Lesson Module Checklist

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

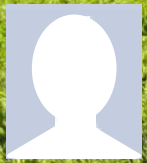
- Opus - hide script tested –
- Practice test uploaded –
- Sun-Hwa – trouble made and rocks hidden
- CCC Confer wallpaper with quiz -

- Set up Polycom phone/extension mics -
- Check that headset is charged –
- Wireless lapel mic backup battery -
- Backup slides, CCC info, handouts on flash drive -

CIS 90 - Lesson 9

Instructor: **Rich Simms**
Dial-in: **888-450-4821**
Passcode: **761867**


Sean C.


Donald

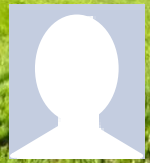

Carlile


Andrew


Sean Fa.


Carter


Sean Fy.


Dajan


Bryn

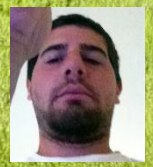

Rita


Kelly

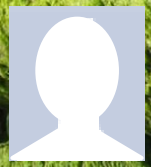

Ben


Ray


Fidel

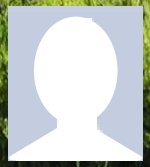

Michael

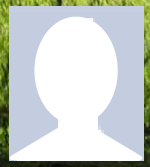

Evan


Josh


Carlos


Gustavo


Jessica


Evie


Jacob


Humberto


Chad

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

Quiz

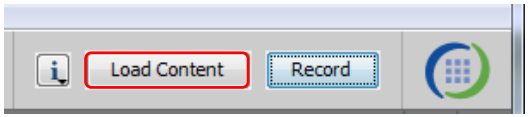
Please answer these questions **in the order** shown:

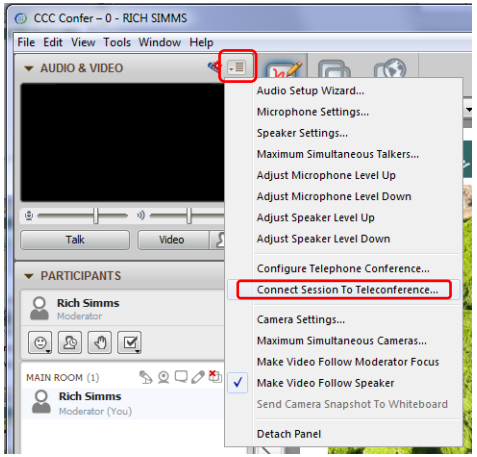
See electronic white board

email answers to: `risimms@cabrillo.edu`

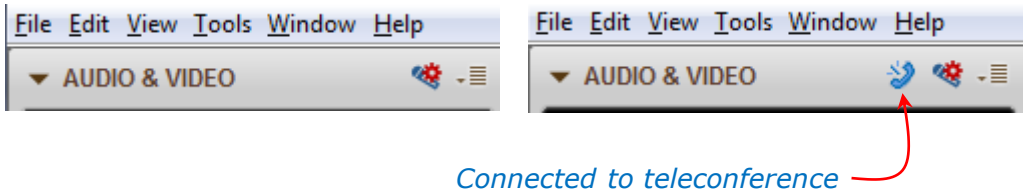
(answers must be emailed within the first few minutes of class for credit)

[ ] Load White Board with *cis*lesson??*-WB*

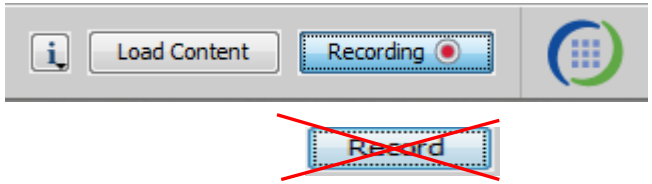




[ ] Connect session to Teleconference



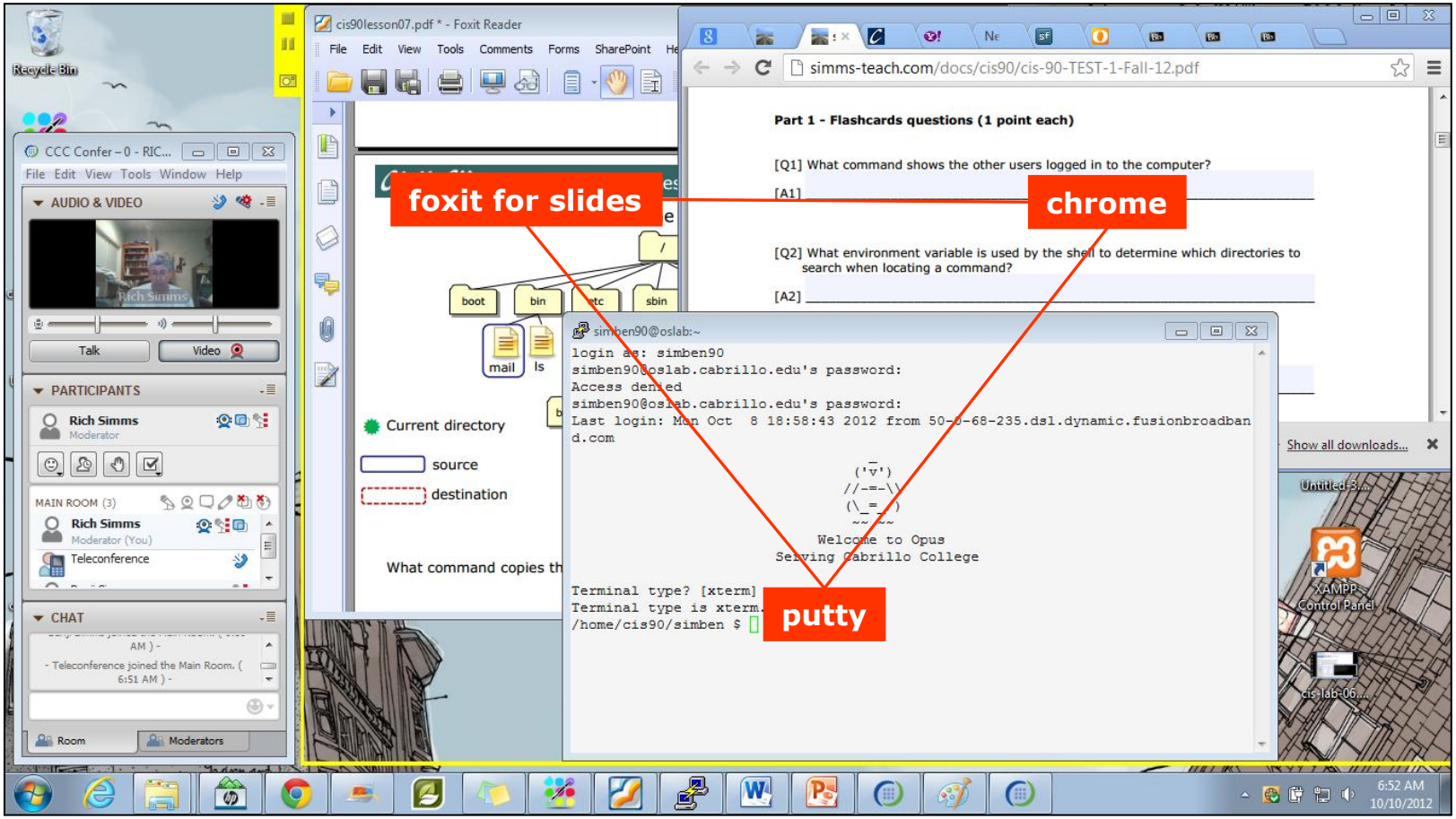
[ ] Is recording on?



