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

- Lab tested –

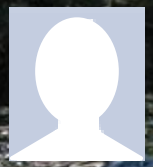
- CCC Confer wall paper / quiz emailed -

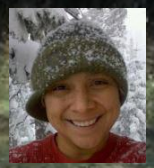
- Pick up Polycom phone/extension mics -
- Check that headset is charged –
- Wireless lapel mic backup battery -
- Backup slides, CCC info, handouts on flash drive -

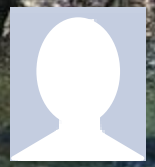
CIS 90 - Lesson 8

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

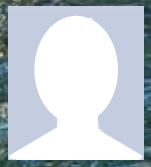

Dieskau


Jonathan


Ana


David


Obie


Dave


Cole


Corey


Nancy


Ryan

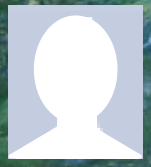

Elia


Tasha


Darren

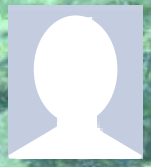

Scott


Devin


Everett


Juan


Raven


Rogan


Mike


Mook


Melissa


Cameron


Jose


Jeff


Matt


Kenneth


Ousmane


Ian


Solomon


Henry


Matthew


Mason


Chan

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

First Minute Quiz

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

email answers to: `risimms@cabrillo.edu`
(within the first few minutes of class)

[ ] Has the phone bridge been added?

[ ] Is recording on?

[ ] Does the phone bridge have the mike?

[ ] Share slides, putty (rsimms, simmsben, roddyduk), and Chrome

[ ] Disable spelling on PowerPoint

Input/Output Processing

Objectives	Agenda
• 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: o find o grep o wc o sort o spell	• Quiz • Questions • Warmup • Housekeeping • Review • File descriptors • Pipelines • New commands • Tasks using pipelines

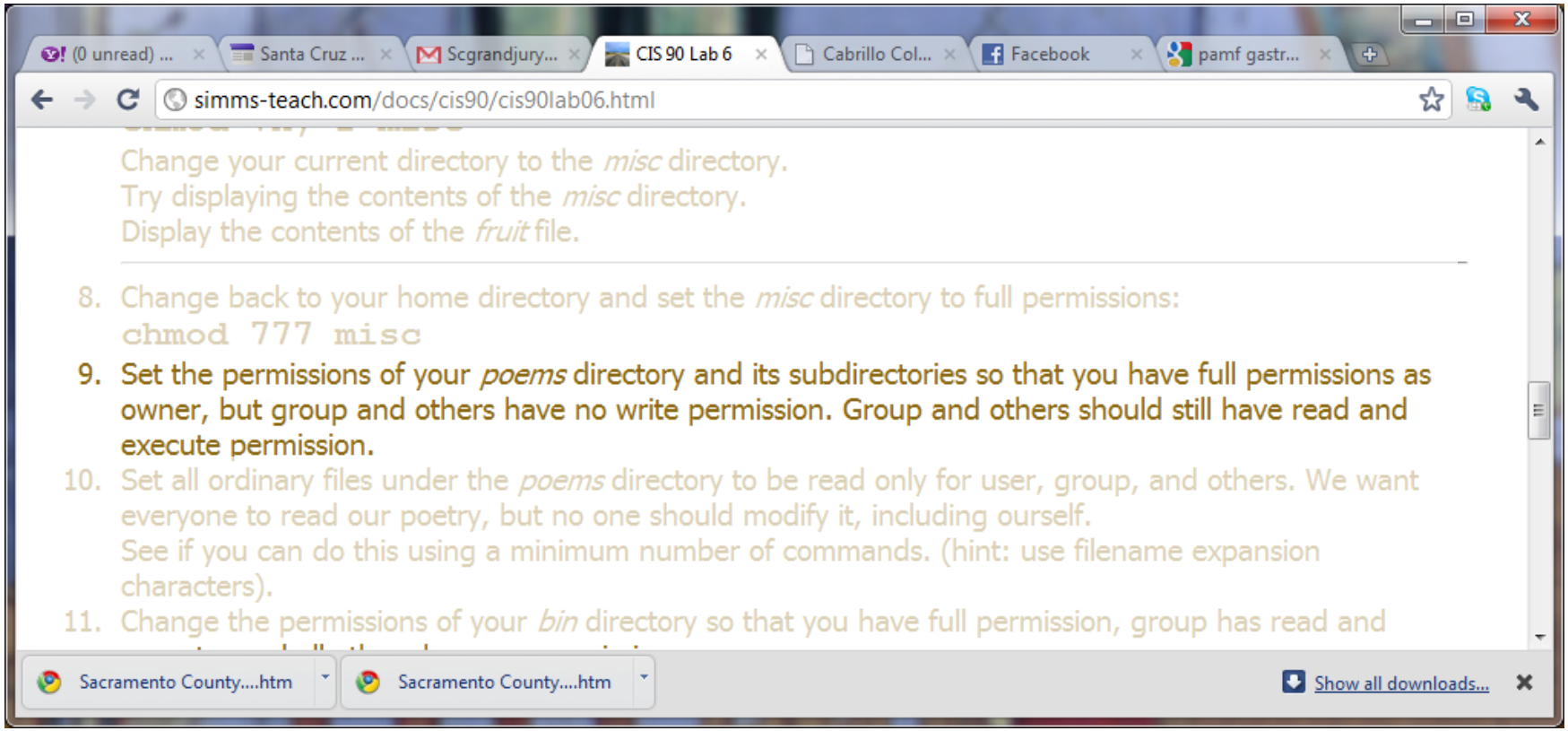
Questions

- Last lab?
- Last class?
- Last test?
- Previous lessons?

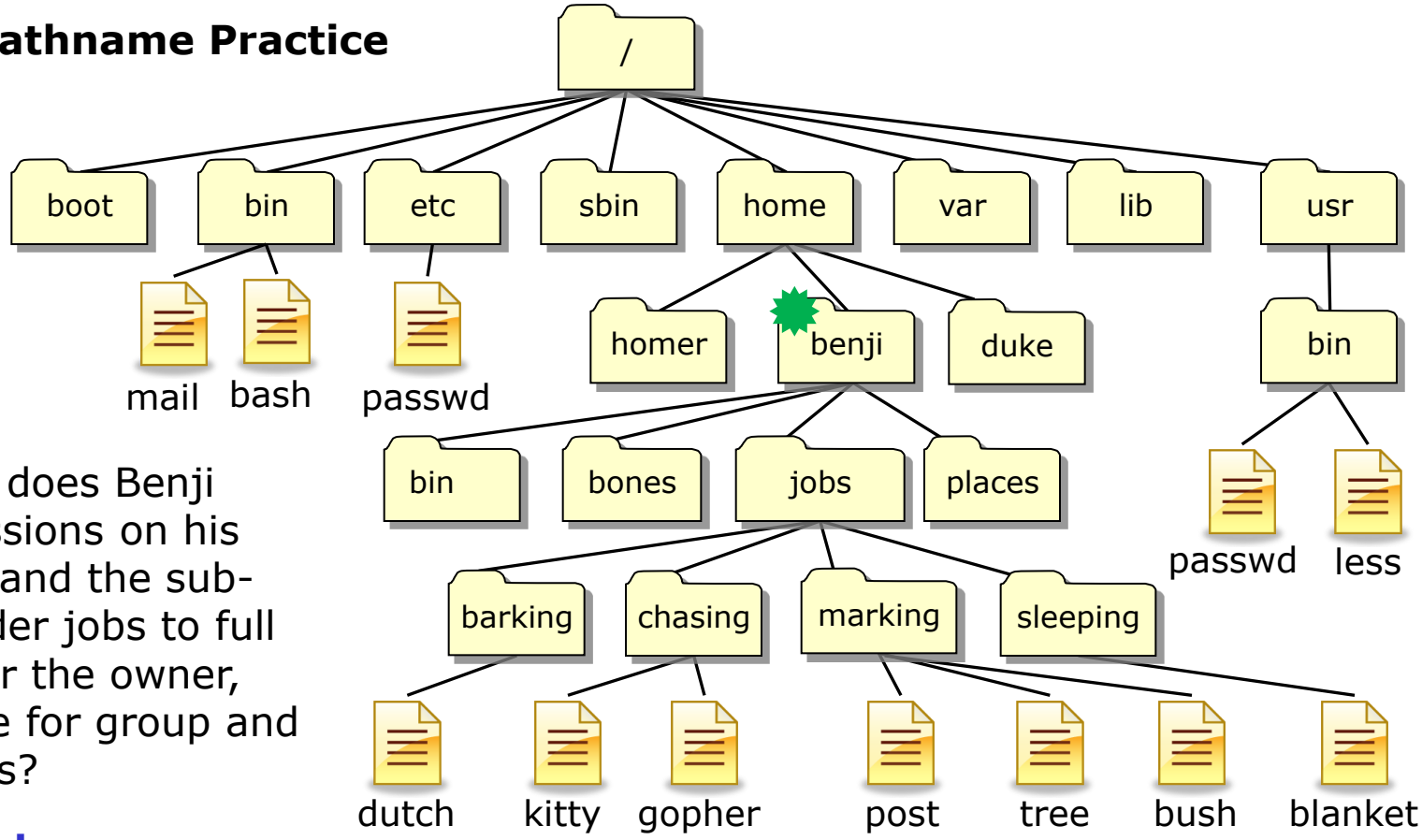
More on pathnames

(useful for Lab 6)

One of the steps in Lab 6



File Tree Pathname Practice

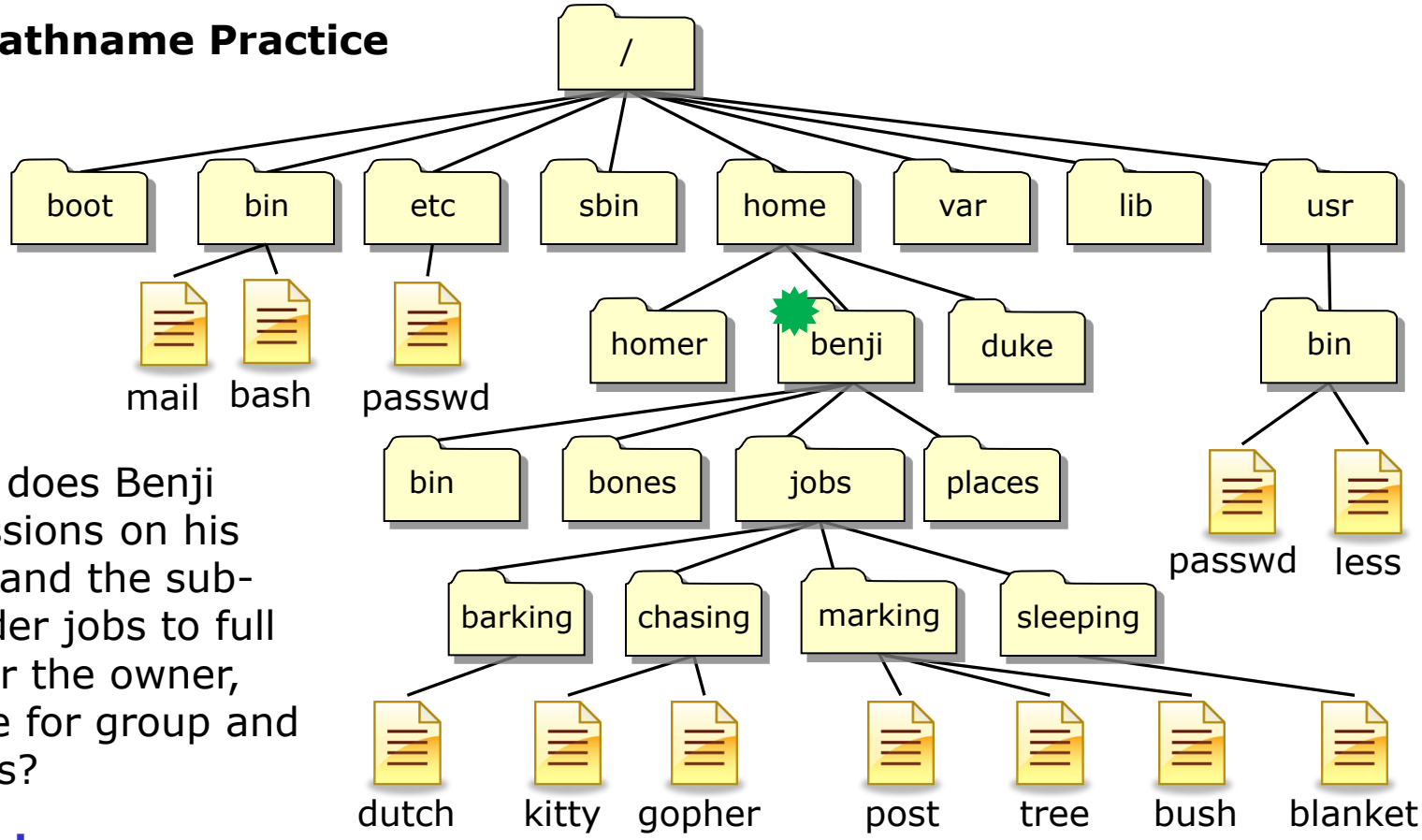


From benji how does Benji change permissions on his jobs directory and the sub-directories under jobs to full permissions for the owner, read & execute for group and none for others?

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

This works and takes 6 commands to complete

File Tree Pathname Practice

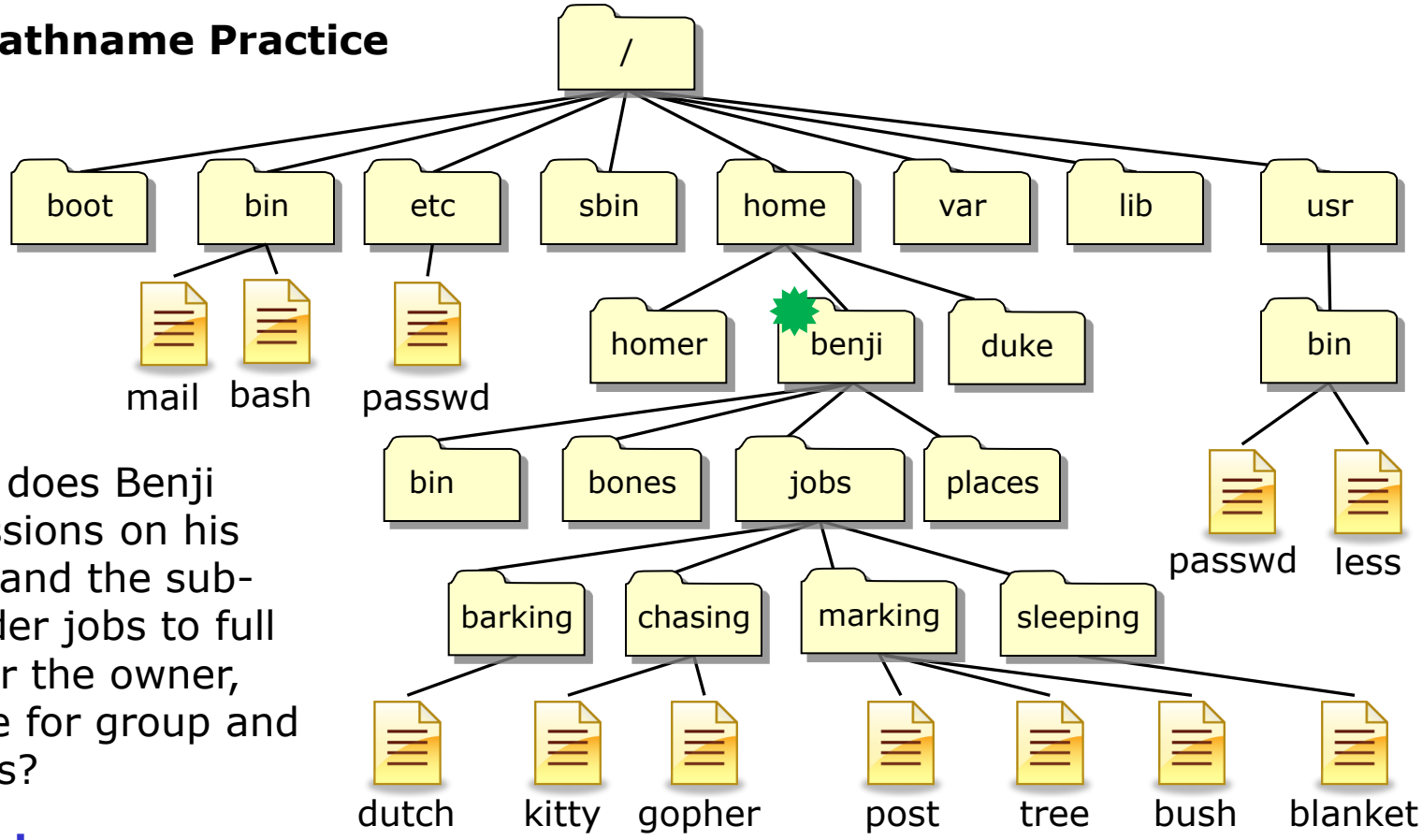


From benji how does Benji change permissions on his jobs directory and the sub-directories under jobs to full permissions for the owner, read & execute for group and none for others?

chmod 750 jobs
chmod 750 jobs/barking
chmod 750 jobs/chasing
chmod 750 jobs/marking
chmod 750 jobs/sleeping

