



Rich's lesson module checklist

Last updated 3/21/2018

- ☐ 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: <http://cabrillo.edu/~jgriffin/>



Rich Simms

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

And thanks to:

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



Student checklist - Before class starts

The screenshot shows the website simms-teach.com/cis90calendar.php. The page title is "Rich's Cabrillo College CIS Classes CIS 90 Calendar". On the left sidebar, the "CIS 90" link is highlighted. The main content area shows the "CIS 90 (Fall 2014) Calendar" with tabs for "Course Details", "Grades", and "Calendar". The "Calendar" tab is selected. Below the tabs, there is a table with columns: "Lesson", "Date", "Topics", and "Link". The first row is for "Lesson 1" on "9/2", with topics including "Class and Linux Overview", "Understand how the course will work", "High level overview of computers, operating systems and virtual machines", "Overview of UNIX/Linux market and architecture", "Using SSH for remote network logs", and "Using terminals and the command line". Below the table, there are links for "Presentation slides (download)", "Supplemental" (including "PowerPoint: Logging into Opus (download)"), "Assignment" (including "Student Survey" and "Lab 1"), "CIS 90 Certificate", "Enter virtual classroom", "Quiz 1", and "Commands".

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

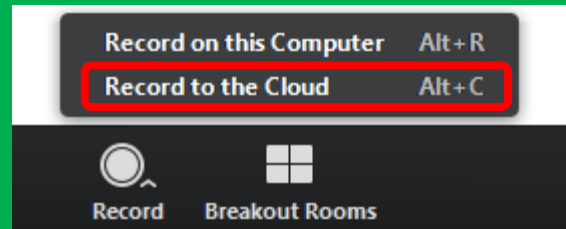
☐ Downloaded PDF of Lesson Slides. I like Foxit Reader so I can take notes using annotations.

The screenshot shows a Zoom meeting interface with several windows open. The main window displays a PDF document titled "Get into the car" with a background image of a white car. Other windows include the Google homepage, the Rich's Cabrillo College CIS 90 website, and a document titled "CIS 90 - Lesson 1" showing a stack of papers and the text "Each student gets their own Arya VM for the term". The Zoom toolbar at the bottom shows options like "Unmute", "Start Video", "Invite", "Participants", "Share Screen", "Chat", "Record", and "Leave Meeting".

☐ CIS 90 website Calendar page

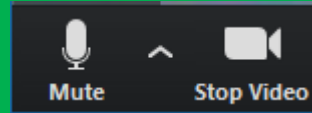
☐ One or more login sessions to Opus-II

Start



Start Recording

Audio Check



Start Recording

Audio & video Check



Instructor: **Rich Simms**
Dial-in: **408-638-0968 (toll)**
Meeting ID: **426 283 384**



Brandon



Shane



Dan



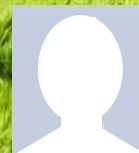
Kage



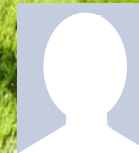
Nathan K.



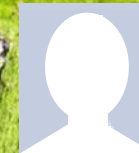
Jo Anne



Darren



Laine



Luis



Christian



Jetta



Cesar



Paul



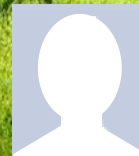
Hilary



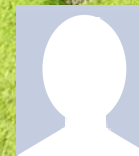
Fritz



Jake



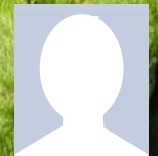
Richard



Nate P.



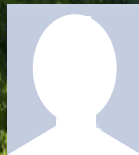
Ciarán



November



Henry



Elena



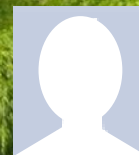
David



Claudius



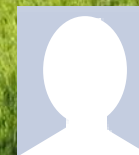
Edgar



Adam

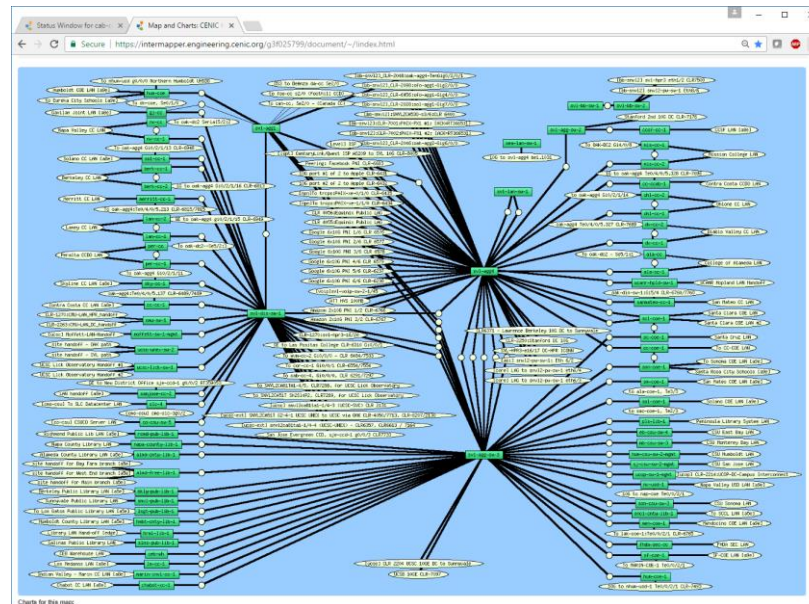


Nathanael T.



Clara

Network Check



<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

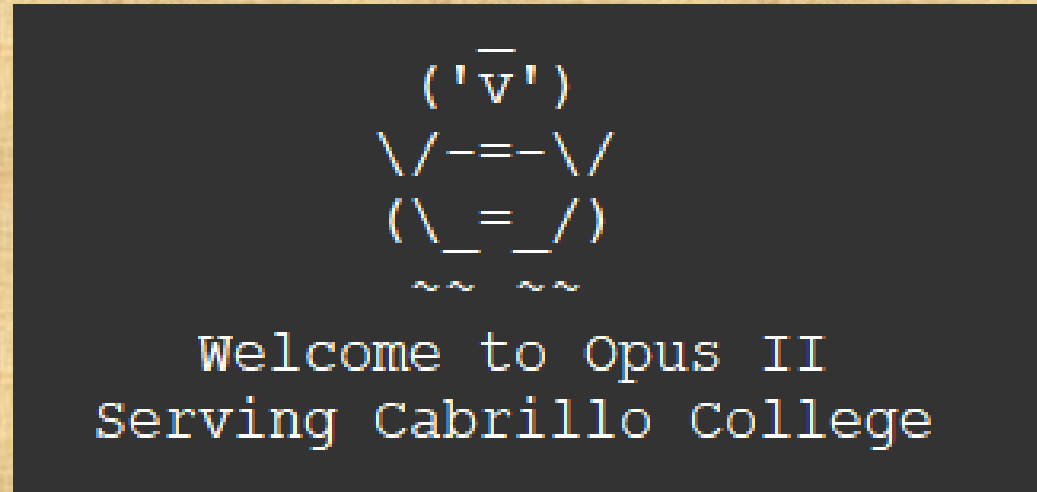
email answers to: risimms@cabrillo.edu

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

Input/Output Processing

Objectives	Agenda
<ul style="list-style-type: none">• 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:<ul style="list-style-type: none">❖ find❖ grep❖ wc❖ sort❖ spell	<ul style="list-style-type: none">• 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

3	2/19	Unit 3 Electronic Mail <ul style="list-style-type: none">• 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 <ul style="list-style-type: none">• Presentation slides (download) Supplemental <ul style="list-style-type: none">• Howto #319: Accessing vlab (download) Assignment <ul style="list-style-type: none">• Read/skim Lesson 8 slides	Lab 2
---	------	--	-----------------------

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

If you haven't already,
download the lesson slides

Class Activity

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

<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 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。

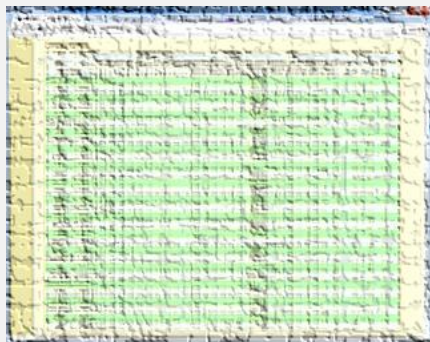
He who asks a question is a fool for five minutes; he who does not ask a question remains a fool forever.

Where to find your grades

Send me your survey to get your LOR code name.

The CIS 90 website Grades page

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



Or check on Opus-II

checkgrades *codename*
(where codename is your LOR codename)



Written by Jesse Warren a past CIS 90 Alumnus

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

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`

In lesson slides (search for extra credit)



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)

Getting Help When Stuck on a Lab Assignment

- Google the topic/error message.
- Search the Lesson Slides (they are PDFs) for a relevant example on how to do something.
- Post a question on the forum. Explain what you are trying to do and what you have tried so far.
- Talk to a STEM center tutor/assistant.
- Come see me during my office or lab hours. **I will be in the CTC (room 1403) every Wednesday afternoon from 3-5:30.**
- 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!

Expect to do a LOT of troubleshooting in this course!

Help Available in the CIS Lab

Instructors, lab assistants and equipment are available for CIS students to work on assignments.



Rich's Cabrillo College CIS Classes
Home Page

Home

Resources

Forums

CIS Lab

Canvas

CIS Lab

webhawks.org/~cislab/

CIS Lab & Datacenter
Aptos Campus

Home Resources NETLAB VLab Location

Announcements

The CIS Lab is in the **STEM Center** in building 800.
A great place to work on lab assignments and get help from student lab assistants and instructors on the schedule below.

STEM CIS/CS hours

Today Jan 28 - Feb 3, 2018

Week

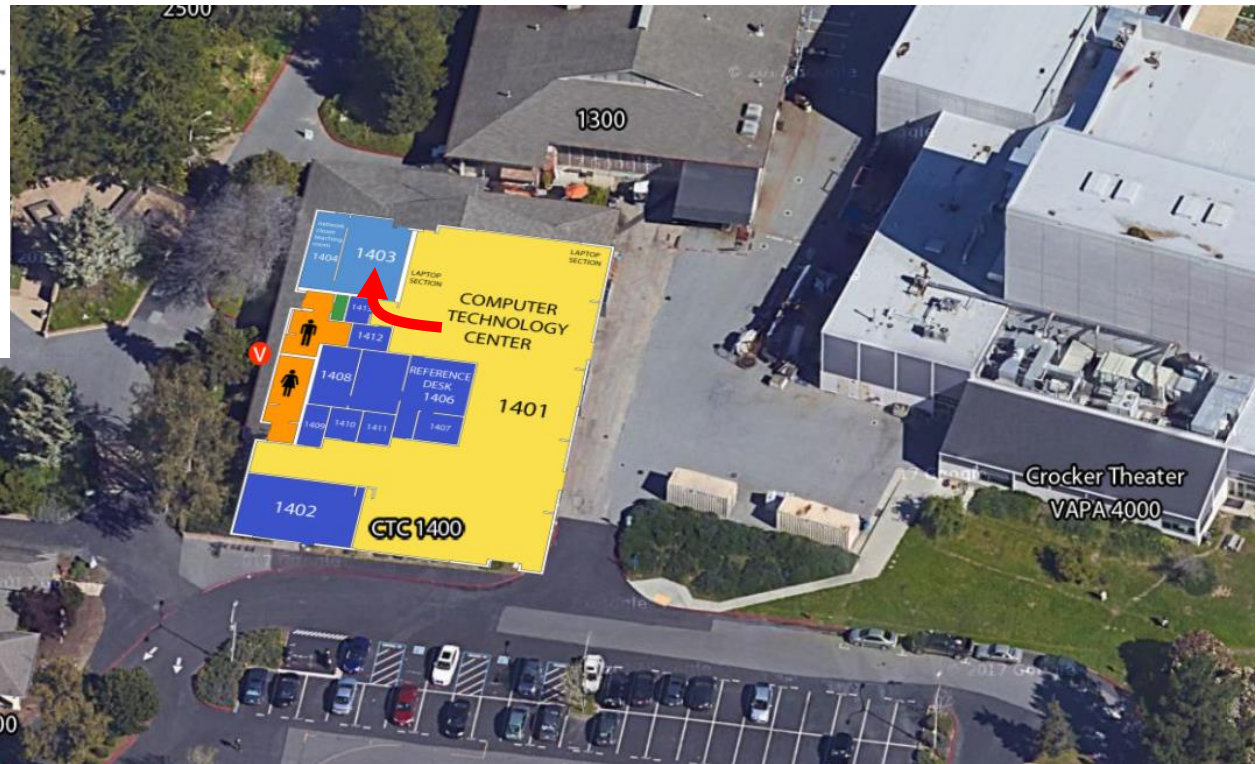
Time	Sun 1/28	Mon 1/29	Tue 1/30	Wed 1/31	Thu 2/1	Fri 2/2	Sat 2/3
10am							
11am							
12pm							
1pm							
2pm		Jeffrey Bergamini CS Instructor 2:10p - 3p Carter Frost CIS/CS	Jeffrey Bergamini CS Instructor 2:10p - 3p Carter Frost CIS/CS	Jeffrey Bergamini CS Instructor 2:10p - 3p Carter Frost CIS/CS	Jeffrey Bergamini CS Instructor 2:10p - 3p Carter Frost CIS/CS		
3pm							
4pm							
5pm							
6pm							
7pm							

Events shown in time zone: Pacific Time

W3C XHTML 1.0 W3C CSS

To see schedule, click the CIS Lab link on the website and use the "Week" calendar view

CTC - Building 1400 On lower campus



I will be in the CTC (room 1403) every Wednesday
afternoon from 3-5:30



The slippery slope



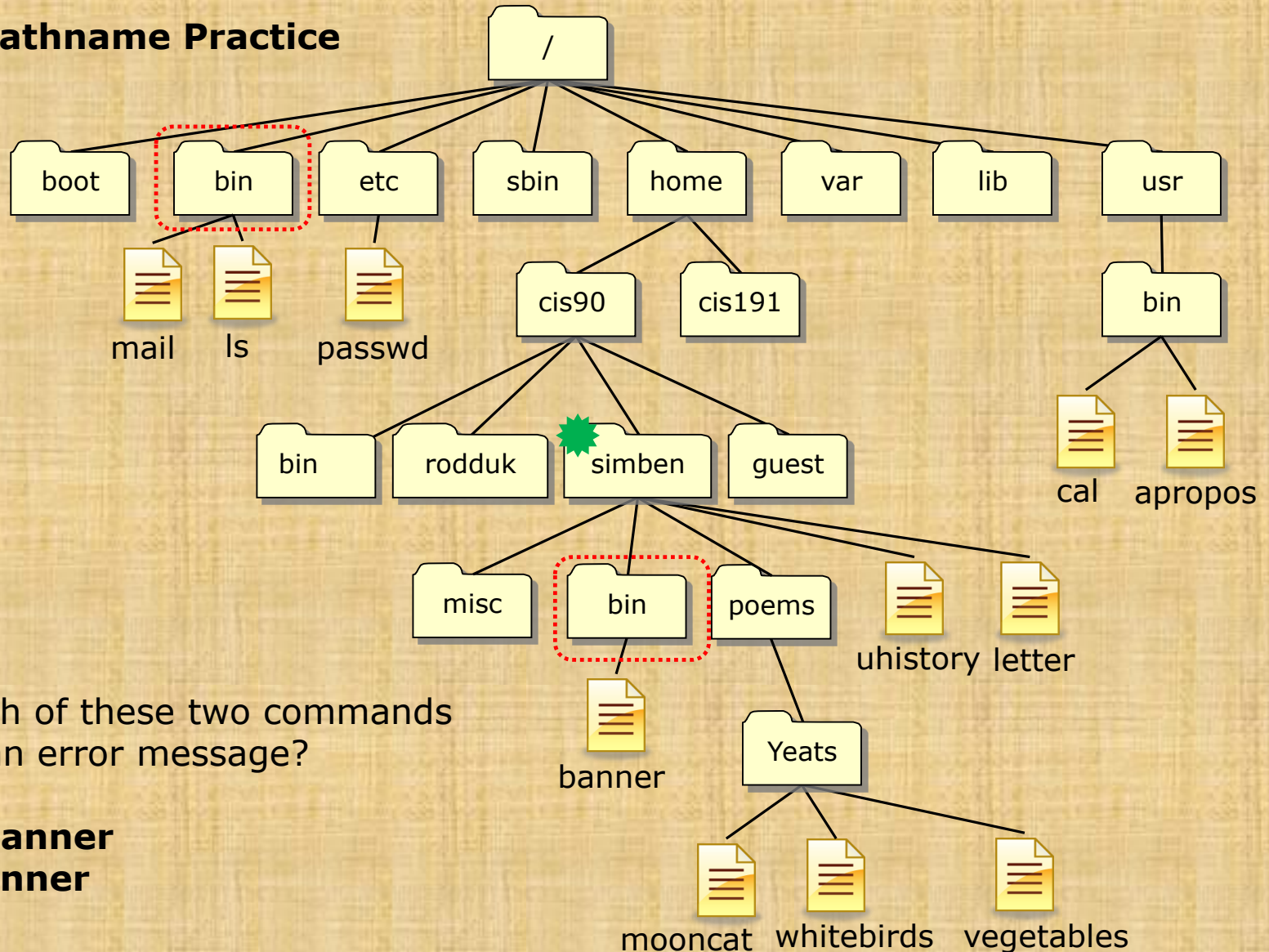
- 1) If you didn't submit the last lab ...
- 2) If you were in class and didn't submit the last quiz ...
- 3) If you didn't send me the student survey assigned in Lesson 1 ...
- 4) If you haven't made a forum post in the last quarter of the course ...


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

Email: risimms@cabrillo.edu

Warmup

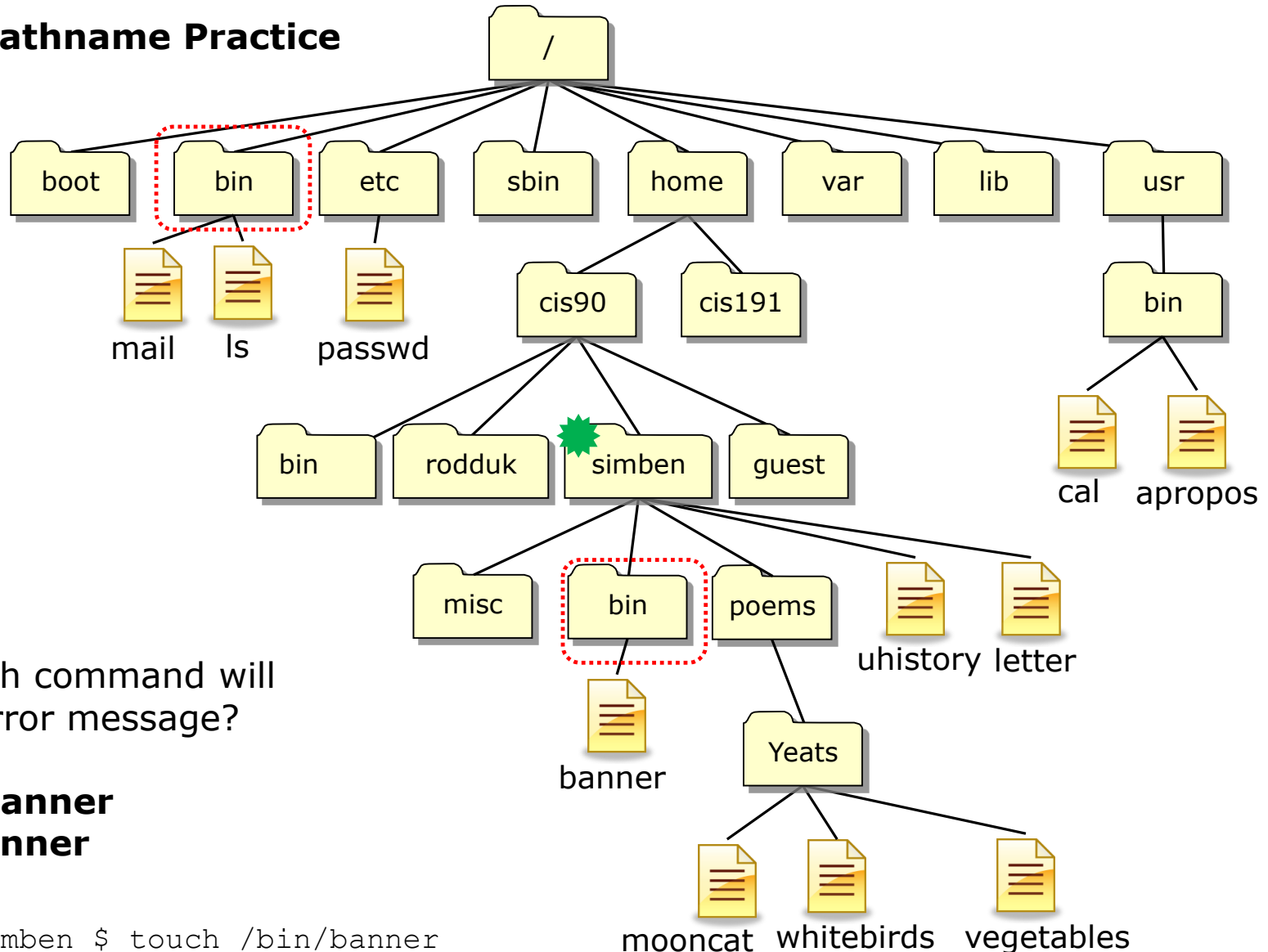
File Tree Pathname Practice




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

touch /bin/banner
touch bin/banner

File Tree Pathname Practice



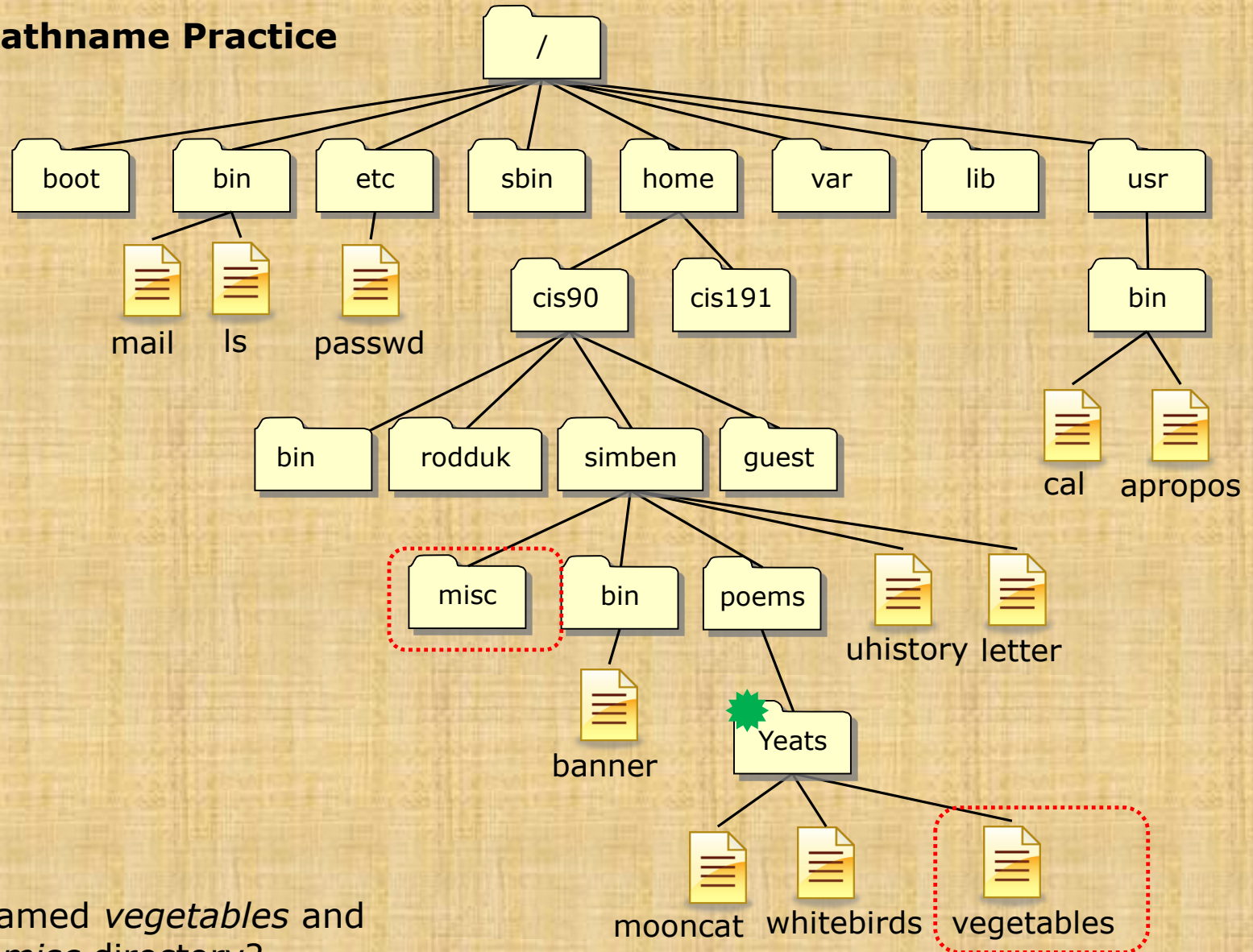
From  which command will generate an error message?

touch /bin/banner
touch bin/banner

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

banner is in your local bin directory

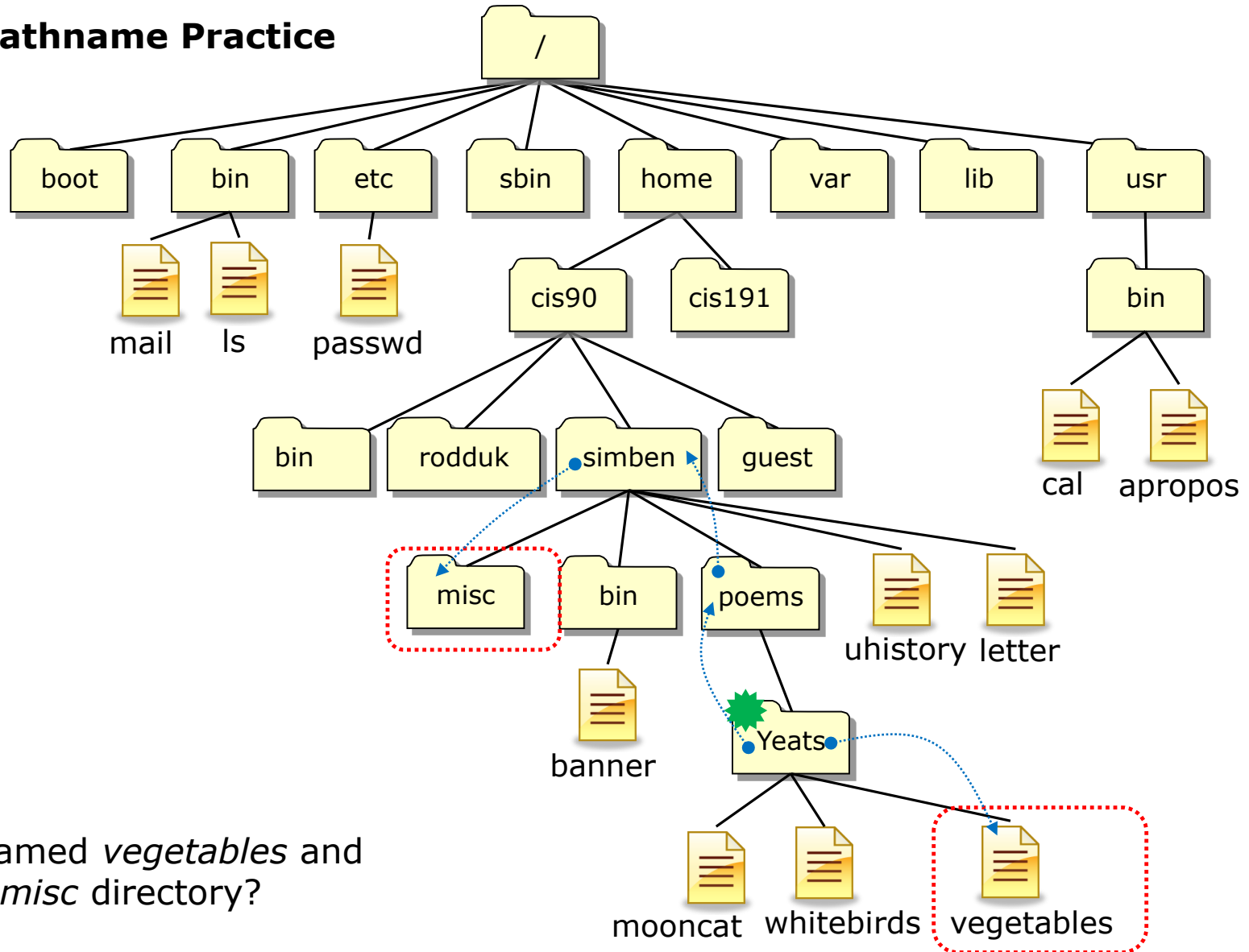
File Tree Pathname Practice



From  how
does Benji:

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

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 <path-to-file> <path-to-directory>

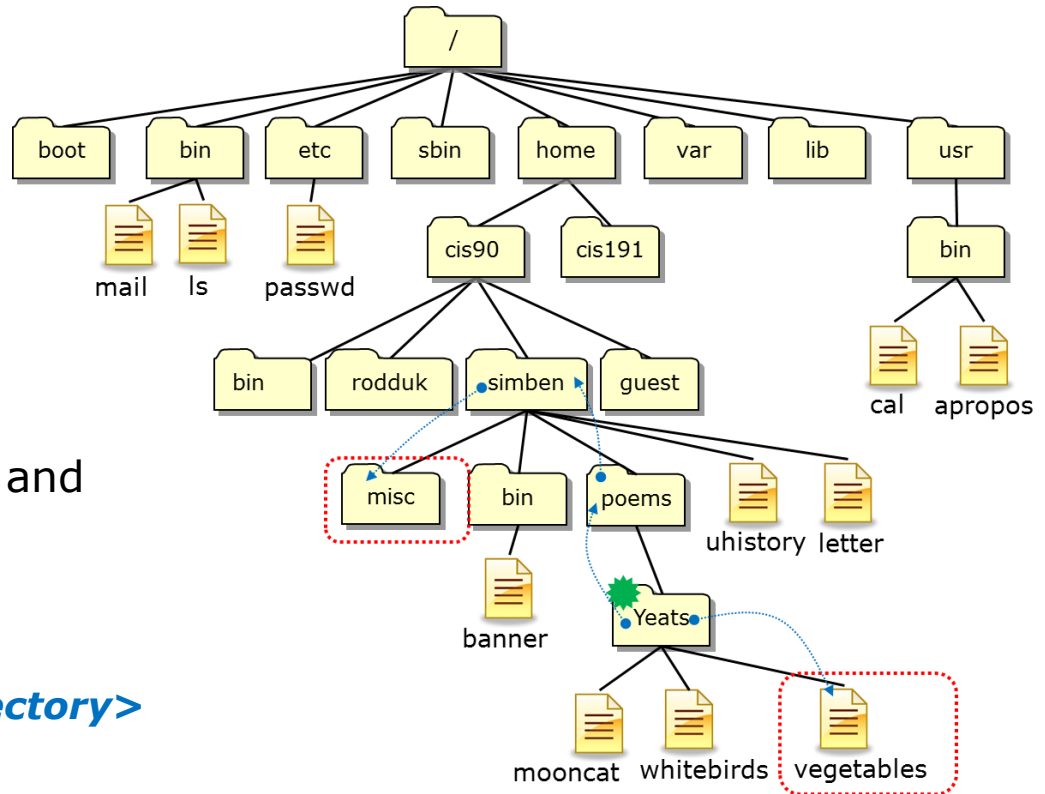
mv vegetables ../../misc/

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

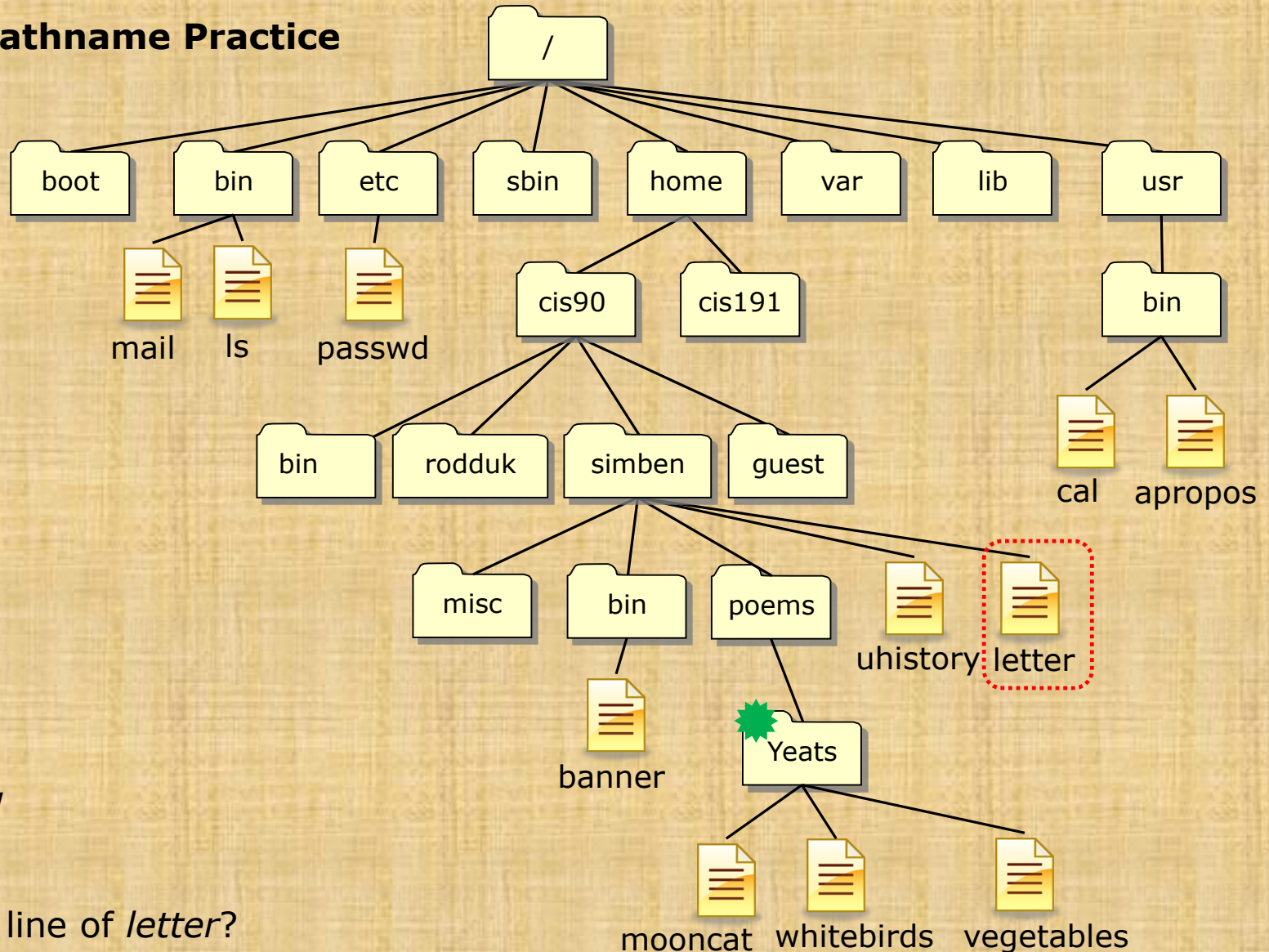
or mv /home/cis90/simben/poems/Yeats/vegetables ../../misc/

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

or mv vegetables ~/misc/



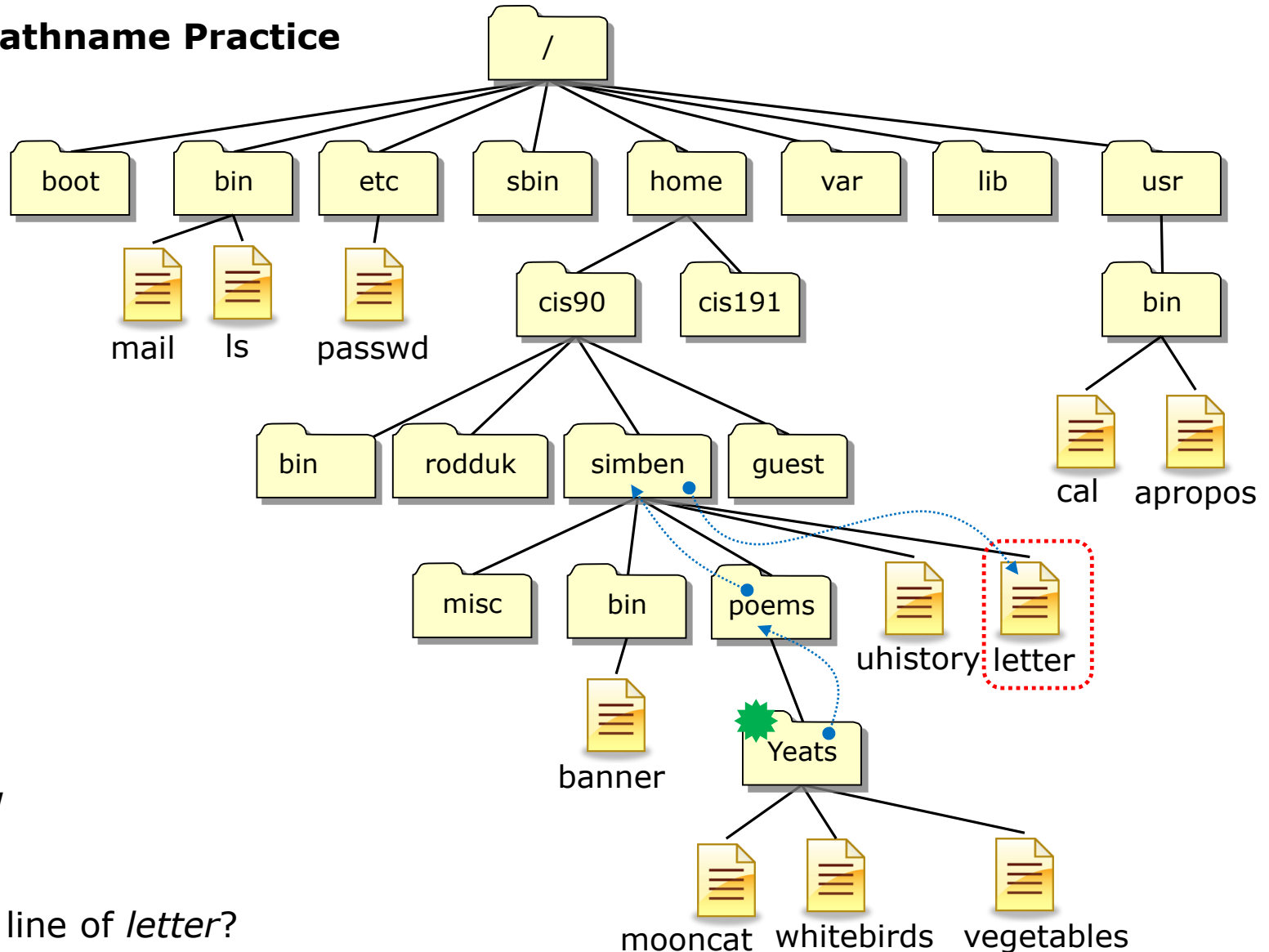
File Tree Pathname Practice



From  how
does Benji:

Print the last line of *letter*?

File Tree Pathname Practice



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

*Other answers
are also
acceptable*

From  how
does Benji:

Print the last line of *letter*?

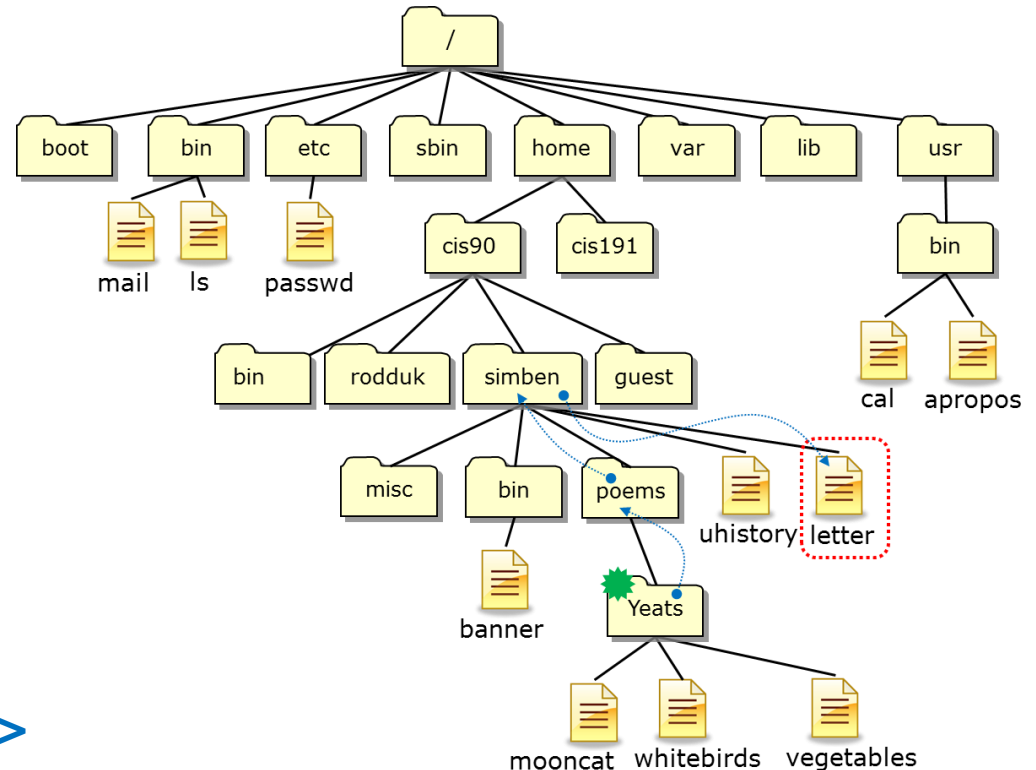
tail -n<number> <path-to-file>

tail -n1 ../../letter

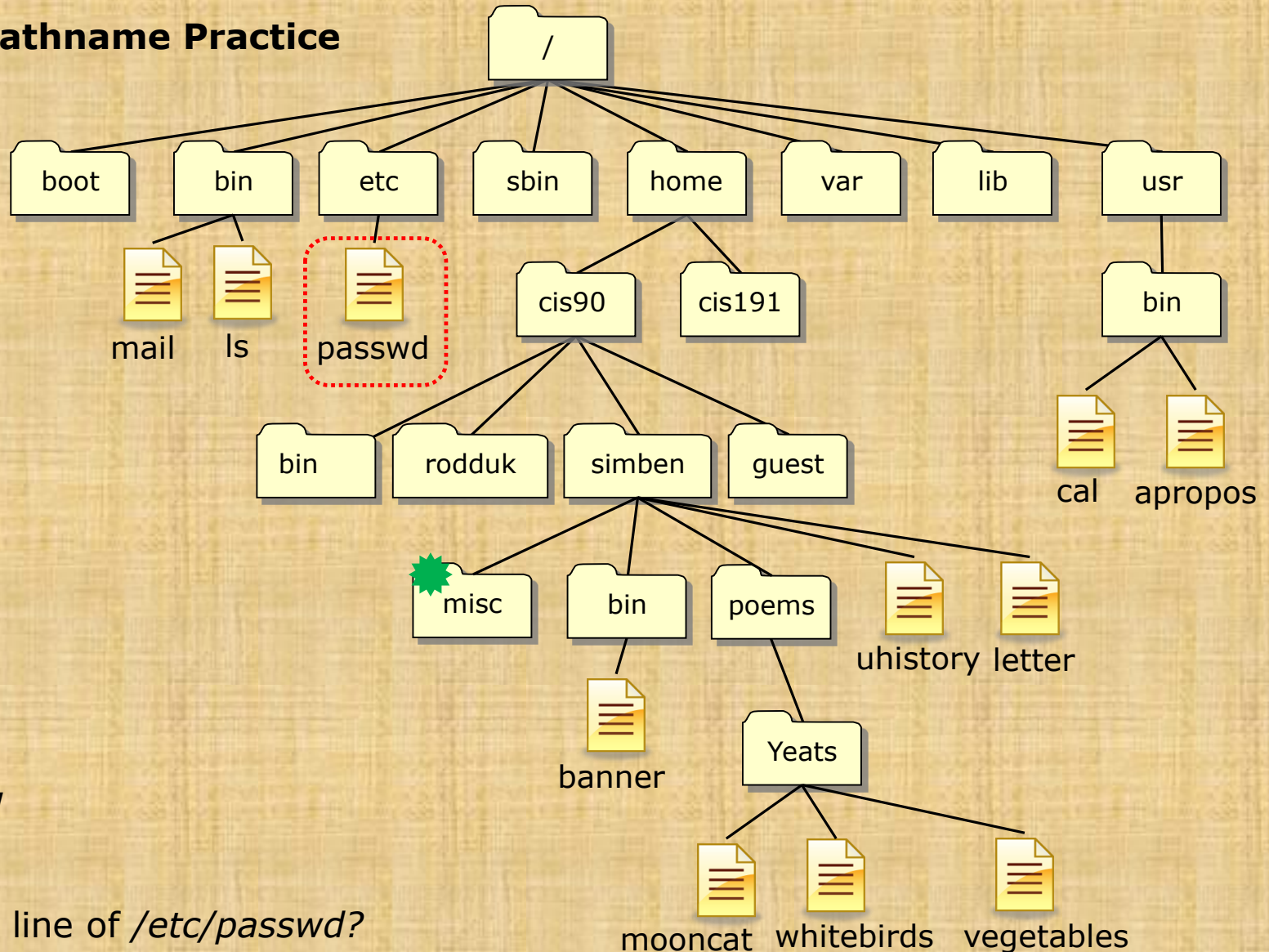
or tail -n1 /home/cis90/simben/letter

or tail -n1 ~/letter

All these answers are correct



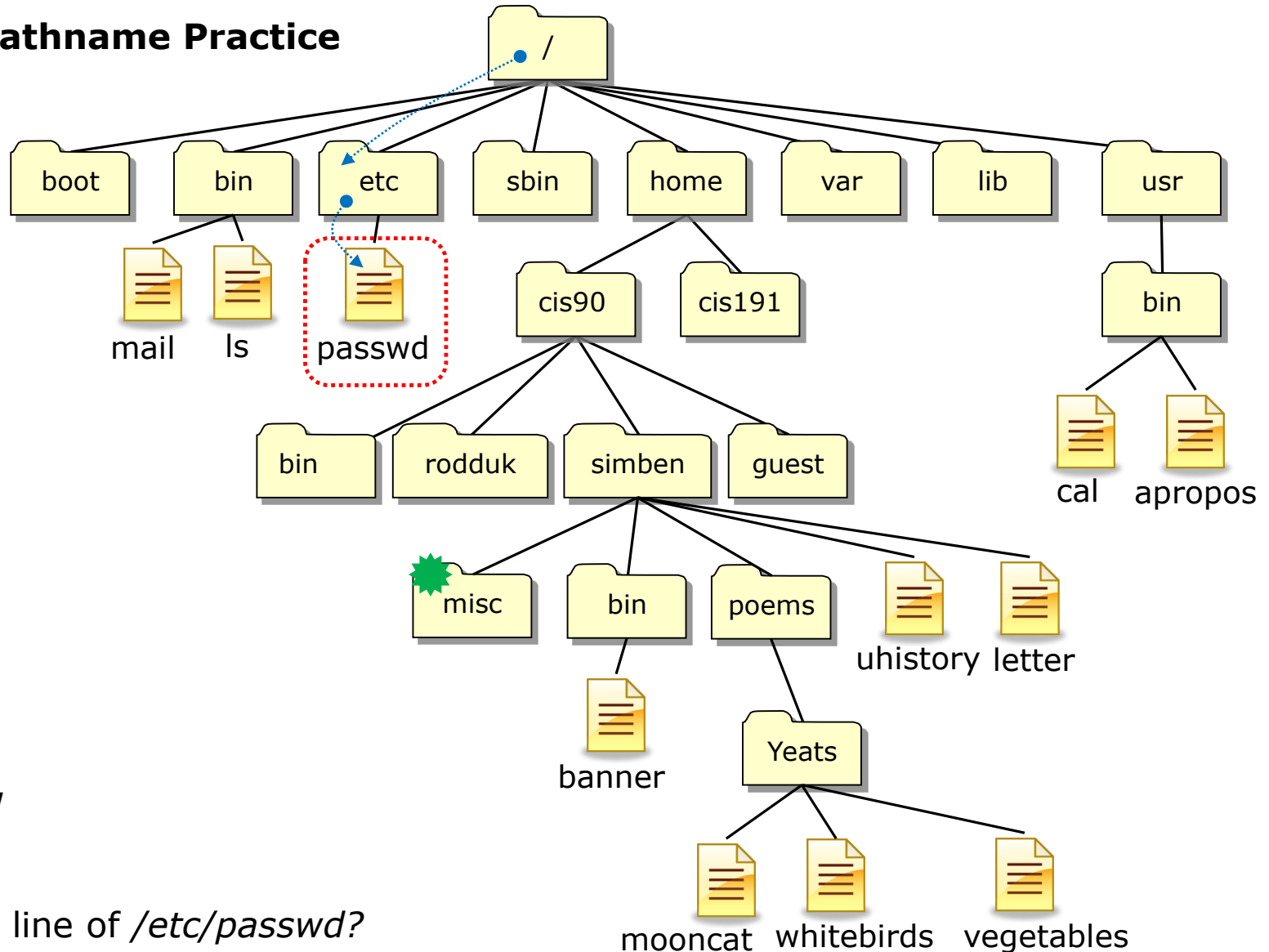
File Tree Pathname Practice



From  how
does Benji:

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

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:

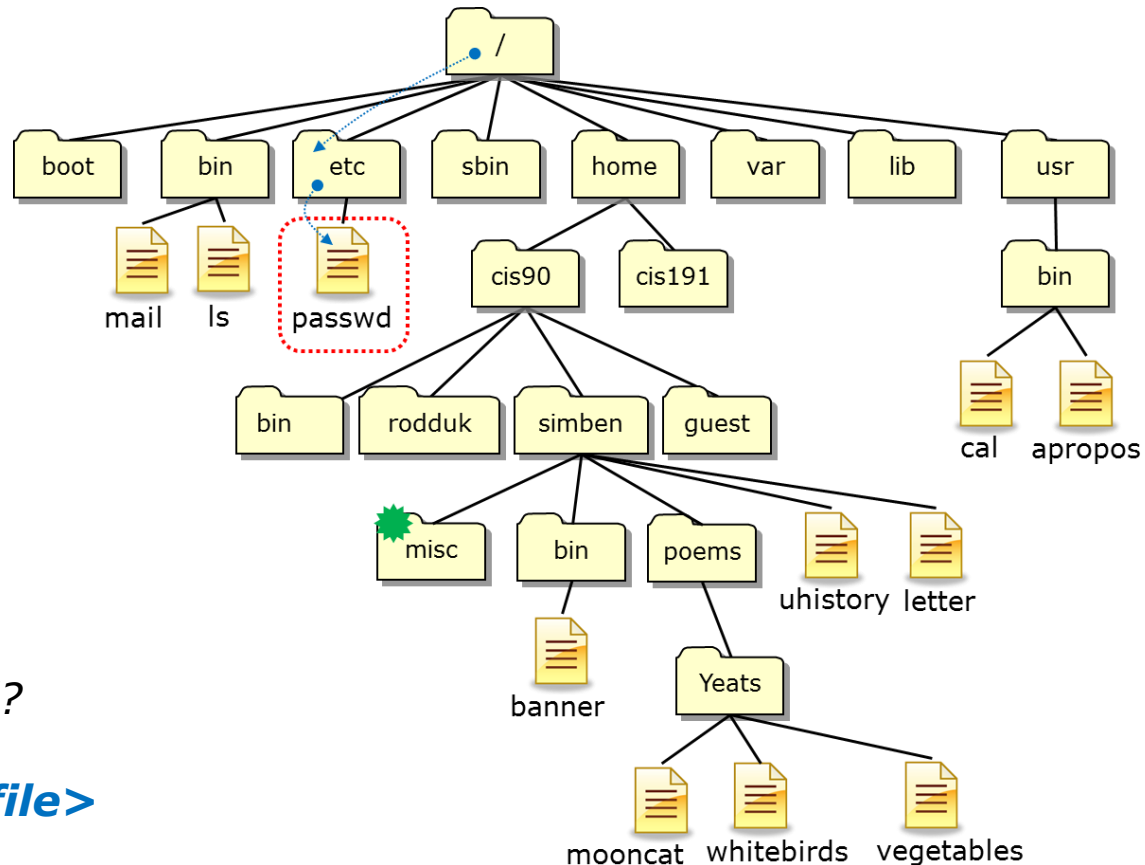
Print the first line of `/etc/passwd`?

