



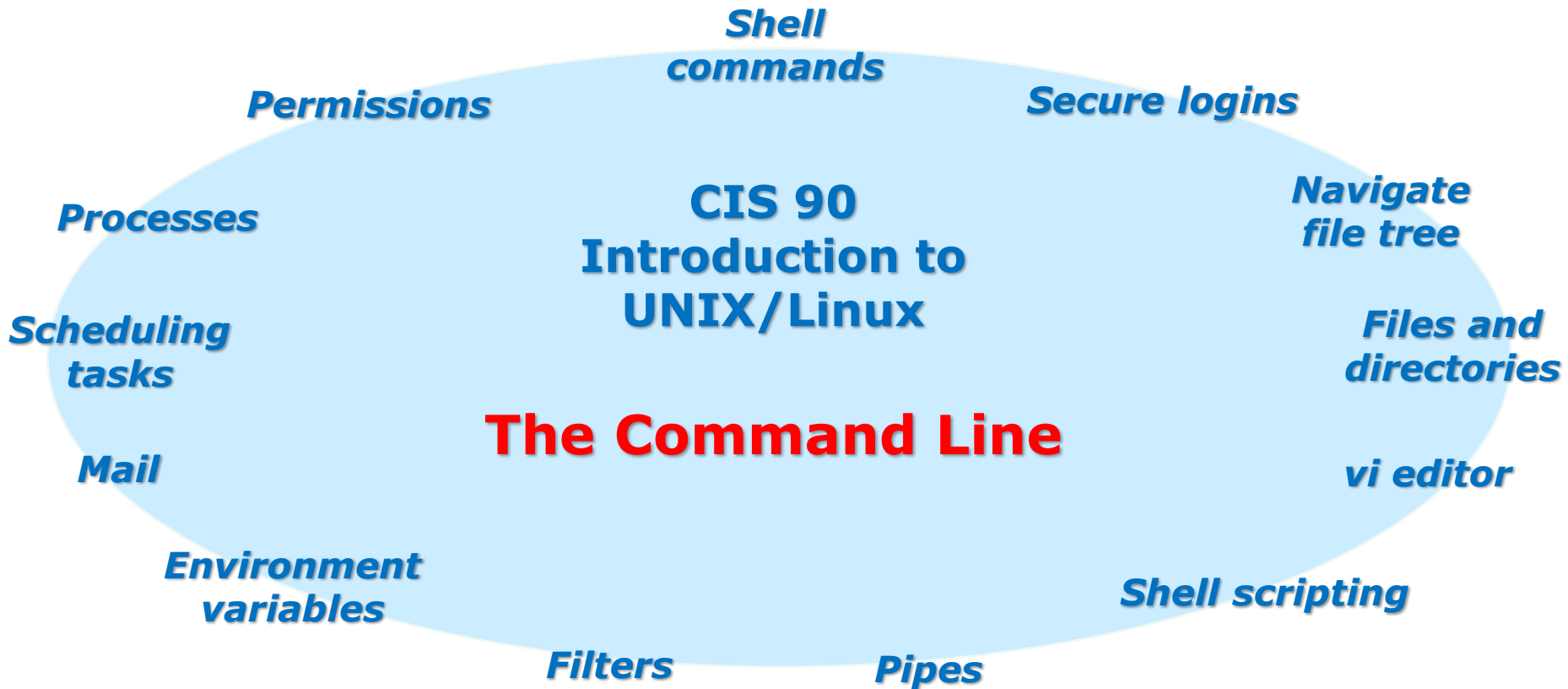
## Rich's lesson module checklist

*Last updated 3/20/2019*

- Zoom recording named and published for previous lesson
- Slides and lab posted
- Print out agenda slide and annotate page numbers
- 1st minute quiz today
- Flash cards
- Calendar page updated
- Schedule lock of turnin directory and submit  
scripts/schedule-submit-locks
- Lab 7 and check7 tested
- Lab X2 updated with kernels and tested
- checkx2 updated (Q1, Q2, Q3, Q9, Q14, Q15)
- 9V backup battery for microphone
- Backup slides, CCC info, handouts on flash drive
- Key card for classroom door

<https://zoom.us>

- Putty, slides, Chrome
- Enable/Disable attendee sharing  
^ > Advanced Sharing Options > Only Host
- Enable/Disable attended annotations  
Share > More > Disable Attendee Sharing



### **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: <https://web.archive.org/web/20140209023942/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. John's site: <http://teacherjohn.com/>
- Jaclyn Kostner for many webinar best practices: e.g. mug shot page.



## Student checklist - Before class starts

The screenshot shows a web browser window with the URL [simms-teach.com/cis90calendar.php](http://simms-teach.com/cis90calendar.php). The page title is "Rich's Cabrillo College CIS Classes CIS 90 Calendar". The main content area is titled "CIS 90 (Fall 2014) Calendar" and includes a "Calendar" link. A table lists lessons, with Lesson 8 highlighted. The details for Lesson 8 are as follows:

Lesson	Date	Topics	Link
8	9/2	<p><b>Class and Linux Overview</b></p> <ul style="list-style-type: none"> <li>Understand how the course will work</li> <li>High-level overview of computers, operating systems, and virtual machines</li> <li>Overview of LINUX/Linux market and architecture</li> <li>Using SSH for remote network exits</li> <li>Using terminals and the command line</li> </ul> <p><b>Materials</b></p> <p><a href="#">Presentation slides (download)</a></p> <p><b>Supplemental</b></p> <ul style="list-style-type: none"> <li>PowerPoint: Logging into Opus (download)</li> </ul> <p><b>Assignments</b></p> <ul style="list-style-type: none"> <li>Student Survey</li> <li>Lab 1</li> </ul> <p><b>CCS Center</b></p> <p><a href="#">Enter virtual classroom</a></p>	<p>2.4.5 p163-172 p164-172 (pdf)</p>

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 ConferZoom.
7. Log into Opus-II with Putty or ssh command.



# Student checklist - Before class starts

Google

ConferZoom

Downloaded PDF of Lesson Slides. I like Foxit Reader so I can take notes using annotations.

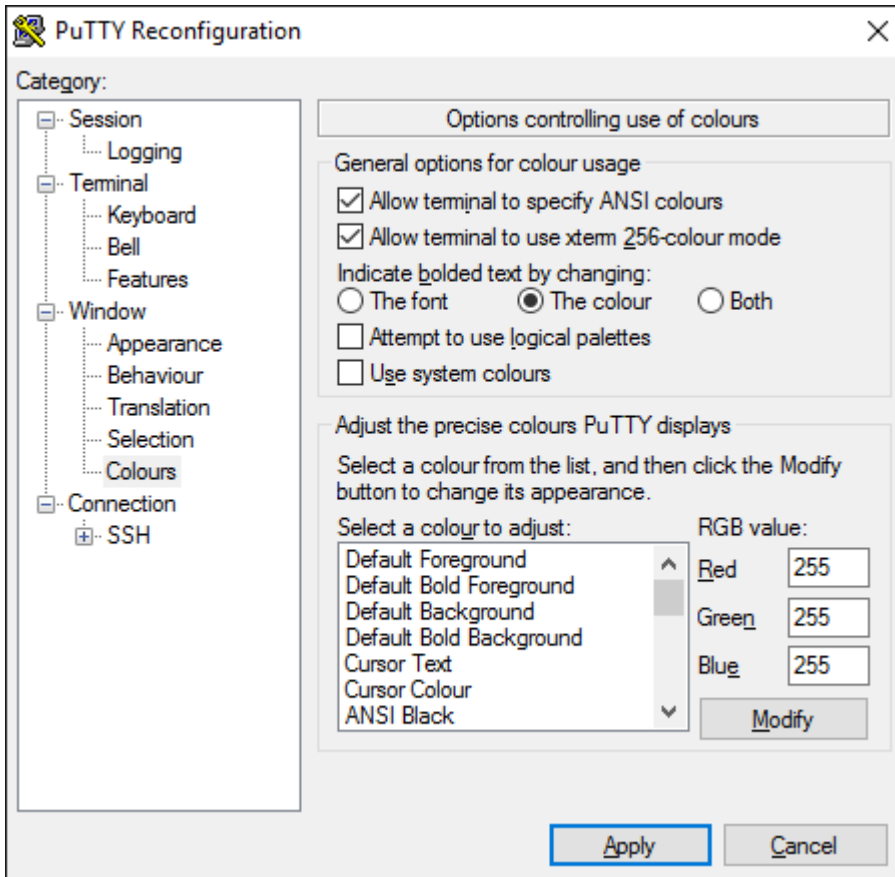
The screenshot shows a Zoom meeting interface with several windows open. The main window displays a login page for 'Rich's Cabrillo College CIS 90' with the text 'Get into the car' overlaid. Other windows include the Google search page, the CIS 90 website's 'CIS 90 Calendar' page, and a PDF of lesson slides titled 'CIS 90 - Lesson 1' with the subtitle '90 System Playground'. The Zoom meeting controls at the bottom show 'Unmute', 'Start Video', 'Invite', 'Participants', 'Share Screen', 'Chat', 'Record', and 'Leave Meeting'.

CIS 90 website Calendar page

One or more login sessions to Opus-II



## Rich's ConferZoom checklist - Putty Colors



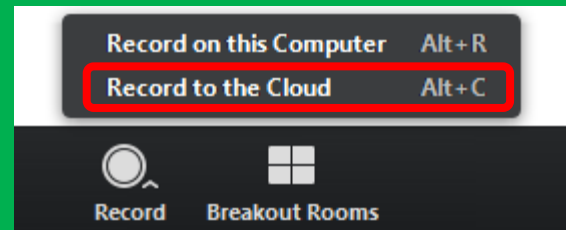
### Putty Colors

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

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



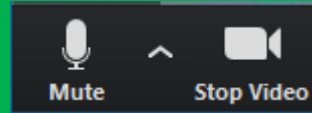
# Start



# Start Recording

Audio Check





Start Recording

# Audio & video Check



Instructor: **Rich Simms**  
Dial-in: **669-900-6833 (toll)**  
Meeting ID: **426 283 384**



Nick



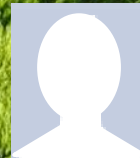
Ryan



Erik



Matt



David



Jon



Cheryl



Wais



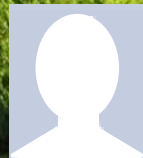
Tanisha



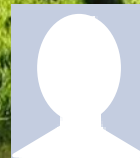
Daniel



Ohunayo



Sequoia



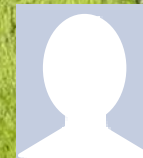
Scott



Lucky



Cole



Shane



Jim



Joseph



Mark



Adina

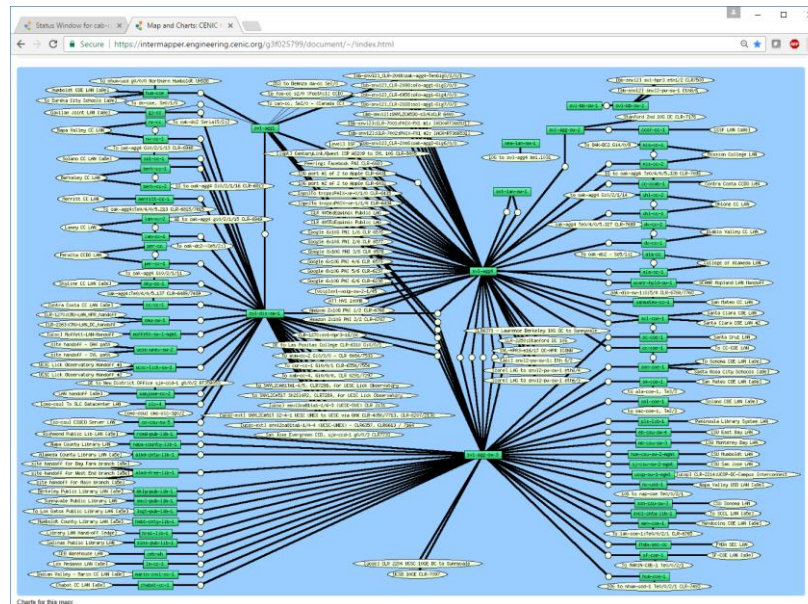


Evie



Cody

# Network Check



[https://intermapper.engineering.cenic.org/g3f025799/  
document/~!/index.html](https://intermapper.engineering.cenic.org/g3f025799/document/~!/index.html)

## First Minute Quiz

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

Use CCC Confer White Board

**email answers to: [risimms@cabrillo.edu](mailto:risimms@cabrillo.edu)**

**(answers must be emailed within the first few minutes of class for credit)** 12

# Input/Output Processing

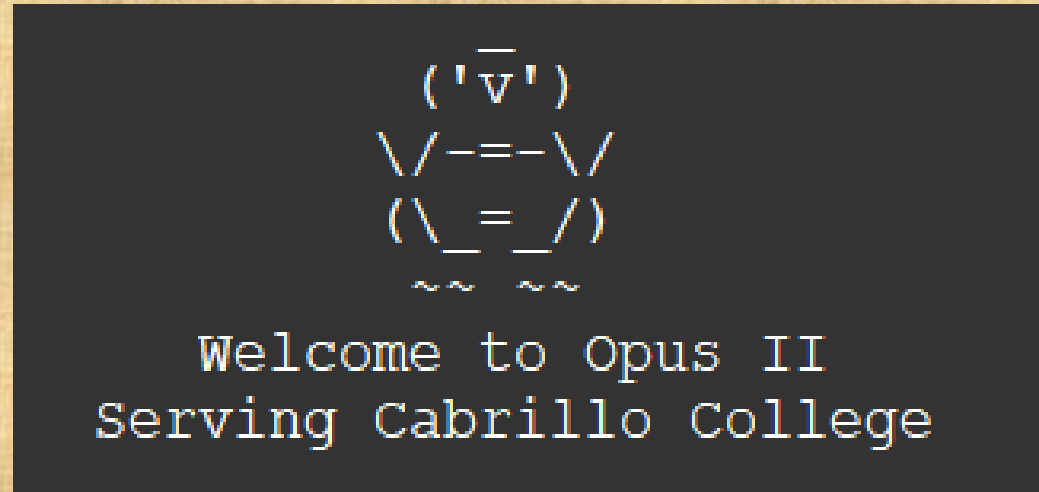
## Objectives

- Identify the three open file descriptors an executing program is given when started.
- Be able to redirect input from files and output to files
- Define the terms pipe, filter, and tee
- Use pipes and tees to combine multiple commands
- Know how to use the following useful UNIX commands:
  - ❖ find
  - ❖ grep
  - ❖ wc
  - ❖ sort
  - ❖ spell

## Agenda

- Quiz
- Questions
- Warmup
- umask continued
- Housekeeping
- New commands (sort)
- Pretend you are a command (imagination)
- Sort command deep dive (good arg, no args, bad arg)
- Bringing it home (reality)
- File redirection
- The bit bucket
- Pipelines
- find command
- Filter commands (grep, spell, tee, cut)
- Pipeline practice
- Permissions, the rest of the story
- Assignment
- Wrap up

## Class Activity



If you haven't already,  
log into Opus-II

## Class Activity

**Lesson 3**

**Electronic Mail**

- Guest speaker: Denise Moore on OTC (On-The-Job) training programs
- Learn how to use the LINC communication tools: write and /bin/mail
- Overview on and-to and mail

**Materials**

- Presentation slides ([download](#))

**Supplemental**

- Howto #318: Accessing vLab ([download](#))

**Assignment**

- Read/skim Lesson 3 slides

<https://simms-teach.com/cis90calendar.php>

If you haven't already,  
download the lesson slides

## Class Activity

	<ul style="list-style-type: none"><li>• <a href="#">Read/skim Lesson 1 slides</a></li><li>• <a href="#">Student Survey</a></li><li>• <a href="#">Lab 1</a></li></ul>
	<b>ConferZoom</b> <ul style="list-style-type: none"><li>• <a href="#">Enter virtual classroom</a></li><li>• <a href="#">Class archives</a></li></ul>
	<b>Quiz 1</b>
	<b>Commenda</b> <ul style="list-style-type: none"><li>• Understand how the UNIX login operation</li></ul>

<https://simms-teach.com/cis90calendar.php>

If you haven't already, join  
ConferZoom classroom





# Questions

# Questions?

Lesson material?

Labs? Tests?

How this course works?

• Graded work & tests  
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.*



Would you be interested in a second online Sunday afternoon (March 24th) workshop on how to do the Test 1 questions in 30 seconds or less?

Since somebody forgot to record all of the first workshop!

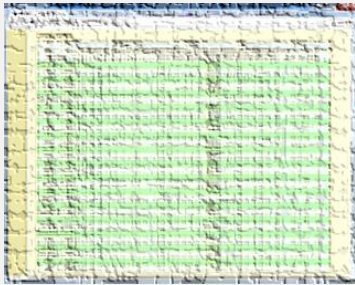
Yes	Maybe	No

*Put a checkmark indicating your interest level above*

## Review your progress in the course

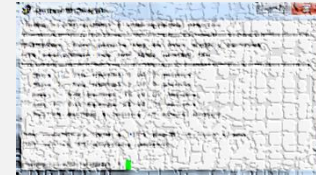
### Check the website Grades page

<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 codename.**
- **Graded labs and tests are in your home directories.**

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:

5 quizzes: 15 points  
 5 labs: 150 points  
 1 test: 30 points  
 1 forum quarter: 20 points  
**Total: 215 points**

## Extra Credit

In lesson slides  
(search for extra credit)

### On the forum

Be sure to monitor the forum as I may post extra credit opportunities without any other notice!

### On some labs

**Extra credit (2 points)**

For a small taste of what you would learn in CIS 191 let's add a new user to your Arya VM. Once added we will see how the new account is represented in `/etc/passwd` and `/etc/shadow`.

1. Log into your Arya VM as the cis90 user. Make sure it's your VM and not someone else's.
2. Install the latest updates:  
`sudo apt-get update`  
`sudo apt-get upgrade`
3. Add a new user account for yourself. You may make whatever username you wish. The example below shows how Benji would make the same username he uses on Opus:  
`sudo useradd -G sudo -c "Benji Simms" -m -s /bin/bash simben90`



### On the website

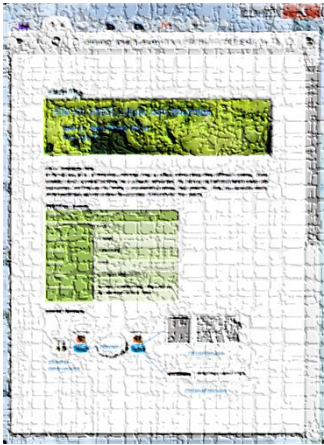
<http://simms-teach.com/cis90grades.php>

For some flexibility, personal preferences or family emergencies there is an additional 90 points available of **extra credit** activities.

<http://simms-teach.com/cis90extracredit.php>

• **Website content review** - The first person to email the instructor pointing out an error or typo on this website will get one point of extra credit for each unique error. The email must specify the specific document or web page, pinpoint the location of the error, and specify what the correction should be. Duplicate errors count as a single point. This does not apply to pre-published material that has been updated but not yet presented in class. (Up to 20 points total)

## Lab Assignments -- Pearls of Wisdom



- Don't wait till the last minute to start.
- Plan for things to go wrong and give yourself time to ask questions and get answers.
- The *slower* you go the *sooner* you will be finished.
- A few minutes reading the forum can save you hour(s).
- Line up materials, references, equipment and software ahead of time.
- It's best if you fully understand each step as you do it. Use Google or refer back to lesson slides to understand the commands you are using.
- Keep a growing cheat sheet of commands and examples.
- Study groups are very productive and beneficial.
- Use the forum to collaborate, ask questions, get clarifications and share tips you learned while doing a lab.
- **Late work is not accepted** so submit what you have for partial credit.

## Getting Help When Stuck on an Assignment

- Google the topic/error message.
- Search the Lesson Slides (they are PDFs) for a relevant example on how to do something.
- Check the forum. Someone else may have run into the same issue and found a way past it. If not start a new topic, explain what you are trying to do and what you have tried so far.
- Talk to a tutor/assistant at the CTC (room 1403) or CIS Lab (STEM Center).
- Come see me during my office or lab hours:

<https://www.cabrillo.edu/salsa/listing.php?staffId=1426>

**I'm in the CTC (room 1403) every Tuesday from 3:30-6:00 pm.**

- Make use of the Open Questions time at the start of every class.
- Make a cheat sheet of commands and examples so you never again get stuck on the same thing!

*CIS Labs always involve some troubleshooting!*

# Help Available! In the CTC and CIS Lab

## Rich's Cabrillo College CIS Classes CIS 90 Calendar

Home

Resources

Forums

**Tutors**

Canvas

CIS Lab  
in STEM Center  
Building 800

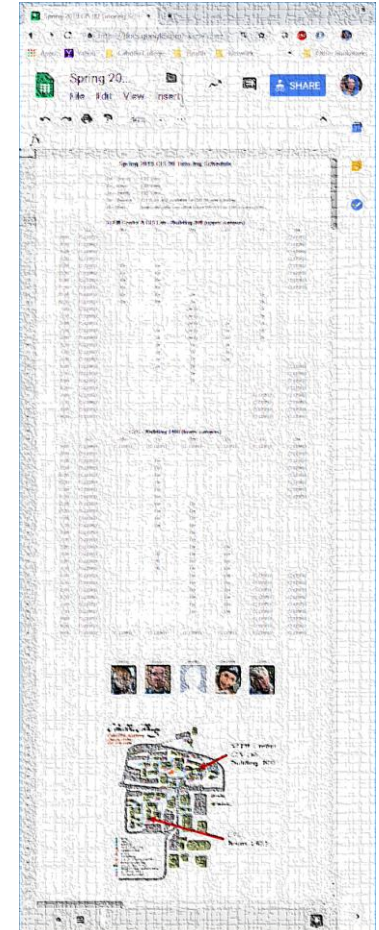
*To see tutor  
schedule, click  
the Tutors link  
on the  
website.*

Cabrillo College  
Cabrillo Gallery  
Library #1002  
831-479-6308



CTC  
Room 1403

*Instructors, tutors  
and equipment are  
available for CIS  
students to work on  
assignments.*





# Help Available! In the CTC and CIS Lab

**Rich's Cabrillo College CIS Classes**  
CIS 90 Calendar

Home Resources Forums **Tutors** Canvas

*To see tutor schedule, click the  
Tutors link on the website.*



*The CIS Lab is in the STEM  
center (Building 800)*



*Room 1403 is in the  
CTC (Building 1400)*



# The slippery slope



- 1) If you didn't submit the last lab ...
- 2) If you were in class and didn't submit the last quiz ...
- 3) If you didn't send me the student survey assigned in Lesson 1 ...
- 4) If you haven't made a forum post in the last quarter of the course ...
- 5) If you had trouble doing the last test ...

*Please contact me by email, see me during my office hours or when I'm in the CTC*

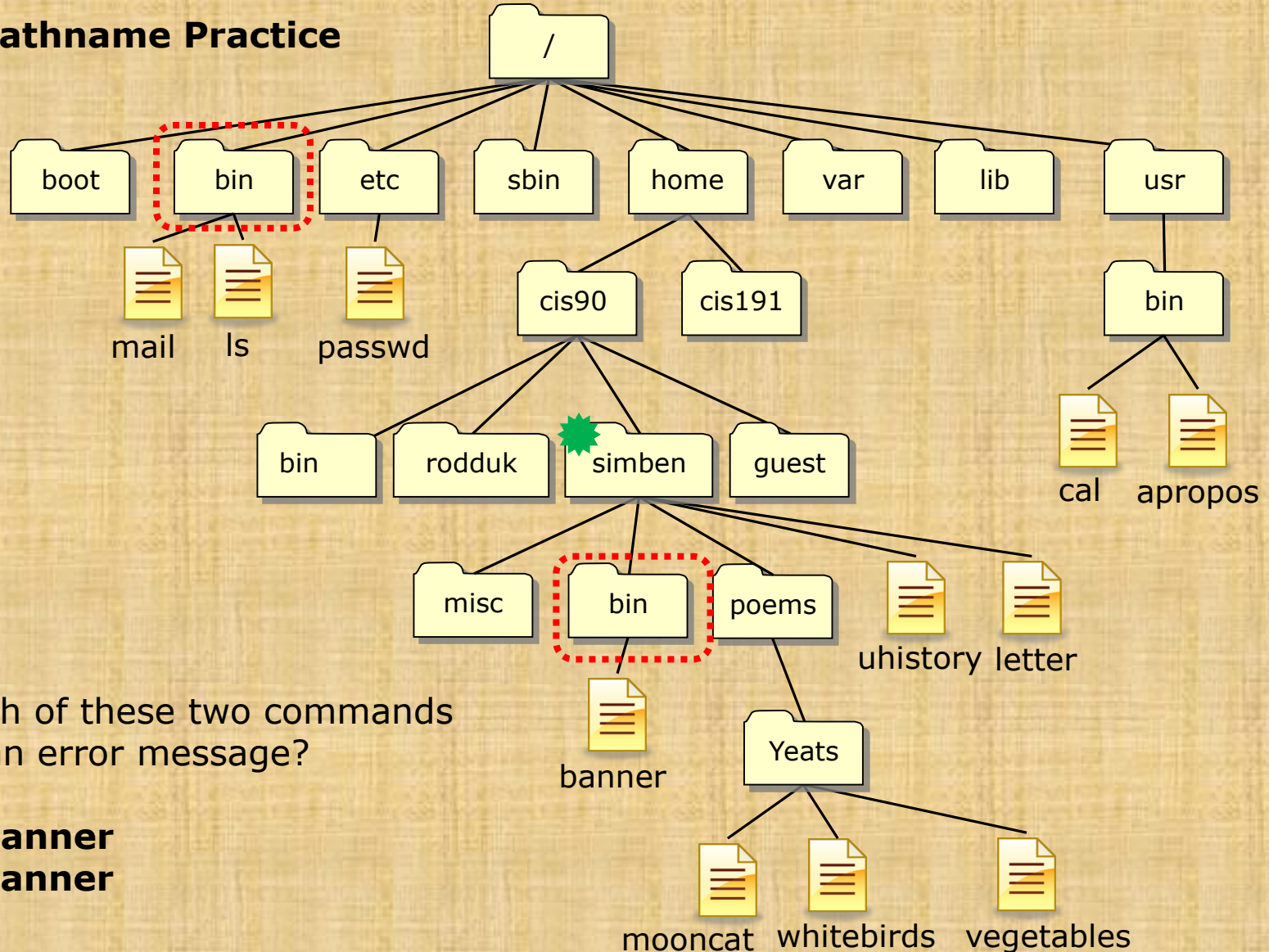
Email: [risimms@cabrillo.edu](mailto:risimms@cabrillo.edu)




# Warmup

pathnames as  
arguments

**File Tree Pathname Practice**

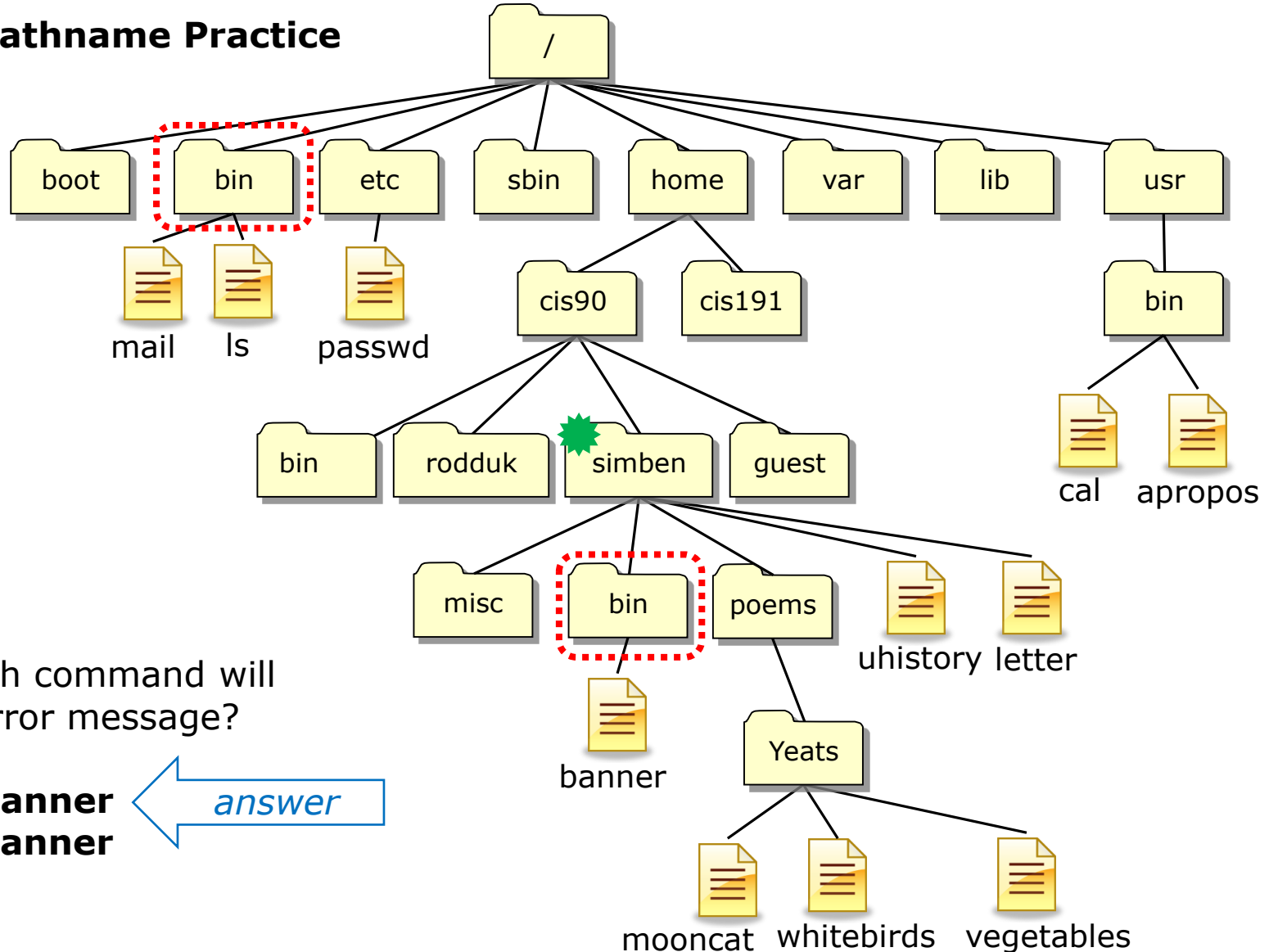



From  which of these two commands will generate an error message?

- touch /bin/banner**
- touch bin/banner**

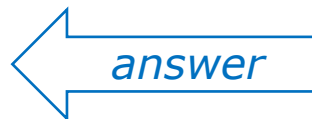
*Write your answer in the chat window*

## File Tree Pathname Practice



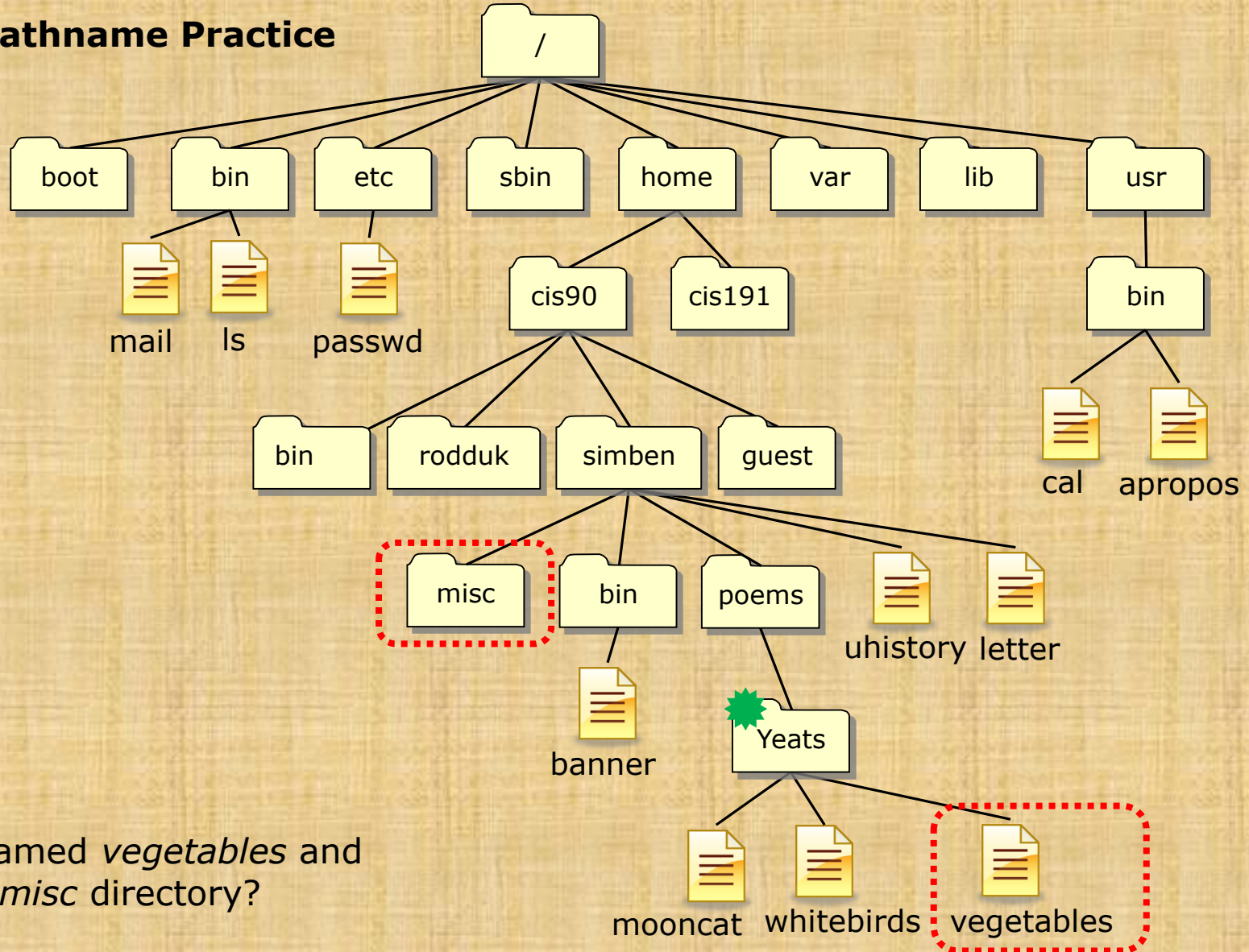
From  which command will generate an error message?

