Rich's lesson module checklist

Last updated 10/30/2017

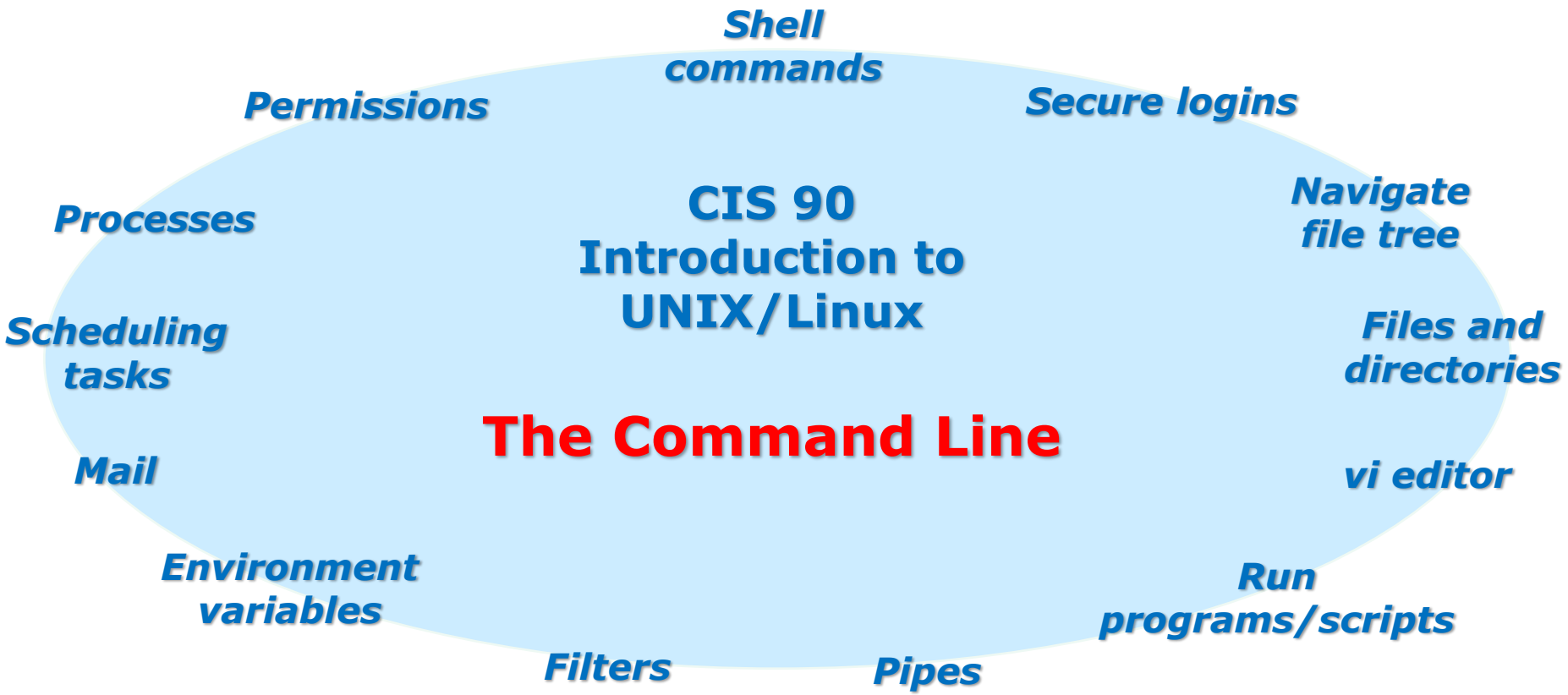
- ❑ Slides posted
- ❑ WB converted from PowerPoint
- ❑ Print out agenda slide and annotate page numbers

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

- ❑ Schedule lock of turnin directory and submit
  scripts/schedule-submit-locks
- ❑ Opus-II - hide script tested
- ❑ Update test Q21 for number of accounts
- ❑ Practice test available on Canvas at end of class
- ❑ P2 Test system online and unlocked at end of class

- ❑ 9V backup battery for microphone
- ❑ Backup slides, CCC info, handouts on flash drive
- ❑ Key card for classroom door

- ❑ Update CCC Confer and 3C Media portals



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.

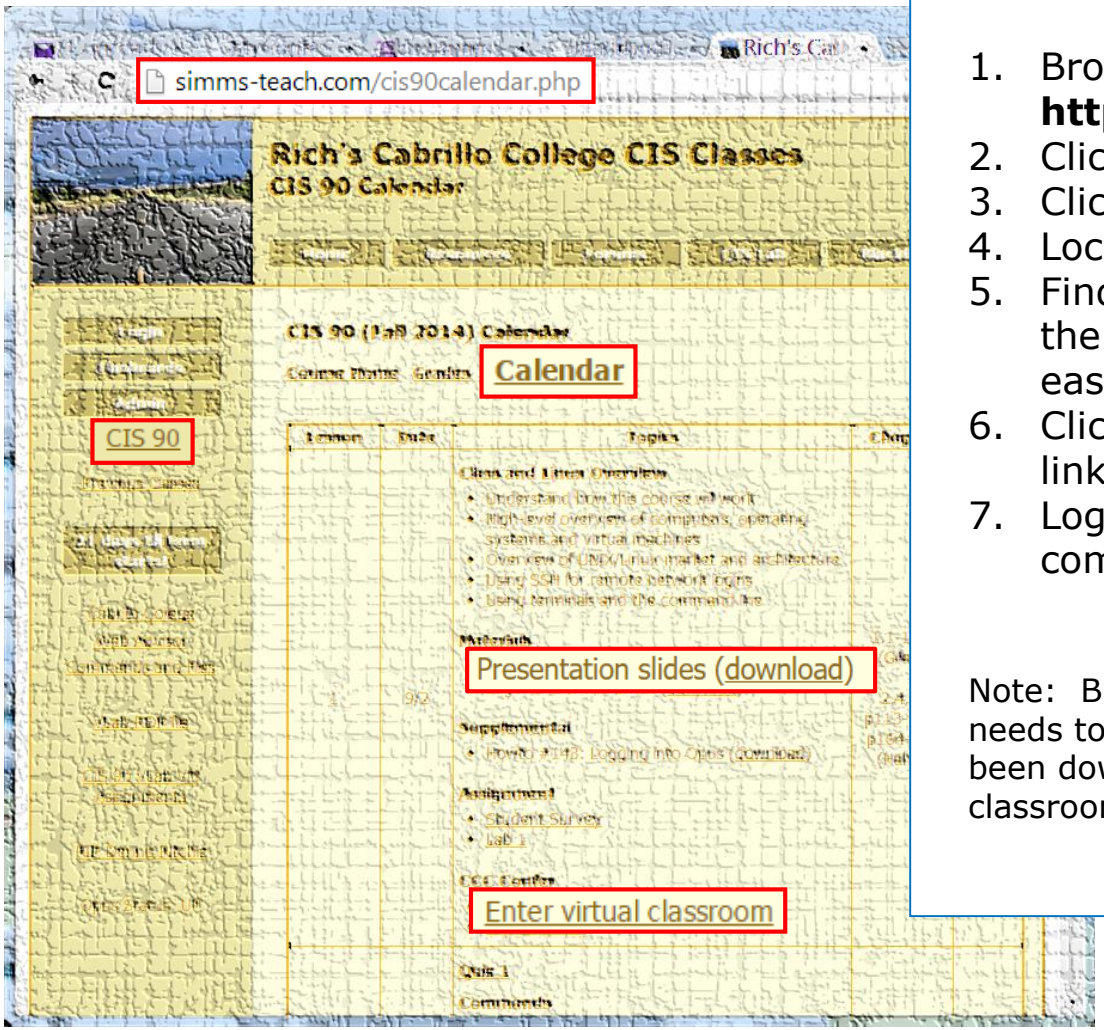
Introductions and Credits

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 for attending class

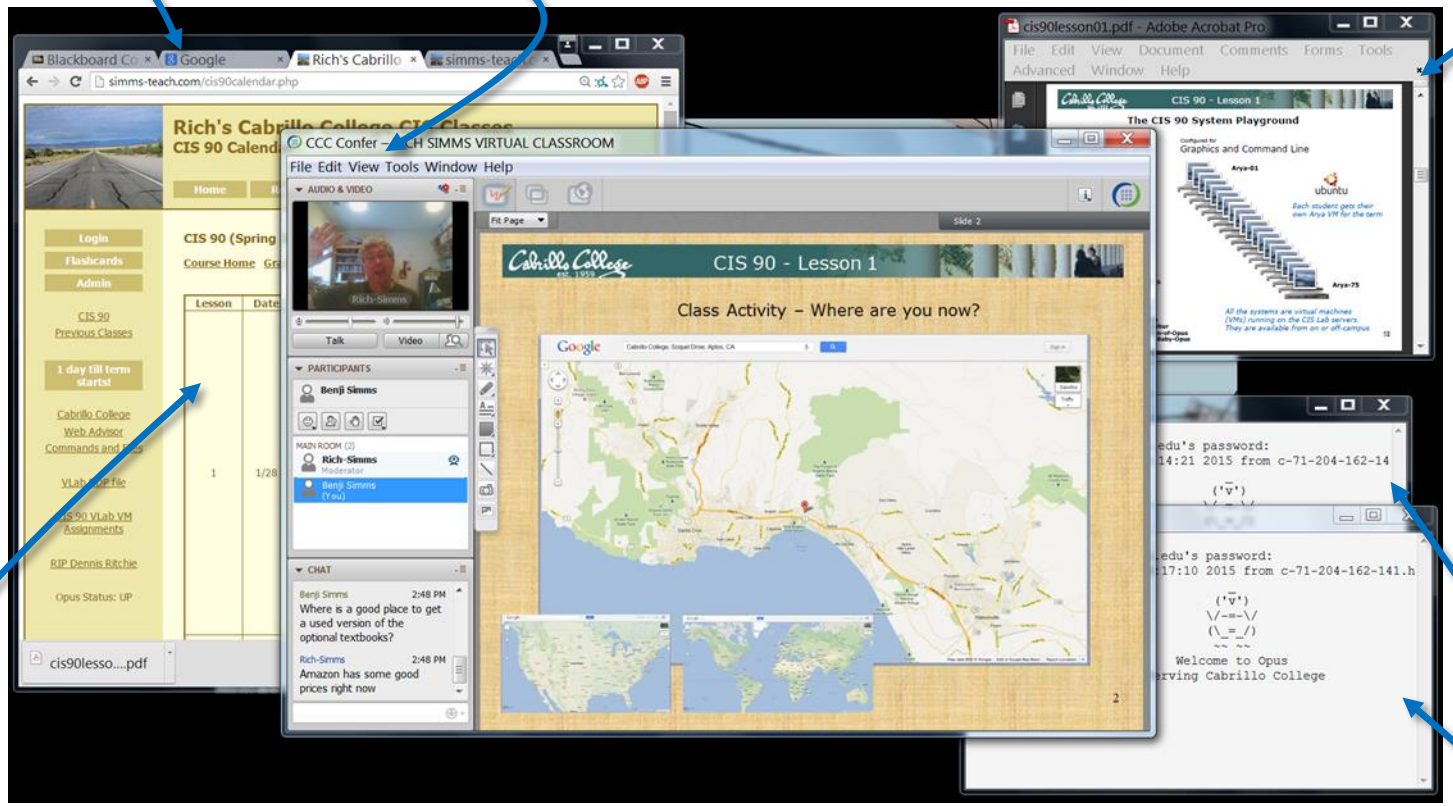


1. Browse to: **http://simms-teach.com**
2. Click the **CIS 90** link.
3. Click the **Calendar** link.
4. Locate today's lesson.
5. Find the **Presentation slides** for the lesson and **download** for easier viewing.
6. Click the **Enter virtual classroom** link to join CCC Confer.
7. Log into Opus-II with Putty or ssh command.

Note: Blackboard Collaborate Launcher only needs to be installed once. It has already been downloaded and installed on the classroom PC's.

Student checklist for suggested screen layout

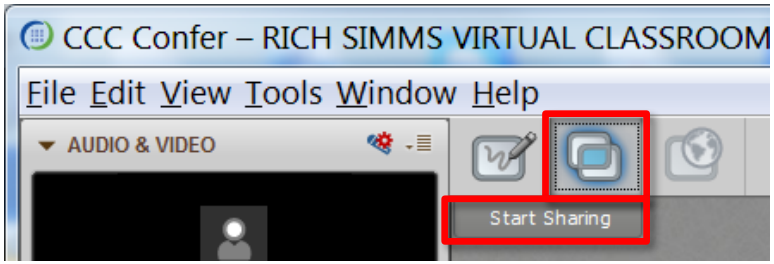
- ❑ *Google*
- ❑ *CCC Confer*
- ❑ *Downloaded PDF of Lesson Slides*



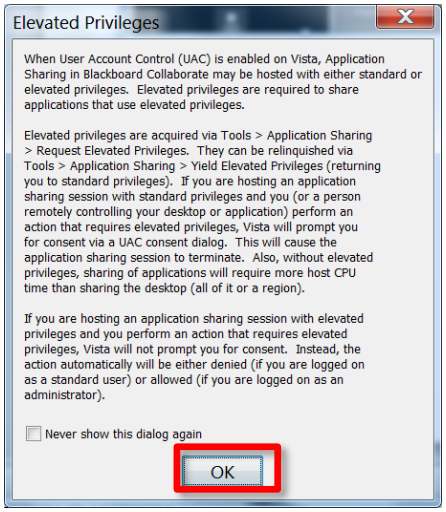
- ❑ *CIS 90 website Calendar page*
- ❑ *One or more login sessions to Opus-II*

Student checklist for sharing desktop with classmates

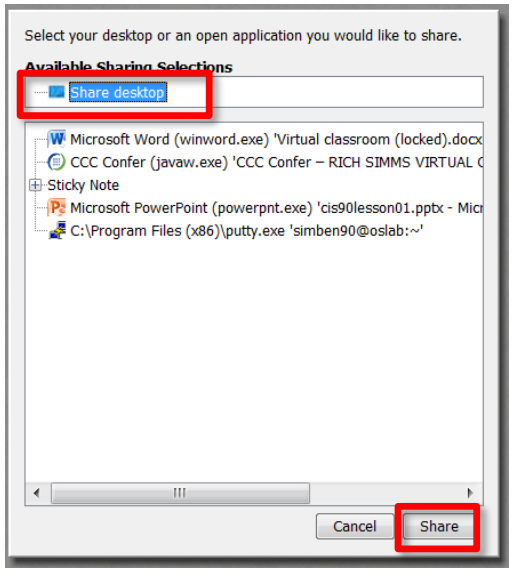
1) Instructor gives you sharing privileges



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



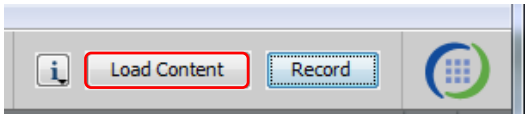
3) Click OK button.