`head -n<number> <path-to-file>`

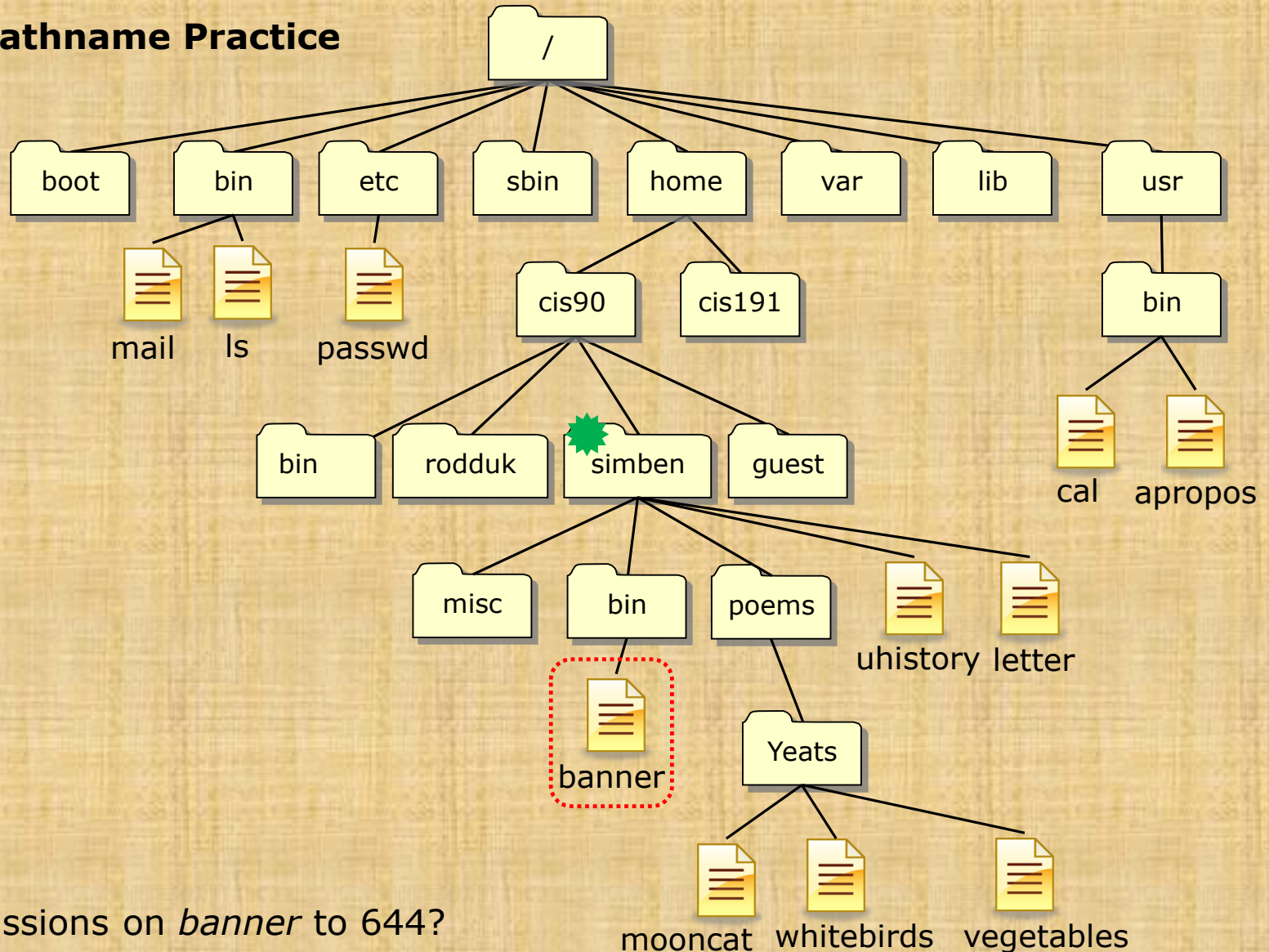
or **`head -n1 /etc/passwd`**

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

Both these answers are correct



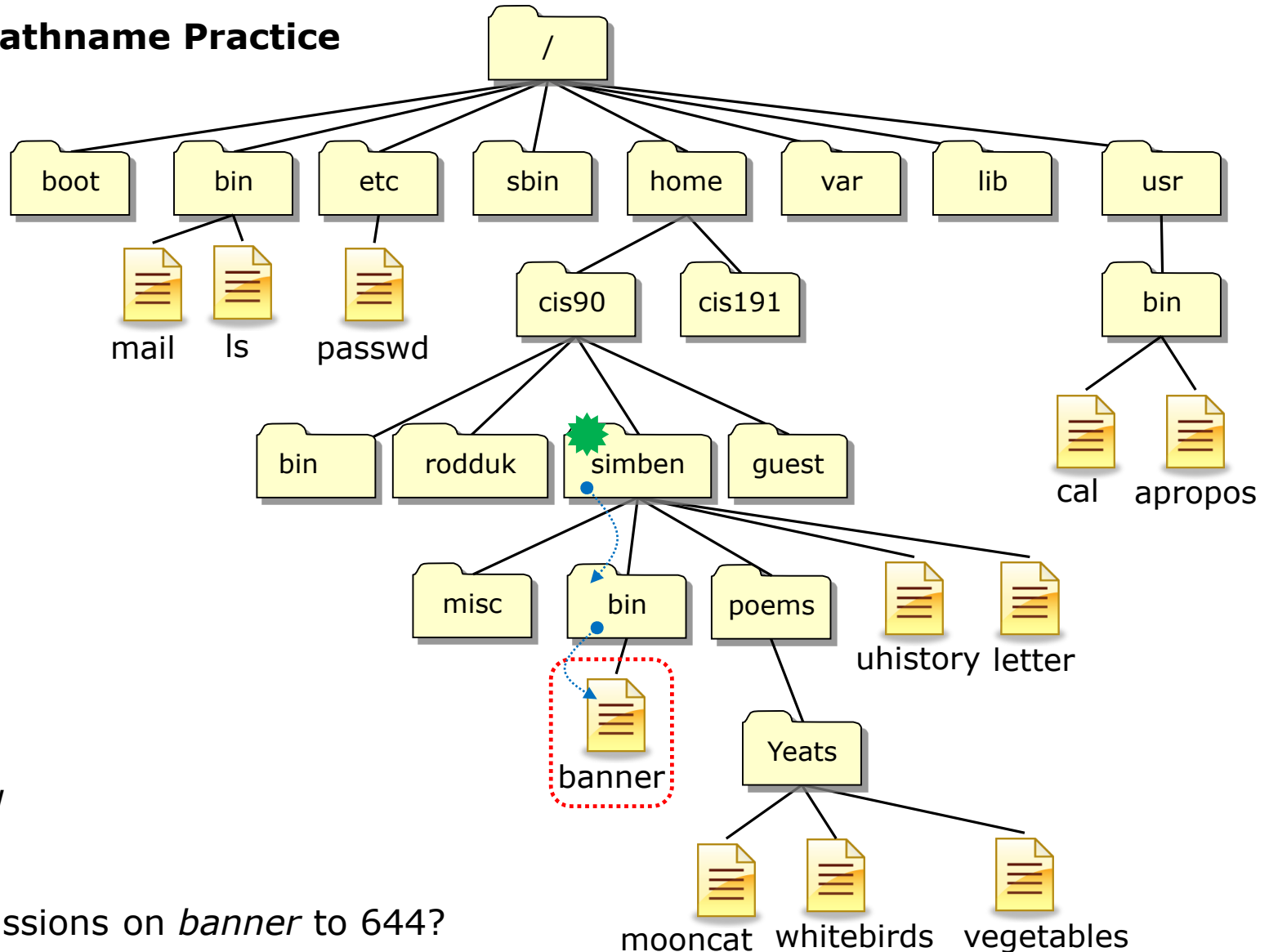
File Tree Pathname Practice



From  how
does Benji:

Change permissions on *banner* to 644?

File Tree Pathname Practice

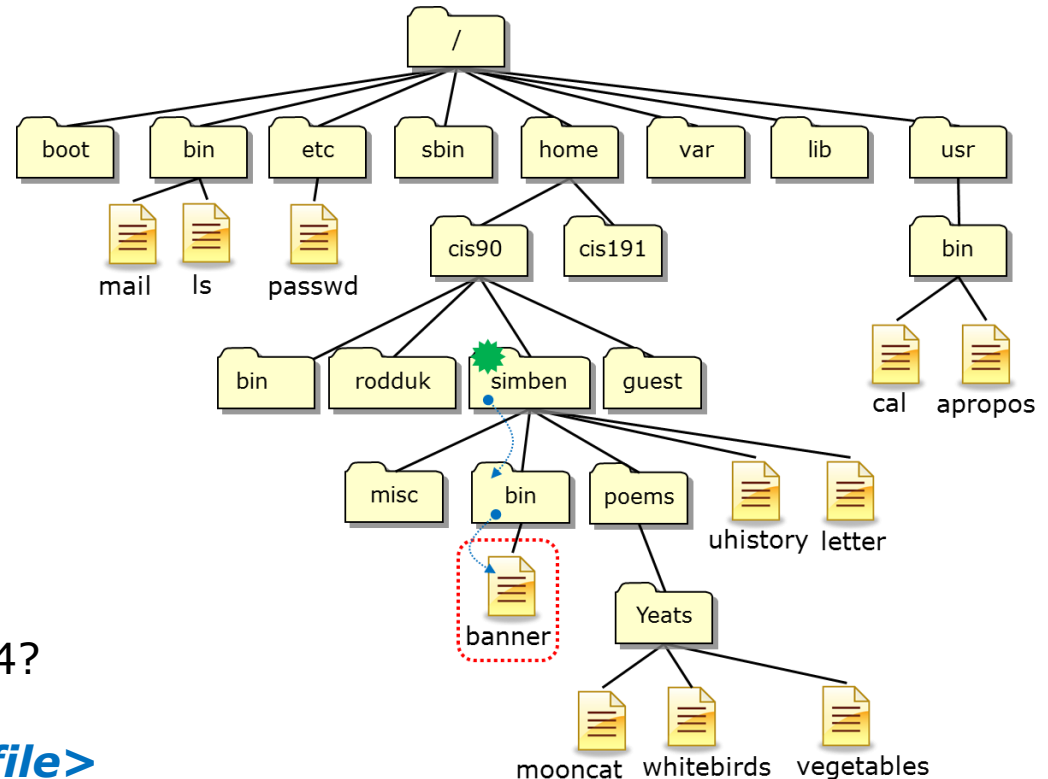


From  how
does Benji:

Change permissions on *banner* to 644?

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

*Other answers
are also
acceptable*



From  how
does Benji:

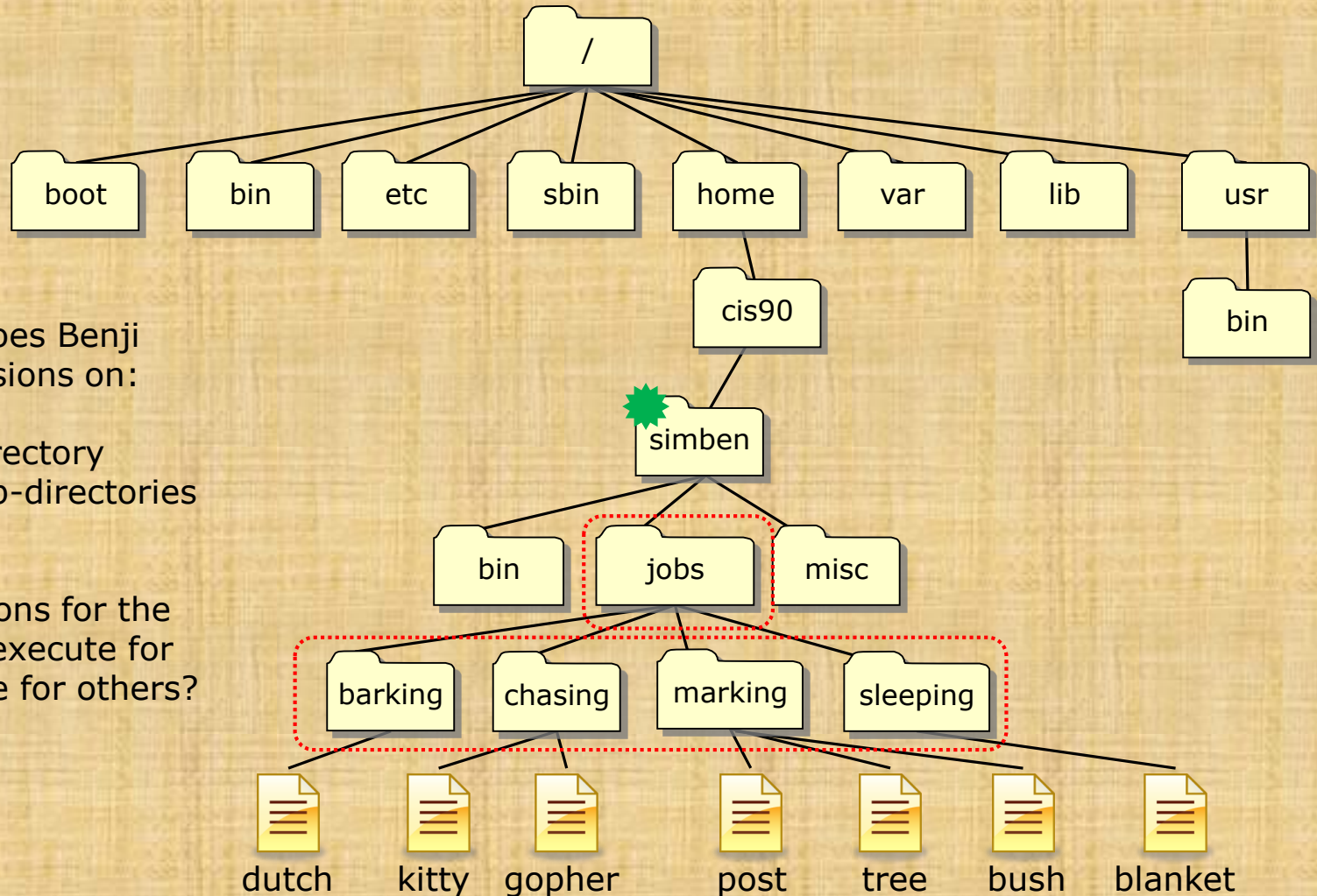
Change permissions on *banner* to 644?


chmod *<permissions>* *<path-to-file>*

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?

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

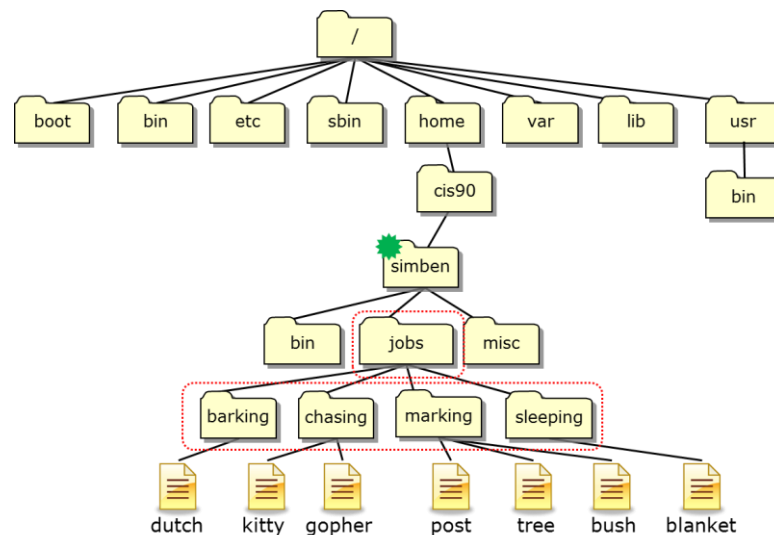
```
chmod 750 jobs
chmod 750 jobs/barking/ jobs/chasing/ jobs/marketing/ jobs/sleeping/
```

This is better though

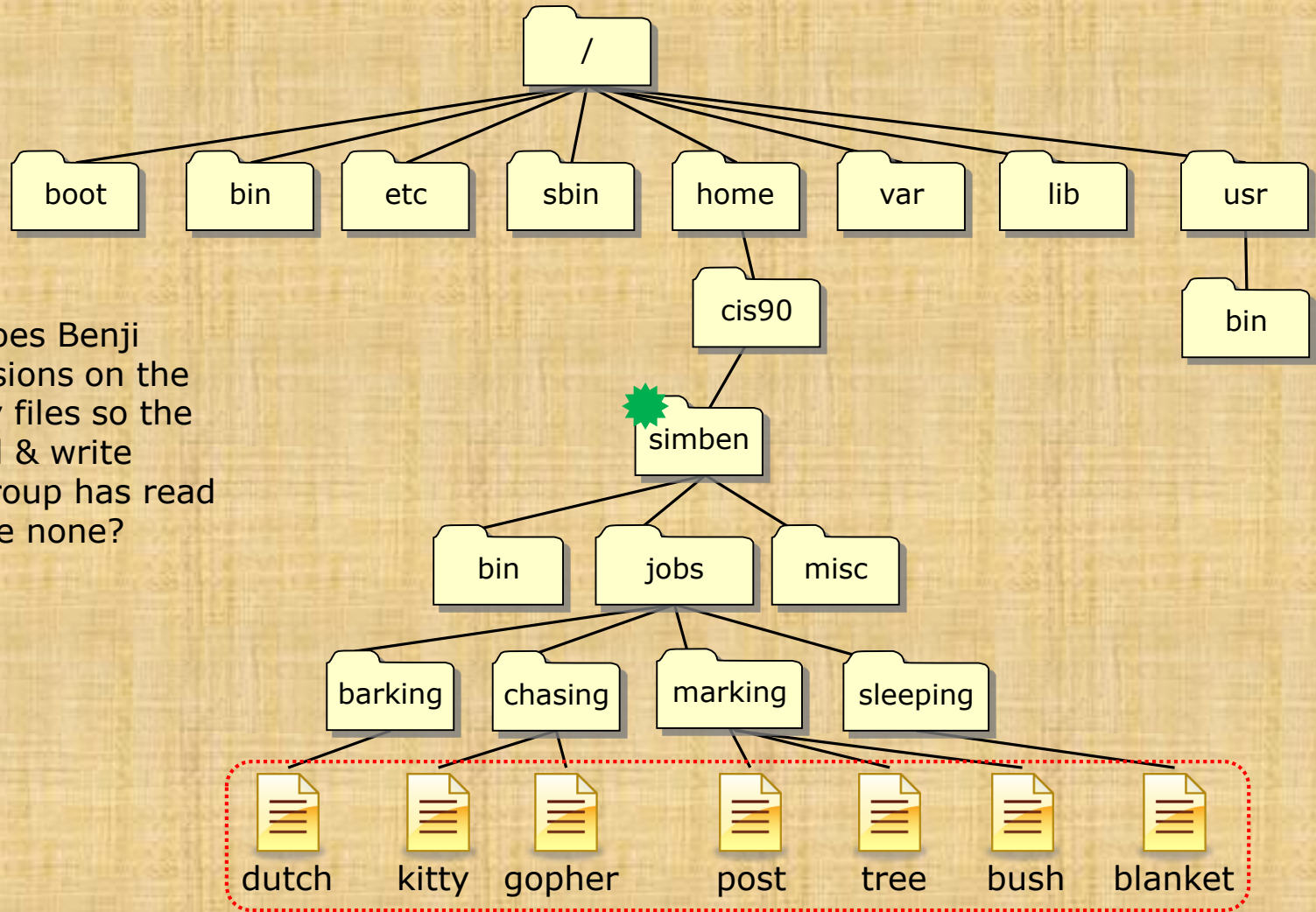
```
chmod 750 jobs
chmod 750 jobs/*
```


I like this the best!

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



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



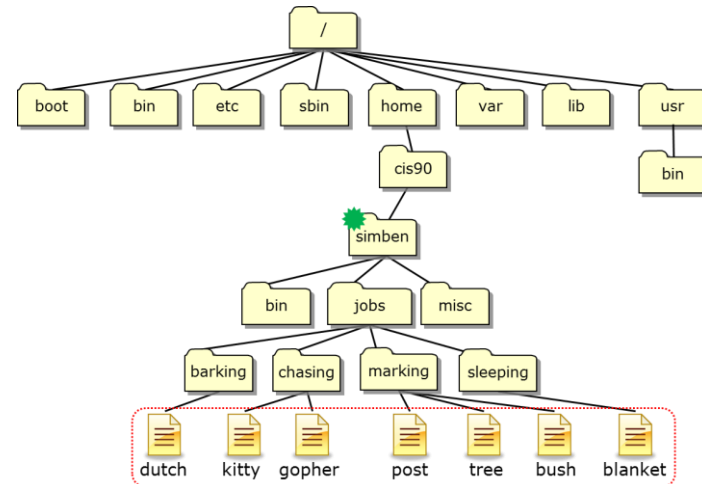
From  how does Benji change permissions on the circled ordinary files so the owner has read & write permissions, group has read and others have none?

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 tree bush
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

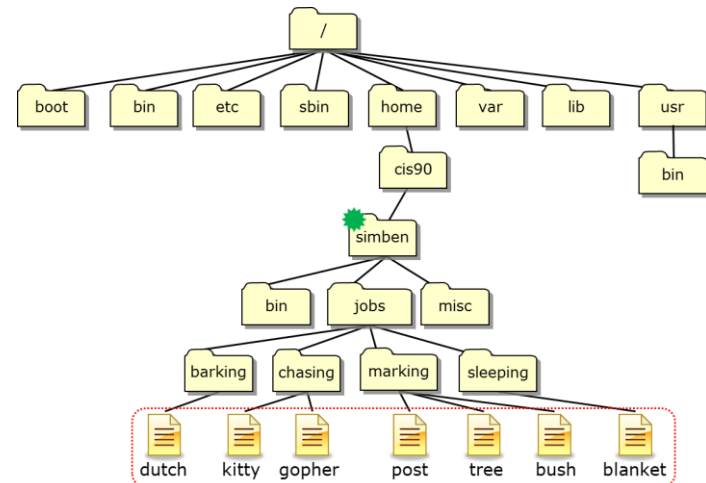
```
cd jobs
cd barking
chmod 640 *
cd ..
cd chasing
chmod 640 *
cd ..
cd marking
chmod 640 *
cd ..
cd sleeping
chmod 640 *
cd
```

This is better

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

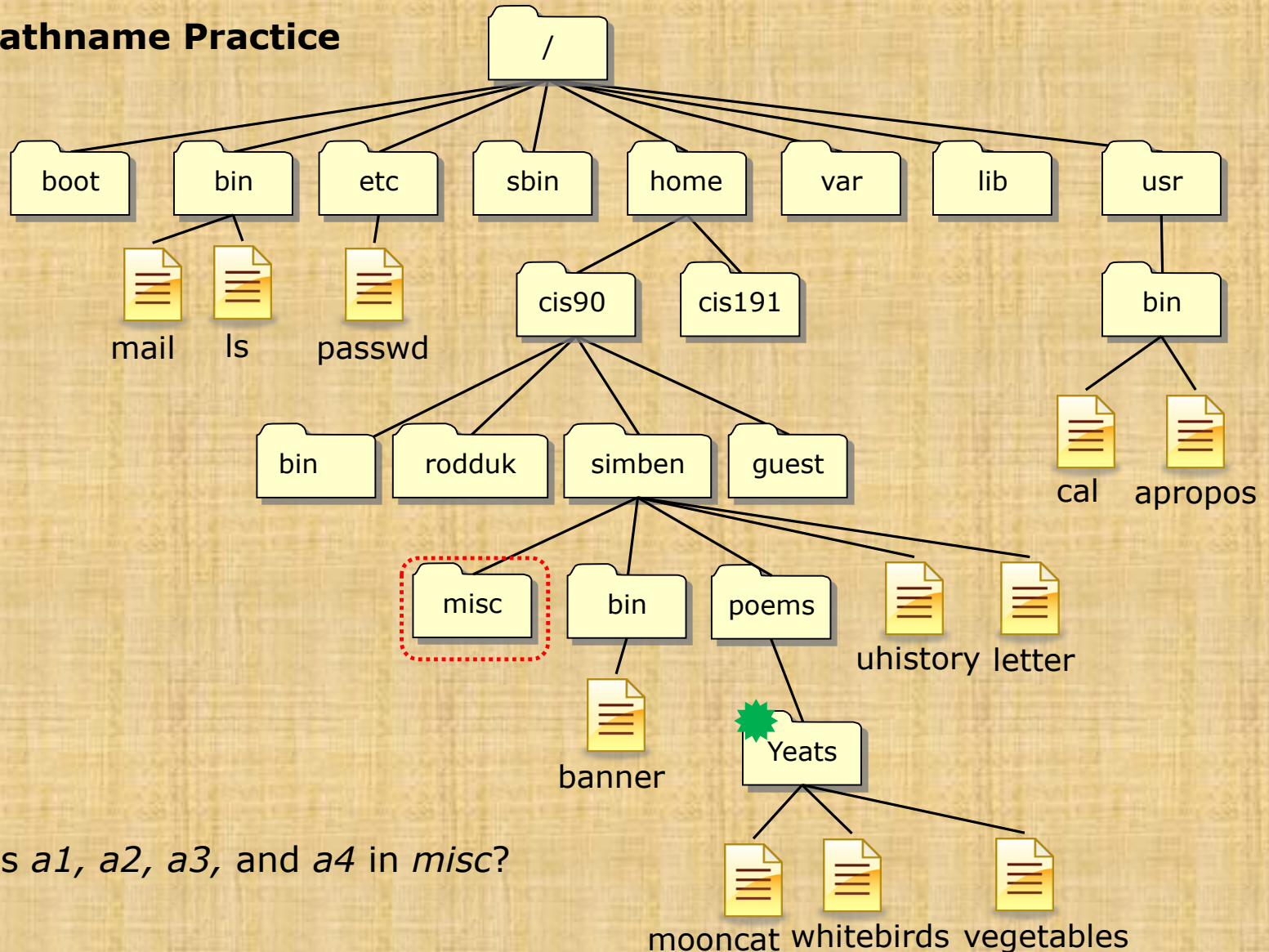
I like this the best!

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



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

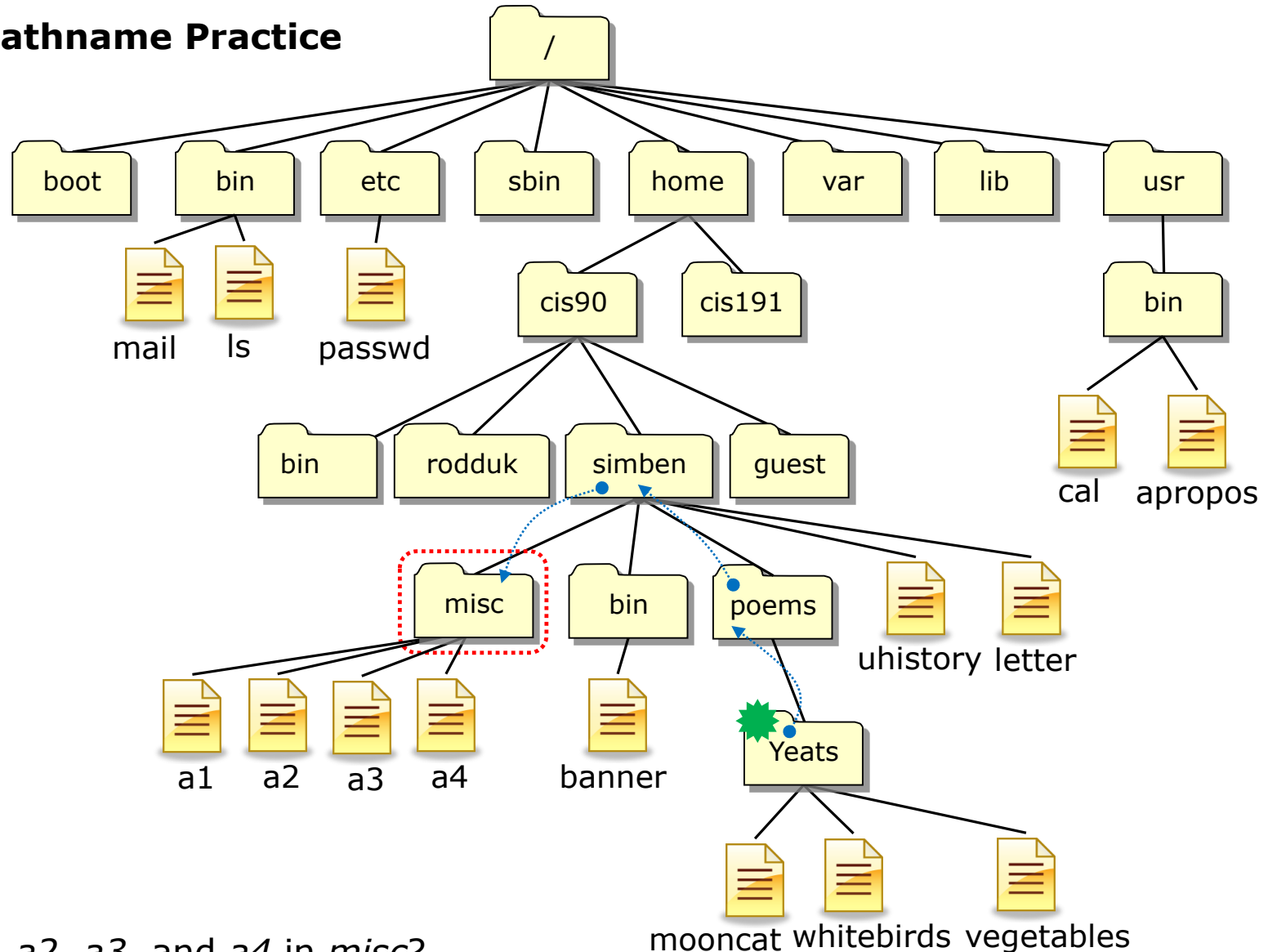
File Tree Pathname Practice



From  how
does Benji:

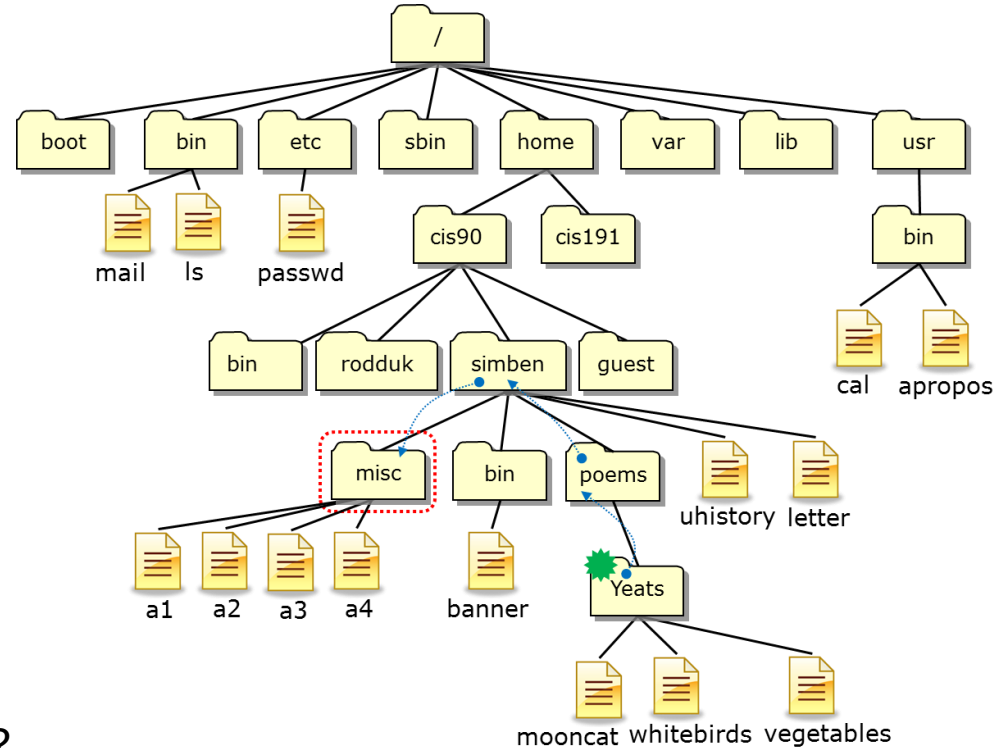
Create new files *a1*, *a2*, *a3*, and *a4* in *misc*?