[ ] Toggle Talk button to not use Mic

[ ] Video (webcam) optional

[ ] layout and share apps



Review

Objectives	Agenda
• Get ready for the next test • Practice skills • Introduction to processes	• Quiz • Questions • Lab 6 • Warmup • Base knowledge • Shell • Metacharacters • Environment variables • File system • File management • Permissions • I/O • Wrap up

Questions

Previous material and assignment

Lab 7 questions?
Extra credit Lab questions?
Questions on redirection and pipes?
Any other material?

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

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

He who asks a question is a fool for five minutes; he who does not ask a question remains a fool forever.
- Chinese Proverb

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

Prompt	Command	Options	Arguments	**Redirection**

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

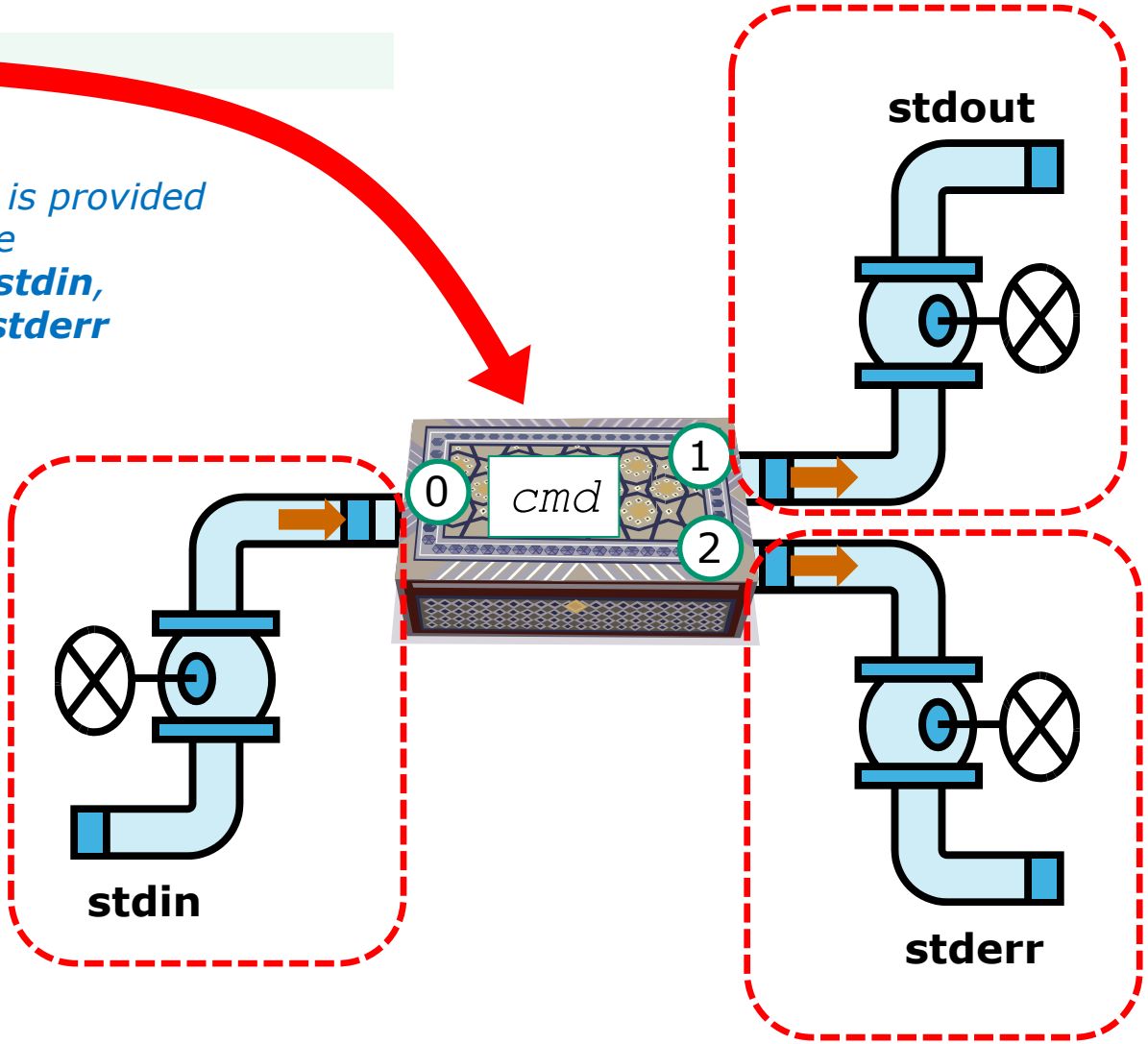
Examples

```
/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**

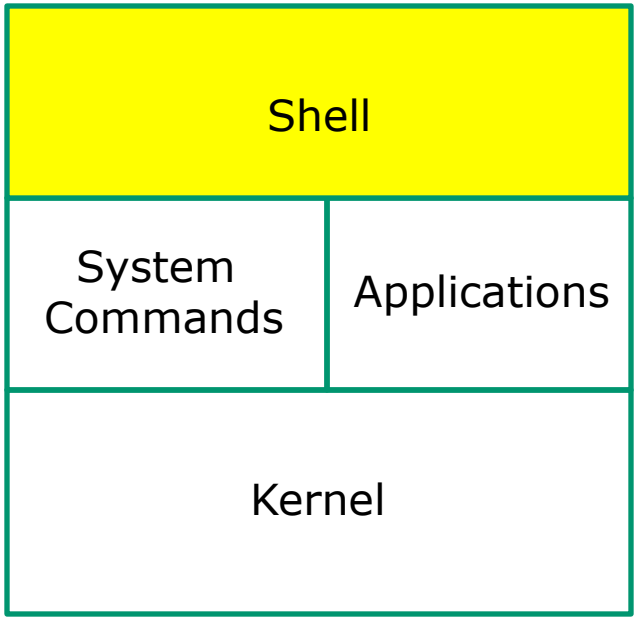
```
$ cmd
```

*Ever 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 $
```