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

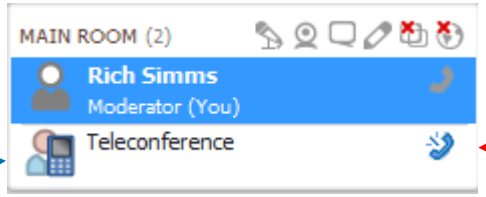
Rich's CCC Confer checklist - setup

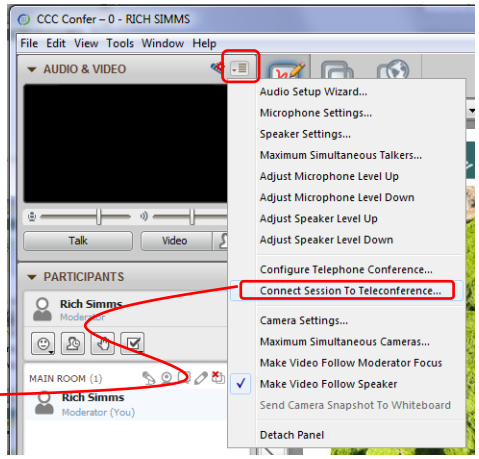
[ ] Preload White Board



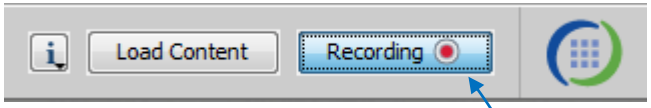
[ ] Connect session to Teleconference

Session now connected to teleconference





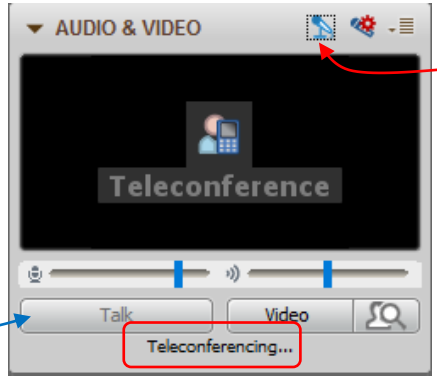
[ ] Is recording on?



Red dot means recording

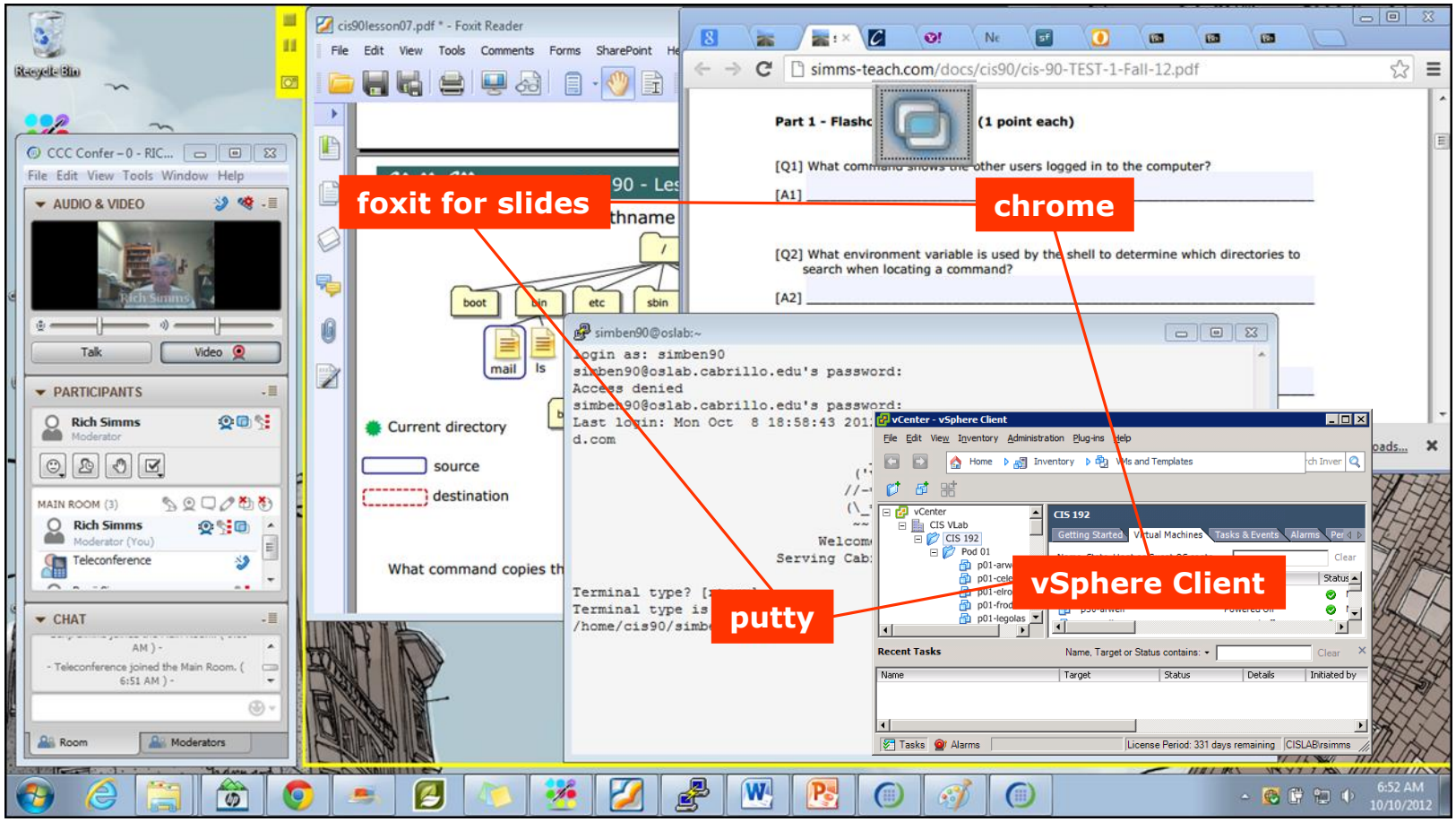
[ ] Use teleconferencing, not mic

Should be grayed out



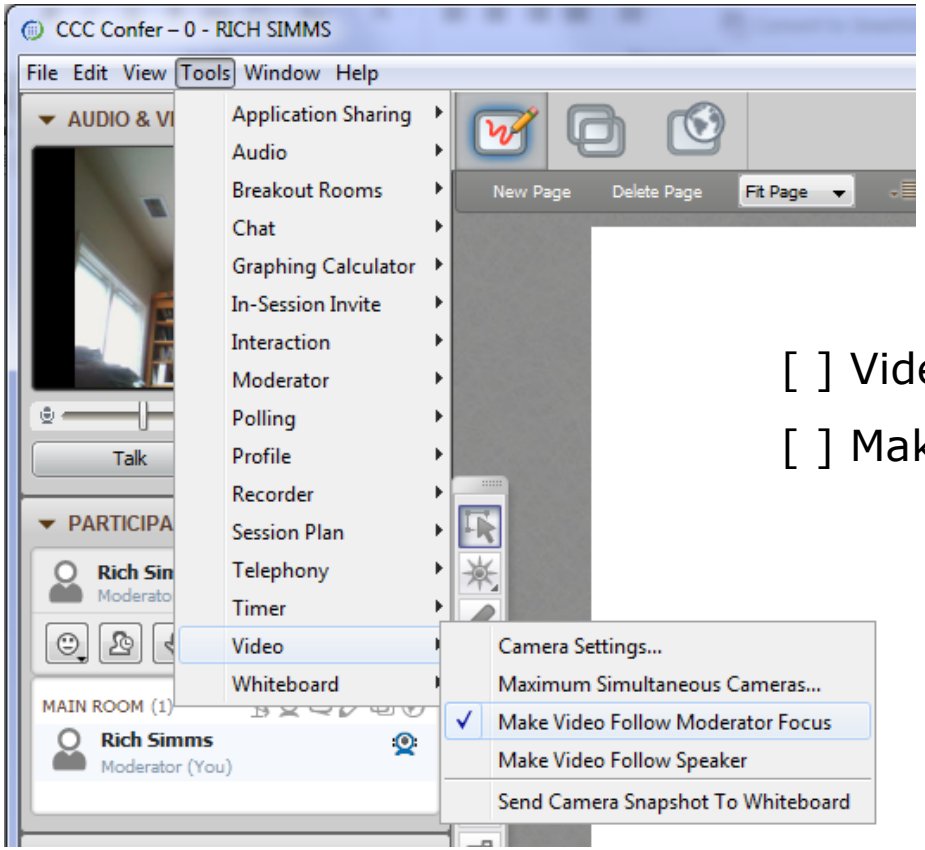
Should change from phone handset icon to little Microphone icon and the Teleconferencing … message displayed

Rich's CCC Confer checklist - screen layout



[ ] layout and share apps

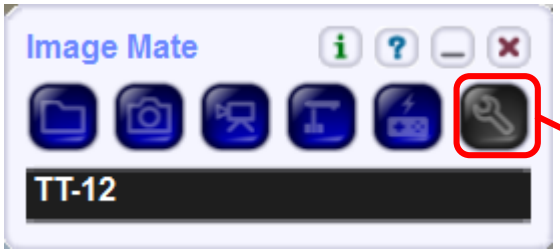
Rich's CCC Confer checklist - webcam setup

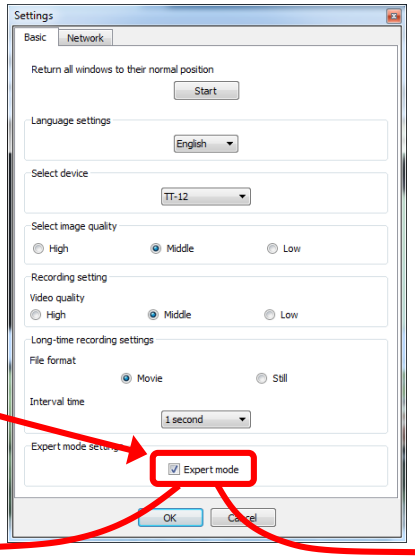


[ ] Video (webcam)

[ ] Make Video Follow Moderator Focus

Rich's CCC Confer checklist - Elmo



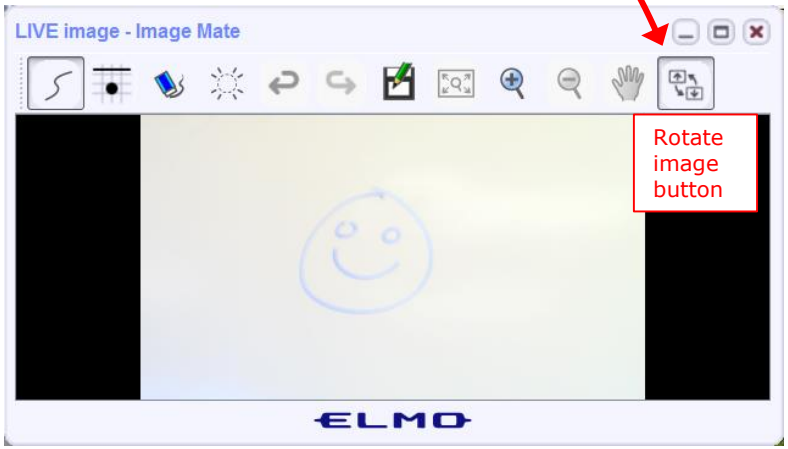


The "rotate image" button is necessary if you use both the side table and the white board.

