: **Rich Simms**
Dial-in: **888-886-3951**
Passcode: **136690**


Al

Tim


Emilio


Chris


Eddie

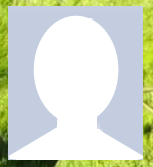

Clara

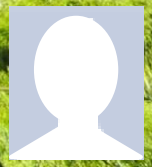

Ryan

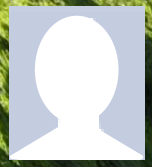

Ethan

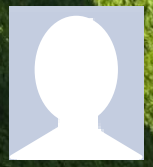

Monte


Cameron

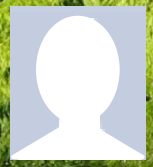

Tanner


Tess


Ahmad


Mike F.

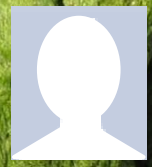

Mario


Roberto

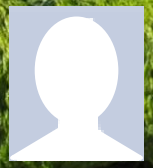

Diego

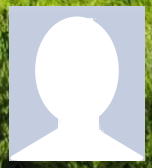

John


Efrain

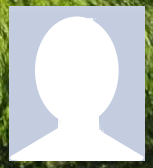

Mateo


Django


Ian


Abraham C.

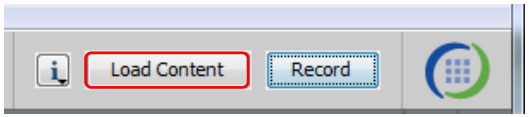

Benji


Abraham N.

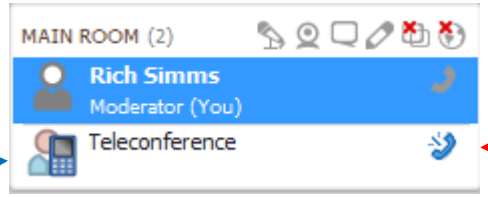
Email me (risimms@cabrillo.edu) a relatively current photo of your face for 3 points extra credit

Rich's CCC Confer checklist - setup

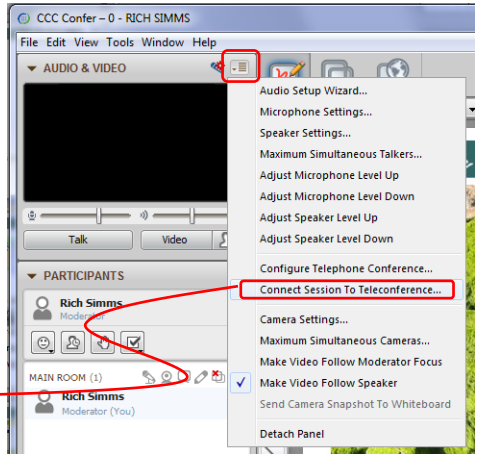
[ ] Preload White Board



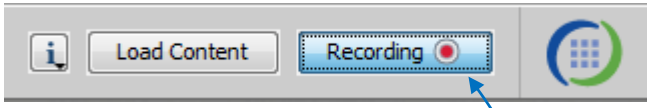
[ ] Connect session to Teleconference



Session now connected to teleconference



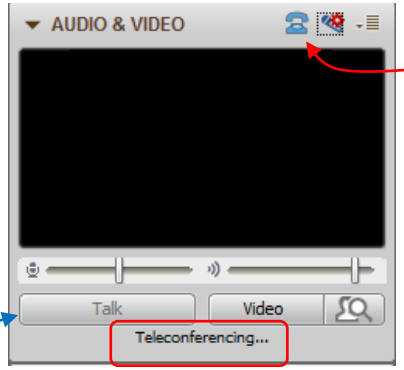
[ ] Is recording on?



Red dot means recording

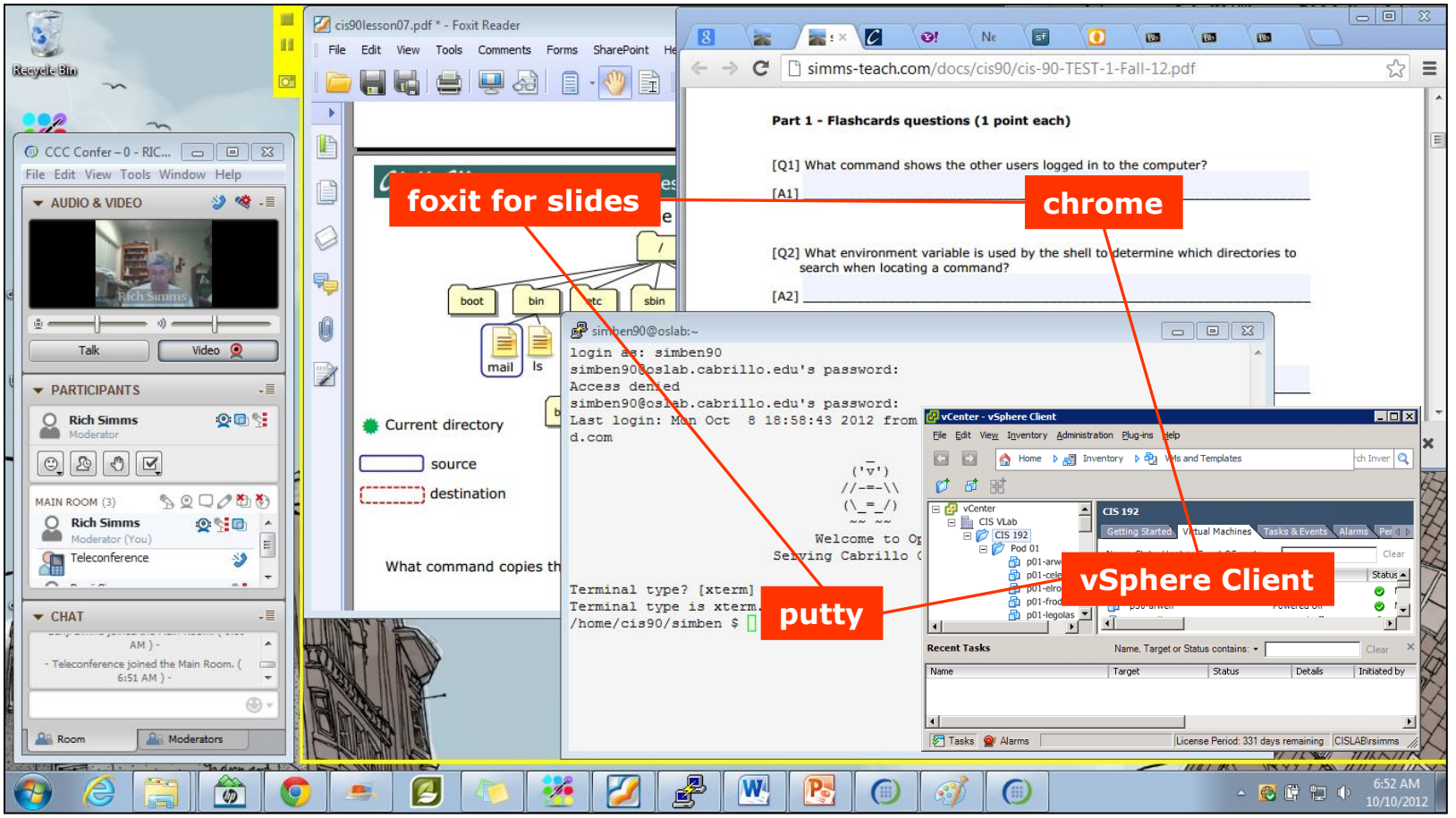
[ ] Use teleconferencing, not mic

Should be greyed out



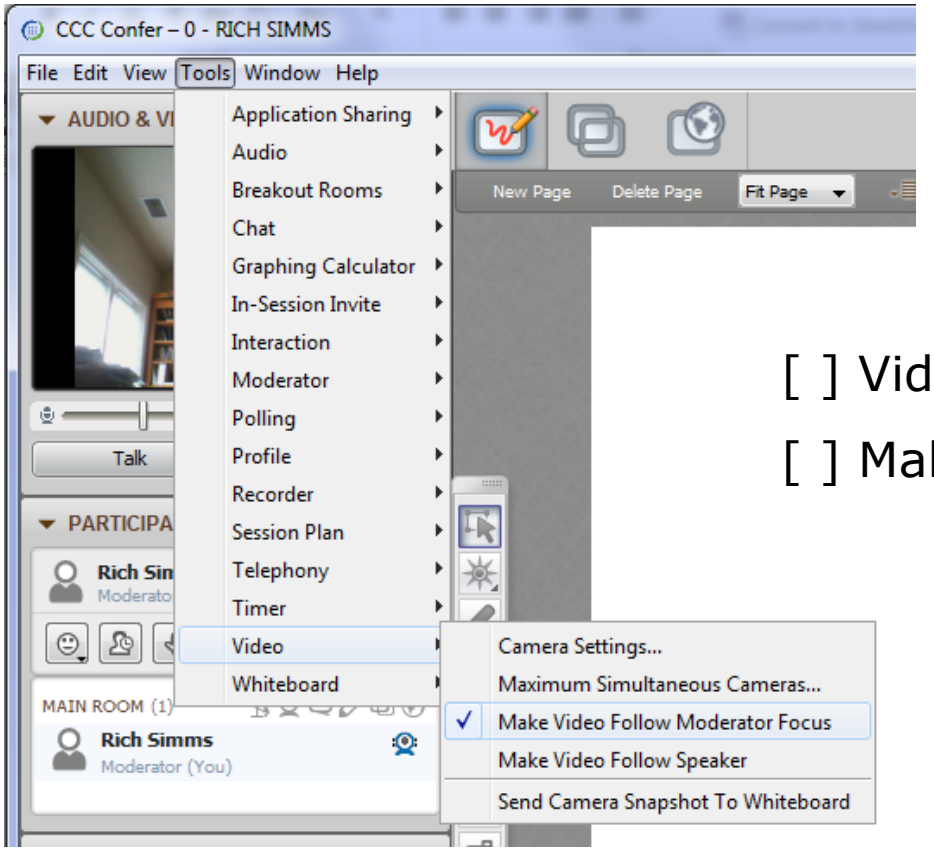
Should show as this live "off hook" telephone handset icon and the Teleconferencing … message displayed

Rich's CCC Confer checklist - app layout



[ ] layout and share apps

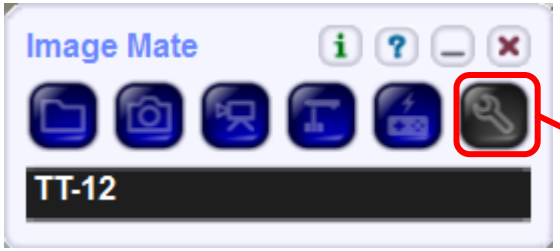
Rich's CCC Confer checklist - video

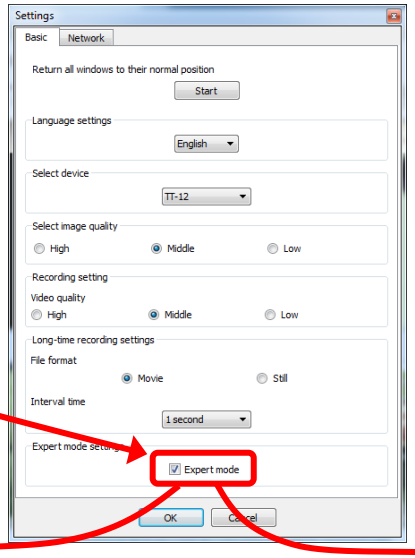


[ ] Video (webcam)

[ ] Make Video Follow Moderator Focus

Rich's CCC Confer checklist - Elmo



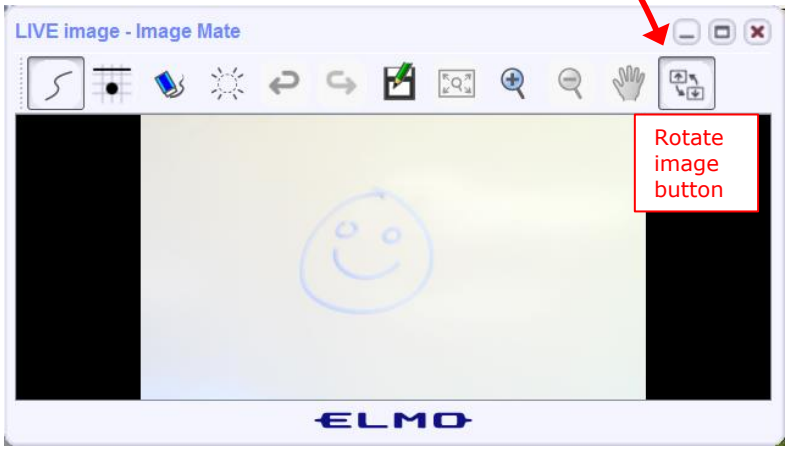


The "rotate image" button is necessary if you use both the side table and the white board.

Quite interesting that they consider you to be an "expert" in order to use this button!

Elmo rotated down to view side table

Elmo rotated up to view white board

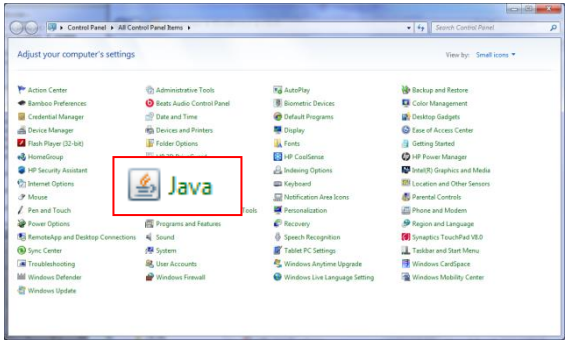


Run and share the Image Mate program just as you would any other app with CCC Confer

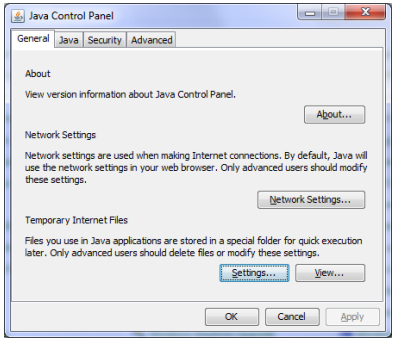
Rich's CCC Confer checklist - universal fix

Universal Fix for CCC Confer:
1) Shrink (500 MB) and delete Java cache
2) Uninstall and reinstall latest Java runtime

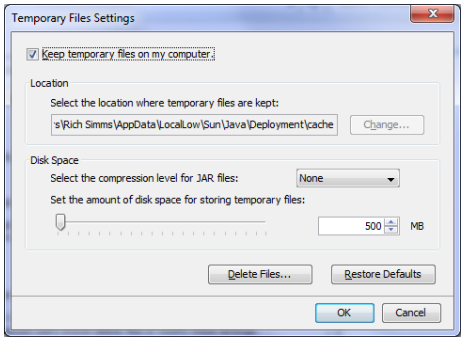
Control Panel (small icons)



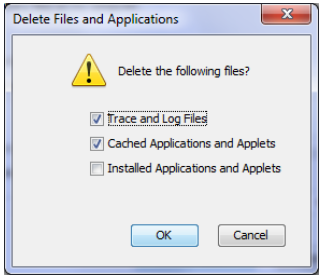
General Tab > Settings…



500MB cache size



Delete these



Google Java download

Quiz

Please answer these questions **in the order** shown:

See electronic white board

email answers to: risimms@cabrillo.edu

(answers must be emailed within the first few minutes of class for credit)

Review

Objectives	Agenda
• Get ready for the next test • Practice skills • Introduction to processes	• Quiz • Questions • Housekeeping • Linux systems at school and at home • More on I/O • All Together Now Example • Subtle Differences • Errors • 2>&1 • C & C++ program I/O • umask review • Pipeline practice • Pipeline and redirection practice • xargs • Things that hide • Flashcards • Practice test • Wrap up

Sound Check

*Students that dial-in should mute their line using *6 to prevent unintended noises distracting the web conference.*

*Instructor can use *96 to mute all student lines.*

Questions

Questions?

Lesson material?

Labs? Tests?

How this course works?