**touch /bin/banner**  
**touch bin/banner**



```
/home/cis90/simben $ touch /bin/banner
touch: cannot touch `/bin/banner': Permission denied
```

## File Tree Pathname Practice

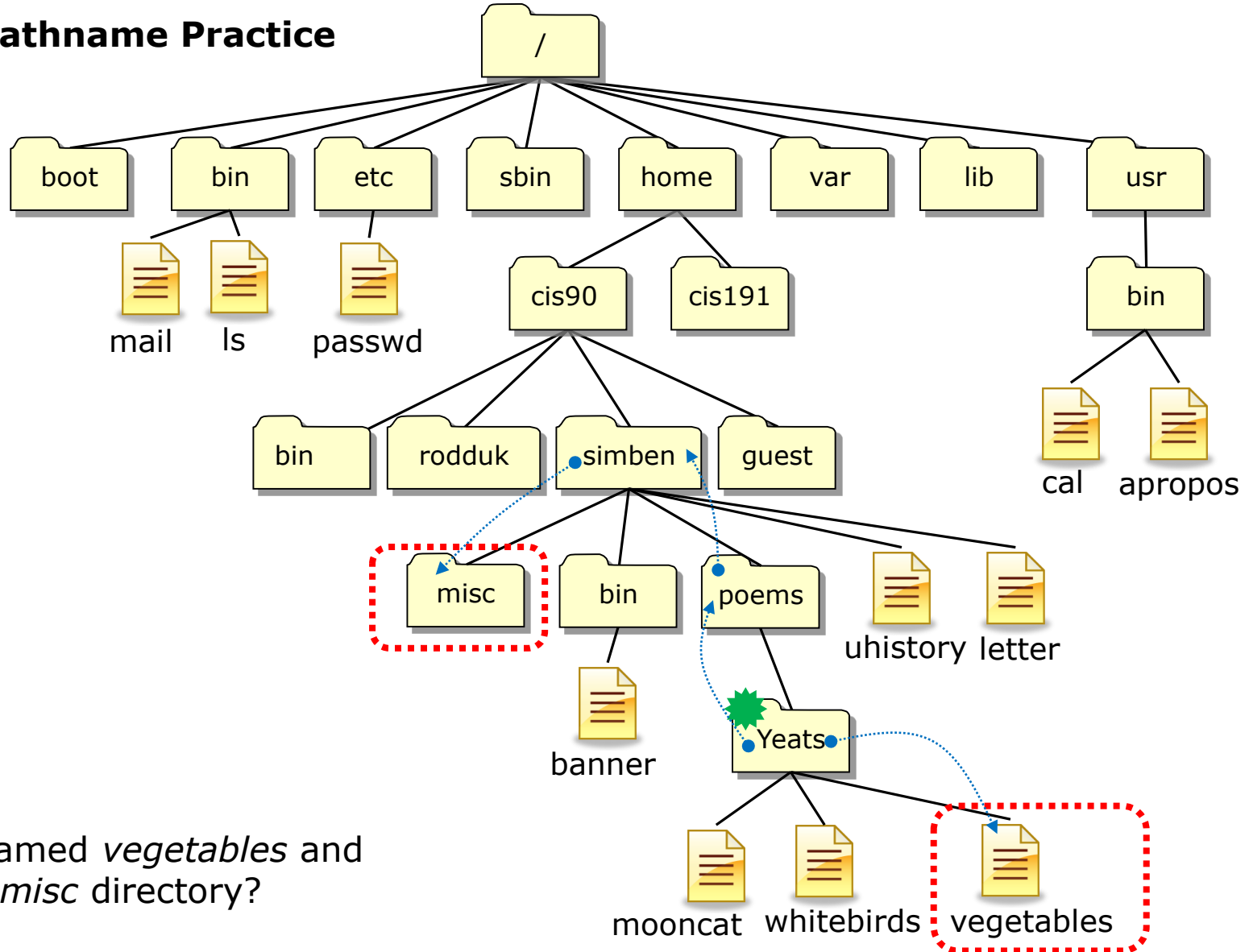


From  how does Benji:

Create a file named *vegetables* and move it to his *misc* directory?

*Write your answer in the chat window*

## File Tree Pathname Practice



From  how does Benji:

Create a file named *vegetables* and move it to his *misc* directory?

```

/home/cis90/simben/poems/Yeats $ touch vegetables
/home/cis90/simben/poems/Yeats $ mv vegetables ../../misc/
    
```

Other answers  
are also  
acceptable

From  how  
does Benji:

Create a file named *vegetables* and  
move it to his *misc* directory?

**touch vegetables**

**mv <file-pathname> <directory-pathname>**

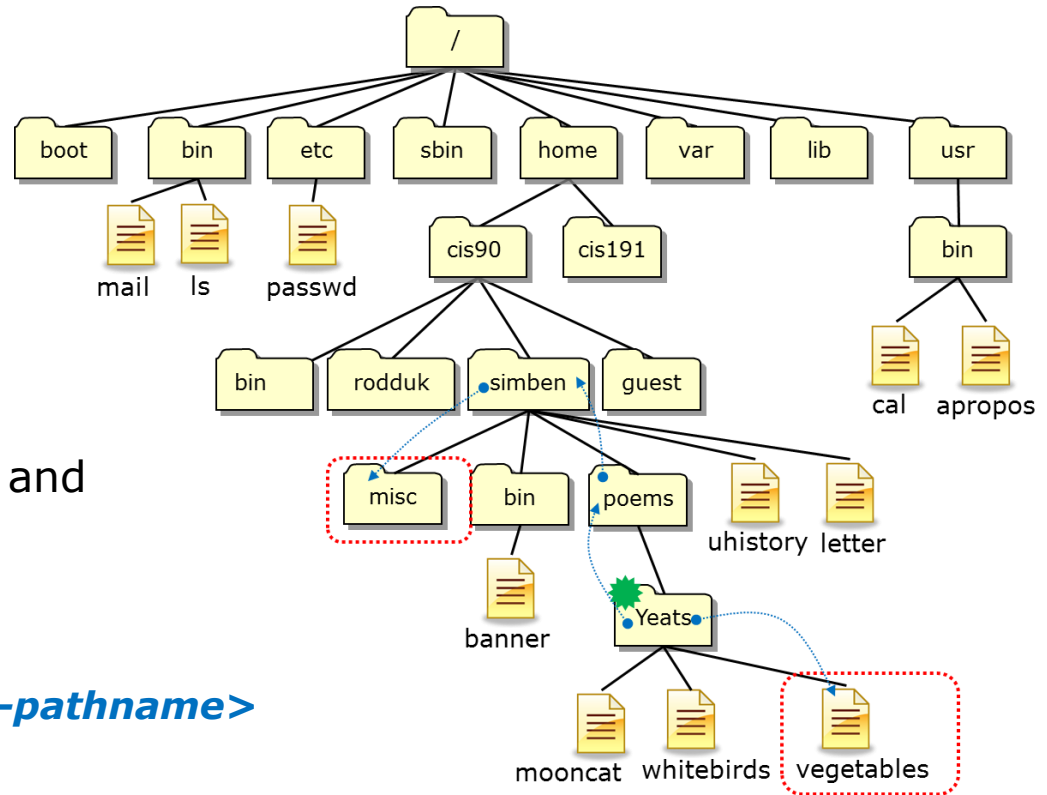
**mv vegetables ../../misc/**

**or mv vegetables /home/cis90/simben/misc/**

**or mv /home/cis90/simben/poems/Yeats/vegetables ../../misc/**

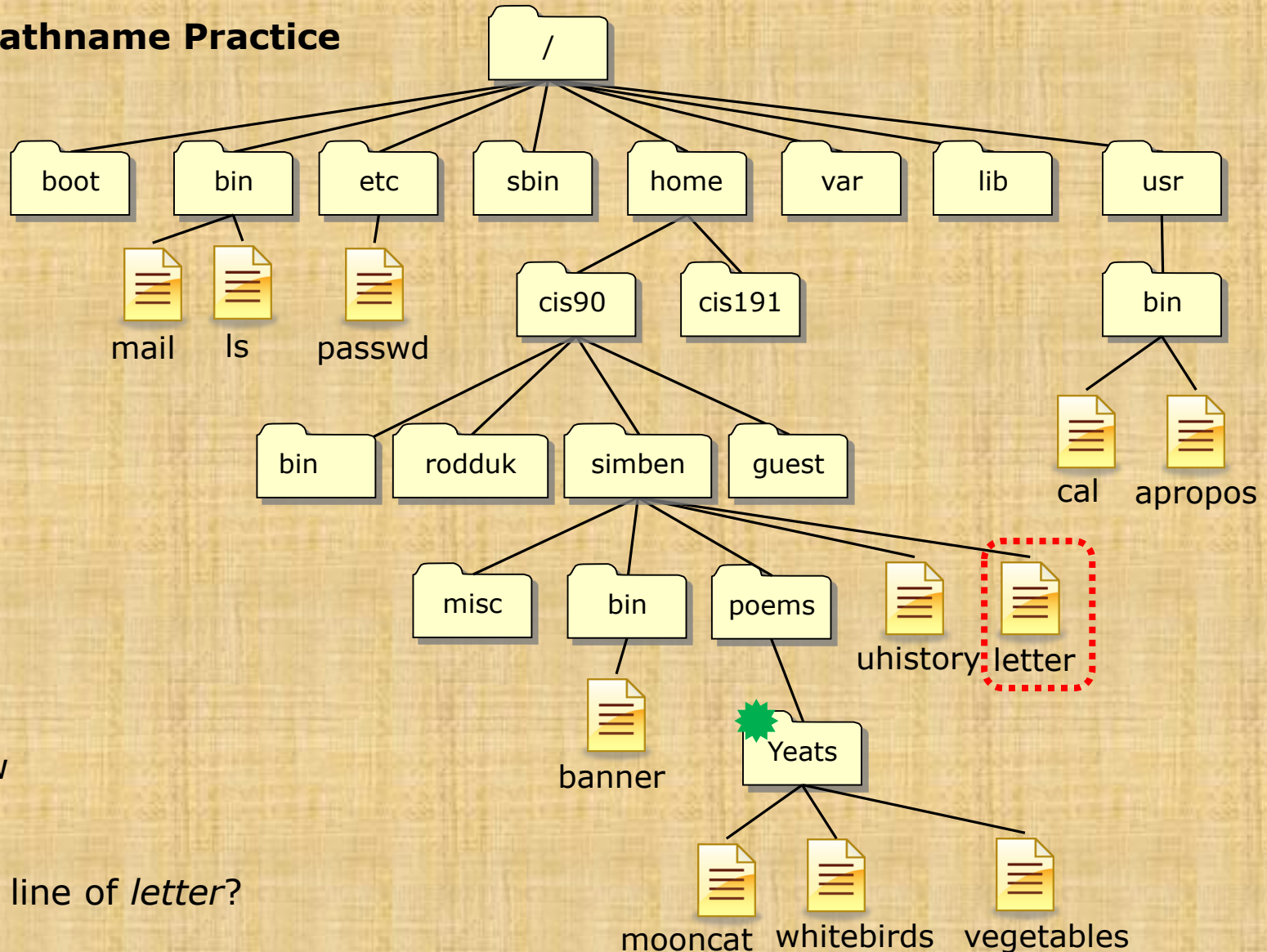
**or mv /home/cis90/simben/poems/Yeats/vegetables /home/cis90/simben/misc/**

**or mv vegetables ~/misc/**





## File Tree Pathname Practice

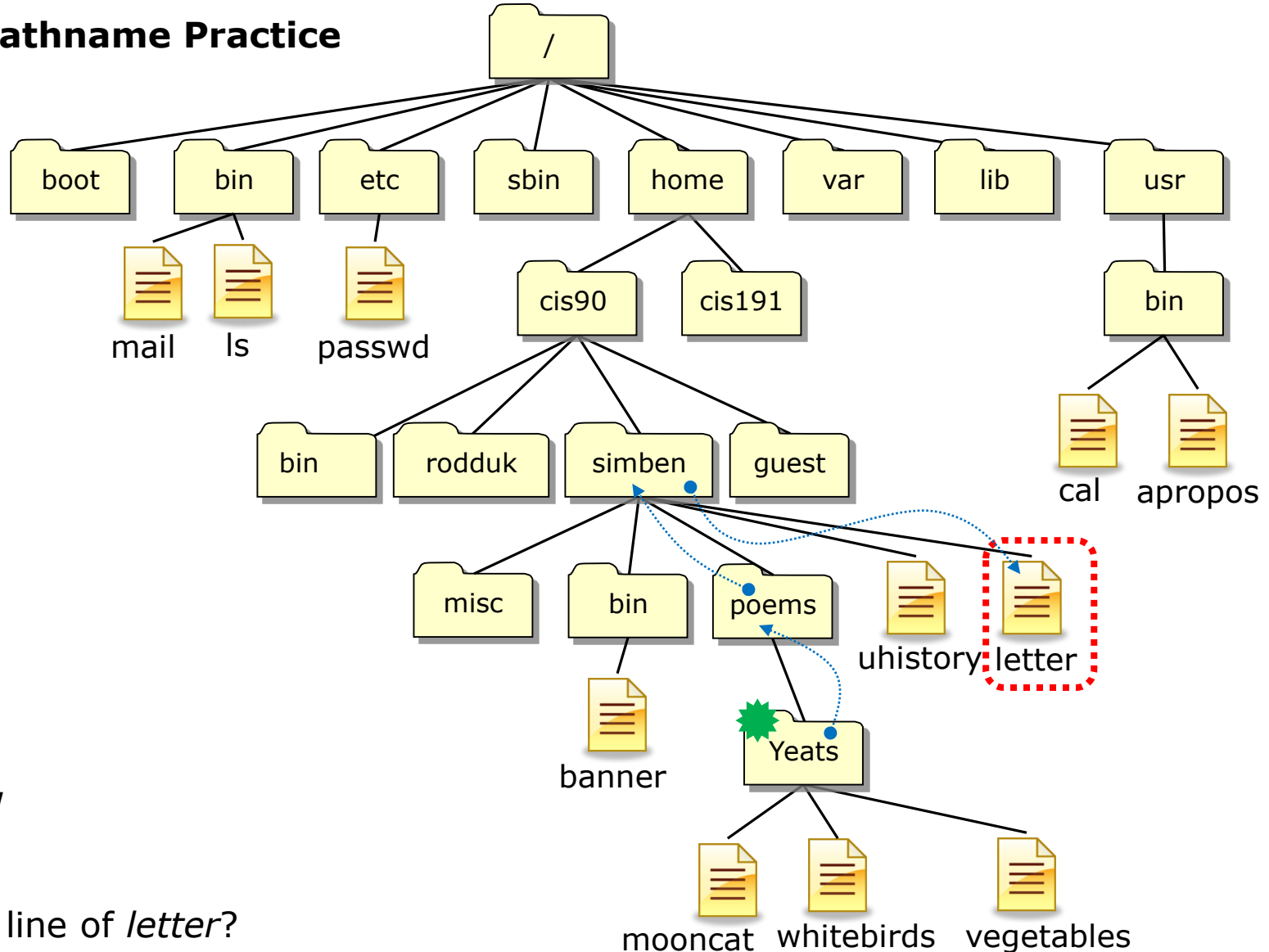


From  how does Benji:

Print the last line of *letter*?

*Write your answer in the chat window*

## File Tree Pathname Practice

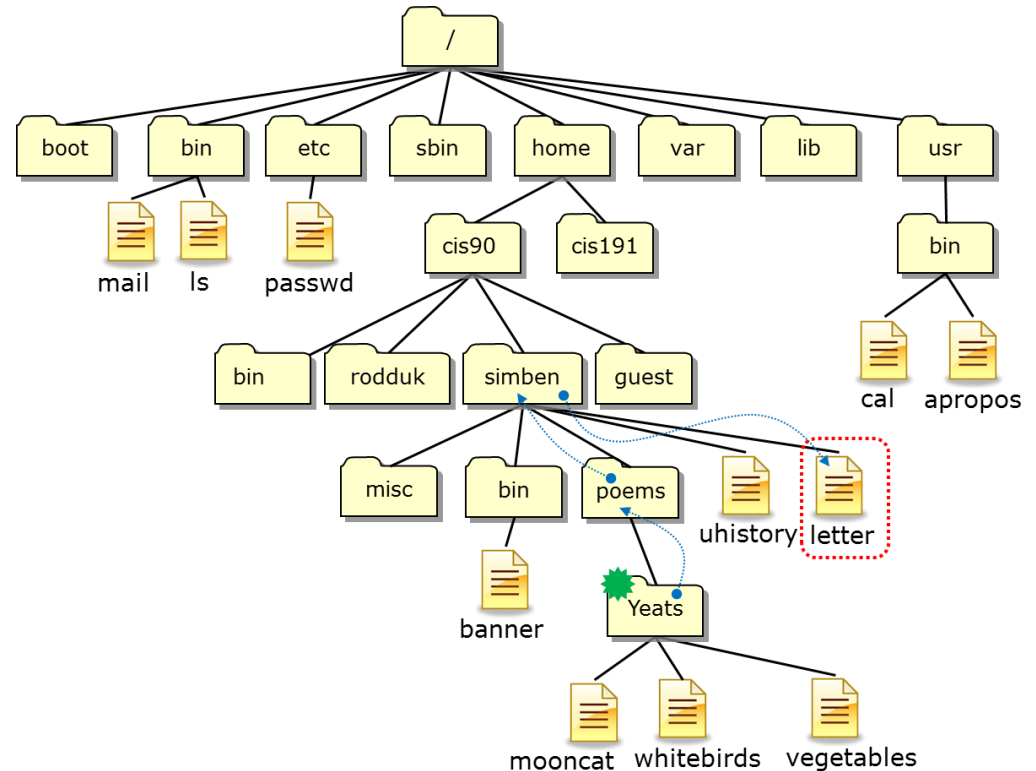


From  how  
does Benji:

Print the last line of *letter*?

`/home/cis90/simben/poems/Yeats $ tail -n1 ../..letter`

Other answers  
are also  
acceptable



From  how  
does Benji:

Print the last line of *letter*?

**tail -n<number> <pathname>**

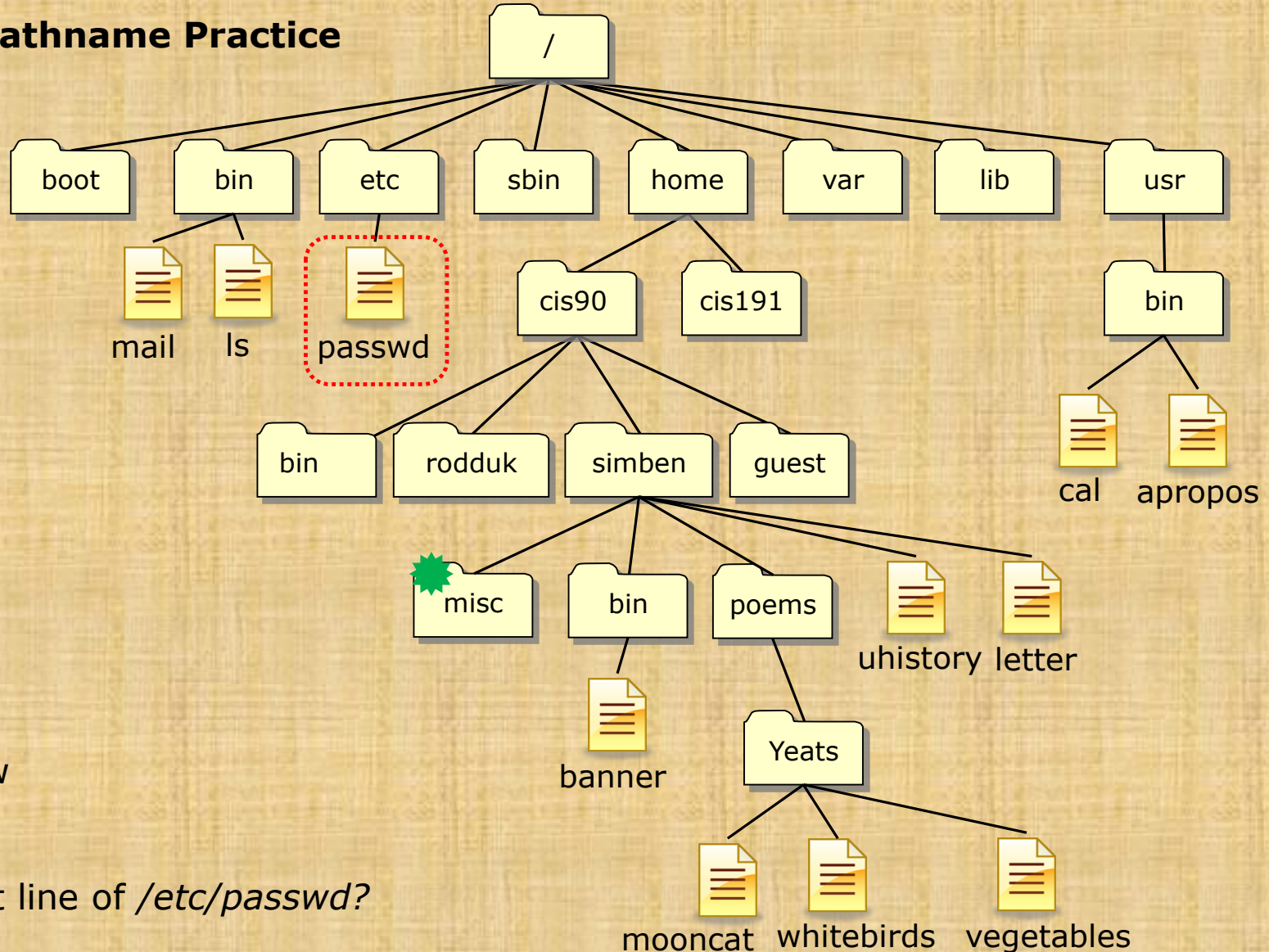
**tail -n1 ../.. /letter**

**or tail -n1 /home/cis90/simben/letter**

**or tail -n1 ~/letter**

*All these answers are correct*

## File Tree Pathname Practice

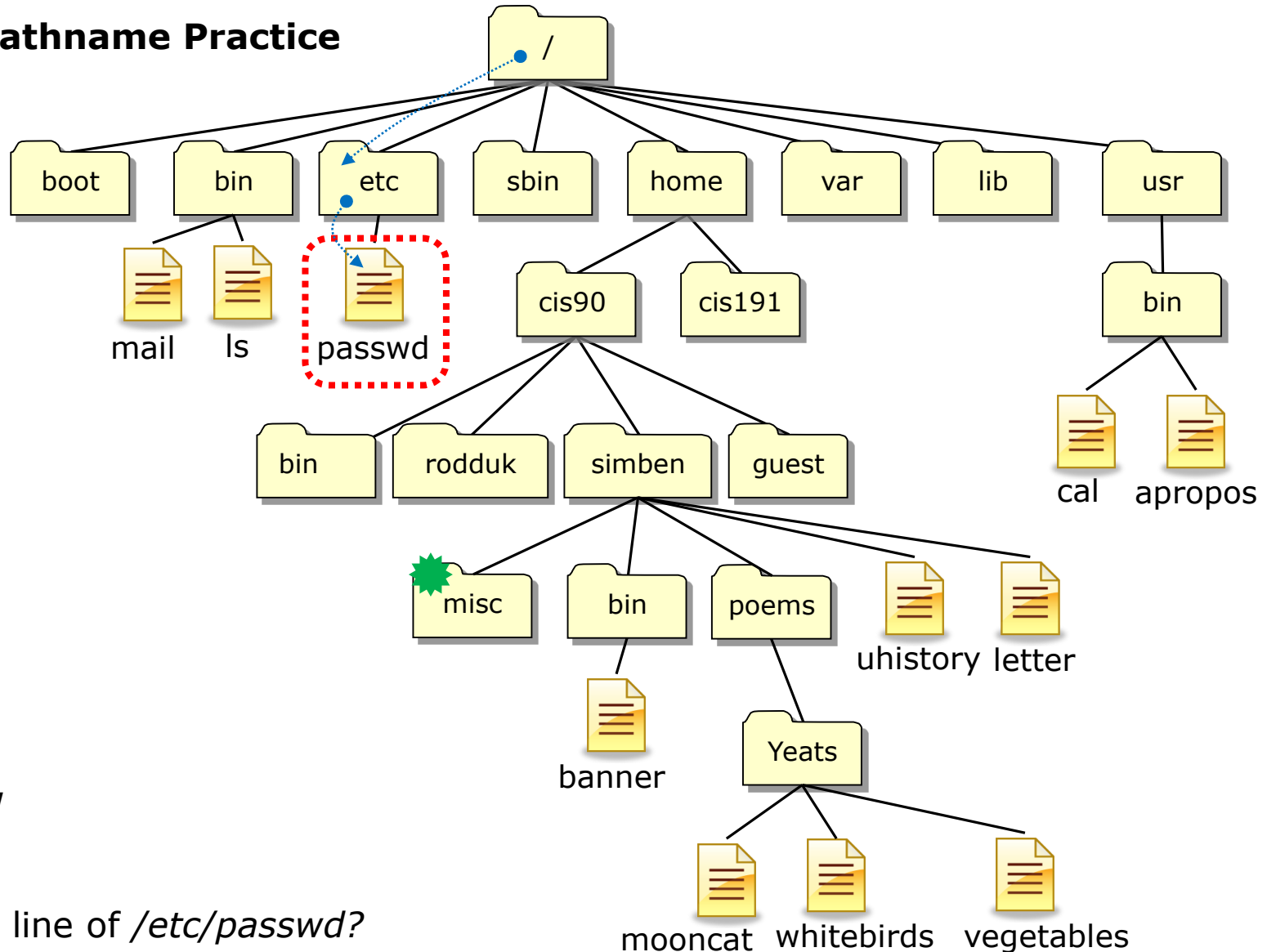


From  how does Benji:

Print the first line of `/etc/passwd`?

*Write your answer in the chat window*

## File Tree Pathname Practice

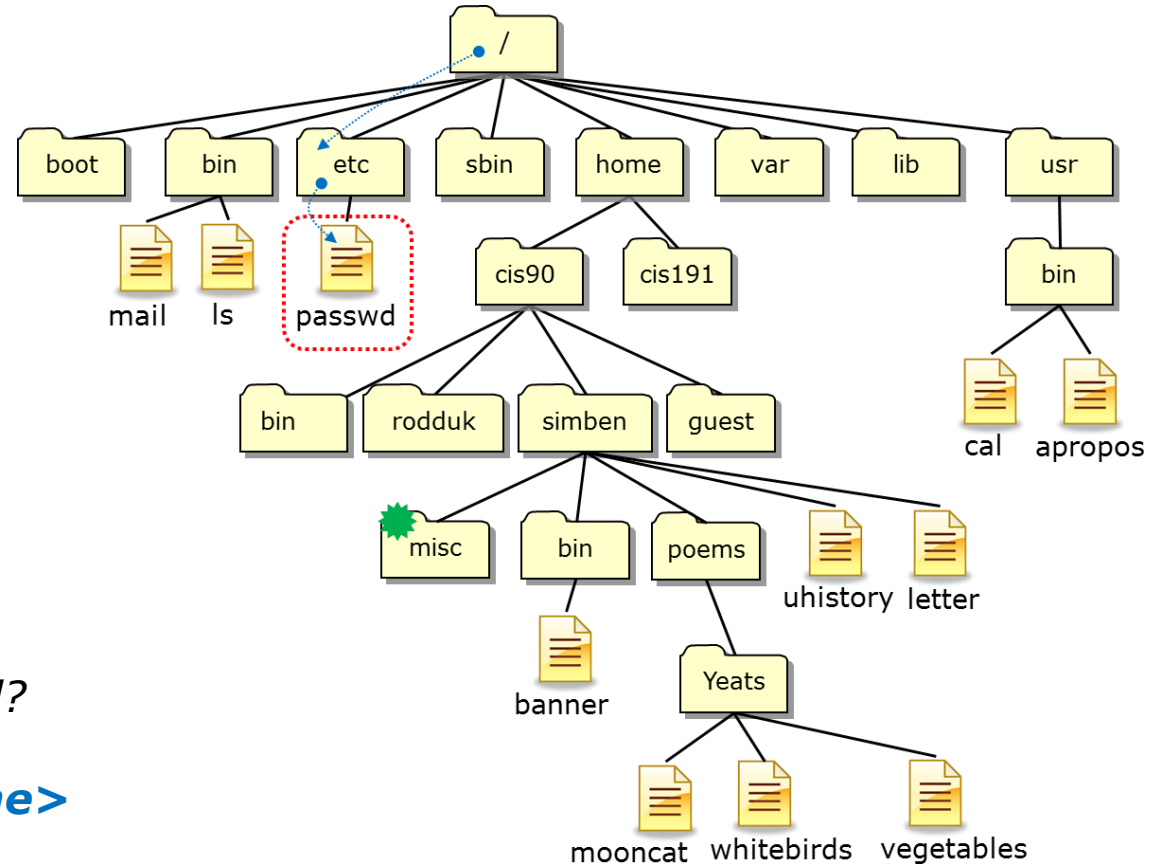


From  how  
does Benji:

Print the first line of `/etc/passwd`?

```
/home/cis90/simben/misc $ head -n1 /etc/passwd
```

*Other answers  
are also  
acceptable*



From  how  
does Benji:

Print the first line of `/etc/passwd`?