This also works and takes 5 commands to complete

File Tree Pathname Practice

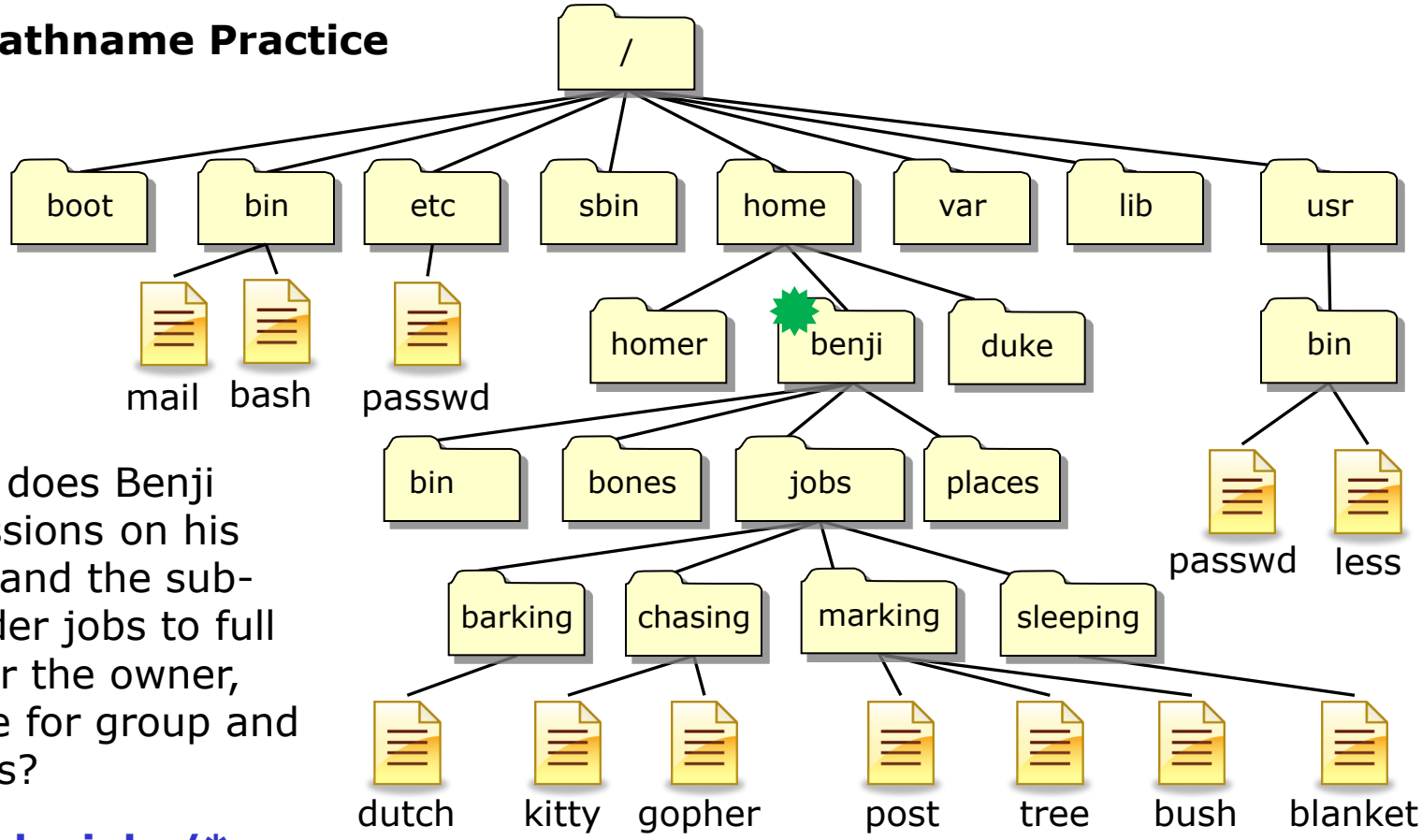


From benji how does Benji change permissions on his jobs directory and the sub-directories under jobs to full permissions for the owner, read & execute for group and none for others?

chmod 750 jobs
chmod 750 jobs/*

This also works and takes 2 commands to complete

File Tree Pathname Practice

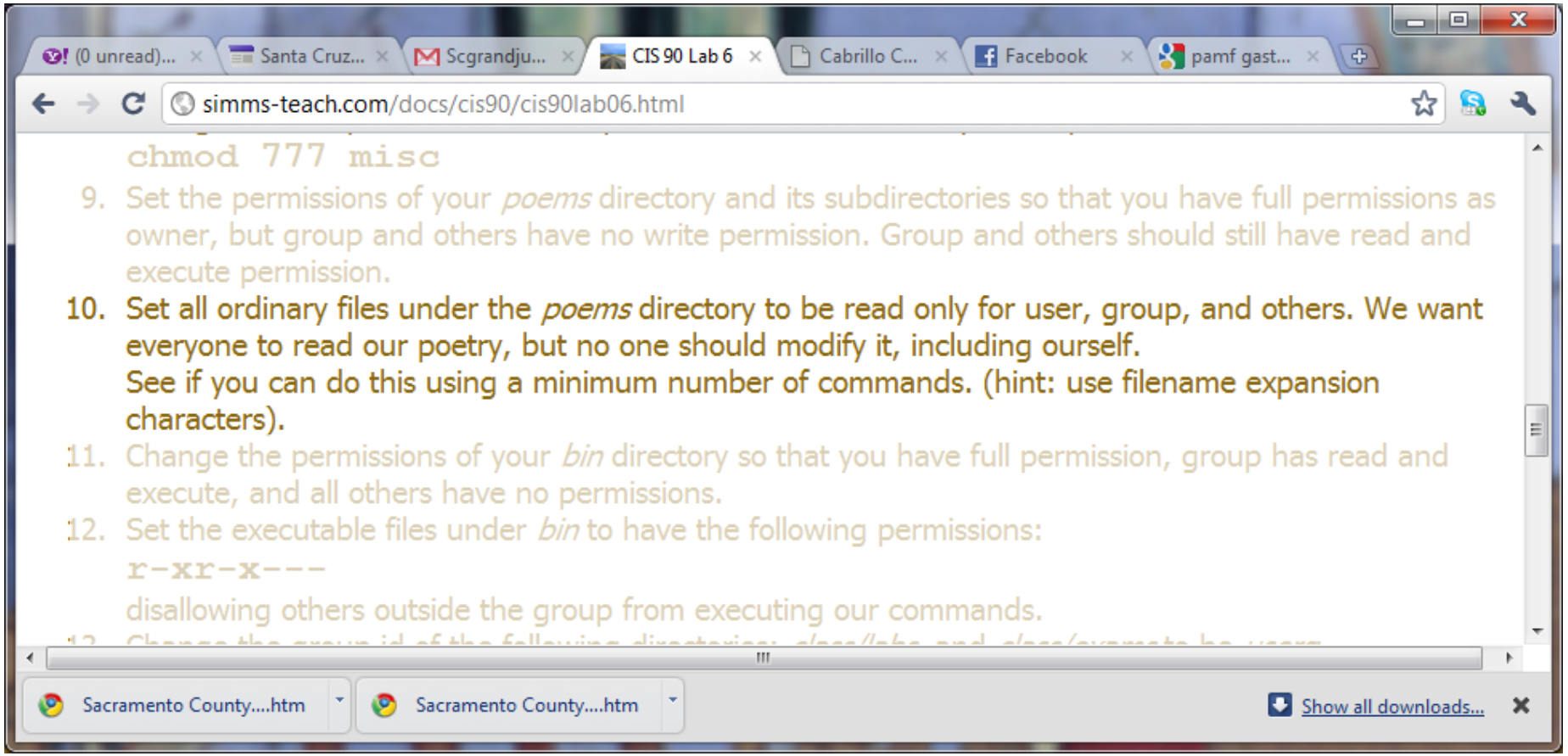


From benji how does Benji change permissions on his jobs directory and the sub-directories under jobs to full permissions for the owner, read & execute for group and none for others?

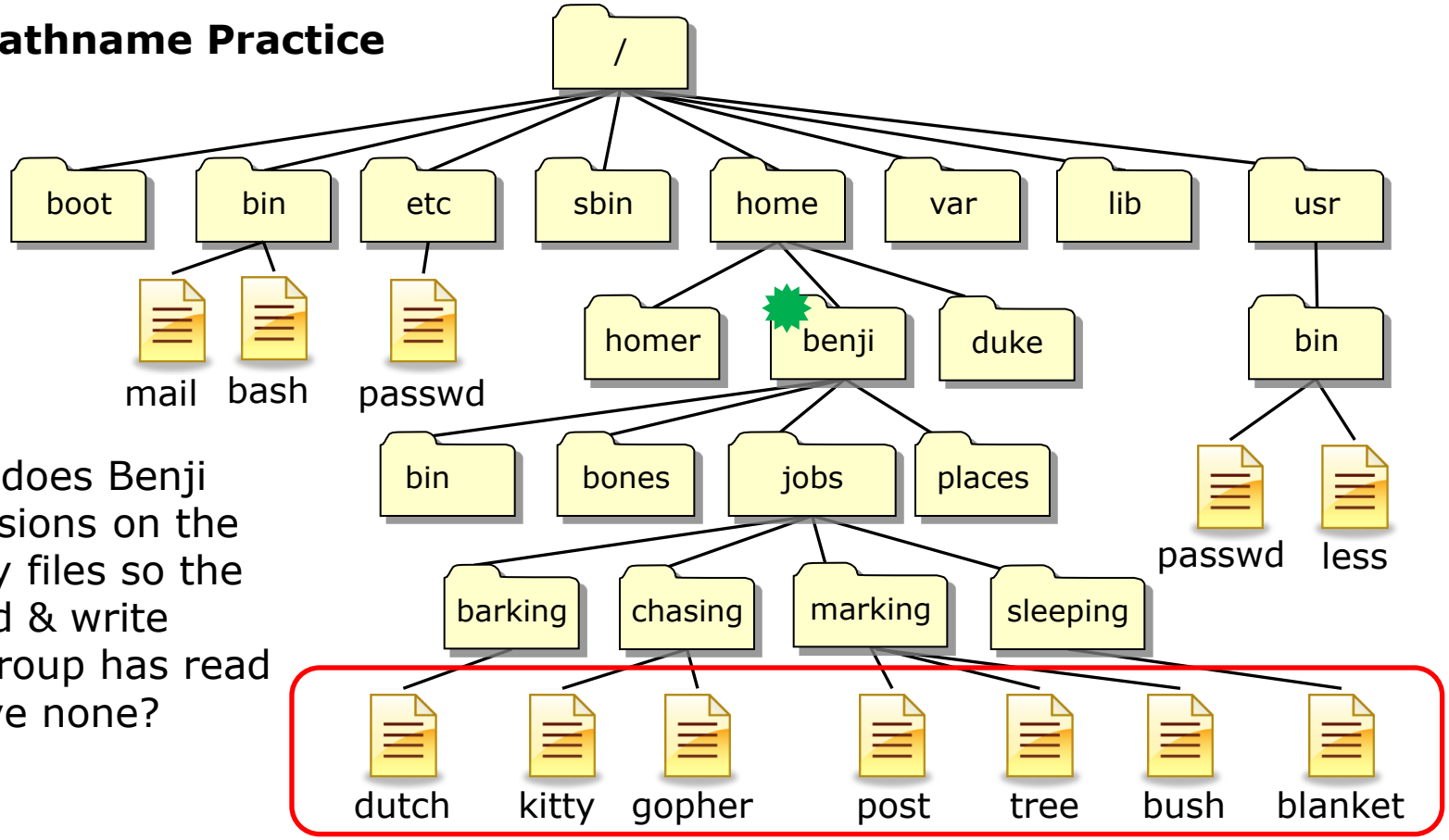
chmod 750 jobs jobs/*

This is how you can do it in a single command

Another step in Lab 6



File Tree Pathname Practice



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?

cd jobs
cd barking
chmod 640 dutch
cd ..

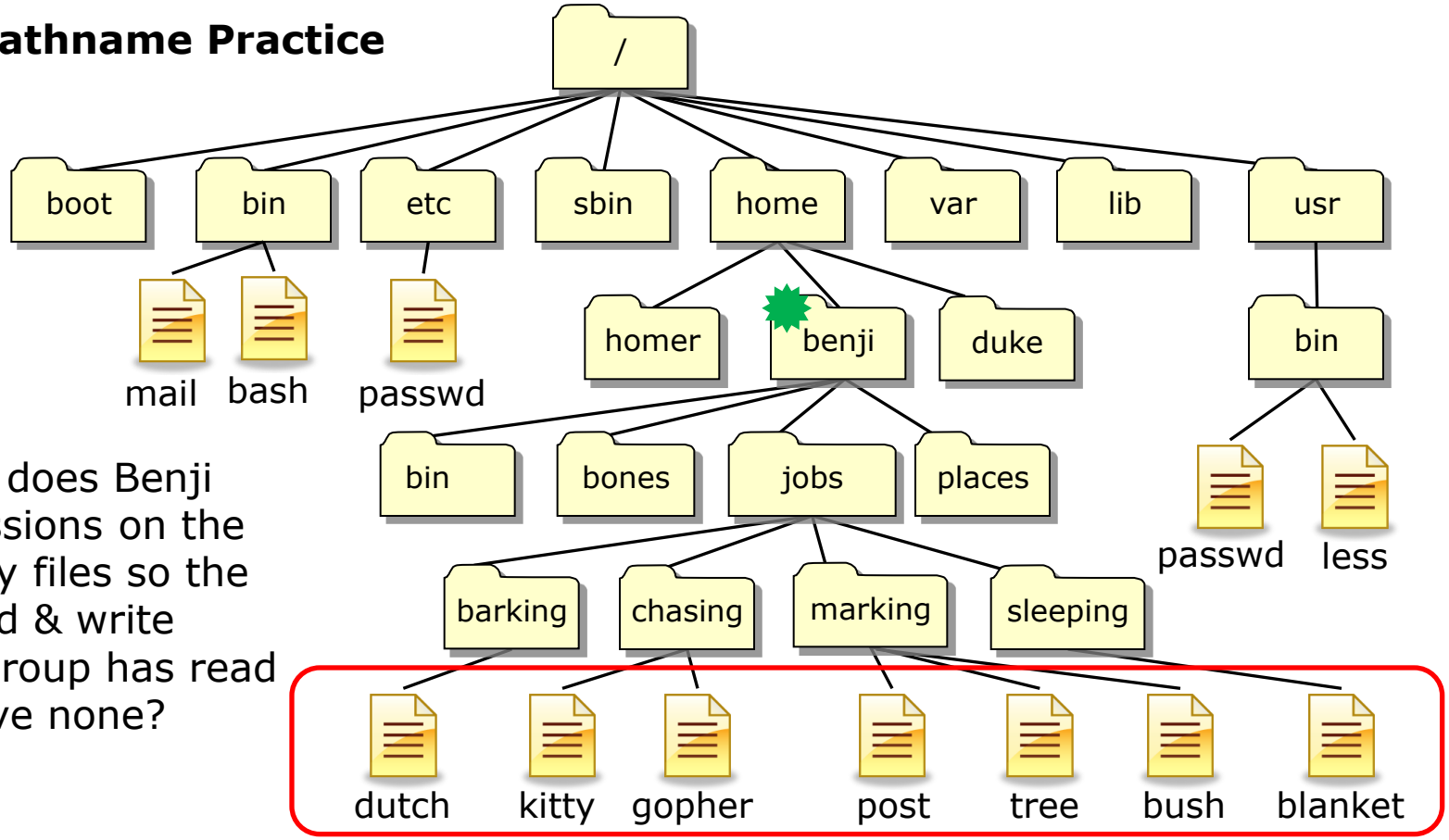
cd chasing
chmod 640 kitty
chmod 640 gopher
cd ..

cd marking
chmod 640 post
chmod 640 tree
chmod 640 bush
cd ..

cd sleeping
chmod 640 blanket
cd

Method 1:
takes 16 commands

File Tree Pathname Practice



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?

cd jobs
cd barking
chmod 640 dutch
cd ..

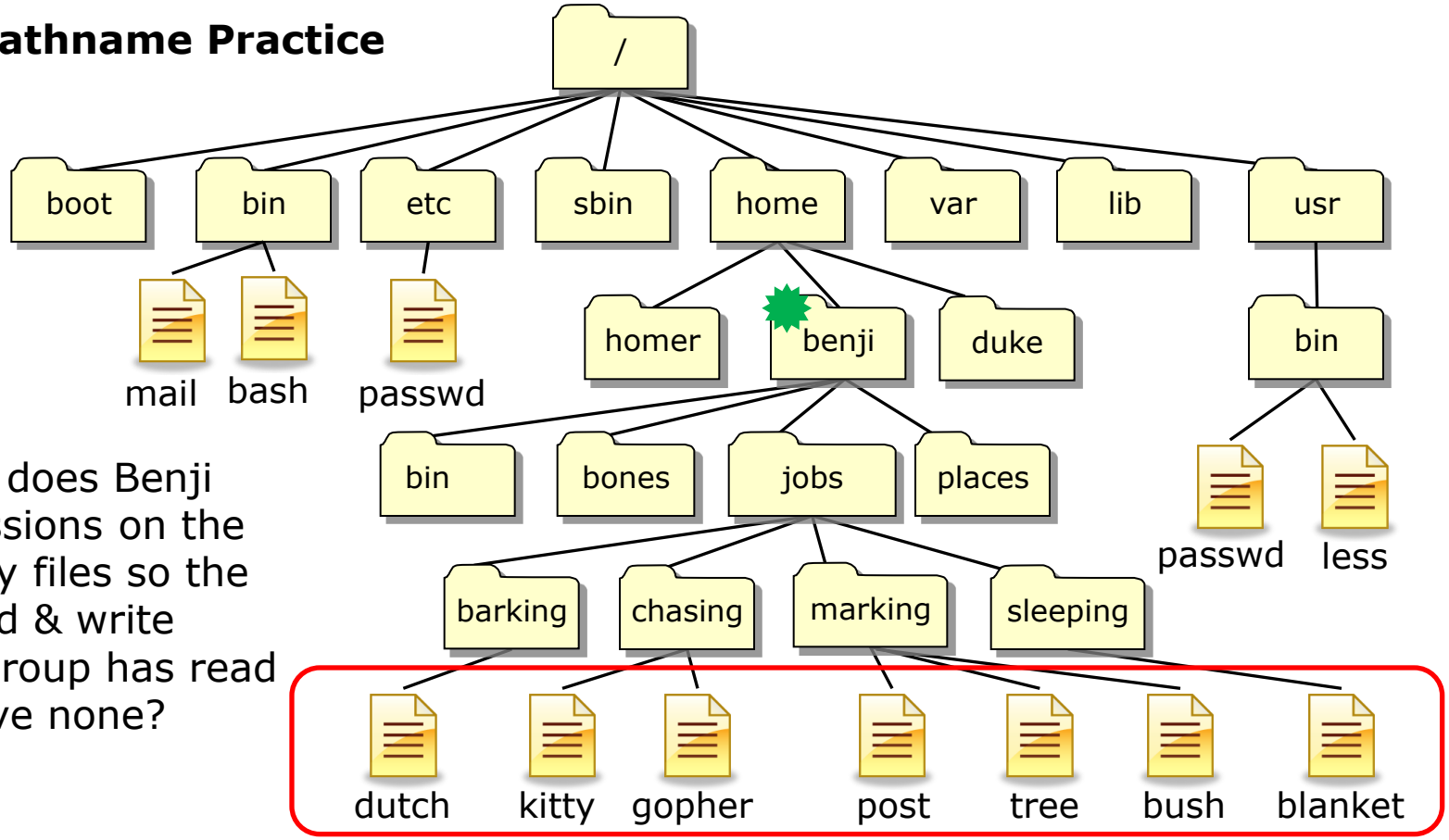
cd chasing
chmod 640 kitty gopher
cd ..

cd marking
chmod 640 post tree bush
cd ..

cd sleeping
chmod 640 blanket
cd

Method 2:
takes 13 commands

File Tree Pathname Practice



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?

cd jobs
cd barking
chmod 640 *
cd ..