This shell prompt is generated by a PS1 variable is set to '$PWD $ ' which shows the current position in the file tree followed by a blank, a $, and another blank.

Example

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

The user then enters a command after the prompt string followed by the Enter key

- *The Enter key generates a <newline> which is a shell metacharacter. Metacharacters have special meanings.*
- *The <newline> characters tells the shell it is time to go to the next step and parse the command.*

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

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

Example

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

*The shell now **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*

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

Parsing results:
*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*
- *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*

`sort`

The shell locates the sort command in the /bin directory which is on the third directory of a CIS 90 students path.

Example

argument

option

redirection

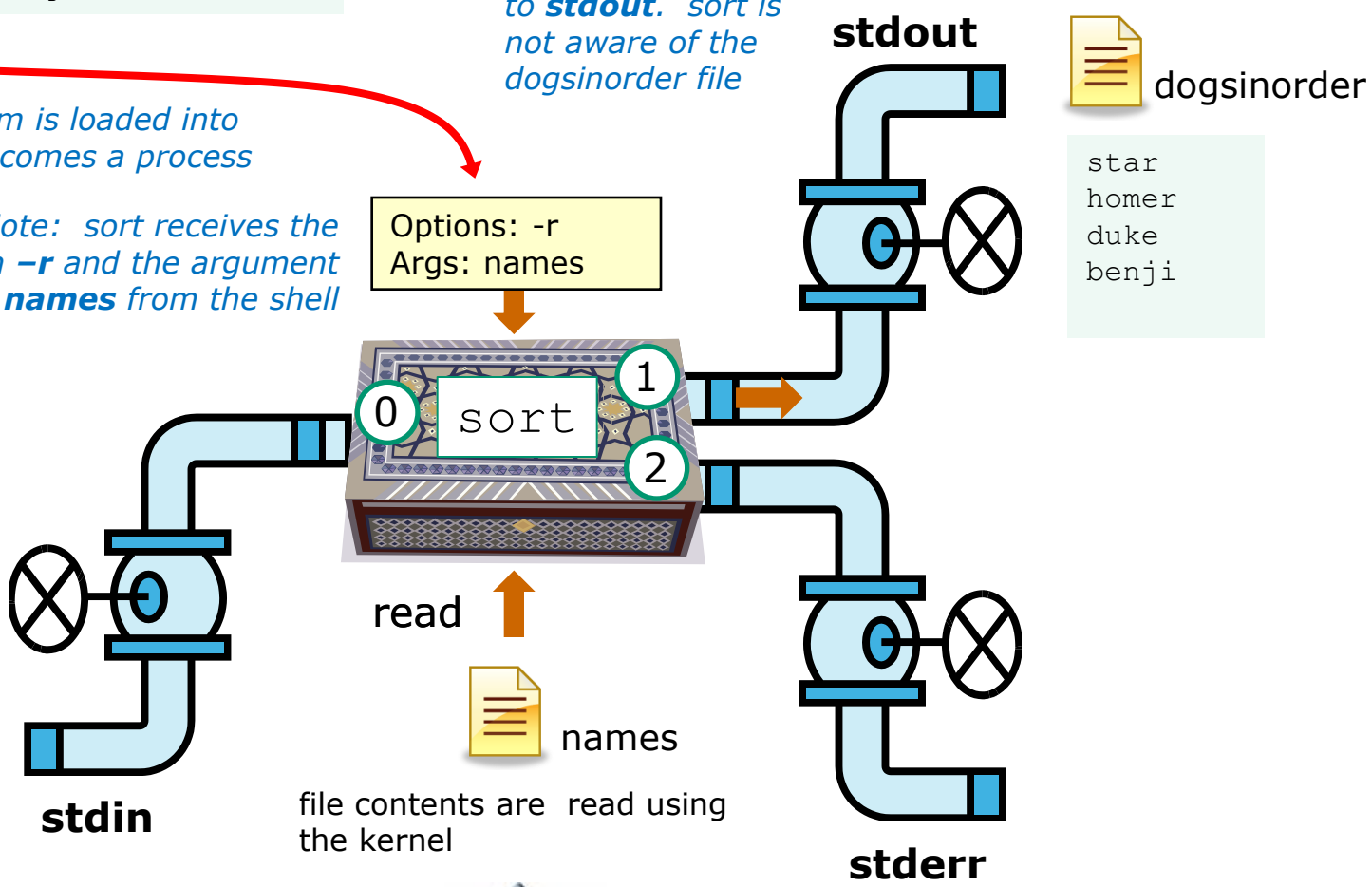
```
$ sort -r names > dogsinorder
```

*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

file contents are read using the kernel

sort opens and reads the names file

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 get and process another 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.
```

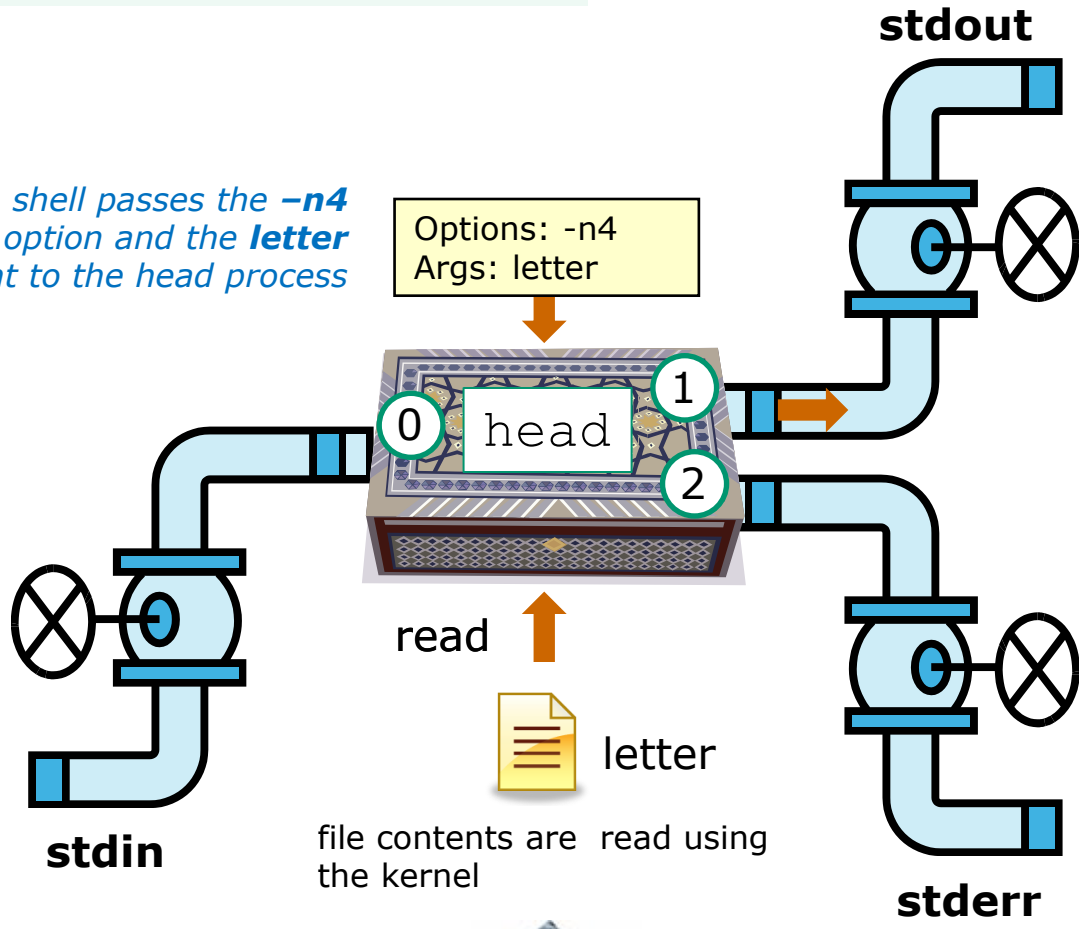
```
/home/cis90/simben $  head -n4 < letter
Hello Mother!  Hello Father!

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
```

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



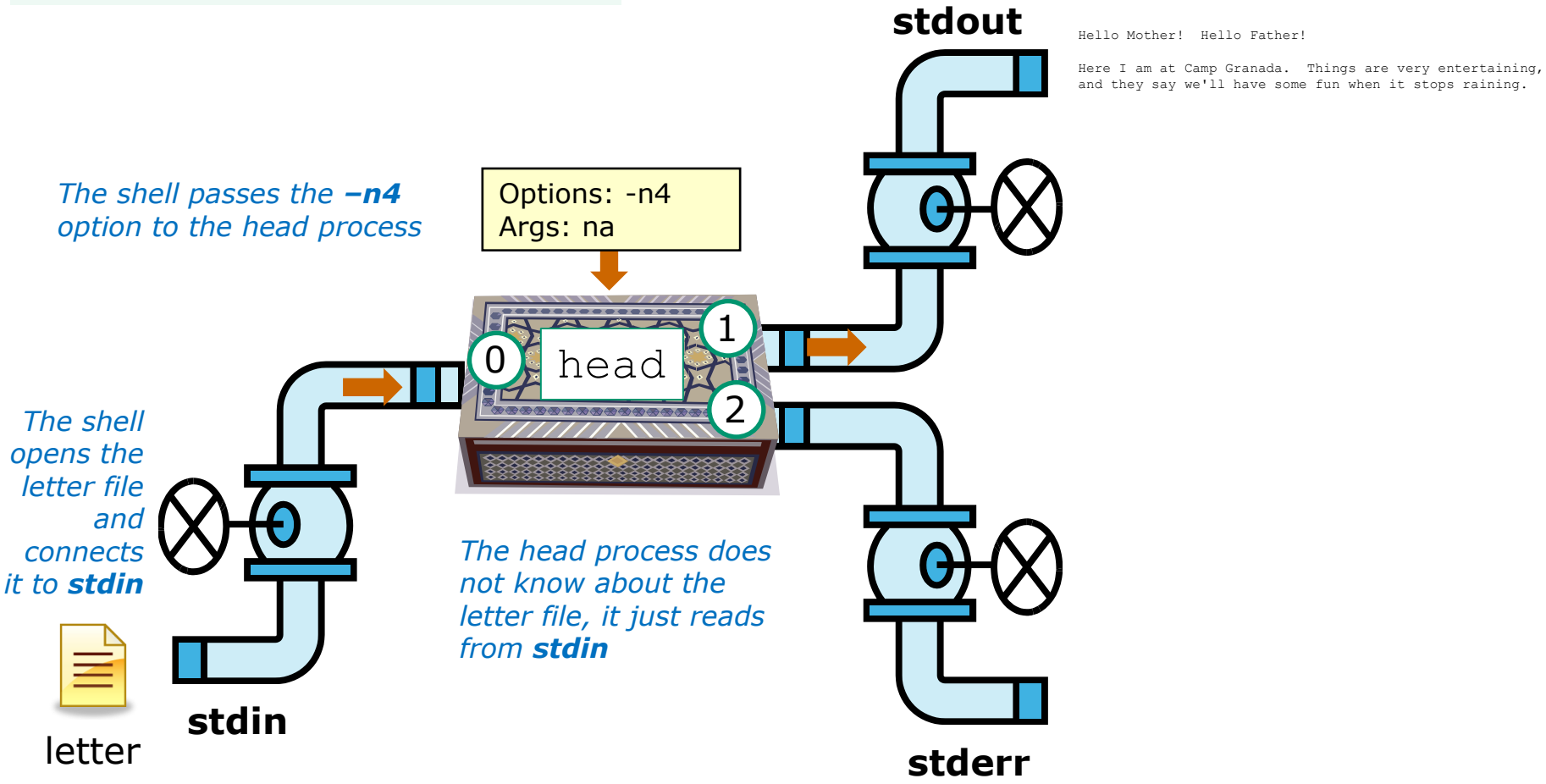