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



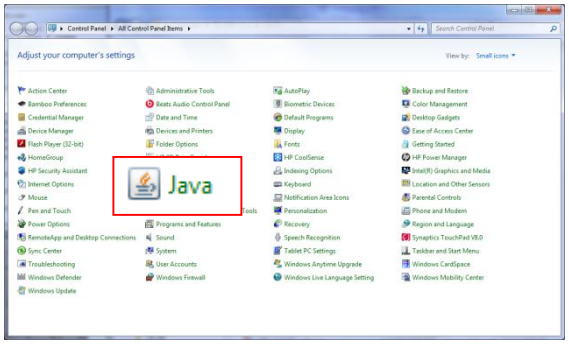
Run and share the Image Mate program just as you would any other app with CCC Confer

Rich's CCC Confer checklist - universal fixes

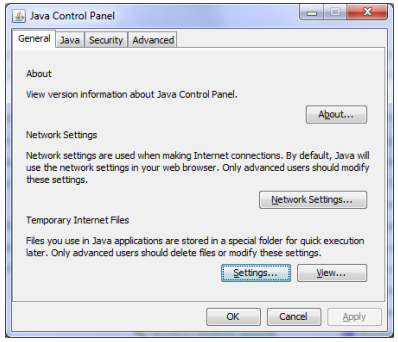
Universal Fix for CCC Confer:

1) Shrink (500 MB) and delete Java cache
2) Uninstall and reinstall latest Java runtime
3) http://www.cccconfer.org/support/technicalSupport.aspx

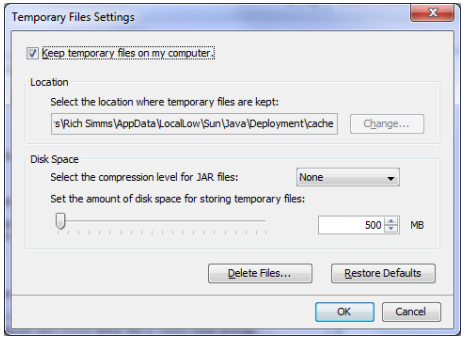
Control Panel (small icons)



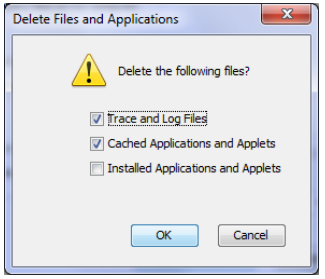
General Tab > Settings…



500MB cache size

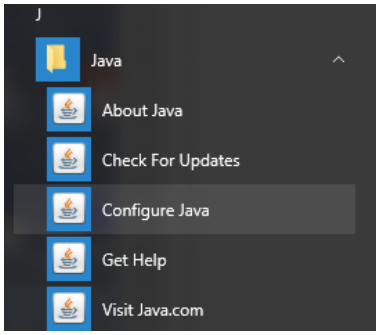


Delete these

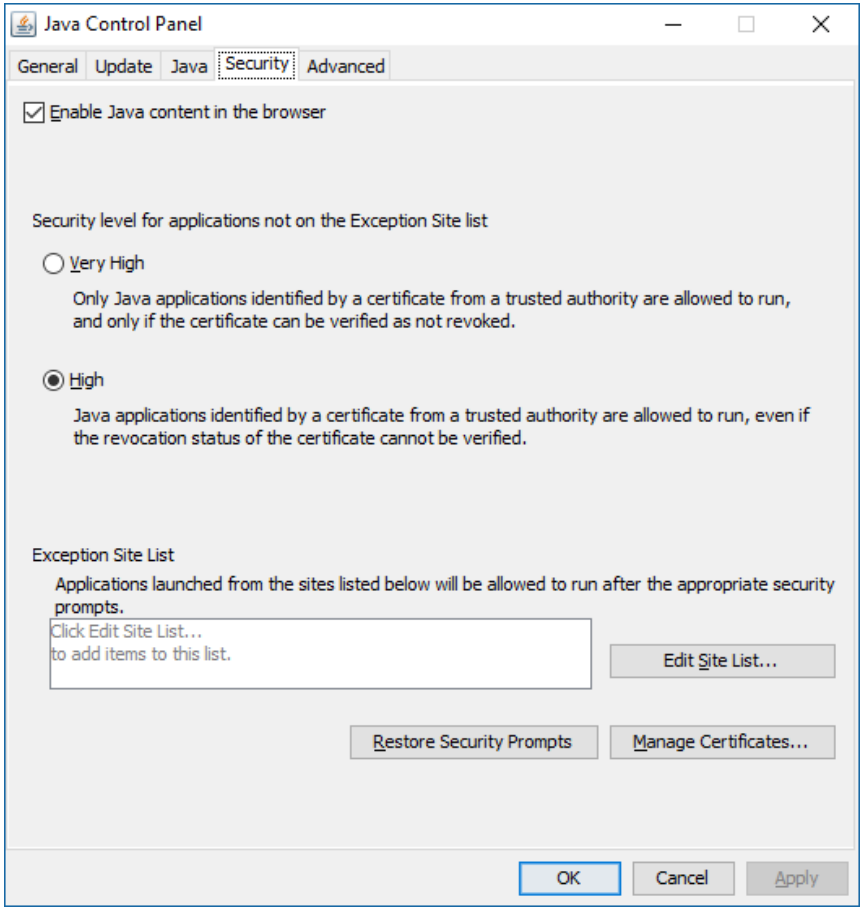


Google Java download

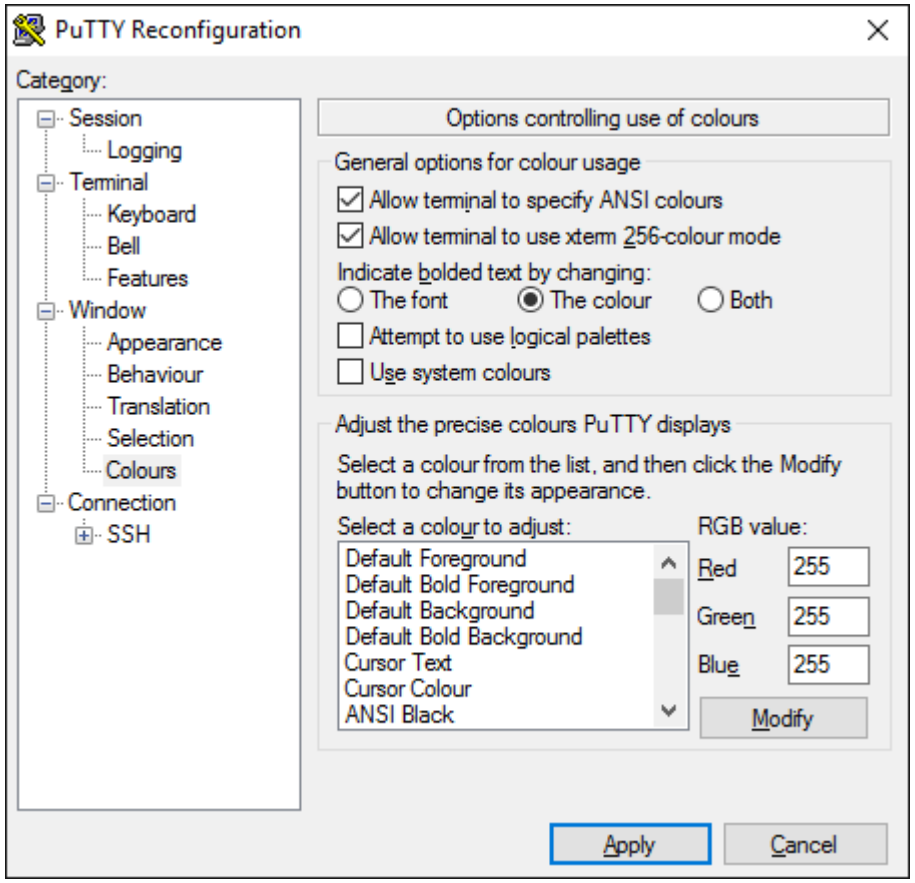
Rich's CCC Confer checklist - digital certificate work around



1. Open the Java Control Panel
2. Select the **Security** tab
3. Select **Edit Site List...**
4. Select **Add**
5. Click into the white box next to the red exclamation mark and type **https://na-downloads.elluminate.com**
6. Press OK
7. Press **Continue** on the pop-up message
8. Press OK
9. Access your session or recording once more



Rich's CCC Confer checklist - Putty Colors



http://looselytyped.blogspot.com/2013/02/zenburn-pleasant-color-scheme-for-putty.html

Putty Colors

Default Foreground 255 255 255
Default Bold Foreground 255 255 255
Default Background 51 51 51
Default Bold Background 255 2 85
Cursor Text 0 0 0
Cursor Color 0 255 0
ANSI Black 77 77 77
ANSI Black Bold 85 85 85
ANSI Red 187 0 0
ANSI Red Bold 255 85 85
ANSI Green 152 251 152
ANSI Green Bold 85 255 85
ANSI Yellow 240 230 140
ANSI Yellow Bold 255 255 85
ANSI Blue 205 133 63
ANSI Blue Bold 135 206 235
ANSI Magenta 255 222 173
ANSI Magenta Bold 255 85 255
ANSI Cyan 255 160 160
ANSI Cyan Bold 255 215 0
ANSI White 245 222 179
ANSI White Bold 255 255 255

Start

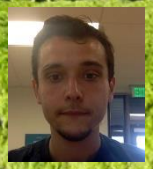
Sound Check

*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.*

Volume
**4 - increase conference volume.*
**7 - decrease conference volume.*
**5 - increase your voice volume.*
**8 - decrease your voice volume.*

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


Vinny

Marvin


William


Dan C.

Hayden


Nick


Ramon

Nicholas

Manuel


Damien

Adam

Oscar


Daniel P.


Jason

Brian


Vincent P.

Kyle

Sam X.


Jacobs

Tyler


Alejandro

Sean


Karina


Michael J.

Victor


Moises


Joseph


David


Emmanuel


Ben

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

First Minute Quiz

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

Use CCC Confer 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 at school • Linux at home • More on I/O • All together now • Subtle differences • Errors • 2>&1 • More on I/O - programming • umask • More pipeline practice • Pipeline and redirection practice • More on pipelines • Eggs, treats and tricks • Review • Make teams • Flashcard practice • Assignment • Wrap up

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.*

Housekeeping

Housekeeping

1. Lab 7 due 11:59PM tonight -- **don't forget to submit your latest version!**

2. Read your Opus-II email for Lab 7 submission status.

3. A **check7** script is available.

4. Fine Print:
   Test #2 is scheduled for our next class!

Test #2 will happen during our next class!

Practice test available after class.

Test #2 is scheduled for our next next class!

Practice test available after class.


Test #2 is scheduled for our next class!
Practice test available after class.

Test #2

1. Test #2 is **scheduled for our next class!** 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.

2. Practice Test #2 is available after class on Canvas!

3. Work the Practice Test BEFORE the real test begins.

4. The Practice Test and Practice Test server will be available until about 30 minutes before the real test starts.

How to pass Test #2 with flying colors

- Keep taking the practice test until you can complete each question in 30 seconds or less. Use the ./restore command to restore your directory on the test system to the original state.
- Use the forum to discuss your approaches and results with classmates.
- If a question takes longer than 30 seconds ask for help. You can ask for help on the forum, see me during office hours, work with a tutor in the STEM center, join a study group or all of the above!
- Create a custom crib sheet of commands and key concepts covered in the course.
- Use the flashcards on the course website and rework any labs and previous tests you want to better understand.
- Tip: Use the -v option on chmod, mv, cp and rm commands to see what actually happened.
- But most important … DON'T WAIT TILL THE LAST MINUTE!

Study Groups are an Excellent Way to Learn!

The 2nd Test is just around the corner...

by **Jade Steck** » Mon Mar 14, 2016 3:28 pm

Hello everyone,

If any of you are interested in coming to the STEM center on Friday's or over the weekend let me know. I'd like to have a study group session and discussions. We can go thru all the commands, how it works, how to find a better way to tackle stress and time management during the EXAM. Please let me know who's interested. The sooner we study the better I know Rich hasn't posted the practice exam2, but we call always work on what we have... moving, changing, making files, chmod, etc... we can do this. If you are struggling its not too late. I find it helpful working with Rich and other CIS tutors at the STEM on Mondays. You can check their schedules too.

I'll see you all on Wednesday,

thanks,
Jade Steck

Note: Exam2 is right after S P R I N G B R E A K

Last edited by Jade Steck on Mon Mar 14, 2016 3:52 pm, edited 2 times in total.

Jade Steck

Posts: 42
Joined: Tue Sep 08, 2015 8:01 pm

An example forum post for a study group

To get notifications of new forum posts

Subscribe to the forum to get email notifications of new posts

After logging in:

1. Go to the CIS 90 class forum.
2. At the bottom of the page, click the "Subscribe forum" link on the lower left. When subscribed you get email notifications when new posts are made.
3. To unsubscribe, click it again.

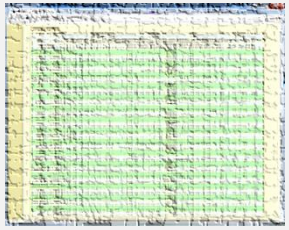
Unsubscribed looks like this.

Subscribed looks like this.

Review your progress in the course

The CIS 90 website Grades page

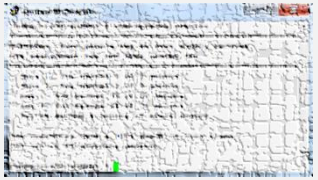
http://simms-teach.com/cis90grades.php



Or check on Opus-II

checkgrades *codename*

(where codename is your LOR codename)



Written by Jesse Warren a past CIS 90 Alumnus

Send me your survey to get your LOR code name.

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

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**

Heads up on Final Exam

Test #3 (final exam) is MONDAY December 11th 1-3:50PM

Mon	12/11	**Test #3 (the final exam)** **Time** • MONDAY 1:00PM - 3:50PM in Room 828 **Materials** • Test (canvas) **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 3:50PM.
- Working and long distance students can take the test online via CCC Confer and Canvas.
- Working students will need to plan ahead to arrange time off from work for the test.
- Test #3 is **mandatory** (even if you have all the points you want)

FALL 2017 FINAL EXAMINATIONS SCHEDULE
DECEMBER 11 TO DECEMBER 16

DAYTIME FINAL SCHEDULE

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 the 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	Monday, December 11
9:00 am and 10:15 am, MW/Daily	7:00 am-9:50 am	Wednesday, December 13
10:20 am and 11:35 am, MW/Daily	10:00 am-12:50 pm	Monday, December 11
11:40 am and 12:55 pm, MW/Daily	10:00 am-12:50 pm	Wednesday, December 13
1:00 pm and 2:15 pm, MW/Daily	1:00 pm-3:50 pm	Monday, December 11
2:20 pm and 3:35 pm, MW/Daily	1:00 pm-3:50 pm	Wednesday, December 13
3:40 pm and 5:30 pm, MW/Daily	4:00 pm-6:50 pm	Monday, December 11
6:30 am and 8:55 am, TTh	7:00 am-9:50 am	
9:00 am and 10:15 am, TTh	7:00 am-9:50 am	
10:20 am and 11:35 am, TTh	10:00 am-12:50 pm	
11:40 am and 12:55 pm, TTH	10:00 am-12:50 pm	
1:00 pm and 2:15 pm, TTh	1:00 pm-3:50 pm	
2:20 pm and 3:35 pm, TTh	1:00 pm-3:50 pm	
3:40 pm and 5:30 pm, TTh	4:00 pm-6:50 pm	
Friday am	9:00 am-11:50 am	
Friday pm	1:00 pm-3:50 pm	
Saturday am	9:00 am-11:50 am	
Saturday pm	1:00 pm-3:50 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. Recommended Preparation: CIS 1L or CIS 72.

Transfer Credit: Transfers to CSU;UC

Section	Days	Times	Units	Instructor	Room
98169	W	1:00PM-4:05PM	3.00	R.Simms	OL
&	Arr.	Arr.		R.Simms	OL

Section 98169 is an ONLINE course. Meets weekly throughout the semester online during the scheduled times by remote technology with an additional 50 min online lab per week. For details, see instructor's web page at go.cabrillo.edu/online.