- Graded work in home directories
- Answers in /home/cis90/answers

Who questions much, shall learn much, and retain much.
- Francis Bacon

If you don't ask, you don't get.
- Mahatma Gandhi

Chinese Proverb	他問一個問題，五分鐘是個傻子，他不問一個問題仍然是一個傻瓜永遠。 *He who asks a question is a fool for five minutes; he who does not ask a question remains a fool forever.*

Lab 6
Post Mortem

Lab 6 Results

(steps where points were taken off)

Step 01 -
Step 02 -
Step 03 - x
Step 04 -
Step 05 - xx
Step 06 -
Step 07 -
Step 08 -
Step 09 - xxxxx xx
Step 10 - xxx
Step 11 - x
Step 12 - xxxxx xxxxx
Step 13 -
Step 14 - xxx
Step 15 -
Step 16 - xx
Step 17 - xx
Step 18 -
Step 19 - x
Step 20 -
Step 21 -
Step 22 -
Step 23 - xxx

9. Set the permissions of your *poems* directory and its subdirectories so that you have full permissions as owner, but group and others have no write permission. Group and others should still have read and execute permission.

poems/ should be 755
poems/ should be 755*

12. Set the executable files under bin to have the following permissions:

r-xr-x---

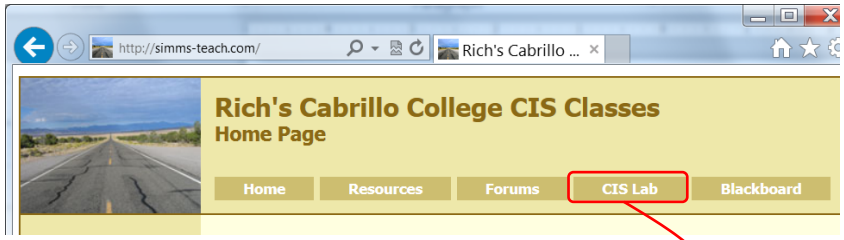
disallowing others outside the group from executing our commands.

17. Add read permission for everyone to all the files in the *misc* directory.

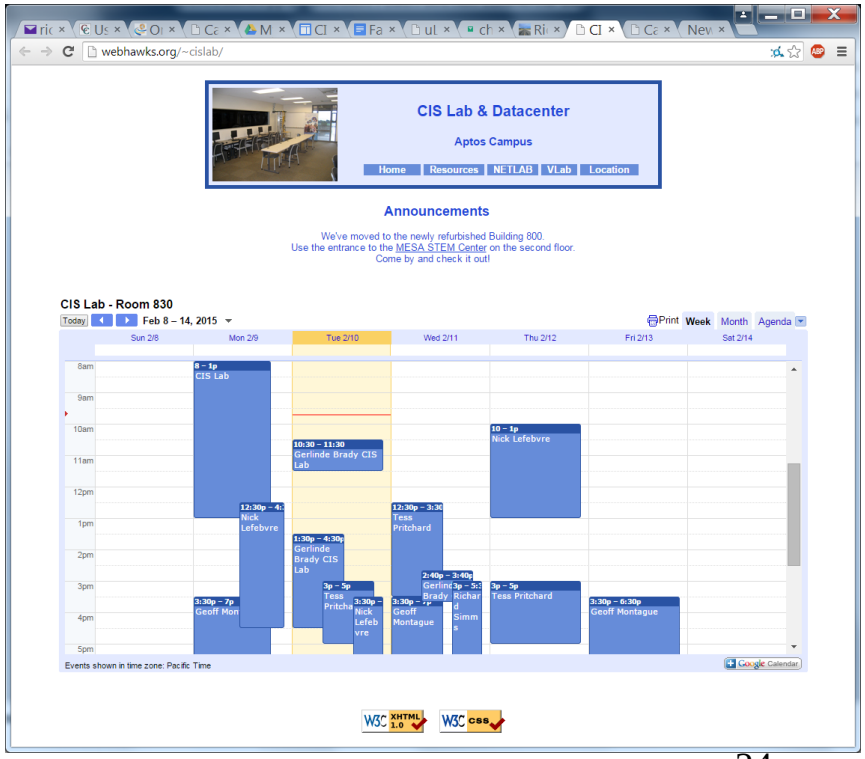
bin/ should be 550, EXCEPT enlightenment which should be 554*

For more on Steps 9-10 see the Backup Slides in Lesson 8 (module titled Lab 6 Tips)

Want some help working the labs?



If you would like some additional come over to the CIS Lab. There are student lab assistants and instructors there to help you.



Nick, Sean, and Geoff are CIS 90 Alumni.

Tess and Mike F. are in our class!

Michael M. is the other Linux instructor.

I'm in there Wednesdays 3:00-5:30pm.

CIS 90 Tutoring Available - Geoff

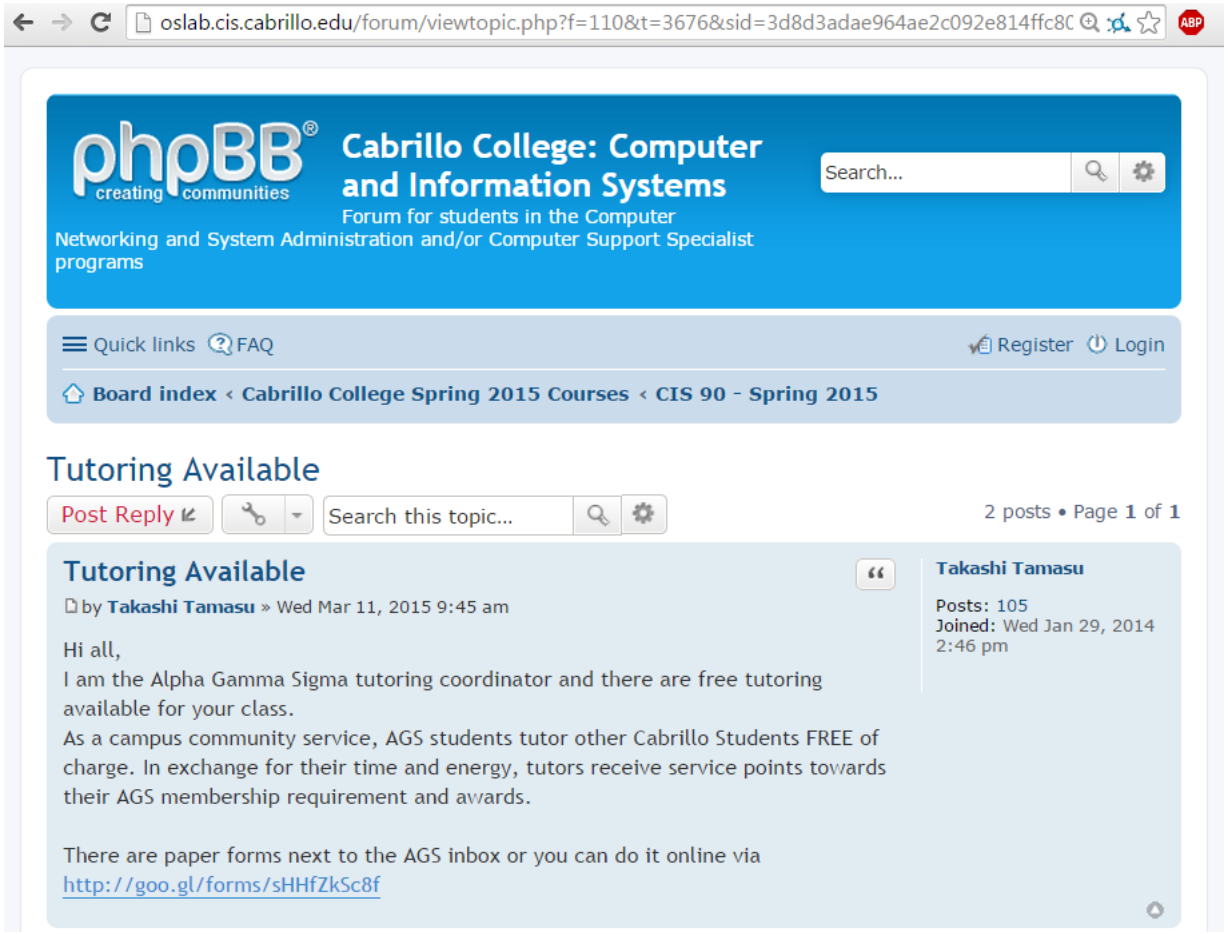
http://www.cabrillo.edu/services/tutorials/

Geoffrey Montague

All students interested in tutoring need to come directly to the Tutorials Center to schedule, register and fill out some paperwork. This is just a one-time visit.

The tutoring will take place at the STEM center.

More CIS 90 Tutoring Available - Takashi and Carter



Housekeeping

Housekeeping

1. Lab 7 due 11:59PM tonight -- **don't forget to submit your latest version!** (read your Opus email for submittal status)

2. A **check7** script is available

3. Test #2 is **two weeks from now!** Same format as before. The test will start during the last hour of class. If you work you can take it later in the day as long as it is completed by 11:59PM.

4. Blackboard Practice Test #2 available at 11AM today. It will not be available when the real test starts.

5. No lab assigned this week (so you can work on the practice test)

6. Ask your questions regarding the test on the forum **BEFORE** the next class starts!

Final Exam

Test #3 (final exam) is MONDAY May 18 7-9:50AM

Monday	5/18	**Test #3 (the final exam)** **Time** • 7:00AM - 9:50AM in Room 828 **Materials** • Test (blackboard) **CCC Confer** • Enter virtual classroom • Class archives		5 posts Lab X1 Lab X2

Extra credit labs and final posts due by 11:59PM

- All students will take the test at the same time. The test must be completed by 9:50AM.
- Working and long distance students can take the test online via CCC Confer and BlackBoard.
- Working students will need to plan ahead to take time off from work for the test.

FINAL EXAMINATIONS SCHEDULE: Spring 2015 May 18 to May 23

Daytime Classes: All times in bold refer to the beginning times of classes. **MW/Daily** means Monday alone, Wednesday alone, Monday and Wednesday **or any 3** or more days in any combination. **TTh** means Tuesday alone, Thursday alone, or Tuesday and Thursday. **Classes meeting other combinations of days and/or hours not listed must have a final schedule approved by their Division Dean.**

STARTING CLASS TIME/DAY(S)	EXAM HOUR	EXAM DATE
Classes starting between:		
6:30 am and 8:55 am, MW/Daily	7:00 am-9:50 am	Wednesday, May 20
9:00 am and 10:15 am, MW/Daily	7:00 am-9:50 am	Monday, May 18
10:20 am and 11:35 am, MW/Daily	10:00 am-12:50 pm	Wednesday, May 20
11:40 am and 12:55 pm, MW/Daily	10:00 am-12:50 pm	Monday, May 18
1:00 pm and 2:15 pm, MW/Daily	1:00 pm-3:50 pm	Wednesday, May 20
2:20 pm and 3:35 pm, MW/Daily	1:00 pm-3:50 pm	Monday, May 18
3:40 pm and 5:30 pm, MW/Daily	4:00 pm-6:50 pm	Wednesday, May 20
6:30 am and 8:55 am, TTh	7:00 am-9:50 am	Thursday, May 21
9:00 am and 10:15 am, TTh	7:00 am-9:50 am	Tuesday, May 19
10:20 am and 11:35 am, TTh	10:00 am-12:50 pm	Thursday, May 21
11:40 am and 12:55 pm, TTh	10:00 am-12:50 pm	Tuesday, May 19
1:00 pm and 2:15 pm, TTh	1:00 pm-3:50 pm	Thursday, May 21
2:20 pm and 3:35 pm, TTh	1:00 pm-3:50 pm	Tuesday, May 19
3:40 pm and 5:30 pm, TTh	4:00 pm-6:50 pm	Thursday, May 21
Friday am	9:00 am-11:50 am	Friday, May 22
Friday pm	1:00 pm-3:50 pm	Friday, May 22
Saturday am	9:00 am-11:50 am	Saturday, May 23
Saturday pm	1:00 pm-3:50 pm	Saturday, May 23

Evening Classes: For the final exam schedule, Evening Classes are those that begin at 5:35 pm or later. Also, **"M & W"** means the class meets on **BOTH** Monday and Wednesday. **"T & TH"** means the class meets on **BOTH** Tuesday and Thursday. The following schedule applies to all Evening Classes.

EVENING FINAL SCHEDULE:

Classes beginning at 5:35 pm or later on:	
Monday Only OR "M & W"	6:00 pm-8:50 pm
Tuesday Only OR "T & TH"	6:00 pm-8:50 pm
Wednesday Only	7:00 pm-9:50 pm
Thursday Only	7:00 pm-9:50 pm
Friday Only	6:00 pm-8:50 pm

CIS 90 Introduction to UNIX/Linux

Provides a technical overview of the UNIX/Linux operating system, including hands-on experience with commands, files, and tools. Prerequisite: CIS 72. Transfer Credit: CSU.

Section	Days	Times	Units	Instructor	Room
88445	W	09:00AM-12:05PM	3.00	R.Simms	OL
&	Arr.	Arr.		R.Simms	OL

Section 88445 is an ONLINE course. Meets weekly throughout the semester online during the scheduled times by remote technology with an additional 50 min arranged online lab per week. For details, see instructor's web page at go.cabrillo.edu/online.

88446	W	09:00AM-12:05PM	3.00	R.Simms	828
&	Arr.	Arr.		R.Simms	OL

Section 88446 is a Hybrid ONLINE course. Meets weekly throughout the semester at the scheduled times with an additional 50 min online lab per week. For details, see instructor's web page at go.cabrillo.edu/online.

Spring Resume Workshop and Job Fair

http://cabrillo.edu/services/jobs/workshops.html

Student Workshops

Spring 2015:

Resume Preparation - April 10, 12-2, SAC 202.

Interview Preparation - April 22, 12-2, SAC 202

2015 Cabrillo College Job Fair

Wednesday April 29, 10-2

Cabrillo College Quad

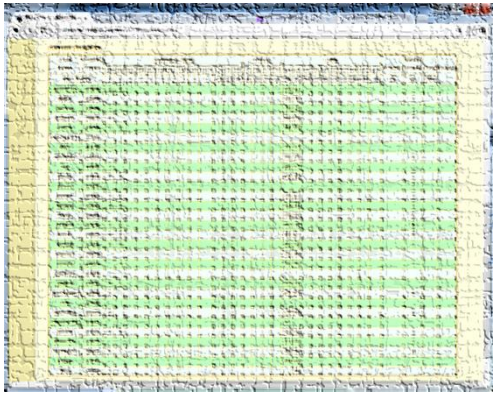
Open to Current Students, Alumni, and the Public

http://cabrillo.edu/services/jobs/fair.html

Monitoring your grades

Points that could have been earned:	
6 quizzes:	18 points
6 labs:	180 points
1 test:	30 points
2 forum quarters:	40 points
Total:	**268 points**

The CIS 90 website



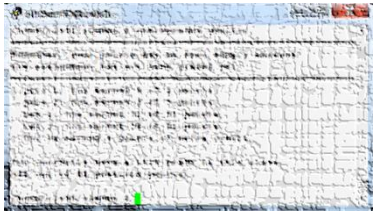
http://simms-teach.com/cis90grades.php

Send me your survey to get your LOR code name.