```
Hello Mother!  Hello Father!

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

head opens and reads the letter file

head –n4 < letter

option *redirection*

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



Test your understanding of how the shell and command work as a team

Given: There is no file named *bogus*, associate each command on the left with an error message on the right

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`

Test your knowledge

Given: There is no file named bogus, associate each command on the left with an error message on the right

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`

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
```

*Its not a good idea to redirect **stdout** and **sderr** to the same file because they write over each other*

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

C Program I/O example

```
[rsimms@opus misc]$ 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);
}
```

This program is available in the depot directory

C Program I/O example

```
[rsimms@opus misc]$ 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));   Write question to stderr
        len = read(0, buffer, 80);              Read name from stdin
        write(1, greeting, sizeof(greeting));   Write greeting to stdout
        write(1, buffer, len);                  Write  name to stdout
}
```

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

C Program I/O example

The make command is used to compile a C program file

```
[rsimms@opus misc]$ make simple
cc      simple.c   -o simple
```

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

C Program I/O example

```
[rsimms@opus misc]$ ./simple
What is your name stranger? Rich
Well I'm very pleased to meet you, Rich
```

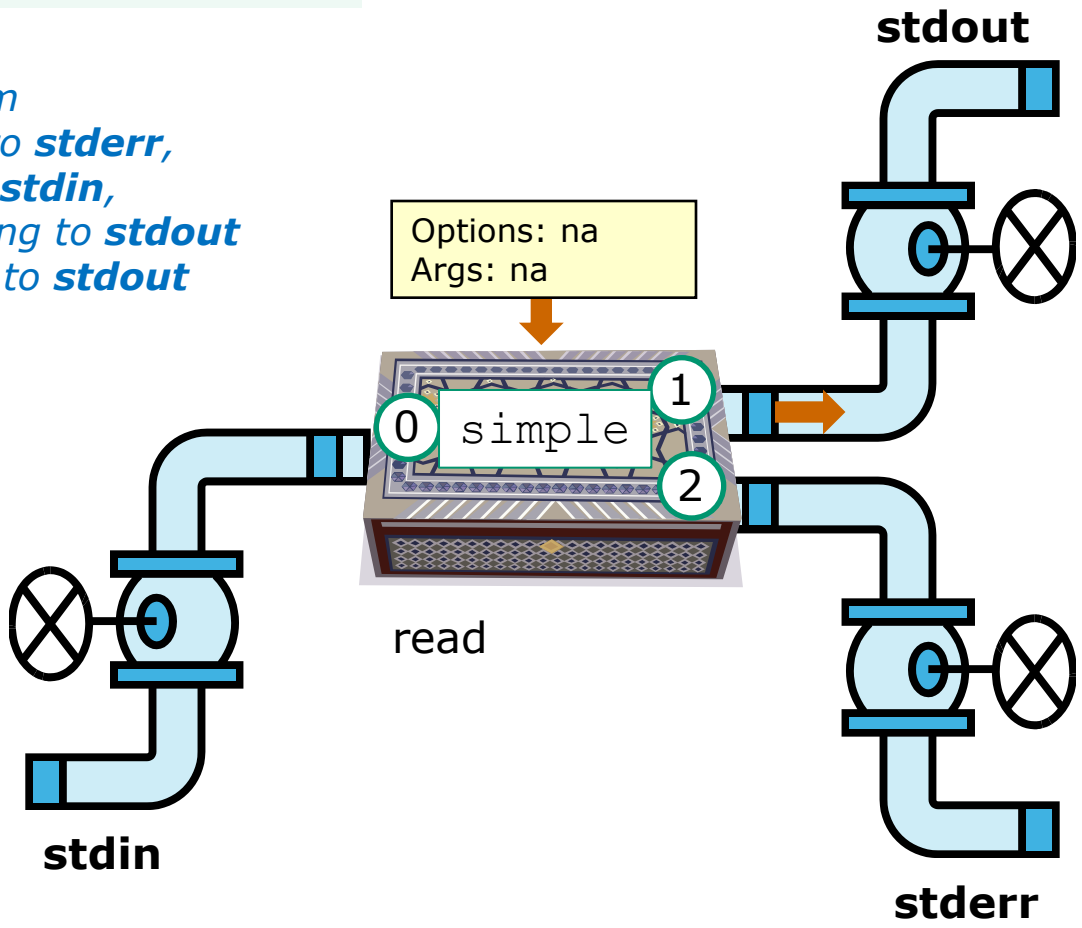
Running the simple program.

*Note I need to preface **simple** with a "**./**" to run it as this directory is not on my path. This is not necessary for CIS 90 students as they already have the . directory in their path.*

C Program I/O example