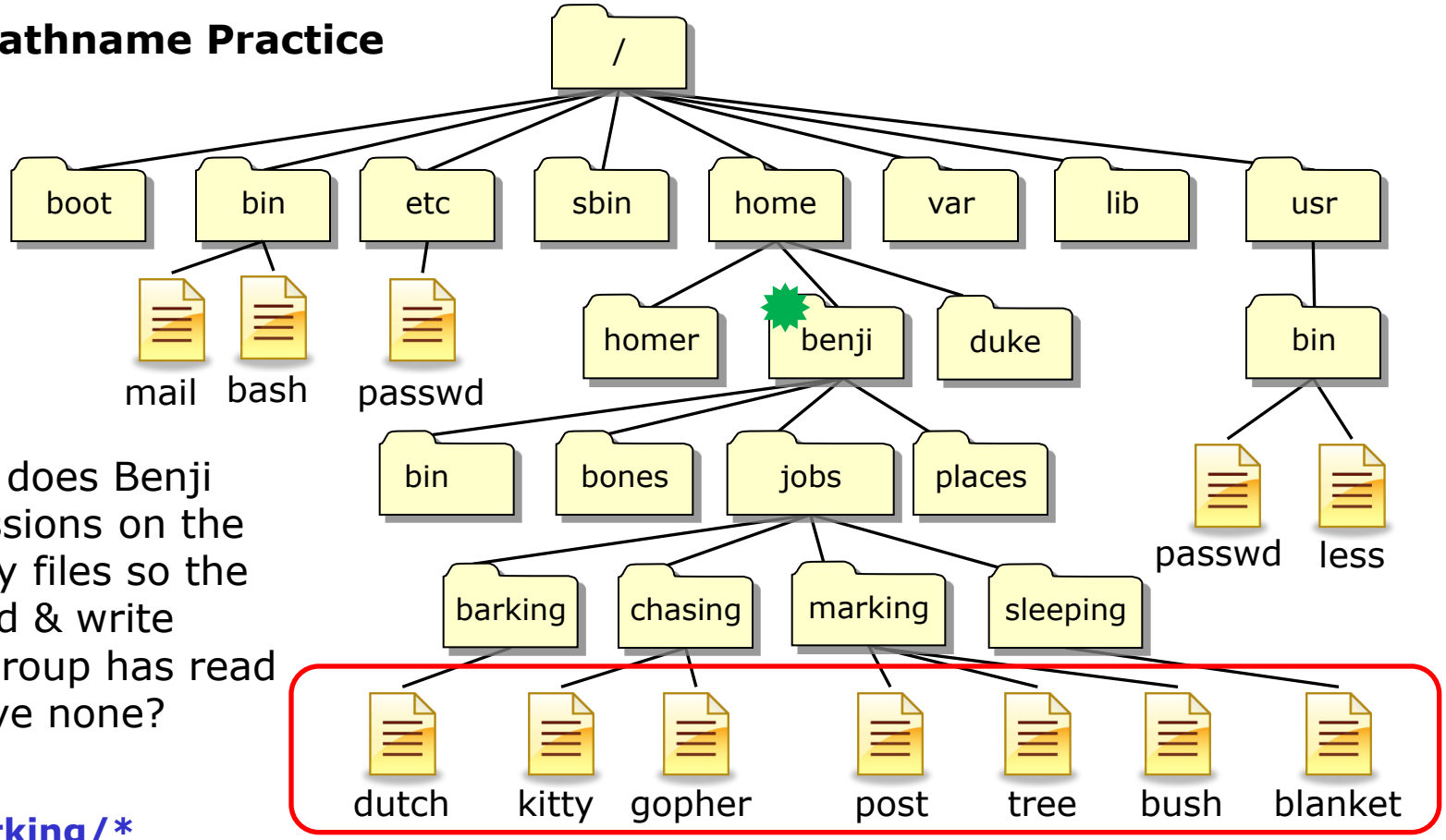
cd chasing
chmod 640 *
cd ..

cd marking
chmod 640 *
cd ..

cd sleeping
chmod 640 *
cd

Method 3:
takes 13 commands

File Tree Pathname Practice

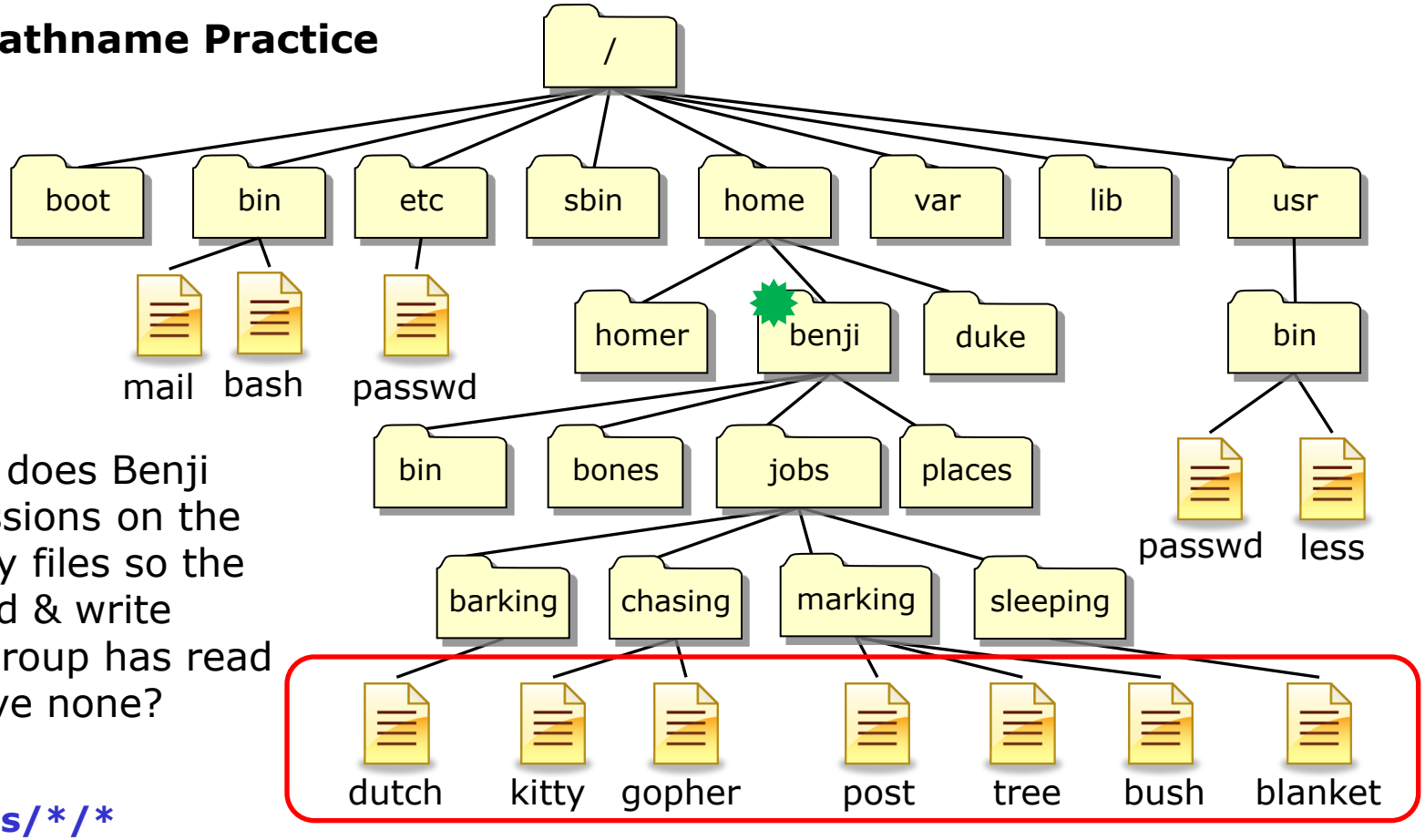


From ✹ how does Benji change permissions on the circled ordinary files so the owner has read & write permissions, group has read and others have none?

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

Method 4:
takes 6 commands

File Tree Pathname Practice



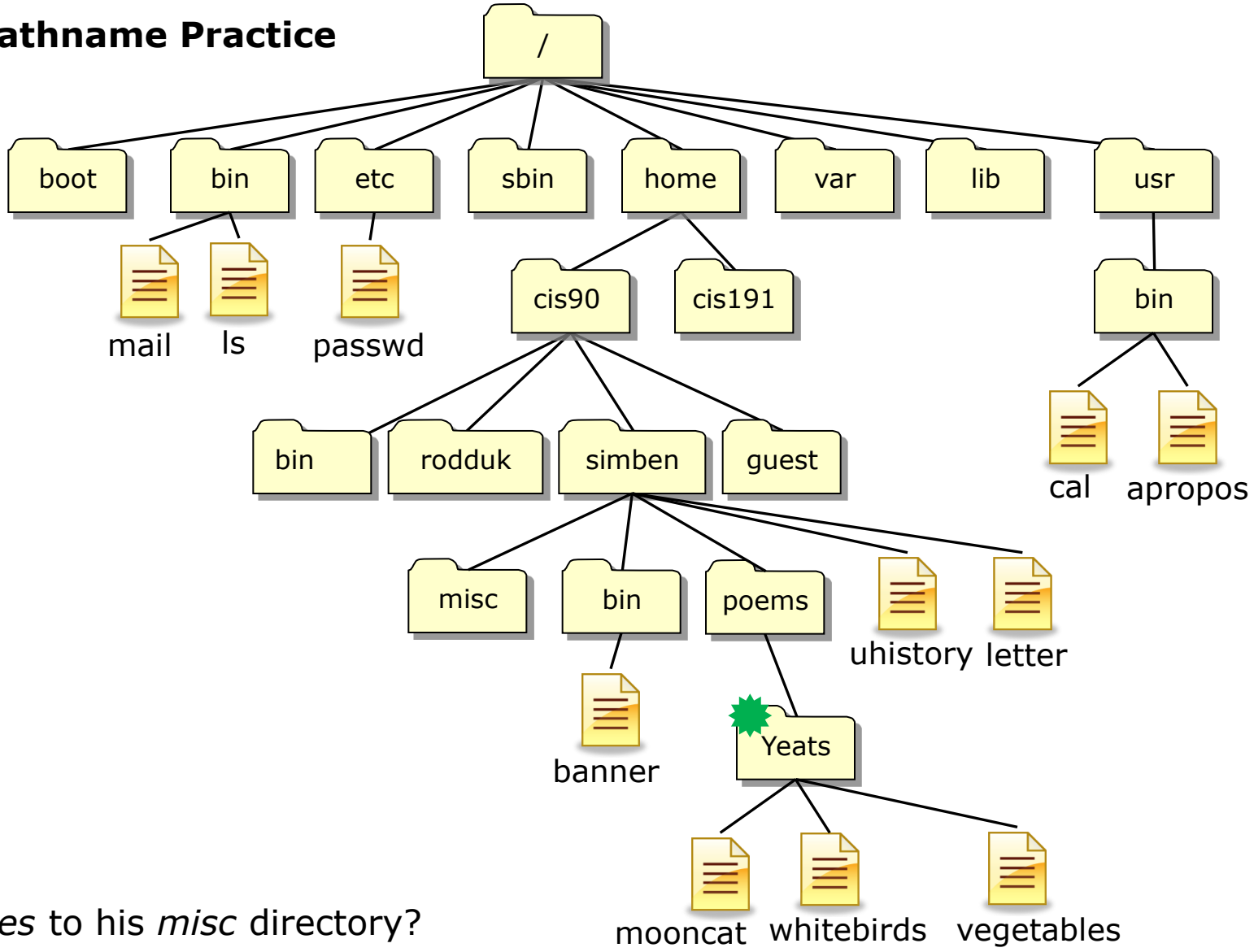
From ✹ how does Benji change permissions on the circled ordinary files so the owner has read & write permissions, group has read and others have none?

chmod 640 jobs/*/*

Method 5:
takes 1 command

Warmup

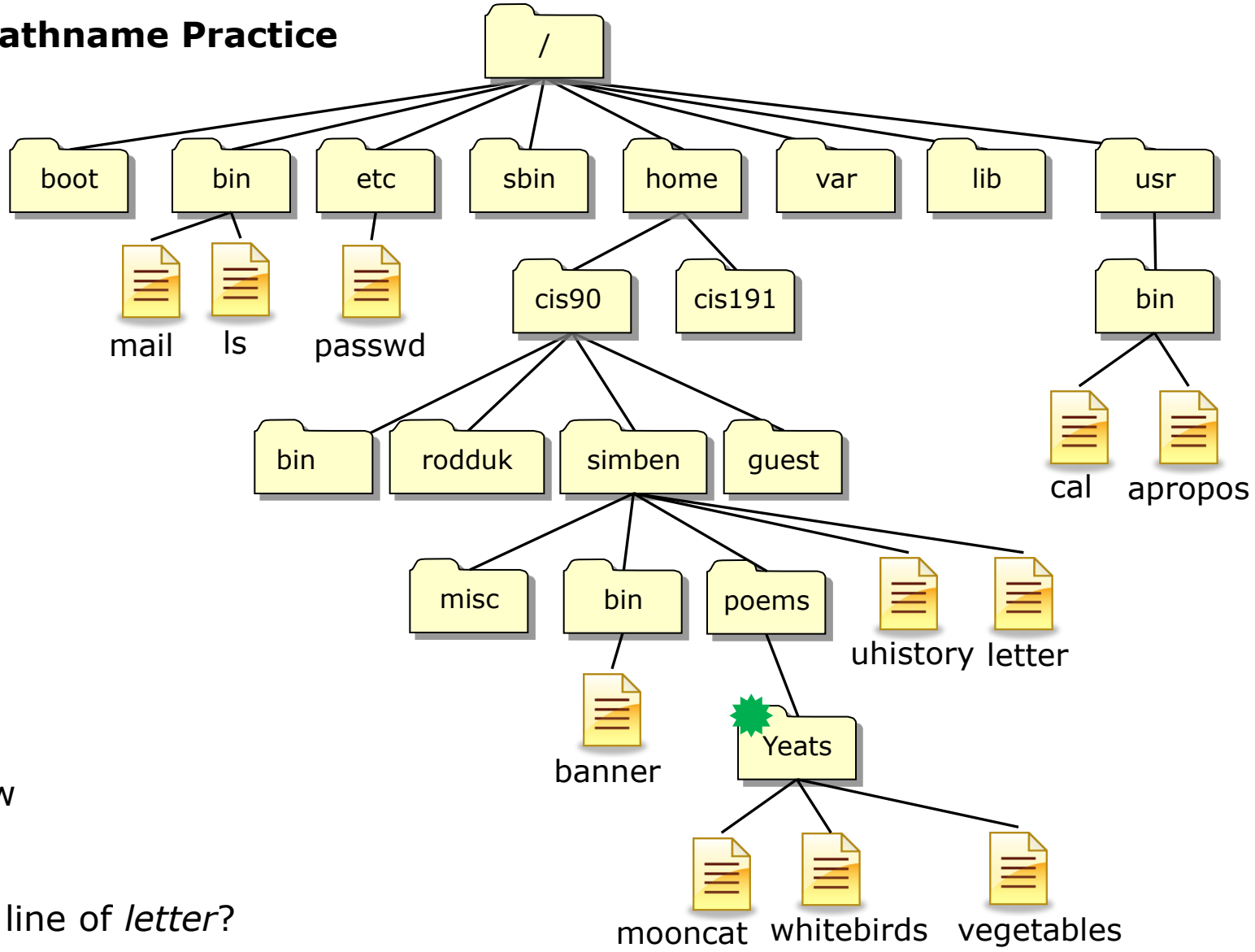
File Tree Pathname Practice



From ✹ how does Benji:

Move *vegetables* to his *misc* directory?