98170	W	1:00PM-4:05PM	3.00	R.Simms	828
&	Arr.	Arr.		R.Simms	OL

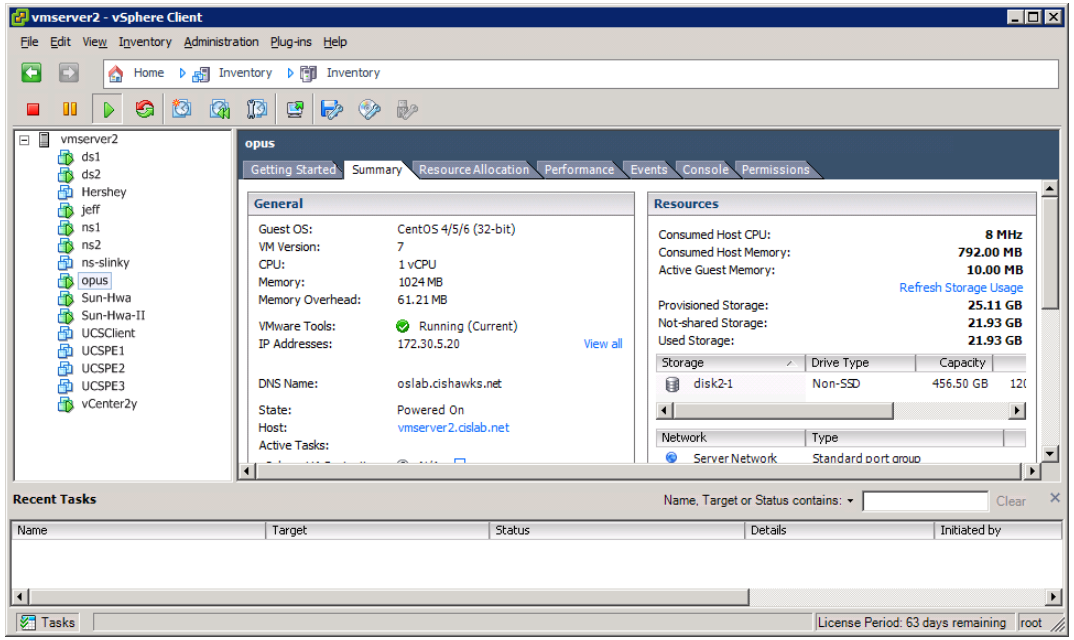
Section 98170 is a Hybrid ONLINE course. Meets weekly throughout the semester at the scheduled times with an additional 50 min online lab per week. For details, see instructor's web page at go.cabrillo.edu/online.

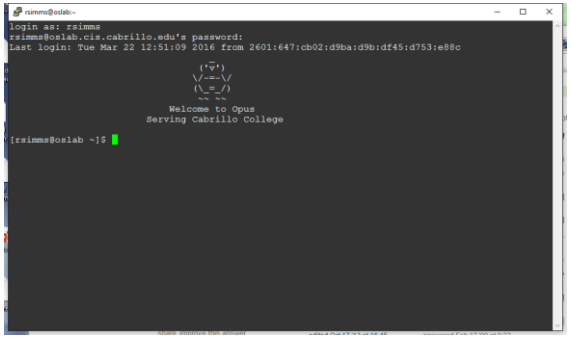
Linux at School

Our Opus-II server on campus

Dell R610 Server

VMware vSphere Client





Opus-II is a VM running on one of the Vmware ESXi servers in the CIS Datacenter

Linux at Home

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

USB "Live" Linux Multi-Boot USB Flash Drive

Windows

CentOS

Kali

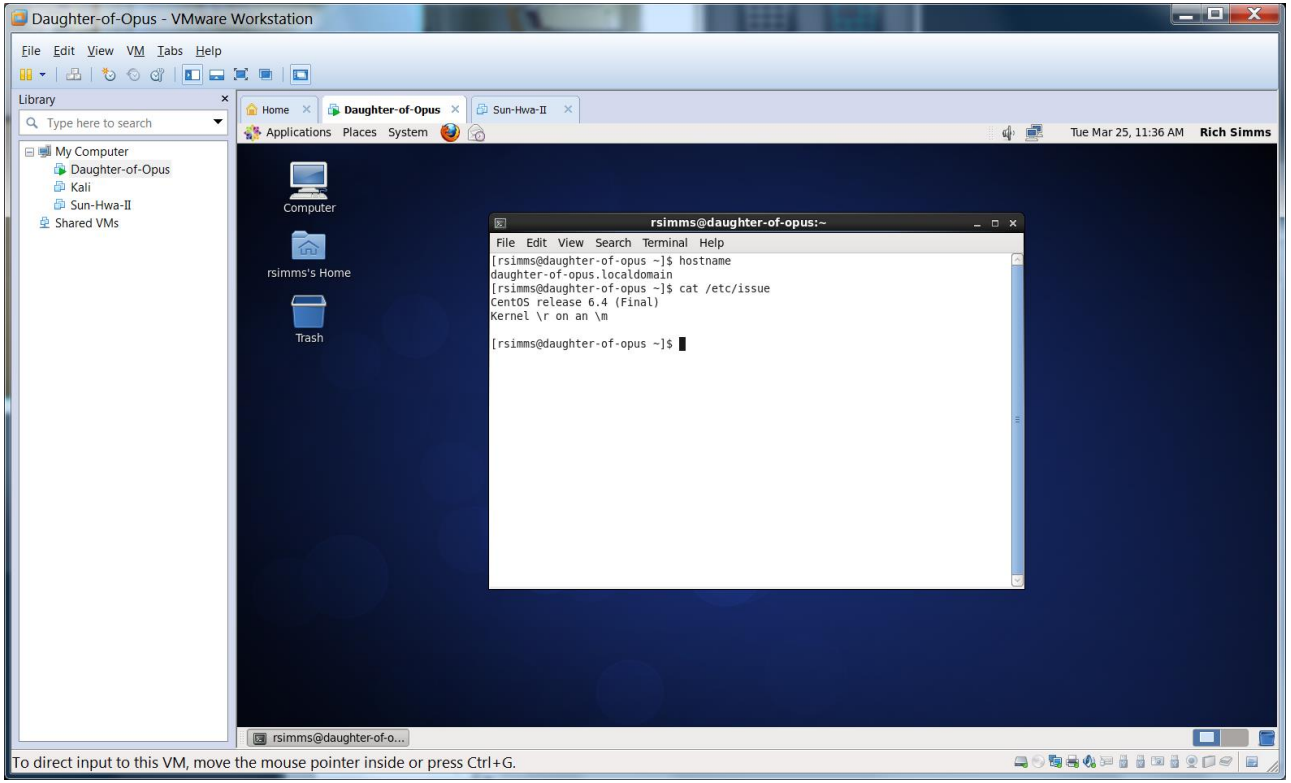
YUMI formatted Flash Drive
(www.pendrivelinux.com)

Linux Mint

Ubuntu

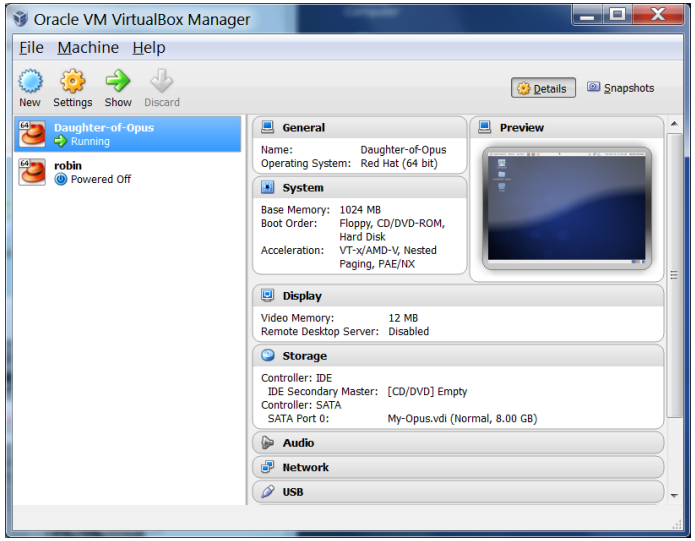
Allows you to use or try out Linux on an existing computer without installing it

VMware Workstation (PC) or Fusion (Mac)

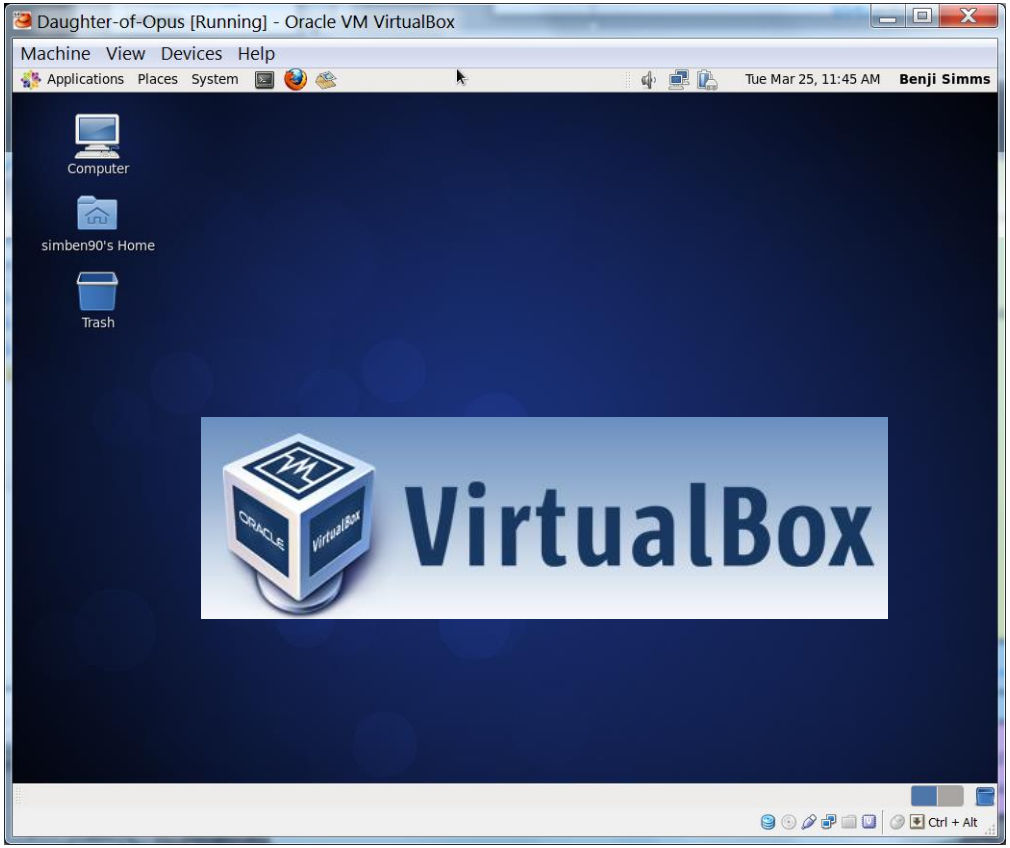


One Daughter-of-Opus is a VM running on my laptop using VMware Workstation (expires in one year)

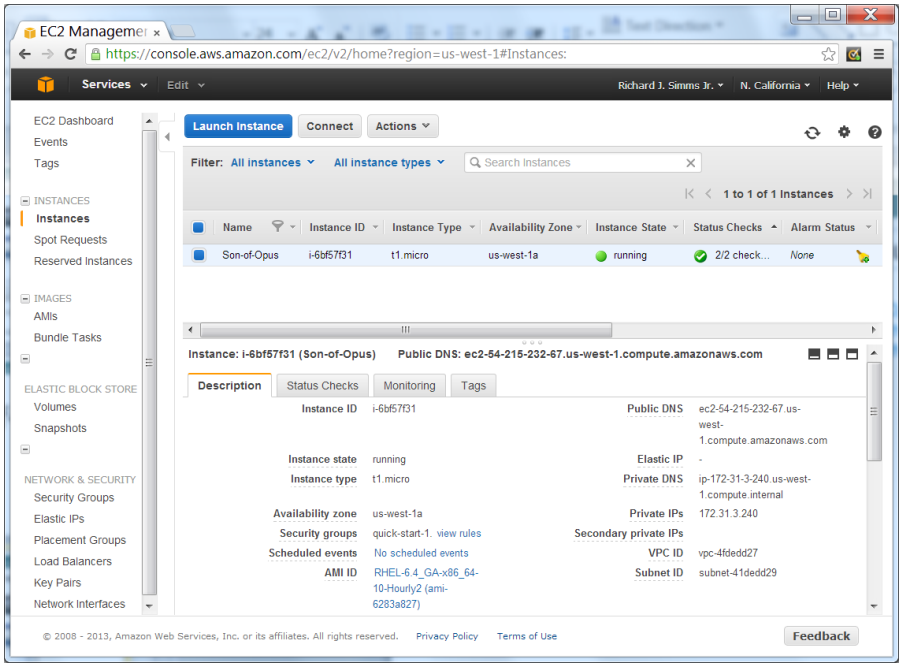
VirtualBox



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



Amazon Web Services

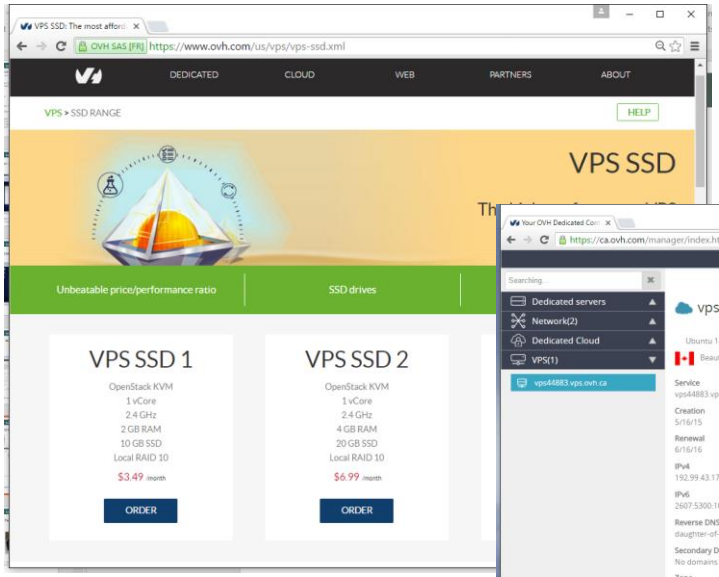


Son-of-Opus is a VM running on Amazon Web Services

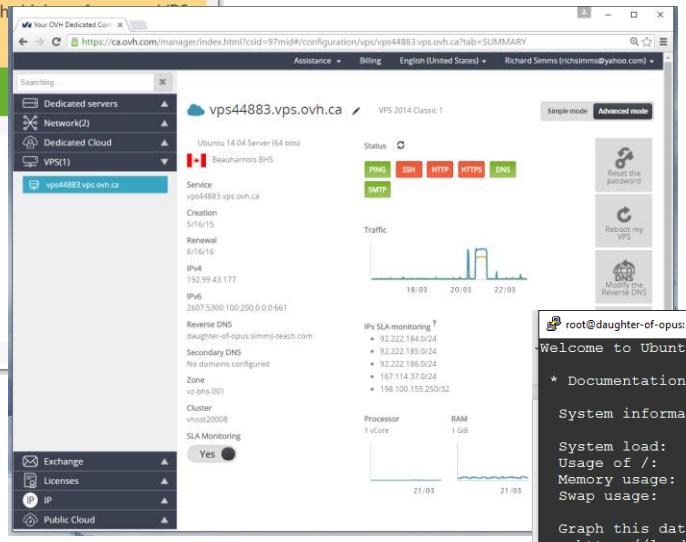
Single VM is free for a year, then about $60 per month after that.