File Tree Pathname Practice



/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 *<path-to-file>* *<path-to-file>* *<path-to-file>* *<path-to-file>*

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

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

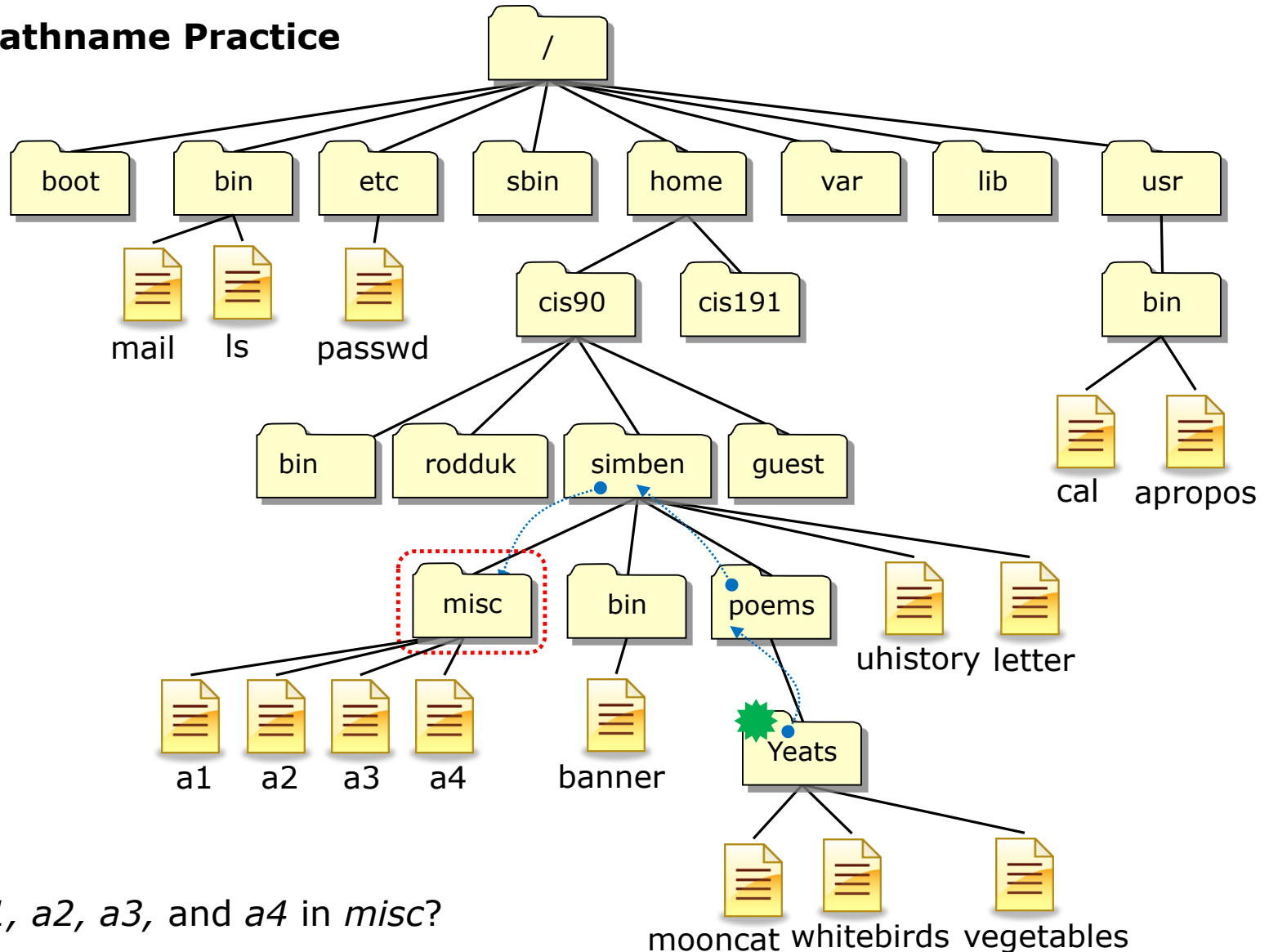
or **touch** /home/cis90/simben/misc/a1 /home/cis90/simben/misc/a2
/home/cis90/simben/misc/a3 /home/cis90/simben/misc/a4 *(all on one line)*

All these answers are correct



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

File Tree Pathname Practice



FYI
only

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}

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**

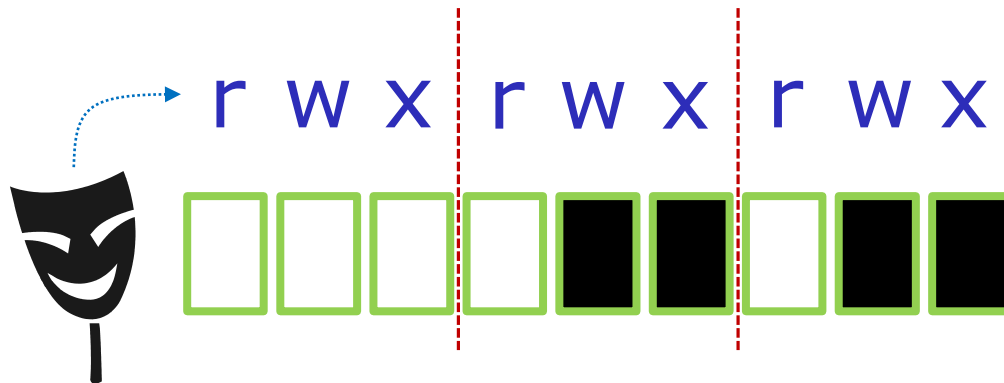
Case 1 – a new directory

With a umask of 033 what permissions would a newly created DIRECTORY have?

Write your answer in the chat window

Case 1 – a new directory

With a umask of 033 what permissions would a newly created DIRECTORY have?



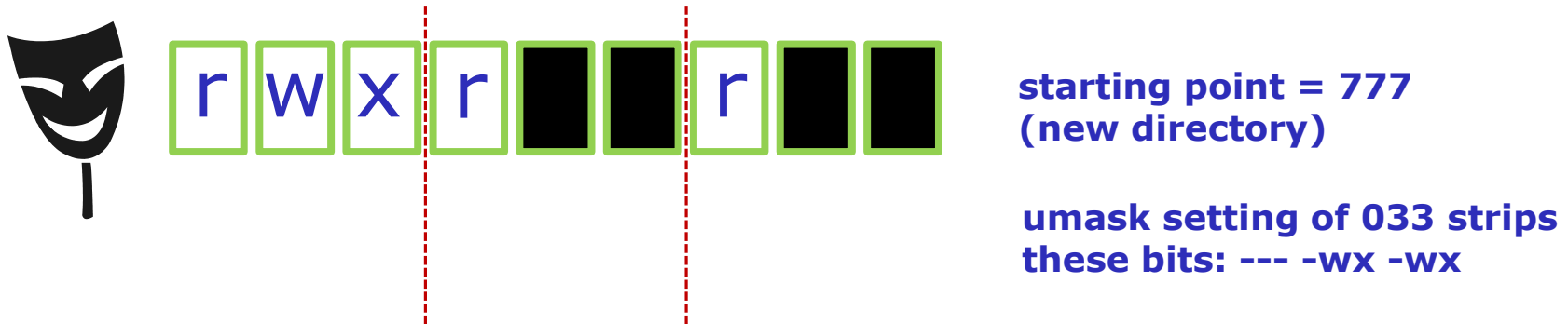
**starting point = 777
(new directory)**

**umask setting of 033 strips
these bits: --- -wx -wx**

Now slide the mask up and over the starting point permissions

Case 1 – a new directory

With a umask of 033 what permissions would a newly created DIRECTORY have?



Answer: 744

Prove it to yourself on Opus-II as shown here

```
/home/cis90ol/simmsben $ umask 033
/home/cis90ol/simmsben $ mkdir brandnewdir
/home/cis90ol/simmsben $ ls -ld brandnewdir/
drwxr--r-- 2 simmsben cis90ol 4096 Apr 21 12:46 brandnewdir/
 7   4   4
```

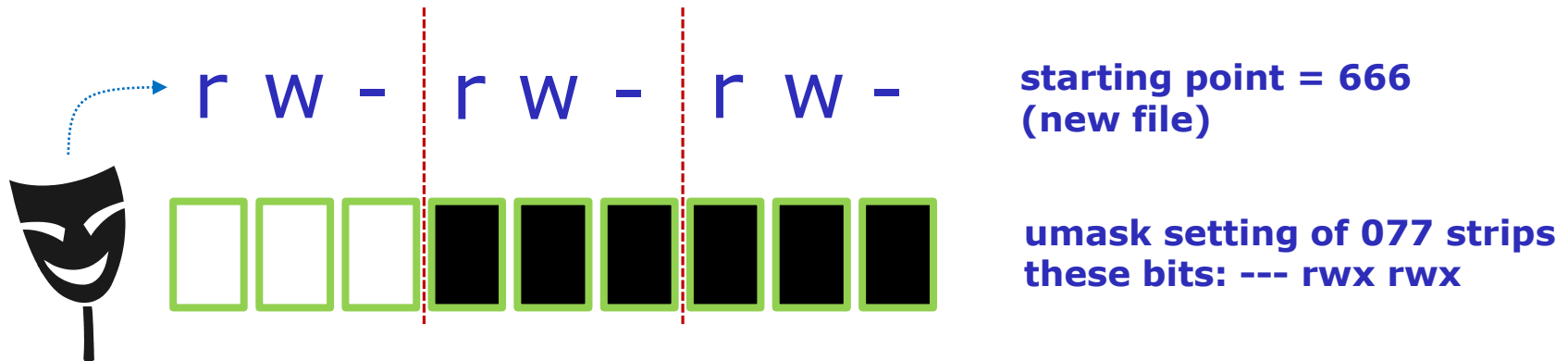
Case 2 – new file

With a umask of 077 what permissions would a newly created FILE have?

Write your answer in the chat window

Case 2 – new file

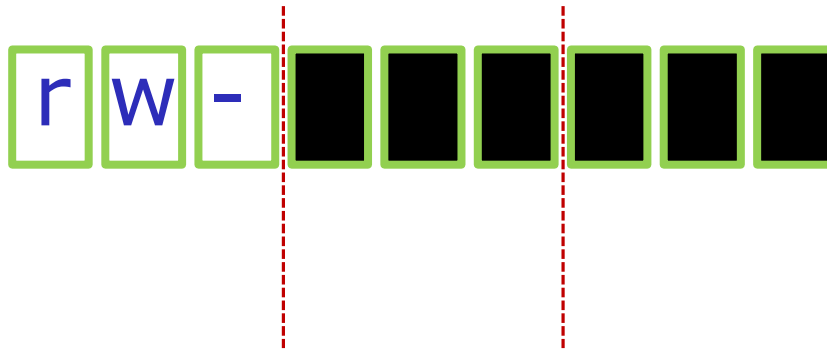
With a umask of 077 what permissions would a newly created FILE have?



Now slide the mask up and over the starting point permissions

Case 2 – new file

With a umask of 077 what permissions would a newly created FILE have?



starting point = 666
(new file)

umask setting of 077 strips
these bits: --- rwx rwx

Answer: 600

Prove it to yourself on Opus-II as shown here

```
/home/cis90ol/simmsben $ umask 077
/home/cis90ol/simmsben $ touch brandnewfile
/home/cis90ol/simmsben $ ls -l brandnewfile
-rw----- 1 simmsben cis90ol 0 Apr 21 12:50 brandnewfile
 6 0 0
```

Case 3 – file copy

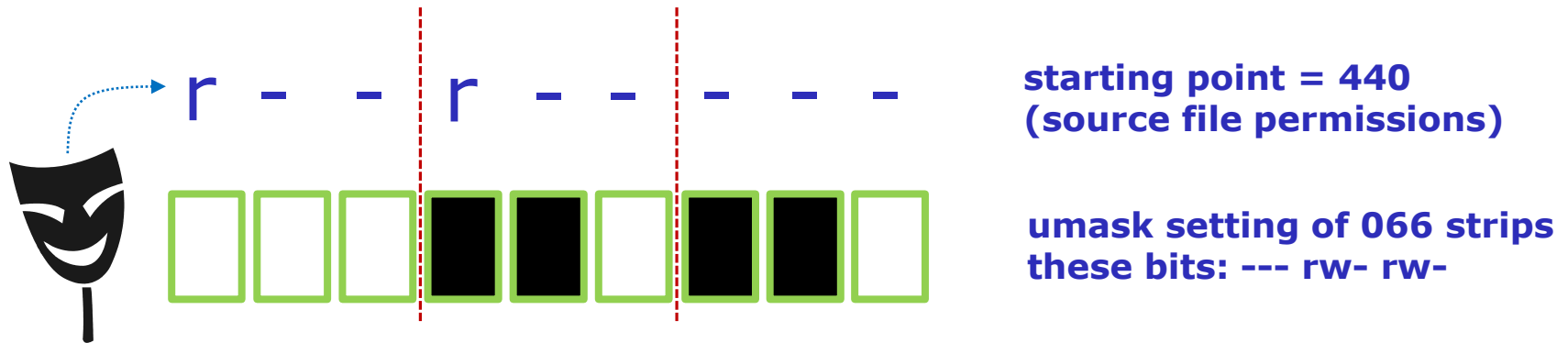
**If umask=066 and the *cinderella* file permissions are 440
What would the permissions be on *cinderella.bak* after:
`cp cinderella cinderella.bak`**

Write your answer in the chat window

Case 3 – file copy

If **umask=066** and the *cinderella* file permissions are **440**
What would the permissions be on *cinderella.bak* after:

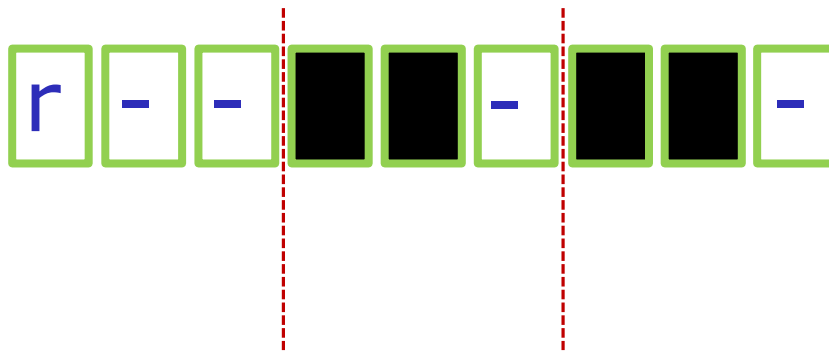
```
cp cinderella cinderella.bak
```



Now slide the mask up and over the starting point permissions

Case 3 – file copy

If **umask=066** and the *cinderella* file permissions are **440**
What would the permissions be on *cinderella.bak* after:
`cp cinderella cinderella.bak`



starting point = 440
(source file permissions)

umask setting of 066 strips
these bits: --- rw- rw-

Answer: 400

Prove it to yourself on Opus-II as shown here

```
/home/cis90/simben $ touch cinderella
/home/cis90/simben $ chmod 440 cinderella
/home/cis90/simben $ umask 066
/home/cis90/simben $ cp cinderella cinderella.bak
/home/cis90/simben $ ls -l cinderella.bak
-r----- . 1 simben90 cis90 0 Oct 22 09:17 cinderella.bak
  4   0   0
```

Housekeeping



Previous material and assignment

1. Lab 6 due 11:59PM.
2. A **check6** script is available.



Don't forget to **submit** your final Lab 6!

3. Use **verify** to view what you submitted for grading.
4. Five more posts due 11:59PM.
5. Early preview of Lab X2 is now available. This is recommended for anyone wanting more practice with pathnames.

Overlap Students

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

Linux Computer Home Loans



<https://docs.google.com/a/cabrillo.edu/spreadsheets/d/1ljwkXZ7BYcCCo3UwqHz0EPm2I3OMSYMYrfYv43C2MBc/edit?usp=sharing>

If interested click the Google Docs link above and request access to the sign-up sheet. Based on the number of requests I'll determine how long they can be checked out for.

CIS Fundraising "Bake Sale"

Donate by answering seven questions on an online CTE survey!

Perkins/VTEA Survey

Please complete the survey by Friday, April 6th

The screenshot shows a forum post on the 'Cabrillo College: Computer and Information Systems' forum. The post is titled 'Carl D. Perkins Vocational and Technical Education Act' and is posted by Rich Shimm. The post text explains the purpose of the Act, which is to provide federal funding for vocational and technical education. It also mentions that Cabrillo College is a recipient of this funding and that students are encouraged to complete the survey to help the college receive more funding. The post includes a link to the survey and a list of instructions for students.

This is an important source of funding for Cabrillo College.

Send me an email stating you completed this Perkins/VTEA survey for **three points extra credit!**

Career Technical Information	
Your answers to these questions will help qualify Cabrillo College for Perkins/VTEA grant funds.	
Are you currently receiving benefits from:	
<input type="radio"/> Yes	TANF/CALWORKS
<input type="radio"/> No	
<input type="radio"/> Yes	SSI (Supplemental Security Income)
<input type="radio"/> No	
<input type="radio"/> Yes	GA (General Assistance)
<input type="radio"/> No	
<input type="radio"/> Yes	Does your <u>income</u> qualify you for a fee waiver?
<input type="radio"/> No	
<input type="radio"/> Yes	Are you a single parent with custody of one or more minor children?
<input type="radio"/> No	
<input type="radio"/> Yes	Are you a <u>displaced homemaker</u> attending Cabrillo to develop job skills?
<input type="radio"/> No	
<input type="radio"/> Yes	Have you moved in the preceding 36 months to obtain, or to accompany parents or spouses to obtain, temporary or seasonal employment in agriculture, dairy, or fishing?
<input type="radio"/> No	

<https://opus-ii.cis.cabrillo.edu/forum/viewtopic.php?f=6&t=349>

New commands



Lesson 8 commands for your toolbox

NEW

find - Find file or content of a file

NEW

grep - "Global Regular Expression Print"

NEW

sort - sort

NEW

spell - spelling correction

wc - word count

NEW

tee - split output

NEW

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

Get the *names* file to use for the next series of slides

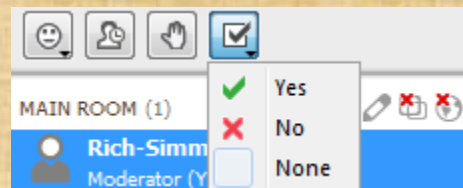
/home/cis90/simben \$ **cd** *return to home directory*

/home/cis90/simben \$ **cp** *relative path to the names file in the depot directory* **../depot/names** .

Think of the single dot file as "here" (it is hard linked to the current directory)

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

duke
benji
star
homer



Give me a green Yes check if you get the same results

Pretend you are a
command
(use your great
imagination)

Shell Steps

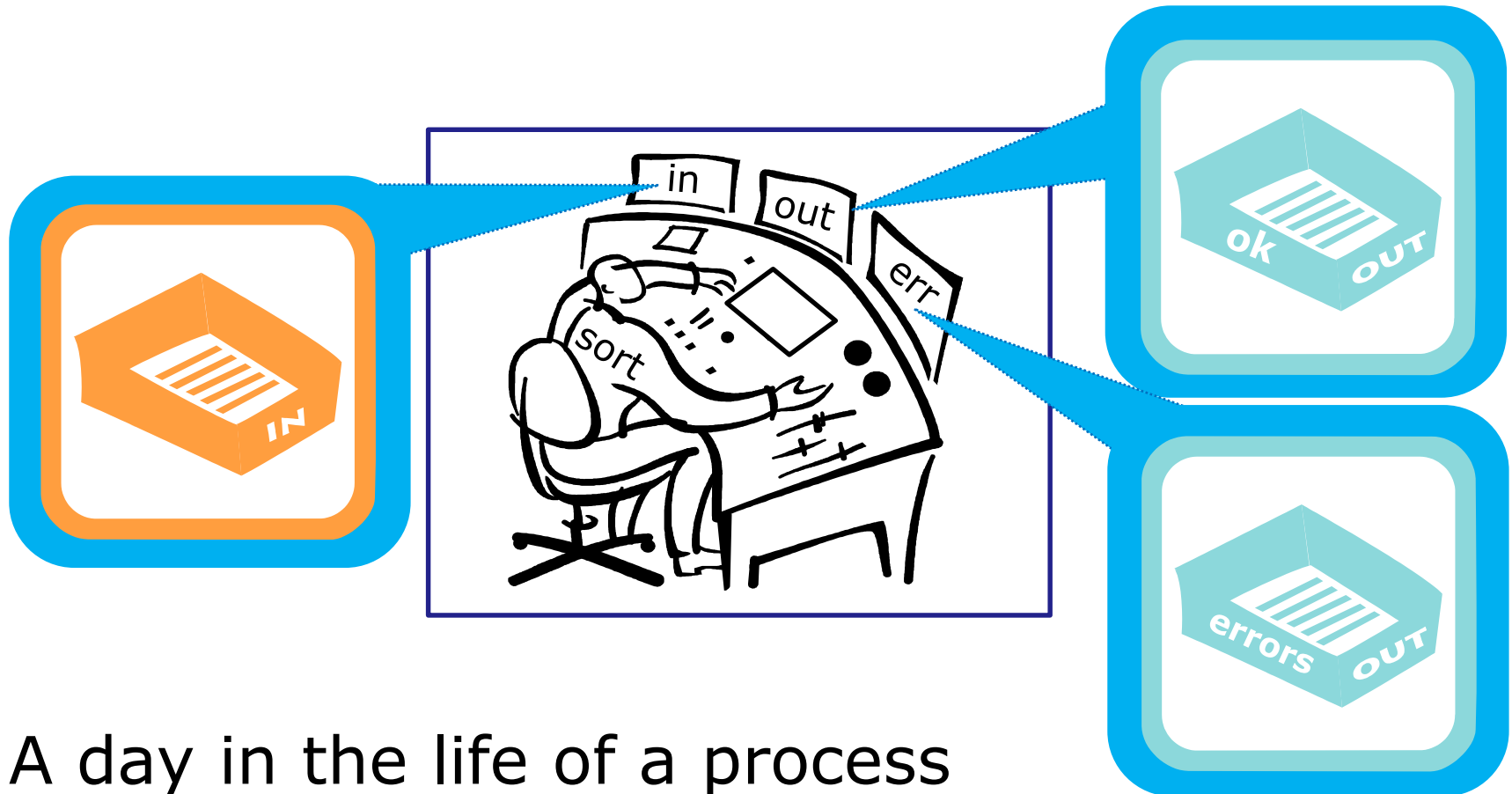
- 1) Prompt
- 2) Parse
- 3) Search
- 4) Execute
- 5) Nap
- 6) Repeat