```
$ ./simple
```

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

```
[rsimms@opus misc]$ ./simple > myfile
What is your name stranger? Rich
[rsimms@opus misc]$ cat myfile
Well I'm very pleased to meet you, Rich
```

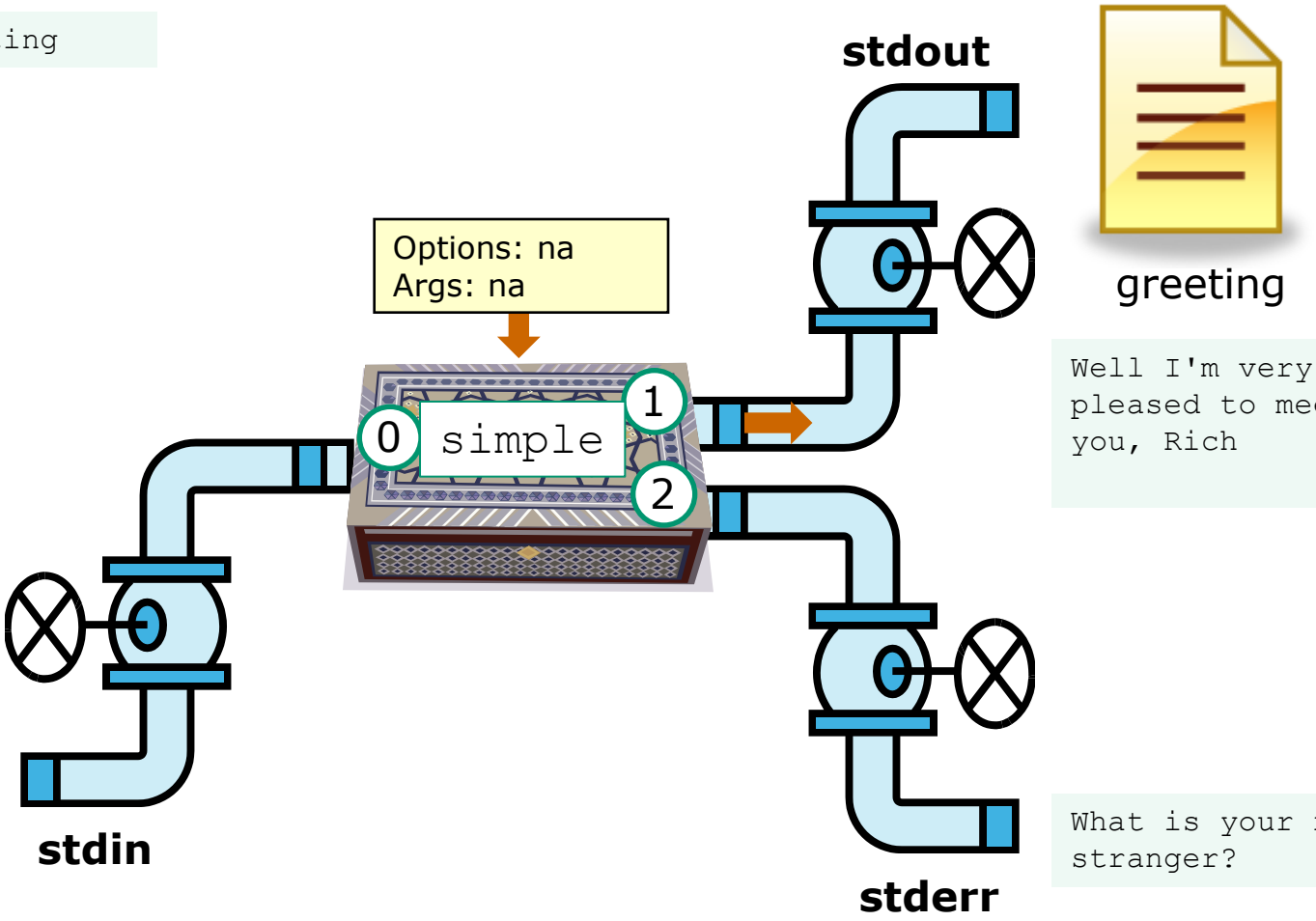
In the second example, output has been redirected to a file named myfile.

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
```




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

```
Rich
```

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

Activity

1. Change to you bin directory
   cd bin

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

3. Compile the program
   make simple

4. Run the program
   simple

More on umask

(shortcut)

Review – applying umask bits

Current umask setting

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

this mask indicates which permissions should NOT be set on the new file or directory

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 block will force the corresponding permission bit to be off in the permissions for the new file or directory

"Subtraction method"

Current umask setting

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

New file - start with 666

```
 666
-002
 664
```

```
/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

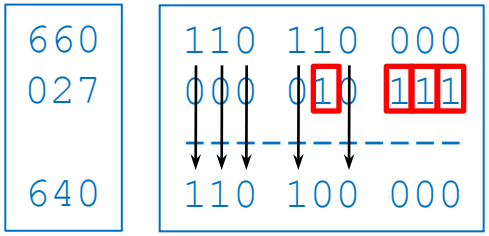
```
 777
-002
 775
```

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

*Shortcut: For new files, when each digit in the **mask** is less than the corresponding digit of the **default permissions** then doing a simple arithmetic subtraction works to determine the new permissions.*

Review - Copying files

```
/home/cis90/simben/lesson9 $ umask 027
/home/cis90/simben/lesson9 $ umask
0027