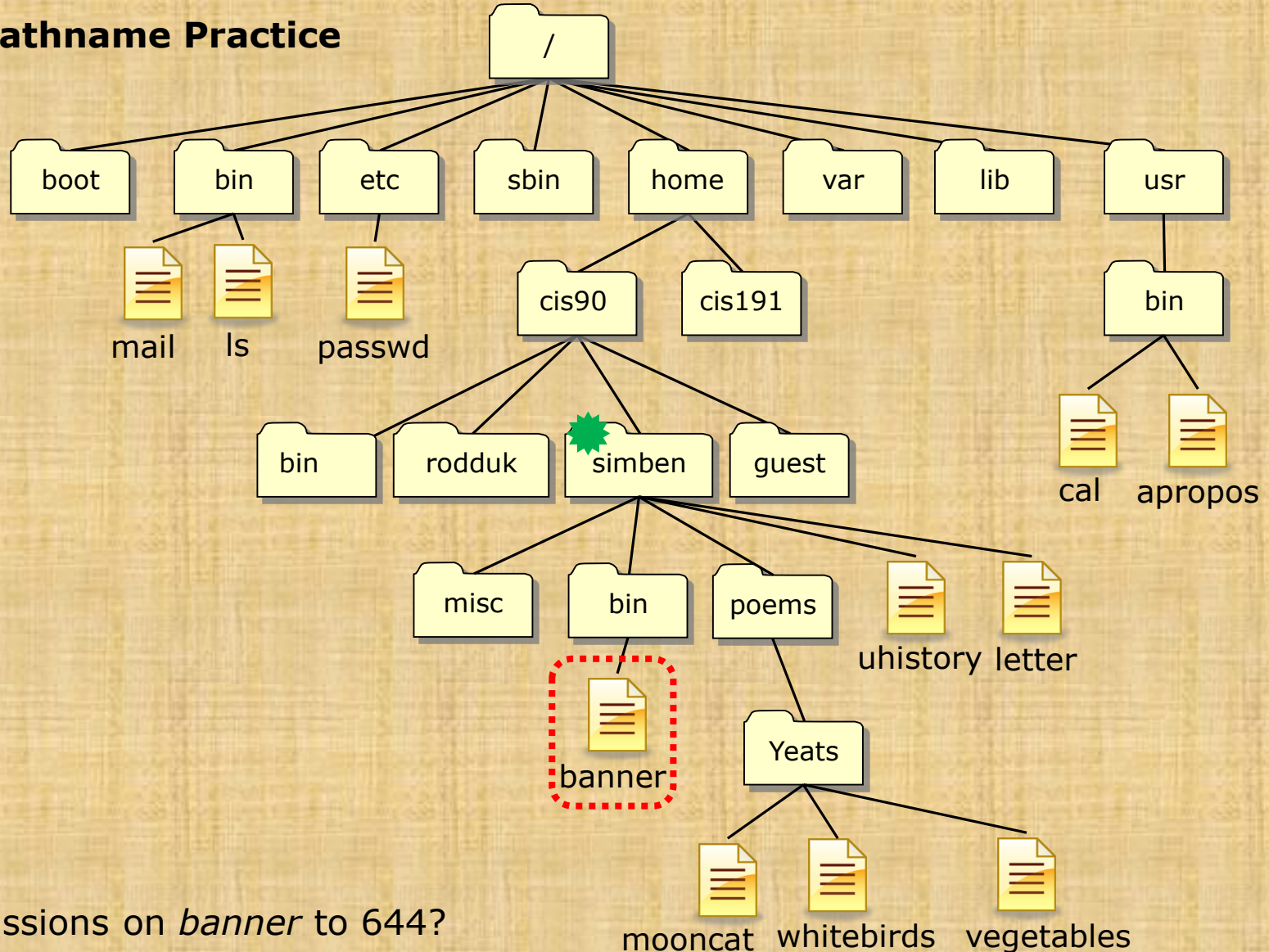
**`head -n<number> <pathname>`**

*or* **`head -n1 /etc/passwd`**

*or* **`head -n1 ../../../../etc/passwd`**

*Both these answers are correct*

File Tree Pathname Practice

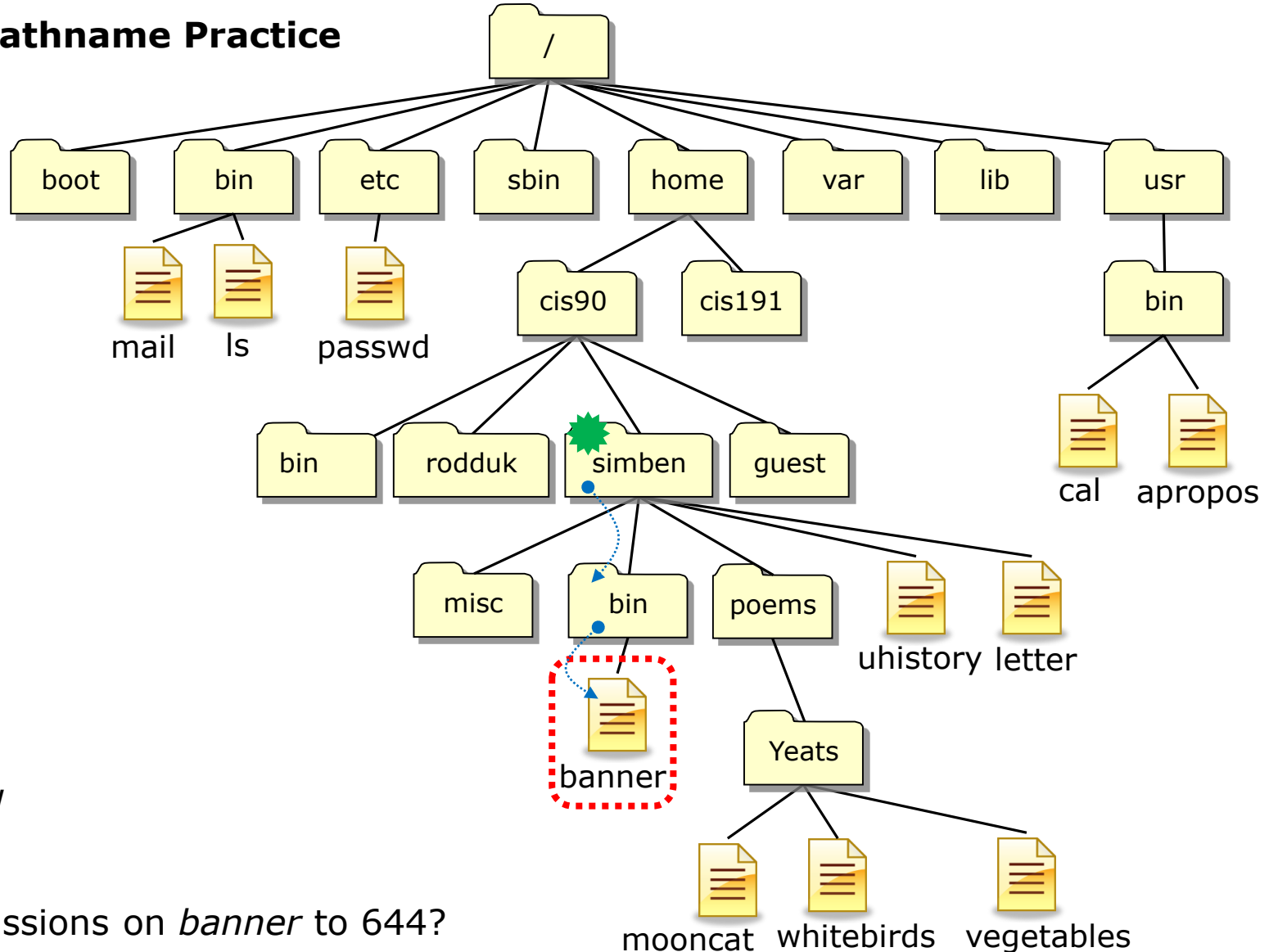


From  how does Benji:

Change permissions on *banner* to 644?

*Write your answer in the chat window*

## File Tree Pathname Practice



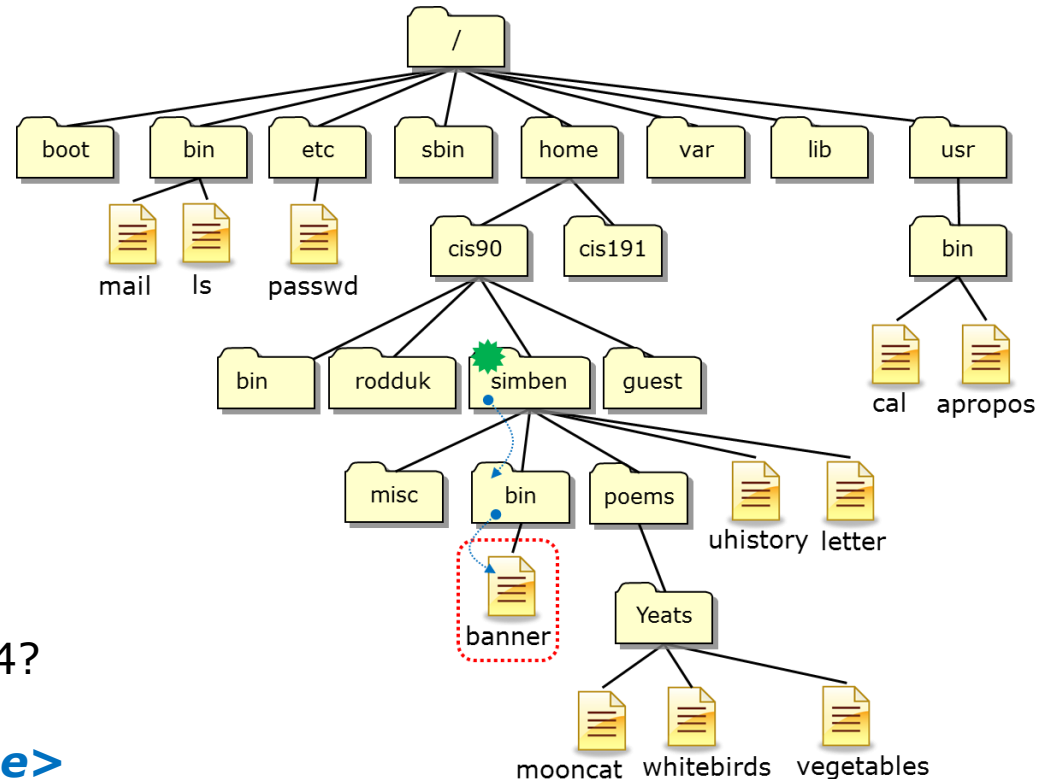
From  how does Benji:

Change permissions on *banner* to 644?

```
/home/cis90/simben $ chmod 644 bin/banner
```



Other answers  
are also  
acceptable



From  how  
does Benji:

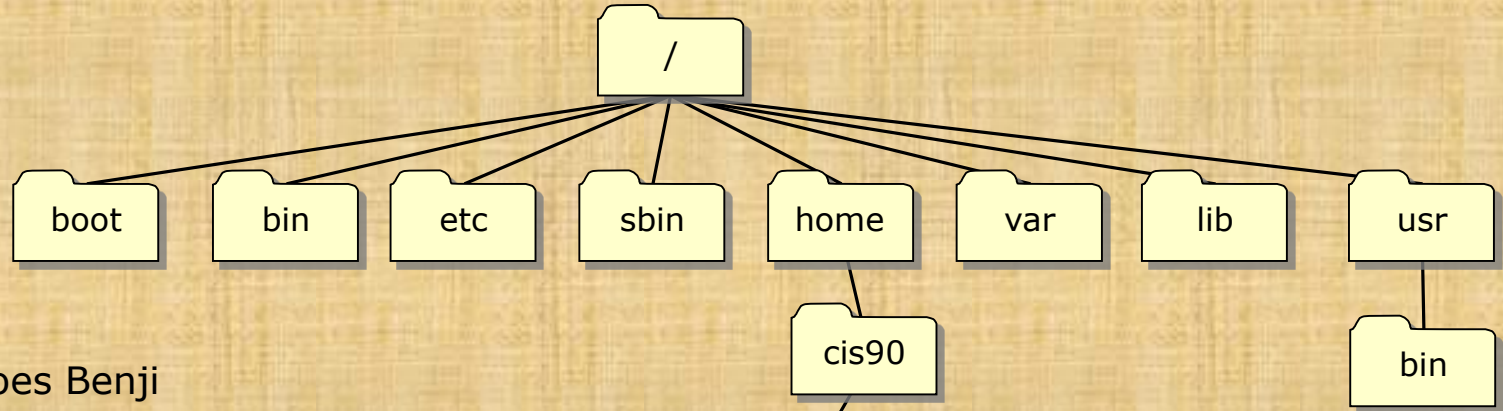
Change permissions on *banner* to 644?

**chmod** *<permissions>* *<pathname>*

or **chmod 644 bin/banner**

or **chmod 644 /home/cis90/simben/bin/banner**

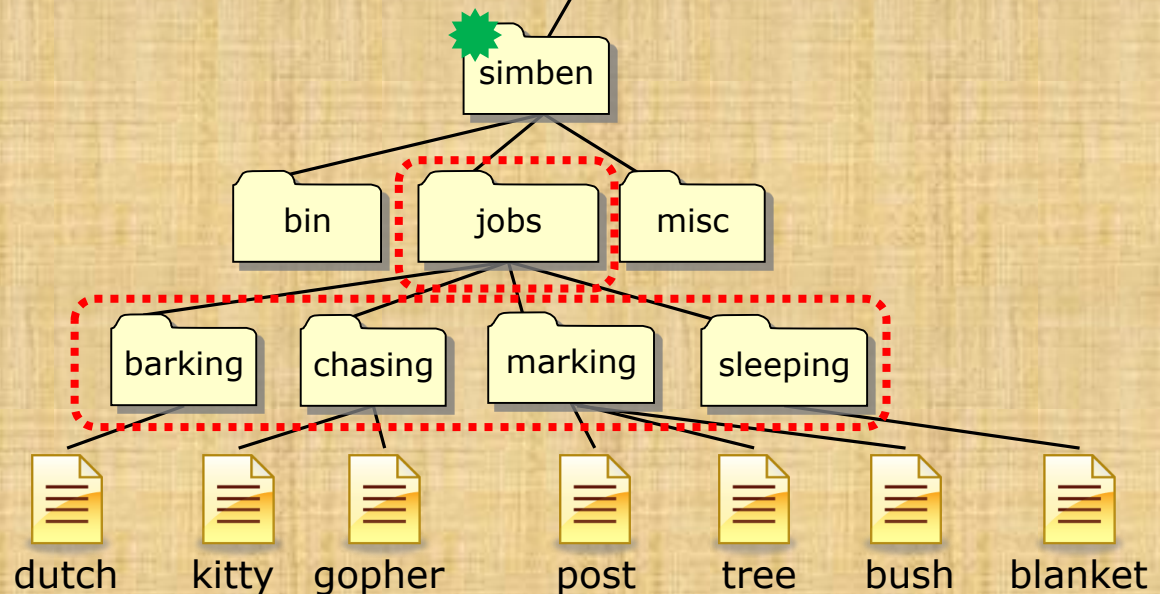
Both these answers are correct



From  how does Benji change permissions on:

1. His *jobs/* directory
2. The four sub-directories under *jobs/*

to full permissions for the owner, read & execute for group and none for others?



*Write your answer in the chat window*

*You can make your own jobs directory by issuing:*

```
cd  
tar xvf ../depot/jobs.tar
```

### *This works*

```
chmod 750 jobs
cd jobs
chmod 750 barking
chmod 750 chasing
chmod 750 marking
chmod 750 sleeping
```

### *So does this*

```
chmod 750 jobs
chmod 750 jobs/barking
chmod 750 jobs/chasing
chmod 750 jobs/marketing
chmod 750 jobs/sleeping
```

### *And this*

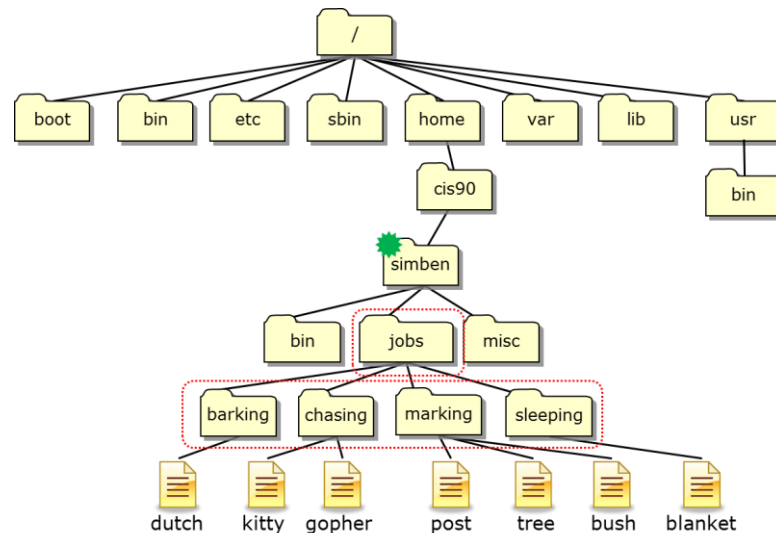
```
chmod 750 jobs
chmod 750 jobs/barking/ jobs/chasing/ jobs/marketing/ jobs/sleeping/
```

### *This is better though*

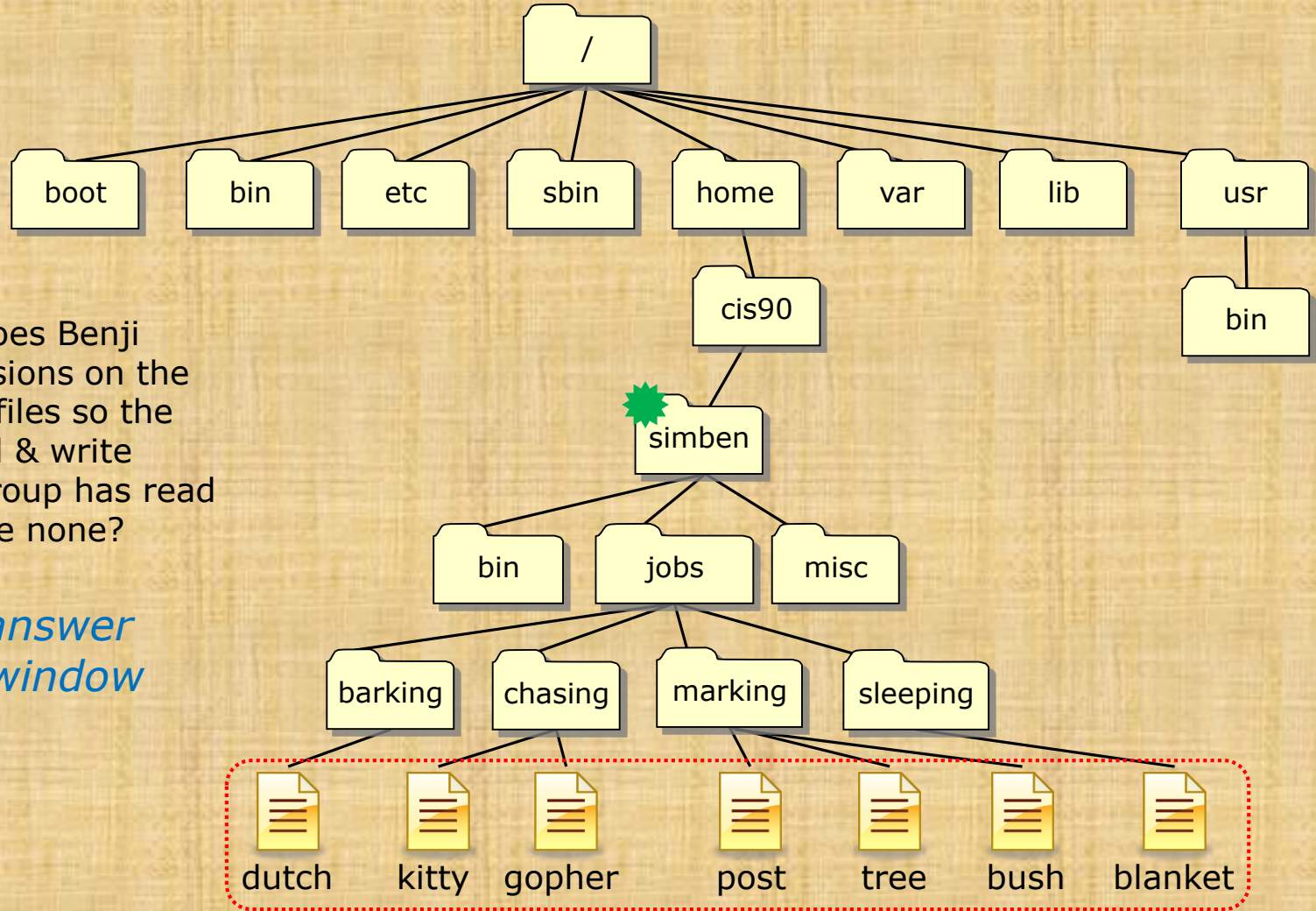
```
chmod 750 jobs
chmod 750 jobs/*
```


### *I like this the best!*

```
chmod 750 jobs jobs/*
```



*And so ... which way did you do step 9 in Lab 6?*



From  how does Benji change permissions on the circled regular files so the owner has read & write permissions, group has read and others have none?

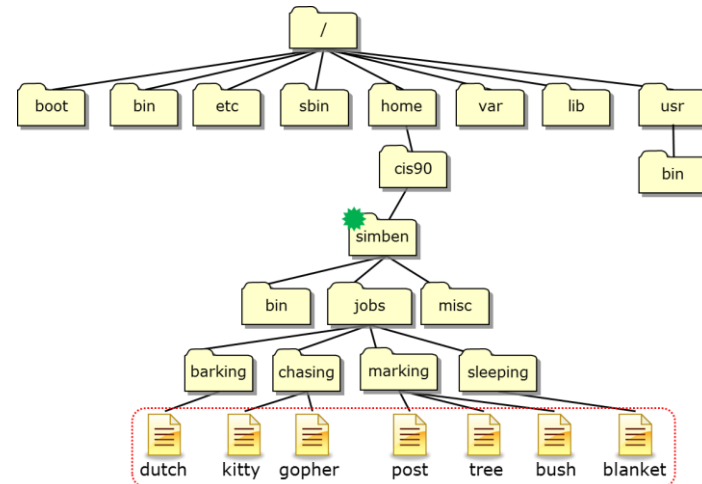
*Write your answer in the chat window*

*This will always work*

```
cd jobs
cd barking
chmod 640 dutch
cd ..
cd chasing
chmod 640 kitty
chmod 640 gopher
cd ..
cd marking
chmod 640 post
chmod 640 tree
chmod 640 bush
cd ..
cd sleeping
chmod 640 blanket
cd
```

*This works too*

```
cd jobs
cd barking
chmod 640 dutch
cd ..
cd chasing
chmod 640 kitty gopher
cd ..
cd marking
chmod 640 post tree bush
cd ..
cd sleeping
chmod 640 blanket
cd
```



*So will this*

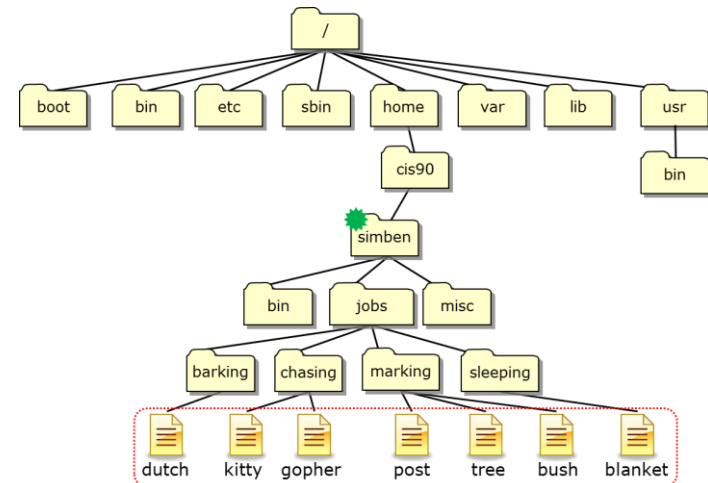
```
cd jobs
cd barking
chmod 640 *
cd ..
cd chasing
chmod 640 *
cd ..
cd marking
chmod 640 *
cd ..
cd sleeping
chmod 640 *
cd
```

*This is better*

```
cd jobs
chmod 640 barking/*
chmod 640 chasing/*
chmod 640 marking/*
chmod 640 sleeping/*
cd ..
```

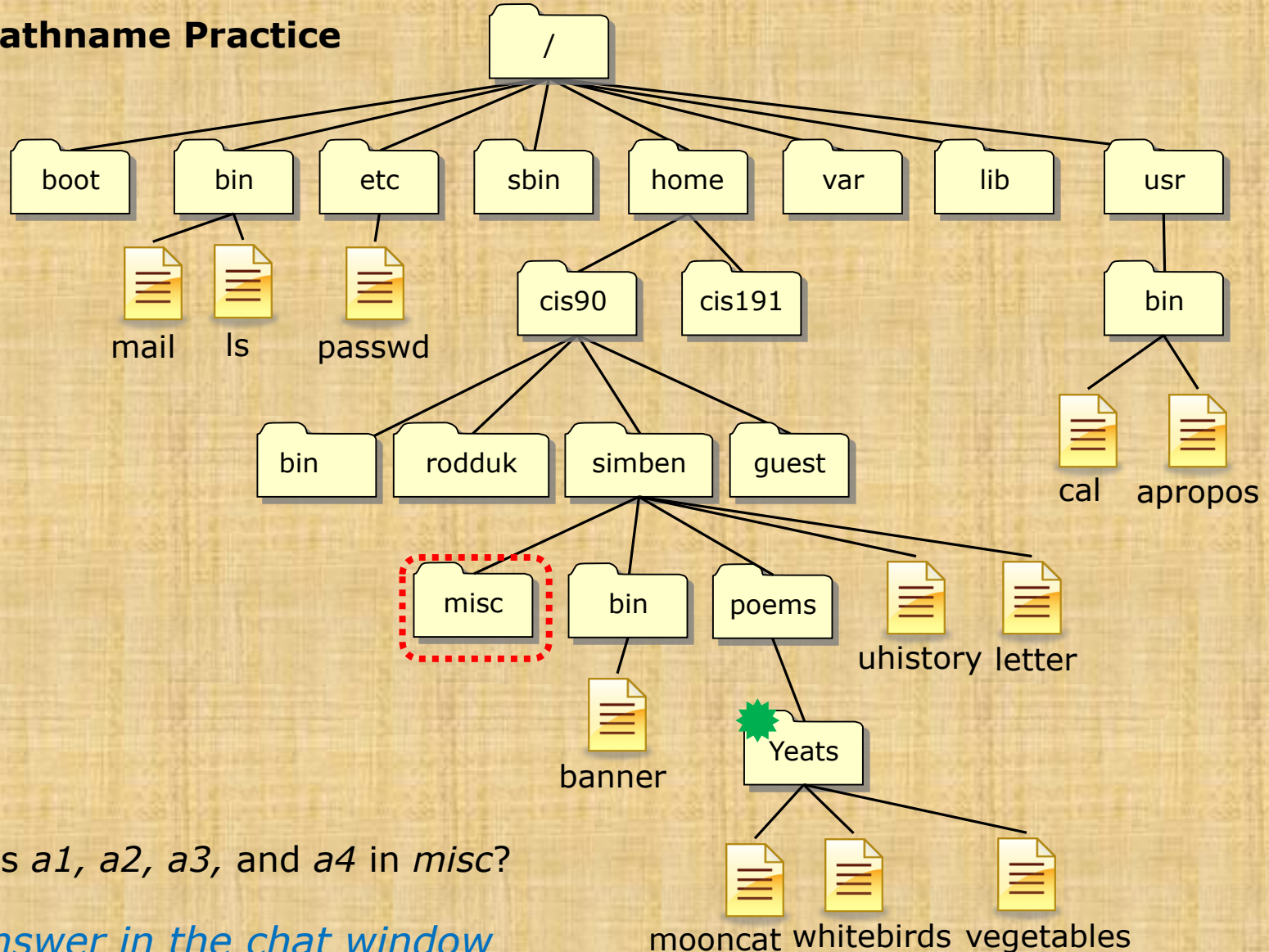
*I like this the best!*

```
chmod 640 jobs/*/*
```



*And so ... which way did you do step 10 in Lab 6?*

**File Tree Pathname Practice**

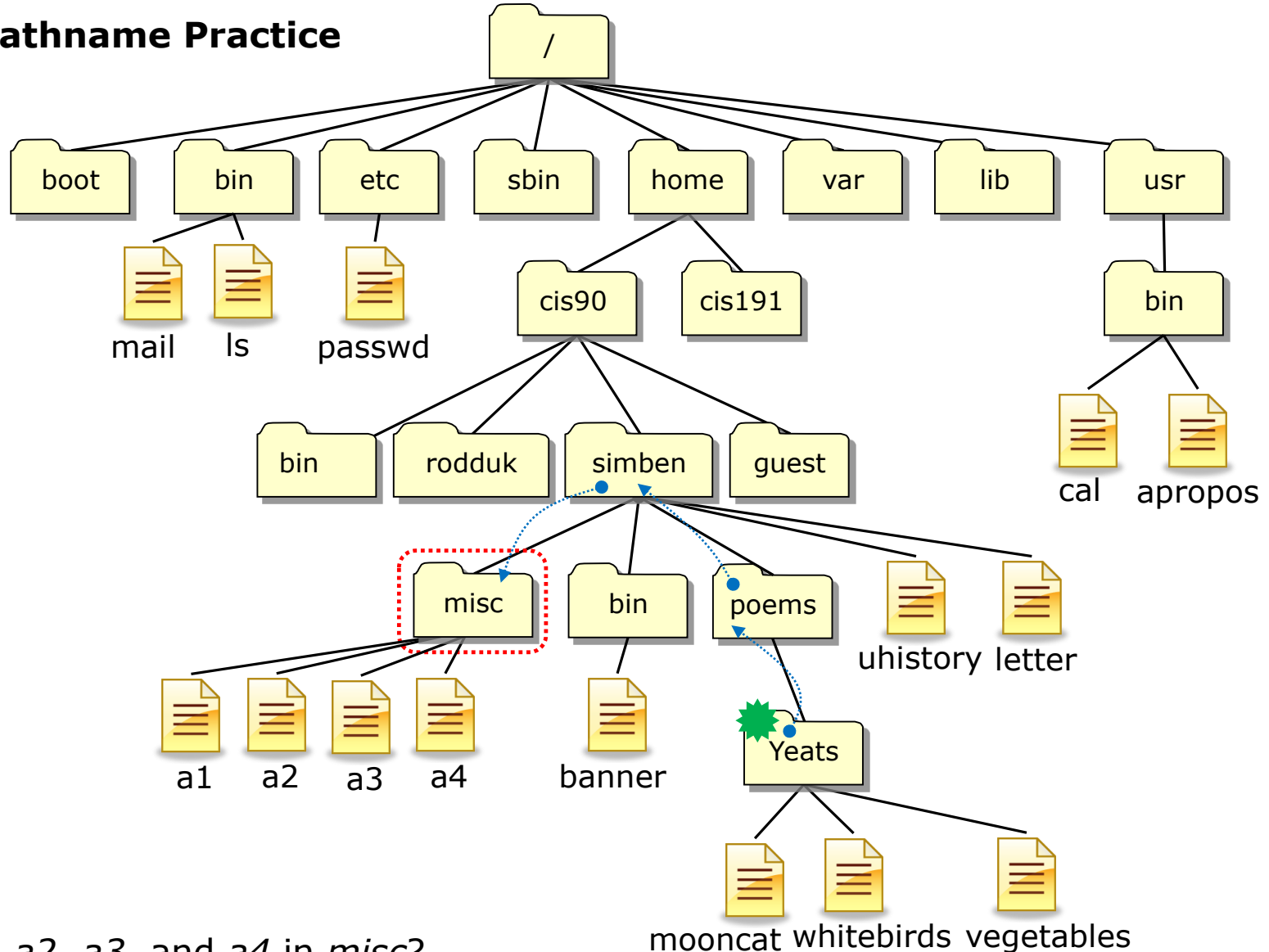


From  how does Benji:

Create new files *a1*, *a2*, *a3*, and *a4* in *misc*?

*Write your answer in the chat window*

## File Tree Pathname Practice



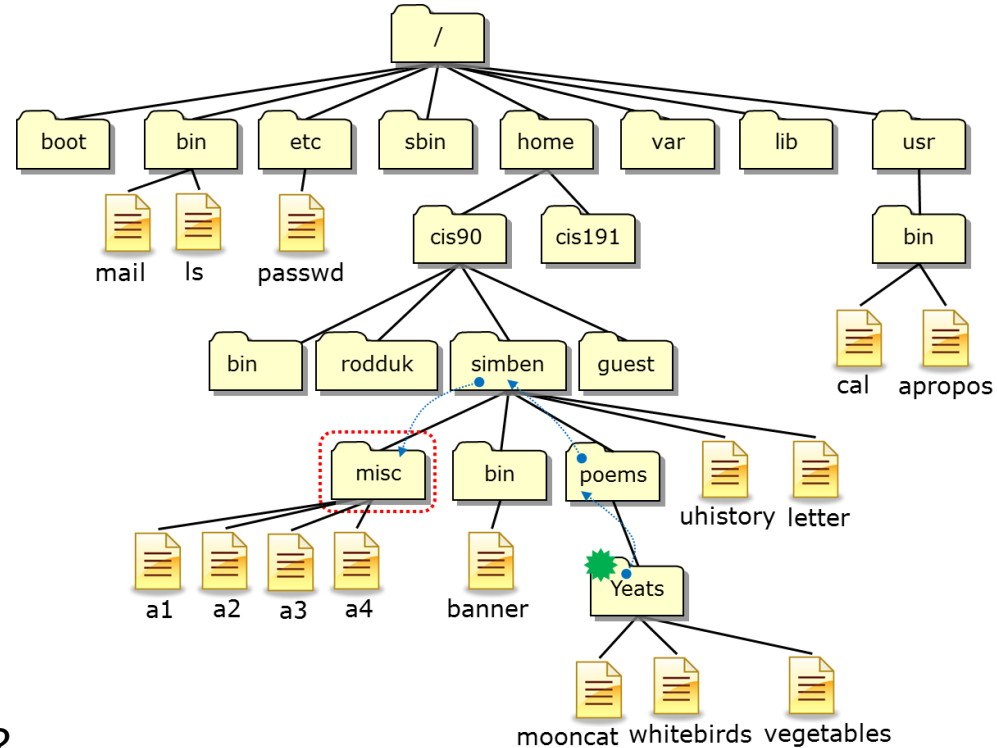
From  how does Benji:

Create files *a1*, *a2*, *a3*, and *a4* in *misc*?

`/home/cis90/simben/poems/Yeats $ touch ../../misc/a1 ../../misc/a2 ../../misc/a3 ../../misc/a4`



*Other answers  
are also  
acceptable*



From  how  
does Benji:

Create files *a1*, *a2*, *a3*, and *a4* in *misc*?