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



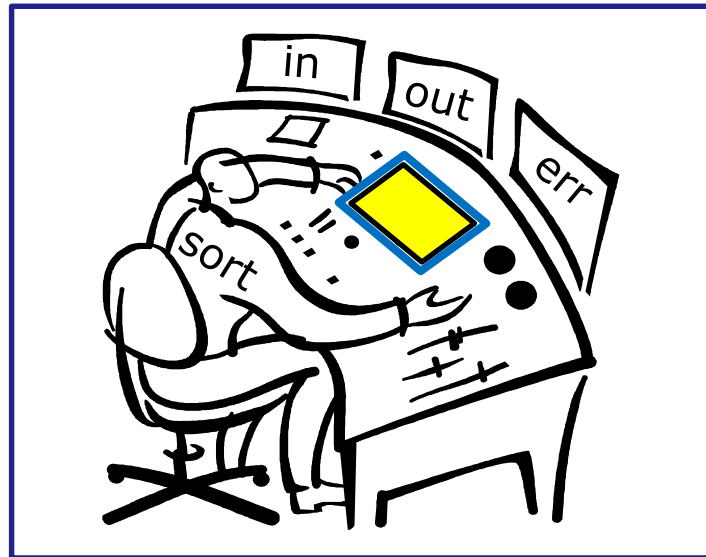
A day in the life of a process

*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

sort command with a filename argument

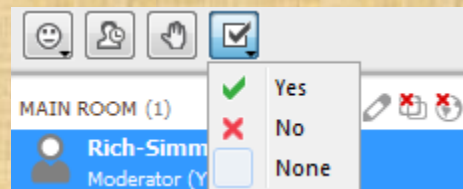
```
/home/cis90/simben $ cat names
```

```
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



Give me a green Yes check if you get the same results

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

Shell Steps

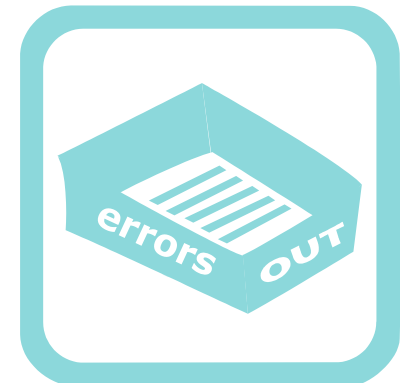
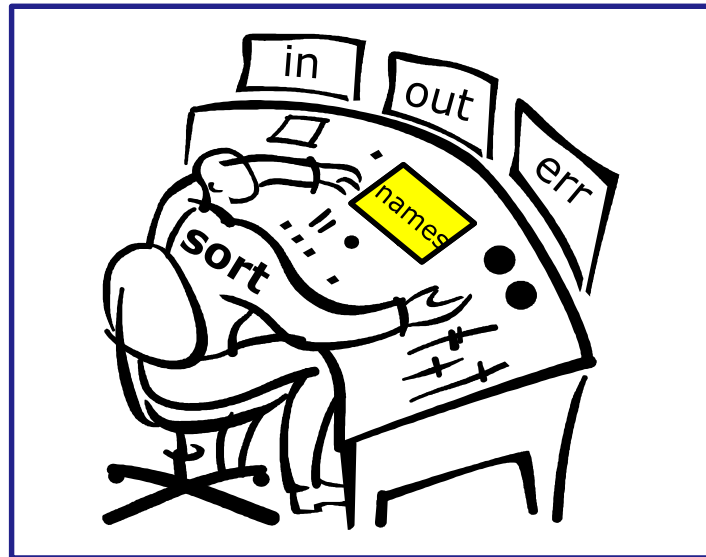
- 1) Prompt
- 2) Parse
- 3) Search
- 4) Execute
- 5) Nap
- 6) Repeat

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

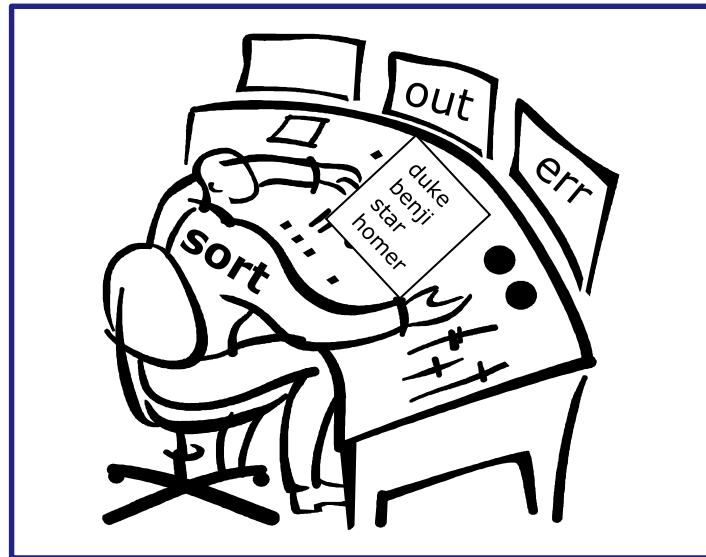
- 1) Prompt
- 2) Parse
- 3) Search
- 4) Execute
- 5) Nap
- 6) Repeat

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



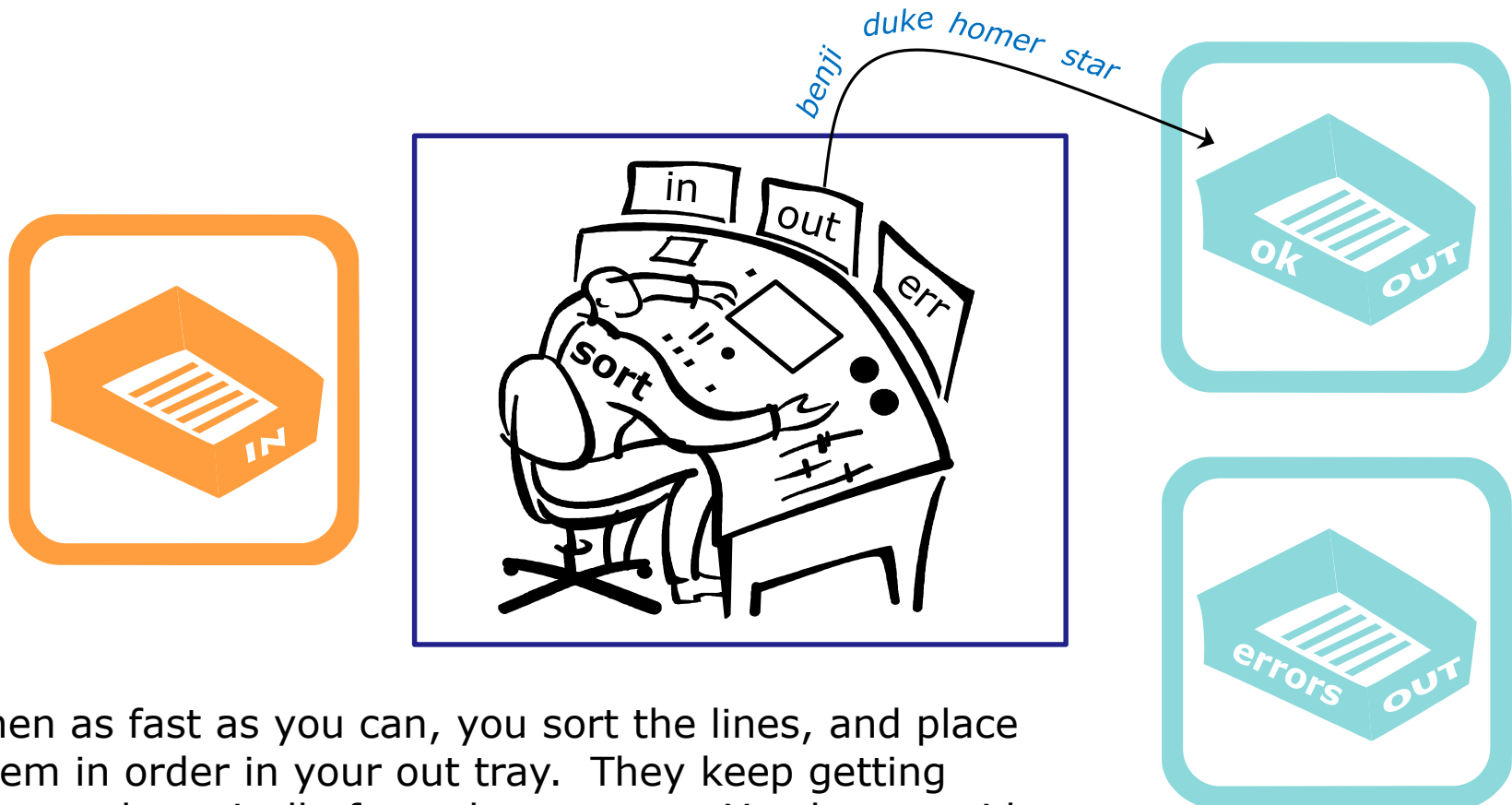
You (the sort process) check your instruction window and see the shell passed one argument "names" to you. You know (given your internal DNA code) that you must contact the kernel and request this file be opened and the contents read.

```
/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

sort command with no arguments

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

kayla

sky

bella

benji

charlie

bella

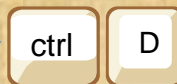
benji

charlie

kayla

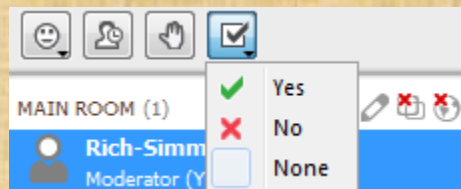
sky

*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



Give me a green Yes check if you get the same results

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

Shell Steps

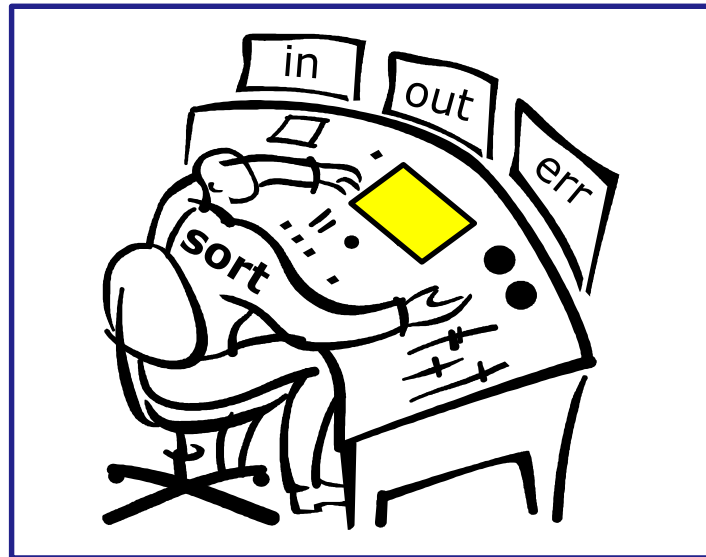
- 1) Prompt
- 2) Parse
- 3) Search
- 4) Execute
- 5) Nap
- 6) Repeat

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

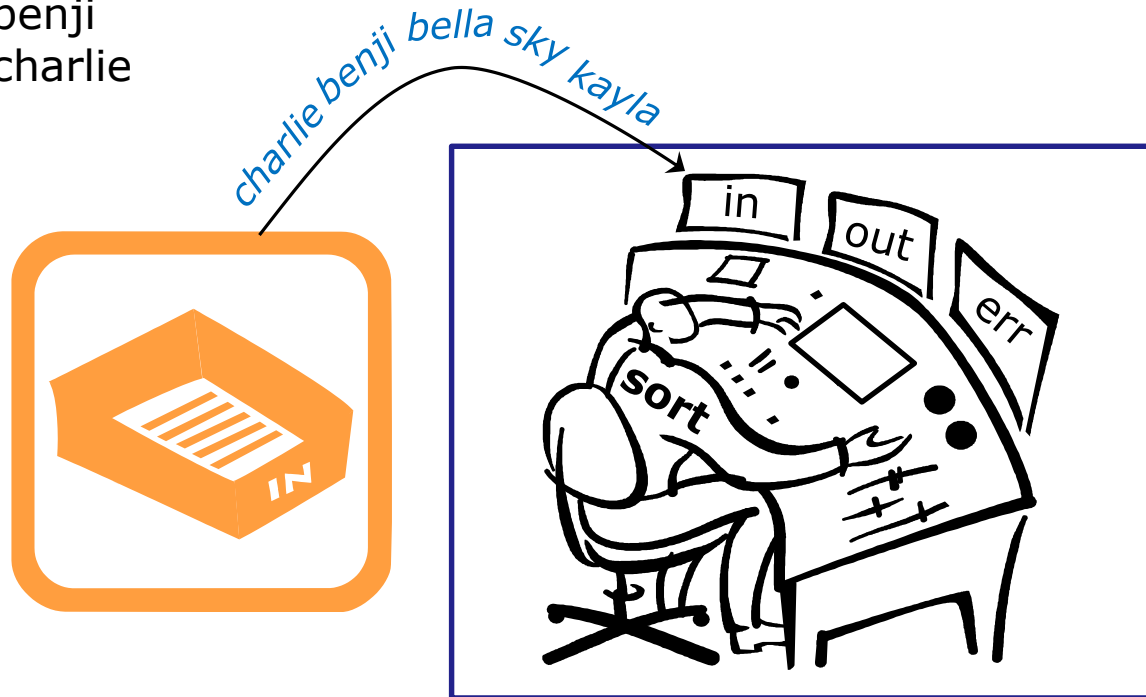
```
/home/cis90/simben $ sort
```



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



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
```

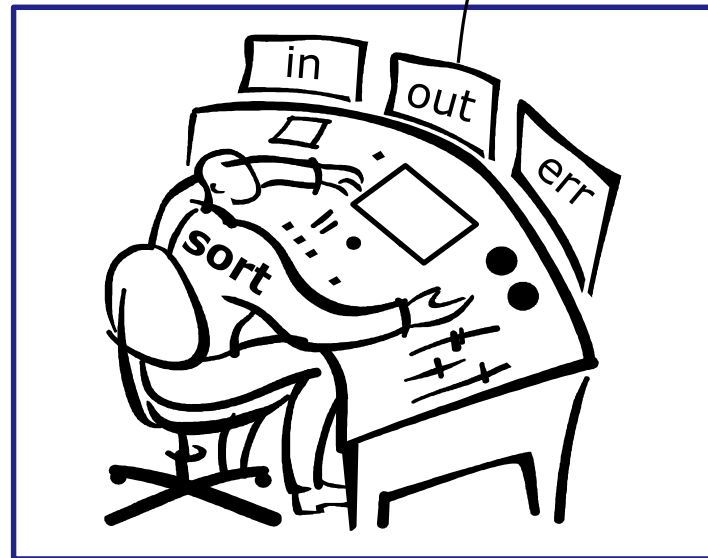


EOF

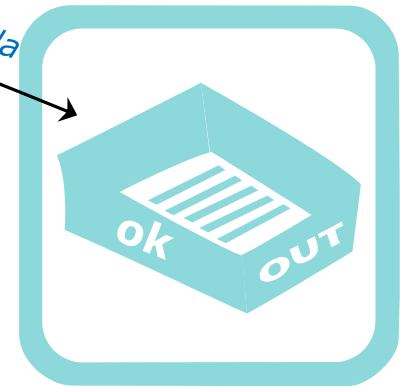


Then suddenly, when you reach for the next line, you find an EOF. You know (your internal DNA code) that this EOF means no more lines coming. You must sort what you have collected so far and place them, in order, into your out tray.

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



sky kayla charlie benji bella



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 $
```

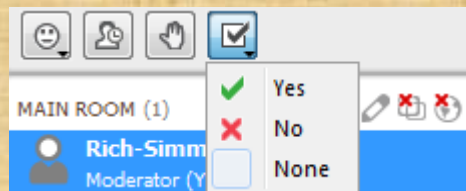
 *No such file*

Activity

sort command with 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



*Give me a green Yes check
if you get the same results*

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

Shell Steps

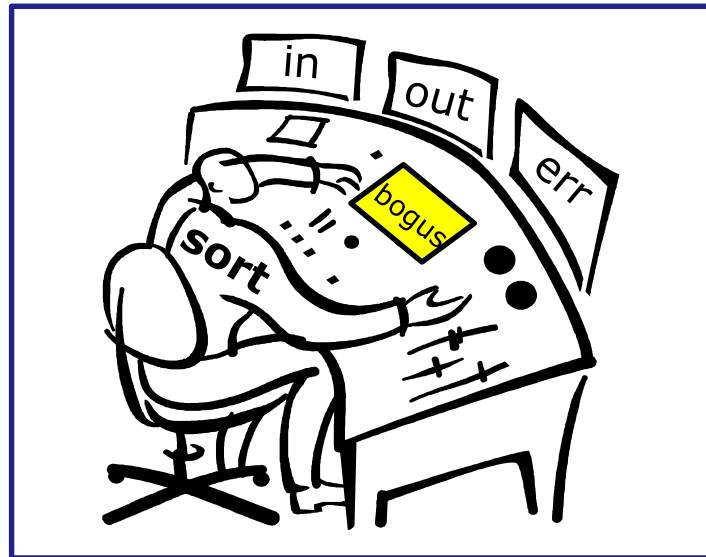
- 1) Prompt
- 2) Parse
- 3) Search
- 4) Execute
- 5) Nap
- 6) Repeat