Or on Opus

`checkgrades` *`codename`*

(where codename is your LOR codename)



The checkgrades script was written by Jessie a past CIS 90 Alumnus

Percentage	Total Points	Letter Grade	Pass/No Pass
90% or higher	504 or higher	A	Pass
80% to 89.9%	448 to 503	B	Pass
70% to 79.9%	392 to 447	C	Pass
60% to 69.9%	336 to 391	D	No pass
0% to 59.9%	0 to 335	F	No pass

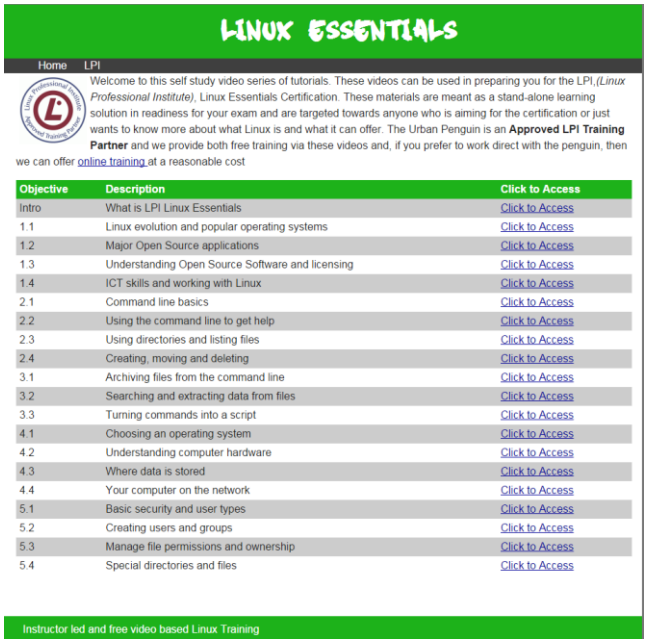
At the end of the term I'll add up all your points and assign you a grade using this table

LPI Linux Essentials Certificate

Linux Essentials Certificate of Achievement

Objective	# of Questions	Cabrillo	Urban Penguin	NDG Linux Essentials
Topic 1: The Linux Community and a Career in Open Source				
1.1 Linux Evolution and Popular Operating Systems	2	CIS90 Lesson 1	1.1	Module 1
1.2 Major Open Source Applications	2	CIS90 Lesson 1	1.2	Module 2
1.3 Understanding Open Source Software and Licensing	1	CIS90 Lesson 1	1.3	Module 2
1.4 ICT Skills and Working in Linux	2	not covered	1.4	Module 3
Topic 2: Finding Your Way on a Linux System				
2.1 Command Line Basics	2	CIS90 Lesson 2	2.1	Module 4
2.2 Using the Command Line to Get Help	2	CIS90 Lesson 2	2.2	Module 5
2.3 Using Directories and Listing Files	2	CIS 90 Lesson 4	2.3	Module 6
2.4 Creating, Moving and Deleting Files	2	CIS90 Lesson 5	2.4	Module 6
Topic 3: The Power of the Command Line				
3.1 Archiving Files on the Command Line	2	CIS 90 Lesson 14	3.1	Module 7
3.2 Searching and Extracting Data from Files	4	CIS 90 Lesson 8	3.2	Module 8
3.3 Turning Commands into a Script	4	CIS 90 Lesson 13 & 14	3.3	Module 9
Topic 4: The Linux Operating System				
4.1 Choosing an Operating System	1	not covered	4.1	Module 1
4.2 Understanding Computer Hardware	2	CIS 90 Lesson 1	4.2	Module 10
4.3 Where Data is Stored	3	CIS 90 Lesson 1	4.3	Module 11
4.4 Your Computer on the Network	2	CIS 192	4.4	Module 12
Topic 5: Security and File Permissions				
5.1 Basic Security and Identifying User Types	2	CIS 191	5.1	Module 13
5.2 Creating Users and Groups	2	CIS 191	5.2	Module 14
5.3 Managing File Permissions and Ownership	2	CIS 90 Lesson 7	5.3	Module 15
5.4 Special Directories and Files	1	CIS 90 Lesson 4	5.4	Module 16

The Urban Penguin



http://www.theurbanpenguin.com/lpi/le.html

No registration, no logging in, just click and watch the videos

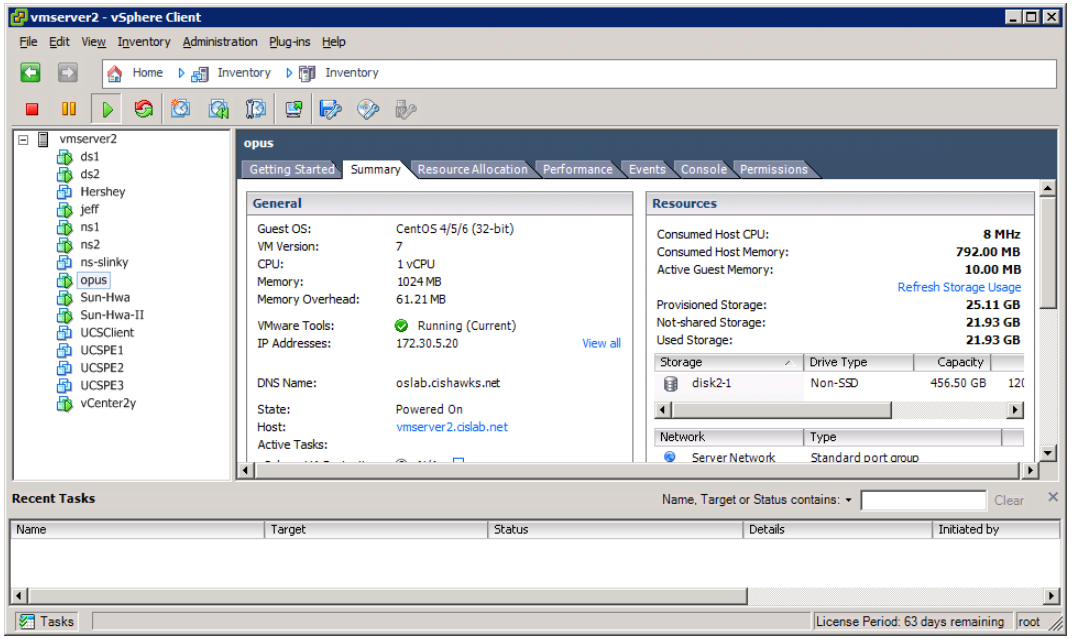
NDG Linux Essentials via Cisco Networking Academy

https://www.netacad.com/

Complete course with reading, live VM and tests. Contact me if you would like a student account for the NDG Linux Essentials course.

Linux at School

Our Opus server on campus



Opus is a VM running on one of the ESXi servers in the CIS Lab

SSH access to Opus
hostname: oslab.cishawks.net (port 2220)

Your own Linux Systems

USB "Live" Linux Multi-Boot USB Flash Drive

Windows

CentOS

Kali

YUMI formatted Flash Drive
(www.pendrivelinux.com)

Linux Mint

Ubuntu

Allows you to use or try out Linux on an existing computer without installing it

USB "Live" Linux Boot USB Drive

Allows you to use or try out Linux on an existing computer without installing it

1) 

Get the Linux distros of your choice
See: http://iso.linuxquestions.org/

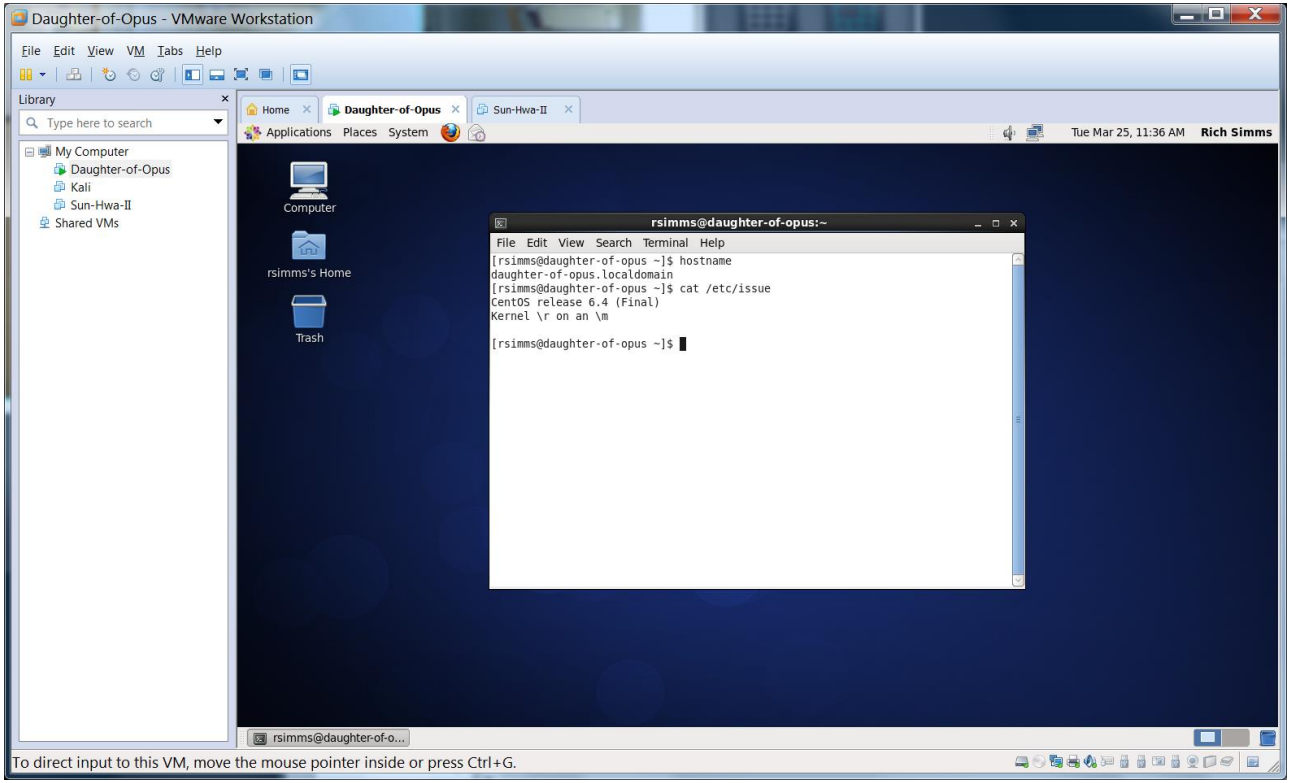
2) 

Get a USB flash drive

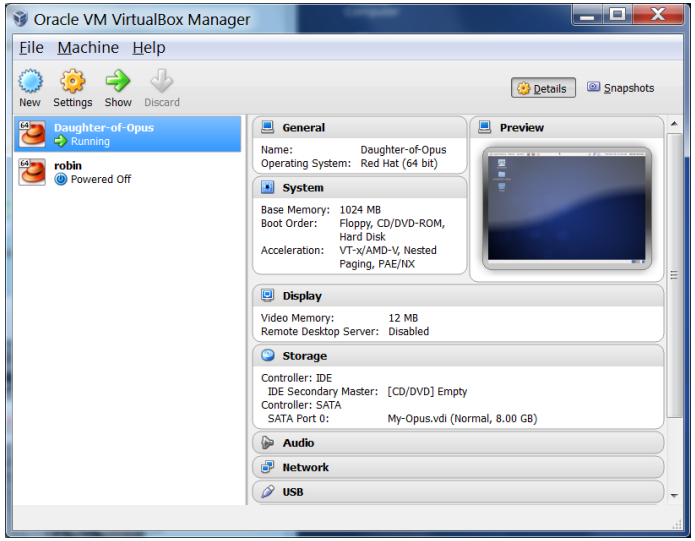
3) Google "boot live linux from usb" for instructions
or see
http://www.pendrivelinux.com/yumi-multiboot-usb-creator/

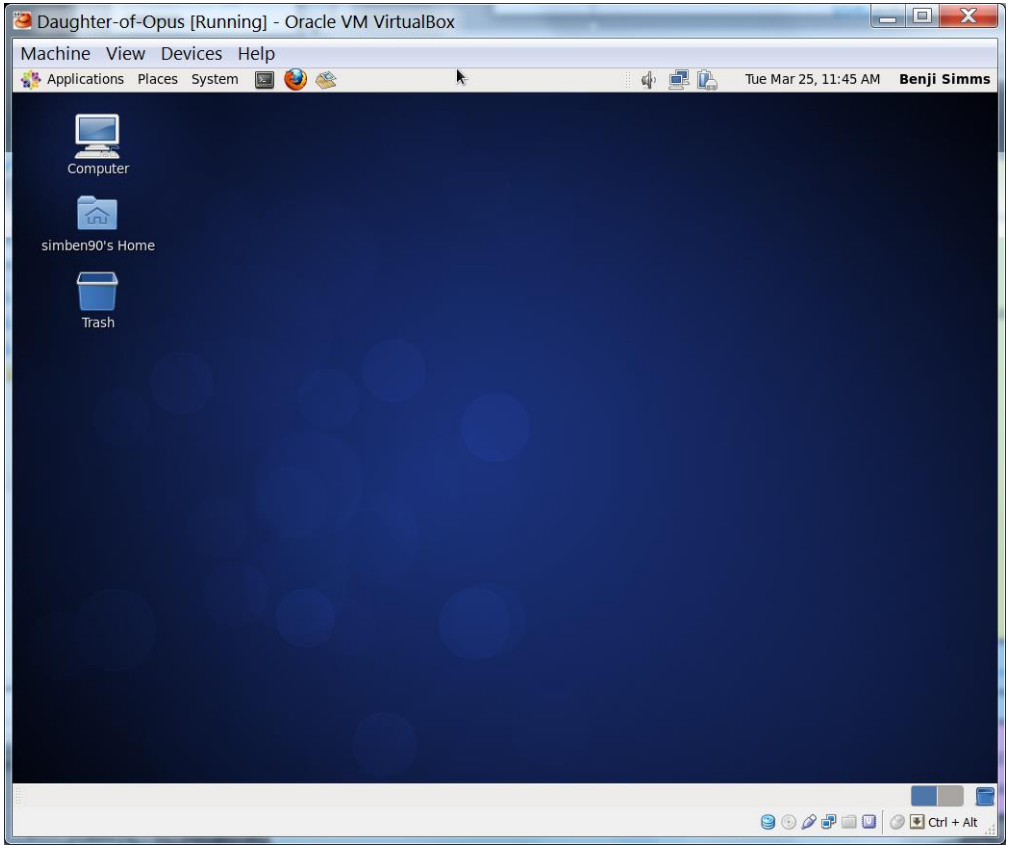
4) 

Configure your BIOS to boot from USB then select the Operating System as your computer boots up



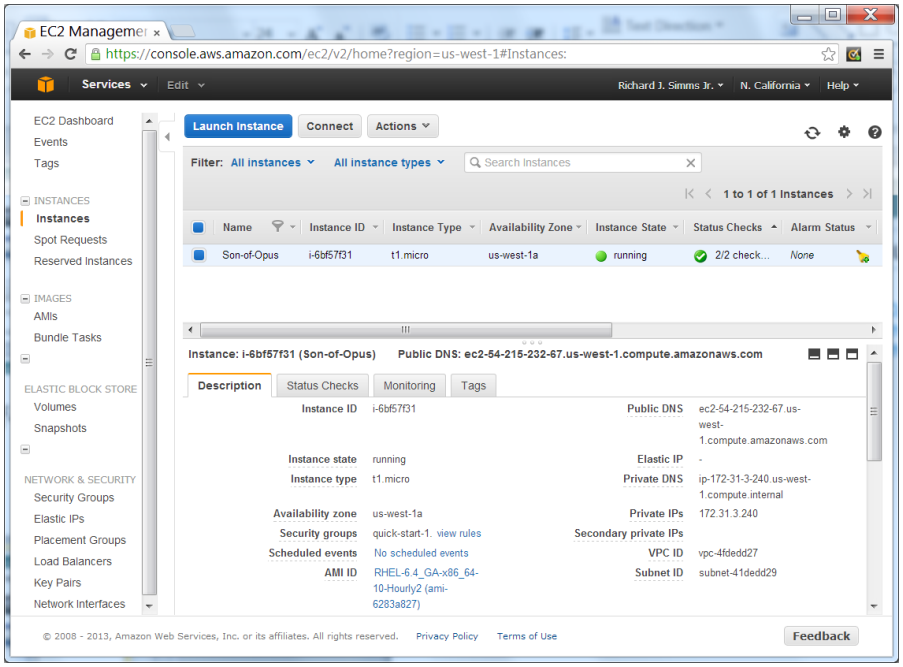
One Daughter-of-Opus is a VM running on my laptop using VMware Workstation (expires in one year)





This Daughter-of-Opus is a VM running on my laptop using Oracle VirtualBox (never expires)

Son-of-Opus
Amazon Web Services



Son-of-Opus is a VM running on Amazon Web Services

ssh -p 2220 <username>@son-of-opus.simms-teach.com

Baby-Opus
Debian 7 (Raspian) Linux Server

Raspberry Pi

Baby-Opus is running on my Raspberry Pi

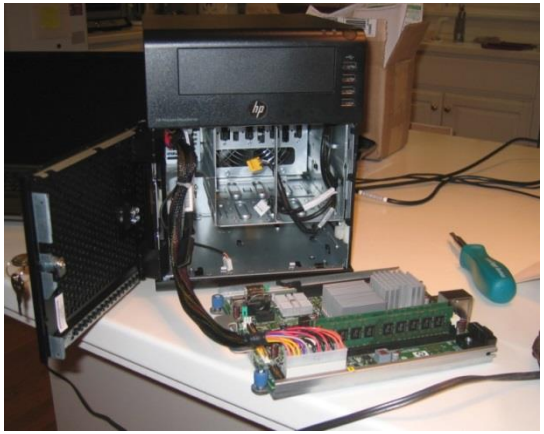
ssh <*username*>@<*ip-address*>

```
root@baby-opus:~# ./led-who-start
root@baby-opus:~# ./led-who-stop
```

```
NoPar#show ip dhcp binding

MAC b8:27:eb:b7:b3:99
Reservation for 172.30.1.31
```