OVH.com

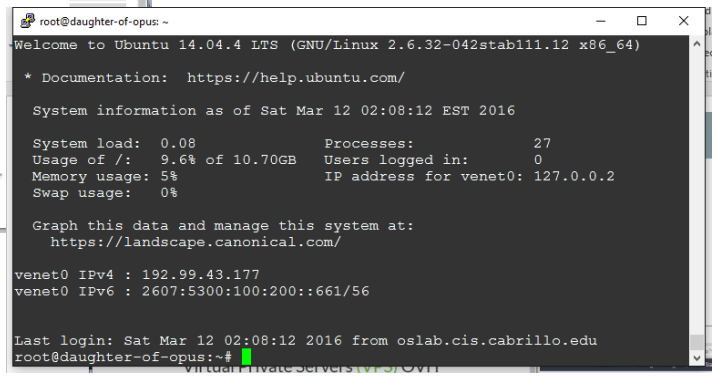
Purchase



Configure



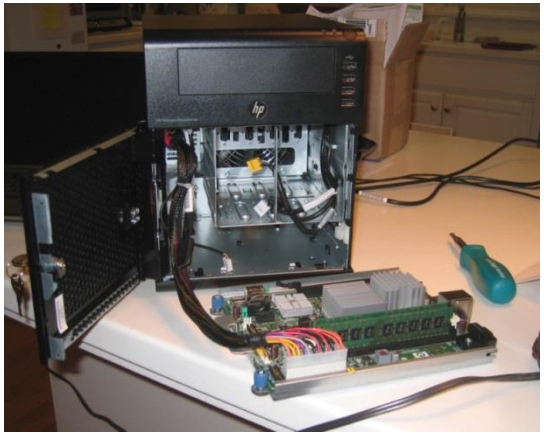
Use



Virtual private servers like daughter-of-opus used on Test #1 costs $3 per month

Small Form Factor Servers

HP Microserver



https://www.hpe.com/us/en/product-catalog/servers/proliant-servers.filters-facet_subbrand_url:ProLiant-MicroServer.hits-12.html

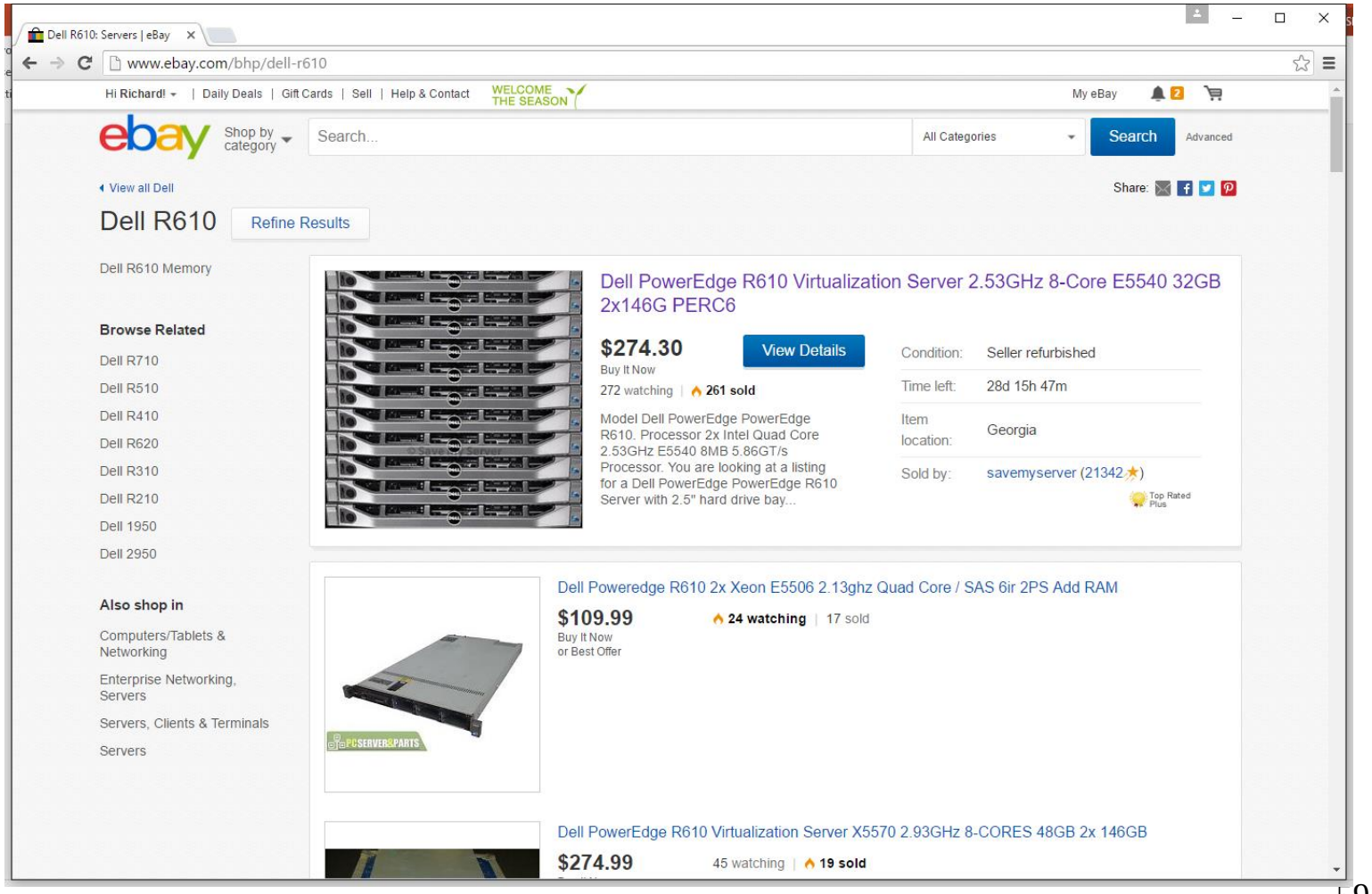
vmware ESXi

SuperMicro

https://tinkertry.com/my-tinkertry-d-xeon-d-bundle-2-supermicro-superserver-bundle-2-of-joy

Comparatively inexpensive "bare bones" servers that come without memory, hard drives or an operating system

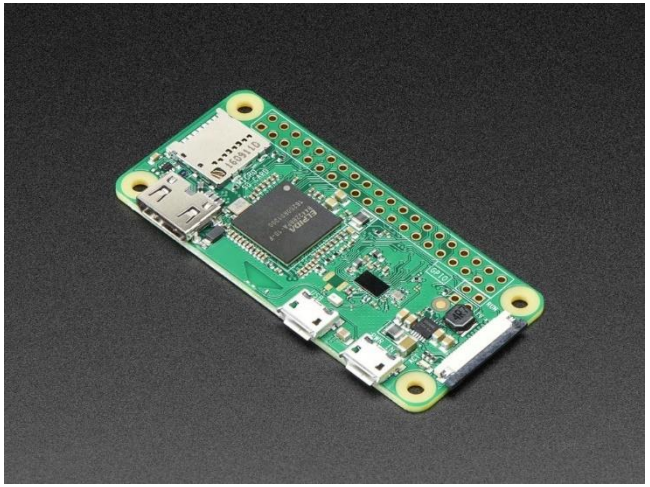
Fantastic Bargains on EBay



Raspberry Pi

Raspberry Pi 3 Model B

https://www.adafruit.com/product/3055

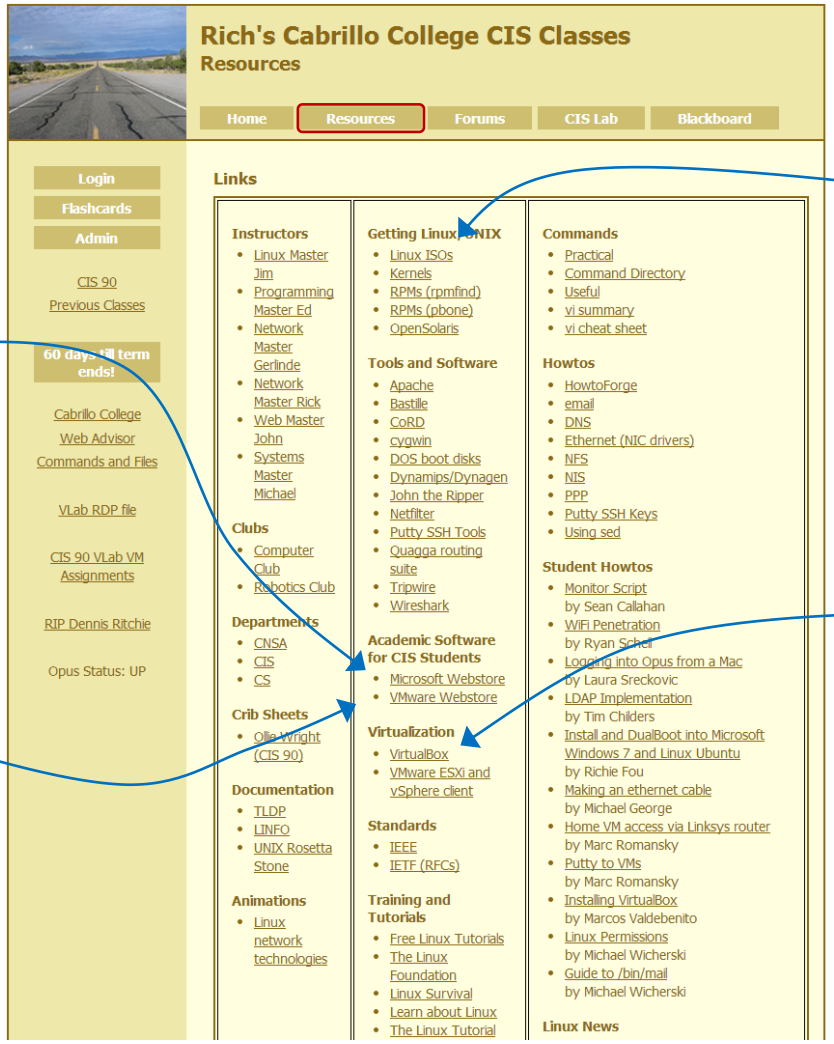


Raspberry Pi Zero W

https://www.adafruit.com/products/3400

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

Microsoft Software (Academic)