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**

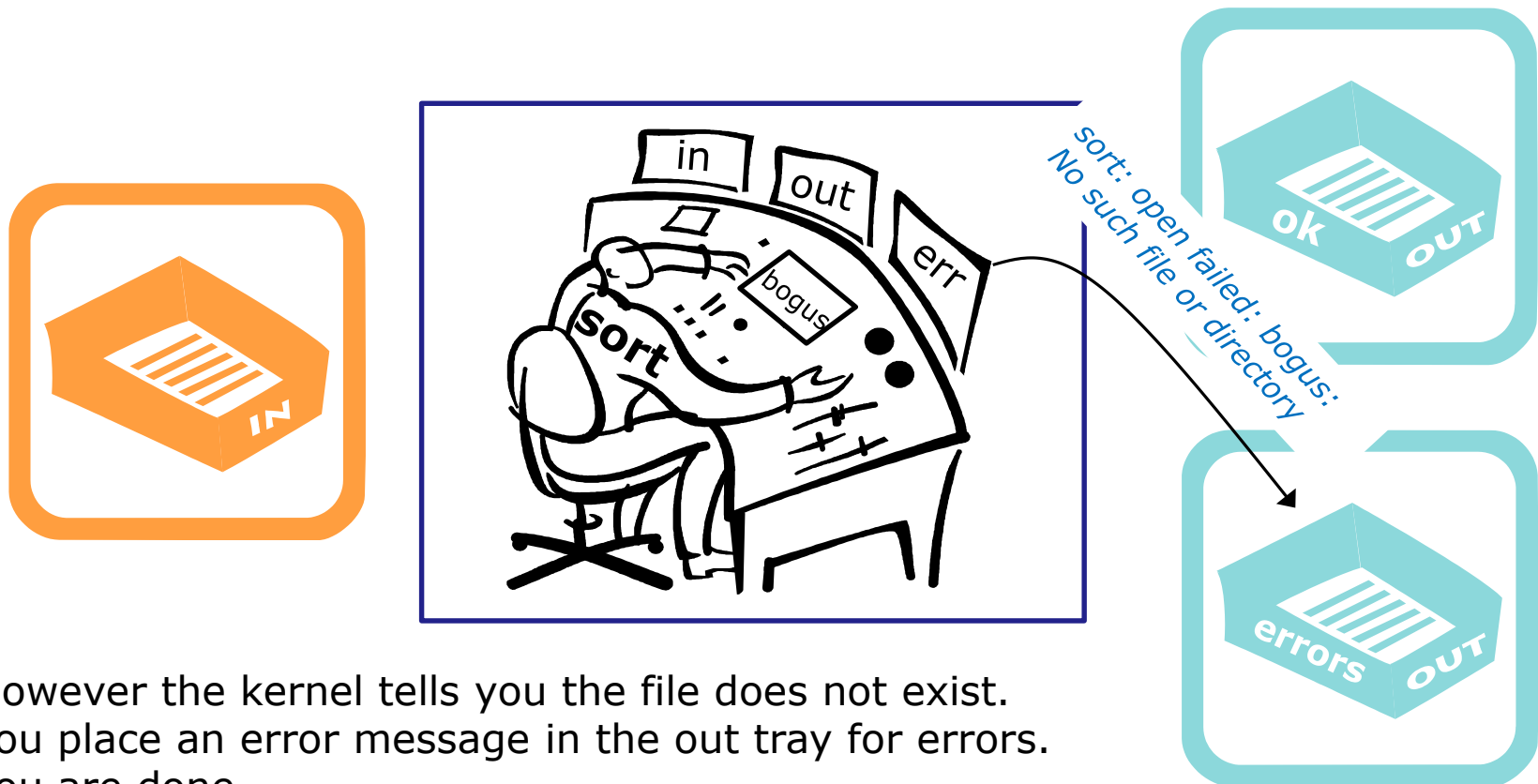
Shell Steps

- 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
```



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)

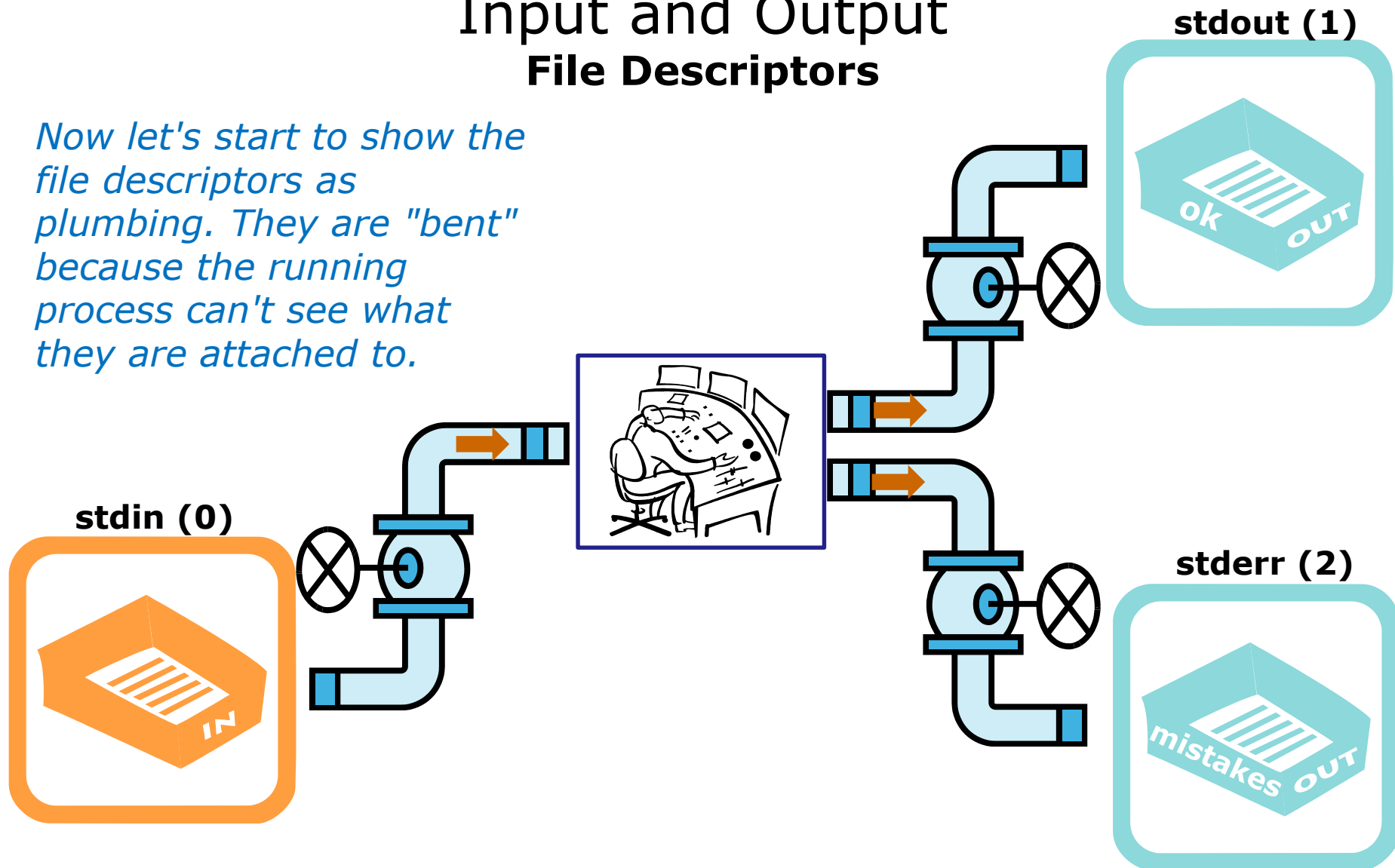


stderr (2)



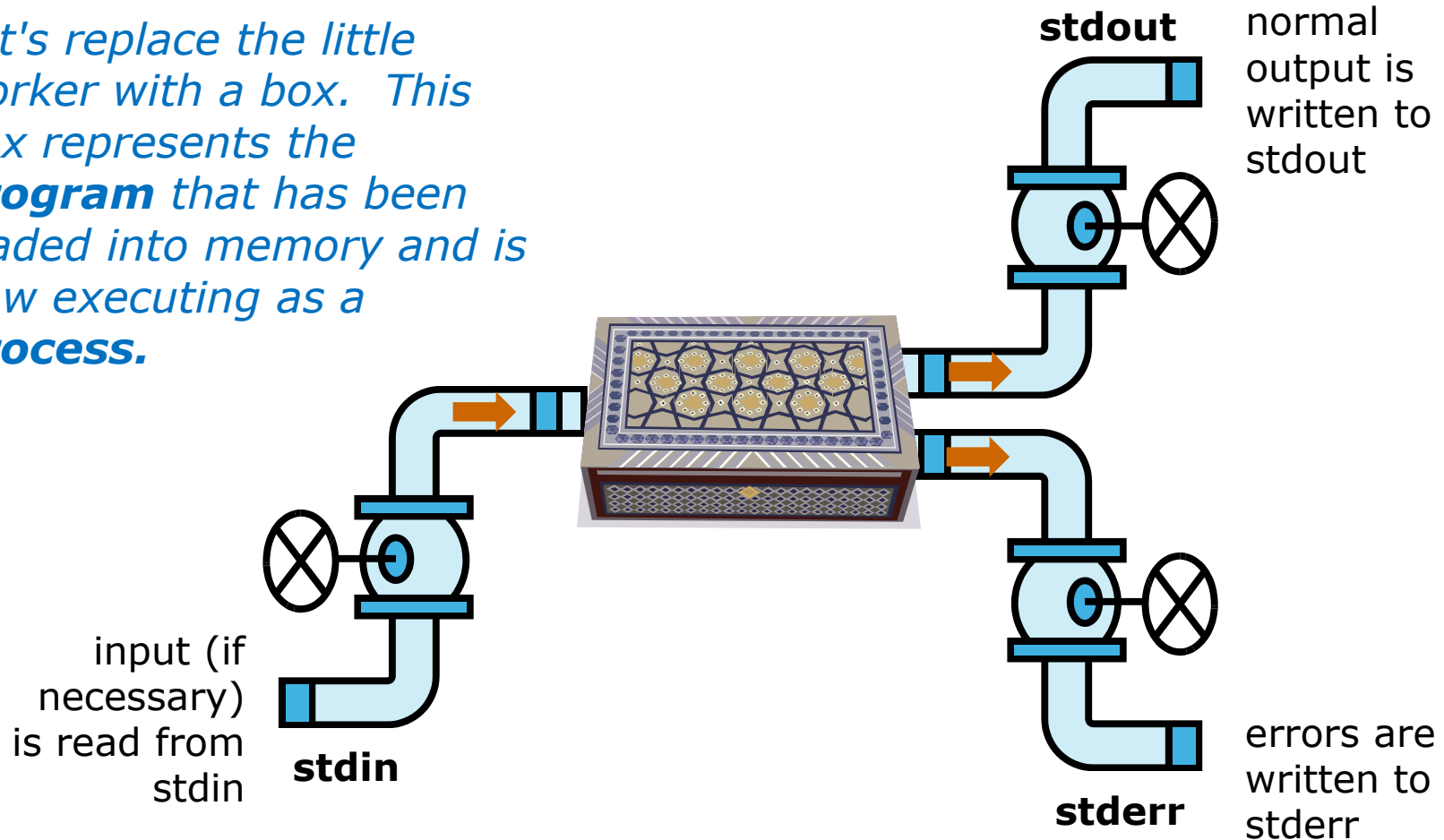
Input and Output File Descriptors

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**.

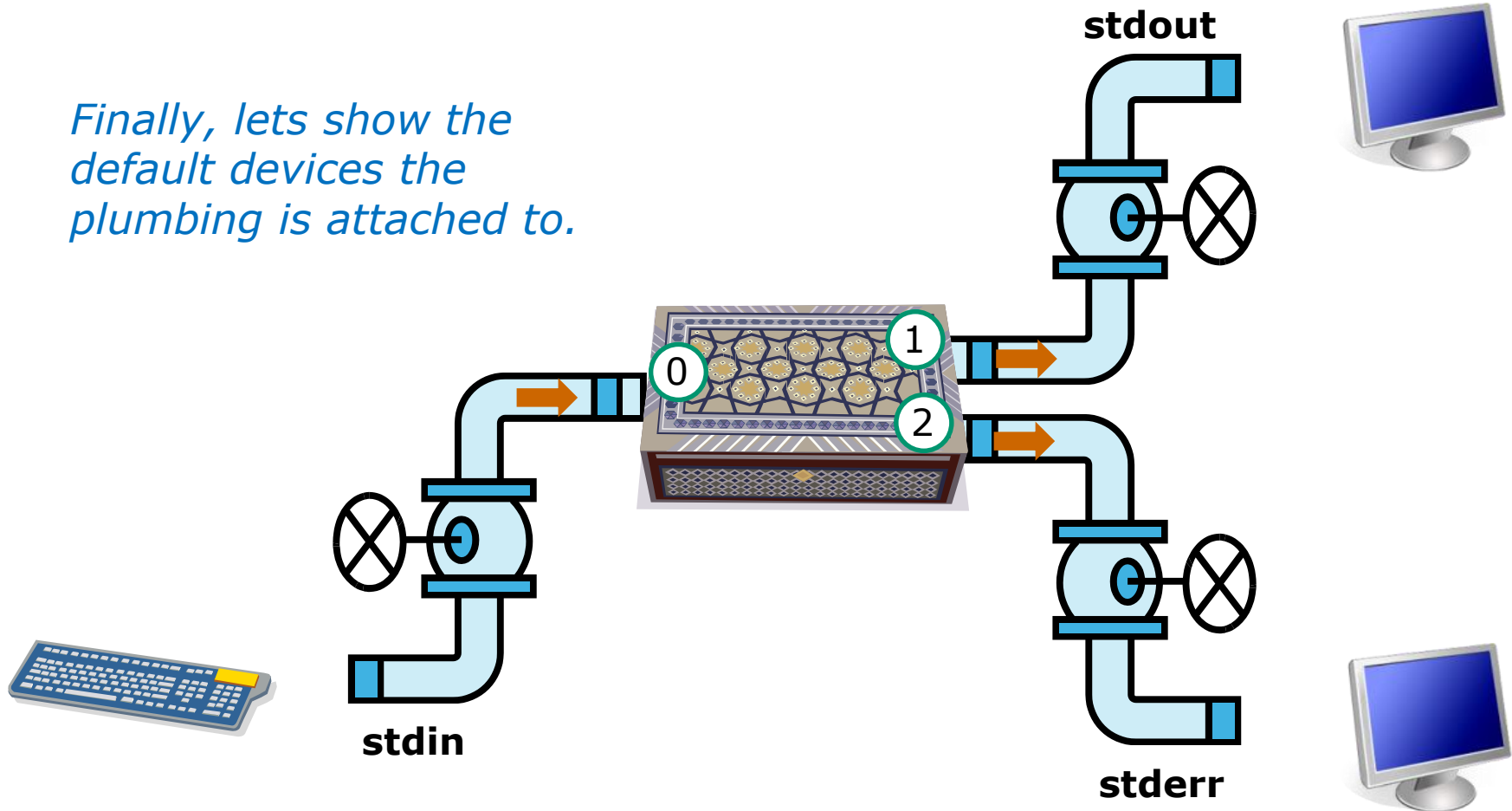


Input and Output

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)

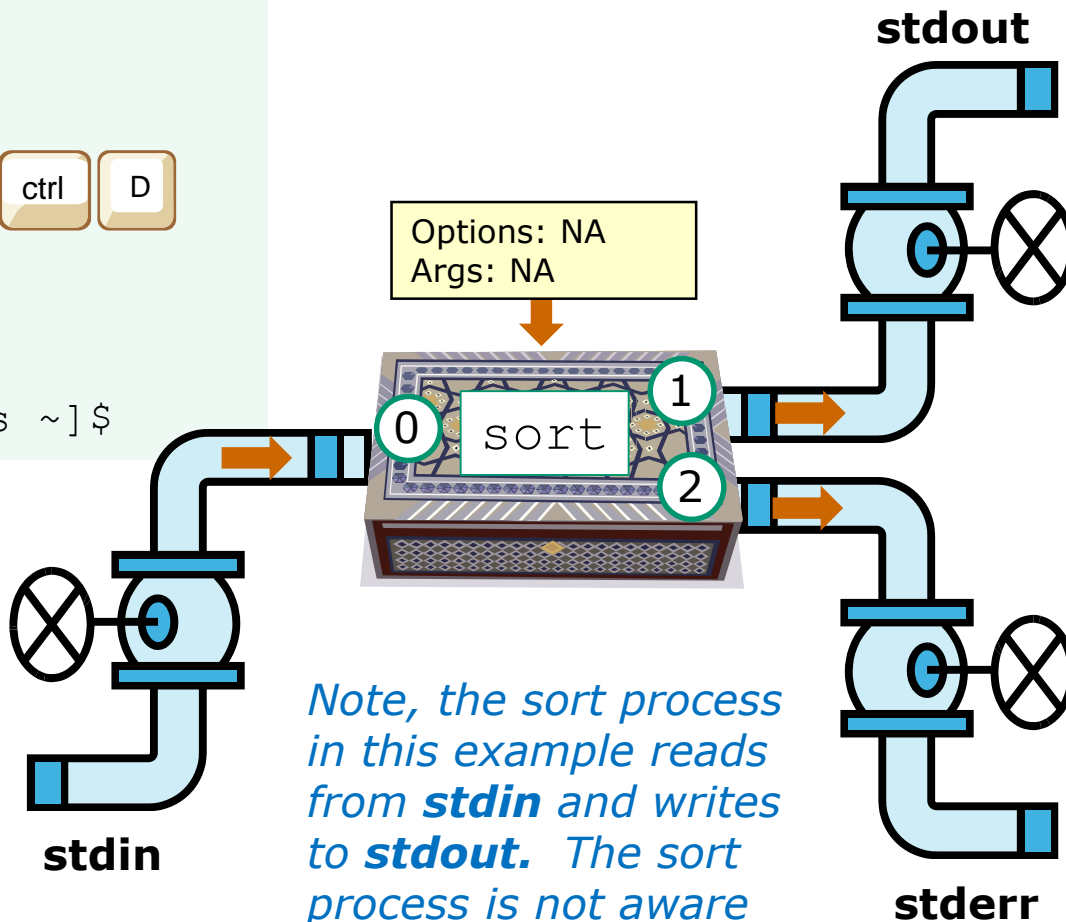
By default is attached to the user's terminal device (screen)

The sort example again with no arguments

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



star
benji
duke
homer



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*

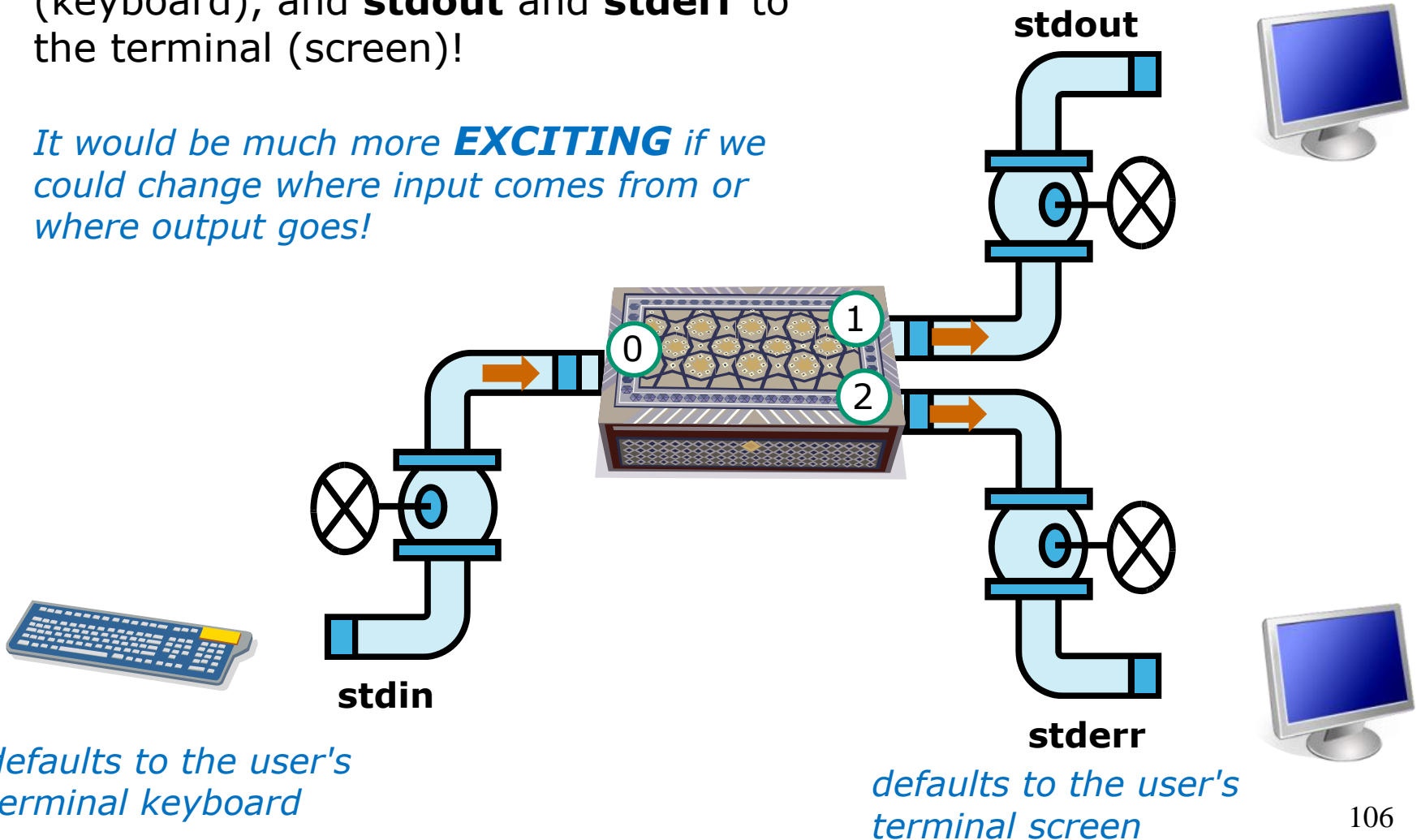


File Redirection

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

defaults to the user's terminal screen

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




Input and Output

File Redirection

*Let's look at the
sort example again*

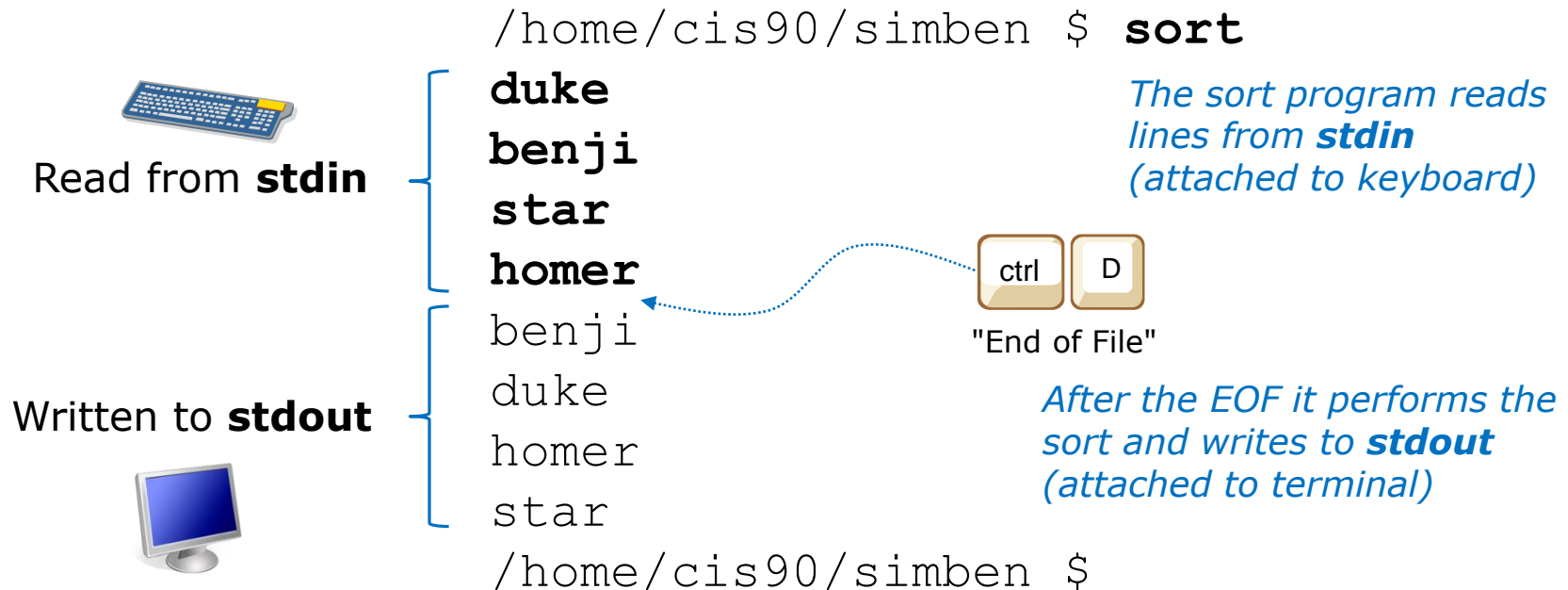
```
/home/cis90/simben $ sort  
duke  
benji  
star  
homer  
benji  
duke  
homer  
star  
/home/cis90/simben $
```



The diagram illustrates the effect of pressing Ctrl+D (EOF) during a command execution. It shows two buttons labeled 'ctrl' and 'D' with a blue dotted arrow pointing from them to the word 'benji' in the output of the 'sort' command. Below the buttons is the text '"End of File"'. The output of the 'sort' command is shown as: **duke**, **benji**, **star**, **homer**, benji, duke, homer, star. The words 'duke', 'benji', 'star', and 'homer' are in bold, while 'benji', 'duke', 'homer', and 'star' are in regular font. The 'benji' in regular font is the one indicated by the arrow, showing that the command was interrupted before it could finish sorting the remaining input.

Input and Output

File Redirection



sort command (no arguments)

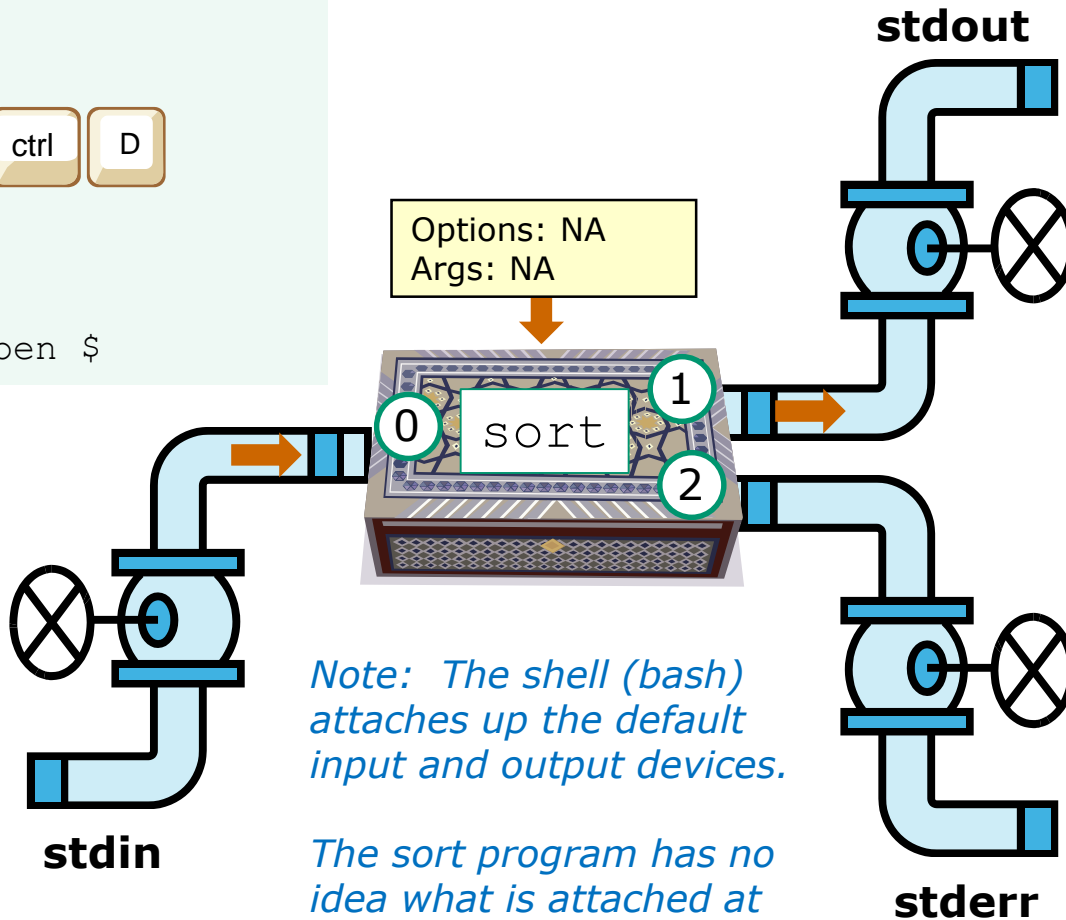
```
/home/cis90/simben $ sort
duke
benji
star
homer
benji
duke
homer
star
/home/cis90/simben $
```



/dev/pts/0



duke
benji
star
homer



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

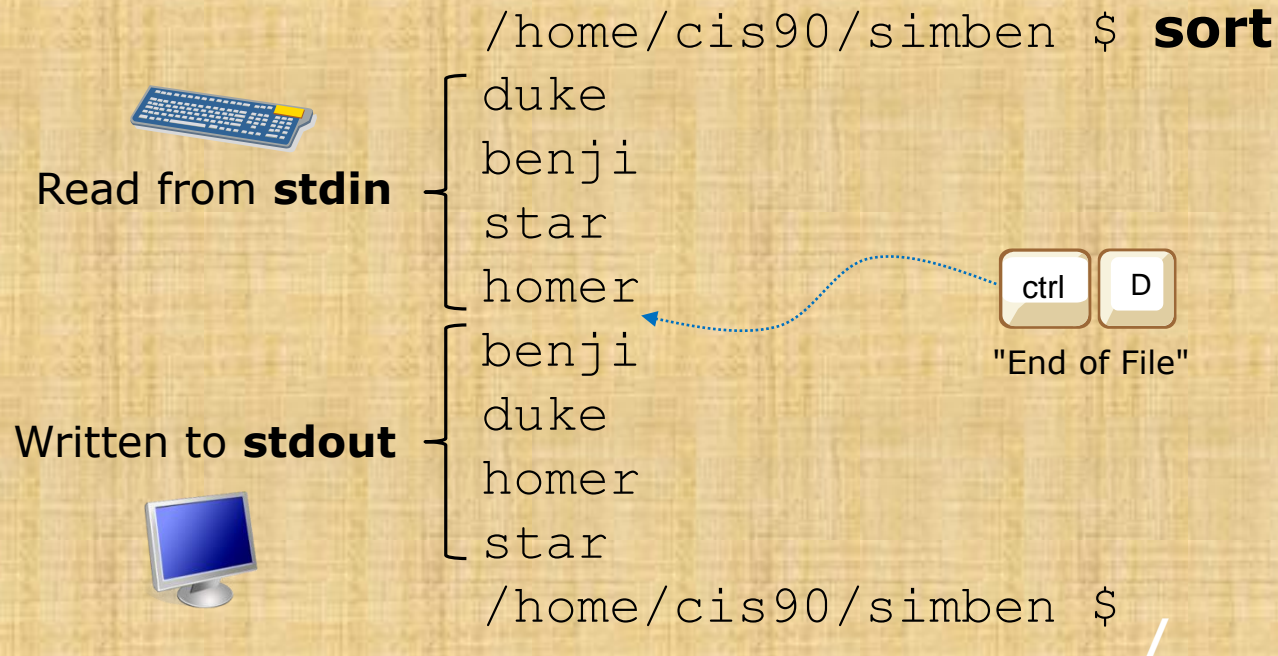
The sort program has no idea what is attached at the end of the pipes.

/dev/pts/0



benji
duke
homer
star

Activity

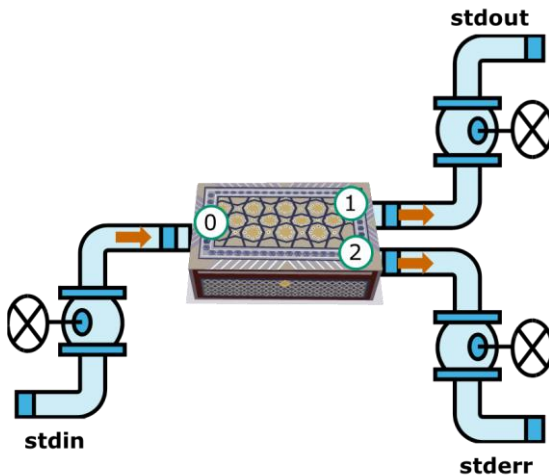


When YOU do this. What specific device is stdin and stdout attached to? Write your answer in the chat window.

Input and Output

File Redirection

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



~~0~~< **filename**

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

~~1~~> **filename**

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

2> **filename**

*To redirect **stderr***

>> **filename**

*To redirect **stdout** and append*

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



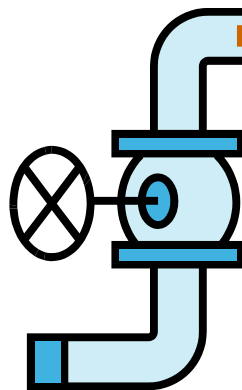
dogsinorder

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

/dev/pts/0



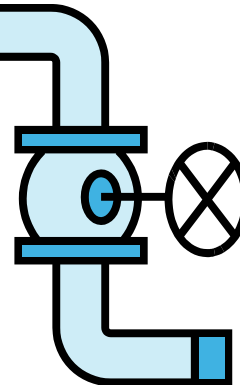
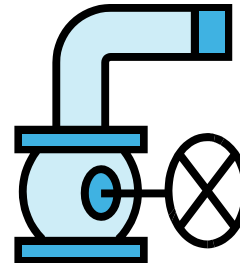
```
duke  
benji  
star  
homer
```



stdin

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

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



stderr

Now you try it

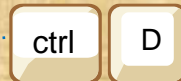
```
[simmsben@opus-ii ~]$ sort > dogsinorder
```

duke

benji

star

homer



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

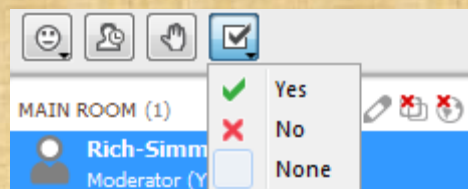
benji

duke

homer

star

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



*Give me a green Yes check
if you get the same results*

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 *dogsinorder*

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

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

```
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

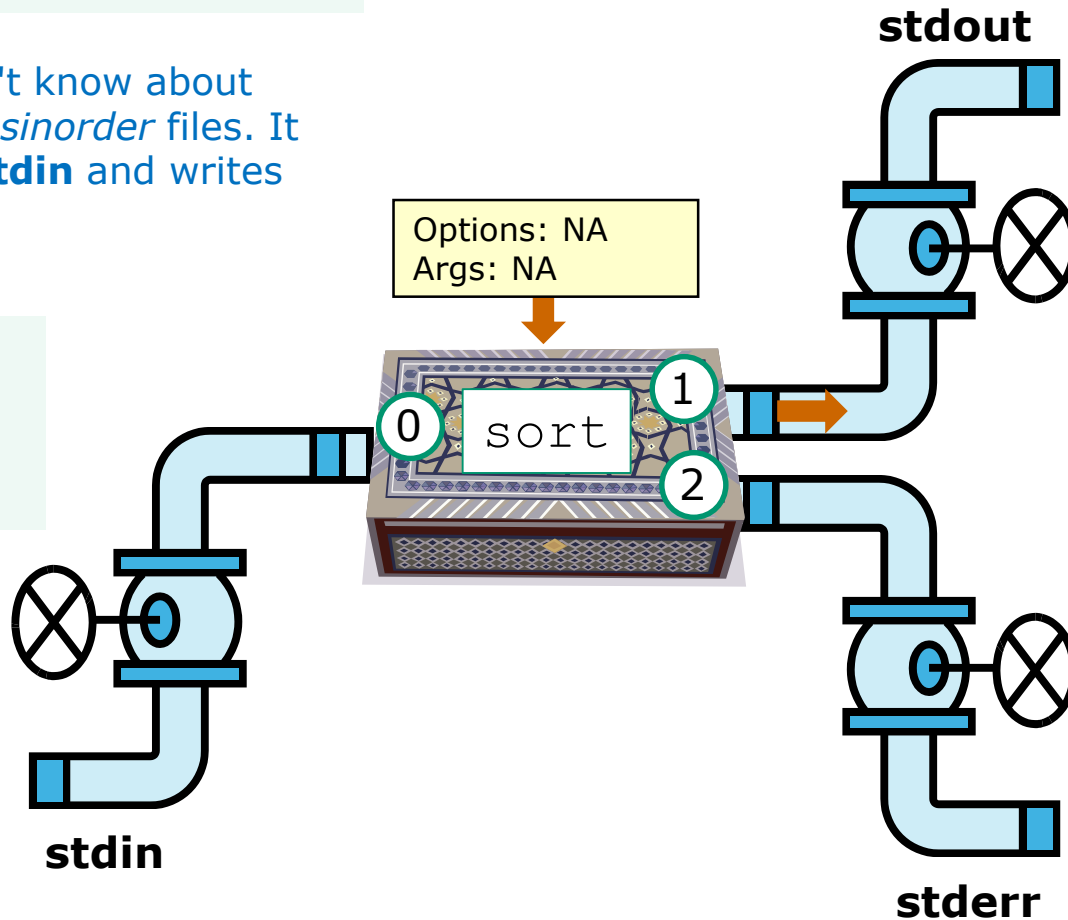
```
$ sort < names > dogsinorder
```

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



dogsorder

```
$ 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 from 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.

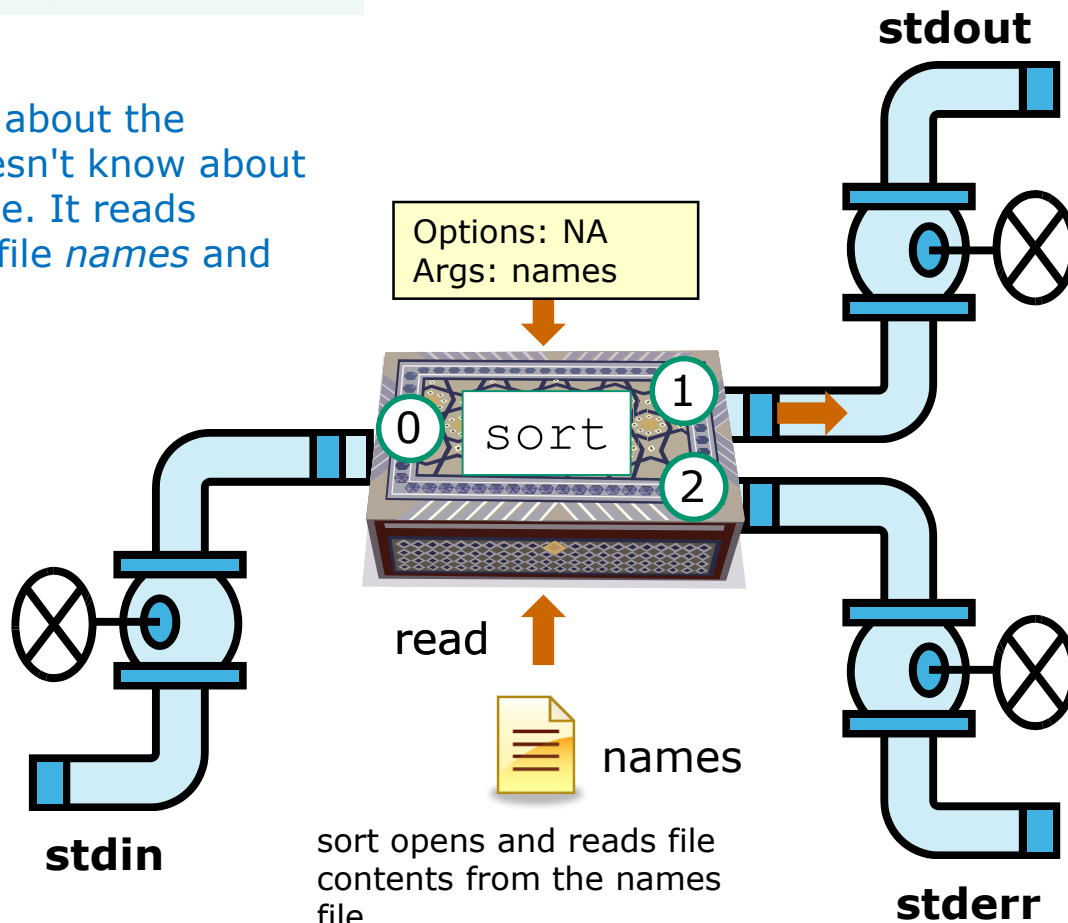
Корисне для
наступного
вікторини!

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!



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

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 ~]$
```

Корисне для
наступного
вікторини!