Small Form Factor Servers

HP Microserver

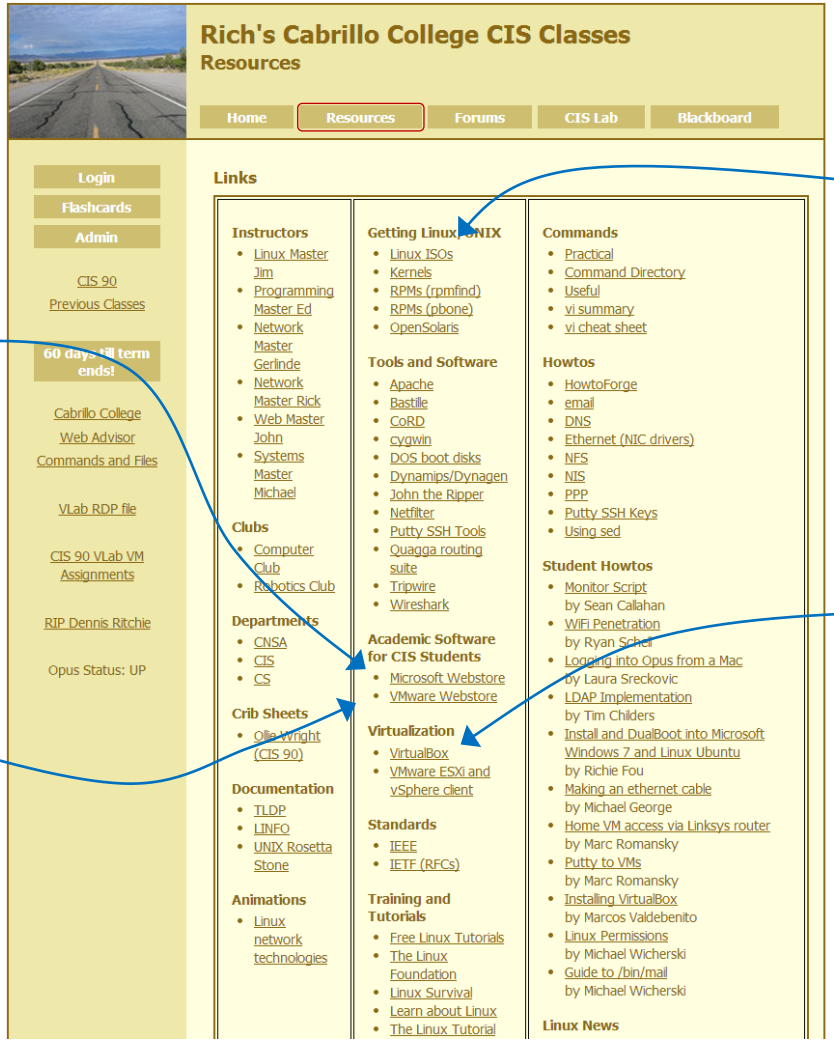


vmware ESXi

Inexpensive "bare bones" servers are available that come without hard drives or an operating system

Microsoft Software (Academic)

http://simms-teach.com/resources.php



Linux Distros (ISOs)

VirtualBox

VMware Software (Academic)

More on I/O
(input/output)

Input and Output
File Redirection

The 3 standard UNIX file descriptors:

Name	Integer Value
stdin (**st**an**d**ard **in**)	0
stdout (**st**an**d**ard **out**)	1
stderr (**st**an**d**ard **err**or)	2

*Every process is provided with three file descriptors: **stdin**, **stdout** and **stderr***

Input and Output
File Redirection

*The input and output of a program can be **redirected** to and from other files as follows:*

0< *filename*

Redirects **stdin**, input will now come from *filename* rather than the keyboard.

1> *filename*

Redirects **stdout**, output will now go to *filename* instead of the terminal.

2> *filename*

Redirects **stderr**, error messages will now go to *filename* instead of the terminal.

>> *filename*

Redirects **stdout**, output will now be appended to *filename*.

The redirection is specified on the command line

Shell prints this to prompt user to enter a command

Shell parses this command line

***Redirection** connects **stdin**, **stdout** and **stderr** to non-default devices*

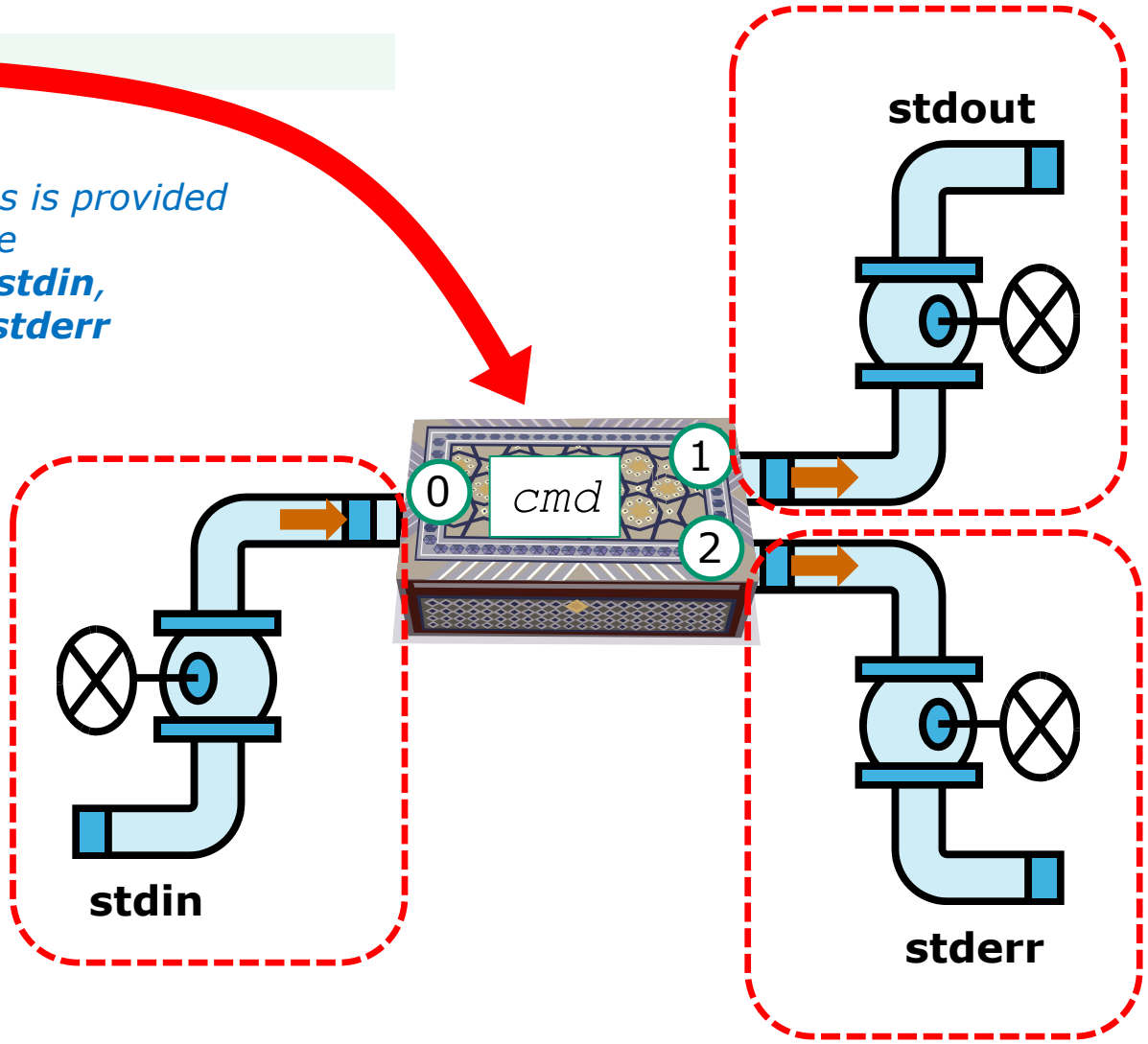
Examples

```
/home/cis90/simben $ cat
/home/cis90/simben $ cat -A letter
/home/cis90/simben $ cat < letter
/home/cis90/simben $ cat -b < letter > out
/home/cis90/simben $ cat bogus 2> /dev/null
/home/cis90/simben $ cat -e < bogus 2> /dev/null
/home/cis90/simben $ cat -e < letter > out 2> /dev/null
```

A program loaded into memory becomes a **process**

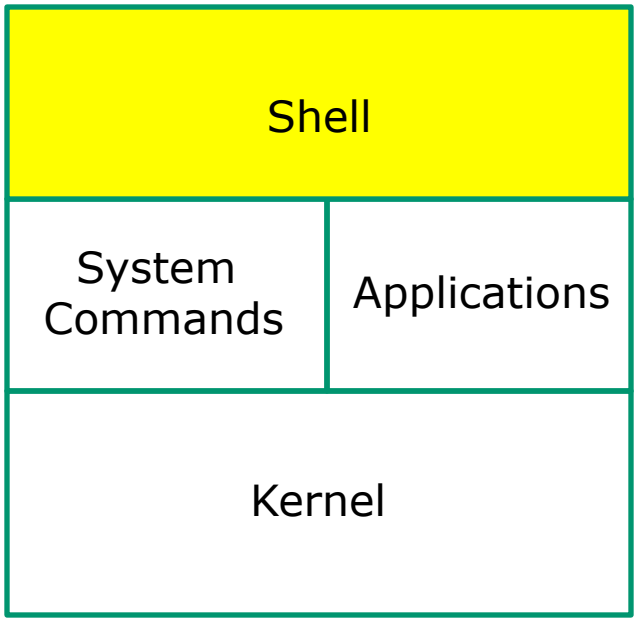
```
$ cmd
```

*Every process is provided with three file descriptors: **stdin**, **stdout** and **stderr***



All Together Now Example

Life of the Shell



1) Prompt
2) Parse
3) Search
4) Execute
5) Nap
6) Repeat

Example

1) Prompt
2) Parse
3) Search
4) Execute
5) Nap
6) Repeat

The shell begins by echoing a **prompt** string to your terminal device:

- Your specific terminal device can be identified by using the **tty** command.
- The format of the prompt is defined by the contents of the PS1 variable.

```
/home/cis90/simben $
```

In this case the PS1 variable is set to '$PWD $ ' which results in a prompt that shows the current location in the file tree followed by a blank, a $, and another blank.

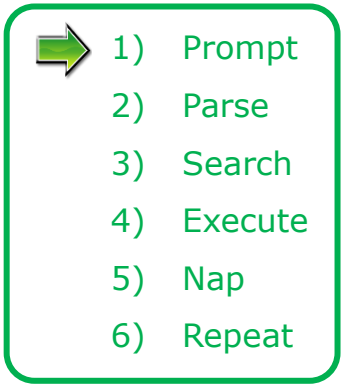
Activity

1) Prompt
2) Parse
3) Search
4) Execute
5) Nap
6) Repeat

The prompt is defined by your PS1 variable

1. Look at the contents of your PS1 variable: **echo $PS1**
2. Look at the contents of your PWD variable: **echo $PWD**
3. Send me and yourself the contents of your prompt variable:
   echo $PS1 | mail -s "my prompt" rsimms $LOGNAME
4. Paste the value of your PWD variable into the chat window when finished

Example



Following the prompt, the user then enters a command followed by the Enter key:

- The Enter key generates a <newline> which is a shell metacharacter. All metacharacters have special meanings to the shell.
- The <newline> characters instructs the shell that the command line is ready to be processed.

```
/home/cis90/simben $ sort -r names > dogsinorder
```

*The user types in a command line followed by the **Enter** key*

1) Prompt
2) Parse
3) Search
4) Execute
5) Nap
6) Repeat

Activity

The newline character is an invisible metacharacter that triggers the shell to process the command

1. Put five characters in a file named *five*: **echo 12345 > five**
2. Show the size of your *five* file: **ls -l five**
3. Do a hex dump of your *five* file: **xxd five**
4. Put the size of your *five* file and the hex value of the newline character in the chat window.
5. Optional: Use **man ascii** to check your answer

Example

1) Prompt
2) Parse
3) Search
4) Execute
5) Nap
6) Repeat

The shell **parses** the command line entered by the user:

- The command line is carefully scanned to identify the command, options, arguments and any redirection information.
- Variables and filename expansion characters (wildcards) get processed.

```
/home/cis90/simben $ sort -r names > dogsinorder
```

Parsing results: `sort` `-r` `names` `> dogsinorder`

The command is: ***sort***
There is one option: ***-r***
There is one argument: ***names***
Redirection is: redirect ***stdout*** *to a file named* ***dogsinorder***

Example

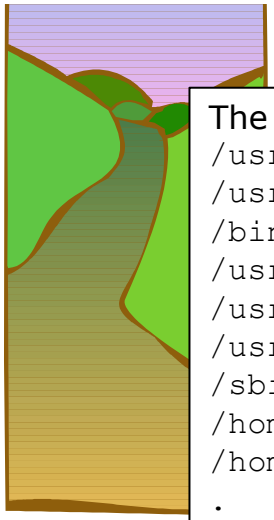
1) Prompt
2) Parse
3) Search
4) Execute
5) Nap
6) Repeat

The shell now **searches** for the command on the path:

- The path, which is an ordered list of directories, is defined by the contents of the PATH variable. Use **echo $PATH** to view.
- The shell will search in order each directory on the path to locate the command.
- If a command, such as xxxx, is not found, the shell will print:

  -bash: xxxx: command not found

- FYI, you can search for commands on the path too, like the shell does, by using the **type** command.



The ***Path*** (**echo $PATH** to show)

```
/usr/lib/qt-3.3/bin:
/usr/local/bin:
/bin:
/usr/bin:
/usr/local/sbin:
/usr/sbin:
/sbin:
/home/cis90/simben/../bin:
/home/cis90/simben/bin:
.
```

`sort` *The shell locates the sort command in the /bin directory which is the third directory of a CIS 90 student's path.*

Example

argument

option

redirection

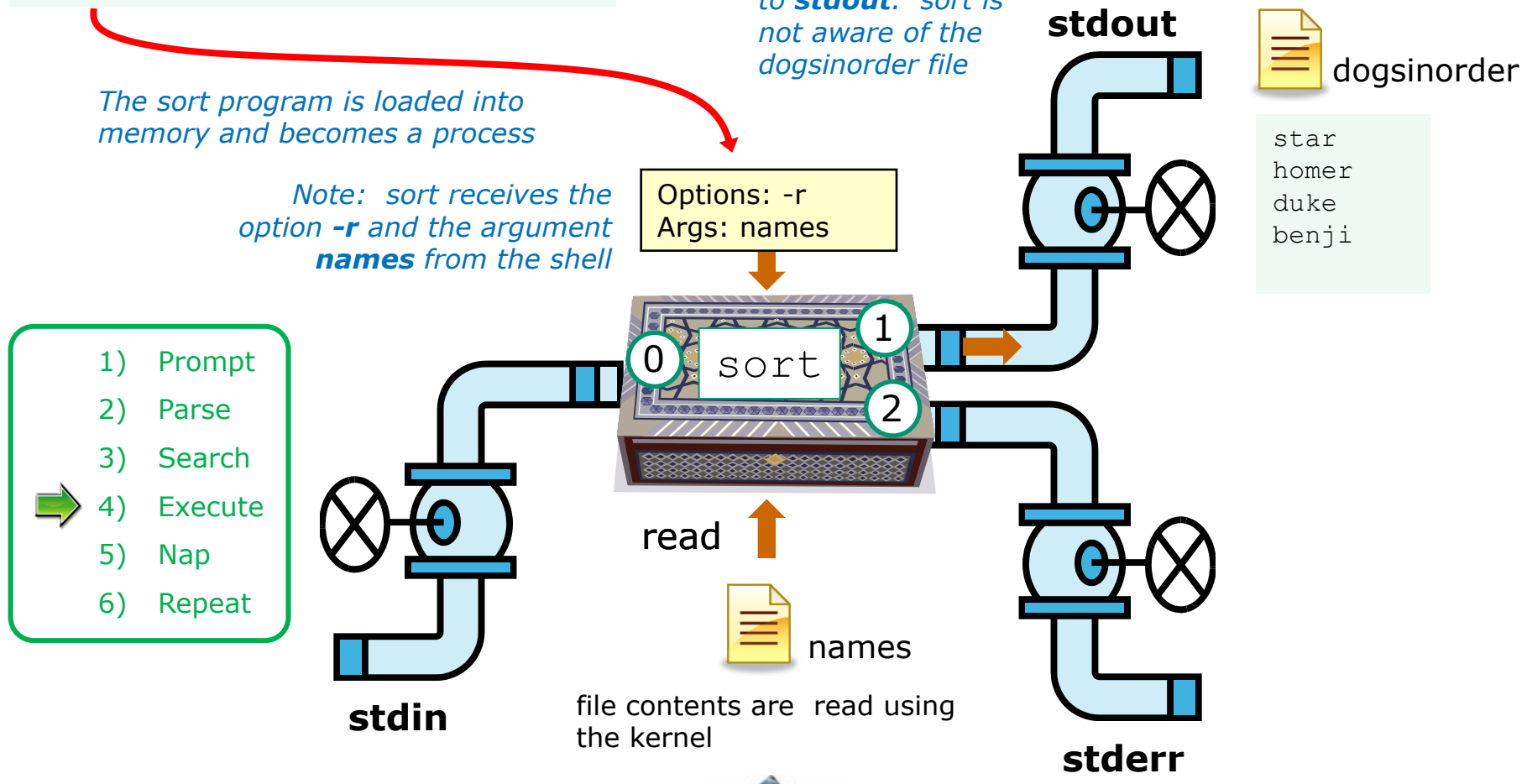
```
$ sort -r names > dogsinorder
```

*The shell connects **stdout** to the dogsinorder file*

*sort sends it's output to **stdout**. sort is not aware of the dogsinorder file*

The sort program is loaded into memory and becomes a process

*Note: sort receives the option **-r** and the argument **names** from the shell*



1) Prompt
2) Parse
3) Search
4) Execute
5) Nap
6) Repeat

Activity

1) Prompt
2) Parse
3) Search
4) Execute
5) Nap
6) Repeat

```
$ sort -r names > dogsinorder
```

What two text strings parsed by the shell were passed to the sort command to process?