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*

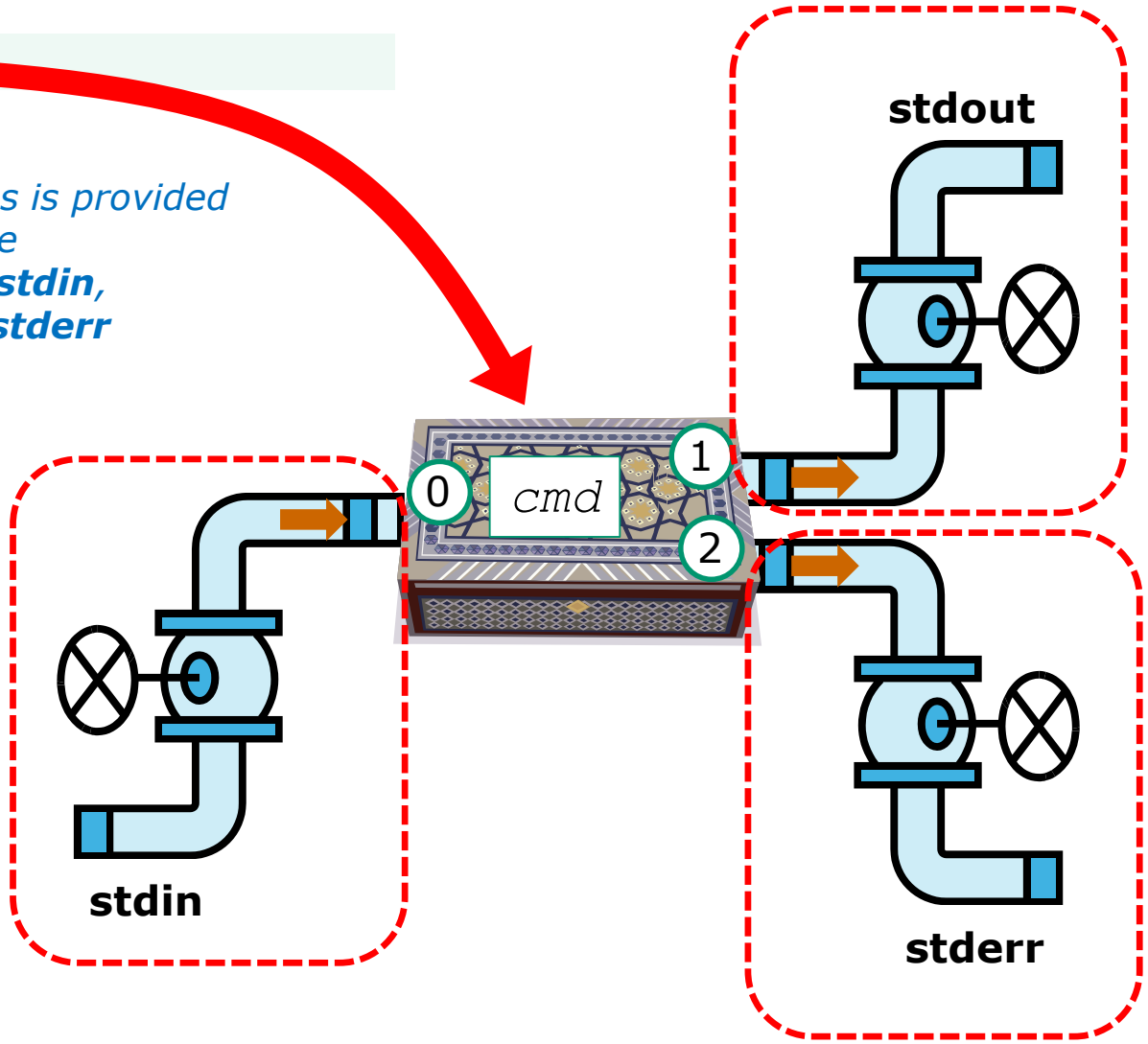
Examples

```
/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**

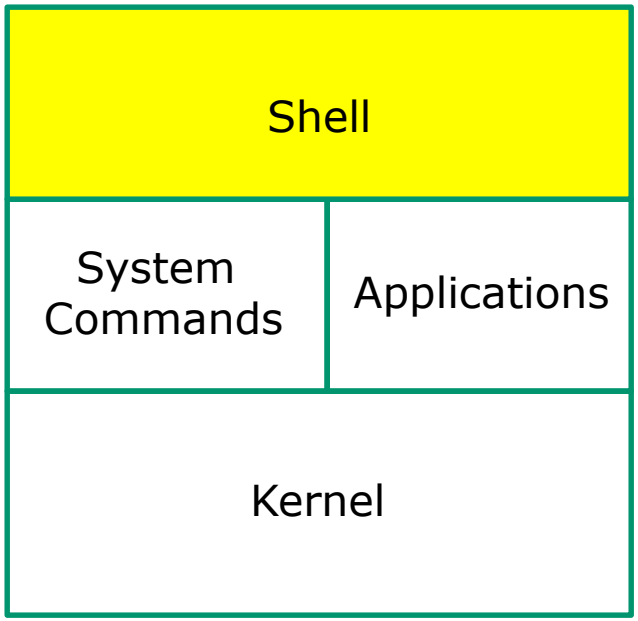
```
$ cmd
```

*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 (show with **echo $PS1**).

```
/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.

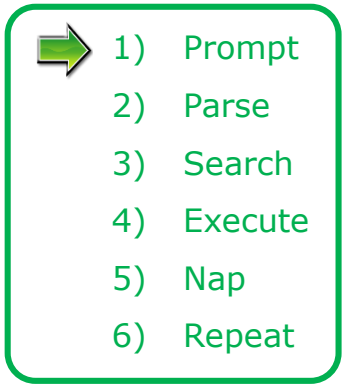
Activity

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*

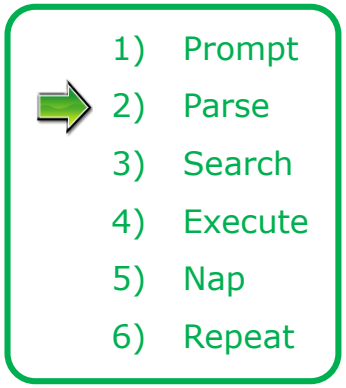
1) Prompt
2) Parse
3) Search
4) Execute
5) Nap
6) Repeat

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



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

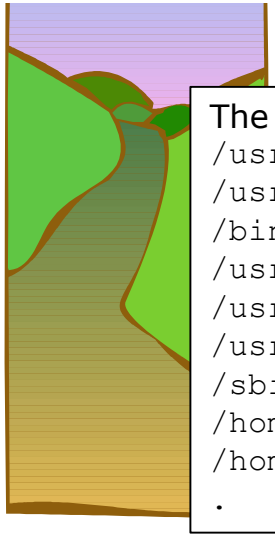
1) Prompt
2) Parse
3) Search
4) Execute
5) Nap
6) Repeat

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.

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

Activity

Prove to yourself that the shell will find the **sort** command in the */bin* directory.

1. Use **echo $PATH** to view your path.

2. Check the first three directories on your path to see if one of them contains the sort command:
   - Use **ls -li /usr/lib/qt-3.3/bin | grep sort**
   - Use **ls -li /usr/local/bin | grep sort**
   - Use **ls -li /bin | grep sort**