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

Put your answer in the chat window

yes

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 *dogsিনorder*

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

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

```
star
```

```
homer
```

```
duke
```

```
benji
```

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

This -r option does the sort in
reverse order

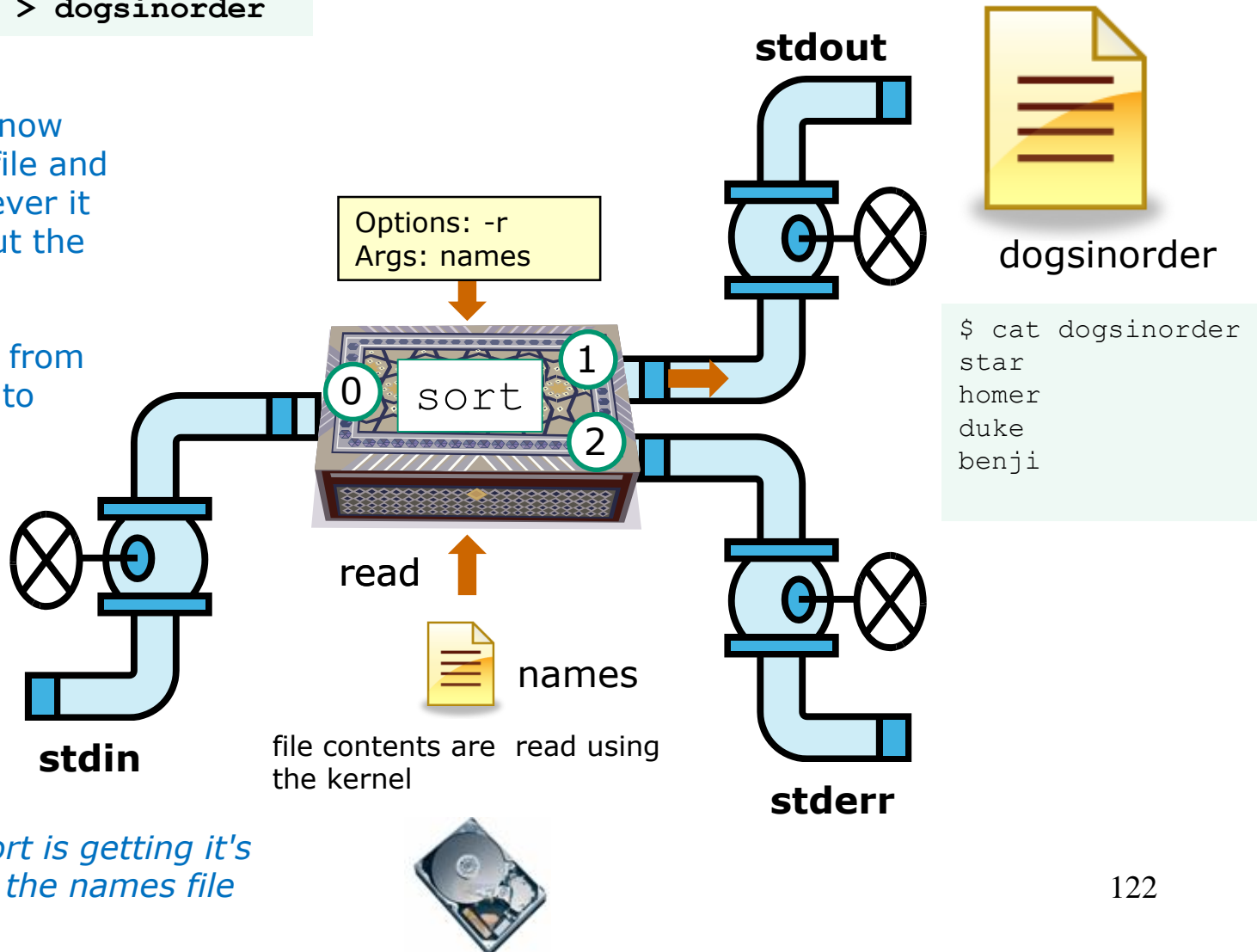
The shell opens the *dogsিনorder*
file. The sort process is not aware
that output is redirected there.

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 $
```

Корисне для
наступного
вікторини!

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

Put your answer in the chat window

no

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

More redirection examples

Example 1

Input from file, redirecting stdout to another terminal device

/dev/pts/0

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

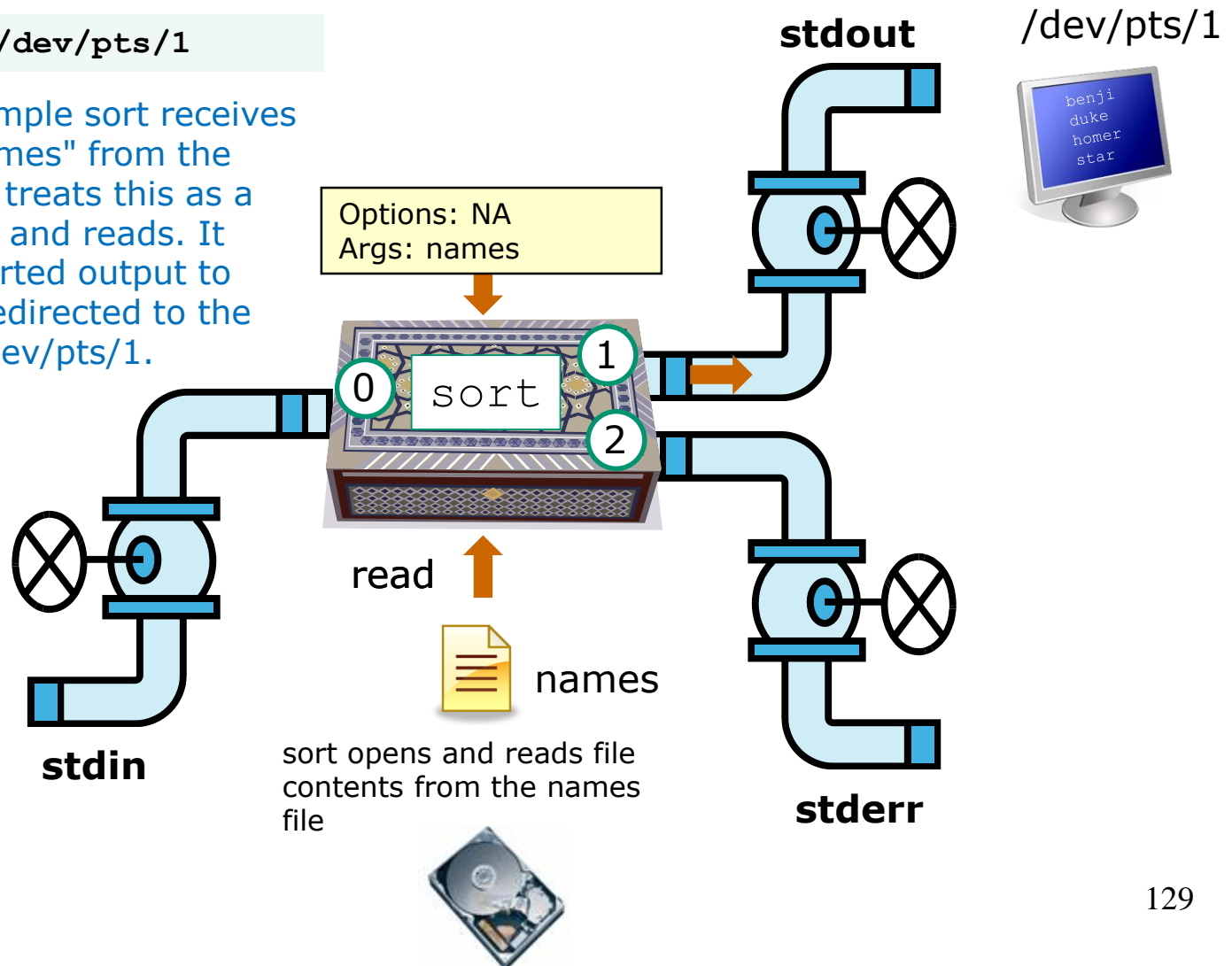
```
[simben@opus ~]$ tty
/dev/pts/1
[simben@opus ~]$ benji
duke
homer
star
```

Example 1 diagram

Input from stdin, redirecting stdout to another terminal device

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

Note: In this example sort receives the argument "names" from the command line. It treats this as a file which it opens and reads. It then writes the sorted output to **stdout** which is redirected to the terminal device /dev/pts/1.



Example 2

Input from the command line, redirecting stdout to file

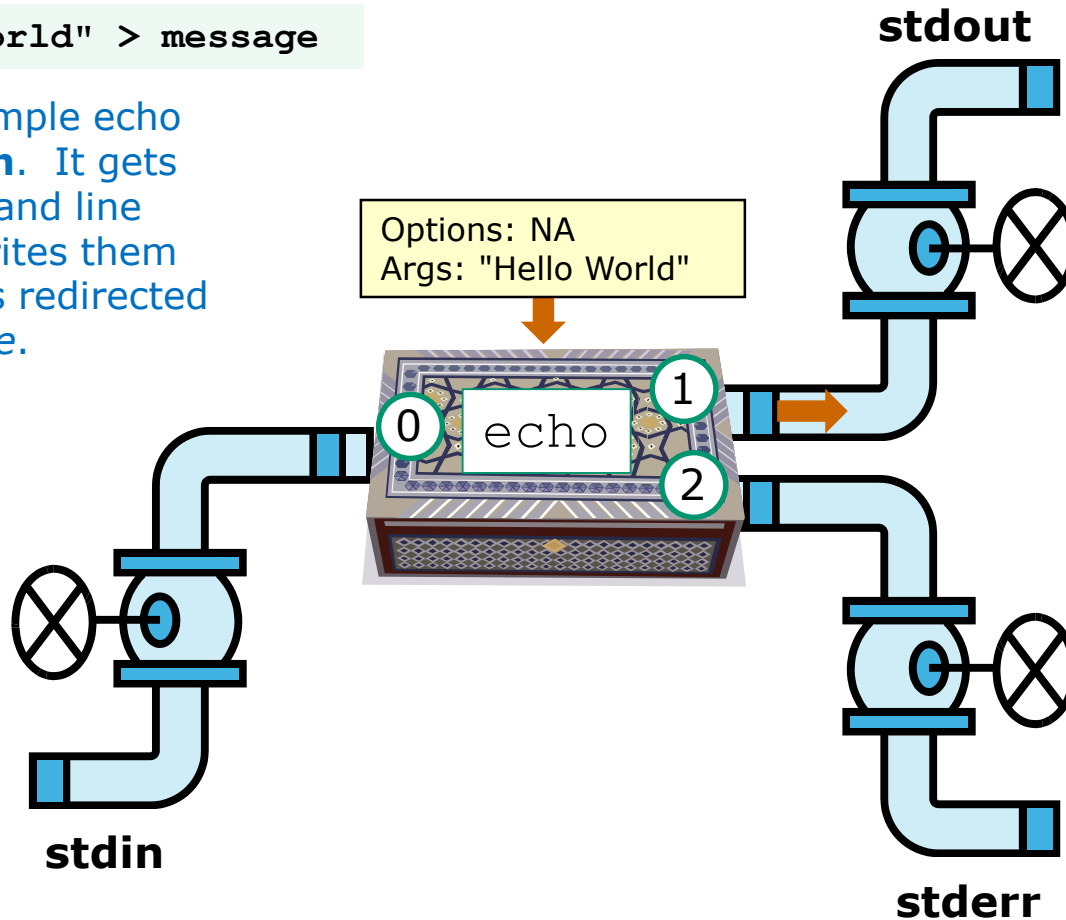
```
/home/cis90/simben $ echo "Hello World" > message  
/home/cis90/simben $ cat message  
Hello World  
/home/cis90/simben $
```

Example 2 diagram

Input from the command line, redirecting stdout to file

```
$ echo "Hello World" > message
```

Note: In this example echo does not use **stdin**. It gets its input as command line arguments and writes them to **stdout** which is redirected to the file *message*.



message

```
$ cat message
Hello World
```

Example 3

Input from command line and OS, redirecting stdout and stderr

```
[simben@opus ~]$ ls -lR > snapshot
```

```
ls: ./Hidden: Permission denied
```

```
[simben@opus ~]$ head -10 snapshot
```

```
..:
```

```
total 296
```

```
-rw-rw-r-- 1 simben cis90 51 Sep 24 17:13 1993
```

```
-rw-r--r-- 21 guest90 cis90 10576 Jul 20 2001 bigfile
```

```
drwxr-x--- 2 simben cis90 4096 Oct 8 09:05 bin
```

```
drwx--x--- 4 simben cis90 4096 Oct 8 09:00 class
```

```
-rw----- 1 simben cis90 484 Sep 24 18:13 dead.letter
```

```
drwxrwxr-x 2 simben cis90 4096 Oct 8 09:05 docs
```

```
-rw-rw-r-- 1 simben cis90 22 Oct 20 10:51 dogsinorder
```

```
drwx----- 2 simben cis90 4096 Oct 16 09:17 edits
```

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

```
[simben@opus ~]$ ls -lR > snapshot 2> errors
```

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

```
ls: ./Hidden: Permission denied
```

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

*Note: errors are written to **stderr**, which is attached by default to the terminal*

*> redirects **stdout** to file named snapshot*

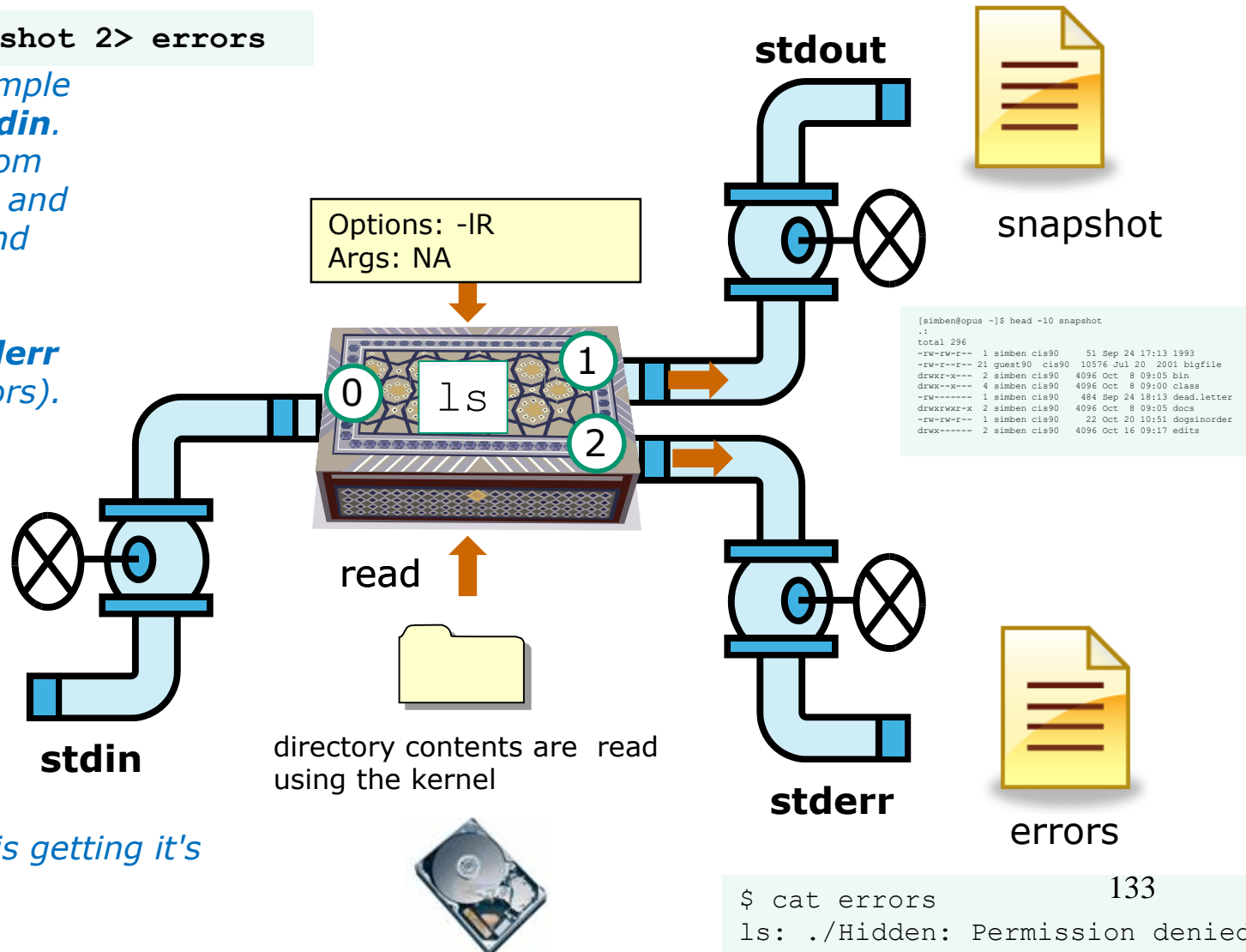
*2> redirects **stderr** to file named errors*

Example 3 diagram

Input from command line and OS, redirecting stdout and stderr

```
$ ls -lR > snapshot 2> errors
```

*Note: In this example ls does not use **stdin**. It gets its input from the command line and the OS (kernel) and writes to **stdout** (redirected to snapshot) and **stderr** (redirected to errors).*



In this example, ls is getting its input from the OS

Redirection Practice

Activity

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

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

Activity

```
/home/cis90/simben $ echo 2+2 > math
```

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

```
2+2
```

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

```
4
```

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

*Redirecting stdin
to the math file*

```
/home/cis90/simben $ echo 4/0 > math
```

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

```
4/0
```

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

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

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

*Redirecting stdin
to the math file*

Activity

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

*Note that >> appends
the output to stdout.*

Activity

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

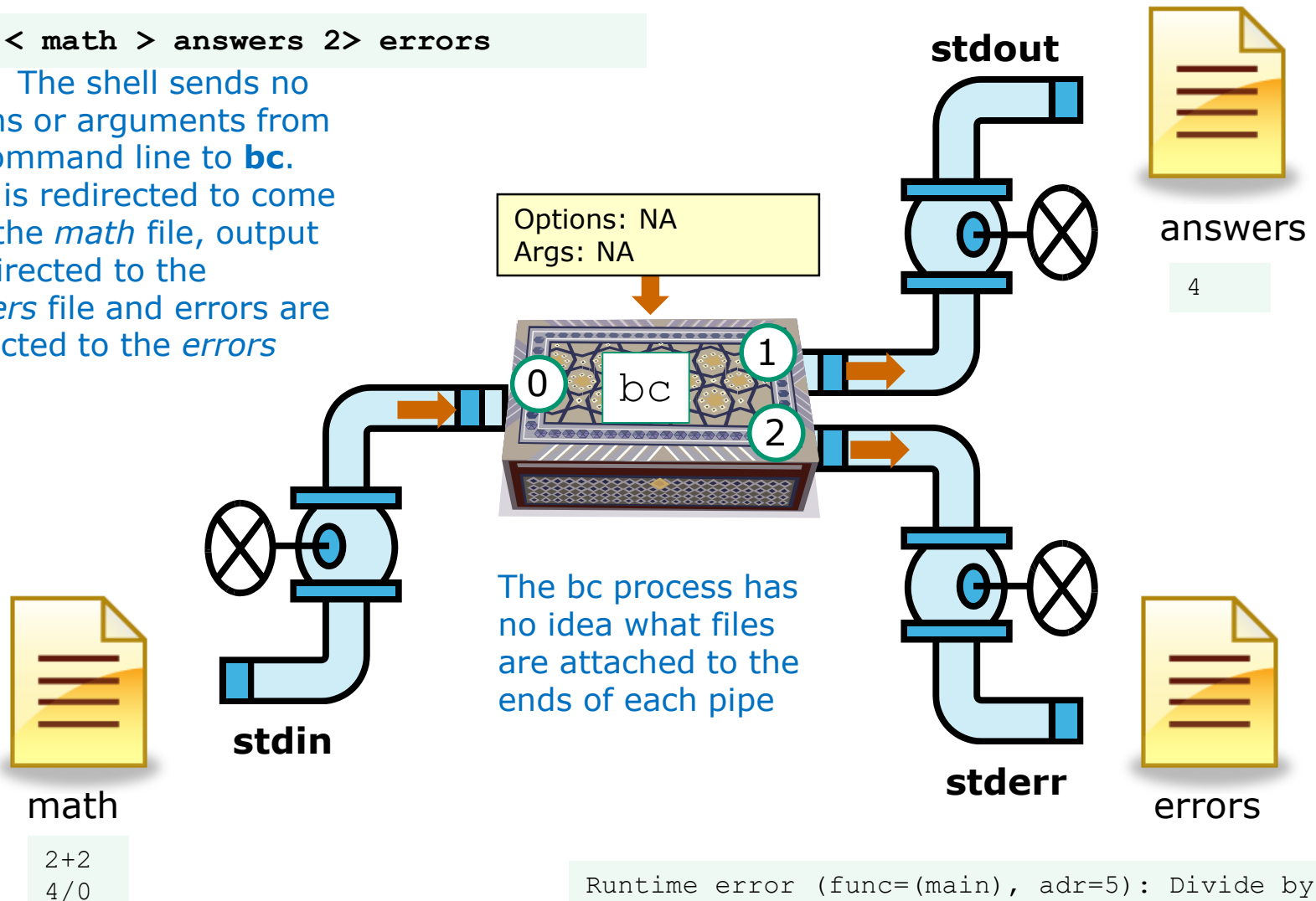
This time we redirect stdin, stdout and stderr!

Example 4 diagram

Redirecting stdin, stdout and stderr

```
$ 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.

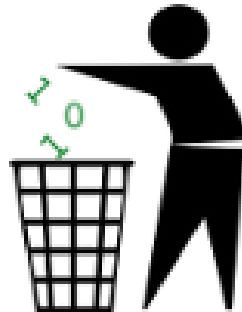


The bit bucket `/dev/null`

`/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 $
```

Корисно для
наступного
вікторини!

This is how you redirect output to the bit bucket

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

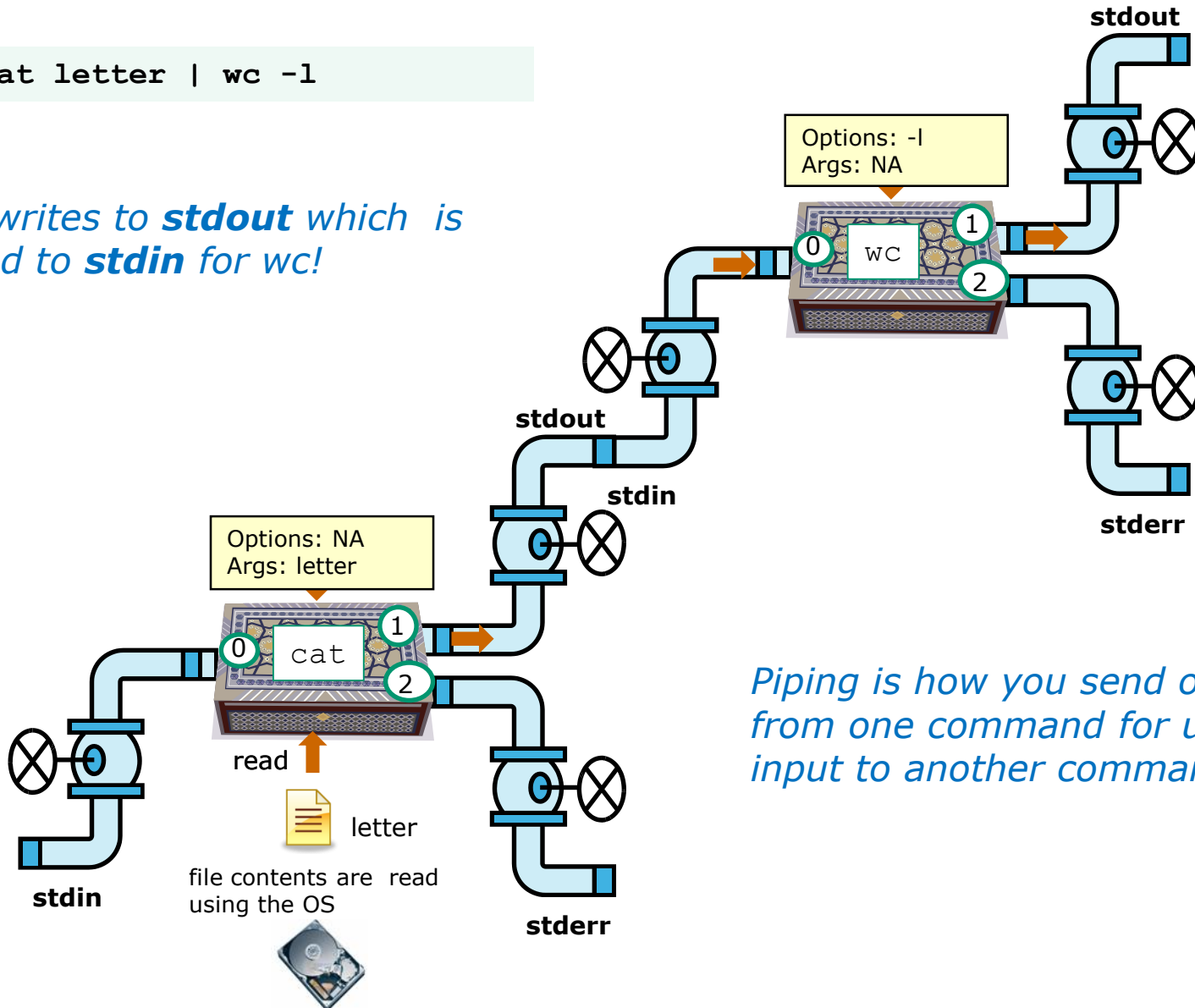
```
[simben@opus ~]$ cat letter | wc -l  
28
```

Counting the lines in the letter file

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

```
/home/cis90/simben $ cat letter | wc -l  
28
```

Counting the number of Shakespeare poems

```
/home/cis90/simben $ ls poems/Shakespeare/ | wc -l  
15
```

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

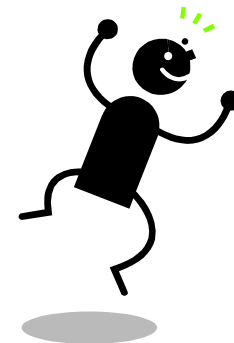
*redirecting stderr
to the "bit bucket"*

```
[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

The directory to start finding files

Redirect errors written to stderr to the bit bucket

```
/home/cis90/simben $ find /home -user root 2> /dev/null
```

The user that owns the files

```
/home  
/home/cis175  
/home/cis172  
/home/cis172/computers.txt  
/home/cis172/science.txt  
/home/lost+found  
/home/cis90/simben $
```

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

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

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 {} \;
```

The {} are replaced by filenames as they are found

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

```
find: `./Hidden': Permission denied
./editors/spellk: ASCII English text
./kshrc: ASCII text
./docs/MarkTwain: ASCII English text
./ssh/known_hosts: ASCII text, with very long lines
/home/cis90/simben $
```

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  
./edit1_small_town  
./edit1_better_town  
[simben@opus-ii ~]$
```

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

Filter commands

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*

[simben@opus 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

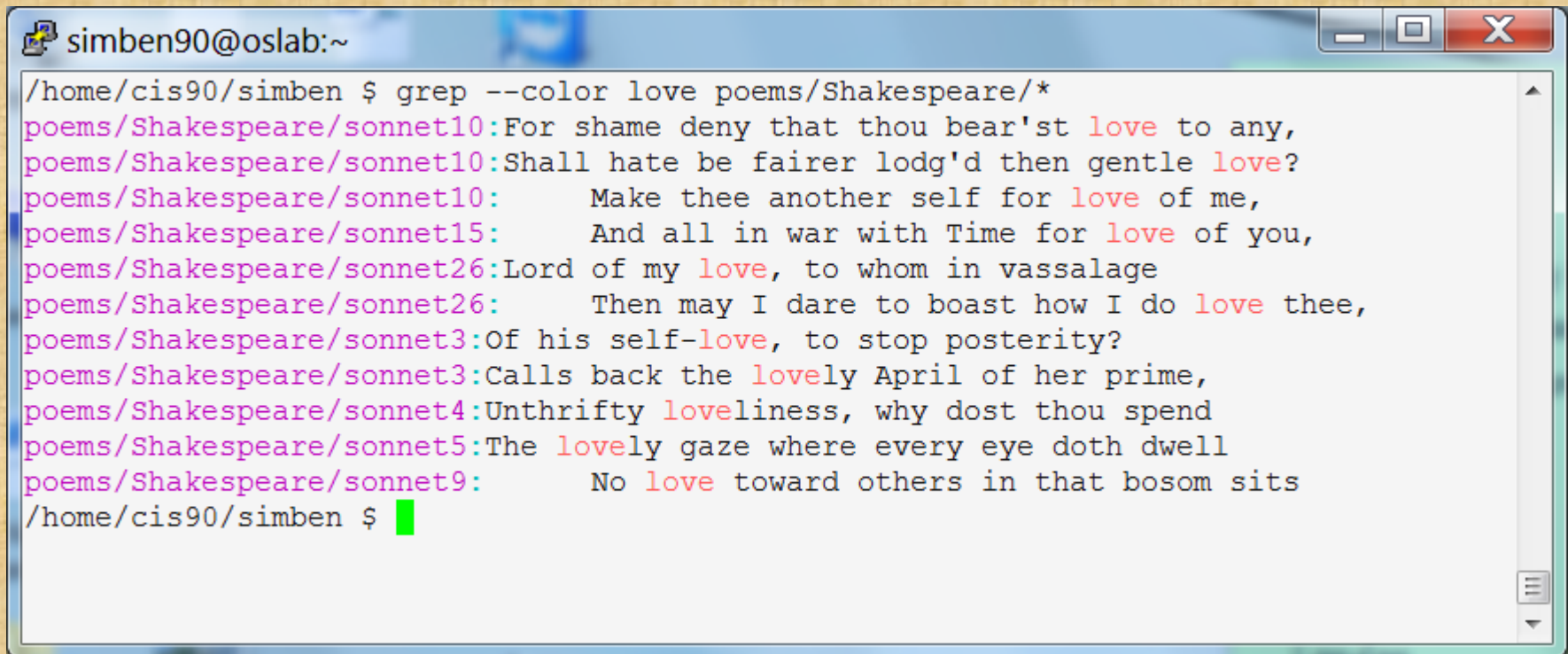
Looking for love in all the wrong places?