**touch <pathname> <pathname> <pathname> <pathname>**

**touch ../../misc/a1 ../../misc/a2 ../../misc/a3 ../../misc/a4**

**or touch ~/misc/a1 ~/misc/a2 ~/misc/a3 ~/misc/a4**

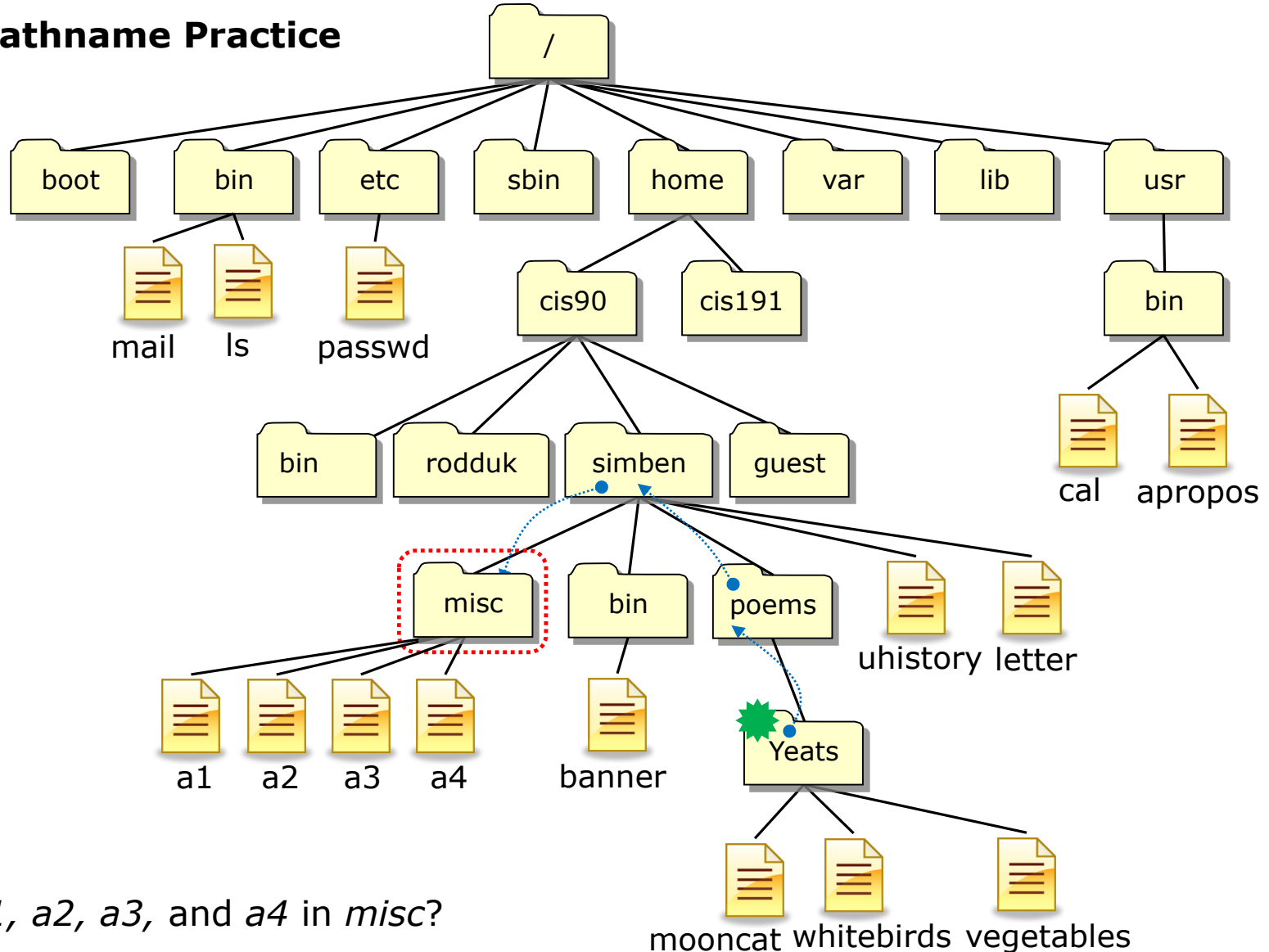
**or touch /home/cis90/simben/misc/a1 /home/cis90/simben/misc/a2  
/home/cis90/simben/misc/a3 /home/cis90/simben/misc/a4 (all on one line)**

*All these answers are correct*



*For the aspiring gurus  
there is an even better  
way to do the last  
operation!*

## File Tree Pathname Practice



From  how does Benji:

Create files *a1*, *a2*, *a3*, and *a4* in *misc*?

`/home/cis90/simben/poems/Yeats $ touch ~/misc/a{1,2,3,4}`

*Using curly braces and commas*

# umask review

## Why umask?

Allows users and system administrators to disable specific permissions on new files and directories when they are created.

*Unlike **chmod**, it does **NOT** change the permissions on existing files or directories.*

## umask summary

To determine permissions on a new file or directory apply the umask to the initial starting permissions:

- For new files, start with **666**
- For new directories, start with **777**
- For file copies, start with **the permission on the source file**

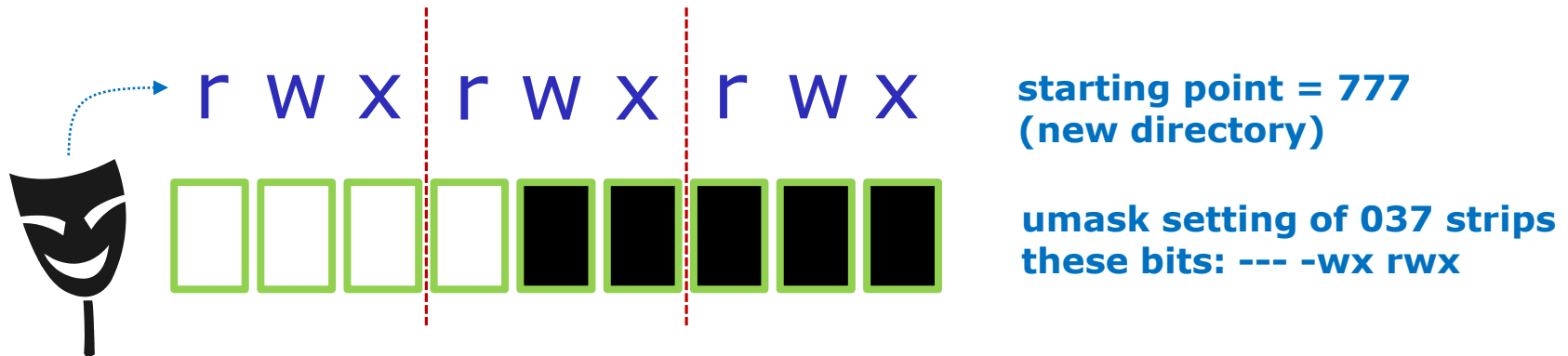
## Case 1 – a new directory

**With a umask of 037 what permissions would a newly created DIRECTORY have?**

*Write your answer in the chat window*

## Case 1 – a new directory

With a umask of 037 what permissions would a newly created DIRECTORY have?

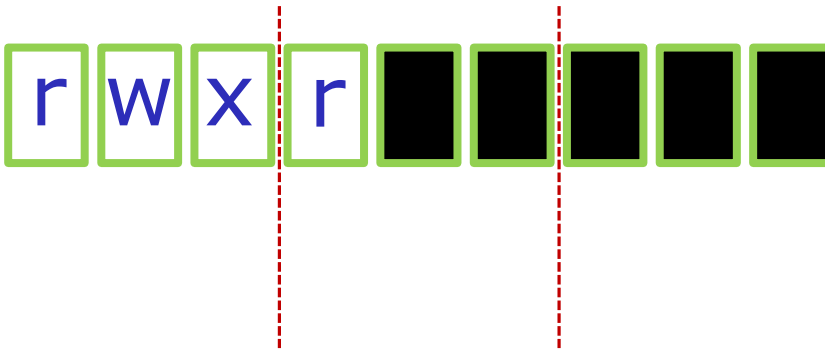


*Now slide the mask up and over the starting point permissions*



## Case 1 – a new directory

**With a umask of 037 what permissions would a newly created DIRECTORY have?**



starting point = 777  
(new directory)

umask setting of 037 strips  
these bits: --- -wx rwx

**Answer: 740**

*Prove it to yourself on Opus-II as shown here*

```
[rsimms@opus-ii ~]$ umask 037
[rsimms@opus-ii ~]$ mkdir brandnewdir
[rsimms@opus-ii ~]$ ls -ld brandnewdir/
drwxr-----. 2 rsimms rsimms 6 Mar 18 16:55 brandnewdir/
 7 4 0
```

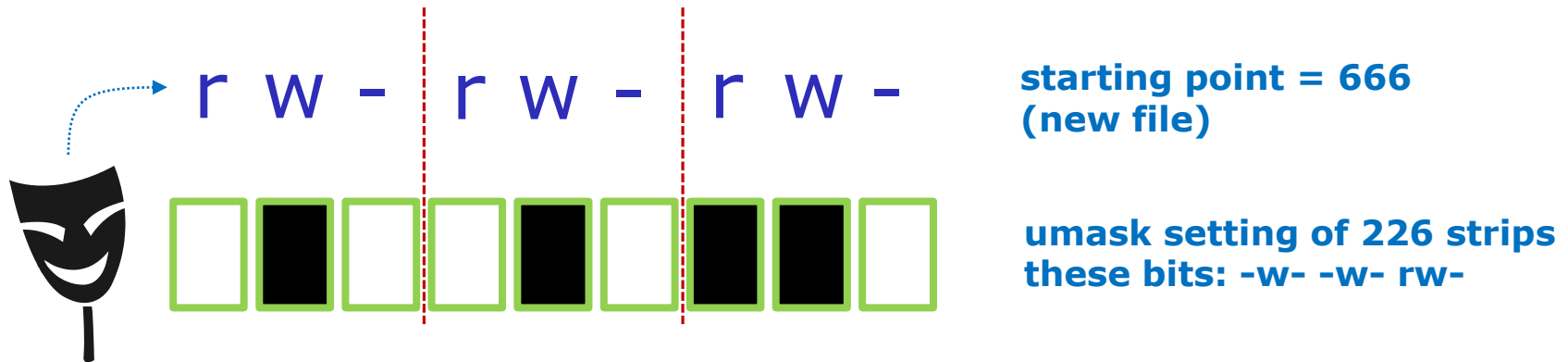
## Case 2 – new file

**With a umask of 226 what permissions would a newly created FILE have?**

*Write your answer in the chat window*

## Case 2 – new file

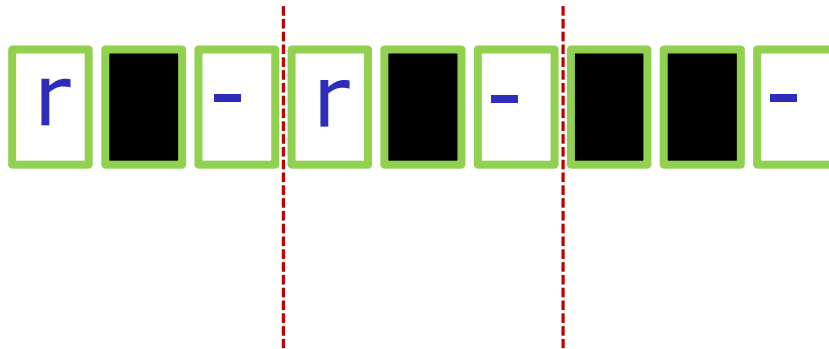
With a umask of 226 what permissions would a newly created FILE have?



*Now slide the mask up and over the starting point permissions*

## Case 2 – new file

With a umask of 226 what permissions would a newly created FILE have?



starting point = 666  
(new file)

umask setting of 226 strips  
these bits: -w- -w- rw-

**Answer: 440**

*Prove it to yourself on Opus-II as shown here*

```
[rsimms@opus-ii ~]$ umask 226
[rsimms@opus-ii ~]$ touch brandnewfile
[rsimms@opus-ii ~]$ ls -l brandnewfile
-r--r-----. 1 rsimms rsimms 0 Mar 18 17:08 brandnewfile
 4 4 0
```

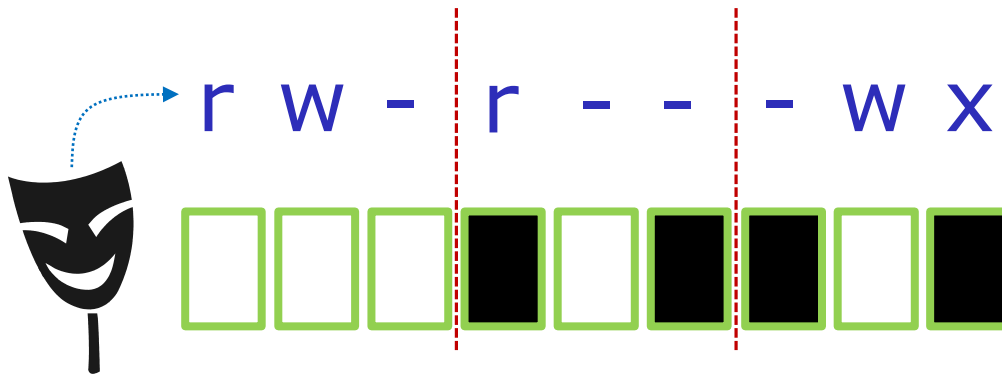
## Case 3 – file copy

**If umask=055 and the *cinderella* file permissions are 643  
What would the permissions be on *cinderella.bak* after:  
cp cinderella cinderella.bak**

*Write your answer in the chat window*

### Case 3 – file copy

If `umask=055` and the *cinderella* file permissions are `643`  
 What would the permissions be on *cinderella.bak* after:  
`cp cinderella cinderella.bak`



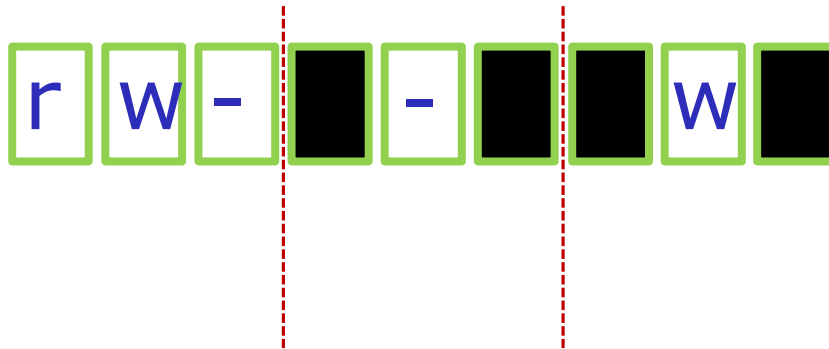
starting point = 643  
 (source file permissions)

umask setting of 055 strips  
 these bits: --- r-x r-x

*Now slide the mask up and over the starting point permissions*

## Case 3 – file copy

If `umask=055` and the *cinderella* file permissions are `643`  
 What would the permissions be on *cinderella.bak* after:  
`cp cinderella cinderella.bak`



starting point = 643  
 (source file permissions)

umask setting of 055 strips  
 these bits: --- r-x r-x

**Answer: 602**

*Prove it to yourself on Opus-II as shown here*

```
[rsimms@opus-ii ~]$ chmod 643 cinderella
[rsimms@opus-ii ~]$ umask 055
[rsimms@opus-ii ~]$ cp cinderella cinderella.bak
[rsimms@opus-ii ~]$ ls -l cinderella.bak
-rw-----w-. 1 rsimms rsimms 0 Mar 18 17:23 cinderella.bak
 6 0 2
```

# Housekeeping







Pause/Stop Recording

# Pause Recording

Audio Check

# Roll Call

If you are watching the archived video please email me to let me know you were here.

[risimms@cabrillo.edu](mailto:risimms@cabrillo.edu)

# Overlap Students

Don't forget to update the Google  
Docs Log when watching the  
recording




Resume/Stop Recording

# Resume Recording

## Audio Check

## Previous material and assignment

1. Lab 6 due 11:59PM.
2. Use **check6** to check your work on the lab.
3. Don't forget to **submit** your final Lab 6! 
4. Use **verify** to view what you submitted for grading.
5. Five more posts due 11:59PM.
6. Early preview of Lab X2 is now available. This is recommended for anyone wanting more practice with pathnames.

# Your turn to grade me!

1. March 18th - NAS Office sends online SurveyMonkey survey to all CIS 90 students. Please complete it by April 17th.
2. I've added the survey link to the website Calendar page as well (see Lesson 9).
3. April 3rd - Department Chair (Mike) will visit our class to observe.

Student Evaluation of Online Instructor Richard Stevens, Spring 2019

One of the major responsibilities of the faculty is to provide high teaching standards among the faculty. Please take the time to evaluate the instructor of the course.

Please be thoughtful and candid in your responses. Be assured that you will remain anonymous in the process and that your answers will be given the highest consideration. They will be forwarded at the end of the semester when grades have been submitted. Please do not put your name on this survey.

1. Course section:

CIS 90 - Sec 1 - Introduction to LINUX/Linux

2. Is there a required text or software for your course?

Yes  No

3. Is there a website for your course?

Yes  No

4. How often do you use the online resources?

Daily  2 times/week  3 times/week  4 times/week  Other

5. Are the computer system requirements for the course clearly stated?

Yes  No

6. Is the instructor contact information clearly posted?

Yes  No

7. Which of the following methods of contact between the instructor and students are regularly used in the course? Check all that apply.

Announcements/Email  Voice mail with students  Correspondence by postal mail  Discussion boards  Office hours  Email  Scheduled lecture/discussion  Live streaming  Mentoring/coaching  Field trips  Telephone  Study and/or review conferences  Other (please describe):

8. Rate your Online Instructor in each of the following categories:

	Excellent	Good	Acceptable	Average	Poor
A. Instructional content (both on and off-line) is clear and well-organized and appropriate and useful.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
B. Course objectives and grading criteria are specifically stated and clearly explained in course materials.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
C. Amount and type of assigned course work are appropriate.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
D. Course content is well organized.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
E. Methods of evaluation are clearly stated and related to stated content. (For example: tests, projects, writing assignments, and portfolios.)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
F. Grading is fair and impartial.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
G. Working contact with your instructor who demonstrates an enthusiasm for the subject.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
H. Working contact with your instructor who is knowledgeable and prepared.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I. Interacts with your instructor who encourages your interest, participation, and intellectual effort.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
J. The instructor communicates clearly.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
K. The instructor is accessible for individual consultation.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
L. The instructor demonstrates respect and concern towards students, including respectful consideration of any accommodations, special needs, or special requests. (If needed.)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
M. Requires graded assignments in reasonable time with logical comments.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
N. NET OVERALL EVALUATION OF THIS INSTRUCTOR(S):	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

9. Describe the strengths of the instructor. (Please be specific and give examples.)

10. What could the instructor do to improve his/her effectiveness or teaching style?

11. Would you recommend the instructor to other students and would you take another class from the instructor? Why or why not?

12. If you have additional comments, please elaborate below.

Done

UNIVERSITY OF CALIFORNIA  
See How We're Making a Difference  
Privacy & Data Policy



# New commmands



## Lesson 8 commands for your toolbox



**find** - Find file or content of a file



**grep** - "Global Regular Expression Print"



**sort** - sort



**spell** - spelling correction

**wc** - word count



**tee** - split output



**cut** - cut fields from a line



# sort command

# sort command

## Basic syntax

(see man page for the rest of the story)

**sort** *<options>* *<filepath>*

The **sort** command can read lines from a file or *stdin* and sort them.

The **-r** option will do a reverse sort

# Activity

Copy the *names* file in the *depot* directory to your home directory.

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

```
/home/cis90/simben $ cp ../depot/names .
```

```
/home/cis90/simben $ cat names
```

```
duke  
benji  
star  
homer
```

*The "." means "here". This is the current directory we are in.*

We will use this file in the next several examples.

*Write "names file copied" into the chat window when done.*



Pretend you are a  
command

(use your great  
imagination)

**Shell Steps**

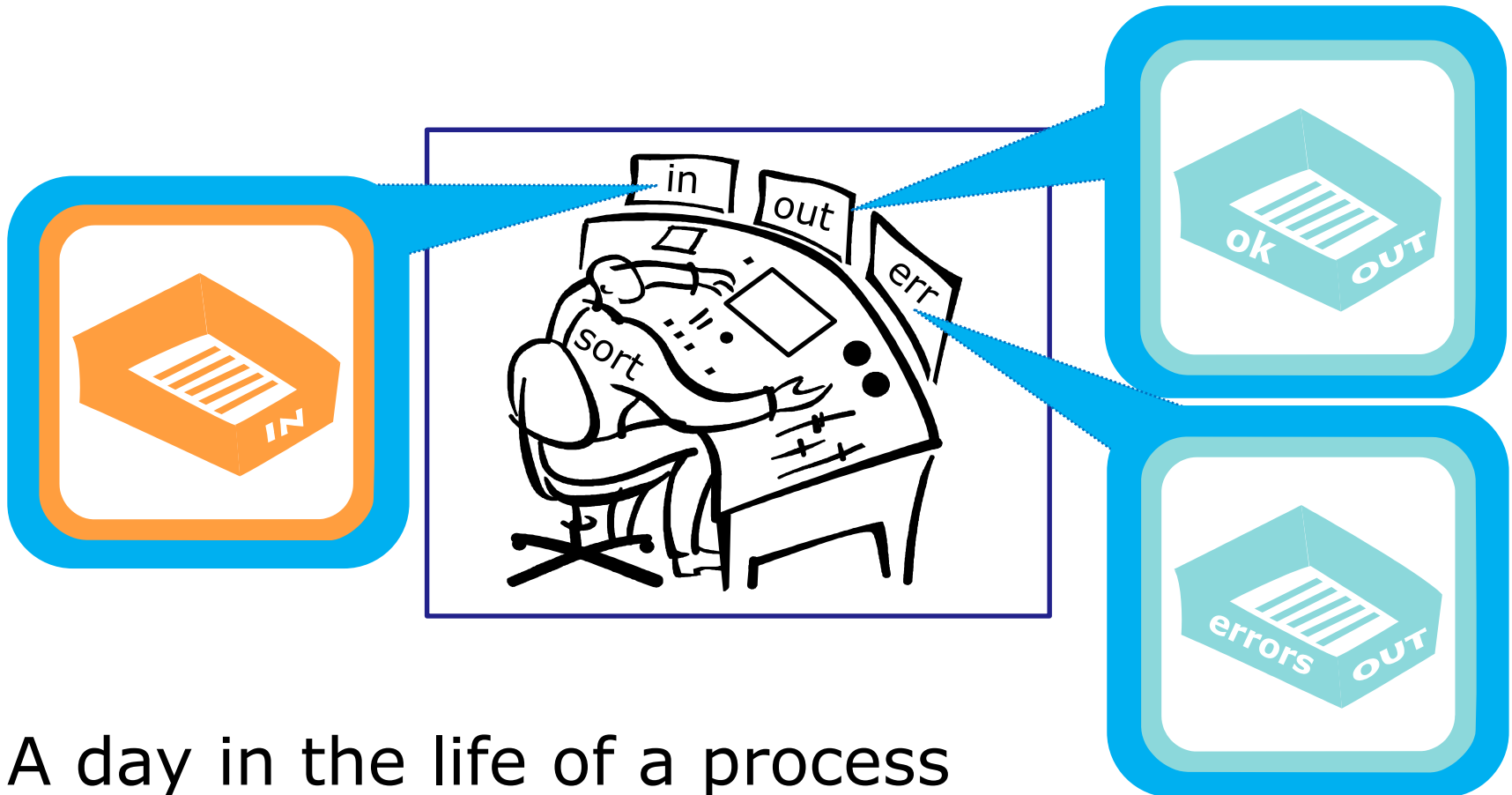
- 1) Prompt
- 2) Parse
- 3) Search
- 4) Execute
- 5) Nap
- 6) Repeat

*Let's visualize being the sort program and being loaded into memory and executing*



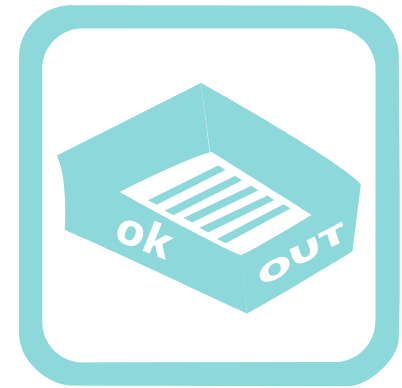
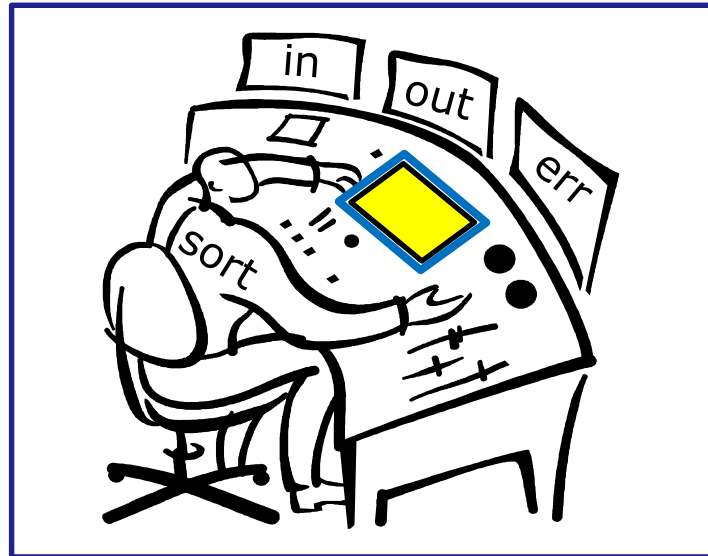
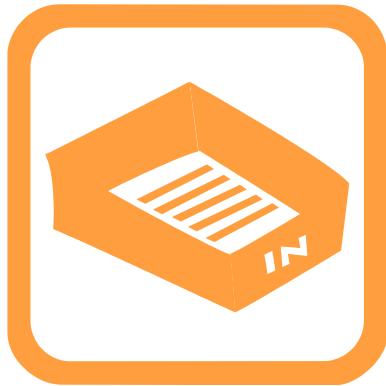
A day in the life of a process

*Looking around you notice there is one in tray and two out trays*



A day in the life of a process

*You also notice an instruction window on your desk. This is where you find out about any options or arguments the shell passes on to you.*



A day in the life of a process



# sort

## deep dive


## examples



# **sort** *<good filepath>*

```
/home/cis90/simben $ sort names  
benji  
duke  
homer  
star  
/home/cis90/simben $
```

*One argument  
which is a  
filename*



## Activity

The **sort** command with a filename argument.

```
/home/cis90/simben $ cat names
```

```
duke
```

```
benji
```

```
star
```

```
homer
```

```
/home/cis90/simben $ sort names
```

```
benji
```

```
duke
```

```
homer
```

```
star
```

The sort command will sort the lines in a file and output the sorted lines.

*Write "file sorted" into the chat window when done.*

```
/home/cis90/simben $ sort names
```

### Shell Steps

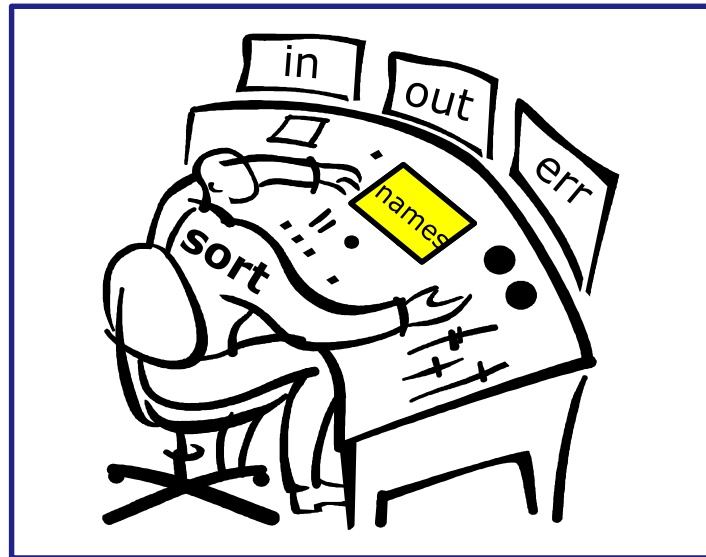
- 1) Prompt
- 2) Parse
- 3) Search
- 4) Execute
- 5) Nap
- 6) Repeat

1. Prompt string is: `"/home/cis90/simben $ "`
2. Parsing results:
  - `command = sort`
  - no options
  - 1 argument = `"names"`
  - no redirection
3. Search user's path and locate the sort program in `/bin`
4. Sort loaded into memory and execution begins

**Shell Steps**

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

/home/cis90/simben \$ **sort names**



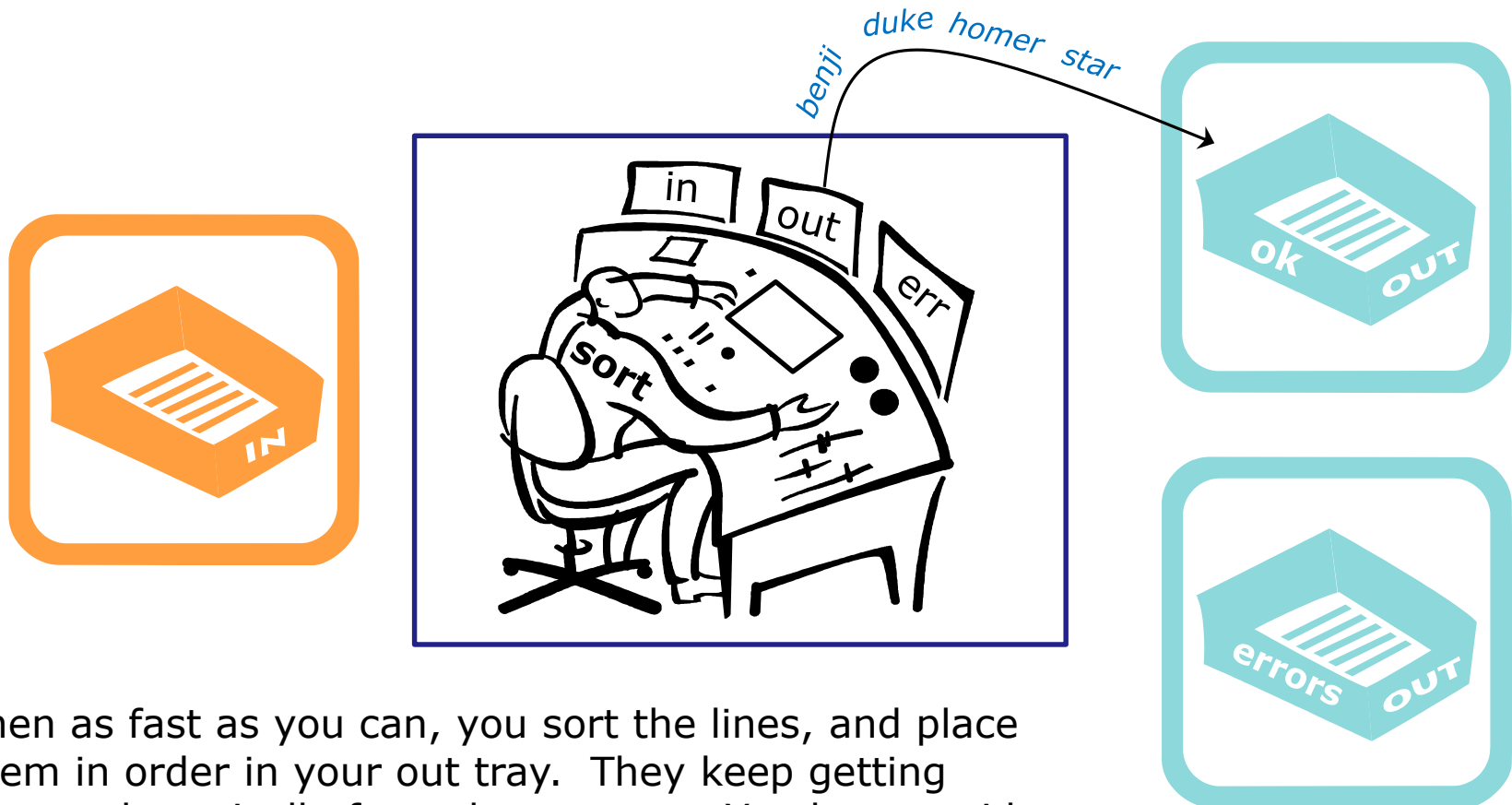
You (the sort process) check your instruction window and see the shell passed one argument "names" to you. You know (given your internal DNA code) that you must contact the kernel and request this file be opened and the contents read.

```
/home/cis90/simben $ sort names
```



Note: Once the names file is opened you read in each line one at a time until you reach the EOF (End of File).

/home/cis90/simben \$ **sort names**



Then as fast as you can, you sort the lines, and place them in order in your out tray. They keep getting removed magically from the out tray. You have no idea where they go after that. You are done.



# sort (*no arguments*)

```
/home/cis90/simben $ sort  
kayla  
sky  
bella  
benji  
charlie  
bella  
benji  
charlie  
kayla  
sky  
/home/cis90/simben $
```

*No arguments  
specified*


*EOF*

## Activity

The **sort** command with no arguments.

```

/home/cis90/simben $ sort
kayla
sky
bella
benji
charlie
bella
benji
charlie
kayla
sky
  
```



A diagram showing two buttons labeled 'ctrl' and 'D' with a blue dotted arrow pointing from the 'D' button to the word 'charlie' in the terminal output above.