3. Write the inode number of the sort program file in the chat window.

Example

argument
option
redirection

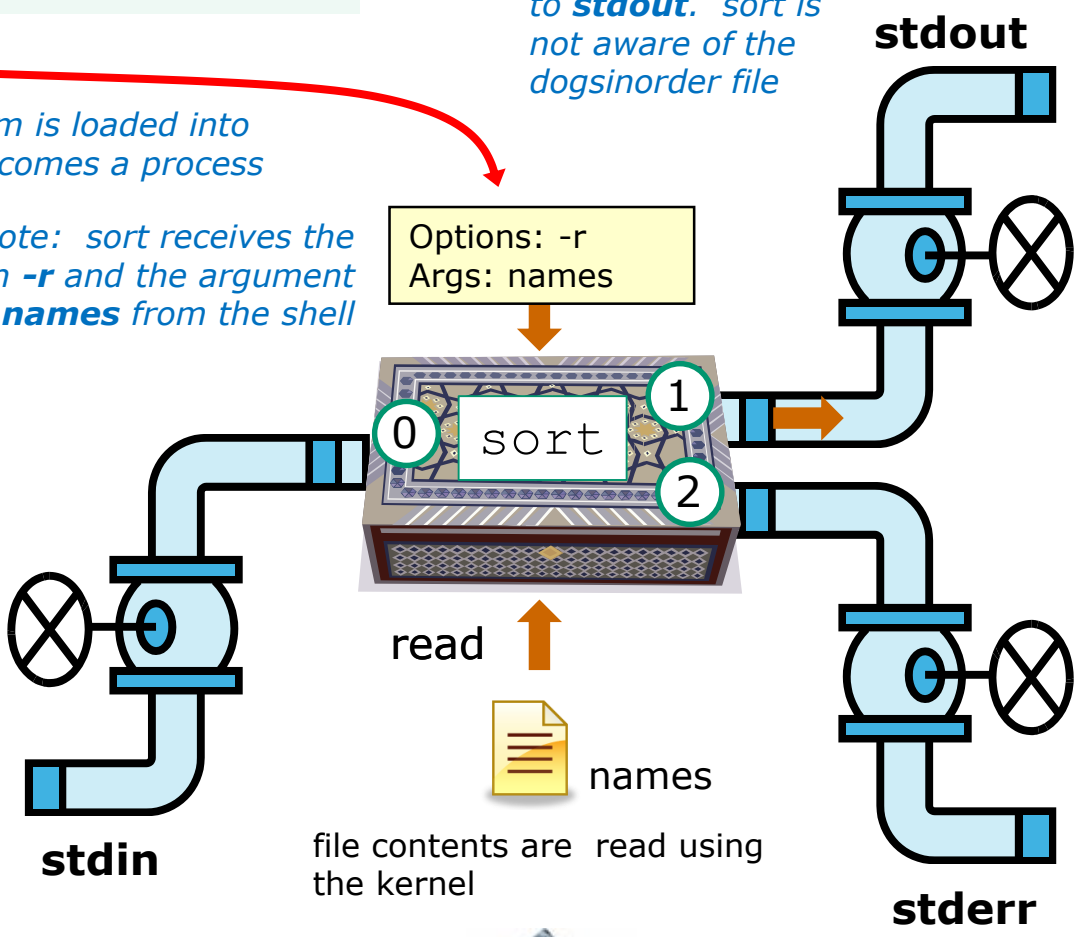
```
$ sort -r names > dogsinorder
```

The sort program is loaded into memory and becomes a process

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

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

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



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

file contents are read using the kernel

sort opens and reads the names file

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.
```

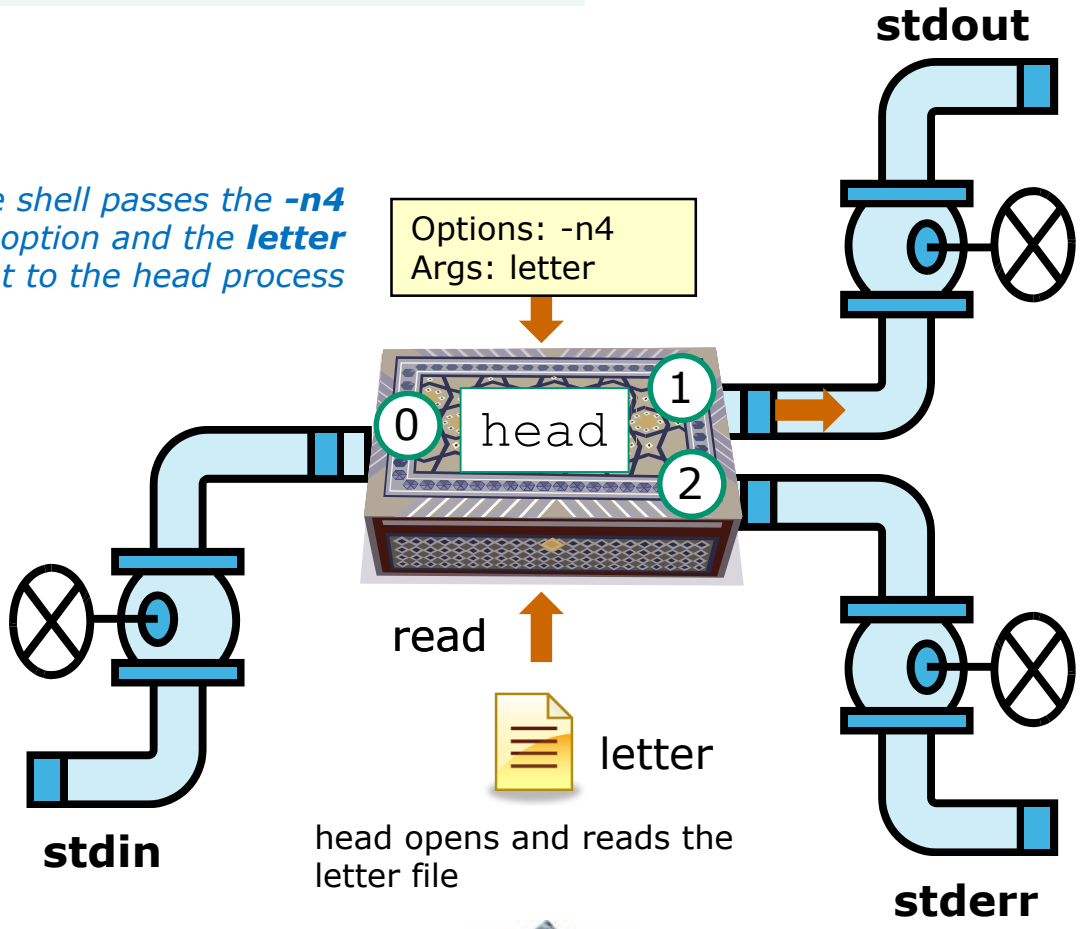
head -n4 letter

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*

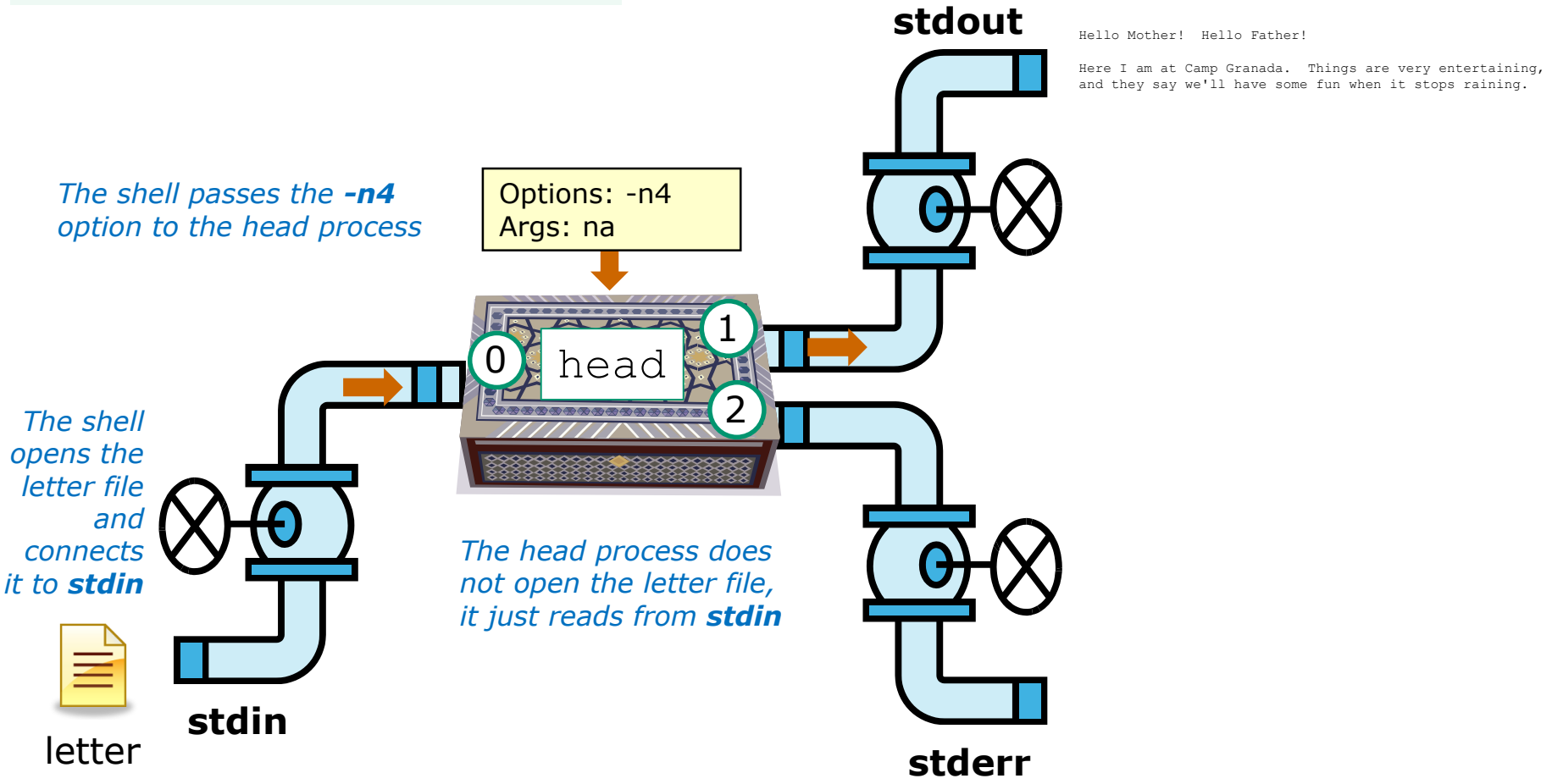


head opens and reads the letter file

head -n4 < letter

option *redirection*

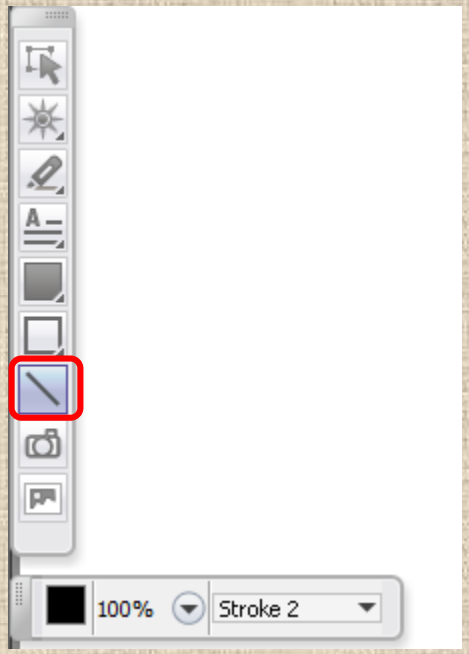
```
$ head -n4 < letter
```



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

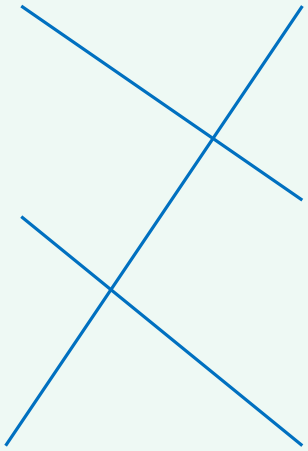
Connect with a straight line the command with the error message

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`

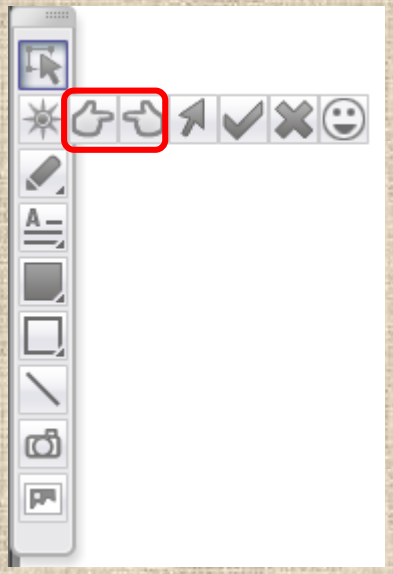
CCC Confer Whiteboard Activity

Connect with a straight line the command with the error message

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



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!

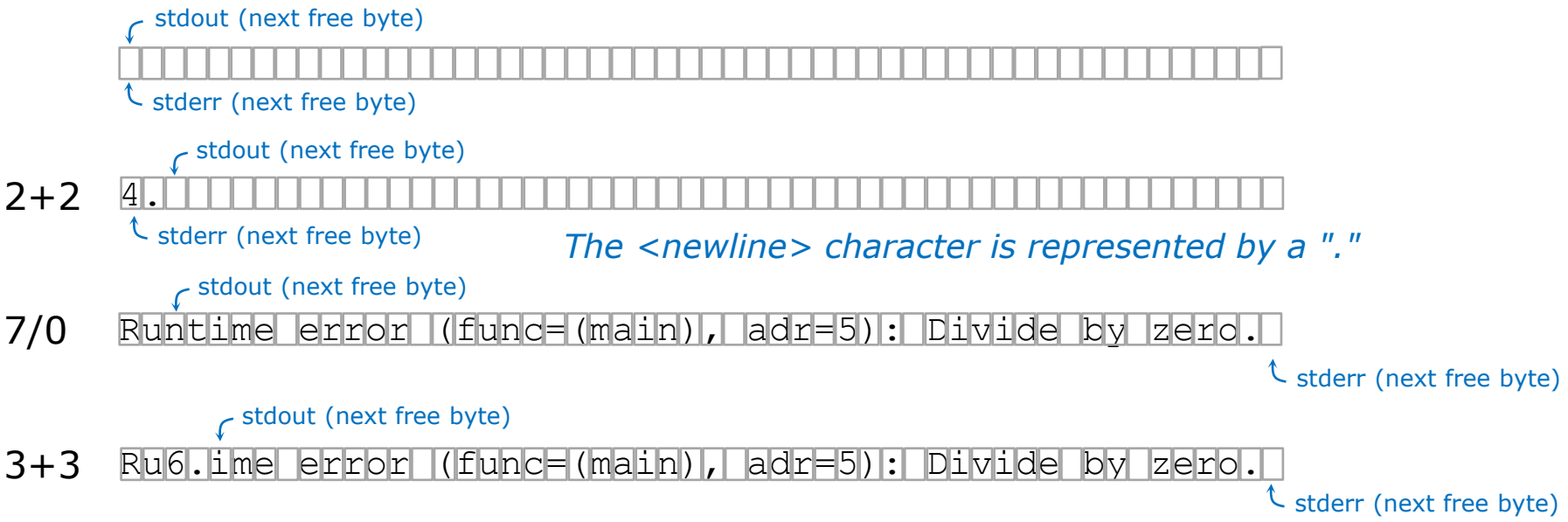
```
/home/cis90/simben $ bc > calculations 2> calculations
2+2
7/0
3+3
quit

/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)

programming examples

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

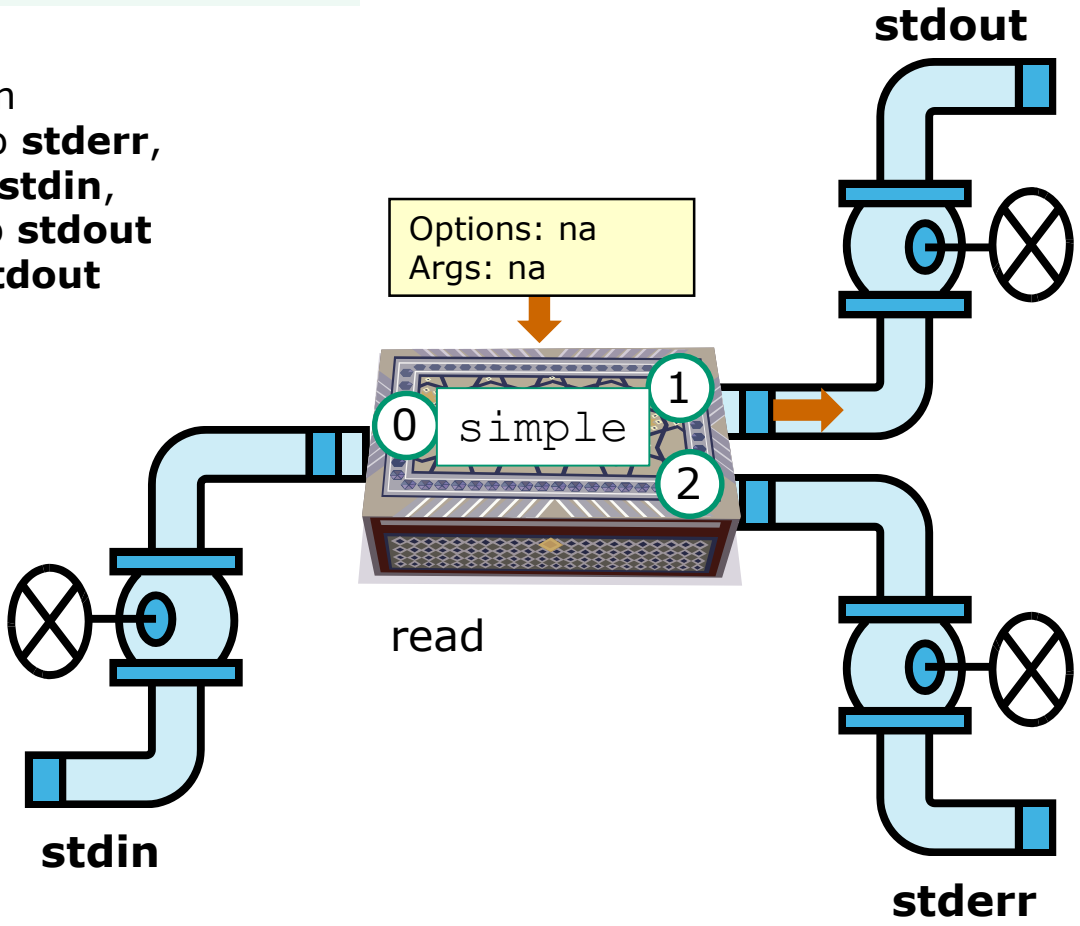
```
/home/cis90/simben/bin $ simple
What is your name stranger? Rich
Well I'm very pleased to meet you, Rich
```

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.

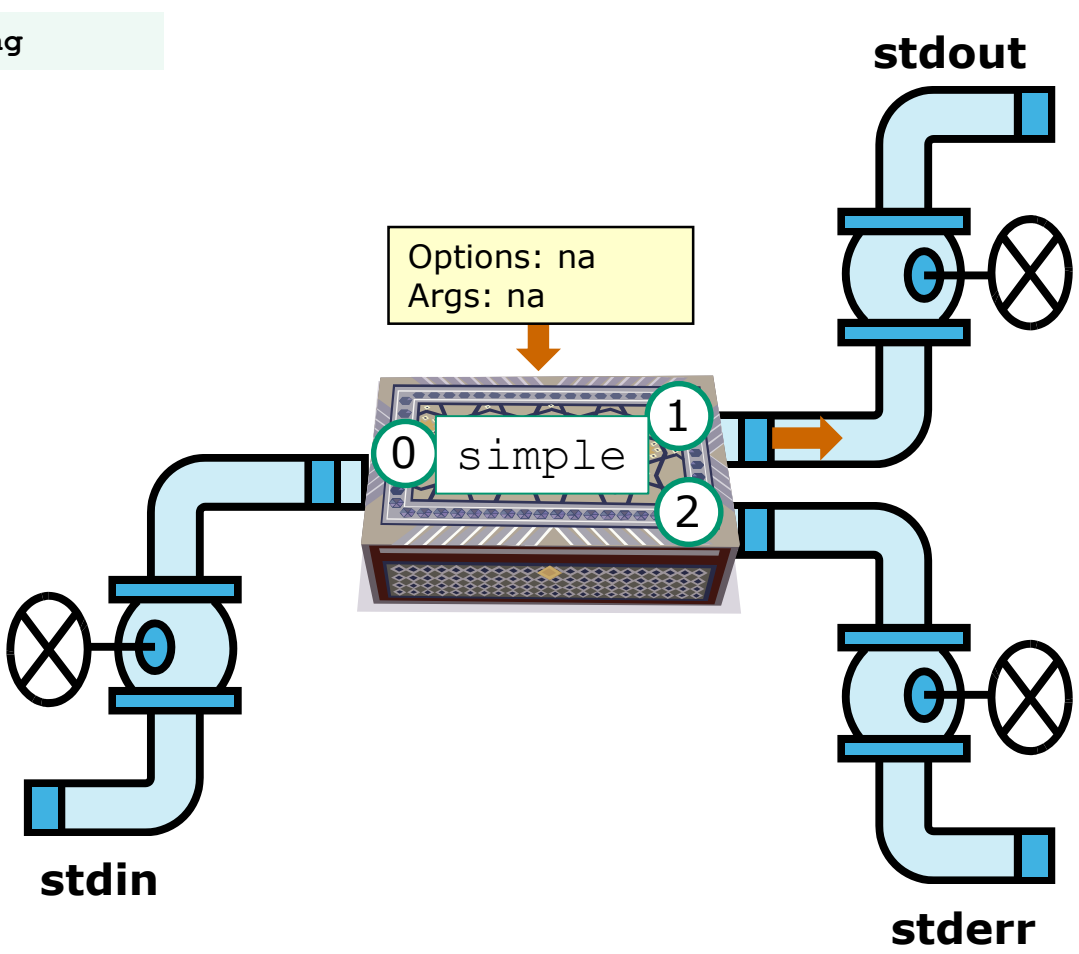
```
/home/cis90/simben/bin $ simple > myfile
What is your name stranger? Rich

/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 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 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

FYI only

Python Script I/O example
View the program

```
/home/cis90/simben $ cat simple.py
import sys
sys.stderr.write("What is your name stranger? ") Output question to stderr
name = sys.stdin.readline() Input name from stdin
sys.stdout.write("Well I'm very pleased to meet you, " + name)
```

*Output greeting and name to **stdout***

This program is available in the depot directory

Python Script I/O example
View the program

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

Running the python simple.py script

Activity

1. Change to your bin directory
   cd ~/bin

2. Copy the python script from the depot directory
   cp ~/../depot/simple.py .

3. Look at your program
   cat simple.py

4. Run the script
   python simple.py

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

666	110 110 110
002	000 000 010
664	110 110 100