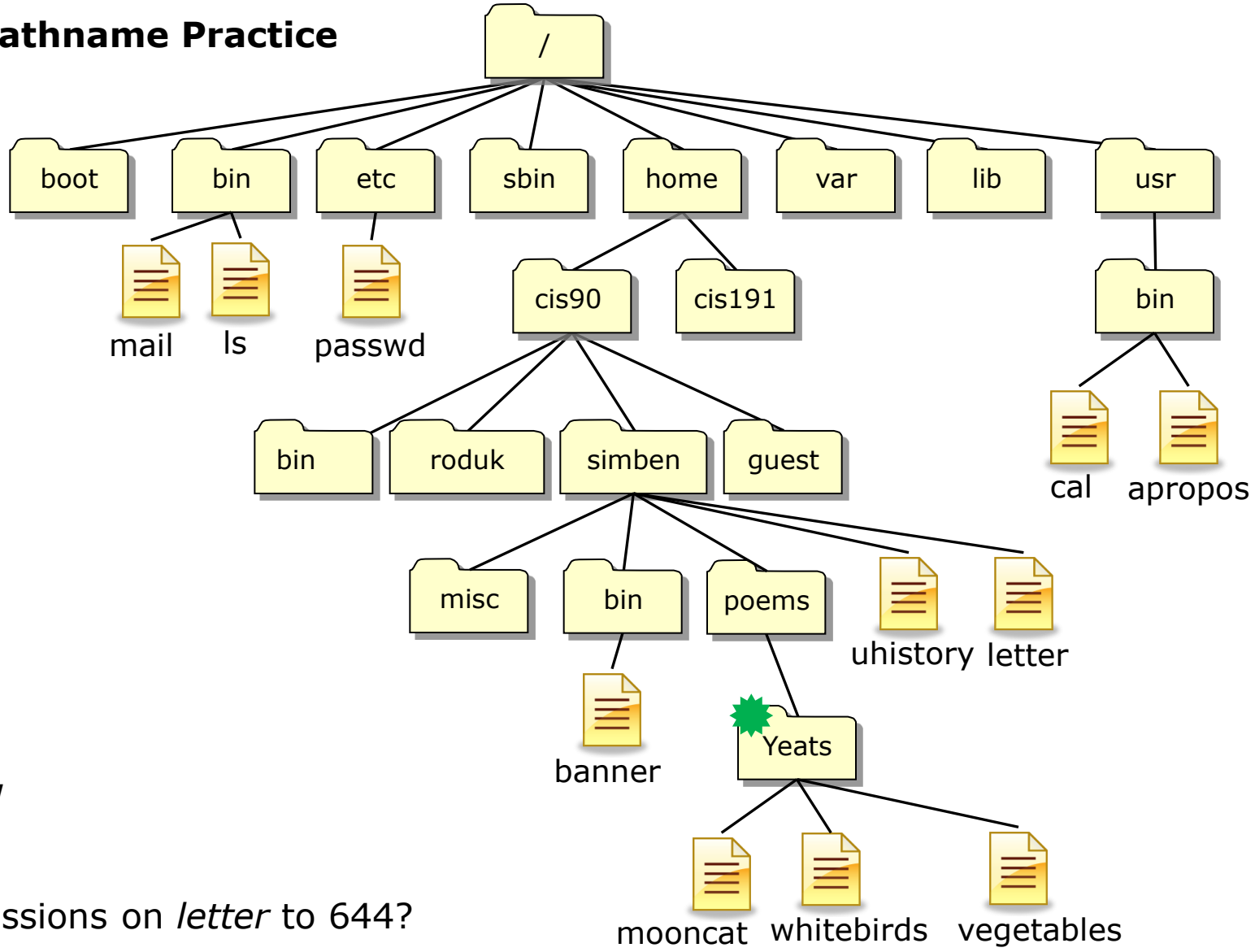
File Tree Pathname Practice



From how does Benji:

Print the last line of *letter*?

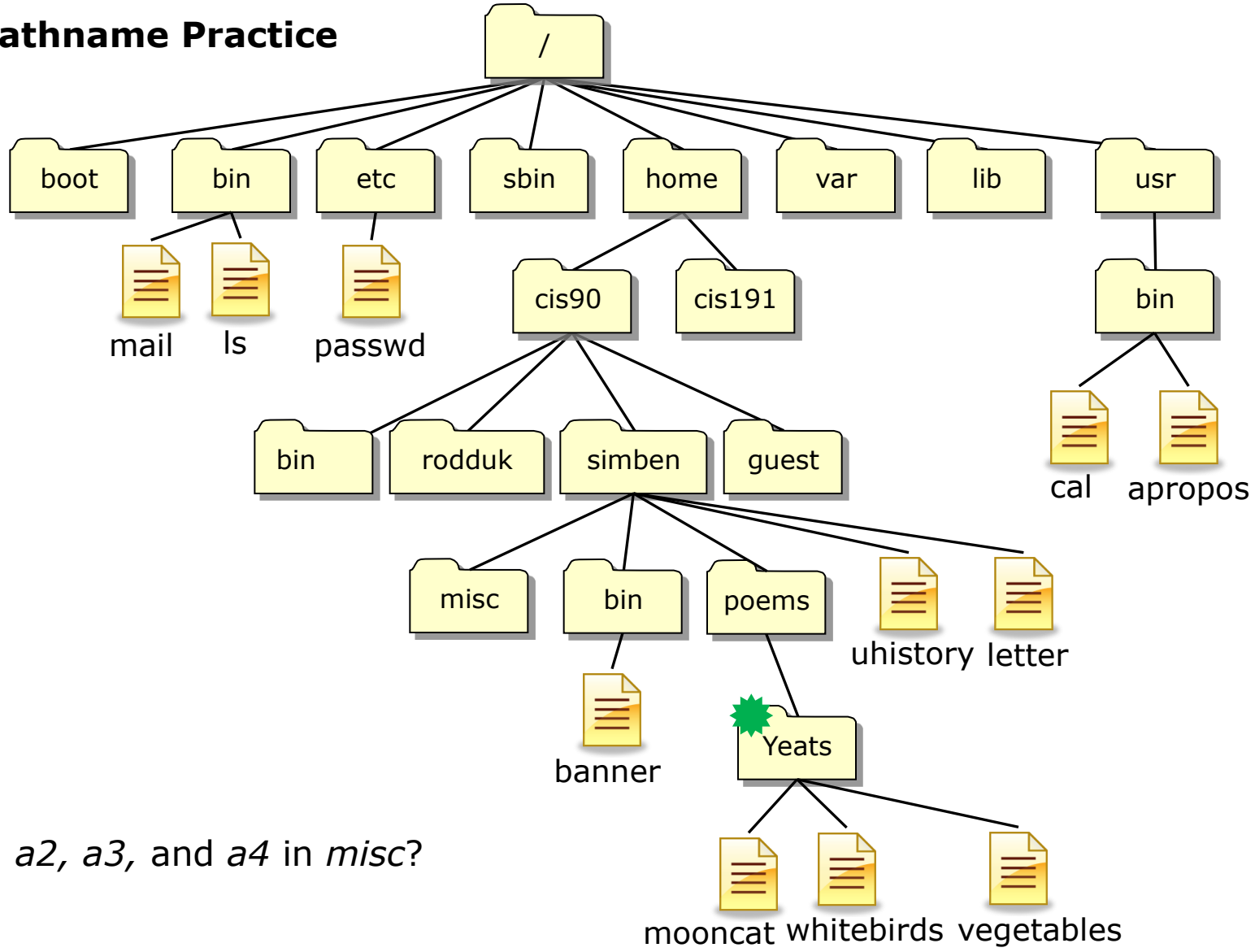
File Tree Pathname Practice



From ✹ how does Benji:

Change permissions on *letter* to 644?

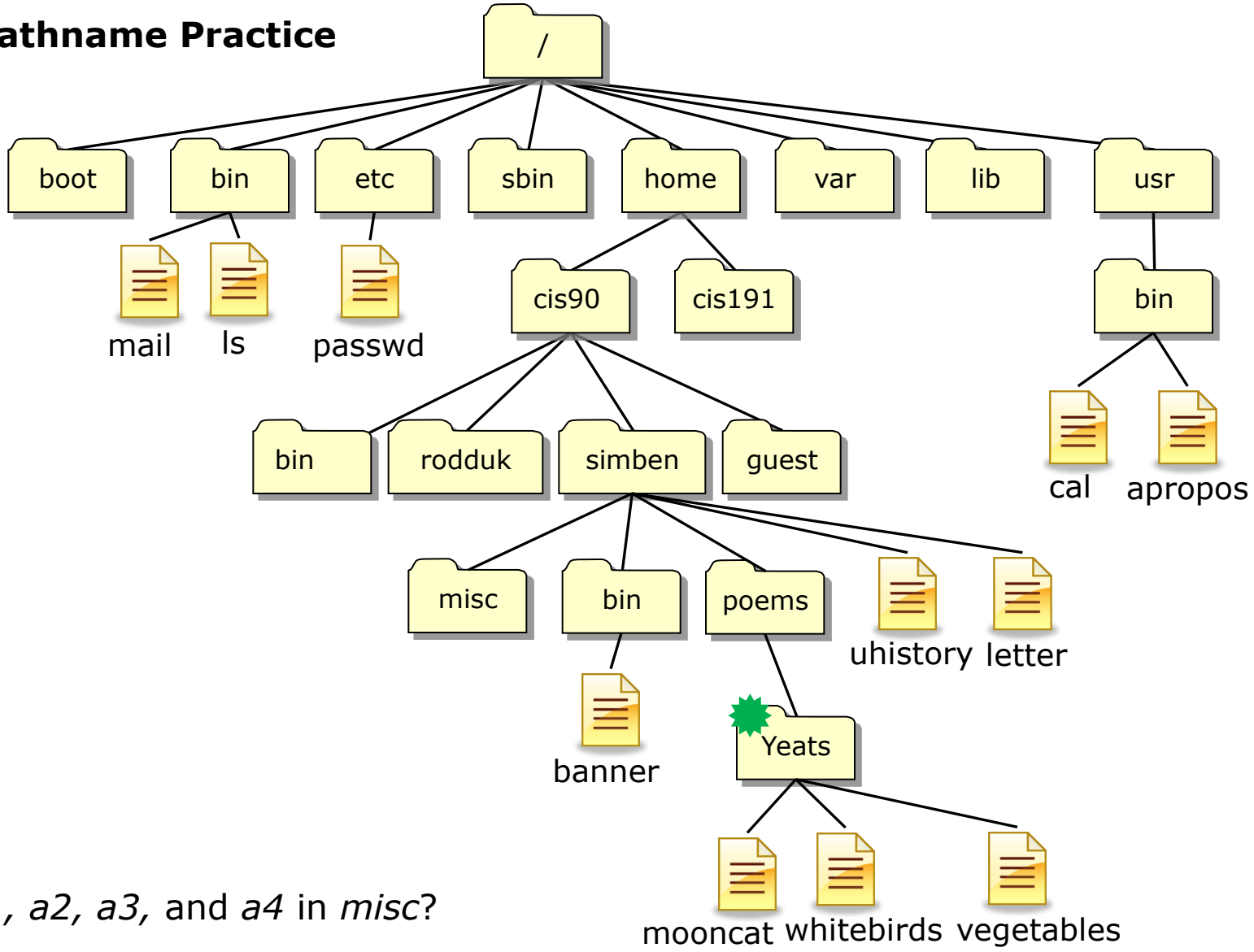
File Tree Pathname Practice



From how does Benji:

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

File Tree Pathname Practice



From how does Benji:

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

Housekeeping

Previous material and assignment

1. Lab 6 due today
2. Five posts due today
3. Early preview of Lab X2

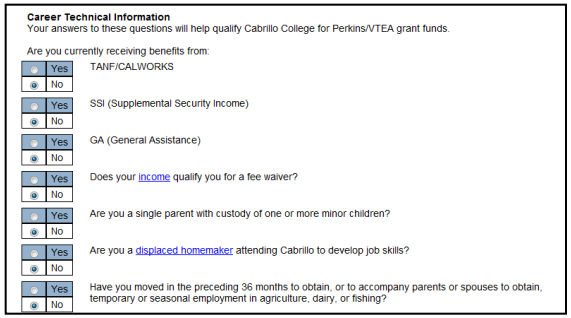
Perkins (VTEA) Web Advisor Survey Instructions

If you already filled this out in another class you don't need to do it again. This is the online survey for online classes.

- Log on to "www.cabrillo.edu" and go to the Cabrillo College Home Page
  - Select "WEBADVISOR" (bottom, lower left)
  - Select the "LOG IN" tab
  - Fill-in the "User ID" and "Password"
  - Click on "SUBMIT"

- Select "STUDENTS: Click Here" (navy blue bar)
  - Under "Academic Profile" Click on "Student Update Form"
    - Use drop down list under "Select the earliest term for which you are registered" and click on the current term (Spring 2012).
    - Select "SUBMIT"

- Scroll down to the "Career Technical Information"
  - **Answer questions** by clicking on the circle to the left of your "Yes" or "No" answers
    - You can get details about a question by clicking on blue underlined phrase
  - After answering all questions Select "SUBMIT"
  - Then "LOG OUT"



Thank you for taking a few minutes to help Cabrillo receive funding to support student services for CTE programs at Cabrillo College.

umask

Why umask?

Using the **umask** command

1. New files temporarily start with 666 permissions

2. New directories temporarily start with 777 permissions

3. The umask value is then applied which will **mask** out any unwanted permissions.

Using the **umask** command

The default umask on your Opus accounts is 002 which will always strip off write permission for others on newly created files

```
/home/cis90/simben $ rm wd3tb1
rm: cannot remove `wd3tb1': No such file or directory

/home/cis90/simben $ umask
0002

/home/cis90/simben $ touch wd3tb1

/home/cis90/simben $ ls -l wd3tb1
-rw-rw-r-- 1 simben90 cis90 0 Mar 28 06:50 wd3tb1
```

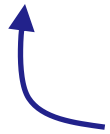
Note: the mnemonic form of 002 is --- --- -w-

write permission for others has been stripped off

Using the **umask** command

The same thing happens when a file is copied

```
/home/cis90/simben $ chmod 777 wd3tb1

/home/cis90/simben $ ls -l wd3tb1
-rwxrwxrwx 1 simben90 cis90 0 Mar 28 06:50 wd3tb1

/home/cis90/simben $ cp wd3tb1 wd3tb2

/home/cis90/simben $ ls -l wd3tb*
-rwxrwxrwx 1 simben90 cis90 0 Mar 28 06:50 wd3tb1
-rwxrwxr-x 1 simben90 cis90 0 Mar 28 06:52 wd3tb2
```



write permission for others has been stripped off

Using the **umask** command

The **umask** command can be used to set or view the current umask value.

With no arguments the umask value is displayed:

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

Note: the mnemonic form of 002 is --- --- -w-

Supply an argument to set the umask value:

```
/home/cis90/simben $ umask 077
/home/cis90/simben $ umask
0077
```

Note: the mnemonic version of 077 is --- rwx rwx

Using the **umask** command

For example a umask setting of 027 will mask out write permission for group and all permissions for others:

```
rw- rw- rw- (666)  starting point for files
--- -w- rwx (027)  umask setting
rw- r-- --- (640)  the permissions a new file will have
```

Prove it to yourself using Opus:

```
/home/cis90ol/simmsben $ rm a_new_file
rm: cannot remove `a_new_file': No such file or directory

/home/cis90ol/simmsben $ umask 027

/home/cis90ol/simmsben $ touch a_new_file

/home/cis90ol/simmsben $ ls -l a_new_file
-rw-r----- 1 simmsben cis90ol 0 Mar 31 10:57 a_new_file
```

Sample umask test question

What umask setting would insure that all new directories made would only have read and execute for owner, read only permission for group and no permission for others?

Answer: 237

```
rwx rwx rwx (777)  starting point for directories
-w- -wx rwx (237)  umask setting
r-x r-- --- (540)  the permissions a new file will have
```

Prove it to yourself using Opus:

```
/home/cis90ol/simmsben $ umask 237
/home/cis90ol/simmsben $ rmdir a_new_dir
rmdir: a_new_dir: No such file or directory

/home/cis90ol/simmsben $ mkdir a_new_dir
/home/cis90ol/simmsben $ ls -ld a_new_dir/
dr-xr----- 2 simmsben cis90ol 4096 Mar 31 11:08 a_new_dir/
```

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

Input and Output
File Descriptors

Example program: sort command

```
/home/cis90/roddyduk $ cat names
duke
benji
homer
lucy
scout
chip
/home/cis90/roddyduk $ sort names
benji
chip
duke
homer
lucy
scout
```

*The sort command will sort the lines in a file and send the sorted lines to **stdout** (defaults to the terminal)*