/home/cis90/simben/lesson9 $ chmod 660 myfile
/home/cis90/simben/lesson9 $ cp myfile myfile.bak
/home/cis90/simben/lesson9 $ ls -l myfile*
-rw-rw---- 1 simben cis90 0 Oct 27 08:02 myfile
-rw-r----- 1 simben cis90 0 Oct 27 08:04 myfile.bak
```



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.*

Housekeeping

Housekeeping

1. Lab 7 due today
2. A check7 script is available
3. Test #2 next week with the Practice Test available now
4. No lab assigned this week (so you can work on the practice test)

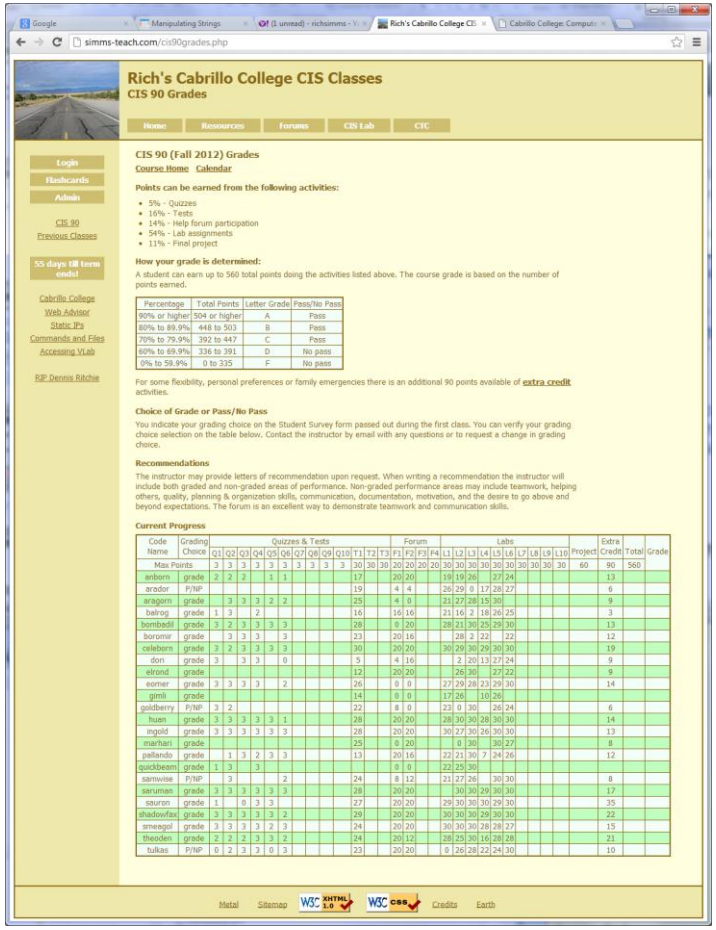
Final Exam

Test #3 (final exam)

- Must be face-to-face (not online using CCC Confer).
- We will be in room 2501 on campus.

	12/12	**Test #3 (the final exam)** **Time** • 1:00PM - 3:50PM in Room 2501 **Materials** • Presentation slides (download) • Test (download)		5 posts Lab X1 Lab X2

Housekeeping



Please monitor your grades on the Grades web page.

Review specific feedback in the *.graded files placed in your home directory.

```
[rsimms@oslab bin]$ date
Sun Oct 21 14:56:54 PDT 2012
```

You can also use Jesse's ***checkgrades*** *script on Opus and provide your code name as an argument.*

If you feel you are not where you want to be then contact me to help you make a development plan.

```
[rsimms@oslab bin]$ ./tally
anborn: 72% (193 of 268 points)
arador: 59% (160 of 268 points)
aragorn: 64% (172 of 268 points)
balrog: 61% (165 of 268 points)
bombadil: 89% (241 of 268 points)
boromir: 58% (157 of 268 points)
celeborn: 105% (284 of 268 points)
dori: 48% (129 of 268 points)
elrond: 61% (166 of 268 points)
eomer: 82% (220 of 268 points)
gimli: 34% (93 of 268 points)
goldberry: 53% (144 of 268 points)
huan: 102% (274 of 268 points)
ingold: 101% (272 of 268 points)
marhari: 52% (140 of 268 points)
pallando: 75% (203 of 268 points)
quickbeam: 31% (84 of 268 points)
samwise: 71% (191 of 268 points)
saruman: 94% (252 of 268 points)
sauron: 107% (287 of 268 points)
shadowfax: 107% (287 of 268 points)
smeagol: 100% (269 of 268 points)
theoden: 91% (246 of 268 points)
tulkas: 79% (214 of 268 points)
```

Bi-annual Campus Climate Student Survey

https://www.surveymonkey.com/s/StudentCampusClimateSurvey2012

This survey will take approximately 15 minutes for students to complete online. If you'd like students to get credit – or extra credit - for completing the survey, Judy will provide names/sections of respondents to you at the end of October. It is otherwise considered optional and voluntary, as there is no "captive audience" online, as we have in classrooms, but it is exceedingly important that we get a good response rate of the student body, overall.

Three points extra credit if I get your name (not your survey answers) from Judy at the end of the month.

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

/home/cis90/simben $
```

xargs command

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

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

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 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 | 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 $
```

xargs *to the rescue.* ***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 $
```

*By the way, 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.*

Trick or Treat

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
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

Flashcards

redhat	suse	CentOS	ubuntu
noreva90	davdon90	farsha90	marray90
fyosea90	ellcar90	kanbry90	frocar90
evaand90	libkel90	lyoben90	mescha90
ramgus90	potjos90	mesmic90	verevi90
ramcar90	rawjes90	kenrit90	calsea90
menfid90	hendaj90	wiljac90	zamhum90
Points:	**Points:**	**Points:**	**Points:**