```
/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

777	111 111 111
002	000 000 010
775	111 111 101

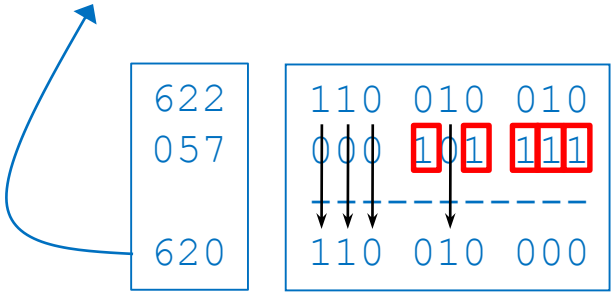
```
/home/cis90/simben/lesson9 $ mkdir newdir
/home/cis90/simben/lesson9 $ ls -ld newdir
drwxrwxr-x 2 simben cis90 4096 Oct 27 07:23 newdir
```

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
```

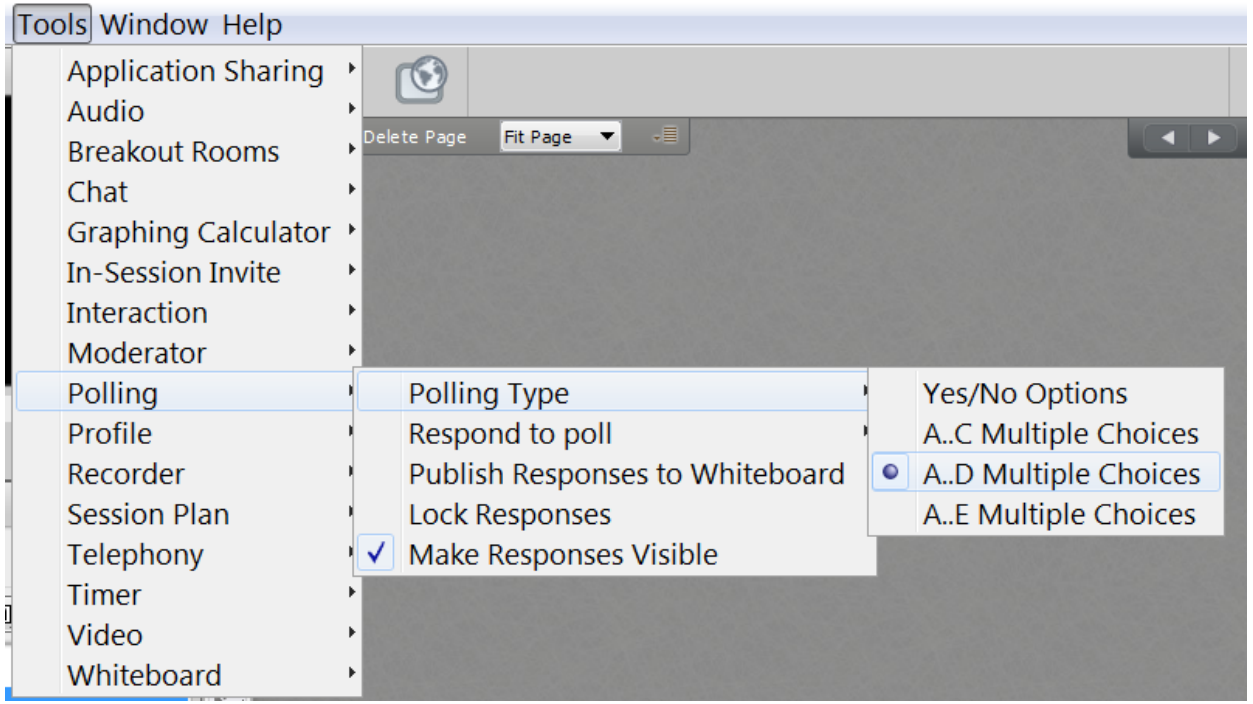
Example umask setting



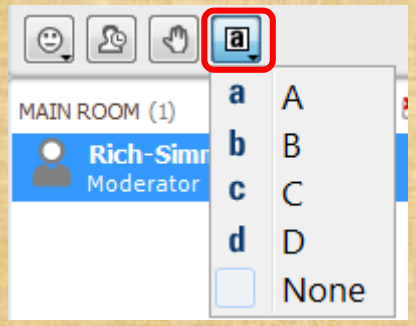
Copied file - start with original file's permissions and apply the mask

*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?

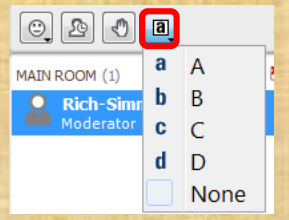
A) stat

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?

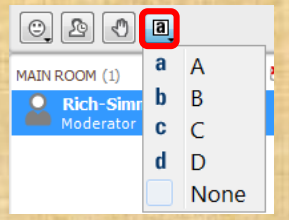
A) stat

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?

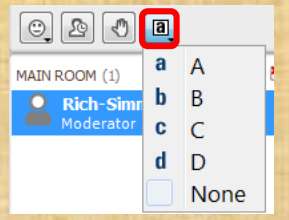
A) stat

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?

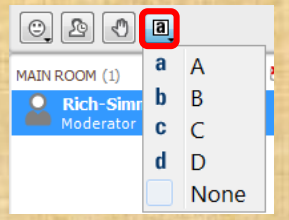
A) stat

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.

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 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 $ 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.*

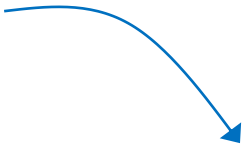
```
/home/cis90/simben $ head -n2 poems/Anon/nursery
Jack and Jill went up the hill
to fetch a pail of water.
/home/cis90/simben $ head -n2 poems/Anon/nursery | wc -l
2
/home/cis90/simben $
```

*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

Memorize This

How to locate files:

To locate by name ... use **find**

To locate by user, type, group, etc[1]. ... use **find**

To locate by contents[2] within specific files ... use **grep**

To locate by contents[2] in a branch of the file tree ... use recursive **grep**

1 File inode properties (e.g. shown in a long listing)
2 File data (e.g. shown when catting a file)

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

Finding Things

Task
Find an account for simben90 in /etc/passwd

```
/home/cis90/simben $ grep simben90 /etc/passwd
simben90:x:1201:190:Benji Simms:/home/cis90/simben:/bin/bash
```

Finding Things

Task
Find your account in /etc/passwd

```
grep ??????90 /etc/passwd
```

Write your results in the chat window

grep $LOGNAME /etc/passwd

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

Eggs, Treats and Tricks

trick or treat

A number of *trick* and *treat* files 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. The rest are scattered in the various subdirectories you own.

2. Make a new directory named *bag* in your home directory and see how many *trick* or *treat* files you can move into it.

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

Instructor: sudo /home/rsimms/cis90/halloween/hidetreats

Review

Jim's Summary Pages

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

Lesson 6 - Managing Files
https://web.archive.org/web/20100708145536/http://www.cabrillo.edu/~jgriffin/CIS90/files/lecture5.html

Lesson 7 - File Permissions
https://web.archive.org/web/20100708151130/http://www.cabrillo.edu/~jgriffin/CIS90/files/lecture6.html

Lesson 8 - Input/Output Processing
https://web.archive.org/web/20100708151725/http://www.cabrillo.edu/~jgriffin/CIS90/files/lecture7.html

Make Teams

Breakout Rooms

Room 1

Room 2

Room 3

Room 4

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 4 rooms)

Flashcard Practice

Flashcards

Room 1

Room 2

Room 3

Room 4

Flashcards
L6=20
L7=15
L8=16

Rules

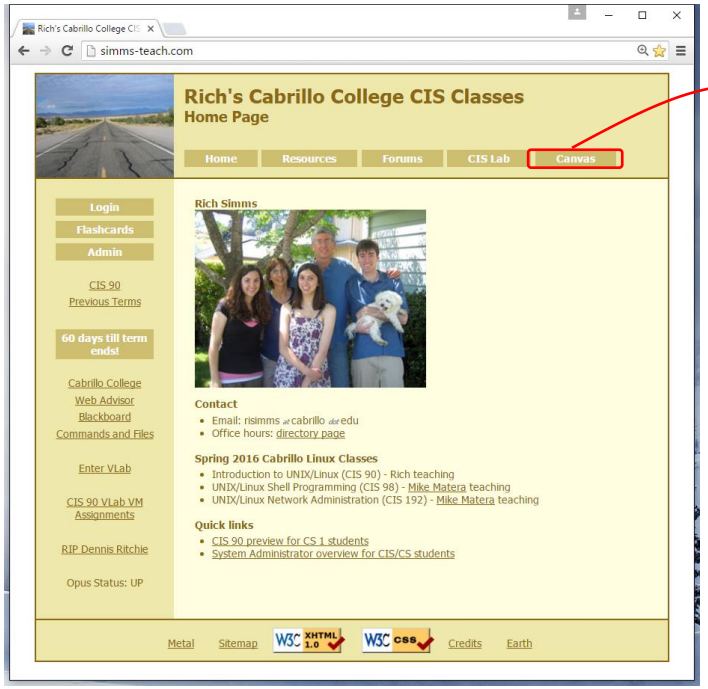
- Chat window belongs to team that is up
- Team gets the point if anyone on the team writes a correct answer in the chat window in 15 seconds

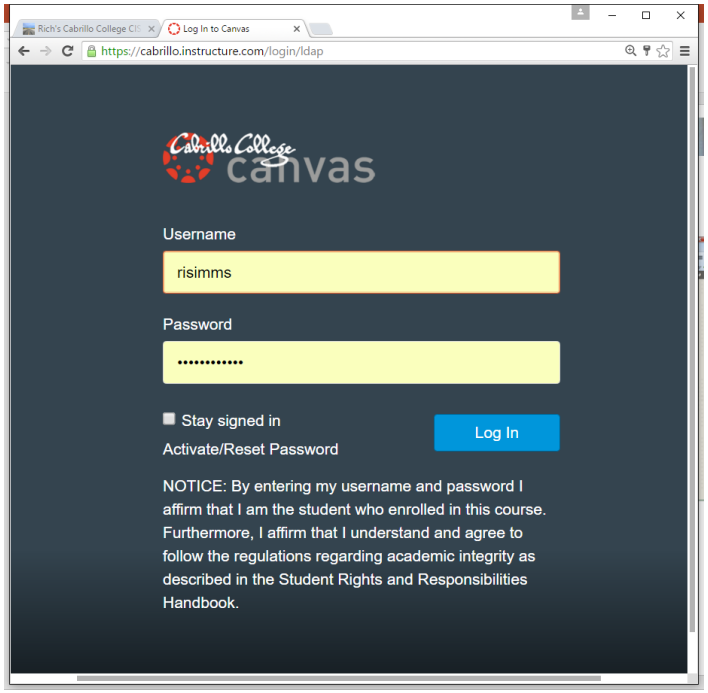
Instructor timer:
i=15; while [$i -gt 0]; do clear; banner $i; let i=i-1; sleep 1; done; clear; banner done
(Use **countdown** alias)


Assignment

Practice Test

Practice Test





Practice test available

- Available on Canvas
- Work alone or together in study groups
- 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 Instructions

HONOR CODE:
This is a practice test and you may work with others on it. You are encouraged to compare and discuss answers with your classmates using the forum, study groups or both. 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-II first, then ssh into sun-hwa-p2. The practice test and 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 a practice test question you can ask your classmates for help on the forum. If you get stuck on the real test and can't proceed you can ask the instructor for help and forfeit the point. For the real test the instructor will be available during class and available by email later in the evening from 8:00-10:00PM.

Please KEEP YOUR ANSWERS TO A SINGLE LINE ONLY !!

This is a practice test and unlike the real test you can take it as many times as you want. To prepare for the real test keep taking this practice test over and over again till you can answer each question in under 30 seconds.


Wrap up

Next Class

No Quiz

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

- Recommended preparation:
  - Work the practice test!
  - Restore your directory with ./restore and work the practice test again!
  - Repeat step above till you can answer all questions in 30 seconds
  - 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

Notes to instructor

Practice Test System

[ ] Start: `echo "/root/unlock-cis90; rm /etc/nologin" | at` *[T-1wk+1hr]*

[ ] End: `echo "/root/lock-cis90; cp /etc/nologin.bak /etc/nologin" | at` *[T-30]*

Canvas Practice Test:

[ ] availability from = *[T-1wk+1hr]*, due & available until = *[T-30]*

[ ] remove password on real test on Canvas *[T-1wk]*

[ ] moderate any accommodations

Real Test system

[ ] Start: `echo "/root/unlock-cis90; rm /etc/nologin" | at` *[T-0]*

[ ] End: `echo "/root/lock-cis90; cp /etc/nologin.bak /etc/nologin" | at` *[splashdown]*

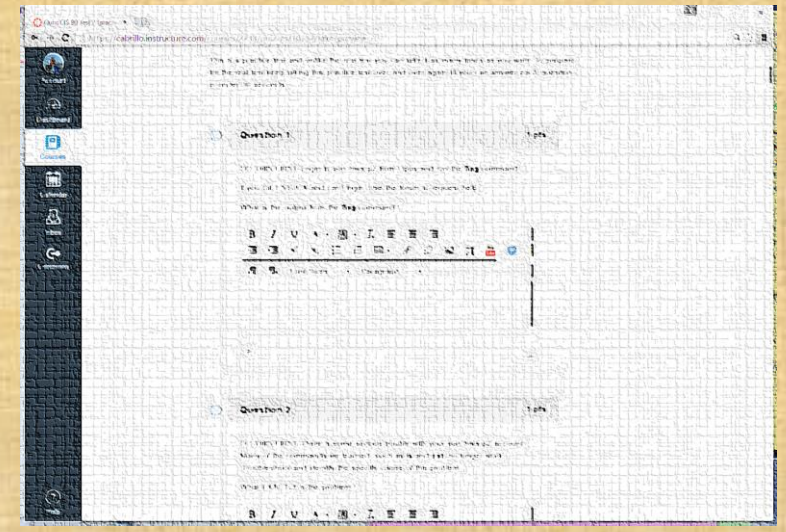
Canvas Real Test:

[ ] availability from = *[T-0]*, due & available until = *[splashdown]*

[ ] remove password on real test on Canvas *[T-0]*

[ ] moderate any accommodations

Optional Workshop Today



Work the practice test till the end of class.

- Collaborate!
- Ask questions!
- Arrange study groups!

Backup