If no filename was specified, **sort** will read input from the keyboard

Ctrl-D specifies the EOF (End Of File).

After sort receives the EOF it sorts the lines and outputs them

*Write "input sorted" into the chat window when done.*



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

### Shell Steps

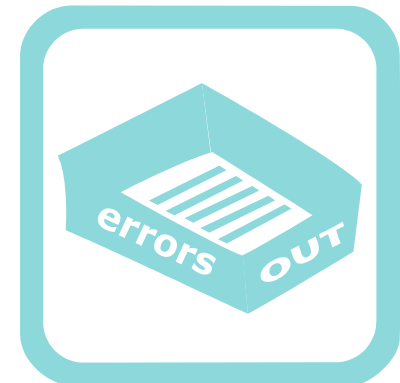
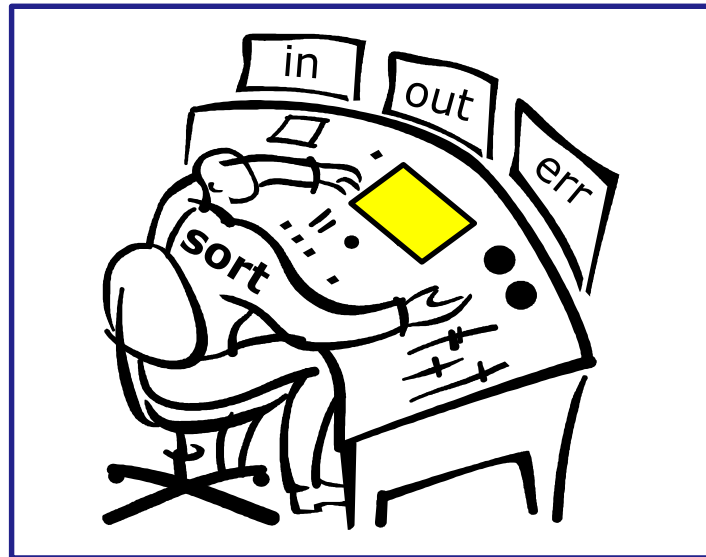
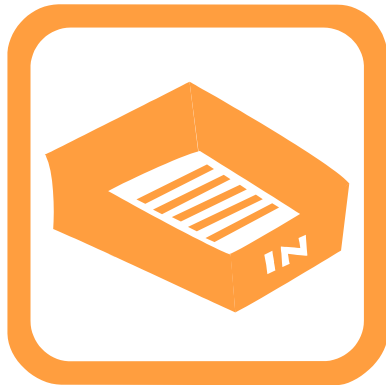
- 1) Prompt
- 2) Parse
- 3) Search
- 4) Execute
- 5) Nap
- 6) Repeat

1. Prompt string is: `"/home/cis90/simben $ "`
2. Parsing results:
  - `command = sort`
  - no options
  - no arguments
  - no redirection
3. Search user's path and locate the sort program in `/bin`
4. Sort loaded into memory and execution begins

**Shell Steps**

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

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

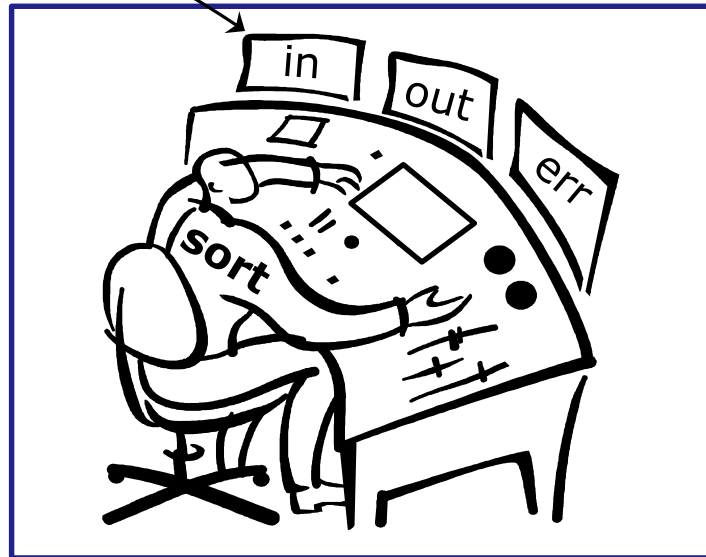


You (the sort process) check your instruction window and see that no options or arguments were passed to you from the shell to handle. You know (given your internal DNA code) that with no arguments you must look for lines to sort in your in tray, so you reach in to grab the first line to sort.

/home/cis90/simben \$ **sort**

kayla  
sky  
bella  
benji  
charlie

*charlie benji bella sky kayla*



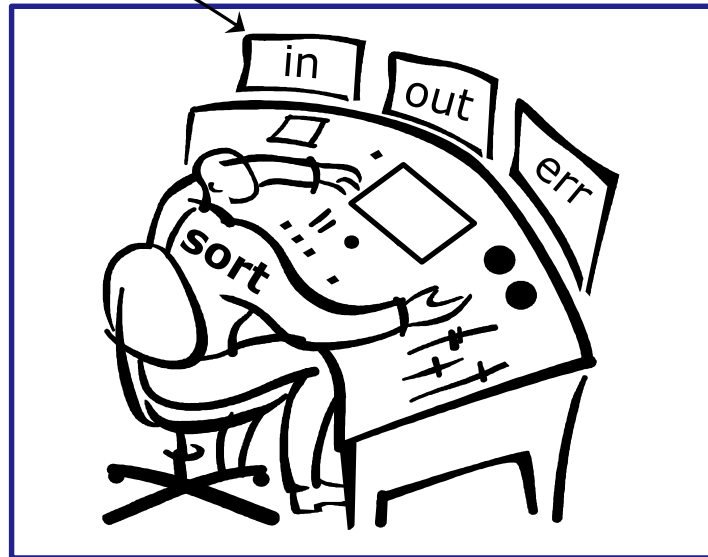
You work hard and fast. Each time you reach into the in tray there is another line! They just magically keep appearing into your in tray. You have no idea where they are coming from.

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

```
kayla  
sky  
bella  
benji  
charlie
```

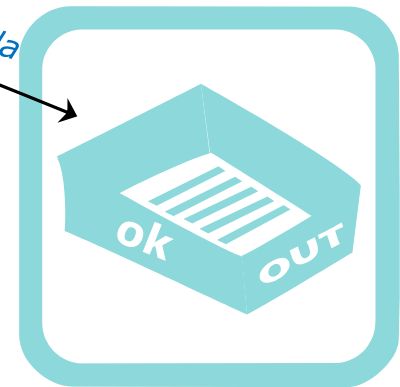
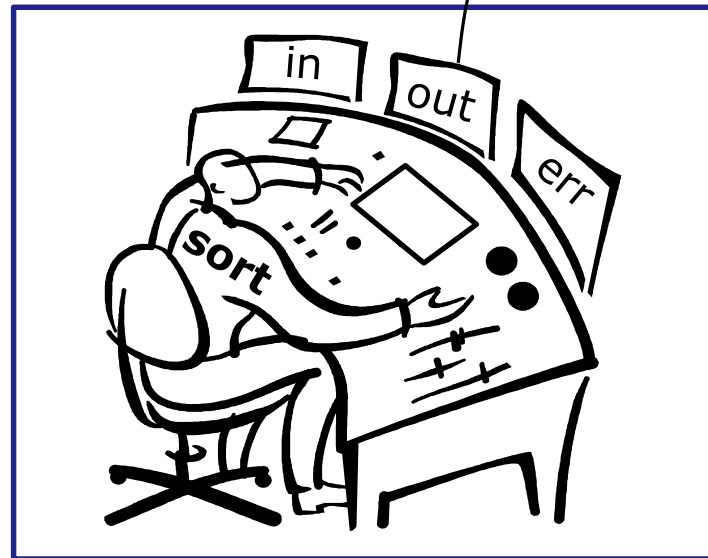


EOF



Then suddenly, when you reach for the next line, you find an EOF. You know (your internal DNA code) that this EOF means no more lines coming. You must sort what you have collected so far and place them, in order, into your out tray.

bella  
benji  
charlie  
kayla  
sky  
/home/cis90/simben \$



As fast as you can, you sort them, and place them in order in your out tray. They keep getting removed magically from the out tray. You have no idea where they go after that. You are done.

# **sort** <*bad filepath*>

```
/home/cis90/simben $ sort bogus  
sort: open failed: bogus: No such file or directory  
/home/cis90/simben $
```

 *No such file*

## Activity

The **sort** command with a bad argument.

```
/home/cis90/simben $ sort bogus  
sort: open failed: bogus: No such file or directory  
/home/cis90/simben $
```

The sort program will try and open the file it receives as an argument and print an error message if the file does not exist

*Write "sort failed" into the chat window when done.*

```
/home/cis90/simben $ sort bogus
```

### Shell Steps

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

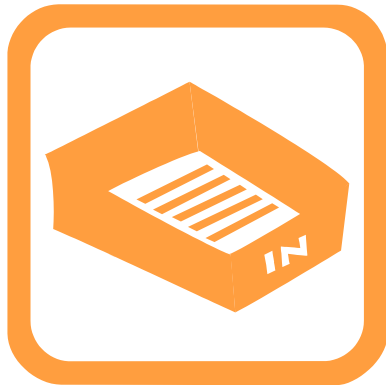
1. Prompt string is: `"/home/cis90/simben $ "`
2. Parsing results:
  - `command = sort`
  - `no options`
  - `1 argument = bogus`
  - `no redirection`
3. Search user's path and locate the `sort` program in `/bin`
4. `Sort` command loaded into memory and execution begins



```
/home/cis90/simben $ sort bogus
```

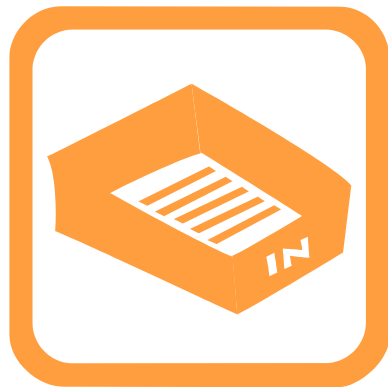
**Shell Steps**

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

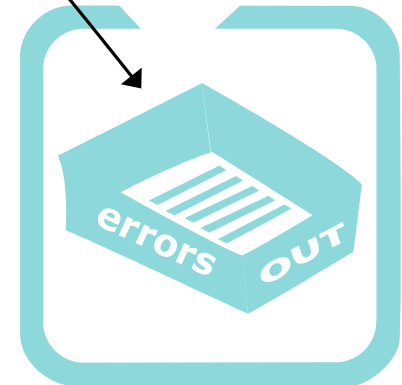


You check the instruction window and notice the shell passed you one argument: "bogus". You know (given your internal DNA code) that you must contact the kernel and request this file be opened.

```
/home/cis90/simben $ sort bogus  
sort: open failed: bogus: No such file or directory
```



sort: open failed: bogus:  
No such file or directory



However the kernel tells you the file does not exist.  
You place an error message in the out tray for errors.  
You are done.



# Bringing it home



# File Descriptors

# Input and Output

## File Descriptors

Every process is given three open files upon its execution. These open files are inherited from the shell.

### **stdin**

Standard Input (0)

*defaults to the user's terminal keyboard*

### **stdout**

Standard Output (1)

*defaults to the user's terminal screen*

### **stderr**

Standard Error (2)

*defaults to the user's terminal screen*

*Ok, lets make the visualization a little more realistic*

The in and out trays are really the three open file descriptors inherited from the shell:  
**stdin (0)**, **stdout (1)** and **stderr (2)**.

**stdin (0)**



**stdout (1)**

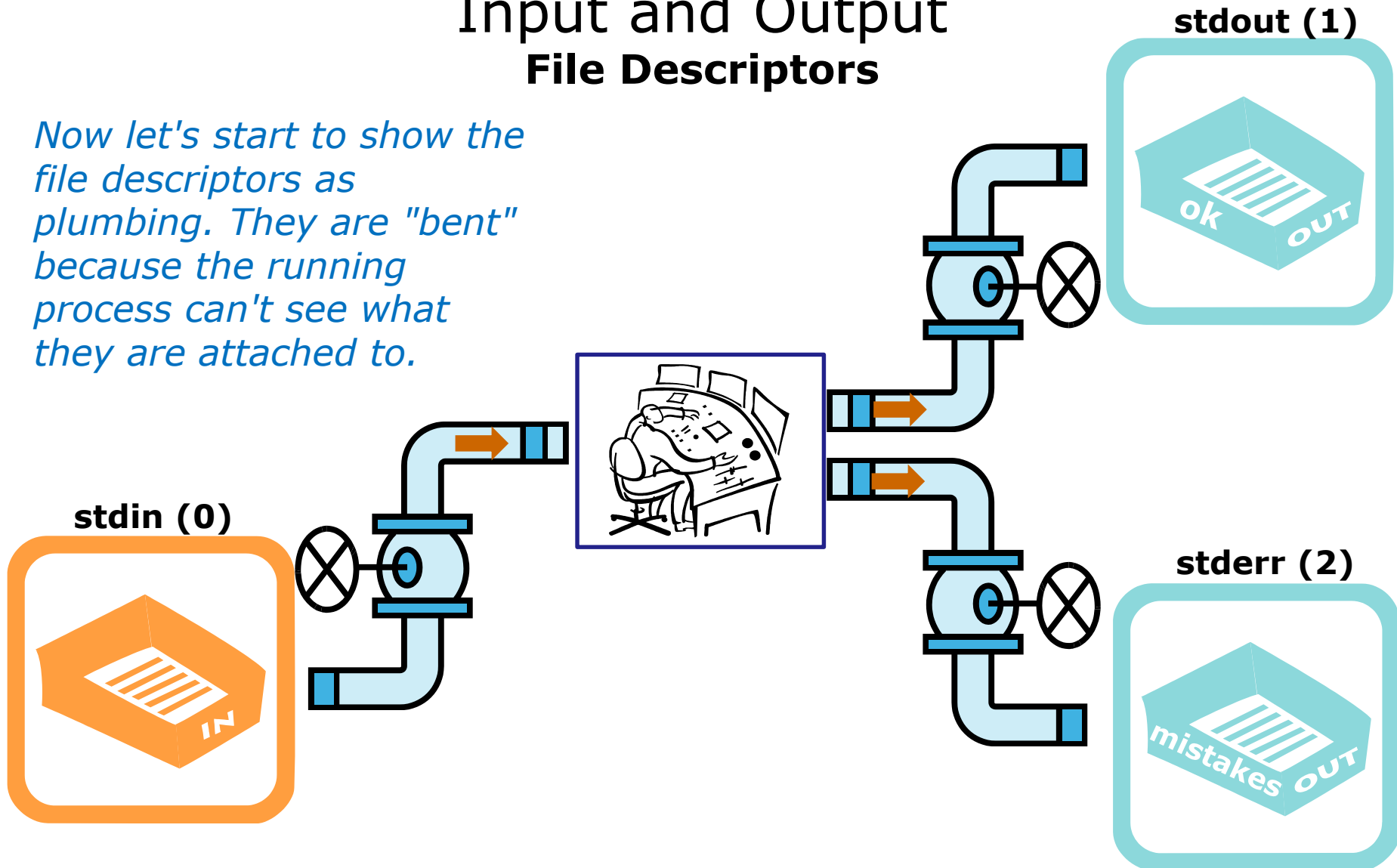


**stderr (2)**



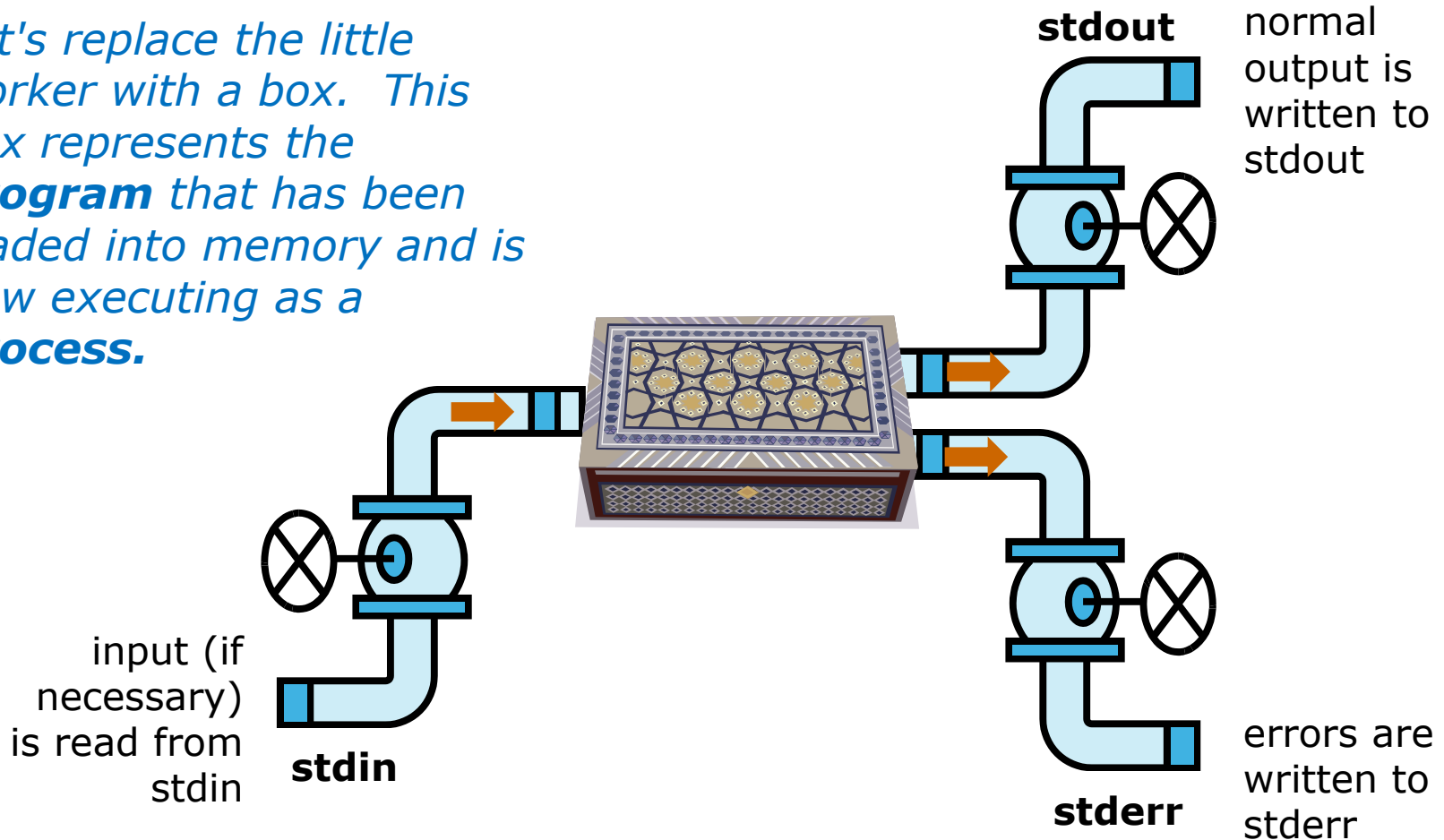
# Input and Output File Descriptors

*Now let's start to show the file descriptors as plumbing. They are "bent" because the running process can't see what they are attached to.*



# Input and Output Loaded Process

Let's replace the little worker with a box. This box represents the **program** that has been loaded into memory and is now executing as a **process**.



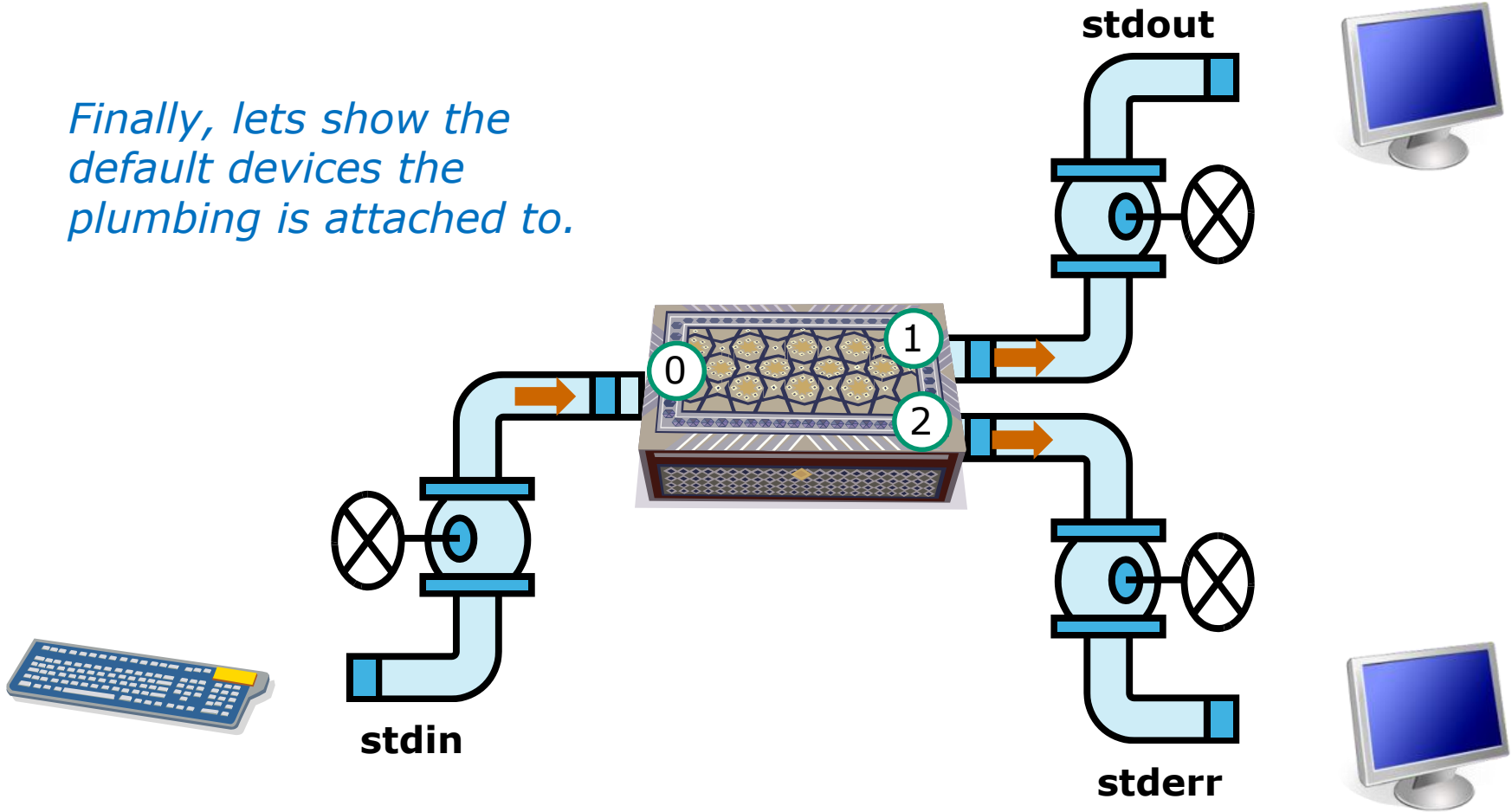


# Input and Output

## Default I/O devices

By default is attached to the user's terminal device (screen)

*Finally, lets show the default devices the plumbing is attached to.*



By default is attached to the user's terminal device (keyboard)

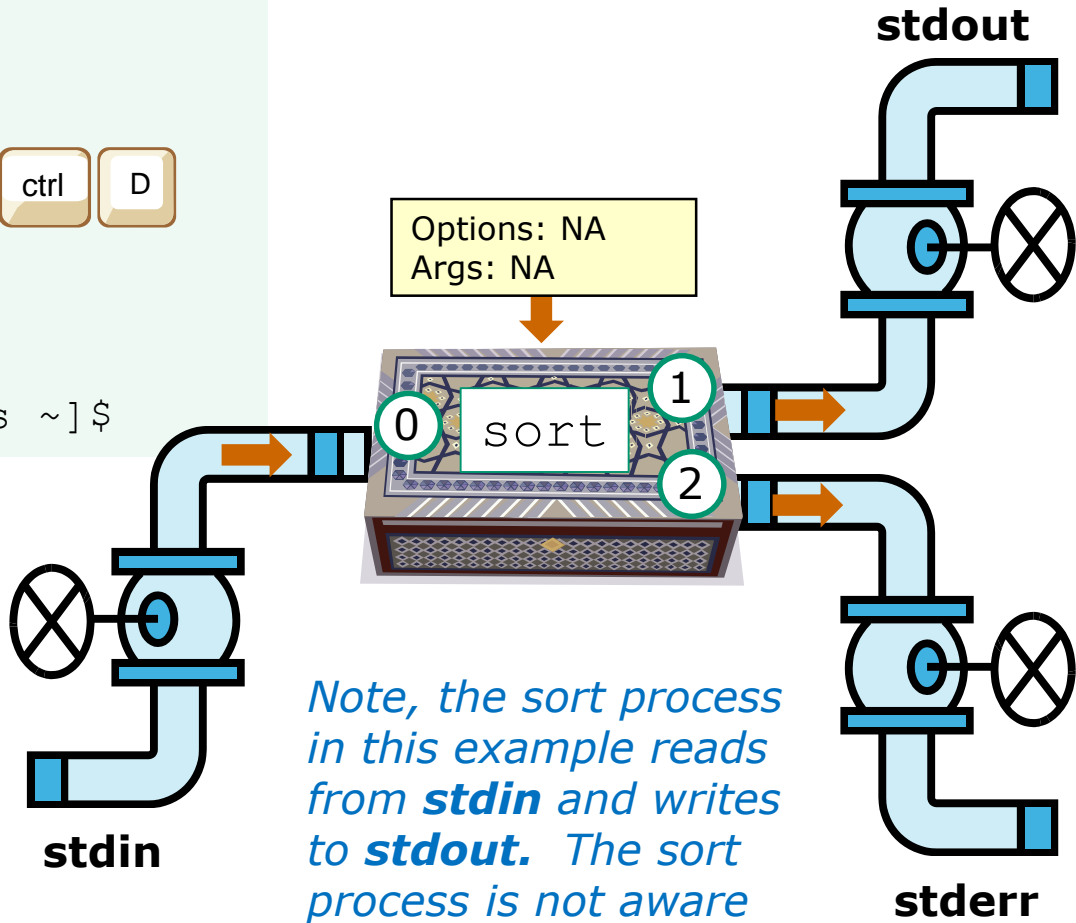
By default is attached to the user's terminal device (screen)

# The sort example again with no arguments

```
[simmsben@opus ~]$ sort
star
benji
duke
homer
benji
duke
homer
star
[simmsben@opus ~]$
```



Options: NA  
Args: NA



benji  
duke  
homer  
star



*Note, the sort process in this example reads from **stdin** and writes to **stdout**. The sort process is not aware what **stdin** or **stdout** are attached to*

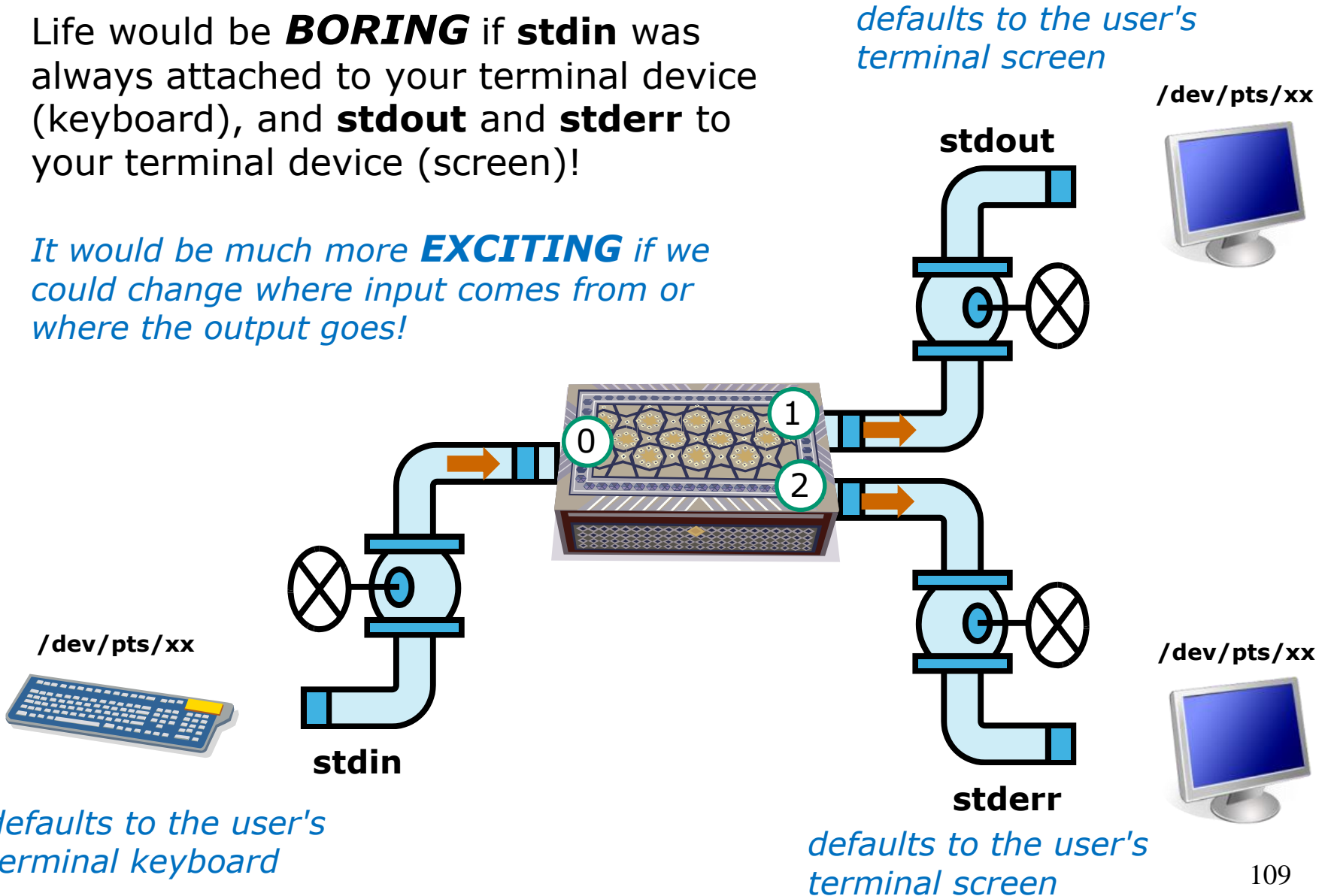
star  
benji  
duke  
homer



# File Redirection

Life would be **BORING** if **stdin** was always attached to your terminal device (keyboard), and **stdout** and **stderr** to your terminal device (screen)!

*It would be much more **EXCITING** if we could change where input comes from or where the output goes!*




# Input and Output

## File Redirection

*Let's look at the  
sort example again*

```
/home/cis90/simben $ sort  
duke  
benji  
star  
homer  
benji  
duke  
homer  
star  
/home/cis90/simben $
```



The diagram illustrates the effect of the Ctrl+D key combination on the output of the `sort` command. It shows two buttons labeled "ctrl" and "D" with a dotted blue arrow pointing from the "D" button to the word "homer" in the output. Below the buttons is the text "End of File".

# Input and Output

## File Redirection



Read from **stdin**

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

```
duke  
benji  
star  
homer
```

*The sort program reads lines from **stdin** (attached to keyboard)*



"End of File"

Written to **stdout**



```
benji  
duke  
homer  
star
```

*After the EOF it performs the sort and writes to **stdout** (attached to terminal)*

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

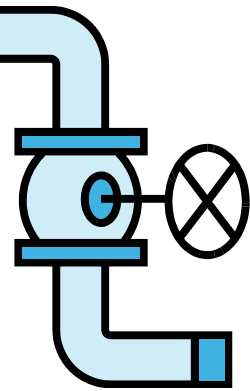
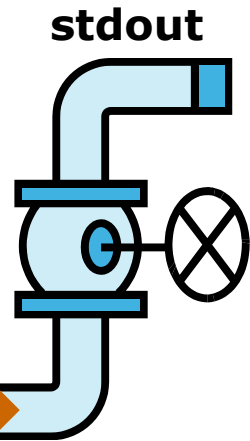
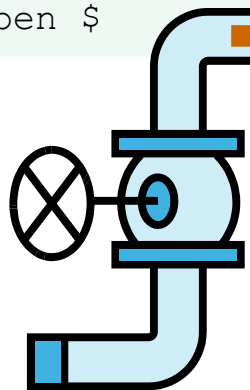
# sort command (no arguments)

```

/home/cis90/simben $ tty
/dev/pts/0
/home/cis90/simben $ sort
duke
benji
star
homer
benji
duke
homer
star
/home/cis90/simben $
    
```



Options: NA  
Args: NA



/dev/pts/0



benji  
duke  
homer  
star

/dev/pts/0




duke  
benji  
star  
homer

*Note: The shell (bash) attaches up the default input and output devices.*

*The sort program has no idea what is attached at the end of the pipes.*

# Activity


**Input and sort some names of dogs.**



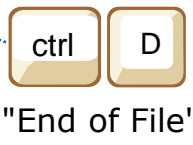
Read from **stdin**

```

/home/cis90/simben $ sort
duke
benji
star
homer
benji
duke
homer
star
/home/cis90/simben $
    
```



Written to **stdout**



"End of File"

The diagram shows a terminal window with a list of dog names being sorted. A keyboard icon is positioned to the left of the input lines, and a computer monitor icon is to the left of the output lines. A bracket on the left groups the first four lines (duke, benji, star, homer) under the label 'Read from stdin'. Another bracket on the left groups the next four lines (benji, duke, homer, star) under the label 'Written to stdout'. A dotted blue arrow points from the 'ctrl D' keys to the second 'benji' line in the output, with the text '"End of File"' below it.