Input and Output
File Descriptors

Example program: sort command

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

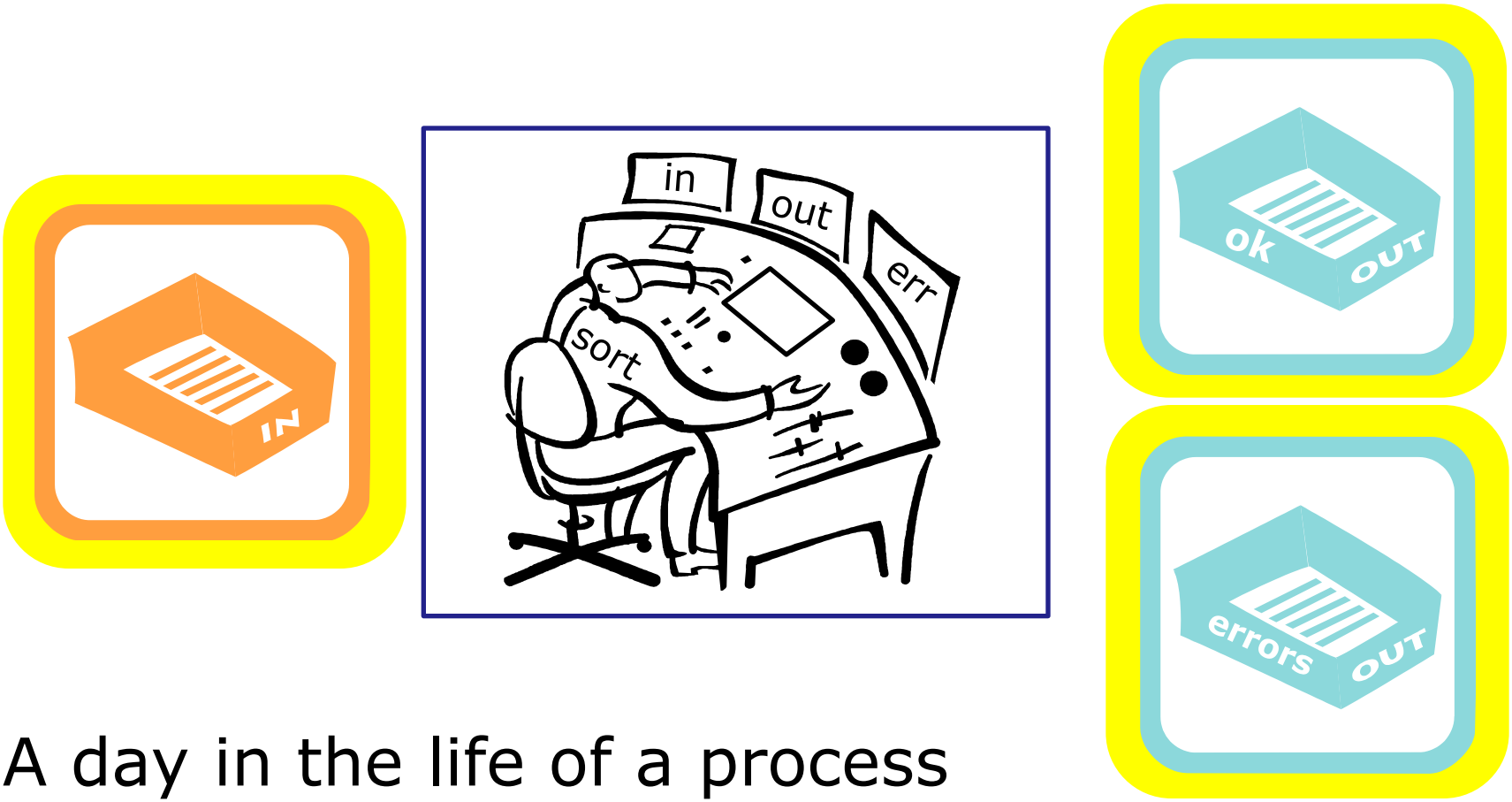
*If a file name is not specified as an argument on the command line, then the **sort** command will start reading from **stdin** (defaults to the keyboard) until it gets an EOF (End of File).*

*After getting the EOF, the lines are sorted and sent to **stdout** (defaults to the terminal)*

Lets visualize the sort program being loaded into memory and running as a process by the kernel

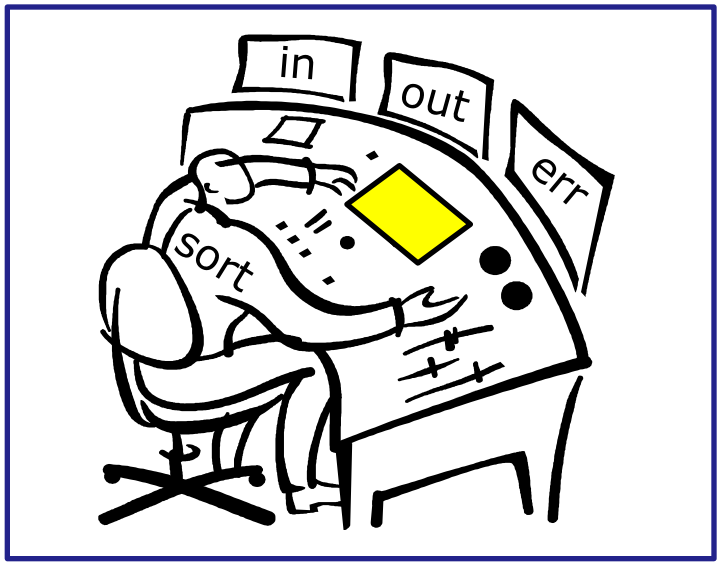
A day in the life of a process

There is one in tray and two out trays



A day in the life of a process

There is also a place where the process can check to see if there were any options or arguments specified on the command line

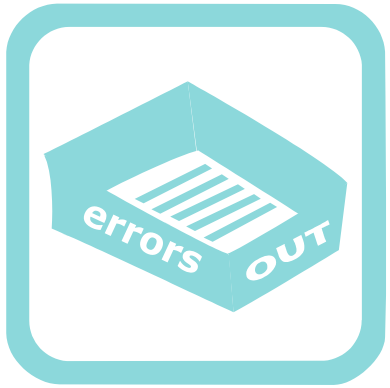


A day in the life of a process

sort process example no args

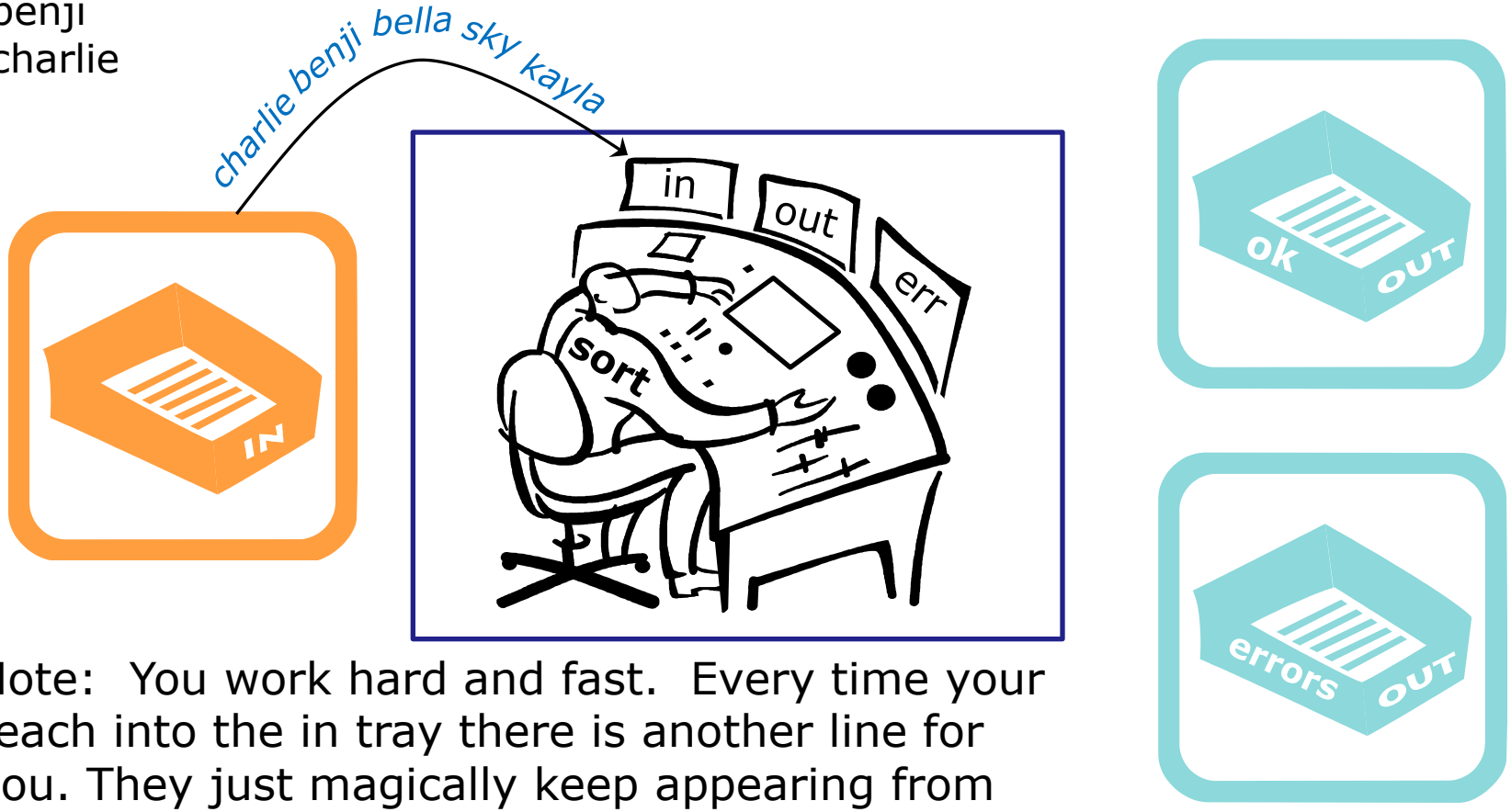
/home/cis90/simben $ **sort**

The sort process begins by checking to see if there are any options or arguments collected (and expanded) by the shell. In this case there are no options and no arguments.



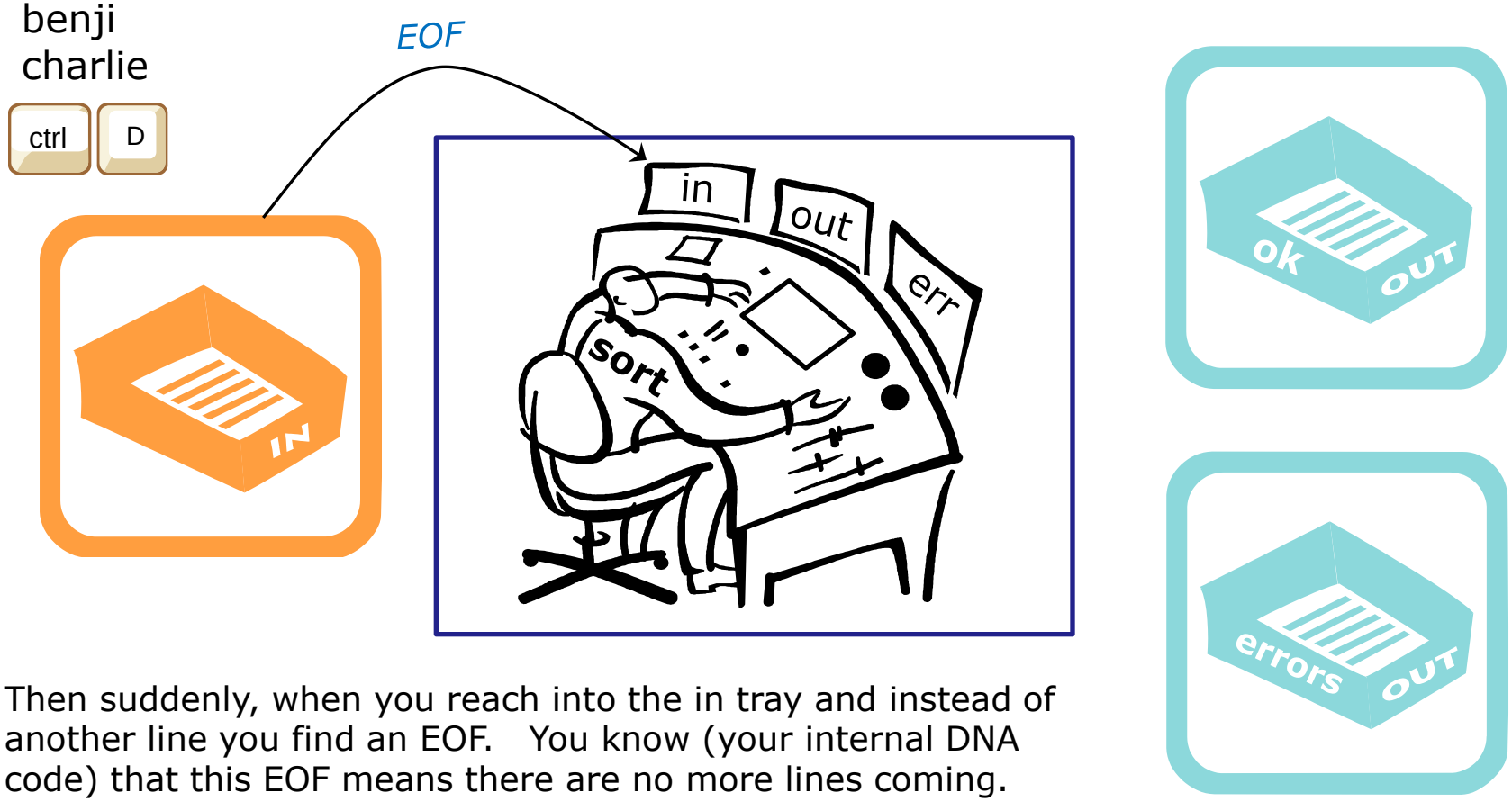
You check your little instruction window and see no options or arguments to handle. Given that you reach into your in tray to grab the first line to sort.

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



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

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



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

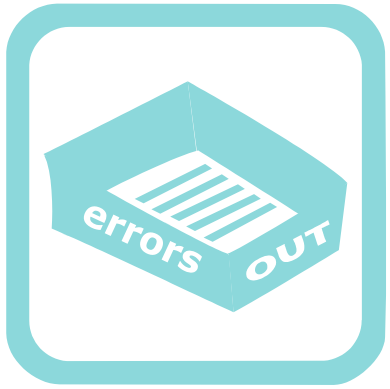
bella
benji
charlie
kayla
sky
/home/cis90/simben $

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

sort process example bad arg

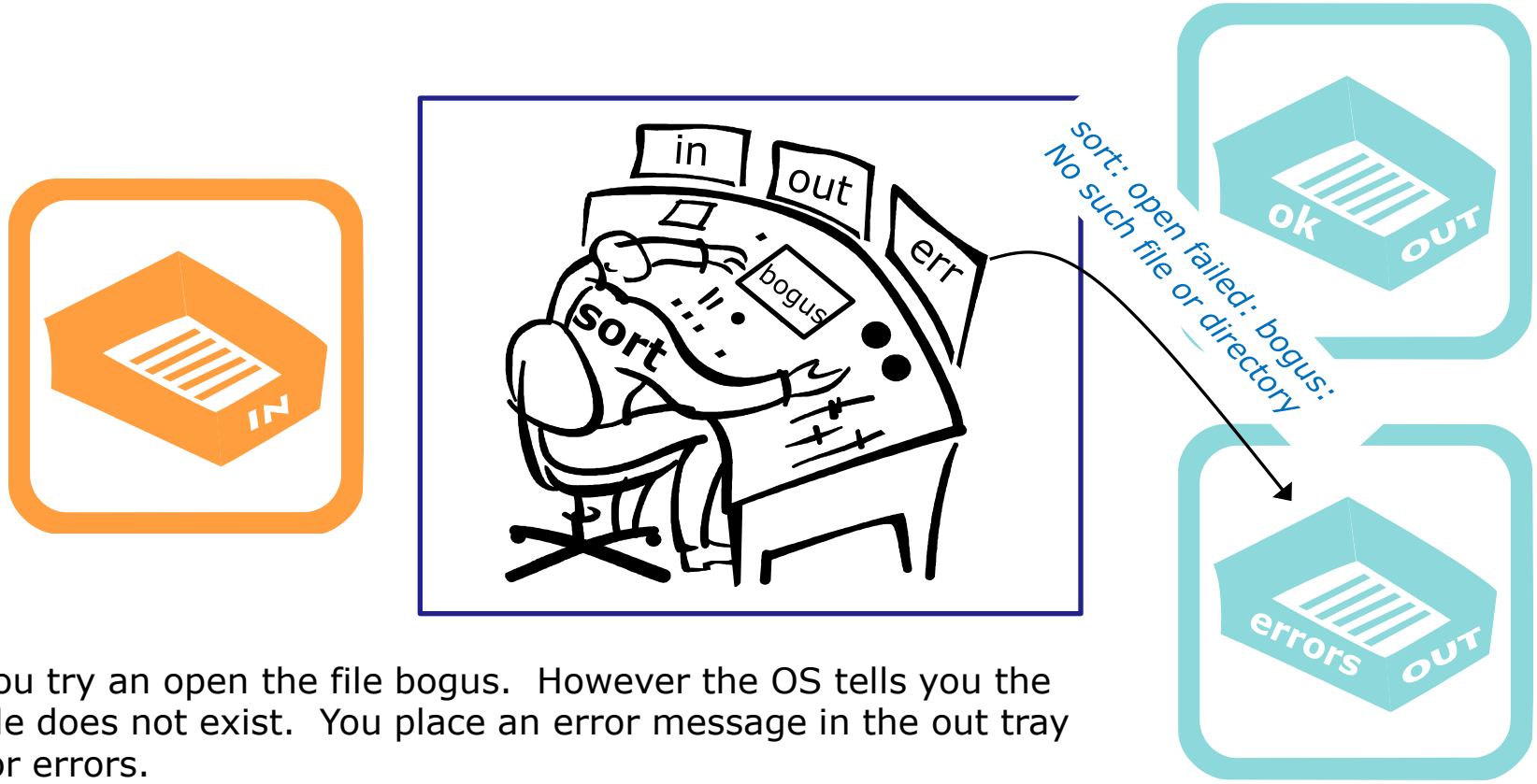
/home/cis90/simben $ **sort bogus**

The sort process begins by checking to see if there are any options or arguments collected (and expanded) by the shell. In this case there is one argument: bogus



You check your little instruction window and see an argument (bogus). You know (your internal DNA) tells you this must be a file name containing lines to sort

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



You try an open the file bogus. However the OS tells you the file does not exist. You place an error message in the out tray for errors.

bringing it home

Ok, lets make the visualization a little more realistic

stdout (1)

stdin (0)