Put your answer in the chat window

Example

1) Prompt
2) Parse
3) Search
4) Execute
5) Nap
6) Repeat

While the sort process executes, the shell sleeps

Example

1) Prompt
2) Parse
3) Search
4) Execute
5) Nap
6) Repeat

When the sort process finishes the shell wakes up and starts all over again to process the next command from the user!

Subtle Differences

What is the difference between:

head -n4 letter

and

head -n4 < letter

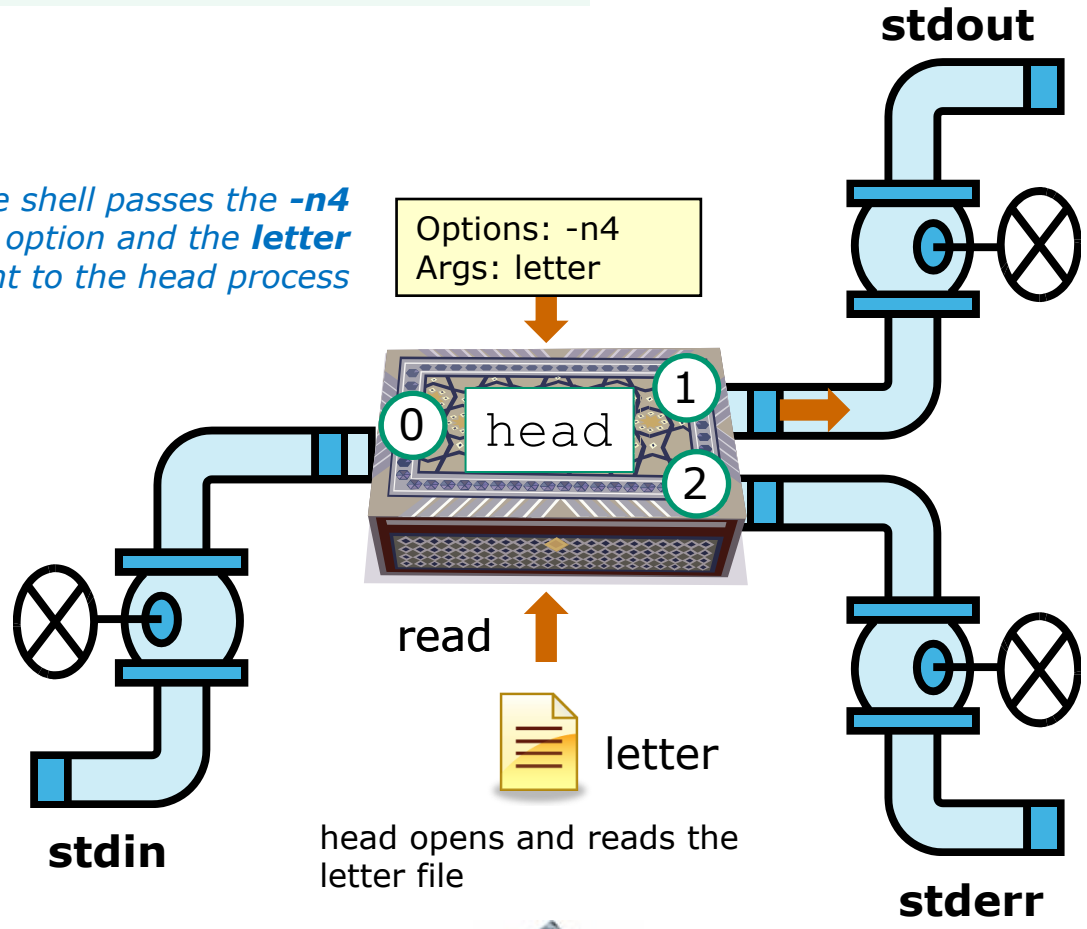
```
/home/cis90/simben $ head -n4 letter
Hello Mother!  Hello Father!

Here I am at Camp Granada.  Things are very entertaining,
and they say we'll have some fun when it stops raining.
```

```
/home/cis90/simben $  head -n4 < letter
Hello Mother!  Hello Father!

Here I am at Camp Granada.  Things are very entertaining,
and they say we'll have some fun when it stops raining.
```

head -n4 letter

option *argument*

```
$ head -n4 letter
```




```
Hello Mother!  Hello Father!

Here I am at Camp Granada.  Things are very entertaining,
and they say we'll have some fun when it stops raining.
```

*The shell passes the **-n4** option and the **letter** argument to the head process*

head opens and reads the letter file

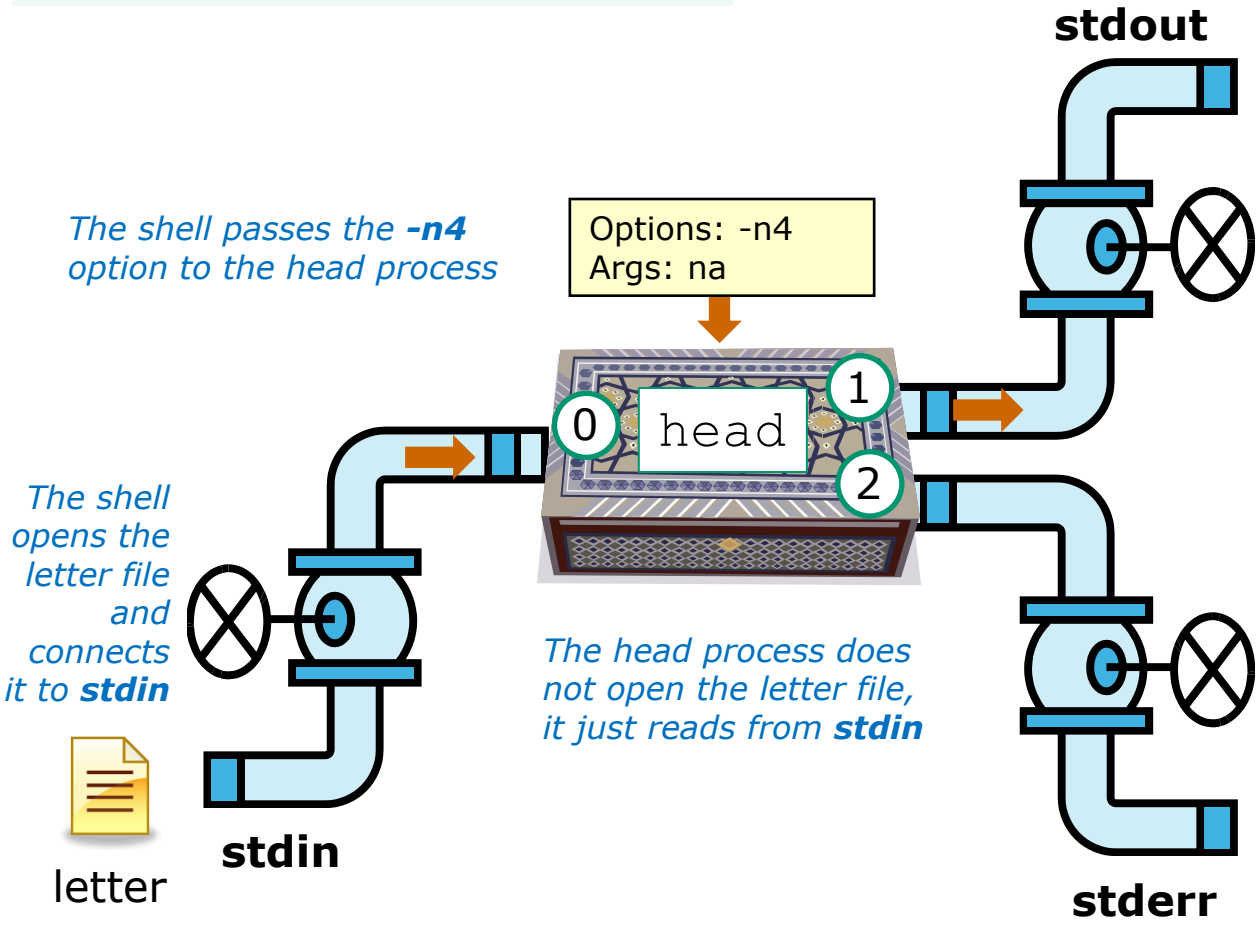
head opens and reads the letter file

head -n4 < letter

option

redirection

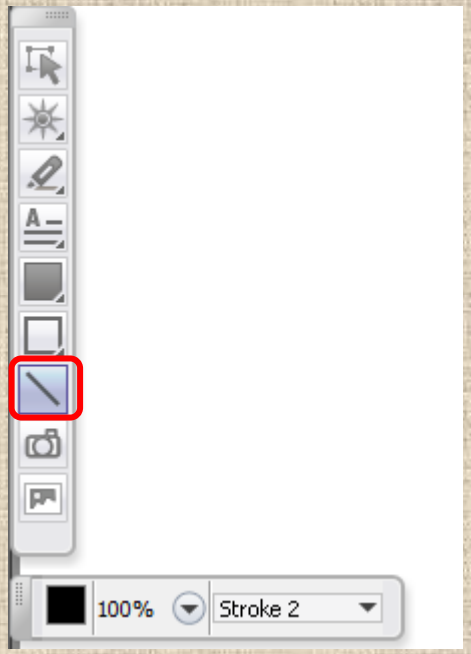
```
$ head -n4 < letter
```



Errors

Instructor: Switch to CCC Confer Whiteboard

CCC Confer Whiteboard Activity



Select the straight line drawing tool and connect the like images

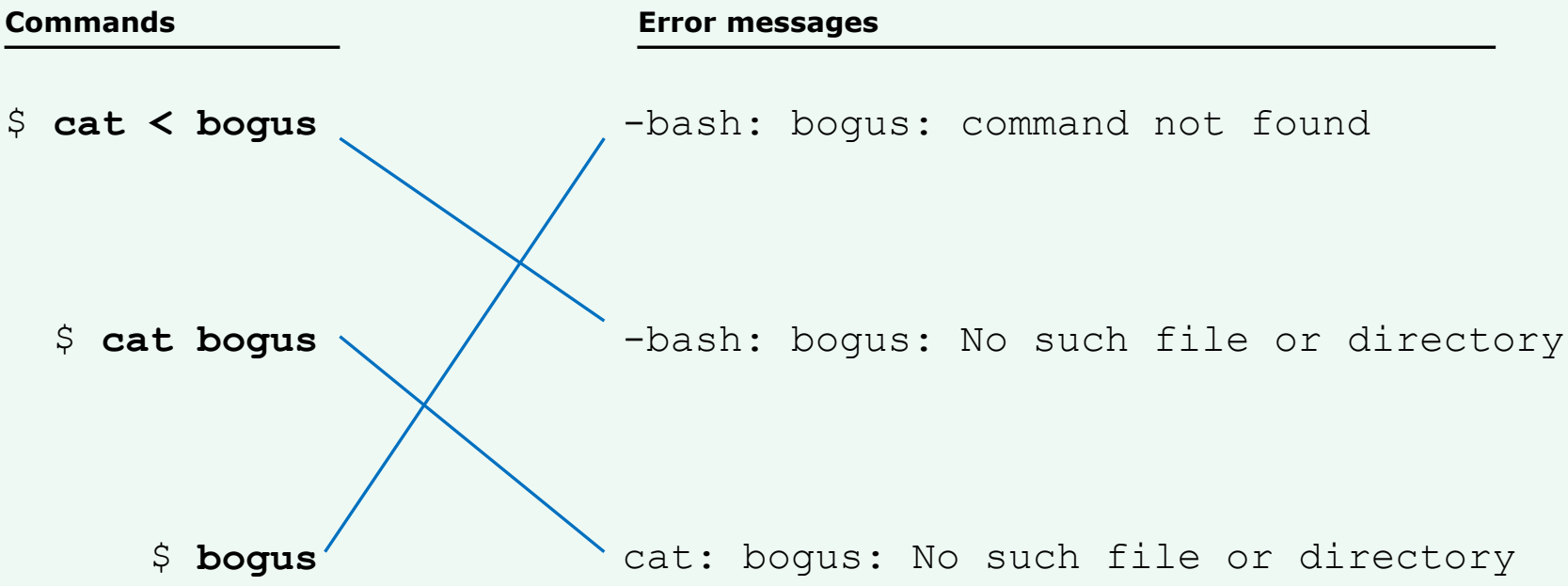
CCC Confer Whiteboard Activity

Connect with a straight line the command with the error message

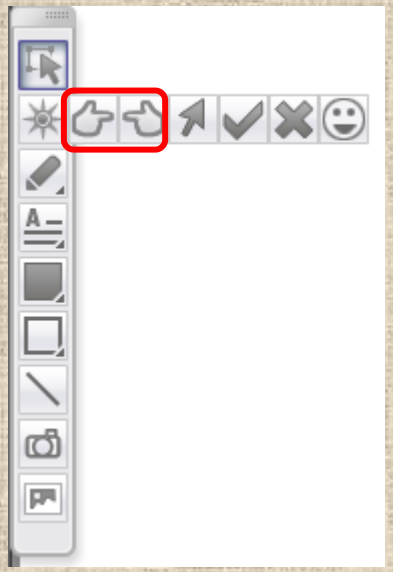
Commands	Error messages
`$ cat < bogus`	`-bash: bogus: command not found`
`$ cat bogus`	`-bash: bogus: No such file or directory`
`$ bogus`	`cat: bogus: No such file or directory`

CCC Confer Whiteboard Activity

Connect with a straight line the command with the error message



CCC Confer Whiteboard Activity



1

2

3

4

Select one of the pointing finger markers and point at the number called out by the instructor

CCC Confer Whiteboard Activity

Given: There is no file named *bogus*

```
[rsimms@oslab ~]$ cat bogus
cat: bogus: No such file or directory
```

Point your electronic finger at the shell step where the error message was generated

Shell Steps

1) Prompt

2) Parse

3) Search

4) Execute

5) Nap

6) Repeat

CCC Confer Whiteboard Activity

Given: There is no file named *bogus*

```
[rsimms@oslab ~]$ bogus
-bash: bogus: command not found
```

Point your electronic finger at the shell step where the error message was generated

Shell Steps

1) Prompt

2) Parse

3) Search

4) Execute

5) Nap

6) Repeat

CCC Confer Whiteboard Activity

Given: There is no file named *bogus*

```
[rsimms@oslab ~]$ cat < bogus
-bash: bogus: No such file or directory
```

Point your electronic finger at the shell step where the error message was generated

Shell Steps

1) Prompt
2) Parse
3) Search
4) Execute
5) Nap
6) Repeat

CCC Confer Whiteboard Activity

Given: There is no file named *bogus*

```
[rsimms@oslab ~]$ bogus < bogus
-bash: bogus: No such file or directory
```

Point your electronic finger at the shell step where the error message was generated

Shell Steps

1) Prompt

2) Parse

3) Search

4) Execute

5) Nap

6) Repeat

CCC Confer Whiteboard Activity

Given: There is no file named *bogus*

```
[rsimms@oslab ~]$ cat bogus
cat: bogus: No such file or directory
```
1) Execute

```
[rsimms@oslab ~]$ bogus
-bash: bogus: command not found
```
3) Search

```
[rsimms@oslab ~]$ cat < bogus
-bash: bogus: No such file or directory
```
2) Parse

```
[rsimms@oslab ~]$ bogus < bogus
-bash: bogus: No such file or directory
```
2) Parse

2>&1

FYI

(more on this in CIS 98)

It's descriptor clobbering time!