When YOU do this. What specific device (e.g. /dev/pts/xx) is stdin and stdout attached to?

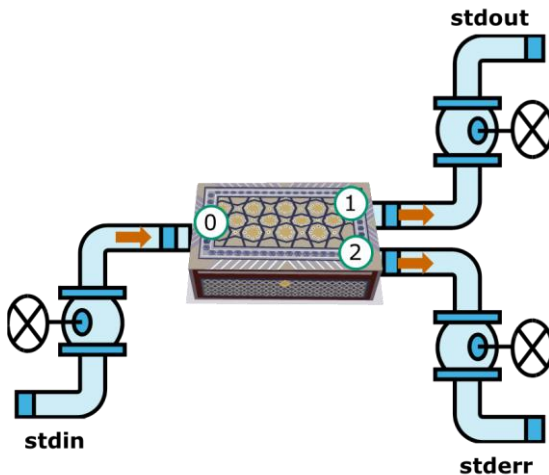
Write your answer in the chat window.



# Input and Output

## File Redirection

The input and output of a program can be **redirected** from and to other files using `<`, `>`, `2>` and `>>`:



~~0~~< *pathname*

To redirect **stdin** (either `0<` or just `<`)

~~1~~> *pathname*

To redirect **stdout** (either `1>` or just `>`)

**2**> *pathname*

To redirect **stderr**

**>>** *pathname*

To redirect **stdout** and append

### Notes:

- The "pathname" above is either an absolute or relative pathname.
- The space between the redirection character and the pathname is optional.

# No arguments, redirecting stdout

sort just reads from **stdin**  
and writes to **stdout**

**stdout** has been  
redirected to the file  
*dogsinorder*

```
[simmsben@opus ~]$ sort > dogsinorder
```

**duke**

**benji**

**star**

**homer**



If the file *dogsinorder* does not exist, it is  
created. If it does exist it is emptied!

```
[simmsben@opus ~]$ cat dogsinorder
```

benji

duke

homer

star

```
[simmsben@opus ~]$
```

## No arguments, redirecting stdout

```
$ sort > dogsinorder
```

```
duke
benji
star
homer
$
```



Options: NA  
Args: NA



stdout



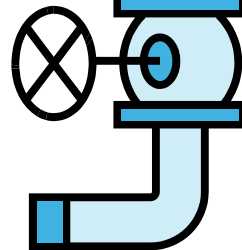
dogsinorder

```
$ cat dogsinorder
benji
duke
homer
star
```

/dev/pts/0

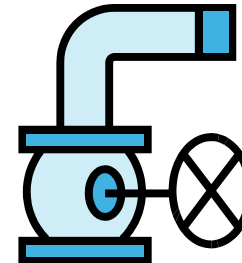


duke  
benji  
star  
homer



stdin

Note: sort doesn't know that input comes from the keyboard or that output will be sent to the *dogsinorder* file.



stderr

It just reads from **stdin** and writes to **stdout**.

## Now you try it

**Redirect the output from sort to a file named *dogsorder*.**

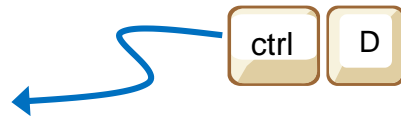
```
[simmsben@opus-ii ~]$ sort > dogsorder
```

```
duke
```

```
benji
```

```
star
```

```
homer
```



```
[simmsben@opus-ii ~]$ cat dogsorder
```

```
benji
```

```
duke
```

```
homer
```

```
star
```

```
[simmsben@opus-ii ~]$
```

*Write "sorted" into the chat windows when finished*

# No arguments, redirecting stdin and stdout

```
[simben@opus ~]$ cat names
```

```
duke
```

```
benji
```

```
star
```

```
homer
```

input is redirected to come  
from the file *names*

output is redirected to the  
file *dogsorder*

```
[simben@opus ~]$ sort < names > dogsorder
```

```
[simben@opus ~]$ cat dogsorder
```

```
benji
```

```
duke
```

```
homer
```

```
star
```

```
[simben@opus ~]$
```

Note: The bash shell handles the  
command line parsing and redirection.  
The sort command has no idea what  
***stdin*** or ***stdout*** are attached to.



# No arguments, redirecting stdin and stdout

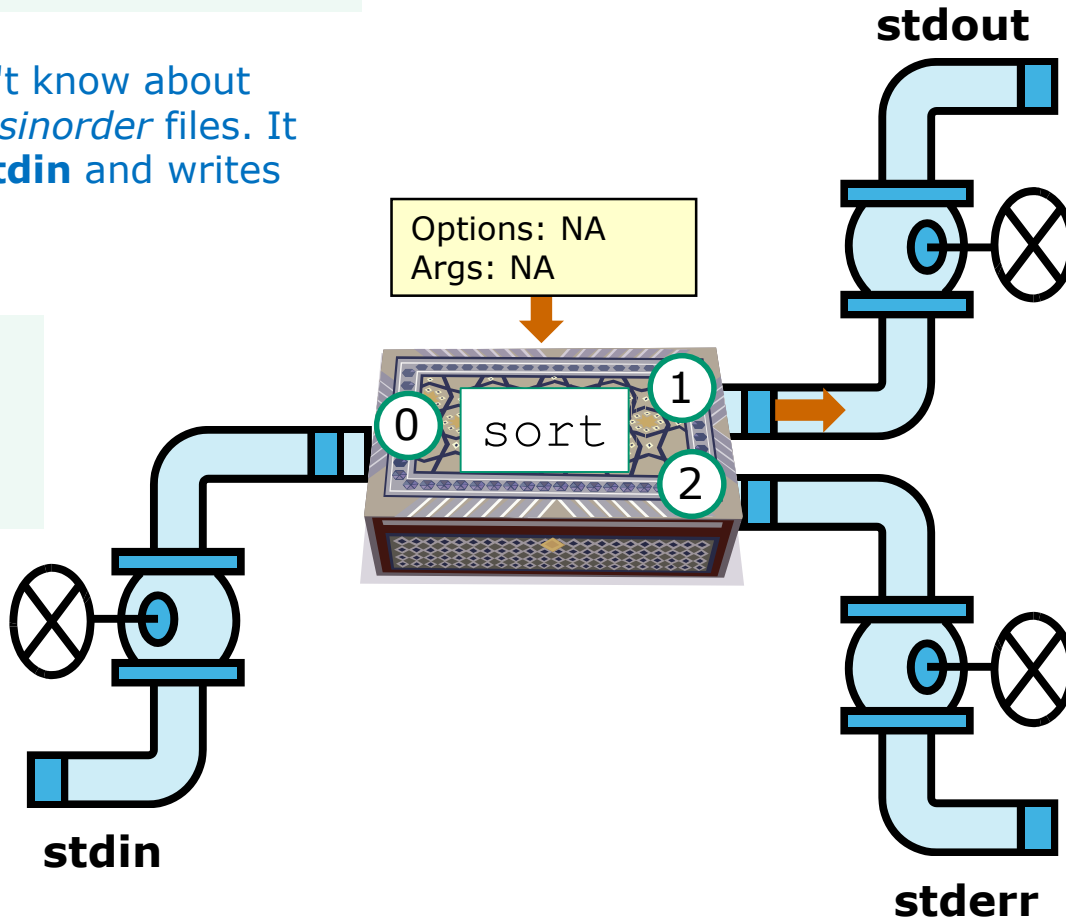
```
$ sort < names > dogsinorder
```

Note: `sort` doesn't know about the `names` or `dogsinorder` files. It just reads from **stdin** and writes to **stdout**.

```
$ cat names
duke
benji
star
homer
```



names



dogsinorder

```
$ cat dogsinorder
benji
duke
homer
star
```

In this example, `sort` is getting its input from **stdin**, which has been redirected to the `names` file

## Now you try it

```
[simben@opus-ii ~]$ cat names
duke
benji
star
homer
[simben@opus-ii ~]$ sort < names > dogsinorder

[simben@opus-ii ~]$ cat dogsinorder
benji
duke
homer
star
[simben@opus-ii ~]$
```

Does the **sort** program know that its input came from the *names* file or its output went to the *dogsinorder* file?

Put your answer in the chat window.

# One argument, redirecting stdout

The *names* file is parsed as an **argument** and is passed to the sort process to handle.

Output written to **stdout** is redirected to the file *dogsinorder*.

The shell, not the sort program, opens the *dogsinorder* file.

```
[simben@opus ~]$ sort names > dogsinorder
[simben@opus ~]$ cat dogsinorder
benji
duke
homer
star
[simben@opus ~]$
```

The sort program, not the shell, opens and reads directly from the *names* file.

Корисне для наступного вікторини!

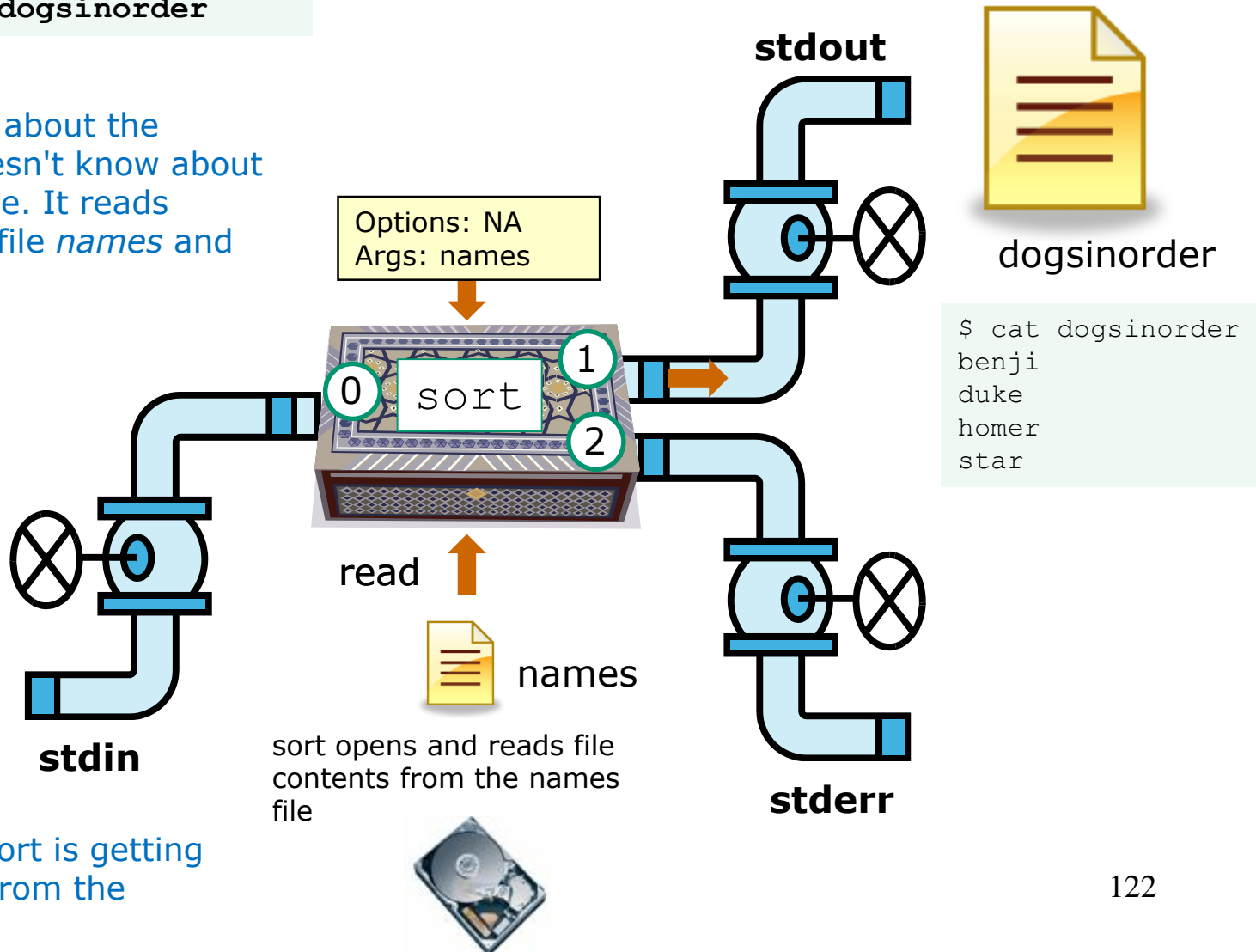


## One argument, redirecting stdout

```
$ sort names > dogsinorder
```

Note: `sort` knows about the `names` file but doesn't know about the `dogsinorder` file. It reads directly from the file `names` and writes to **stdout**.

Корисне для наступного вікторини!



In this example, `sort` is getting its input directly from the `names` file

## Now you try it

```
[simben@opus-ii ~]$ sort names > dogsinorder  
[simben@opus-ii ~]$ cat dogsinorder  
benji  
duke  
homer  
star  
[simben@opus-ii ~]$
```

yes

Does the **sort** program know that its input came from the names file?

*Put your answer in the chat window.*

Корисне для  
наступного  
вікторини!

# One option, one argument, redirecting stdout

specifying an option  
(for reverse order)

*names* is parsed as an  
argument and passed to the  
sort command

sort writes to **stdout**, which is  
redirected to the file *dogsinorder*

```
[simben@opus ~]$ sort -r names > dogsinorder
```

```
[simben@opus ~]$ cat dogsinorder
```

```
star
```

```
homer
```

```
duke
```

```
benji
```

```
[simben@opus ~]$
```

This **-r** option does the sort in  
reverse order

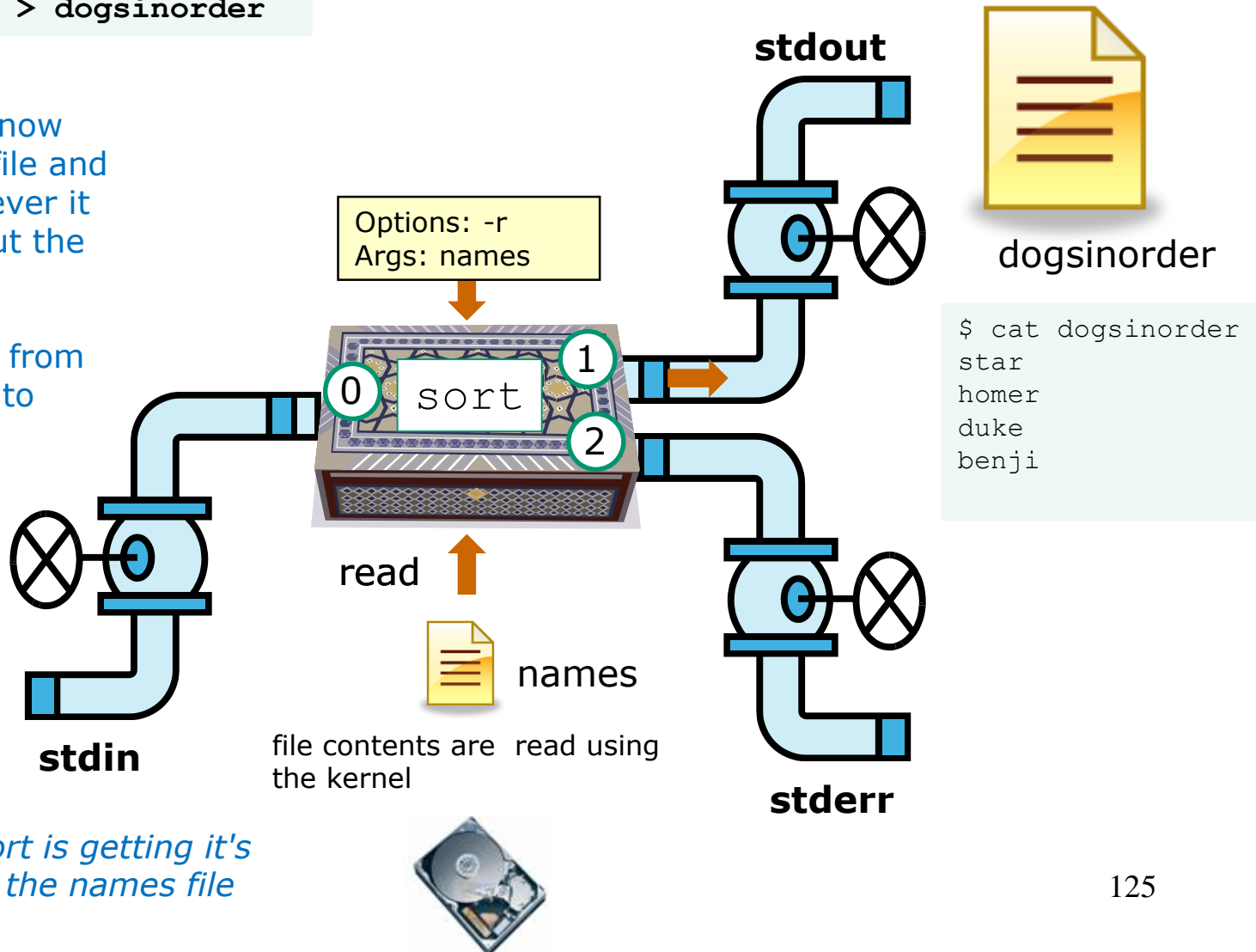
The shell opens the *dogsinorder*  
file. The sort process is not aware  
that output is redirected there.

# One option, one argument, redirecting stdout

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

Note: `sort` does know about the `names` file and the `-r` option however it doesn't know about the `dogsinorder` file.

`sort` reads directly from `names` and writes to **stdout**.



*In this example, `sort` is getting its input directly from the `names` file*

## Now you try it

```
/home/cis90/simben $ sort -r names > dogsinorder  
/home/cis90/simben $ cat dogsinorder  
star  
homer  
duke  
benji  
/home/cis90/simben $
```

Does the **sort** program know that its output is going to the *dogsinorder* file?

*Put your answer in the chat window*

# Append vs Overwrite

## > (overwrites) vs >> (appends)

```
[simben@opus ~]$ echo "Hello World" > message
```

```
[simben@opus ~]$ cat message
```

```
Hello World
```

```
[simben@opus ~]$ echo "Hello Universe" >> message
```

```
[simben@opus ~]$ cat message
```

```
Hello World
```

```
Hello Universe
```

*>> does not empty  
file, just appends to  
the end*

```
[simben@opus ~]$ echo "Oops" > message
```

```
[simben@opus ~]$ cat message
```

```
Oops
```

*> empties then  
**overwrites** anything  
already in the file!*

```
[simben@opus ~]$ > message
```

```
[simben@opus ~]$ cat message
```

```
[simben@opus ~]$
```

## 2> (overwrites) vs 2>> (appends)

```
/home/cis90/simben $ ls bogus 2> errors
/home/cis90/simben $ cat errors
ls: cannot access bogus: No such file or directory
/home/cis90/simben $ ls crud 2> errors
/home/cis90/simben $ cat errors
ls: cannot access crud: No such file or directory
```

*2> causes the file errors to be emptied and overwritten with error output*

```
/home/cis90/simben $ ls bogus 2> errors
/home/cis90/simben $ ls crud 2>> errors
/home/cis90/simben $ cat errors
ls: cannot access bogus: No such file or directory
ls: cannot access crud: No such file or directory
/home/cis90/simben $
```

*2>> appends error output to the errors file*



## Activity

```
echo "I am $LOGNAME" > mystuff
```

```
echo -n "My terminal device is: " >> mystuff
```

```
tty >> mystuff
```

```
cat mystuff
```

*The -n option on echo suppresses the ending newline character*

*Copy and paste the output of the cat command into the chat window*

## Activity

```
echo oops > mystuff  
cat mystuff
```

*Copy and paste the output of the cat command into the chat window*

## Activity

```
> mystuff  
cat mystuff
```

*Copy and paste the output of the cat command into the chat window (better put quotes around it)*



# More redirection examples

# Example 1

## Redirecting stdout to another terminal device

/dev/pts/0

```
[simben@opus ~]$ cat names
duke
benji
star
homer
[simben@opus ~]$ tty
/dev/pts/0
[simben@opus ~]$ sort names > /dev/pts/1
[simben@opus ~]$
```

*Note, everything in UNIX is a file so we can even redirect to another terminal*

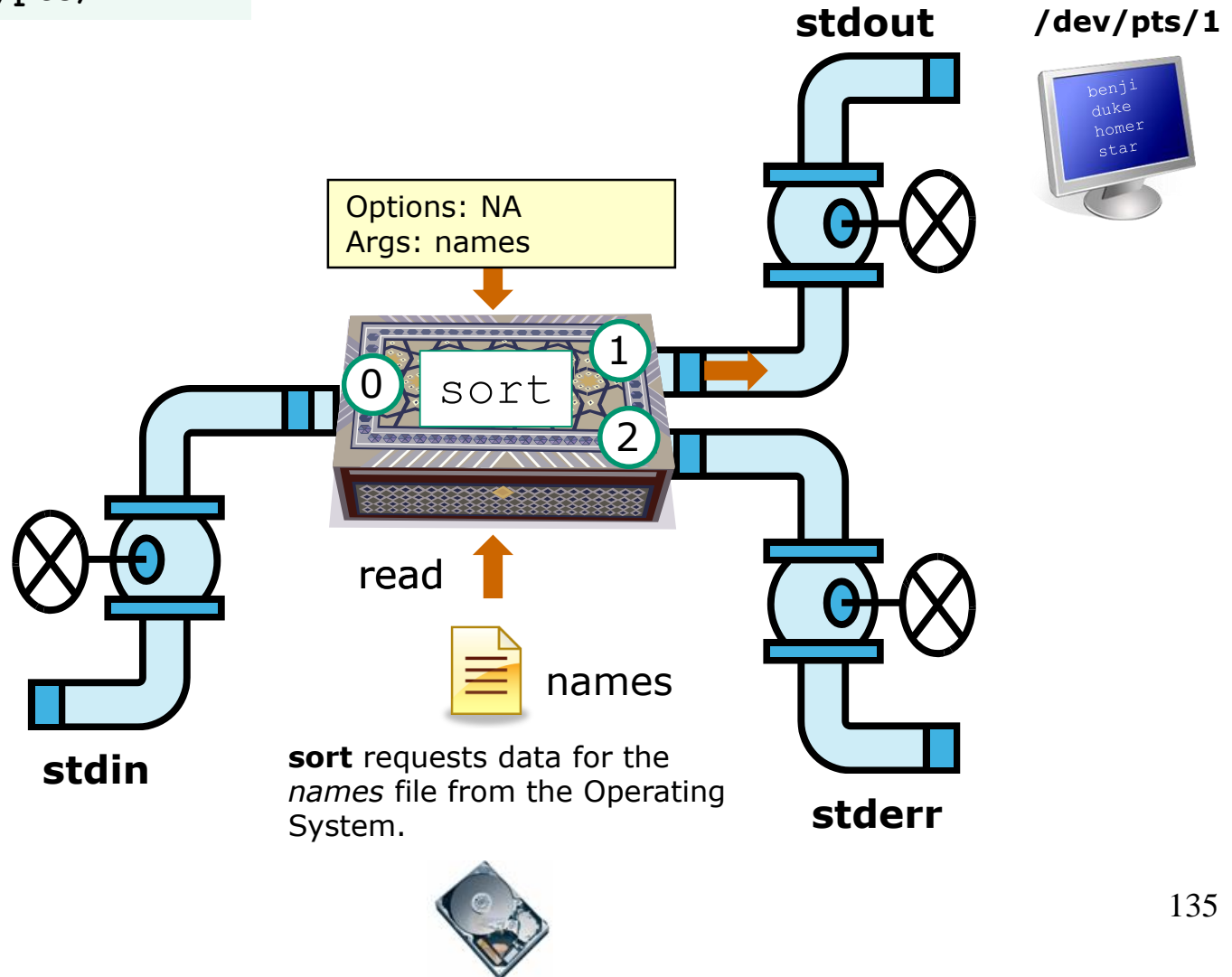
/dev/pts/1

```
[simben@opus ~]$ tty
/dev/pts/1
[simben@opus ~]$ benji
duke
homer
star
```

# Now visualize what is going on

```
$ sort names > /dev/pts/1
```

The **sort** command is loaded into memory and runs as a process. The **sort** process does NOT use **stdin** for input. Instead it uses the command line argument (**names**) parsed by the shell as input. It treats this as a file which it opens and inputs the contents to be sorted. It then writes the sorted output to **stdout** which is redirected to the terminal device **/dev/pts/1**.



## Example 2

### Redirecting stdout to a file

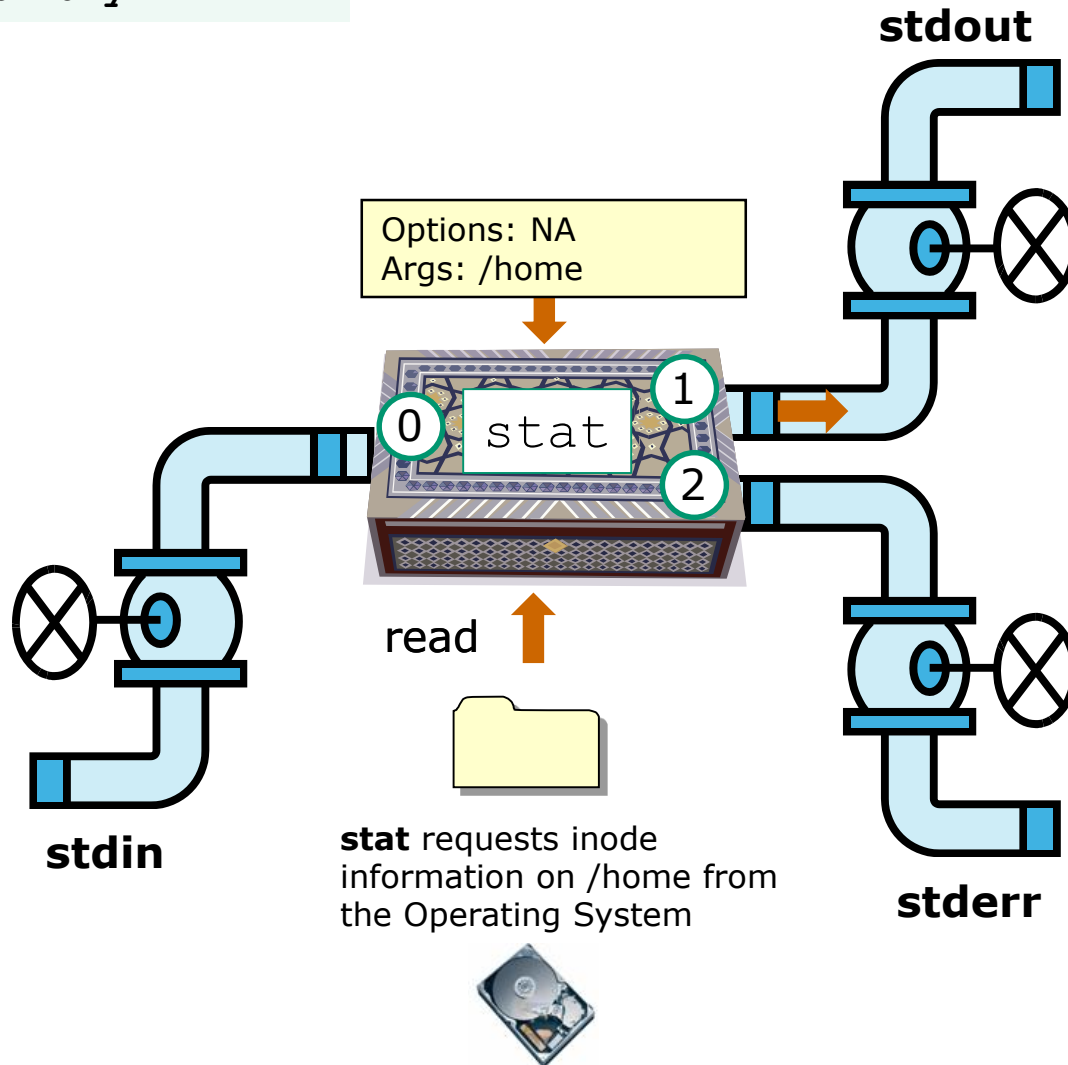
```
/home/cis90/simben $ stat /home > summary
/home/cis90/simben $ cat summary
  File: `/home'
  Size: 162          Blocks: 0          IO Block: 4096   directory
Device: fd02h/64770d Inode: 64          Links: 13
Access: (0755/drwxr-xr-x)  Uid: (   0/   root)   Gid: (   0/   root)
Context: system_u:object_r:home_root_t:s0
Access: 2018-10-15 15:45:06.788355565 -0700
Modify: 2018-10-05 15:23:47.814885578 -0700
Change: 2018-10-05 15:23:47.814885578 -0700
  Birth: -
/home/cis90/simben $
```

*Redirecting the output of the stat command to a file named summary.*

# Now visualize what is going on

```
$ stat /home > summary
```

The **stat** command is loaded into memory and runs as a process. The **stat** process does NOT use **stdin** for input. Instead it takes the command line argument (`/home`) parsed by the shell and requests inode information from the O.S. The information is formatted and output to **stdout** which is redirected to the summary file.



summary

```
File: '/home'
Size: 162          Blocks:
0          IO Block: 4096
directory
Device: fd02h/64770d  Inode:
64          Links: 13
Access: (0755/drwxr-xr-x)  Uid:
(  0/   root)  Gid: (  0/
root)
Context:
system_u:object_r:home_root_t:s
0
Access: 2018-10-15
15:45:06.788355565 -0700
Modify: 2018-10-05
15:23:47.814885578 -0700
Change: 2018-10-05
15:23:47.814885578 -0700
Birth: -
```



## Example 3

### Redirectiong stdout and stderr

```
/home/cis90/simben $ ls -l letter log bogus > listing 2> errors
```

```
/home/cis90/simben $ cat listing
```

```
-rw-r--r--. 1 simben90 cis90 1044 Jul 20 2001 letter  
-rw-r--r--. 1 simben90 cis90 832 Oct 7 15:47 log
```

```
/home/cis90/simben $ cat errors
```

```
ls: cannot access bogus: No such file or directory
```

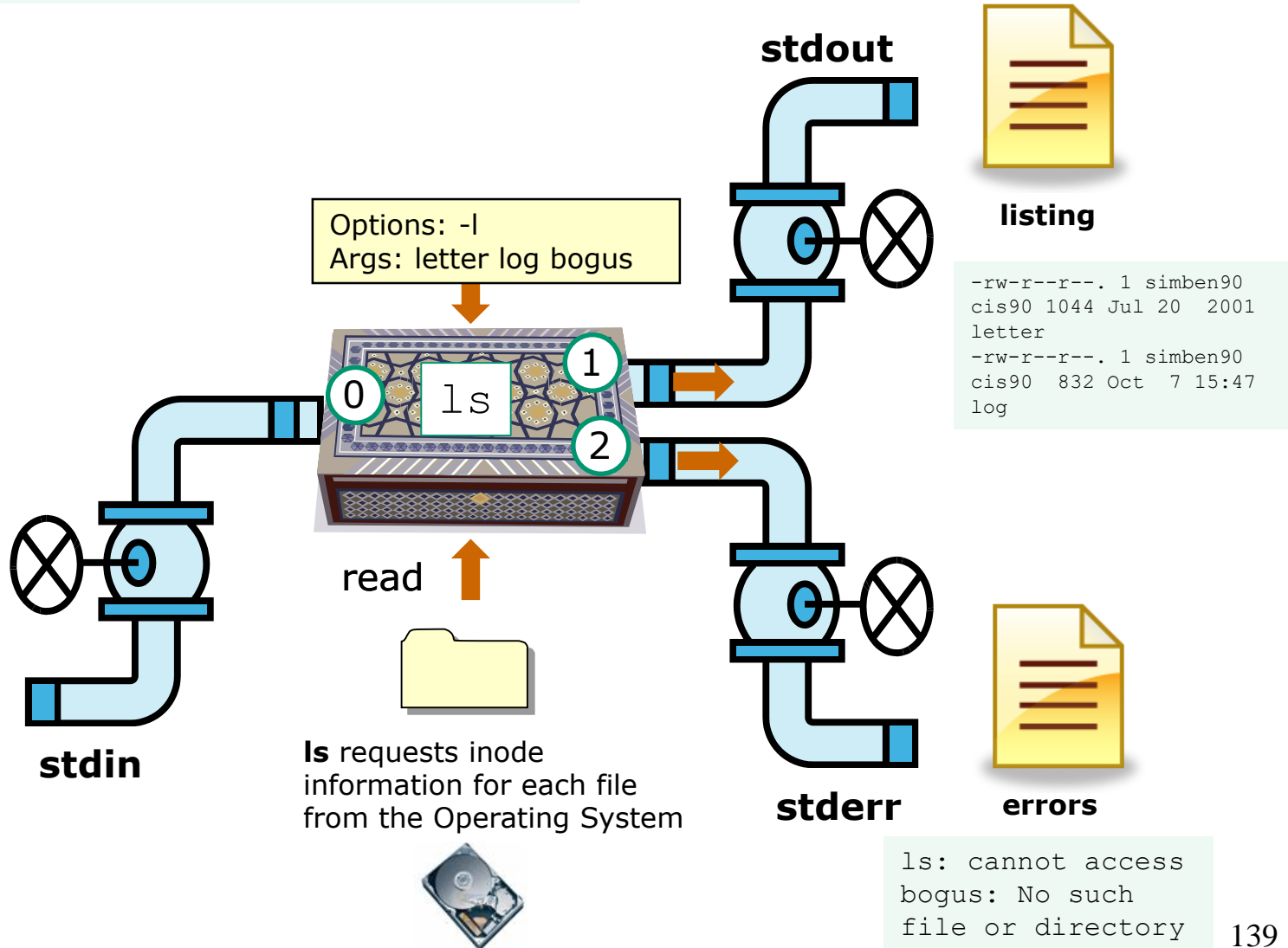
```
/home/cis90/simben $
```

*Doing a long listing on three filenames however the file named bogus does not exist*

# Now visualize what is going on

```
$ ls -l letter log bogus > listing 2> errors
```

The **ls** command is loaded into memory and runs as a process. The **ls** process does NOT use **stdin** for input. Instead it uses the command line options and arguments (-l, letter, log, bogus) parsed by the shell. *ls* obtains file information from the OS and writes a long listing to **stdout** (redirected to listing) and errors to **stderr** (redirected to errors).