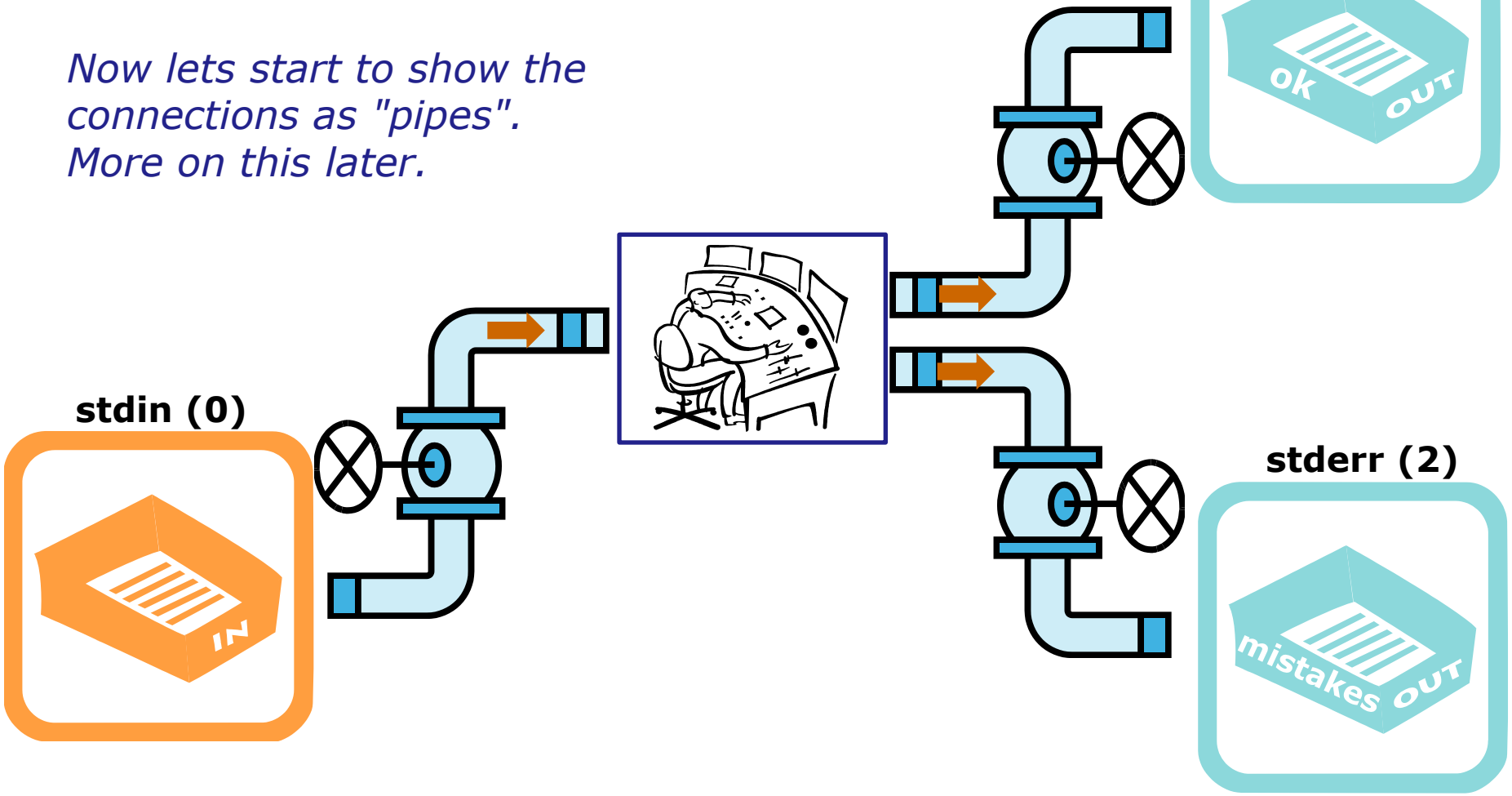
stderr (2)

The actual in and out trays have names as well as numbers **... stdin (0) stdout (1**) and **stderr (2)**.

Input and Output

File Descriptors

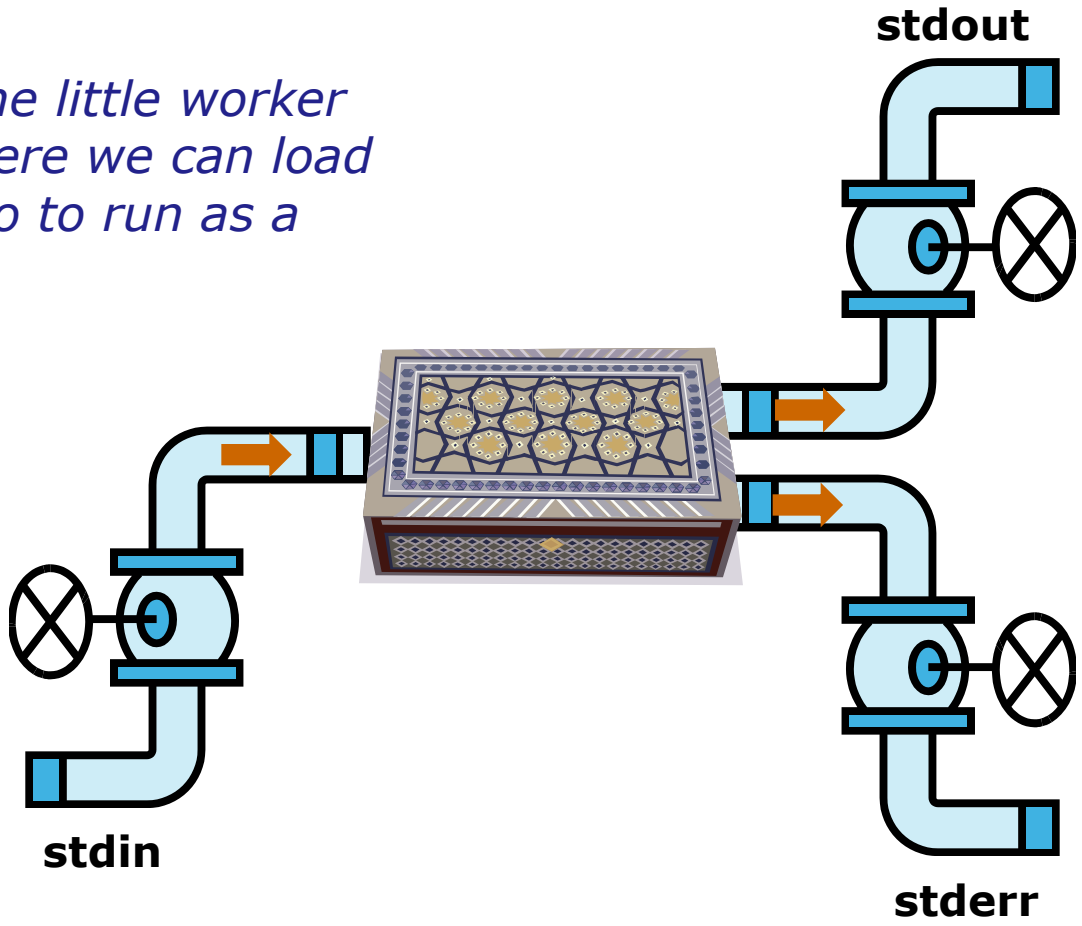
Now lets start to show the connections as "pipes". More on this later.



Input and Output

File Descriptors

Lets replace the little worker with a box where we can load ***programs*** *into to run as a* ***process***



normal output is written to stdout

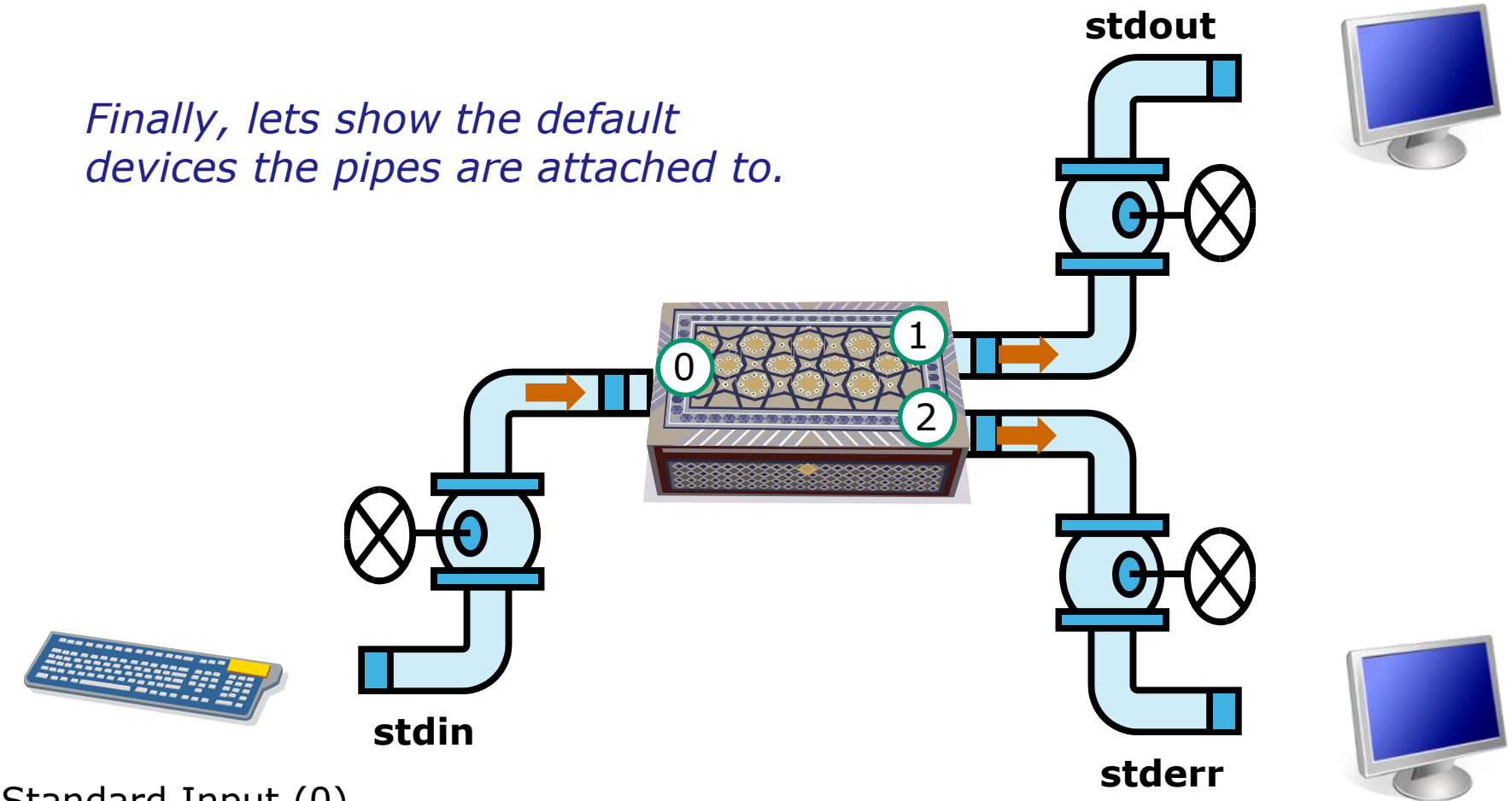
input (if necessary) is read from stdin

errors are written to stderr

Input and Output
File Descriptors

Standard Output (1)
defaults to the user's terminal

Finally, lets show the default devices the pipes are attached to.



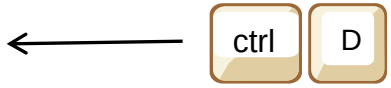
Standard Input (0)
defaults to the user's keyboard

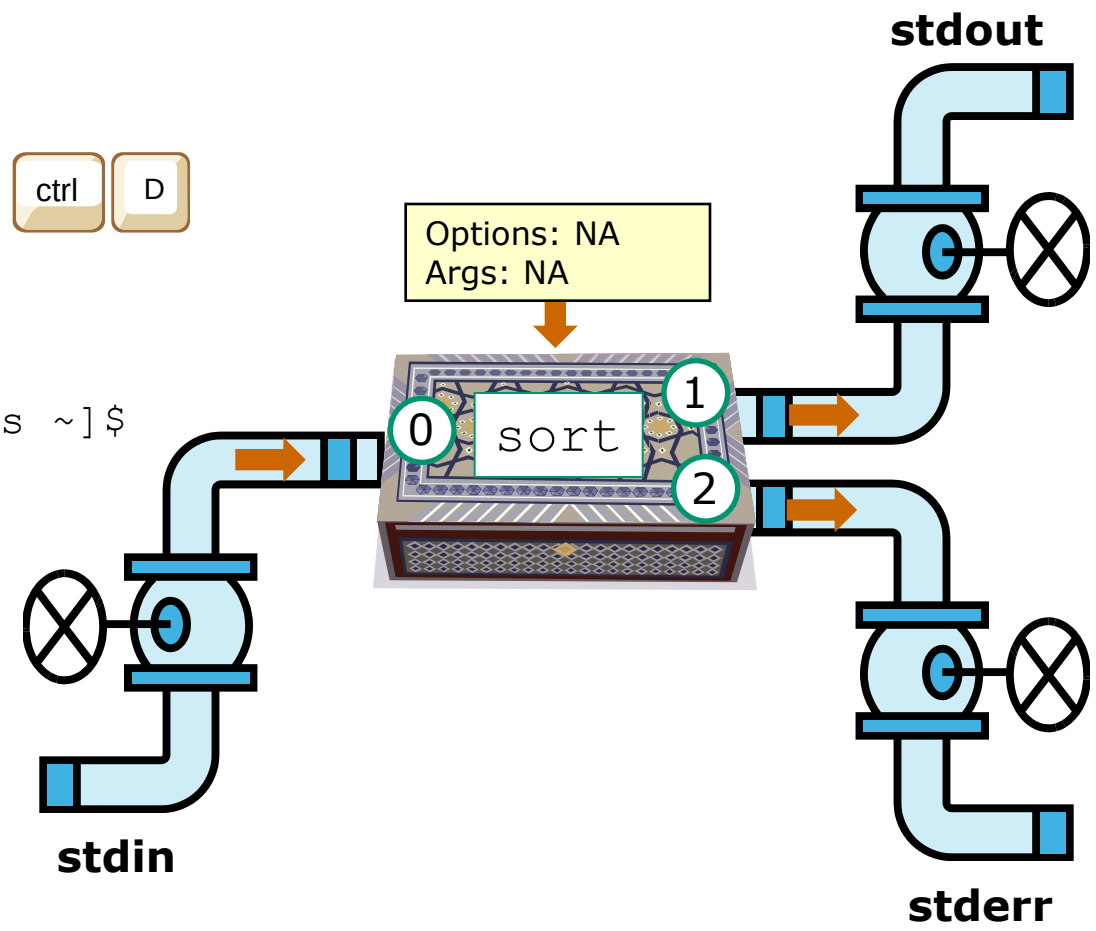
Standard Error (2)
defaults to the user's terminal

Input and Output

File Descriptors

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





```
benji
duke
homer
star
```

```
star
benji
duke
homer
```

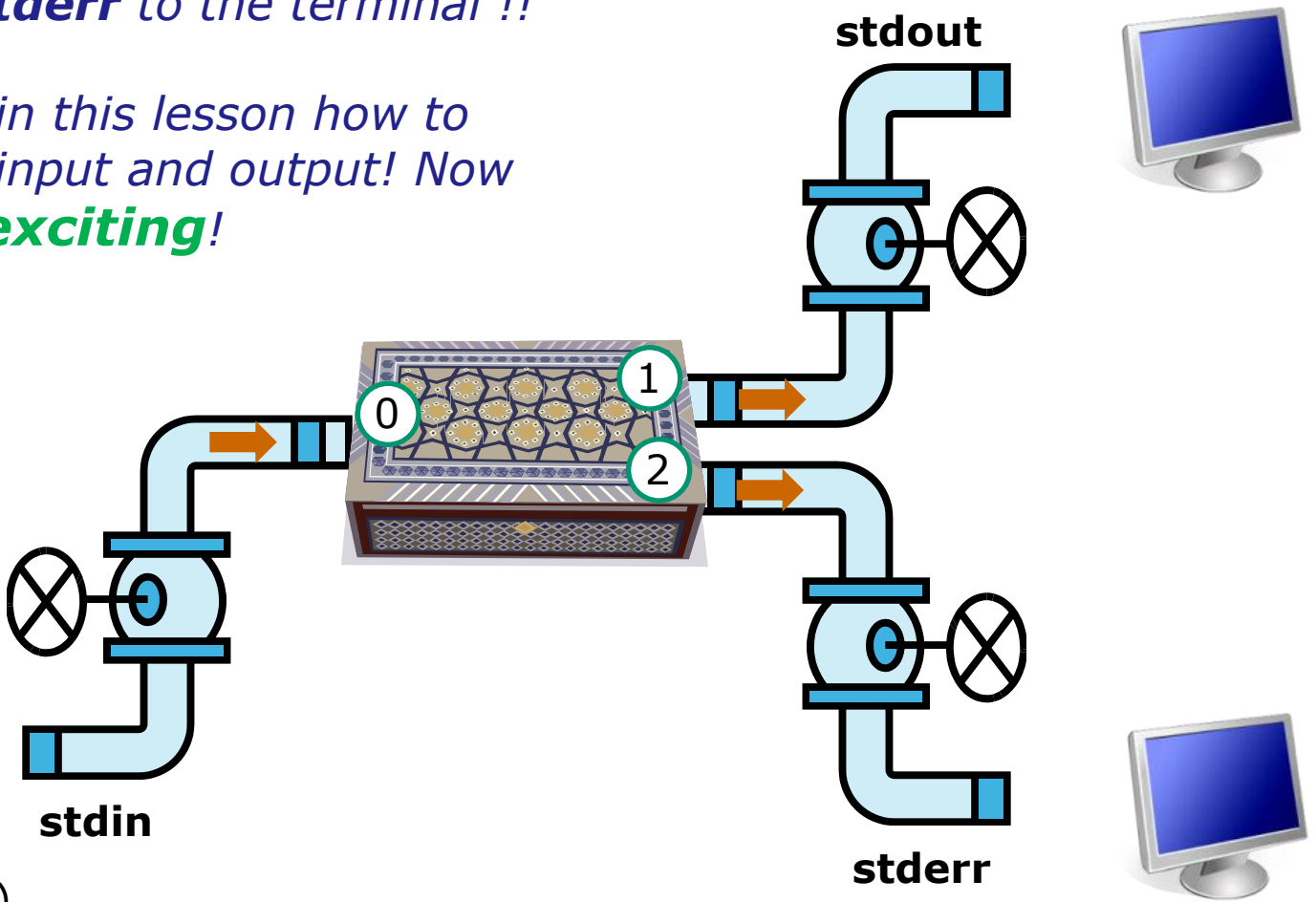
*Note, the sort program in this example gets its input from the keyboard via **stdin***