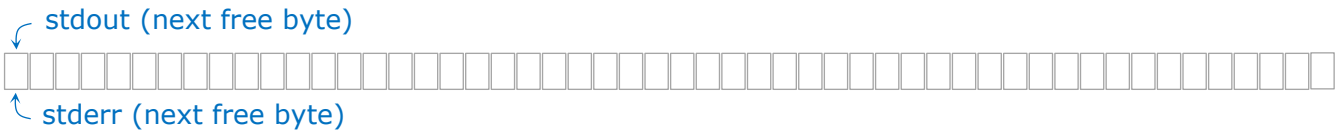
```
/home/cis90/simben $ bc > calculations 2> calculations
2+2
7/0
3+3
quit

/home/cis90/simben $ cat calculations
Ru6
ime error (func=(main), adr=5): Divide by zero
```

*Oops! Its not a good idea to redirect **stdout** and **sderr** to the same file because they clobber each other*

FYI only

It's descriptor clobbering time!

/home/cis90/simben $ **bc > calculations 2> calculations**



/home/cis90/simben $ **cat calculations**
Ru6
ime error (func=(main), adr=5): Divide by zero

Each file descriptor keeps its own separate index into the calculations file for where to write the next line.

It's descriptor collaboration time!

```
/home/cis90/simben $ bc > calculations 2>&1
2+2
7/0
3+3
quit

/home/cis90/simben $ cat calculations
4
Runtime error (func=(main), adr=5): Divide by zero
6
```

*This is the correct way to redirect **stdout** and **sderr** to the same file*

More on I/O

(input/output)

C program example

FYI only

C Program I/O example
View the program

```
/home/cis90/simben/bin $ cat simple.c
char question[] = "What is your name stranger? ";
char greeting[] = "Well I'm very pleased to meet you, ";
char buffer[80];
main()
{
        int len;

        write(2, question, sizeof(question));
        len = read(0, buffer, 80);
        write(1, greeting, sizeof(greeting));
        write(1, buffer, len);
}
```

*Write question to **stderr***

*Read name from **stdin***

*Write greeting to **stdout***

*Write name to **stdout***

This simple program asks for a name, then responds with a greeting using the name

C Program I/O example
Compile the program

The make command is used to compile a C source text file into a binary executable

```
/home/cis90/simben/bin $ make simple
cc      simple.c    -o simple
```

Unlike a bash script, the C program source code must be compiled into a binary executable before it can be run

C Program I/O example
Execute the program

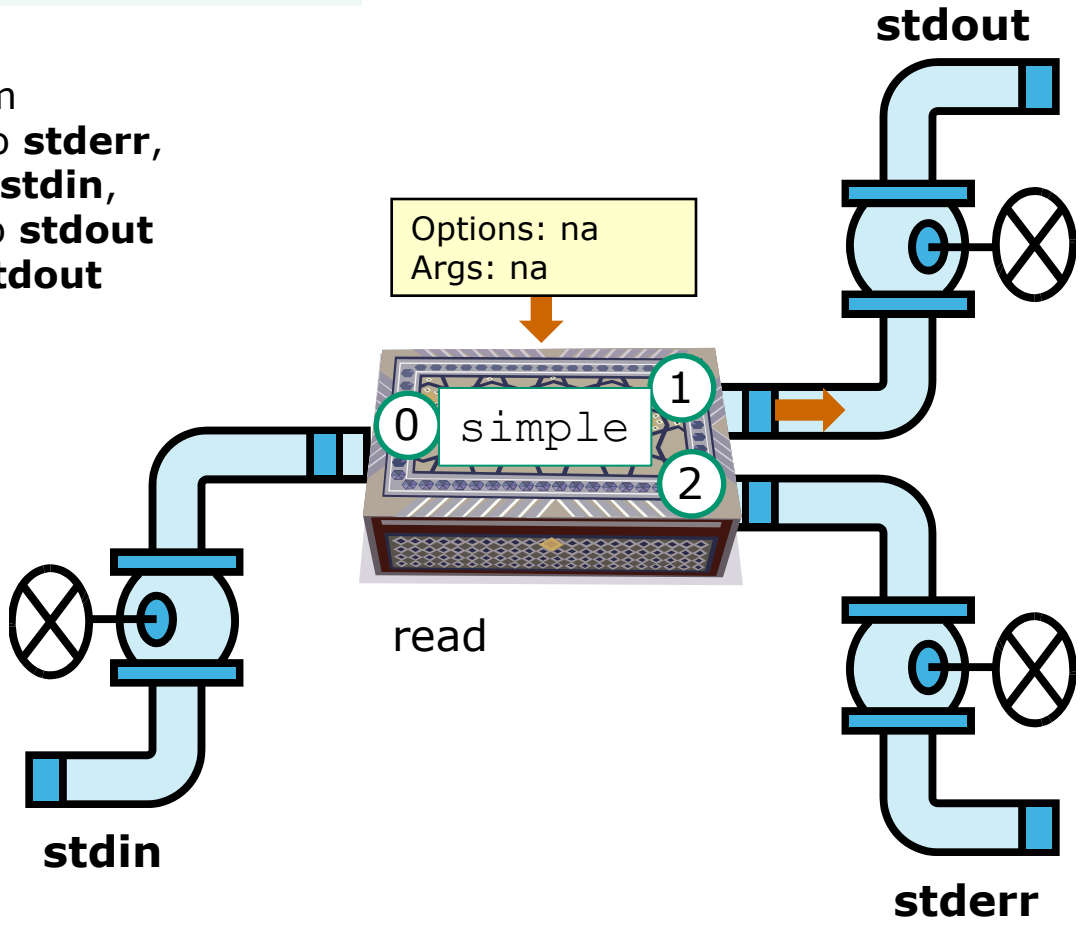
```
/home/cis90/simben/bin $ simple
What is your name stranger? Rich
Well I'm very pleased to meet you, Rich
```

Running the simple program.

C Program I/O example

```
$ simple
```

The **simple** program
1. writes question to **stderr**,
2. reads input from **stdin**,
3. writes greeting to **stdout**
4. writes name to **stdout**



3

```
Well I'm very
pleased to meet
you, Rich
```

4

2

```
Rich
```

1

```
What is your name
stranger?
```

C Program I/O example

In this example, output has been redirected to a file named myfile.

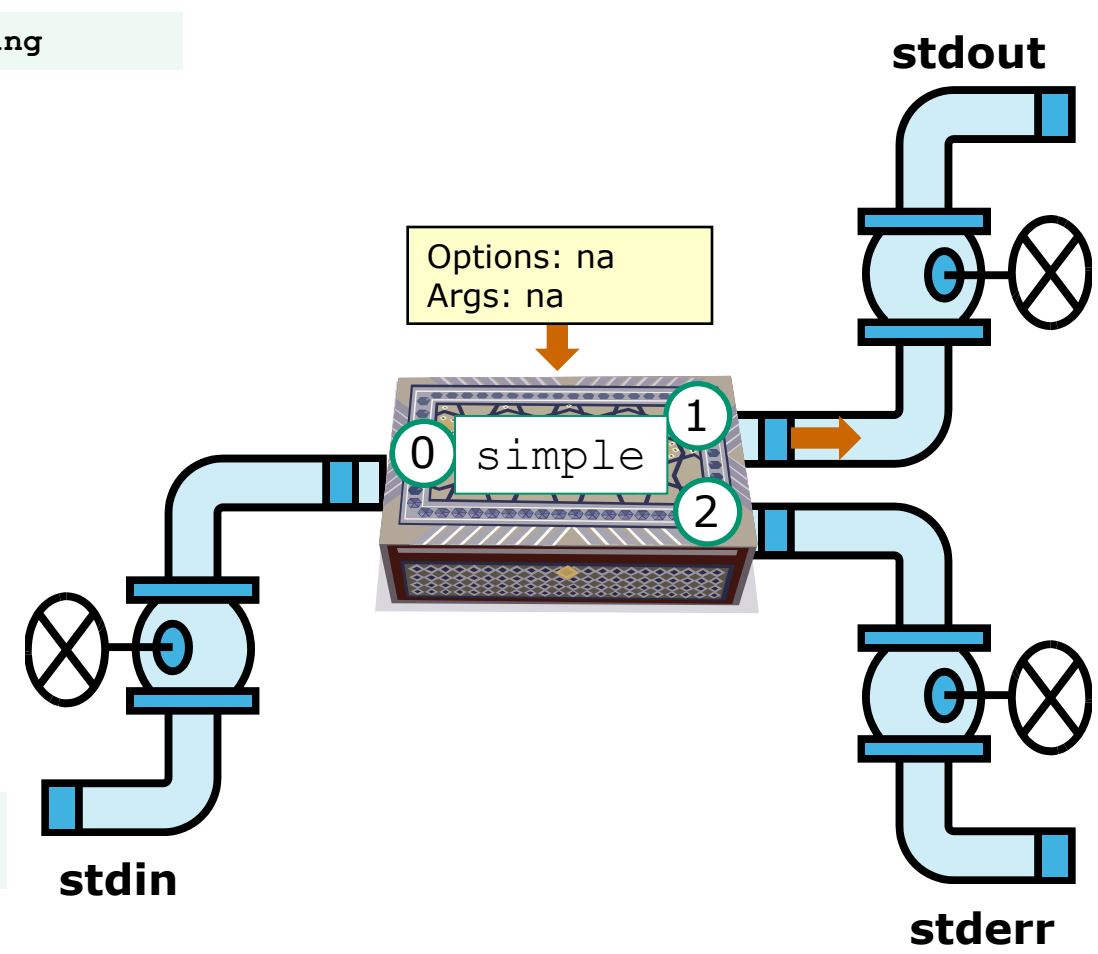
```
/home/cis90/simben/bin $ simple > myfile
What is your name stranger? Rich

/home/cis90/simben/bin $ cat myfile
Well I'm very pleased to meet you, Rich
```

*The simple program has no special knowledge (coding instructions) for a file named myfile. It just writes to **stdout** and that output will go to wherever **stdout** had been directed.*

C Program I/O example

redirection

```
$ simple > greeting
```



Activity

1. Change to your bin directory
   cd bin

2. Copy the simple.c source code from the depot directory
   cp ~/../depot/simple.c .

3. Look at your program
   cat simple.c

4. Compile the program
   make simple

5. Run the program
   simple

FYI only

C++ Program I/O example
View the program

```
/home/cis90/simben/bin $ cat simpleplus.cpp
#include <iostream>
using namespace std;

int main() {
    string question = "What is your name stranger? ";
    cerr << question;

    string buffer;
    cin >> buffer;

    string greeting = "Well I'm very pleased to meet you, ";
    cout << greeting << buffer << endl;

    return 0;
}
```

*Write question to **stderr***

*Read name from **stdin***

*Write greeting and name to **stdout***

This program is available in the depot directory

C++ Program I/O example
Compile the program

The make command is used to compile a C++ source text file into a binary executable

```
/home/cis90/simben/bin $ make simpleplus
g++      simpleplus.cpp   -o simpleplus
```

Unlike a bash script, the C++ program source code must be compiled into a binary executable before it can be run

C++ Program I/O example
Execute the program

```
/home/cis90/simben/bin $ simpleplus
What is your name stranger? Rich
Well I'm very pleased to meet you, Rich
```

Running the simpleplus program

Activity

1. Change to your bin directory
   cd bin

2. Copy the simple.c source code from the depot directory
   cp ~/../depot/simpleplus.cpp .

3. Look at your program
   cat simpleplus.cpp

4. Compile the program
   make simpleplus

5. Run the program
   simpleplus

umask

(review)

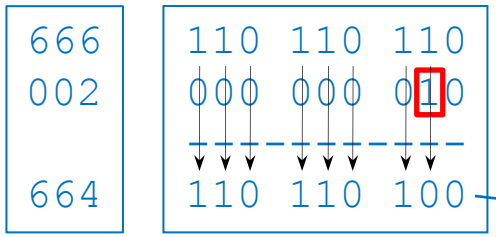
Review - applying umask bits

Example umask setting

```
/home/cis90/simben/lesson9 $ umask
0002
```

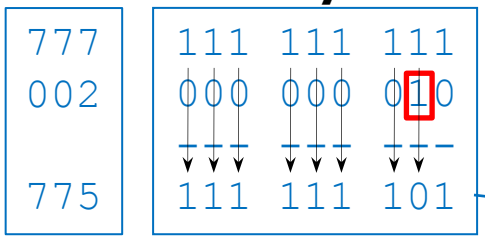
this mask indicates which permissions should NOT be set on new files or directories

New file - start with 666 and apply mask




```
/home/cis90/simben/lesson9 $ touch newfile
/home/cis90/simben/lesson9 $ ls -l newfile
-rw-rw-r-- 1 simben cis90 0 Oct 27 07:22 newfile
```

New directory - start with 777 and apply mask



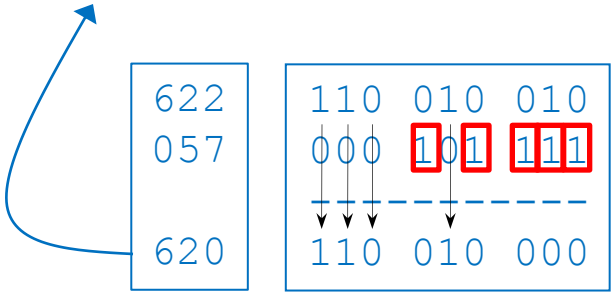

```
/home/cis90/simben/lesson9 $ mkdir newdir
/home/cis90/simben/lesson9 $ ls -ld newdir
drwxrwxr-x 2 simben cis90 4096 Oct 27 07:23 newdir
```

Any umask bits set to 1 removes the corresponding permission bit for future new files and directories

Review - Copying files

```
/home/cis90/simben $ umask 057
/home/cis90/simben $ umask
0057

/home/cis90/simben $ chmod 622 myfile
/home/cis90/simben $ cp myfile myfile.bak
/home/cis90/simben $ ls -l myfile*
-rw--w--w-. 1 simben90 cis90 0 Mar 24 17:50 myfile
-rw--w----. 1 simben90 cis90 0 Mar 24 17:51 myfile.bak
```

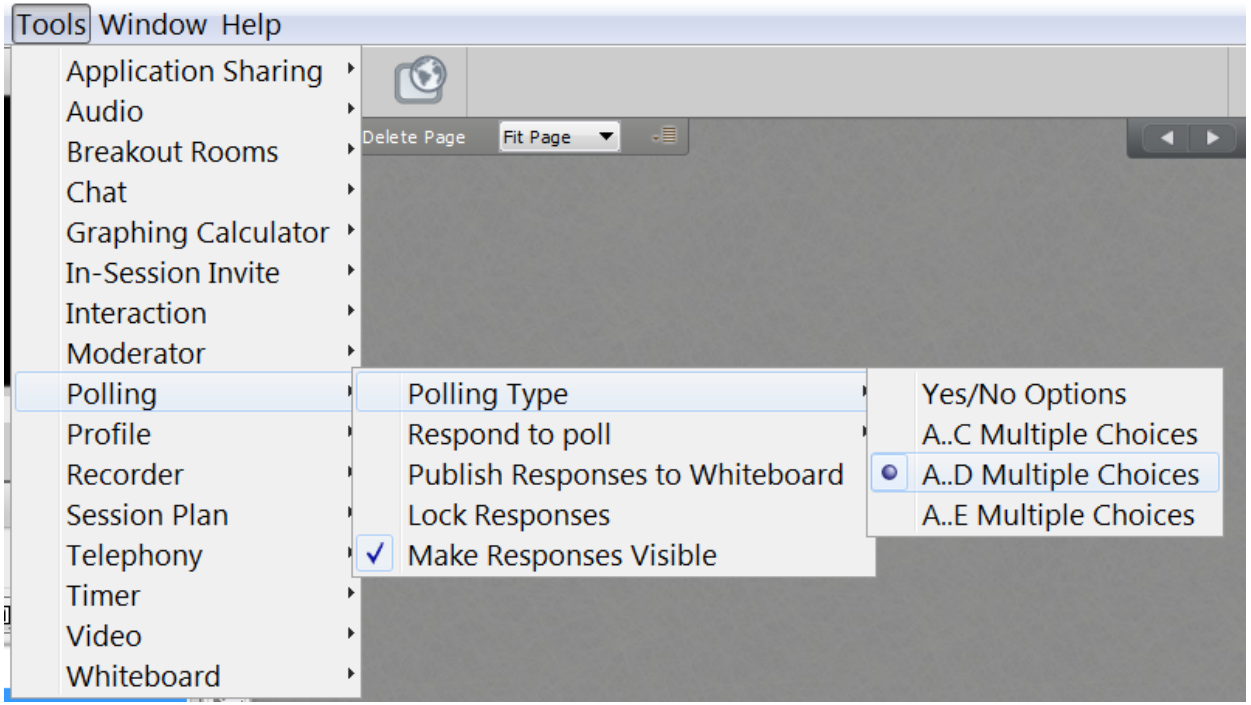
Example umask setting



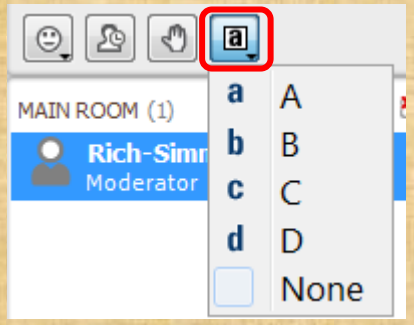
Copied file - start with original file's permissions and apply the mask

*Remember, for new files resulting from copying, instead of using the **default permissions** (666 for file and 777 for directory), use the **original file permissions** as the starting point for the mask to be applied to.*

Rich's CCC Confer poll setup



Activity



Which pizza is the best?