# Redirection Practice

## Activity

The `bc` command reads from *stdin*. It writes computed results to *stdout* and errors to *stderr*.

```
/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
```

*Write "bc done" into the chat window when finished.*

## Activity

Redirect *stdin* to a file.

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

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

*Write "stdin redirected" into the chat window when finished.*

## Activity

Redirect *stdin* and *stdout*.

```
/home/cis90/simben $ cat math
```

```
2+2
```

```
4/0
```

```
/home/cis90/simben $ bc < math > answers
```

```
Runtime error (func=(main), adr=5): Divide by zero
```

```
/home/cis90/simben $ cat answers
```

```
4
```

*Write "stdin and stdout redirected" into the chat window when finished.*

## Activity

This time we redirect *stdin*, *stdout* and *stderr*!

```
/home/cis90/simben $ cat math
```

```
2+2
```

```
4/0
```

```
/home/cis90/simben $ bc < math > answers 2> errors
```

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

```
/home/cis90/simben $ cat answers
```

```
4
```

```
/home/cis90/simben $ cat errors
```

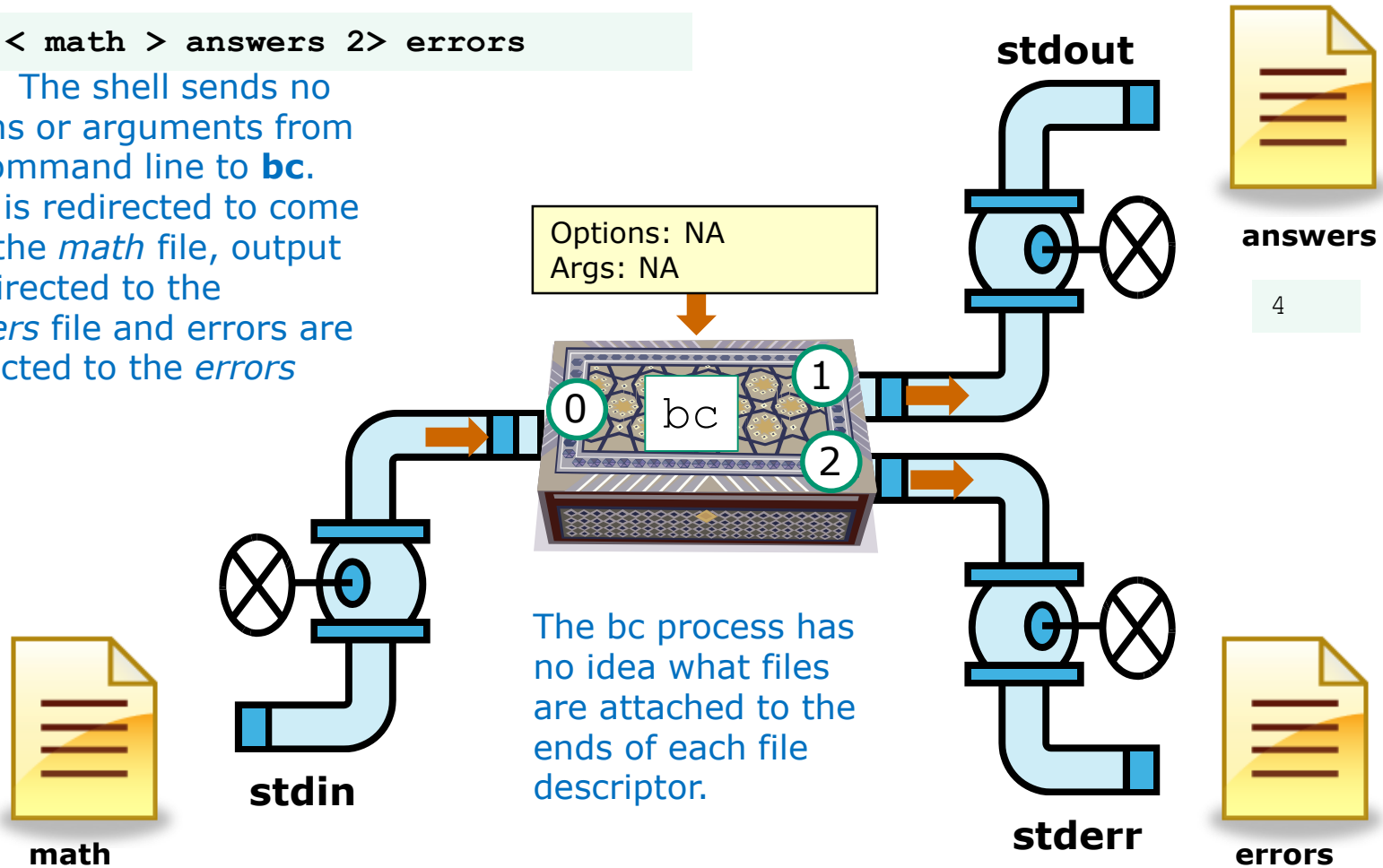
```
Runtime error (func=(main), adr=5): Divide by zero
```

*Write "all redirected" into the chat window when finished.*

# Now visualize what is going on

```
$ bc < math > answers 2> errors
```

Note: The shell sends no options or arguments from the command line to **bc**. Input is redirected to come from the *math* file, output is redirected to the *answers* file and errors are redirected to the *errors* file.



```
2+2
4/0
```

```
Runtime error (func=(main),
adr=5): Divide by zero
```



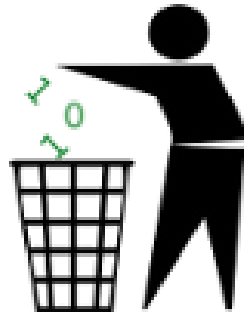


# The bit bucket `/dev/null`

## /dev/null = "bit bucket"

*A bit bucket is very handy. You can throw stuff into it and never see it again!*

<http://www.adrianmouat.com/bit-bucket/>



<http://didyouknowarchive.com/?p=1755>

*It's like having your own black hole to discard those unwanted bits into!*

# /dev/null = "bit bucket"


Whatever you redirect to /dev/null/ is gone forever

```

/home/cis90/simben $ echo Clean up your room! > orders
/home/cis90/simben $ cat orders
Clean up your room!
/home/cis90/simben $

/home/cis90/simben $ echo Clean up your room! > /dev/null
/home/cis90/simben $ cat /dev/null
/home/cis90/simben $

```



*This is how you redirect output to the bit bucket*

Корисне для  
наступного  
вікторина!

*Write "bucketed" into the chat window when finished.*

# Pipelines

# Input and Output Pipelines

Commands may be chained together in such a way that the **stdout** of one command is "piped" into the **stdin** of a second process.

## Filters

A program that both reads from **stdin** and writes to **stdout**.

## Tees

A filter program that reads **stdin** and writes it to **stdout and the file** specified as the argument.

# Input and Output

## Pipelines

Note:

Use **redirection** operators (<, >, >>, 2>) to redirect input and output from and to **files**

Use the **pipe** operator (|) to pipe output from one **command** for use as input to another **command**

# Pipeline Example

Counting the lines in the letter file

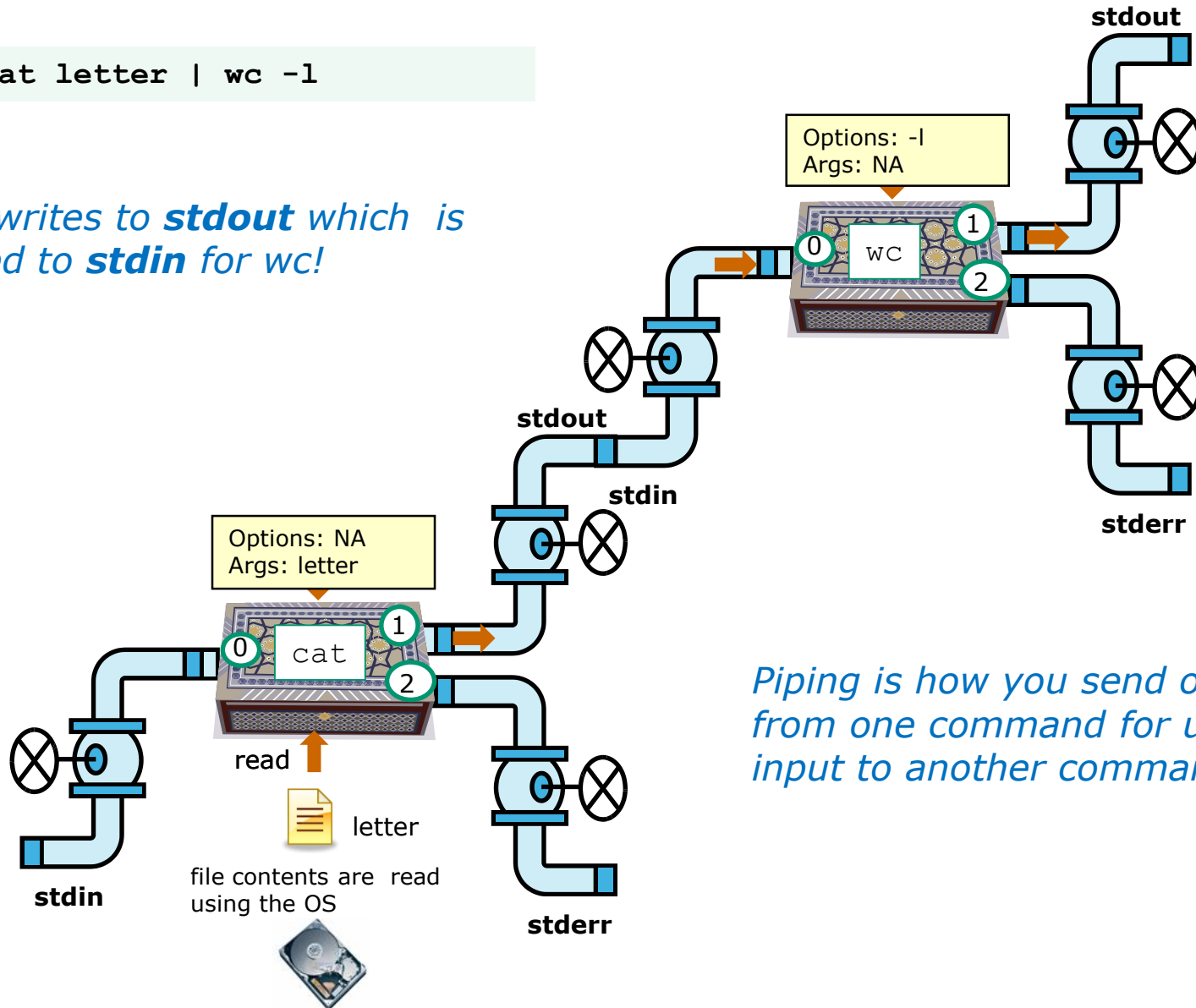
```
[simben@opus ~]$ cat letter | wc -l  
28
```

*Output from the **cat** command is piped to the **wc** command*

## Counting lines in the letter file

```
$ cat letter | wc -l
```

*cat writes to **stdout** which is piped to **stdin** for wc!*



*Piping is how you send output from one command for use as input to another command*



## You try it

Counting the lines in the letter file

```
cat letter | wc -l
```

Counting the number of Shakespeare sonnets

```
ls poems/Shakespeare/ | wc -l
```

Counting the words In Maya Angelou's poems

```
cat poems/Angelou/* | wc -w
```

*Write your counts in the chat window.*



# find command

# Find Command

## Basic syntax

(see man page for the rest of the story)

```
find <start-directory> -name <filename>  
                -type <filetype>  
                -user <username>  
                -group <groupname>  
                -exec <command> {} \;
```

Use the **find** command to find files by their name, type, owner, group (or other attributes) and optionally run a command on each of the files found.

The find command is **recursive** by default. It will start finding files at the <start directory> and includes all files and sub-directories in that branch of the file tree.

# find command with no options or arguments

The **find** command by itself lists all files in the current directory and recursively down into any sub-directories.

```
[simben@opus poems]$ find
```

```
./Blake
./Blake/tiger
./Blake/jerusalem
./Shakespeare
./Shakespeare/sonnet1
./Shakespeare/sonnet2
./Shakespeare/sonnet3
./Shakespeare/sonnet4
./Shakespeare/sonnet5
./Shakespeare/sonnet7
./Shakespeare/sonnet9
./Shakespeare/sonnet10
./Shakespeare/sonnet15
./Shakespeare/sonnet17
./Shakespeare/sonnet26
./Shakespeare/sonnet35
./Shakespeare/sonnet11
./Shakespeare/sonnet6
./Yeats
./Yeats/whitebirds
./Yeats/mooncat
./Yeats/old
./Anon
./Anon/ant
./Anon/nursery
./Anon/twister
```

*Because no start directory was specified the find command will start listing files in the current directory (poems)*

*note: reduced font size so it will fit on this slide*

```
[simben@opus poems]$
```

## find command - the starting directory

*One or more starting directories in the file tree can be specified as an argument to the find command which will list recursively all files and sub-folders from that directory and down*

```
/home/cis90/simben $ find /etc/ssh
/etc/ssh
/etc/ssh/ssh_config
/etc/ssh/ssh_host_dsa_key.pub
/etc/ssh/moduli
/etc/ssh/ssh_host_key
/etc/ssh/ssh_host_dsa_key
/etc/ssh/ssh_host_rsa_key.pub
/etc/ssh/ssh_host_rsa_key
/etc/ssh/ssh_host_key.pub
/etc/ssh/sshd_config
/home/cis90/simben $
```

*this find command will start listing files from the /etc/ssh directory*

## The find command -name option

*Since no starting directory was specified find will start in the current directory (simben90's home directory.*

*Directs the find command to only look for files whose names start with "sonnet"*

```

/home/cis90/simben $ find -name 'sonnet*'
find: `./Hidden': Permission denied
./poems/Shakespeare/sonnet10
./poems/Shakespeare/sonnet15
./poems/Shakespeare/sonnet26
./poems/Shakespeare/sonnet3
./poems/Shakespeare/sonnet35
./poems/Shakespeare/sonnet6
./poems/Shakespeare/sonnet2
./poems/Shakespeare/sonnet4
./poems/Shakespeare/sonnet1
./poems/Shakespeare/sonnet11
./poems/Shakespeare/sonnet7
./poems/Shakespeare/sonnet5
./poems/Shakespeare/sonnet9
./poems/Shakespeare/sonnet17
/home/cis90/simben $
  
```

## All those permission errors

*An error is printed for every directory lacking read permission!*

*Where to start finding files*

*only include files  
named sonnet6*

```
[simben@opus ~]$ find /home/cis90 -name sonnet6
```

```
find: /home/cis90/guest/.ssh: Permission denied
find: /home/cis90/guest/Hidden: Permission denied
/home/cis90/guest/Poems/Shakespeare/sonnet6
find: /home/cis90/guest/.gnupg: Permission denied
find: /home/cis90/guest/.gnome2: Permission denied
find: /home/cis90/guest/.gnome2_private: Permission denied
find: /home/cis90/guest/.gconf: Permission denied
find: /home/cis90/guest/.gconfd: Permission denied
find: /home/cis90/simben/Hidden: Permission denied
```

*Yuck! How  
annoying is this?*

*<snipped>*

```
find: /home/cis90/wichemic/class: Permission denied
find: /home/cis90/crivejoh/Hidden: Permission denied
/home/cis90/crivejoh/poems/Shakespeare/sonnet6
[simben@opus ~]$
```



# Redirecting find errors to the bit bucket

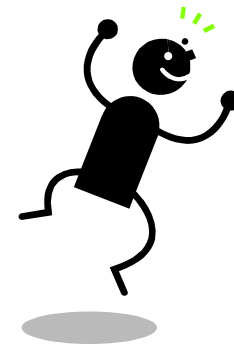
*redirecting stderr  
to the "bit bucket"*

```
[simben@opus ~]$ find /home/cis90 -name sonnet6 2> /dev/null
```

```
/home/cis90/guest/Poems/Shakespeare/sonnet6
/home/cis90/simben/poems/Shakespeare/sonnet6
/home/cis90/stanlcha/poems/Shakespeare/sonnet6
/home/cis90/seatocol/poems/Shakespeare/sonnet6
/home/cis90/wrigholi/poems/Shakespeare/sonnet6
/home/cis90/dymesdia/poems/Shakespeare/sonnet6
/home/cis90/lyonsrob/poems/Shakespeare/sonnet6
/home/cis90/ybarrser/poems/Shakespeare/sonnet6
/home/cis90/ybarrser/poems/Sonnets/sonnet6
/home/cis90/valdemar/poems/Shakespeare/sonnet6
/home/cis90/elliokat/poems/Shakespeare/sonnet6
/home/cis90/jessuwes/poems/Shakespeare/sonnet6
/home/cis90/luisjus/poems/Shakespeare/sonnet6
/home/cis90/meyerjas/poems/Shakespeare/sonnet6
/home/cis90/bergelyl/sonnet6
/home/cis90/bergelyl/poems/Shakespeare/sonnet6
/home/cis90/gardnnic/poems/Shakespeare/sonnet6
/home/cis90/mohanchi/poems/Shakespeare/sonnet6
/home/cis90/whitfbob/poems/Shakespeare/sonnet6
/home/cis90/crivejoh/poems/Shakespeare/sonnet6
[simben@opus ~]$
```

*Ahhh ... much better!*

*All the annoying error  
messages are redirected  
to the bit bucket*



*This is why we want a  
bit bucket*



# find command examples

*start finding in /  
(the top of the file tree)*

*wc counts the number of  
lines read from stdin*

```
[simben@opus ~]$ find / 2> /dev/null | wc -l
154033
```

*redirect permission  
errors into the bit  
bucket (discard them)*

*pipe the output of the find  
command as input to the wc  
command*

Корисне для  
наступного  
вікторини!

*Getting an approximate count of all the files on Opus and suppressing any permission errors*

# find command examples

```

/home/cis90/simben $ find /home -user root 2> /dev/null
/home
/home/cis175
/home/cis172
/home/cis172/computers.txt
/home/cis172/science.txt
/home/lost+found
/home/cis90/simben $

```

*The directory to start finding files*

*Redirect errors written to stderr to the bit bucket*

*The user that owns the files*

*Find all files in the /home directory that belong to the root user and discard any error messages*

# find command examples

*The directory to start finding files*

*Redirect errors to the bit bucket*

```

/home/cis90/simben $ find /home -type d -user milhom90 2> /dev/null
/home/turnin/cis90/milhom90
/home/cis90/milhom
/home/cis90/milhom/Hidden
/home/cis90/milhom/Lab2.0
/home/cis90/milhom/Miscellaneous
/home/cis90/milhom/bin
/home/cis90/milhom/Poems
/home/cis90/milhom/Poems/Shakespeare
/home/cis90/milhom/Poems/Yeats
/home/cis90/milhom/Poems/Blake
/home/cis90/milhom/Lab2.1
/home/cis90/milhom/Lab2.1/filename
/home/cis90/milhom/cis90_html
/home/cis90/milhom/cis90_html/images
/home/cis90/milhom/cis90_html/css
/home/cis90/milhom/.ssh
/home/cis90/simben $
    
```

*Only find type d files (directories)*      *Only those that belong to milhom90*

*Find all directories starting in /home that belong to milhom90 and suppress permission errors*

# find command examples

*start from "here"* →

```
[simben@opus ~]$ find . -type d -name '[BSYA]*'
find: ./Hidden: Permission denied
./poems/Blake
./poems/Shakespeare
./poems/Yeats
./poems/Anon
[simben@opus ~]$
```

*specifies directories only*

*specifies only files whose names start with a B, S, Y or A*

*Find all directories, starting from the current directory that start with a capital B, S, Y or A.*

# find command examples

*No start directory specified so start in current directory*

*file type "f" (regular)*

*file names contain the letter "k"*

*The command to run on each file found*

```
/home/cis90/simben $ find -type f -name '*k*' -exec file {} \;
find: `./Hidden': Permission denied
./edits/spellk: ASCII English text
./kshrc: ASCII text
./docs/MarkTwain: ASCII English text
./.ssh/known_hosts: ASCII text, with very long lines
/home/cis90/simben $
```

**-exec file {} \;**

*The {} are replaced by filenames as they are found*

*Escape the ; so it will be passed to the find command*

*Run the file command on all regular files found starting in the current directory whose names contain the letter "k"*

## Now you try it

*start from "here"*

*specifies only files  
whose names  
contain "town"*

```
[simben@opus-ii ~]$ find . -name '*town*'
find: ./Hidden: Permission denied
./edits/small_town
./edits/better_town
[simben@opus-ii ~]$
```

Find all files starting from your current location whose names contain "town"

*Write "towns found" in the chat window when finished.*



# Filter commmands



A command is called a "**filter**" if it can read from *stdin* and write to *stdout*

**cat** - concatenate

**grep** - "Global Regular Expression Print"

**sort** - sort

**spell** - spelling correction

**wc** - word count

**tee** - split output

**cut** - cut fields from a line

*Filters enable building useful pipelines*



# grep command

# grep command

## Basic syntax

(see man page for the rest of the story)

**grep** *<options>* "search string" *<filenames...>*

**grep -R** *<options>* "search string" *<start-directory>*

Use the **grep** command to search the **contents** of files. Use the **-R** option to do a recursive search starting from a directory

Some other useful options:

- i (case insensitive)
- w (whole word)
- v (does not contain)
- n (show line number)
- color (uses color to show matches)

# grep for text string

*string to search for*    *files to search contents of*

```

/home/cis90/simben/poems $ grep love Shakespeare/son*
Shakespeare/sonnet10:For shame deny that thou bear'st love to any,
Shakespeare/sonnet10:Shall hate be fairer lodg'd then gentle love?
Shakespeare/sonnet10:   Make thee another self for love of me,
Shakespeare/sonnet15:   And all in war with Time for love of you,
Shakespeare/sonnet26:Lord of my love, to whom in vassalage
Shakespeare/sonnet26:   Then may I dare to boast how I do love thee,
Shakespeare/sonnet3:Of his self-love, to stop posterity?
Shakespeare/sonnet3:Calls back the lovely April of her prime,
Shakespeare/sonnet4:Unthrifty loveliness, why dost thou spend
Shakespeare/sonnet5:The lovely gaze where every eye doth dwell
Shakespeare/sonnet9:   No love toward others in that bosom sits

```

*files that contain love*

*Looking for love in all the wrong places?*

Find the string "love" in Shakespeare's sonnets

## Now you try it

*Looking for love in all the wrong places?*

```
grep love poems/Shakespeare/*
```

*Write "love found" in the chat windows when finished.*

# grep the output of a grep

*string to search for*    *files to search contents of*    *string to search for in the output of the previous command*

```

[simben@opus poems]$ grep love Shakespeare/son* | grep hate
Shakespeare/sonnet10:Shall hate be fairer lodg'd then gentle love?
[simben@opus poems]$
    
```

Find all lines with both love and hate

# grep using the -n (line number) option

*string to search for*      *file to search contents of*

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

Show account in /etc/passwd for simben90

*Option to show line number*      *string to search for*      *file to search contents of*

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

*Line number of /etc/passwd*

Same as before but include line number it was found on



## grep using the -i (case insensitive) option

```
/home/cis90/simben $ grep "so" poems/Shakespeare/sonnet[345]
poems/Shakespeare/sonnet3:Thou dost beguile the world, unbless some mother.
poems/Shakespeare/sonnet3:For where is she so fair whose unear'd womb
poems/Shakespeare/sonnet3:Or who is he so fond will be the tomb,
poems/Shakespeare/sonnet5:A liquid prisoner pent in walls of glass,
```

Look for "so" in sonnet3, sonnet4 and sonnet5

*Use the -i option to make searches case insensitive*



```
/home/cis90/simben $ grep -i "so" poems/Shakespeare/sonnet[345]
poems/Shakespeare/sonnet3:Thou dost beguile the world, unbless some mother.
poems/Shakespeare/sonnet3:For where is she so fair whose unear'd womb
poems/Shakespeare/sonnet3:Or who is he so fond will be the tomb,
poems/Shakespeare/sonnet3:So thou through windows of thine age shalt see,
poems/Shakespeare/sonnet4:So great a sum of sums, yet canst not live?
poems/Shakespeare/sonnet5:A liquid prisoner pent in walls of glass,
```

Look for "so" (case insensitive) in sonnet3, sonnet4 and sonnet5

## grep using the -w (whole word) option

```
/home/cis90/simben $ grep so poems/Shakespeare/sonnet[345]
poems/Shakespeare/sonnet3:Thou dost beguile the world, unbless some mother.
poems/Shakespeare/sonnet3:For where is she so fair whose unear'd womb
poems/Shakespeare/sonnet3:Or who is he so fond will be the tomb,
poems/Shakespeare/sonnet5:A liquid prisoner pent in walls of glass,
```

Look for "so" in sonnet3, sonnet4 and sonnet5

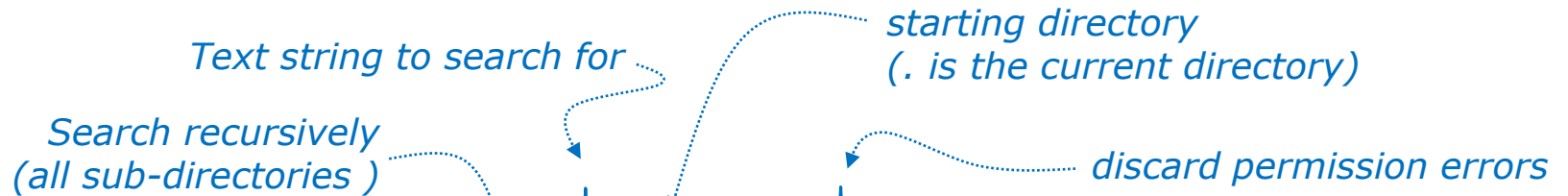
*Use the -w option for whole word only searches*

```
/home/cis90/simben $ grep -w so poems/Shakespeare/sonnet[345]
poems/Shakespeare/sonnet3:For where is she so fair whose unear'd womb
poems/Shakespeare/sonnet3:Or who is he so fond will be the tomb,
```

Look for "so" (whole word only) in sonnet3, sonnet4 and sonnet5



# grep recursively with the -R option



```
/home/cis90/simben $ grep -R kind . 2> /dev/null
./bin/enlightenment: echo "to find out what kind of file \"what_am_i\" is"
./letter:Mother, Father, kindly disregard this letter.
./trash:When two close kindred meet,
./misc/mystery: echo "to find out what kind of file \"what_am_i\" is"
./poems/Shakespeare/sonnet10:Be as thy presence is gracious and kind,
./poems/Shakespeare/sonnet10:Or to thyself at least kind-hearted prove:
./poems/Shakespeare/sonnet35: Let no unkind, no fair beseechers kill;
./poems/Yeats/mooncat:When two close kindred meet,
./poems/Anon/ant:distorted out of kind,
```

Search recursively for files containing "kind"

# grep command

## Background

Apache is the worlds most popular web server and it's installed on Opus-II. Try it, you can browse to [opus-ii.cis.cabrillo.edu](http://opus-ii.cis.cabrillo.edu).

Every Apache (httpd) configuration file must specify the location (an absolute pathname) of the documents to publish on the world wide web. This is done with the **DocumentRoot** directive. This directive is found in every Apache configuration file.

All configuration files are kept in /etc.

## Tasks

- Can you use **grep** to find the Apache configuration file?  
*Hint: use the -R option to recursively search all sub-directories*
- What are the names of the GIF file in the Apache's document root directory on Opus-II?  
*Hint: Use the ls command on the document root directory*



**ONLY**  
**If Time Allows**



# Regular Expressions

grep = **G**lobal **R**egular **E**xpression **P**rint

## Regular Expressions

- [Regular Expressions \(Goyvaerts\)](#)
- [Cheat sheet \(RexEgg\)](#)
- [Examples \(Vasili\)](#)

*Find the regular expression links on the Resources page of the website*

*or*

**Google regular expression examples**

<https://simms-teach.com/resources.php>

FYI  
only

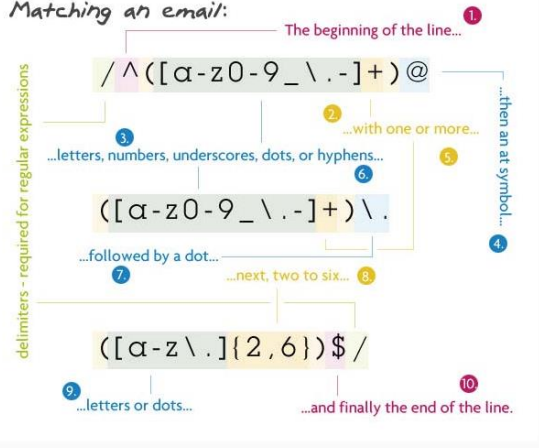
# Regular Expressions

## Regular Expressions

- Regular Expressions (Goyvaerts)
- Cheat sheet (RexEgg)
- Examples (Vasili)

### 5. Matching an Email

Matching an email:



Pattern:

```
1 | /^[a-z0-9_ \ .-]+@[ \da-z \ .-]+ \ . ([a-z \ .]{2,6})$/
```

<https://code.tutsplus.com/tutorials/8-regular-expressions-you-should-know--net-6149>

Find all the email addresses in /usr/share/doc

```
grep -Er "([\a-z0-9_ \ .-]+)@[ \da-z \ .-]+ \ . ([a-z \ .]{2,6})/" /usr/share/doc 2> /dev/null
```

Note we stripped off the leading / ^ and trailing \$ / from the example to find email address embedded in other text strings

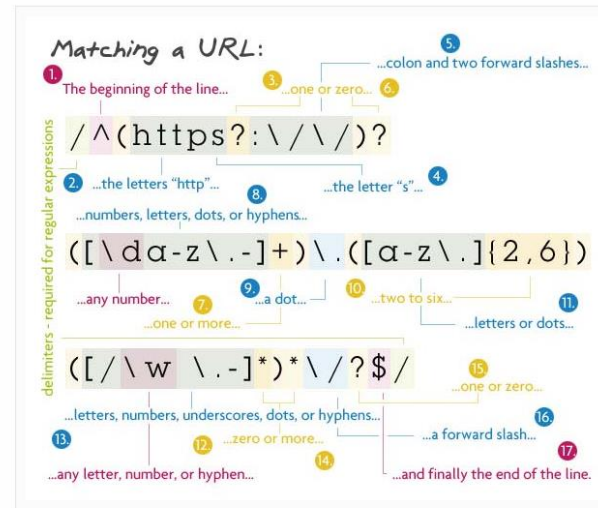
FYI  
only

# Regular Expressions

## Regular Expressions

- Regular Expressions (Goyvaerts)
- Cheat sheet (RexEgg)
- Examples (Vasili)

### 6. Matching a URL



<https://code.tutsplus.com/tutorials/8-regular-expressions-you-should-know--net-6149>

Find all the https URLs in /usr/share/doc

```
grep -Er "(https:\\/\\/)([\\da-z\\.-]+)\\.([a-z\\.]{2,6})([\\/\\w \\.-]*)*\\/?" /usr/share/doc 2> /dev/null
```

Note we stripped off the leading /<sup>^</sup> and trailing \$/ from the example to find URLs embedded in other text strings. The ?'s were also stripped so to make the "https" match mandatory.