File Redirection

*Life would be **boring** if **stdin** was always attached to the keyboard, and **stdout** and **stderr** to the terminal !!*

*We will learn in this lesson how to redirect both input and output! Now that is more **exciting**!*

Standard Output (1)
defaults to the user's terminal



Standard Input (0)
defaults to the user's keyboard

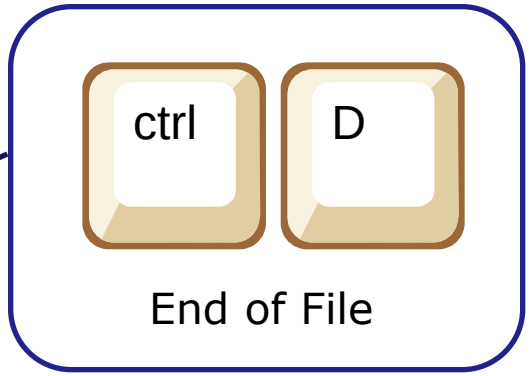
Standard Error (2)
defaults to the user's terminal

Input and Output

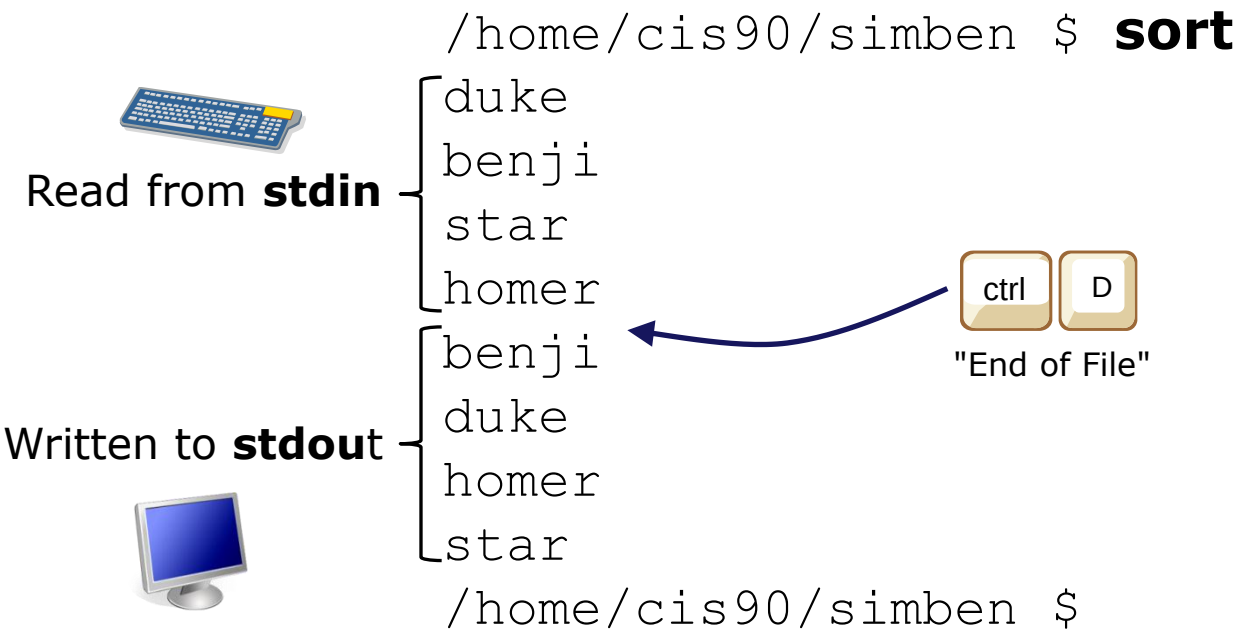
File Redirection

Let's look at the sort example again

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



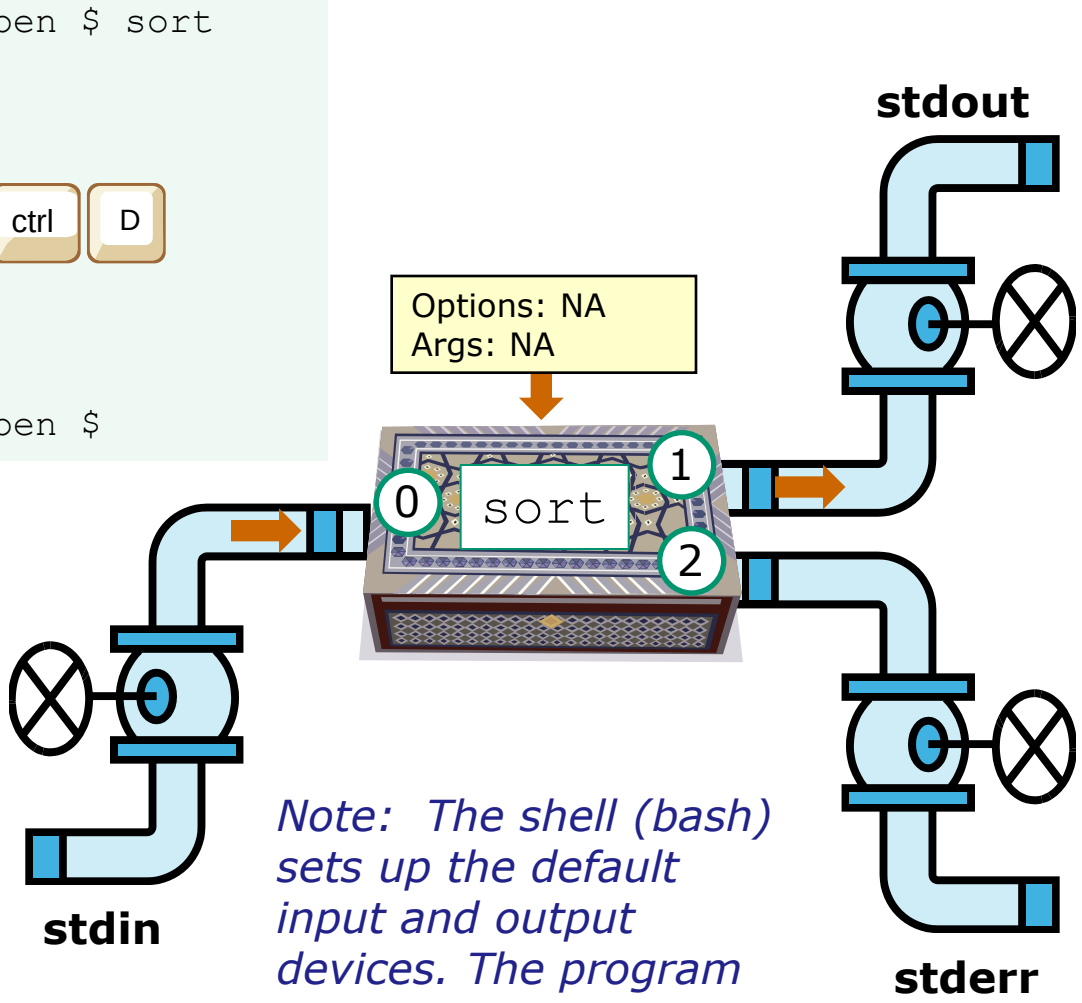
Input and Output
File Redirection



*The sort program reads lines from **stdin** (attached to keyboard), performs the sort, then writes to **stdout** (attached to terminal)*

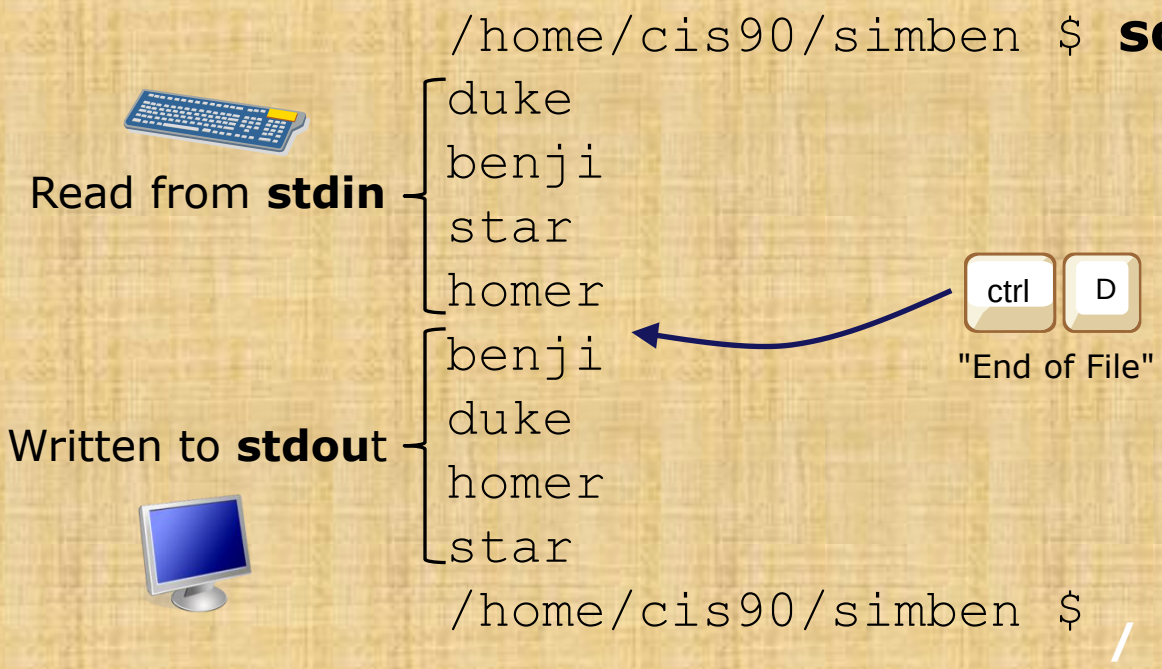
Example program to process: sort command

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



Note: The shell (bash) sets up the default input and output devices. The program is never even aware of what is at the end of the pipes.

Activity



Now you try it with your own list

Input and Output
File Redirection

But what if we could tell the shell (bash) to change the devices at the end of the pipes? We can!

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

0< filename

Input will now come from filename rather than the keyboard.

1> filename

Output will now go to filename instead of the terminal.

2> filename

Error messages will now go to filename instead of the terminal.

>> filename

Output will now be appended to filename.

The redirection is specified on the command line using the syntax specified below …

Input and Output
File Redirection

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

0< filename

Input will now come from filename rather than the keyboard.

1> filename

Output will now go to filename instead of the terminal.

2> filename

Error messages will now go to filename instead of the terminal.

>> filename

Output will now be appended to filename.

*The 0 in 0< is **not** necessary, just use < to redirect stdin*
*The 1 in 1> is **not** necessary, just use > to redirect stdout*
*The 2 in 2> **is** necessary, always use 2> to redirect stderr*

Input and Output

File Redirection

Lets try redirecting stdout …

sort writes to stdout, and stdout has been redirected to the file dogsinorder

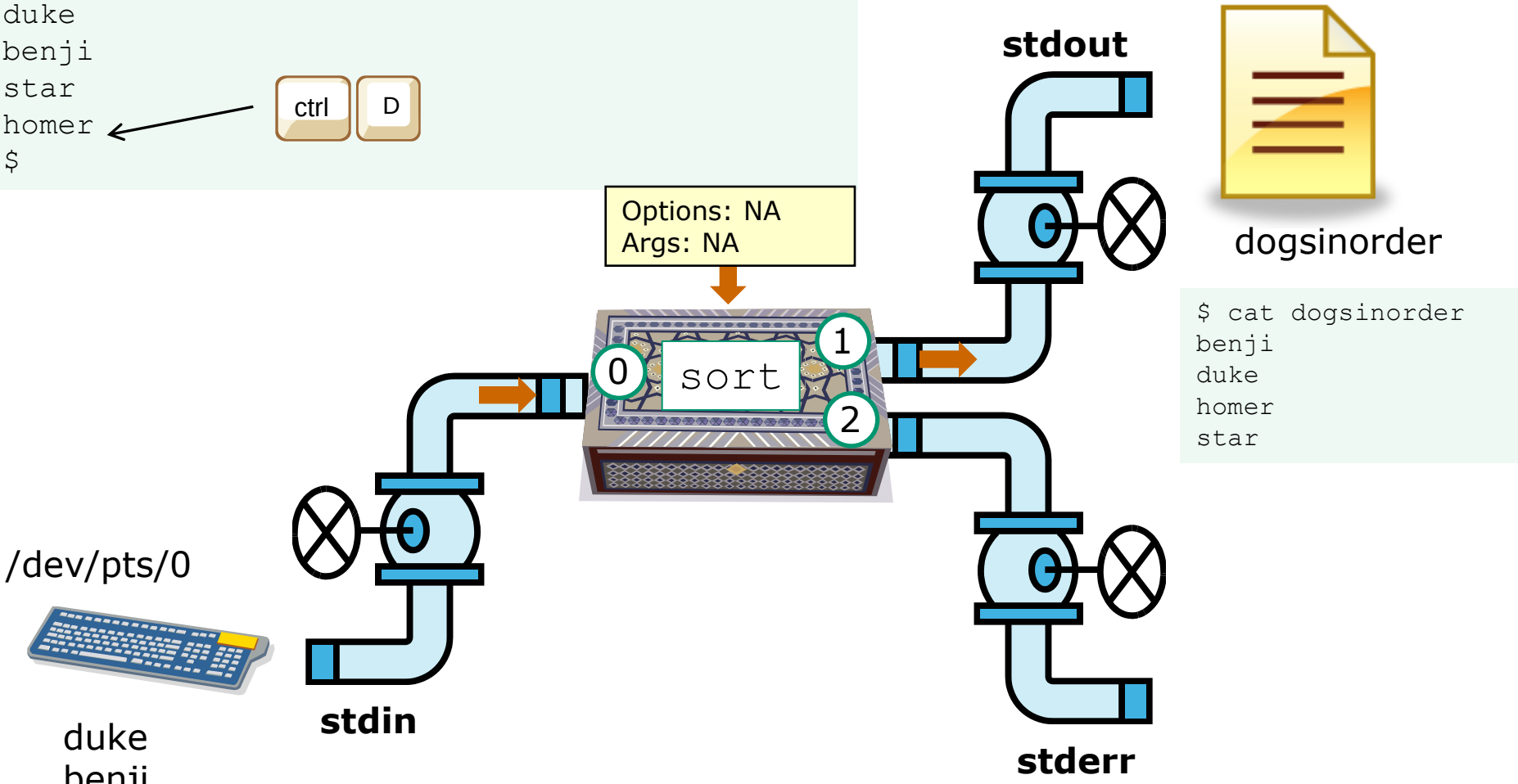
```
[simmsben@opus ~]$ sort > dogsinorder
duke
benji
star
homer
[simmsben@opus ~]$ cat  dogsinorder
benji
duke
homer
star
[simmsben@opus ~]$
```

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

Example program to process: sort command

```
$ sort > dogsinorder
duke
benji
star
homer
$
```

ctrl D




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

*Note: sort doesn't know about the keyboard (/dev/pts/0) or dogsinorder file. It just reads from **stdin** and writes to **stdout**.*

Input and Output
File Redirection

Create a file named names and fill it with your favorite dog names to use in the next example

```
/home/cis90/simben $ echo duke  >  names
/home/cis90/simben $ echo benji  >>  names
/home/cis90/simben $ echo star  >>  names
/home/cis90/simben $ echo homer >> names
```

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

Note, the use of >> to append the output of the echo command to the end of the names file

/

Input and Output

File Redirection

Let's try redirecting BOTH stdin and stdout …

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

input is redirected from the file names

output is redirected to the file dogsinorder

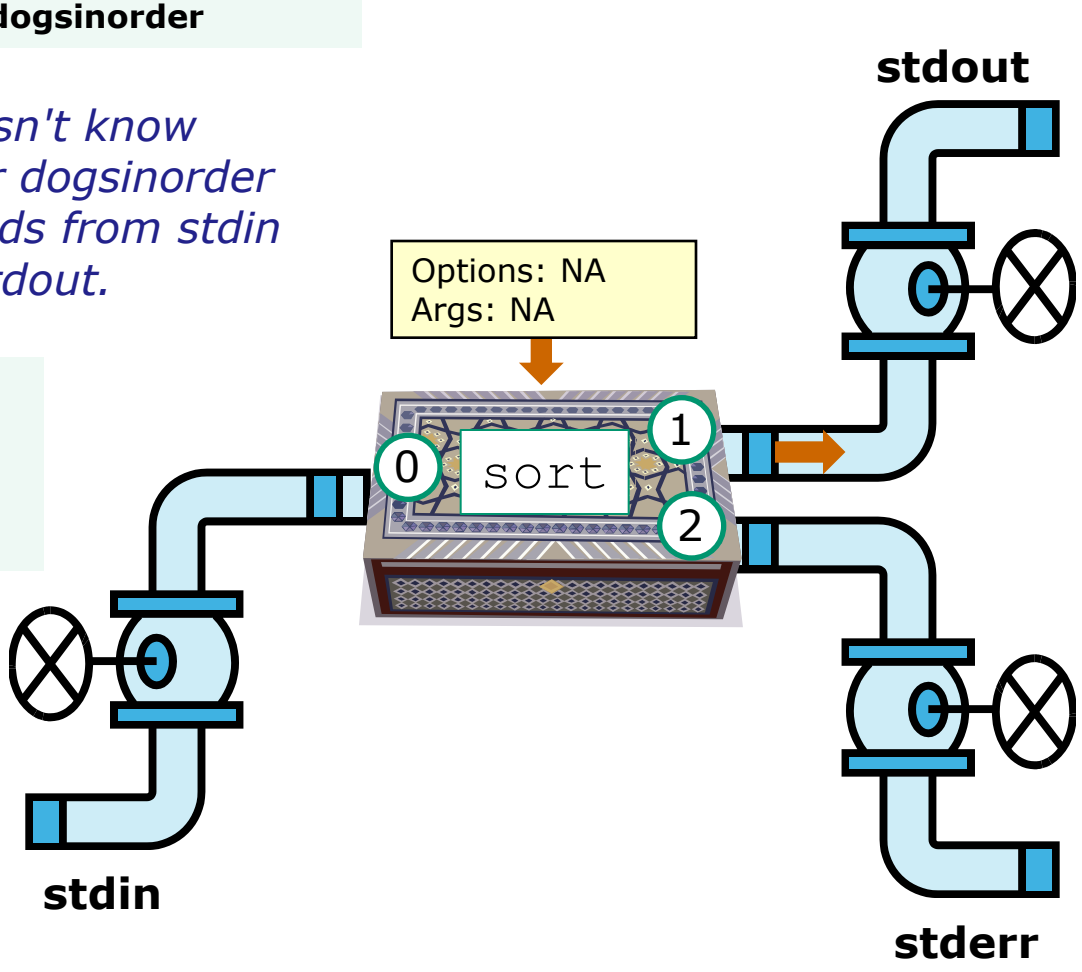
```
[simben@opus ~]$ cat dogsinorder
benji
duke
homer
star
[simben@opus ~]$
```