A. Round Table
B. Pizza My Heart
C. Tony & Alba's
D. Upper Crust

Respond to the poll above

Activity

I want to change the permissions on an existing file

Which command does this?

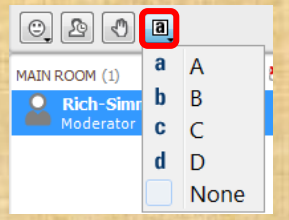
A) stat

B) ls -l

C) chmod

D) umask

Respond to the poll above



Activity

I want to restrict specific permissions on files that have not been created yet

Which command does this?

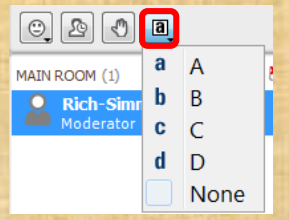
A) stat

B) ls -l

C) chmod

D) umask

Respond to the poll above



Activity

I want to show the owner of a file and its permissions in mnemonic format e.g. rwxr-xr-x

Which command does this?

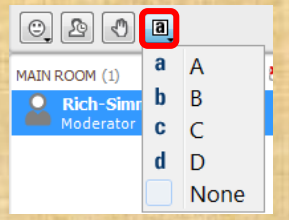
A) stat

B) ls -l

C) chmod

D) umask

Respond to the poll above



Activity

I want to show the permissions on a file in numeric format e.g. 750

Which command does this?

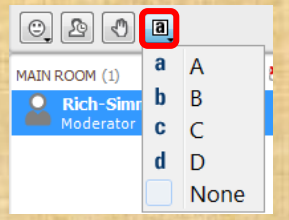
A) stat

B) ls -l

C) chmod

D) umask

Respond to the poll above



More Pipeline Practice

Pipelines

Task
Record the last times Homer Miller logged in on a Monday to a file named *mylog* AND count them

grep Homer /etc/passwd
milhom90:x:1202:190:Homer Miller:/home/cis90/milhom:/bin/bash

last
last | grep milhom90
last | grep milhom90| grep "Mon"
last | grep milhom90| grep "Mon" | tee mylog
cat mylog
last | grep milhom90| grep "Mon" | tee mylog | wc -l
cat mylog

Class Exercise
Pipeline Tasks

Task
Count the last times Rich Simms was logged in on a Tuesday and record them in a file named mylog

grep "?????" /etc/passwd

last | grep ??????
last | grep ?????? | grep "Tue"
last | grep ?????? | grep "Tue" | ??? mylog
cat mylog

last | grep ?????? | grep "Tue" | ??? mylog | wc -?
cat mylog

Put your answer in the chat window.

last | grep rsimms | grep "Tue" | tee mylog | wc -l

Pipelines

Task
Print your last name as shown in /etc/passwd:

cat /etc/passwd
cat /etc/passwd | grep $LOGNAME
cat /etc/passwd | grep $LOGNAME | cut -f 5 -d ":"
cat /etc/passwd | grep $LOGNAME | cut -f 5 -d ":" | cut -f2 -d" "

Class Exercise
Pipeline Tasks

Task
What is the first name of the user milhom90?

cat /etc/passwd
cat /etc/passwd | grep ????????
cat /etc/passwd | grep ???????? | cut -f 5 -d ":"
cat /etc/passwd | grep ???????? | cut -f 5 -d ":" | cut -f? -d" "

Put your answer in the chat window.

Pipelines

Task
Print a sorted list of the first names for CIS 172 students

```
cat /etc/passwd
cat /etc/passwd | grep cis172
cat /etc/passwd | grep cis172 | cut -f 5 -d ":"
cat /etc/passwd | grep cis172 | cut -f 5 -d ":" | cut -f1 -d" "
cat /etc/passwd | grep cis172 | cut -f 5 -d ":" | cut -f1 -d" " | sort
```

Class Exercise
Pipeline Tasks

Task
Print a sorted list of the first names for CIS 90 students

cat /etc/??????
cat /etc/?????? | grep ?????
cat /etc/?????? | grep ????? | cut -f ? -d "?"
cat /etc/?????? | grep ????? | cut -f ? -d "?" | cut -f? -d"?" | ????

Put your list in the chat window.

Pipeline and Redirection Practice

bc command with no redirection or piping

```
/home/cis90/simben $ bc
bc 1.06.95
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006 Free
Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
2+2
4
4/0
Runtime error (func=(main), adr=5): Divide by zero
quit
/home/cis90/simben $
```

Piping output to bc command

```
/home/cis90/simben $ echo 2+2 | bc
4
/home/cis90/simben $ echo 4/0 | bc
Runtime error (func=(main), adr=5): Divide by zero
```

Redirecting stdin of bc command

Setup:

```
/home/cis90/simben $ echo 2+2 > datafile
/home/cis90/simben $ echo 4/0 >> datafile
/home/cis90/simben $ cat datafile
2+2
4/0
```

Example:

```
/home/cis90/simben $ bc < datafile
4
Runtime error (func=(main), adr=5): Divide by zero
```

Piping output to bc copmmand

Setup:

```
/home/cis90/simben $ echo 2+2 > datafile
/home/cis90/simben $ echo 4/0 >> datafile
/home/cis90/simben $ cat datafile
2+2
4/0
```

Example:

```
/home/cis90/simben $ cat datafile | bc
4
Runtime error (func=(main), adr=5): Divide by zero
```

Redirecting stdin, stdout and stderr of bc command

Setup:

```
/home/cis90/simben $ echo 2+2 > datafile
/home/cis90/simben $ echo 4/0 >> datafile
/home/cis90/simben $ cat datafile
2+2
4/0
```

Example:

```
/home/cis90/simben $ bc < datafile > results 2> errors
/home/cis90/simben $ cat results
4
/home/cis90/simben $ cat errors
Runtime error (func=(main), adr=5): Divide by zero
```

Piping stdout and redirecting stdin, stderr of bc command

Setup:

```
/home/cis90/simben $ echo 2+2 > datafile
/home/cis90/simben $ echo 4/0 >> datafile
/home/cis90/simben $ cat datafile
2+2
4/0
```

Example:

```
/home/cis90/simben $ bc < datafile 2> errors | mail -s "Example" simben90
/home/cis90/simben $ cat errors
Runtime error (func=(main), adr=5): Divide by zero
```

Activity

Setup:

```
/home/cis90/simben $ echo 2+2 > datafile
/home/cis90/simben $ echo 4/0 >> datafile
/home/cis90/simben $ cat datafile
2+2
4/0
```

Example:

```
/home/cis90/simben $ bc < datafile 2> errors | mail -s "Example" $LOGNAME
/home/cis90/simben $ cat errors
Runtime error (func=(main), adr=5): Divide by zero
```

Past the email you receive into the chat window

More on pipelines

Not all commands are filters
(filters read from stdin and write to stdout)

*The **wc** command is a filter.*

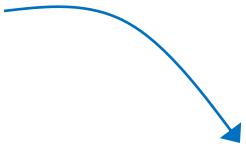
```
/home/cis90/simben $ head -n2 poems/Anon/nursery
Jack and Jill went up the hill
to fetch a pail of water.
/home/cis90/simben $ head -n2 poems/Anon/nursery | wc -l
2
/home/cis90/simben $
```

*But the **echo** command isn't (doesn't read from **stdin**)*

```
/home/cis90/simben $ head -n2 poems/Anon/nursery | echo

Oops …. this doesn't work!
/home/cis90/simben $
```

xargs command

xargs *to the rescue!*



```
/home/cis90/simben $ head -n2 poems/Anon/nursery | xargs echo
Jack and Jill went up the hill to fetch a pail of water.
```

*The **xargs** command will read **stdin** and call another command using the input as the arguments.*

Another example

Why can't Benji make a banner using the output of the date command?

```
/home/cis90/simben $ date | banner
Enter a string of up to 10 characters.
/home/cis90/simben $
```

huh? Oh, this is what banner prints when it receives no arguments on the command line

Because banner is not a filter and does not read from stdin!

Another example

```
/home/cis90/simben $ date | xargs banner
#     # ####### #     #
##   ## #     # ##    #
# # # # #     # # #   #
#  #  # #     # #  #  #
#     # #     # #   # #
#     # #     # #    ##
#     # ####### #     #

#######  #####  #######
#     # #     #    #
#     # #          #
#     # #          #
#     # #          #
#     # #     #    #
#######  #####     #

 #####   #####
#     # #     #
      #       #
 #####   #####
#       #
#       #
####### #######

   #       #             #####   #####         #######  #####
  ##      ##      ###   #     # #     #   ###   #       #     #
 # #     # #      ###         # #         ###   #       #     #
   #       #             #####  ######         ######   #####
   #       #      ###   #       #     #   ###         # #     #
   #       #      ###   #       #     #   ###   #     # #     #
 #####   #####          #######  #####          #####   #####

######  ######  #######
#     # #     #    #
#     # #     #    #
######  #     #    #
#       #     #    #
#       #     #    #
#       ######     #

 #####    ###      #     #####
#     #  #   #    ##    #     #
      # #     #  # #          #
 #####  #     #    #     #####
#       #     #    #    #
#        #   #     #    #
#######   ###    #####  #######
```

xargs *to the rescue again!*

Not all commands are filters
(filters read from stdin and write to stdout)

*The **ls** command does not read from **stdin** either*

```
/home/cis90/simben $ find poems -type d
poems
poems/Shakespeare
poems/Yeats
poems/Anon
poems/Blake


/home/cis90/simben $ find poems -type d | ls -ld
drwxr-xr-x. 18 simben90 cis90 4096 Oct 22 09:49 .
/home/cis90/simben $
```

Benji was hoping that he could get a long listing of his poems directory and all its sub-directories. Instead he gets a long listing of his home directory!

Not all commands are filters
(filters read from stdin and write to stdout)


```
/home/cis90/simben $ find poems -type d | xargs ls -ld
drwxr-xr-x. 6 simben90 cis90 4096 Oct 20 15:06 poems
drwxr-xr-x. 2 simben90 cis90 4096 Oct  5 10:26 poems/Anon
drwxr-xr-x. 2 simben90 cis90 4096 Oct 20 15:06 poems/Blake
drwxr-xr-x. 2 simben90 cis90 4096 Oct 20 15:06 poems/Shakespeare
drwxr-xr-x. 2 simben90 cis90 4096 Oct 20 15:06 poems/Yeats
/home/cis90/simben $
```

*The **ls** command is not a filter so it does not read from **stdin***

***xargs** reads the names of the files found by the **find** command and uses them as arguments on the **ls -ld** command*

Not all commands are filters
(filters read from stdin and write to stdout)

```
/home/cis90/simben $ find poems -type d -exec ls -ld {} \;
drwxr-xr-x. 6 simben90 cis90 4096 Oct 20 15:06 poems
drwxr-xr-x. 2 simben90 cis90 4096 Oct 20 15:06 poems/Shakespeare
drwxr-xr-x. 2 simben90 cis90 4096 Oct 20 15:06 poems/Yeats
drwxr-xr-x. 2 simben90 cis90 4096 Oct  5 10:26 poems/Anon
drwxr-xr-x. 2 simben90 cis90 4096 Oct 20 15:06 poems/Blake
/home/cis90/simben $
```

*The **find** command also has a **-exec** option that will run a command on what is found. The **{}** represent the arguments which are names of files found by the **find** command.*