# spell command



# spell command

## Basic syntax

(see man page for the rest of the story)

**spell** *<filepath>*

**spell** *<filepath>* *<filepath>* ...

The **spell** command is used to check spelling of words in one or more text files



# spell command

Run a spell check on the magna\_carta file

```
/home/cis90/simben $ cd docs  
/home/cis90/simben/docs $ ls  
magna_carta MarkTwain policy  
/home/cis90/simben/docs $ spell magna_carta  
Anjou  
Arundel  
Aymeric  
Bergh  
Daubeny  
de  
honour  
kingdon  
Pandulf  
Poitou  
Poppeley  
seneschal  
subdeacon  
Warin
```

*The spell command will  
show any words not  
found in the dictionary.*

# spell command

Count the number of misspelled words in the magna\_carta file

*The -l option instructs the **wc** command to just count the number of lines*

```
/home/cis90/simben/docs $ spell magna_carta | wc -l  
14
```

*Pipe the output of the **spell** command (the misspelled words) into the input of the **wc** command*

## Activity

```
/home/cis90/simben $ cat edits/spellk  
Spell Check
```

```
Eye halve a spelling chequer  
It came with my pea sea  
It plainly marques four my revue  
Miss steaks eye kin knot sea.  
Eye strike a key and type a word  
And weight four it two say  
Weather eye am wrong oar write  
It shows me strait a weigh.  
As soon as a mist ache is maid  
It nose bee fore two long  
And eye can put the error rite  
Its rare lea ever wrong.  
Eye have run this poem threw it  
I am shore your pleased two no  
Its letter perfect awl the weigh  
My chequer tolled me sew.
```

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

*How many misspelled  
word are in your spellk  
file?*

*Write your answer in the  
chat window.*



# tee command

# tee command

## Basic syntax

(see man page for the rest of the story)

**tee** <filepath>

The **tee** command, a filter, reads from **stdin** and writes to **stdout** AND to the file specified as the argument.

# tee command

*For example, the following command sends a sorted list of the current users logged on to the system to the screen, and saves an unsorted list to a file named users.*

```
/home/cis90/simben $ who | tee users | sort
caumar98 pts/5      2014-03-17 17:29 (75.140.158.6)
caumar98 pts/6      2014-03-17 17:41 (75.140.158.6)
chejul98 pts/1      2014-03-17 19:42 (acbe4f9e.ipt.aol.com)
goojun172 pts/7     2014-03-17 19:53 (c-67-169-144-100.hsd1.ca.comcast.net)
hovdav98 pts/2      2014-03-16 14:48 (c-76-126-1-130.hsd1.ca.comcast.net)
mmatera pts/4        2014-03-13 16:06 (2607:f380:80f:f828:e108:c48e:9e1a:57ff)
rsimms pts/0        2014-03-17 09:40 (2001:470:1f05:9b3:3044:7820:6ce0:8a4)
/home/cis90/simben $
```

```
/home/cis90/simben $ cat users
rsimms pts/0        2014-03-17 09:40 (2001:470:1f05:9b3:3044:7820:6ce0:8a4)
chejul98 pts/1      2014-03-17 19:42 (acbe4f9e.ipt.aol.com)
hovdav98 pts/2      2014-03-16 14:48 (c-76-126-1-130.hsd1.ca.comcast.net)
mmatera pts/4        2014-03-13 16:06 (2607:f380:80f:f828:e108:c48e:9e1a:57ff)
caumar98 pts/5      2014-03-17 17:29 (75.140.158.6)
caumar98 pts/6      2014-03-17 17:41 (75.140.158.6)
goojun172 pts/7     2014-03-17 19:53 (c-67-169-144-100.hsd1.ca.comcast.net)
/home/cis90/simben $
```

# tee command

```
/home/cis90/simben $ head edits/spellk
Spell Check
```

```
Eye halve a spelling chequer
It came with my pea sea
It plainly marques four my revue
Miss steaks eye kin knot sea.
Eye strike a key and type a word
And weight four it two say
Weather eye am wrong oar write
```

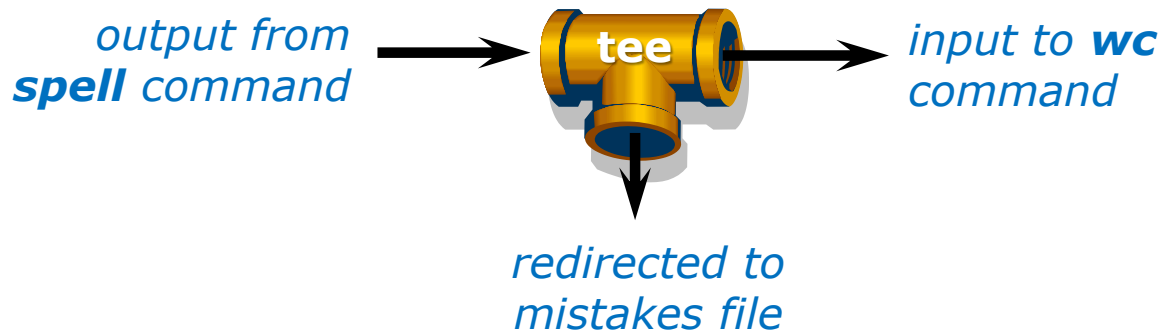
*The misspelled words from spell are piped to the tee command*

*The tee command copies the misspelled words to stdout and to the file named mistakes*

```
/home/cis90/simben $ spell edits/spellk | tee mistakes | wc -l
1
```

```
/home/cis90/simben $ cat mistakes
chequer
```

*The wc command counts the misspelled words*



# cut command



# cut command

## Basic syntax

(see man page for the rest of the story)

**cut -f** *<num>* **-d** "*<delimiter-character>*" *<pathname>*

**cut -c** *<start column>*-*<end column>* *<pathname>*

*The **cut** command can cut text from a line by delimited fields or by a range of columns.*

# cut command

(cut text using delimited fields)

```
[rsimms@oslab ~]$ grep $LOGNAME /etc/passwd
rsimms:x:201:503:Rich Simms:/home/rsimms:/bin/bash
```

1<sup>st</sup>  
field

2<sup>nd</sup>  
field

3<sup>rd</sup>  
field

4<sup>th</sup>  
field

5<sup>th</sup>  
field

6<sup>th</sup>  
field

7<sup>th</sup>  
field

```
[rsimms@oslab ~]$ grep $LOGNAME /etc/passwd | cut -f 7 -d ":"
/bin/bash
```

Cut the 7<sup>th</sup> field

Using ":" as the delimiter

# cut command

(cut text by column numbers)

```
/home/cis90/simben $ ls -l letter
-rw-r--r--. 1 simben90 cis90 1044 Jul 20 2001 letter
123456789012345678901234567890123456789012345678901234567890
  ^           ^
  |           |
column 2     column 10
```

```
/home/cis90/simben $ ls -l letter | cut -c 2-10
rw-r--r--
Cut columns
2 through 10
```

```
/home/cis90/simben $ perm=$(ls -l letter | cut -c 2-10)
This puts the output of the pipeline
above into a variable named perm
```

```
/home/cis90/simben $ echo The permissions on letter are $perm
The permissions on letter are rw-r--r--
```

*Which we can use to  
build a custom message*

# Pipeline Practice

## Class Exercise

### Pipeline Tasks

### Background

The **last** command searches through `/var/log/wtmp` and prints out a list of users logged in since that file was created.

### Task

Can you see the last times you were logged in on a Wednesday and then count them?

```
last | grep $LOGNAME
```

```
last | grep $LOGNAME | grep "Wed"
```

```
last | grep $LOGNAME | grep "Wed" | wc -l
```

How many times did you log in on a Wednesday?  
Write your answer in the chat window.

## Class Exercise

### Pipeline Tasks

### Background

The **cut** command can cut a field out of a line of text where each field is delimited by some character.

The */etc/passwd* file uses the ":" as the delimiter between fields. The 5<sup>th</sup> field is a comment field for the user account.

### Task

Build up a pipeline, one pipe at a time:

```
cat /etc/passwd
```

```
cat /etc/passwd | grep $LOGNAME
```

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

What gets printed with the last pipeline?  
Write your answer in the chat window.



**ONLY**  
**If Time Allows**

# Permissions

“The rest of the story”

- Special Permissions
- ACLs
- Extended Attributes
- SELinux



*This module is for your information only. We won't use this in CIS 90 but its good to know they exist. More in CIS 191, 192 and 193*





## Special Permissions

**Sticky bit** - used on directories, e.g. /tmp, so that only owners can rename or remove files even though other users may have write permission on the directory.

**SetUID or SetGID** - allows a user to run an program file with the permissions of the file's owner (Set User ID) or the file's group (Set Group ID). Examples include **ping** and **passwd** commands.



# Special Permissions

**Sticky bit** - used on directories, e.g. /tmp, so that only owners can rename or remove files even though other users may have write permission on the directory.

```
/home/cis90/simben $ ls -ld /tmp
drwxrwxrwt. 3 root root 4096 Oct 16 16:13 /tmp
```

*green background with black text*

```
/home/cis90/simben $ mkdir tempdir
/home/cis90/simben $ chmod 777 tempdir/
/home/cis90/simben $ ls -ld tempdir/
drwxrwxrwx. 2 simben90 cis90 4096 Oct 16 15:25 tempdir/
```

*green background with blue text*

```
/home/cis90/simben $ chmod 1777 tempdir
/home/cis90/simben $ ls -ld tempdir/
drwxrwxrwt. 2 simben90 cis90 4096 Oct 16 15:25 tempdir/
```

*set sticky bit*

*sticky bit set*

*green background with black text*



## Special Permissions

**SetUID or SetGID** - allows a user to run a program file with the permissions of the file's owner (Set User ID) or the file's group (Set Group ID). Examples include **ping** and **passwd** commands.

```
/home/cis90/simben $ ls -l /bin/ping /usr/bin/passwd
-rwsr-xr-x. 1 root root 36892 Jul 18 2011 /bin/ping
-rwsr-xr-x. 1 root root 25980 Feb 22 2012 /usr/bin/passwd
```

*red background  
with gray text*

```
/home/cis90/simben $ echo banner Hola > hola; chmod +x hola; ls -l hola
-rwxrwxr-x. 1 simben90 cis90 12 Oct 16 16:45 hola
```

```
/home/cis90/simben $ chmod 4775 hola
/home/cis90/simben $ ls -l hola
-rwsrwxr-x. 1 simben90 cis90 12 Oct 16 16:45 hola
/home/cis90/simben $ chmod 2775 hola
/home/cis90/simben $ ls -l hola
-rwxrwsr-x. 1 simben90 cis90 12 Oct 16 16:45 hola
```



## ACLs (Access Control Lists)

**ACLs** - offer a finer granularity of control allowing additional permissions to be set for specific users or groups.



## ACLs (Access Control Lists)

**ACLs** - offer a finer granularity of control allowing additional permissions to be set for specific users or groups.

```

/home/cis90/simben $ echo yabadabadoo > yogi
/home/cis90/simben $ chmod 400 yogi
/home/cis90/simben $ ls -l yogi
-r-----. 1 simben90 cis90 12 Oct 16 17:02 yogi

/home/cis90/simben $ getfacl yogi
# file: yogi
# owner: simben90
# group: cis90
user::r--
group:---
other:---
    
```

*Create a file and set permissions to 400*

*Use **getfacl** to show ACLs*

```

[milhom90@oslab ~]$ cat ../simben/yogi
cat: ../simben/yogi: Permission denied
    
```

*Homer, a member of the cis90 group can't read the file*

```

[rodduk90@oslab ~]$ cat ../simben/yogi
cat: ../simben/yogi: Permission denied
    
```

*Duke, a member of the cis90 group can't read the file either*



# ACLs (Access Control Lists)

*Let's give special permissions to one user*

```

/home/cis90/simben $ setfacl -m u:milhom90:rw yogi
/home/cis90/simben $ ls -l yogi
-r--rw---+ 1 simben90 cis90 12 Oct 16 17:02 yogi
/home/cis90/simben $ getfacl yogi
# file: yogi
# owner: simben90
# group: cis90
user::r--
user:milhom90:rw-
group:---
mask::rw-
other:---
    
```

*modify*

*Allow milhom90 to have read/write access*

```

[milhom90@oslab ~]$ cat ../simben/yogi
yabadabadoo
    
```

*Homer can now read the file*

```

[rodduk90@oslab ~]$ cat ../simben/yogi
cat: ../simben/yogi: Permission denied
    
```

*But not Duke*



# ACLs (Access Control Lists)

*Let's remove the special permissions to that user*

*remove all base ACLs*

```

/home/cis90/simben $ setfacl -b yogi
/home/cis90/simben $ ls -l yogi
-r-----. 1 simben90 cis90 12 Oct 16 17:02 yogi
/home/cis90/simben $ getfacl yogi
# file: yogi
# owner: simben90
# group: cis90
user::r--
group:----
other:----
    
```

*Remove all ACLs on yogi file*

```

[milhom90@oslab ~]$ cat ../simben/yogi
cat: ../simben/yogi: Permission denied
    
```

*Now Homer can't read it again*

```

[rodduk90@oslab ~]$ cat ../simben/yogi
cat: ../simben/yogi: Permission denied
    
```

*Same for Duke*



## Extended File Attributes

**Extended Attributes** - the root user can set some extended attribute bits to enhance security.



## Extended File Attributes

**FYI**  
only

*Let's use extended file attributes to totally lock down a file against changes, even by its owner!*

```
/home/cis90/simben $ echo yabadabadoo > yogi
/home/cis90/simben $ ls -l yogi
-rw-rw-r--. 1 simben90 cis90 12 Oct 16 17:29 yogi
```

*Create a sample file to work on*

*The root user sets the **immutable bit (i)** so Benji cannot remove his own file*

```
[root@oslab ~]# lsattr /home/cis90/simben/yogi
-----e- /home/cis90/simben/yogi
[root@oslab ~]# chattr +i /home/cis90/simben/yogi
[root@oslab ~]# lsattr /home/cis90/simben/yogi
----i-----e- /home/cis90/simben/yogi
```

```
/home/cis90/simben $ ls -ld ~
drwxr-xr-x. 17 simben90 cis90 4096 Oct 16 17:29 /home/cis90/simben
/home/cis90/simben $ rm yogi
rm: remove write-protected regular file `yogi'? yes
rm: cannot remove `yogi': Operation not permitted
```

!!



## Extended File Attributes

**Extended Attributes** - the root user can set some extended attribute bits to enhance security.

*The root user removes the **immutable bit (i)** so Benji can remove his own file again*

```
[root@oslab ~]# chattr -i /home/cis90/simben/yogi
[root@oslab ~]# lsattr /home/cis90/simben/yogi
-----e- /home/cis90/simben/yogi
```

```
/home/cis90/simben $ ls -ld ~
drwxr-xr-x. 17 simben90 cis90 4096 Oct 16 17:29 /home/cis90/simben
/home/cis90/simben $ rm yogi
/home/cis90/simben $
```



## Extended File Attributes

*Let's use extended file attributes to allow the file to be appended (but still not emptied or removed)*

```
/home/cis90/simben $ ls -l yogi
-rw-rw-r--. 1 simben90 cis90 12 Oct 16 17:41 yogi
```

*The root user sets the **append only bit (a)** so Benji can only append to his file*

```
[root@oslab ~]# lsattr /home/cis90/simben/yogi
-----e- /home/cis90/simben/yogi
[root@oslab ~]# chattr +a /home/cis90/simben/yogi
[root@oslab ~]# lsattr /home/cis90/simben/yogi
-----a-----e- /home/cis90/simben/yogi
```

```
/home/cis90/simben $ rm yogi
rm: cannot remove `yogi': Operation not permitted
/home/cis90/simben $ > yogi
-bash: yogi: Operation not permitted
/home/cis90/simben $ echo yowser >> yogi
/home/cis90/simben $
```



## SELinux context

**SELinux** - Security Enhanced Linux. SELinux is a set of kernel modifications that provide Mandatory Access Control (MAC). In MAC-enabled systems there is a strict set of security policies for all operations which users cannot override. The primary original developer of SELinux was the NSA (National Security Agency).



## SELinux context

*Use the Z option on the ls command to show the SELinux context on a file*

```
[root@oslab selinux]# ls -lZ test*
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 test01.html
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 test02.html
```

*user*
*role*
*type*
*level*



## SELinux context

*Create two identical web pages with identical permissions*

```
[root@oslab selinux]# cp test01.html test02.html
cp: overwrite `test02.html'? yes
```

```
[root@oslab selinux]# ls -lZ test*
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 test01.html
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 test02.html
```

*Use chcon command to change the SELinux context on one file*

```
[root@oslab selinux]# chcon -v -t home_root_t test02.html
changing security context of `test02.html'
```

```
[root@oslab selinux]# ls -lZ test*
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 test01.html
-rw-r--r--. root root unconfined_u:object_r:home_root_t:s0 test02.html
```

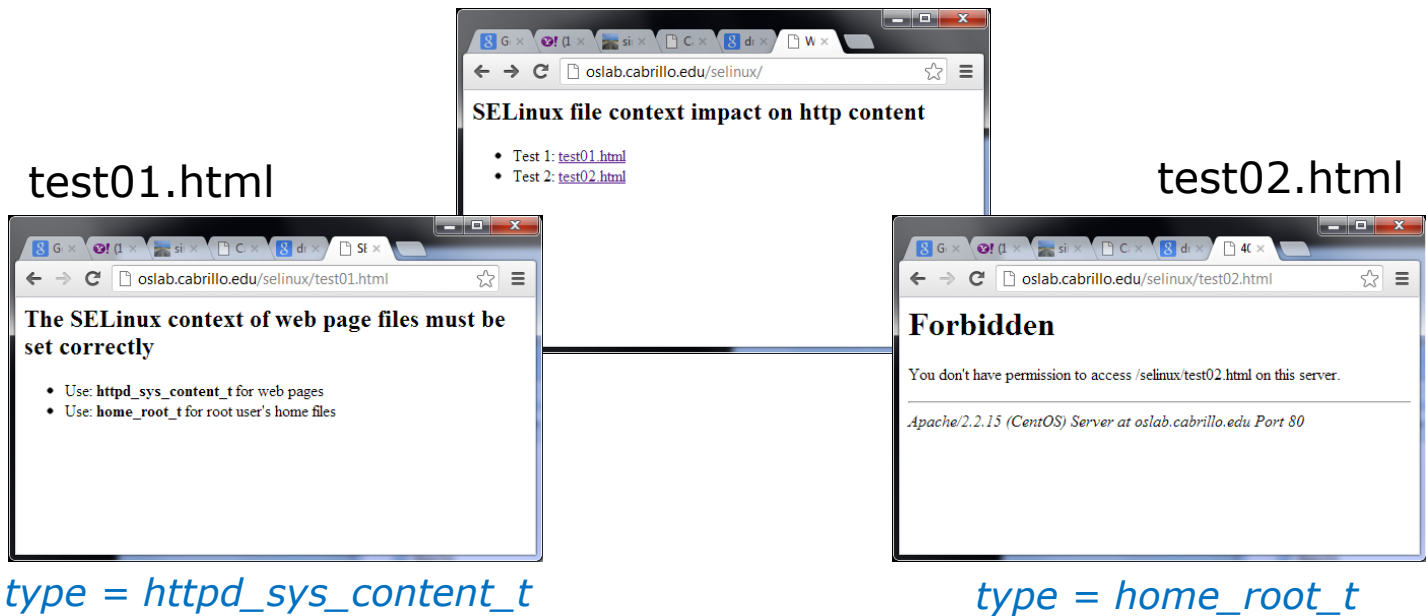
*Note, the root user's home files are not appropriate web content*



# SELinux context

*SELinux won't let Apache publish a file with an inappropriate context*

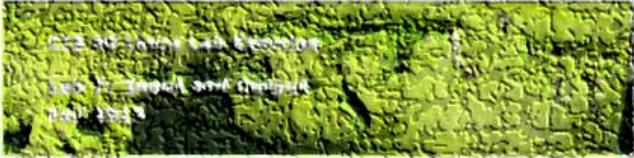
```
[root@oslab selinux]# ls -lZ test*
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 test01.html
-rw-r--r--. root root unconfined_u:object_r:home_root_t:s0 test02.html
[root@oslab selinux]#
```



# Assignment







**Lab 7: Input and Output**

The goal of this lab is to gain proficiency in using I/O redirection to perform tasks on the system. You will combine commands you have learned in this course using shell redirection, pipes and tee to perform a variety of tasks on the system.

**Preparation**

- Be sure to make the changes to your home directory asked for in Lab 6. This lab assumes the new names and directory structures.
- Slides Lesson 8 slides: <http://simms-teach.com/cis90calendar.php>
- Check the forum for news on this lab: <http://oslab.cis.cabrillo.edu/forum/>
- For additional assistance come to the CIS Labs: <http://webhawks.org/~cislab/>

**Prerequisites**

Log on to Opus so that you have a command line shell at your service. Be sure you are in your home directory in case this lab. We are going to experiment with how commands get their input and what they do with their output. Then we will perform a series of tasks by combining commands together and saving the output to a file.

**The find command**

The syntax of the find command is:

```
find [starting-directory] [-name filename] [-user username]
```

When the `-name` option and its argument are omitted all files are displayed.

1. Find all the files under your home directory by issuing the command:  
`find ~`
2. Find all the files named `old` that are somewhere in or below your parent directory using the command:  
`find .. -name old`  
Were there any error messages?
3. Filter out the error messages by redirecting `stderr` to a file called `errors` in your home directory:

## Lab 7

*Note: The **submit** and **verify** scripts are no longer used in the remaining labs.*

*Instead of **submit** you will copy your work into a turnin directory.*

*Instead of **verify** you will get an email on Opus-II with the status of your submission.*



# Wrap up

New commands:

find

find files or content

grep

look for text strings

last

show last logins

sort

perform sorts

spell

spell checking

tee

save output to a file

wc

count lines or words in a file

## Next Class

Assignment: Check Calendar Page on web site to see what is due next week.

Lab 7

Quiz questions for next class:

- How do you redirect error messages to the bit bucket?
- What command could you use to get an approximate count of all the files on Opus and ignore the permission errors?
- For **sort dognames > dogsinorder** where does the sort process obtain the actual names of the dogs to sort?
  - a) stdin
  - b) the command line
  - c) directly from the file dognames

# Backup



# Permissions Review

# File Permissions

## Binary

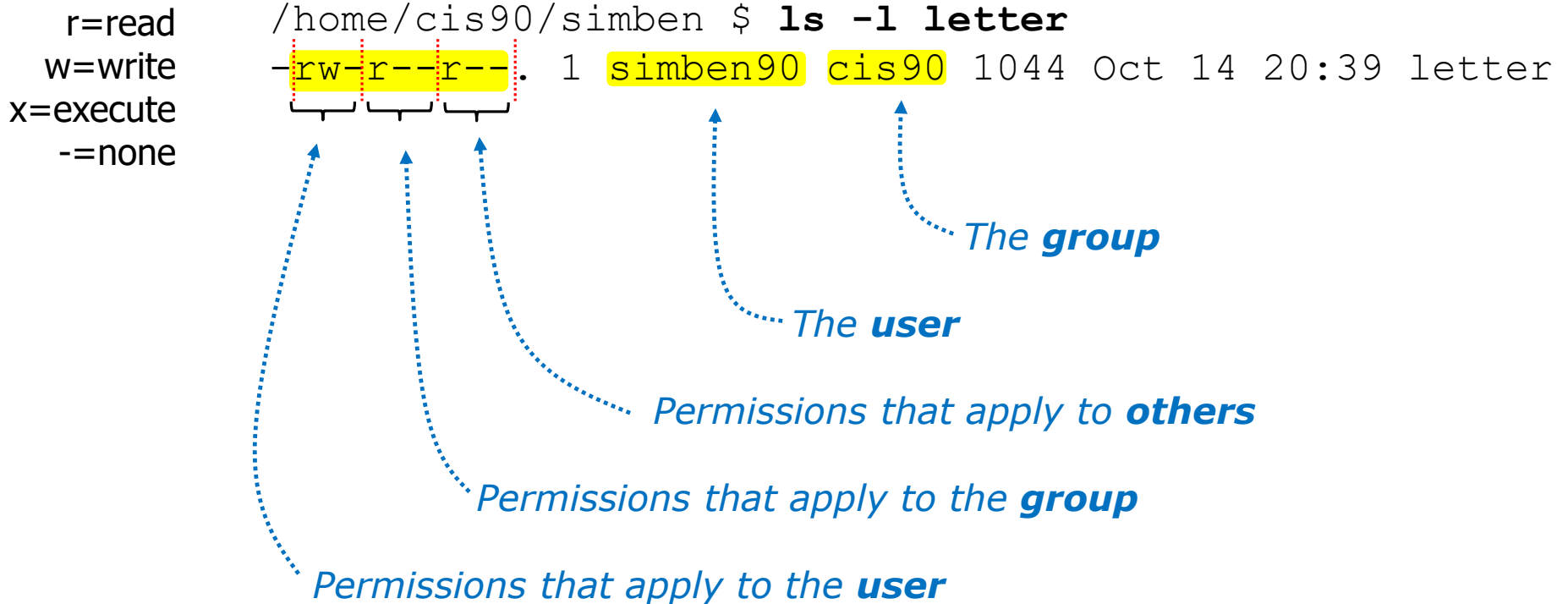
*Permissions are stored internally using binary numbers and they can be specified using decimal numbers*

rwX	Binary	Convert	Decimal
- - -	0 0 0	0 + 0 + 0	0
- - X	0 0 1	0 + 0 + 1	1
- W -	0 1 0	0 + 2 + 0	2
- W X	0 1 1	0 + 2 + 1	3
r - -	1 0 0	4 + 0 + 0	4
r - X	1 0 1	4 + 0 + 1	5
r W -	1 1 0	4 + 2 + 0	6
r W X	1 1 1	4 + 2 + 1	7

r (read) is the 4's column  
w (write) is the 2's column  
x (execute) is the 1's column

# File Permissions

*An example long listing*





# File Permissions

```
/home/cis90/simben $ ls -l letter
-rw-r--r--. 1 simben90 cis90 1044 Oct 14 20:39 letter
```

user (owner)			group			others		
r	w	-	r	-	-	r	-	-
read	write	execute	read	write	execute	read	write	execute

The permissions on letter:  
 The **user** *simben90* has read and write permission  
 The **group** *cis90* has read permission  
 All **others** have read permission

*Use long listings to show permissions*

# File Permissions

*Use long listings to show permissions*

r=read  
 w=write  
 x=execute  
 -=none

```

/home/cis90/simben $ ls -l letter
-rw-r--r-- 1 simben90 cis90 1044 Oct 14 20:39 letter
  
```

*Permissions that apply to the **user***  
*Permissions that apply to the **group***  
*Permissions that apply to **others***  
*The **user***  
*The **group***

Does the simben90 user have execute permission on the letter file?  
*Type answer in chat window*

# File Permissions

*Use long listings to show permissions*

r=read  
 w=write  
 x=execute  
 -=none

```

/home/cis90/simben $ ls -l letter
-rw-r--r-- 1 simben90 cis90 1044 Oct 14 20:39 letter
  
```

*Permissions that apply to the **user***  
*Permissions that apply to the **group***  
*Permissions that apply to **others***  
*The **user***  
*The **group***

Does the simben90 user have execute permission on the letter file?

*No*

# File Permissions

*Use long listings to show permissions*

r=read  
 w=write  
 x=execute  
 -=none

```

/home/cis90/simben $ ls -l letter
-rw-r--r-- 1 simben90 cis90 1044 Oct 14 20:39 letter
  
```

*Permissions that apply to the **user***  
*Permissions that apply to the **group***  
*Permissions that apply to **others***  
*The **user***  
*The **group***

Does the zamhum90 user have write permission on the letter file?  
*Type answer in chat window*

# File Permissions

*Use long listings to show permissions*

r=read  
 w=write  
 x=execute  
 -=none

```

/home/cis90/simben $ ls -l letter
-rw-r--r-- 1 simben90 cis90 1044 Oct 14 20:39 letter
  
```

*Permissions that apply to the **user***  
*Permissions that apply to the **group***  
*Permissions that apply to **others***  
*The **user***  
*The **group***

Does the zamhum90 user have write permission on the letter file?

*No*

# File Permissions

*Use long listings to show permissions*

r=read  
 w=write  
 x=execute  
 -=none

```

/home/cis90/simben $ ls -l letter
-rw-r--r-- 1 simben90 cis90 1044 Oct 14 20:39 letter
  
```

*Permissions that apply to the **user***  
*Permissions that apply to the **group***  
*Permissions that apply to **others***  
*The **user***  
*The **group***

Does the zamhum90 user have read permission on the letter file?  
*Type answer in chat window*

# File Permissions

*Use long listings to show permissions*

r=read  
 w=write  
 x=execute  
 -=none

```

/home/cis90/simben $ ls -l letter
-rw-r--r-- 1 simben90 cis90 1044 Oct 14 20:39 letter
  
```

*Permissions that apply to the **user***  
*Permissions that apply to the **group***  
*Permissions that apply to **others***  
*The **user***  
*The **group***

Does the zamhum90 user have read permission on the letter file?

Yes

# File Permissions

*Use long listings to show permissions*

r=read  
 w=write  
 x=execute  
 -=none

```

/home/cis90/simben $ ls -l letter
-rw-r--r--. 1 simben90 cis90 1044 Oct 14 20:39 letter
  
```

*Permissions that apply to the **user***  
*Permissions that apply to the **group***  
*Permissions that apply to **others***  
*The **user***  
*The **group***

Does the smimat172 user have read permission on the letter file?  
*Type answer in chat window*



# File Permissions

*Use long listings to show permissions*

r=read  
 w=write  
 x=execute  
 -=none

```

/home/cis90/simben $ ls -l letter
-rw-r--r-- 1 simben90 cis90 1044 Oct 14 20:39 letter
  
```

*Permissions that apply to the **user***  
*Permissions that apply to the **group***  
*Permissions that apply to **others***  
*The **user***  
*The **group***

Does the smimat172 user have read permission on the letter file?

Yes



## Tools for managing permissions

**chown** - Changes the ownership of a file. (Only the superuser has this privilege)

**chgrp** - Changes the group of a file. (Only to groups that you belong to)

**chmod** - Changes the file mode "permission" bits of a file.

- Numeric: **chmod 640 letter** (sets the permissions)
- Mnemonic: **chmod ug+rw letter** (changes the permissions)  
**u**=user(owner), **g**=group, **o**=other  
**r**=read, **w**=write, **x**=execute

**umask** - Allows specific permissions to be removed on future newly created files and directories



## Tools for managing permissions

### chown

- Changes the ownership of a file. (Only the superuser has this privilege)
- Syntax: **chown <owner> <pathname>**

```
/home/cis90/simben $ ls -l letter  
-rw-r--r--. 1 simben90 cis90 1044 Oct 14 20:39 letter
```

```
/home/cis90/simben $ chown rsimms letter  
chown: changing ownership of `letter': Operation not permitted
```

*Only root (superuser) can change the ownership of a file*



## Tools for managing permissions

### chgrp

- Changes the group of a file. (Only to groups the owner belongs to)
- Syntax: **chgrp <group> <pathname>**

```
/home/cis90/simben $ ls -l letter  
-rw-r--r--. 1 simben90 cis90 1044 Oct 14 20:39 letter
```

```
/home/cis90/simben $ groups  
cis90 users
```

```
/home/cis90/simben $ chgrp users letter
```

```
/home/cis90/simben $ ls -l letter  
-rw-r--r--. 1 simben90 users 1044 Oct 14 20:39 letter
```

*The owner can change the group to any he/she belongs to*



## Tools for managing permissions

### chmod

- Changes the file mode "permission" bits of a file
- "Numeric" syntax: **chmod <numeric permission> <pathname>**

```
/home/cis90/simben $ ls -l letter
-rw-r--r--. 1 simben90 cis90 1044 Oct 14 20:39 letter
```

```
/home/cis90/simben $ chmod 750 letter
/home/cis90/simben $ ls -l letter
-rwxr-x---. 1 simben90 cis90 1044 Oct 14 20:39 letter
```

```
/home/cis90/simben $ chmod 644 letter
/home/cis90/simben $ ls -l letter
-rw-r--r--. 1 simben90 cis90 1044 Oct 14 20:39 letter
```



## Tools for managing permissions

### chmod

- Changes the file mode "permission" bits of a file.
- "Mnemonic" syntax: **chmod <u|g|o><+|-|=><r|w|x> <pathname(s)>**  
**u**=user(owner), **g**=group, **o**=other  
**r**=read, **w**=write, **x**=execute

```
/home/cis90/simben $ ls -l letter
-rw-r--r--. 1 simben90 cis90 1044 Oct 14 20:39 letter
```

```
/home/cis90/simben $ chmod u+x,g+w,o-r letter
/home/cis90/simben $ ls -l letter
-rwxrw----. 1 simben90 cis90 1044 Oct 14 20:39 letter
```

```
/home/cis90/simben $ chmod u=rw,g=r,o=r letter
/home/cis90/simben $ ls -l letter
-rw-r--r--. 1 simben90 cis90 1044 Oct 14 20:39 letter
```



## Tools for managing permissions

**umask** – Allows specific permissions to be removed on future newly created files and directories