Find the string "love" in Shakespeare's sonnets

Now you try it

The color option

grep --color love poems/Shakespeare/*



```
simben90@oslab:~  
/home/cis90/simben $ grep --color love poems/Shakespeare/*  
poems/Shakespeare/sonnet10:For shame deny that thou bear'st love to any,  
poems/Shakespeare/sonnet10:Shall hate be fairer lodg'd then gentle love?  
poems/Shakespeare/sonnet10:      Make thee another self for love of me,  
poems/Shakespeare/sonnet15:      And all in war with Time for love of you,  
poems/Shakespeare/sonnet26:Lord of my love, to whom in vassalage  
poems/Shakespeare/sonnet26:      Then may I dare to boast how I do love thee,  
poems/Shakespeare/sonnet3:Of his self-love, to stop posterity?  
poems/Shakespeare/sonnet3:Calls back the lovely April of her prime,  
poems/Shakespeare/sonnet4:Unthrifty loveliness, why dost thou spend  
poems/Shakespeare/sonnet5:The lovely gaze where every eye doth dwell  
poems/Shakespeare/sonnet9:      No love toward others in that bosom sits  
/home/cis90/simben $
```

Searching for love with colors

grep the output of a grep

string to search for in the output of the previous command

string to search for *files to search contents of*

```
[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  
52:simben90:x:1201:190:Benji Simms:/home/cis90/simben:/bin/bash
```

*Found in line 52 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

Text string to search for

*Search recursively
(all sub-directories)*

*starting directory
(. is the current directory)*

discard permission errors

```
/home/cis90/simben $ grep -R kind . 2> /dev/null
./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,
./letter:Mother, Father, kindly disregard this letter.
./bin/enlightenment: echo "to find out what kind of file \"what_am_i\" is"
./misc/mystery: echo "to find out what kind of file \"what_am_i\" is"
```

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.

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>

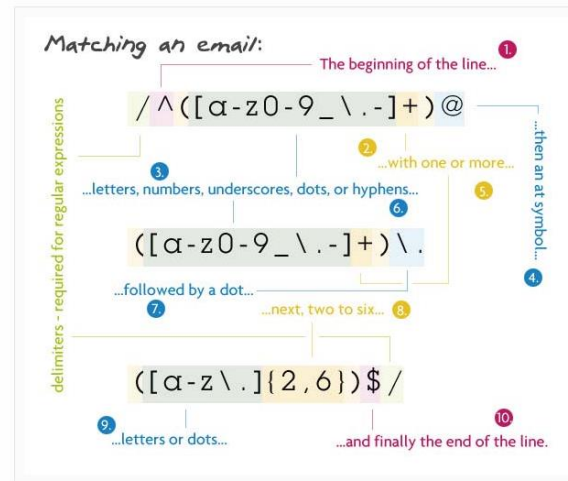
FYI
only

Regular Expressions

Regular Expressions

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

5. Matching an Email



<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

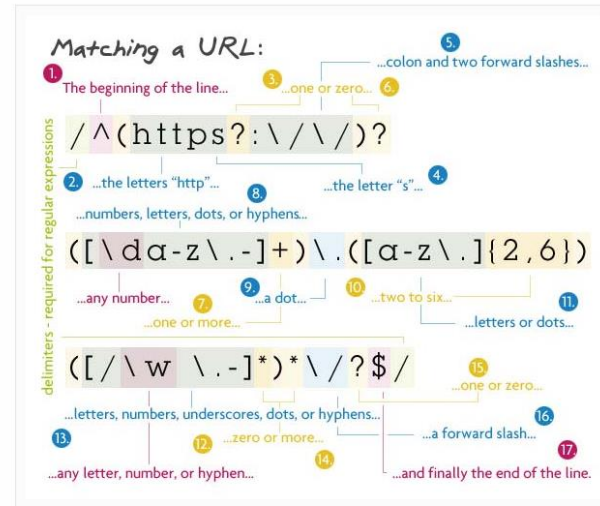
FYI
only

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 /^ 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

Task: Run a spell check on the magna_cart 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  
Daubeney  
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
```

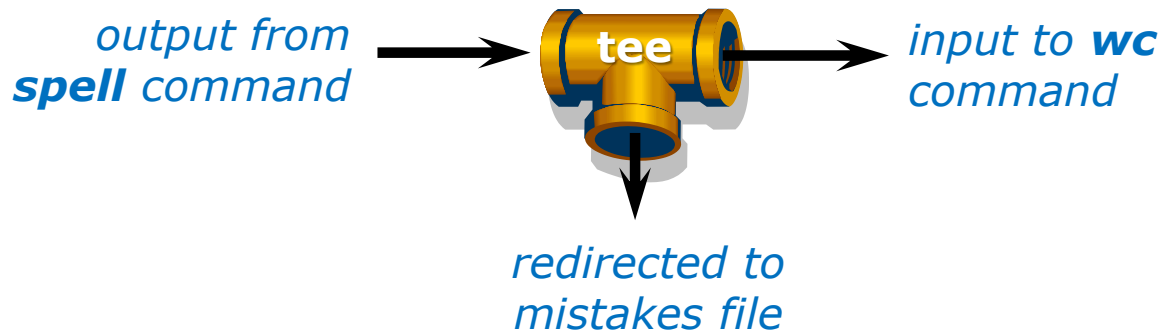
```
Eye halve a spelling chequer
It came with my pea sea
It plainly marques four my revue
Miss steaks eye kin knot sea.
Eye strike a key and type a word
And weight four it two say
Weather eye am wrong oar write
```

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
```

1st
field

2nd
field

3rd
field

4th
field

5th
field

6th
field

7th
field

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

Cut the 7th 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 5th 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.

FYI
only

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*

set sticky bit

```
/home/cis90/simben $ chmod 1777 tempdir
```

```
/home/cis90/simben $ ls -ld tempdir/
```

```
drwxrwxrwt. 2 simben90 cis90 4096 Oct 16 15:25 tempdir/
```

sticky bit set

*green background
with black text*

FYI
only

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.

FYI
only

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*

FYI
only

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

FYI
only

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.

FYI
only

Extended File Attributes

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*

FYI
only

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]#
```

test01.html


test02.html

type = httpd_sys_content_t

type = home_root_t

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 then to perform a variety of tasks on the system.

Preparation

- Be sure to make the changes to your home directory asked for in Lab 5. This lab assumes the new names and directory structures.
- Slides & slides: <http://simms-teach.com/cis90/calendar.php>
- Check the forum for help on this lab: <http://oslab.cis.cabrillo.edu/forum/>
- For additional assistance come to the CIS Lab: <http://webhawks.org/~cislab/>

Procedure

Log on to Open so that you have a command line shell at your service. Be sure you are in your home directory in start this lab. We are going to experiment with find 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 in a file.

The find command

The syntax of the find command is:

```
find starting-directory name filename options 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 home 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

*If you get stuck
please ask questions
on the forum or ask
the Lab Assistants in
the CIS Lab.*

A full-page background image showing a sunset over a beach. The sky is filled with vibrant orange, pink, and purple clouds. The sun is low on the horizon, casting a warm glow. To the right, a dark, silhouetted cliff rises from the beach. The foreground shows the wet sand of the beach reflecting the colors of the sky, with some dark rocks scattered about.

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

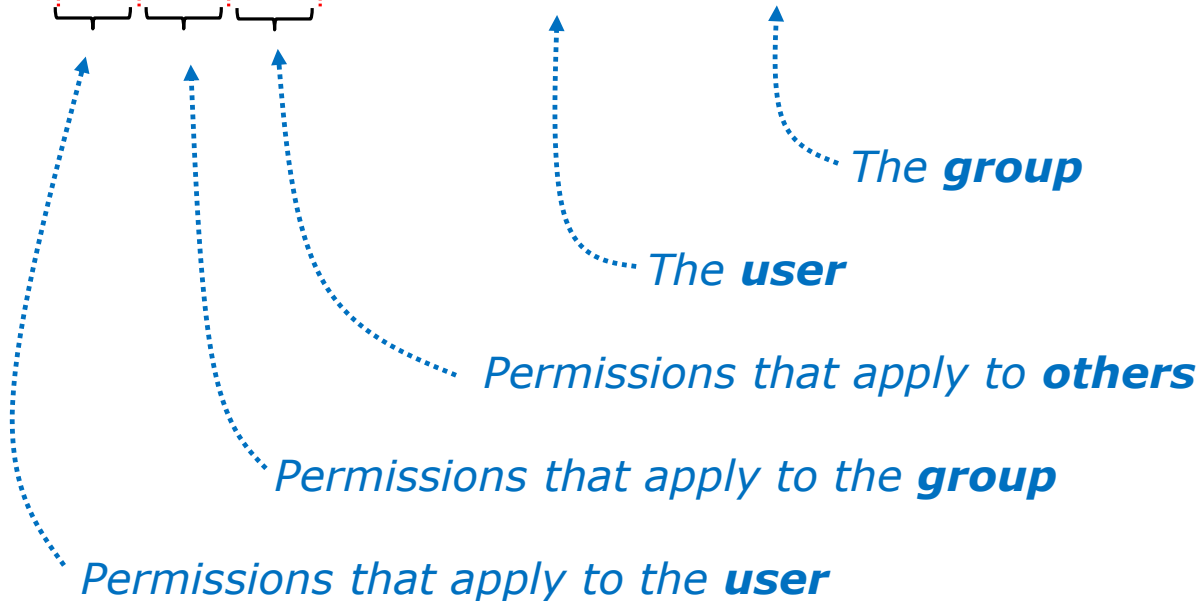
r (read) is the 4's column
 w (write) is the 2's column
 x (execute) is the 1's column

File Permissions

An example long listing

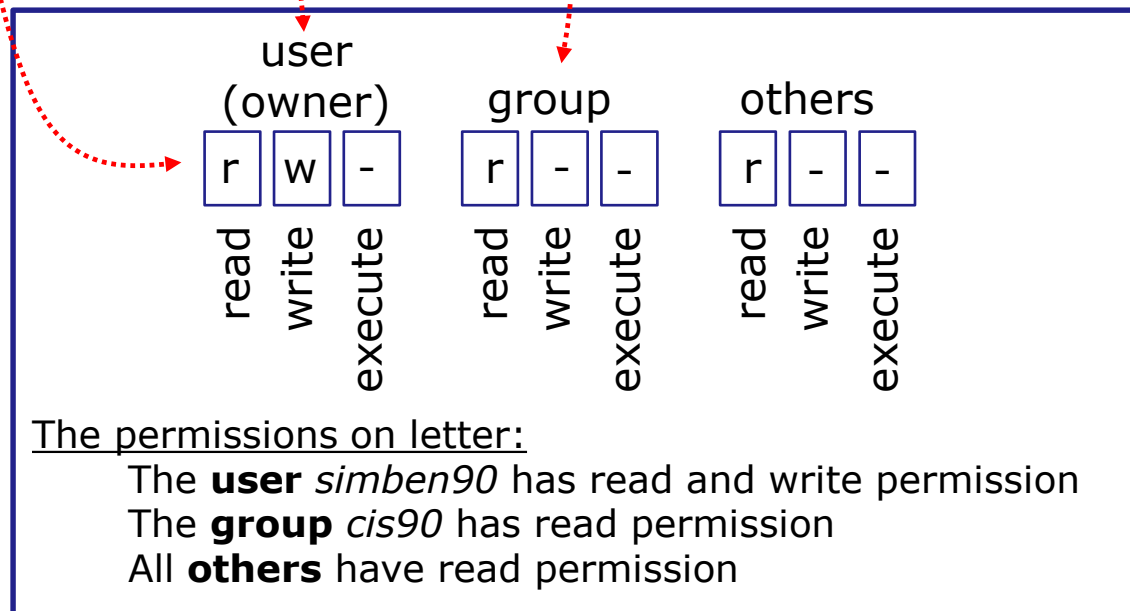
r=read
w=write
x=execute
-=none

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



File Permissions

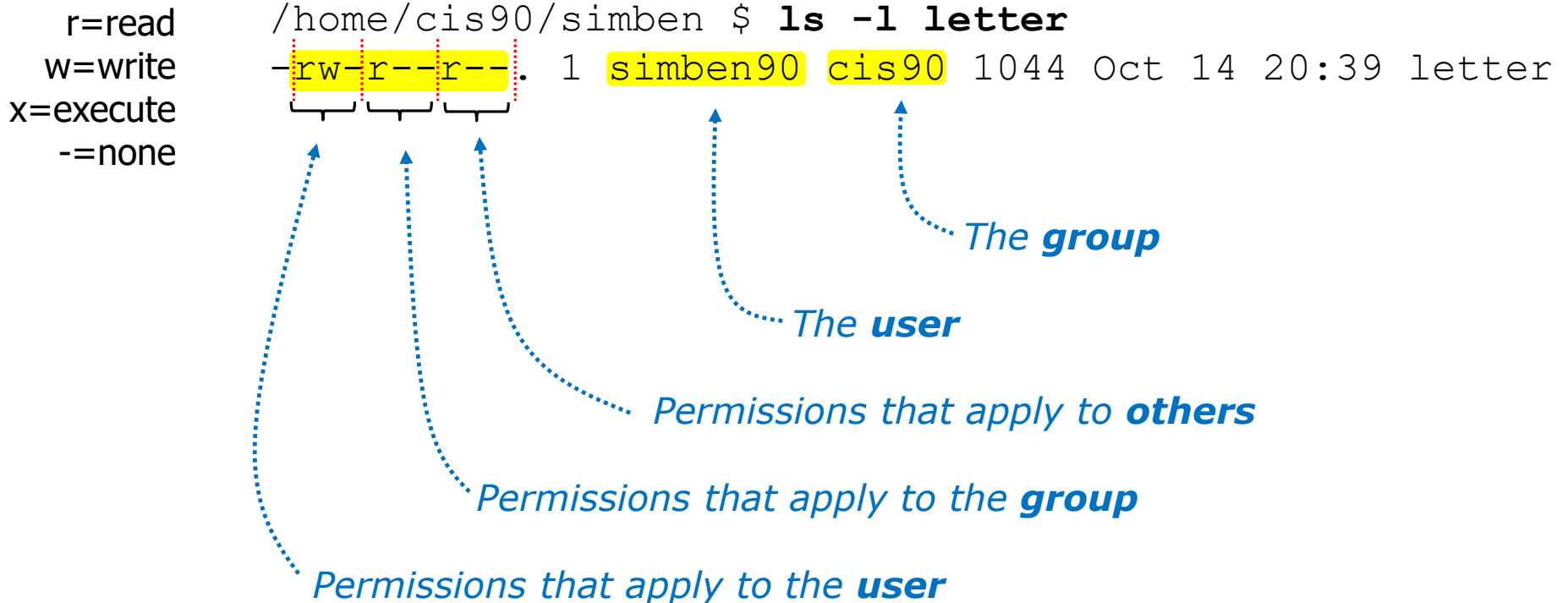
```
/home/cis90/simben $ ls -l letter
-rw-r--r--. 1 simben90 cis90 1044 Oct 14 20:39 letter
```



Use long listings to show permissions

File Permissions

Use long listings to show permissions



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
  
```

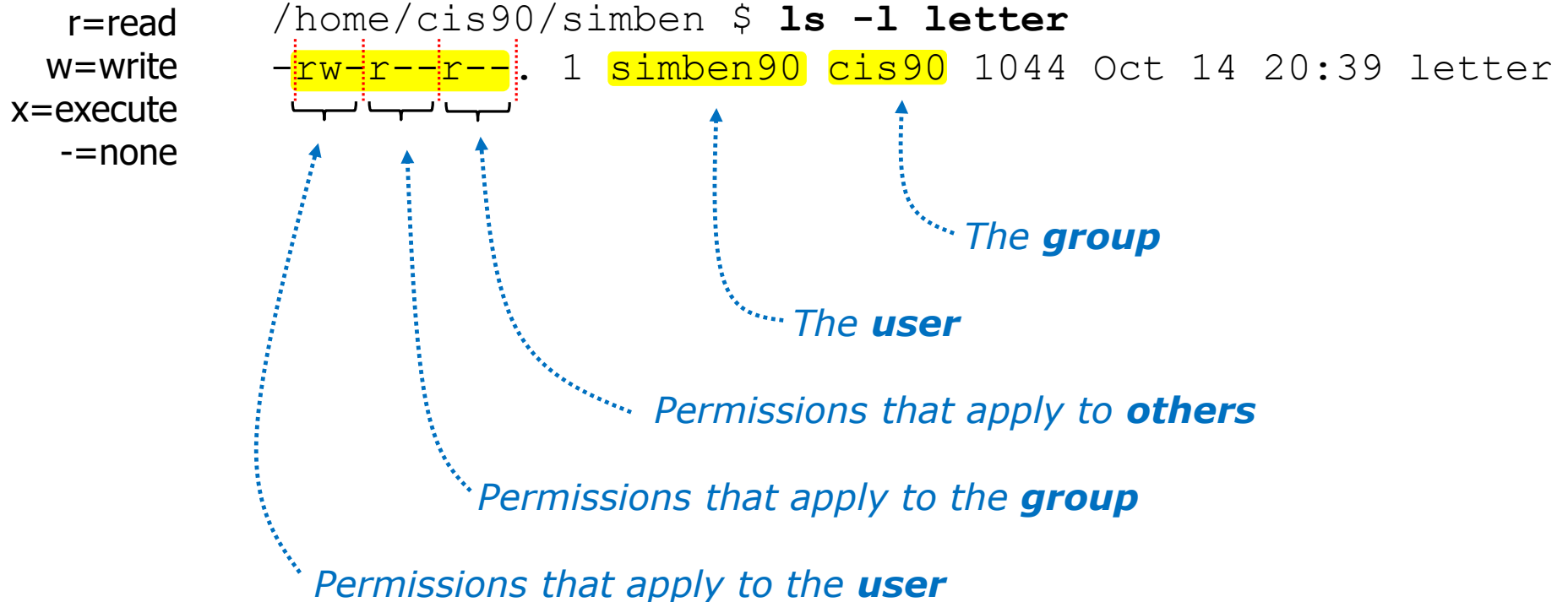
*Permissions that apply to the **user***
*Permissions that apply to the **group***
*Permissions that apply to **others***
*The **user***
*The **group***

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

No

File Permissions

Use long listings to show permissions



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
  
```

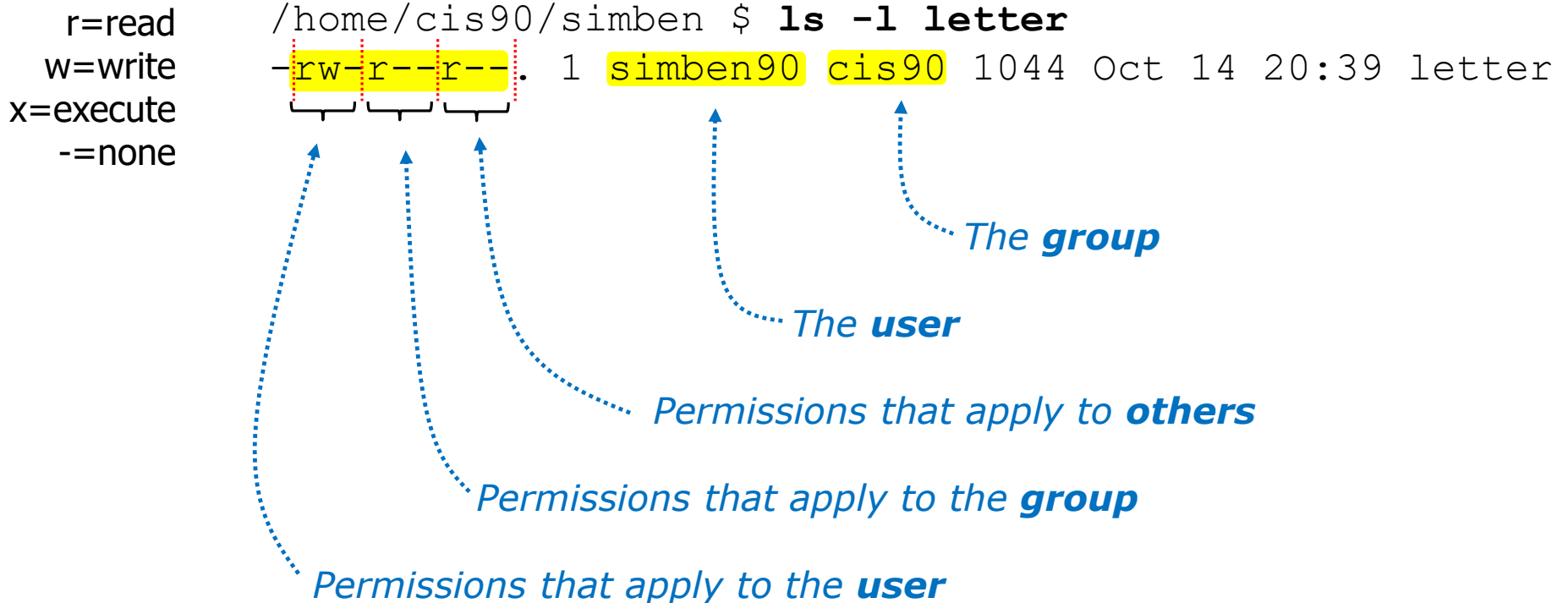
*Permissions that apply to the **user***
*Permissions that apply to the **group***
*Permissions that apply to **others***
*The **user***
*The **group***

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

No

File Permissions

Use long listings to show permissions



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
  
```

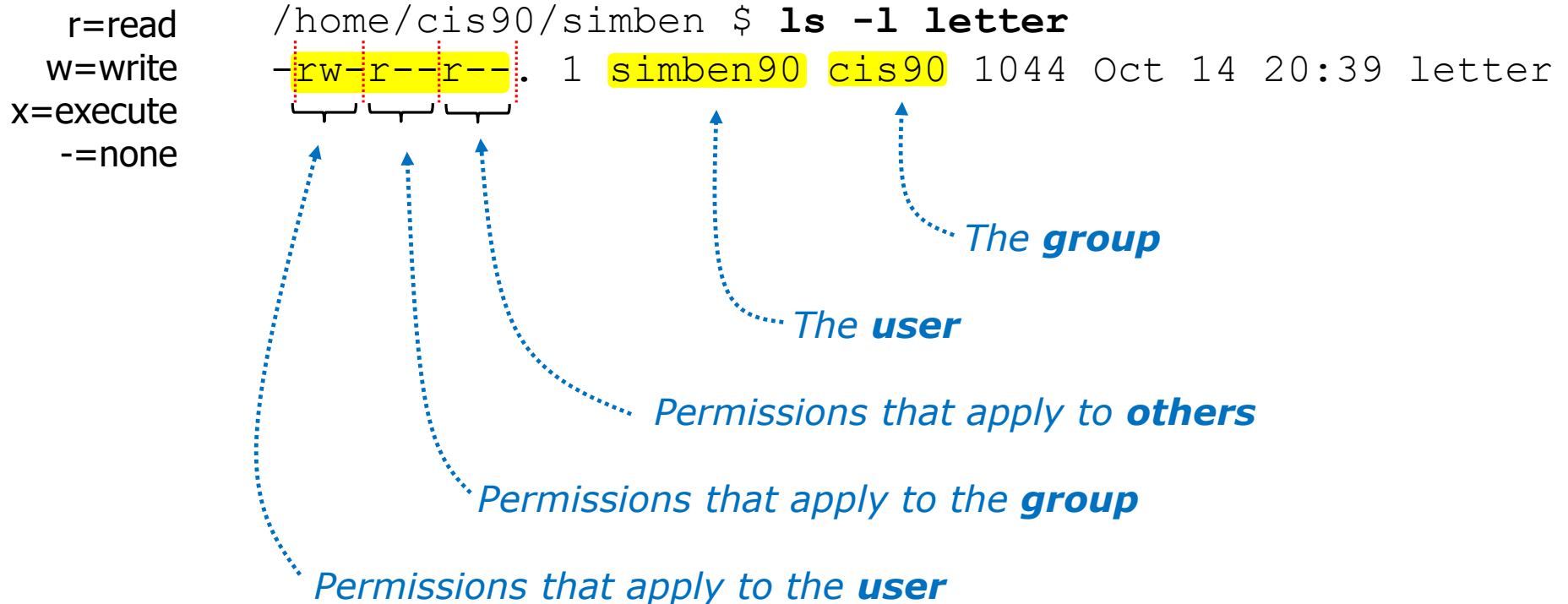
*Permissions that apply to the **user***
*Permissions that apply to the **group***
*Permissions that apply to **others***
*The **user***
*The **group***

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

Yes

File Permissions

Use long listings to show permissions

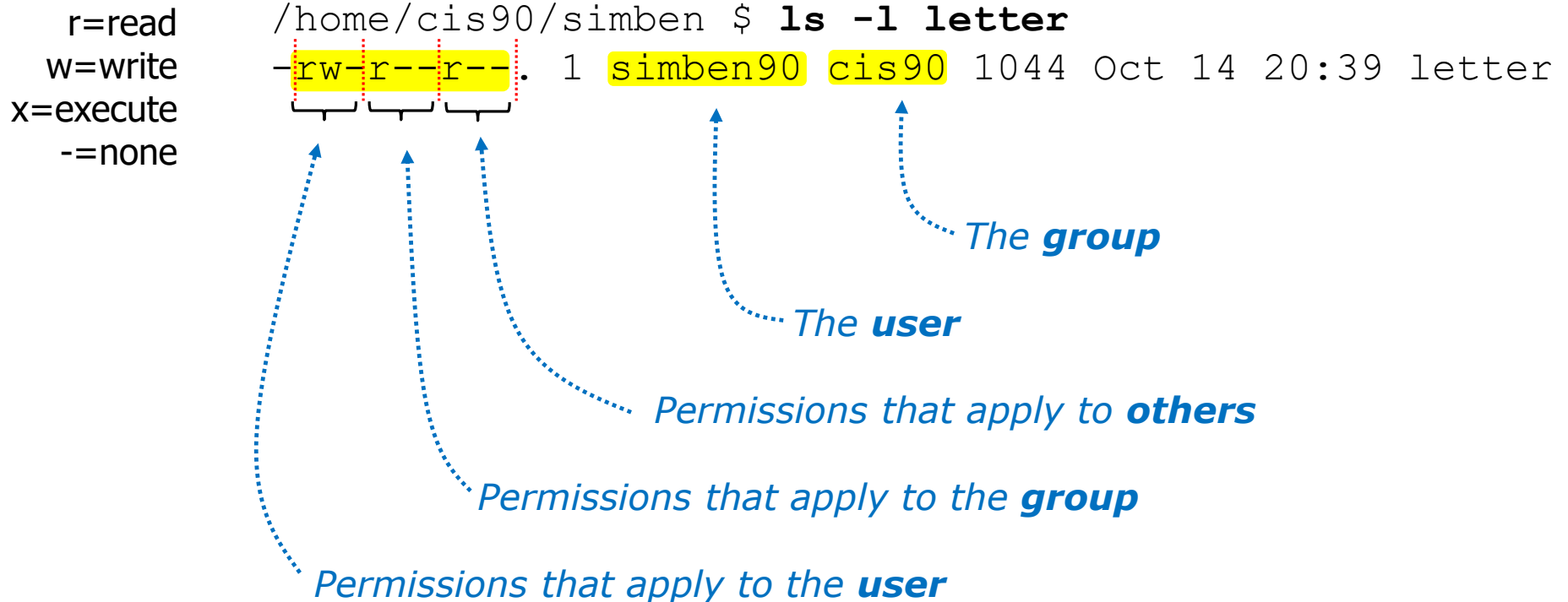


Does the smimat172 user have read permission on the letter file?

Type answer in chat window

File Permissions

Use long listings to show permissions



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