Things that Hide

Finding Things

Task

Find all files in the */usr/src* branch of the file tree that contain "Torvalds"

```
grep -r "Torvalds" /usr/src
```

```
/usr/src/kernels/2.6.32-220.23.1.el6.i686/arch/powerpc/Makefile:# Copyright (C) 1994 by Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/arch/s390/Makefile:# Copyright (C) 1994 by Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/arch/parisc/Makefile:# Copyright (C) 1994 by Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/arch/alpha/Makefile:# Copyright (C) 1994 by Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/arch/alpha/boot/Makefile:# Copyright (C) 1994 by Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/arch/x86/include/asm/i387.h: * Copyright (C) 1994 Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/arch/x86/include/asm/irq.h: * (C) 1992, 1993 Linus Torvalds, (C) 1997 Ingo Molnar
/usr/src/kernels/2.6.32-220.23.1.el6.i686/arch/x86/include/asm/thread_info.h: * - Incorporating suggestions made by Linus Torvalds and Dave Miller
/usr/src/kernels/2.6.32-220.23.1.el6.i686/arch/x86/include/asm/bitops.h: * Copyright 1992, Linus Torvalds.
/usr/src/kernels/2.6.32-220.23.1.el6.i686/arch/x86/include/asm/stacktrace.h: *  Copyright (C) 1991, 1992  Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/arch/x86/include/asm/hw_irq.h: * (C) 1992, 1993 Linus Torvalds, (C) 1997 Ingo Molnar
/usr/src/kernels/2.6.32-220.23.1.el6.i686/arch/x86/include/asm/delay.h: * Copyright (C) 1993 Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/arch/x86/include/asm/sync_bitops.h: * Copyright 1992, Linus Torvalds.
/usr/src/kernels/2.6.32-220.23.1.el6.i686/arch/x86/boot/Makefile:# Copyright (C) 1994 by Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/include/linux/ioport.h: * Authors:    Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/include/linux/pagemap.h: * Copyright 1995 Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/include/linux/thread_info.h: * - Incorporating suggestions made by Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/include/linux/ext2_fs.h: *  Copyright (C) 1991, 1992  Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/include/linux/nilfs2_fs.h: *  Copyright (C) 1991, 1992  Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/include/linux/dcache.h: * with heavy changes by Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/include/linux/completion.h: * (C) Copyright 2001 Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/include/linux/ext3_fs.h: *  Copyright (C) 1991, 1992  Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/include/linux/ide.h: *  Copyright (C) 1994-2002  Linus Torvalds & authors
/usr/src/kernels/2.6.32-220.23.1.el6.i686/include/linux/delay.h: * Copyright (C) 1993 Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/include/linux/ext3_fs_sb.h: *  Copyright (C) 1991, 1992  Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/include/linux/ext3_fs_i.h: *  Copyright (C) 1991, 1992  Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/include/linux/ext2_fs_sb.h: *  Copyright (C) 1991, 1992  Linus Torvalds
/usr/src/kernels/2.6.32-220.23.1.el6.i686/include/asm-generic/tlb.h: * Based on code from mm/memory.c Copyright Linus Torvalds and others.
/usr/src/kernels/2.6.32-220.23.1.el6.i686/scripts/package/builddeb:Copyright: 1991 - 2009 Linus Torvalds and others.
/usr/src/kernels/2.6.32-220.23.1.el6.i686/scripts/get_maintainer.pl:push(@penguin_chief,"Linus Torvalds:torvalds\@linux-foundation.org");
/usr/src/kernels/2.6.32-220.23.1.el6.i686/scripts/checkstack.pl:#    Inspired by Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/arch/powerpc/Makefile:# Copyright (C) 1994 by Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/arch/s390/Makefile:# Copyright (C) 1994 by Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/arch/parisc/Makefile:# Copyright (C) 1994 by Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/arch/alpha/Makefile:# Copyright (C) 1994 by Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/arch/alpha/boot/Makefile:# Copyright (C) 1994 by Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/arch/x86/include/asm/i387.h: * Copyright (C) 1994 Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/arch/x86/include/asm/irq.h: *       (C) 1992, 1993 Linus Torvalds, (C) 1997 Ingo Molnar
/usr/src/kernels/2.6.32-71.el6.i686/arch/x86/include/asm/thread_info.h: * - Incorporating suggestions made by Linus Torvalds and Dave Miller
/usr/src/kernels/2.6.32-71.el6.i686/arch/x86/include/asm/bitops.h: * Copyright 1992, Linus Torvalds.
/usr/src/kernels/2.6.32-71.el6.i686/arch/x86/include/asm/hw_irq.h: * (C) 1992, 1993 Linus Torvalds, (C) 1997 Ingo Molnar
/usr/src/kernels/2.6.32-71.el6.i686/arch/x86/include/asm/delay.h: * Copyright (C) 1993 Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/arch/x86/include/asm/sync_bitops.h: * Copyright 1992, Linus Torvalds.
/usr/src/kernels/2.6.32-71.el6.i686/arch/x86/boot/Makefile:# Copyright (C) 1994 by Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/include/linux/ioport.h: * Authors:  Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/include/linux/pagemap.h: * Copyright 1995 Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/include/linux/thread_info.h: * - Incorporating suggestions made by Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/include/linux/ext2_fs.h: *  Copyright (C) 1991, 1992  Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/include/linux/nilfs2_fs.h: *  Copyright (C) 1991, 1992  Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/include/linux/dcache.h: * with heavy changes by Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/include/linux/completion.h: * (C) Copyright 2001 Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/include/linux/ext3_fs.h: *  Copyright (C) 1991, 1992  Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/include/linux/ide.h: *  Copyright (C) 1994-2002  Linus Torvalds & authors
/usr/src/kernels/2.6.32-71.el6.i686/include/linux/delay.h: * Copyright (C) 1993 Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/include/linux/ext3_fs_sb.h: *  Copyright (C) 1991, 1992  Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/include/linux/ext3_fs_i.h: *  Copyright (C) 1991, 1992  Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/include/linux/ext2_fs_sb.h: *  Copyright (C) 1991, 1992  Linus Torvalds
/usr/src/kernels/2.6.32-71.el6.i686/include/asm-generic/tlb.h: * Based on code from mm/memory.c Copyright Linus Torvalds and others.
/usr/src/kernels/2.6.32-71.el6.i686/scripts/package/builddeb:Copyright: 1991 - 2009 Linus Torvalds and others.
/usr/src/kernels/2.6.32-71.el6.i686/scripts/get_maintainer.pl:push(@penguin_chief,"Linus Torvalds:torvalds\@linux-foundation.org");
/usr/src/kernels/2.6.32-71.el6.i686/scripts/checkstack.pl:#    Inspired by Linus Torvalds
[rsimms@oslab ~]$
```

*Do a recursive grep to search the **contents** of files in an entire branch of the file tree.*

Finding Things

Task
Count the number of files in the */usr/src* branch of the file tree that contain "Stallman"

```
grep -? "Stallman" /???/??? | wc -?
```

Write your answer in the chat window

grep -r "Stallman" /usr/src | wc -l

Finding Things

Task
Find all files in the */usr/share/doc* branch of the file tree that are named "BUGS"

```
find /usr/share/doc -name "BUGS"
```

```
/usr/share/doc/ppl-0.10.2/BUGS
/usr/share/doc/ltrace-0.5/BUGS
/usr/share/doc/perl-IO-Socket-SSL-1.31/BUGS
/usr/share/doc/glibc-2.12/BUGS
/usr/share/doc/parted-2.1/BUGS
/usr/share/doc/cvs-1.11.23/BUGS
/usr/share/doc/patchutils-0.3.1/BUGS
/usr/share/doc/procps-3.2.8/BUGS
/usr/share/doc/gettext-0.17/BUGS
/usr/share/doc/curl-7.19.7/BUGS
/usr/share/doc/sed-4.2.1/BUGS
/usr/share/doc/SDL-1.2.14/BUGS
/usr/share/doc/cairo-1.8.8/BUGS
/usr/share/doc/emacs-common-23.1/BUGS
/usr/share/doc/tcsh-6.17/BUGS
/usr/share/doc/unzip-6.0/BUGS
/usr/share/doc/vsftpd-2.2.2/BUGS
/usr/share/doc/dejavu-fonts-common-2.30/BUGS
/usr/share/doc/nano-2.0.9/BUGS
[rsimms@oslab ~]$
```

Use find to search for files by name, type, user, group, etc.

Finding Things

Task
Count all the files in the */home* branch of the file tree that are owned by rsimms. Discard any permission errors.

```
find /???? -user ?????? 2> /dev/??? | ?? -l
```

Write your answer in the chat window

/home/cis90/simben $ find /home -user rsimms 2> /dev/null | wc -l

Finding Things

Task

Find all files in the */home/cis90/bin that are regular files and belong to the staff group.*

```
find /home/cis90/bin -group staff -type f
```

```
/home/cis90/bin/enlightenment
/home/cis90/bin/allscripts
/home/cis90/bin/list
/home/cis90/bin/submit.sp15.v1
/home/cis90/bin/tinsam90/schedule.pyc
/home/cis90/bin/tinsam90/schedule.py
/home/cis90/bin/tinsam90/forums.py
/home/cis90/bin/tinsam90/tips.py
/home/cis90/bin/tinsam90/grade.py
/home/cis90/bin/submitx
/home/cis90/bin/old/submit.fa14.v5
/home/cis90/bin/old/checkgrades.py.fa14
/home/cis90/bin/old/allscripts.sp14
/home/cis90/bin/old/check10.v2
/home/cis90/bin/old/submit.fa14.v1
/home/cis90/bin/old/check10.v1
/home/cis90/bin/old/submit.fa14.v4
/home/cis90/bin/old/checkgrades.py.sp14
/home/cis90/bin/old/submit.fa14.v2
/home/cis90/bin/old/submit.fa14.v3
/home/cis90/bin/old/submit.fa14.v6
/home/cis90/simben $
```

Use find to search for files by name, type, user, group, etc.

Finding Things

Task
Count all the directories in the */home/cis90* branch of the file tree that belong to the cis90 group. Discard any permission errors.

```
???? /home/????? -type ? -group ????? ?? /dev/null | ?? -?
```

Write your answer in the chat window

find /home/cis90 -type d -group cis90 2> /dev/null | wc -l

Eggs, Treats and Tricks

Egg Hunt

Instructor: sudo /home/rsimms/cis90/basket/hide-the-eggs

A number of colored eggs have been distributed within your home directory and sub-directories!

1. Can you find them? There should be an obvious one in your home directory. Who is the owner and group for this egg file? The rest are scattered in the various subdirectories you own.

2. Make a new directory named *basket* in your home directory and see how many egg files you can move into it.

3. Put a Green Check in CCC Confer next to your name when you have collected 3 eggs, electronically "clap" if you collect all 17.

Review

Jim's Summary Pages

Jim has some really good summary information on Lessons 6-8 on his web site:

Lesson 6 - Managing Files
http://cabrillo.edu/~jgriffin/CIS90/files/lecture5.html

Lesson 7 - File Permissions
http://cabrillo.edu/~jgriffin/CIS90/files/lecture6.html

Lesson 8 - Input/Output Processing
http://cabrillo.edu/~jgriffin/CIS90/files/lecture7.html

Make Teams

Breakout Rooms

Room 1

Room 2

Room 3

Room 4

Room 5

Once you are in your rooms:

1) Write your team's distro name at the top of your room's white board
2) Everyone write their first names under the distro's team name
3) If you want to be fancy add your distro logo to the top of your room's white board!

Make Teams:
CCC Confer: Tools > Breakout Rooms > Create Breakout Rooms … (make 6 rooms)

Flashcard Practice

Flashcards

Room 1	Room 2	Room 3	Room 4	Room 5
Points:	**Points:**	**Points:**	**Points:**	**Points:**

Flashcards
L6=20
L7=15
L8=16

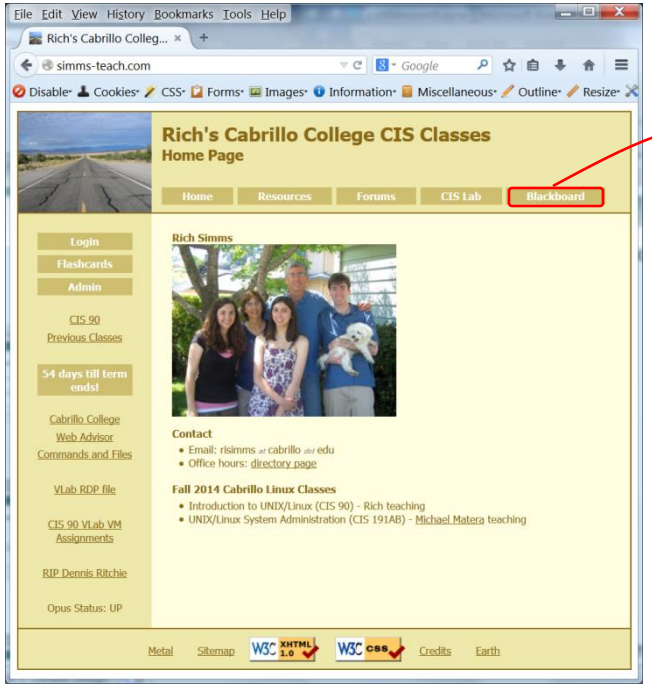
Rules

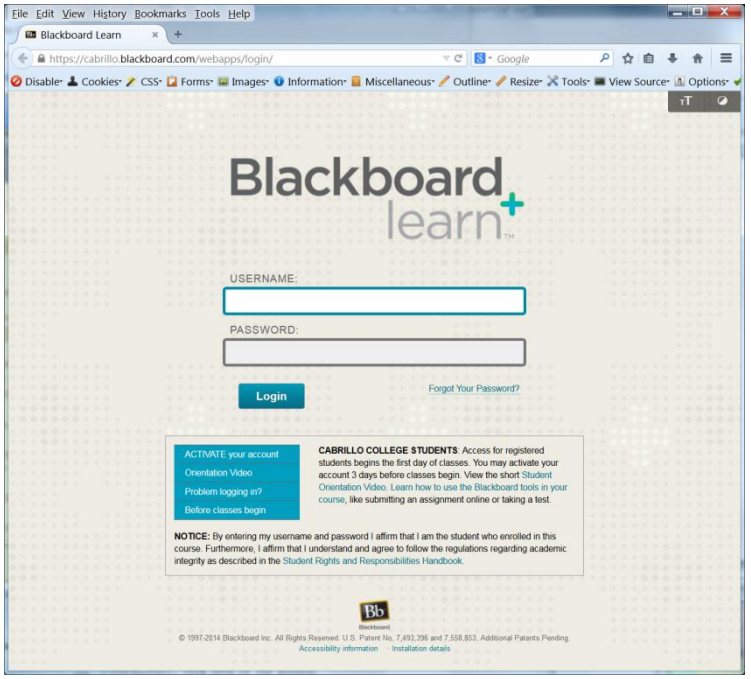
- Chat window belongs to team that is up
- Team gets the point if anyone on the team writes a correct answer in the chat window in 15 seconds

Instructor timer:
i=15; while [$i -gt 0]; do clear; banner $i; let i=i-1; sleep 1; done; clear; banner done

Practice Test

Practice Test





Practice test available

- Available on Blackboard
- Work alone or together
- Use the forum to compare answers and approaches to questions
- Test #2 will be graded by looking at both your answers to the questions and the work you did on the testing server.

Practice Test Honor Code and Instructions

Description	Practice Test 2 (Spring 2015)
Instructions	**HONOR CODE:** This is a practice test and you may work with others on it. Feel free to compare and discuss answers to the practice test on the forum. However on the real test you must work alone. **INSTRUCTIONS:** Test system: sun-hwa-p2.cis.cabrillo.edu (port 22) This test should be completed using the sun-hwa-p2 system only. Because this system is on a private network log into Opus first then ssh into sun-hwa-p2. The sun-hwa-p2 system will **not be available** after the real test starts. Grading will be based on your answers AND that you correctly implemented the "DO THIS FIRST" portion of the question. **If you get stuck on any question you may ask for help on the forum. Please use the forum to ask and answer each other's practice test questions. If you get stuck on the real test you can "purchase" consulting services from the instructor.** Please KEEP YOUR ANSWERS TO A SINGLE LINE ONLY !! This test must be completed in one sitting. The submittal will be made automatically when the time is up. If you submit early by accident you will not be able to re-enter and continue. If this happens on the real test don't panic! Just email the instructor any remaining answers before the time is up.
Total Questions	33
Total Points	30

Make sure you can log into the testing server. Login consulting is free on the practice test.

On the real test though you can "purchase" login consulting from the instructor using some of your points!

Wrap up

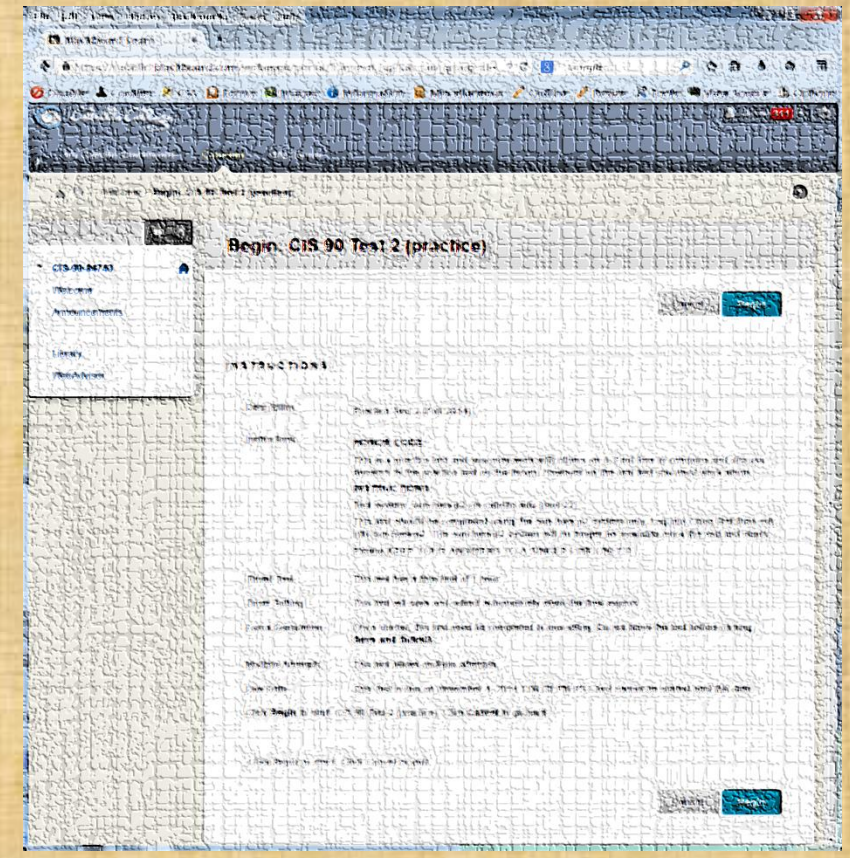
Next Class

No Quiz

Cumulative Test (30 points) with focus on Lessons 6-9:

- Recommended preparation:
  - **Work the practice test!**
  - **Work the practice test!**
  - **Work the practice test!**
  - **Make a personal reference "crib sheet" document**
  - **Collaborate with others on the forum to compare answers**
  - Review Lessons 6-9 slides and Labs 5-7
  - Try doing some or all of Lab X2 (pathnames)
  - Practice with flash cards
  - Scan previous Lessons so you know where to find things if needed

Optional Workshop Today



Work the practice test till the end of class today and into office hours after class.

- Collaborate!
- Ask questions!
- You may leave class once you know how to approach and hopefully answer each question

Backup