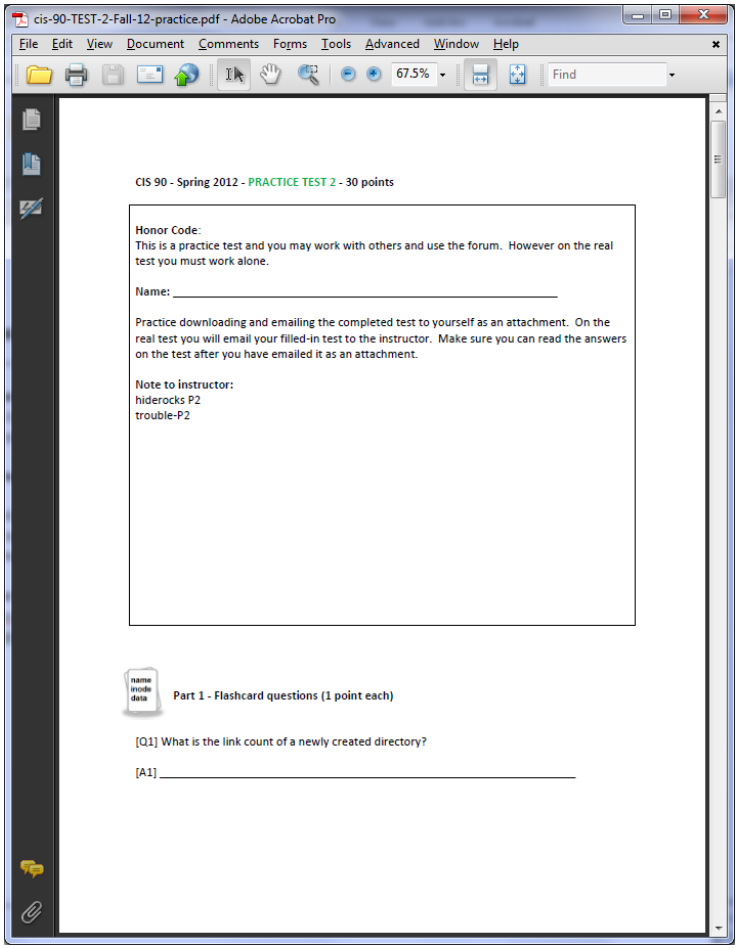
Flashcards
L6=20
L7=15
L8=16

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 10 seconds

Instructor timer:
i=10; while [$i -gt 0]; do clear; banner $i; let i=i-1; sleep 1; done; clear; banner Done

Practice Test



Practice test available

- Work alone or together
- Use the forum to compare answers and approaches to questions

Wrap up

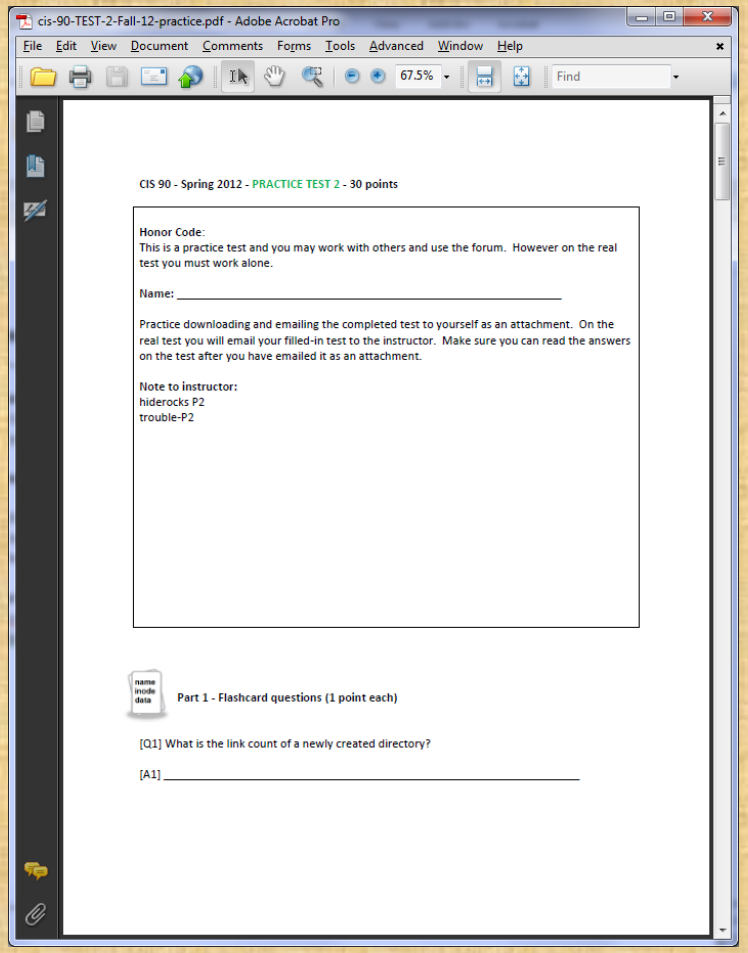
Next Class

No Quiz

Test 2

Cumulative Test (30 points) with focus on Lessons 6-8:

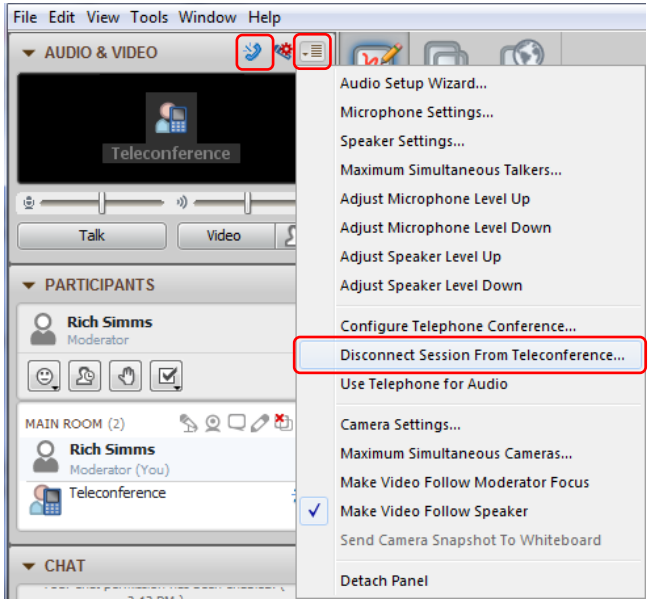
- Recommended preparation:
  - **Work the practice test!**
  - **Collaborate with others on the forum to compare answers**
  - Review Lessons 6-8 slides and Labs 5-7
  - Try doing some or all of Lab X2 (pathnames)
  - Practice with flash cards
  - Scan previous Lessons so you know where to find things if needed

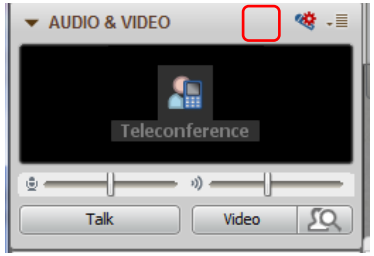


Work the practice test

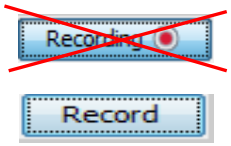
- Collaborate!
- Ask questions!
- You may leave class once you know how to approach and hopefully answer each question

[ ] Disconnect session to Teleconference





[ ] Turn recording off



Backup