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

Example program to process: sort command

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

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




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

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

In this example, sort is getting it's input from stdin, which has been connected to the names file

Input and Output

File Redirection

*Now let's try something different. The difference on the command line is very subtle. The names file is now an **argument** passed to sort from the command line.*

*Output is redirected to the file dogsinorder. The sort program writes to **stdout** and has no idea **stdout** is really connected to the file dogsinorder. It is the shell that opens the file dogsinorder.*

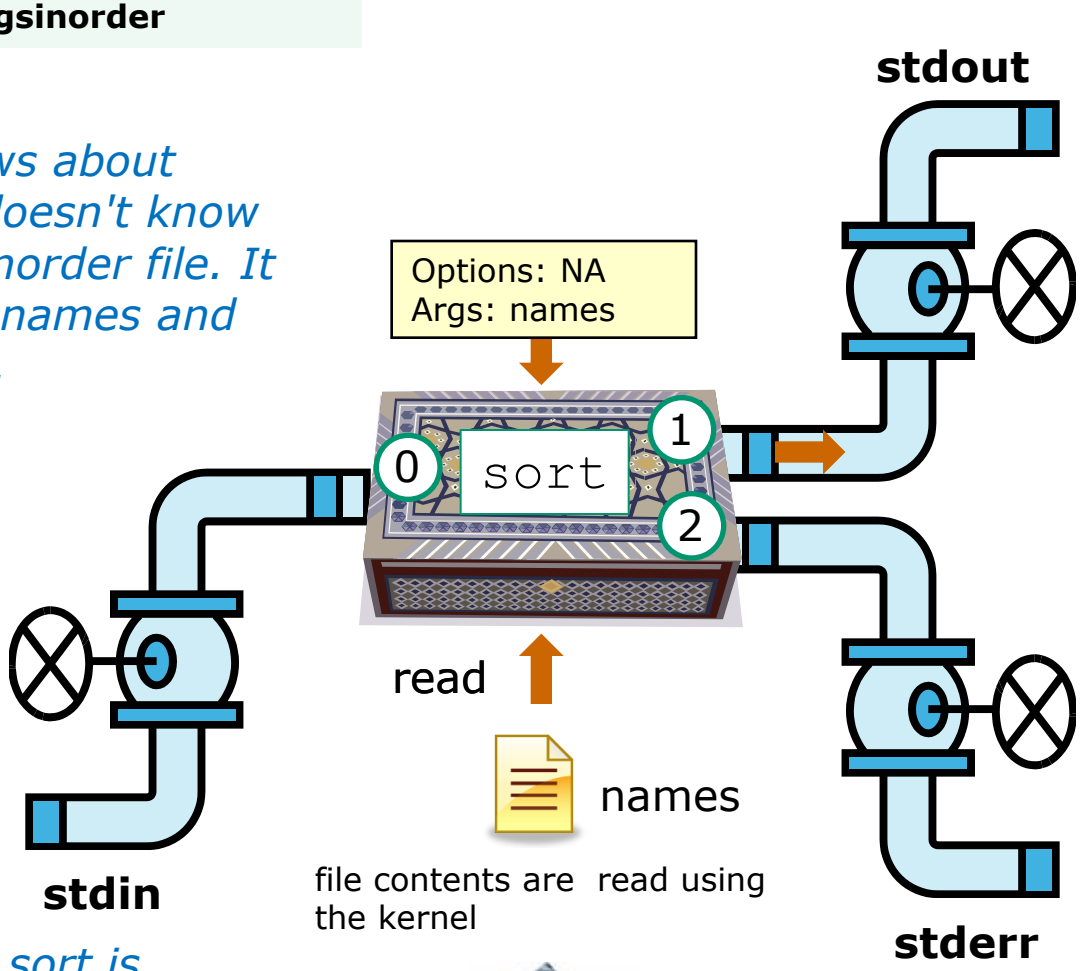
```
[simben@opus ~]$ sort names > dogsinorder
[simben@opus ~]$ cat dogsinorder
benji
duke
homer
star
[simben@opus ~]$
```

The sort program is fully aware of the names file. It is the sort program's responsibility to directly open this file and read it. This is done by the sort code making requests to the kernel to read data from the file on the hard drive.

Example program to process: sort command

```
$ sort names > dogsinorder
```

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




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

In this example, sort is getting it's input from the names file

Input and Output

File Redirection

OK, another little twist, lets pass in an option as well this time

names is an argument passed to the sort command

specifying an option (for reverse order)

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

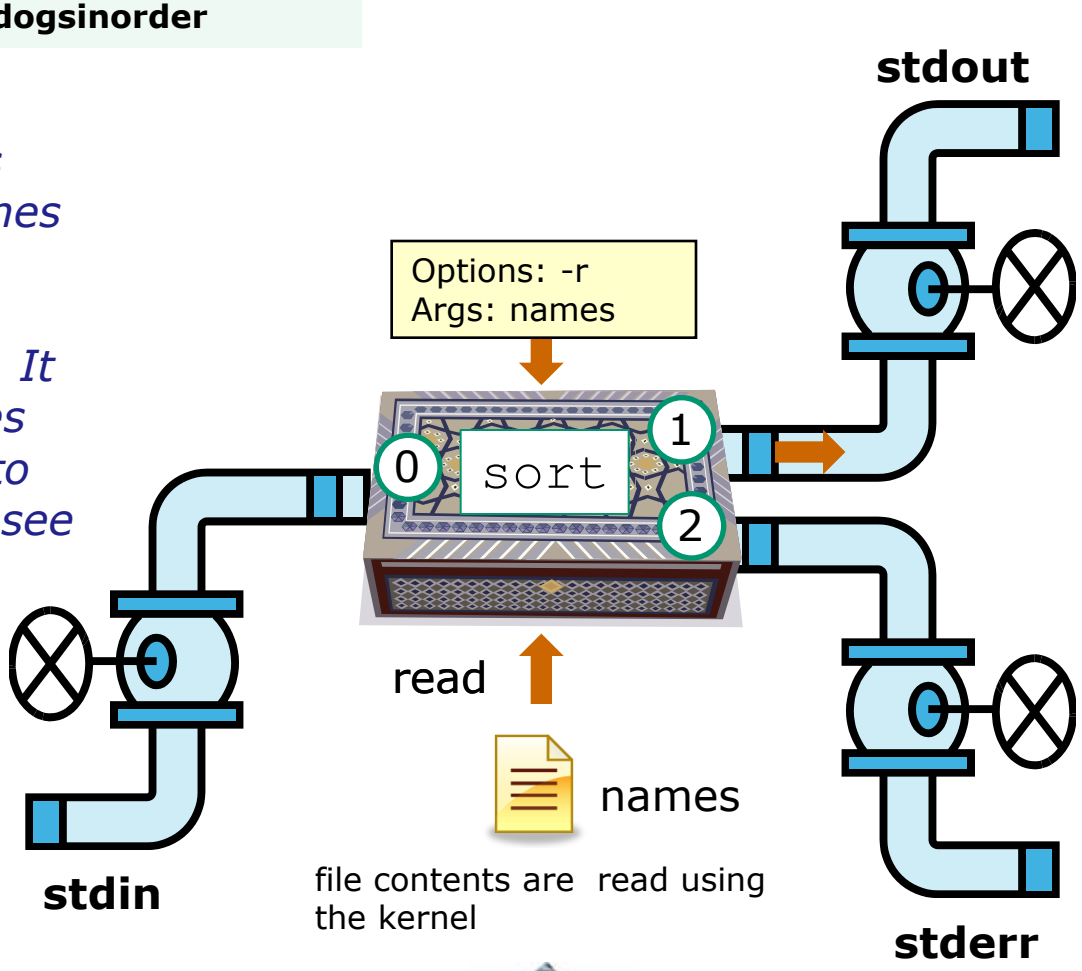
```
[simben@opus ~]$ sort -r names > dogsinorder
[simben@opus ~]$ cat dogsinorder
star
homer
duke
benji
[simben@opus ~]$
```

This –r option does the sort in reverse order

Example program to process: sort command

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

Note: sort does know about names file but doesn't know about dogsinorder file. It just reads names file and writes to stdout. It does see the option and modifies how it sorts.




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

In this example, sort is getting it's input from the names file

Input and Output
File Redirection

/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
```

Input and Output
File Redirection

Be careful using > for redirection!

```
[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
```

>> appends to the end of the file

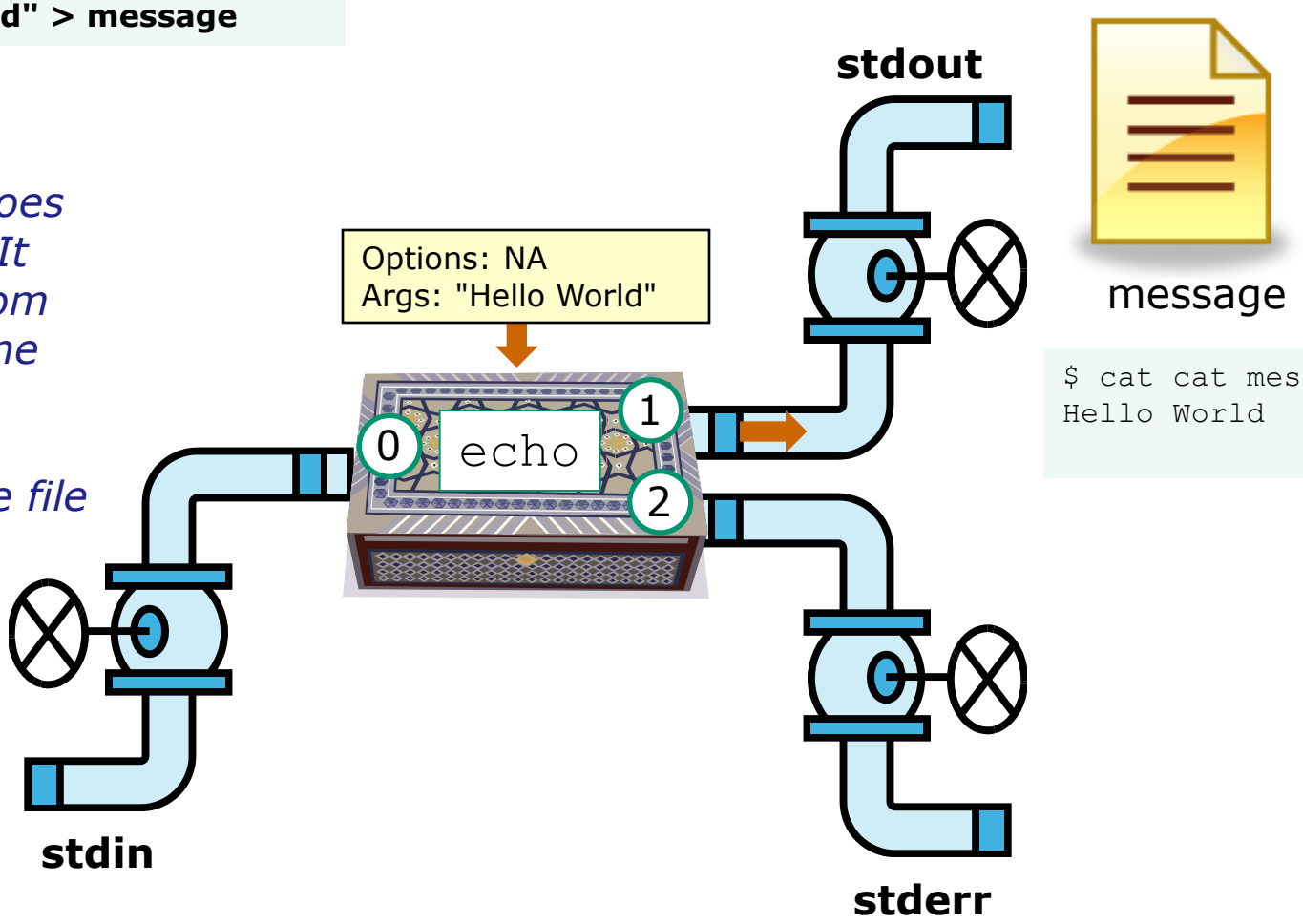
```
[simben@opus ~]$ echo "Oops" > message
[simben@opus ~]$ cat message
Oops
[simben@opus ~]$ > message
[simben@opus ~]$ cat message
[simben@opus ~]$
```

*> will **overwrite** anything already in the file!*

Example program to process: echo command

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

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




```
$ cat cat message
Hello World
```

In this example, echo is getting it's input from the command line

Input and Output

File Redirection

Another example …

```
[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 defaults to the terminal

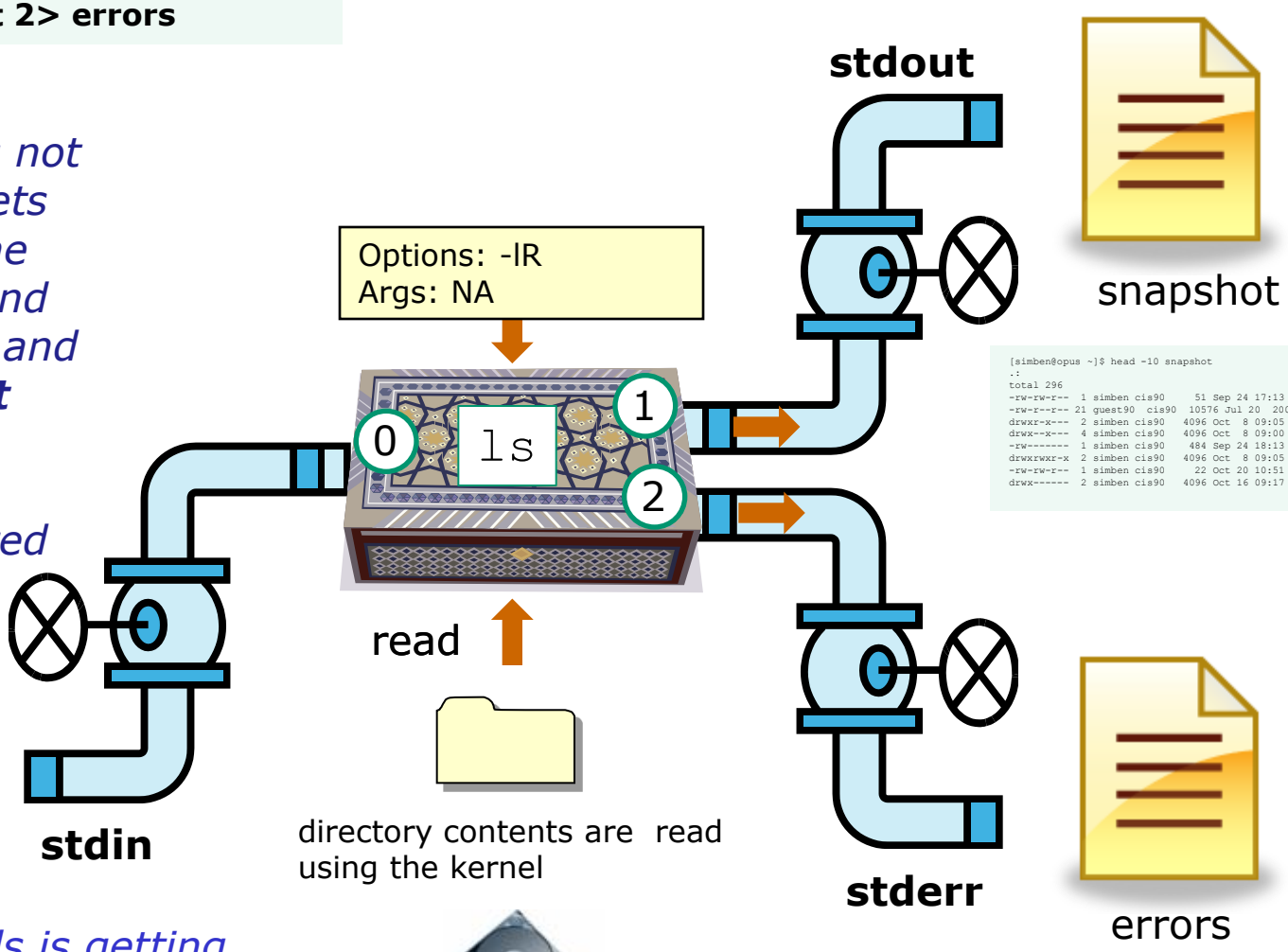
> redirects stdout to file named snapshot

2> redirects stderr to file named errors

Example program to process: ls command

```
$ 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 it's input from the OS

```
$ cat errors
ls: ./Hidden: Permission denied
```

Input and Output
File Redirection

Another example … using all three

```
[simben@opus ~]$ echo 2+2 > math
[simben@opus ~]$ bc < math
4
[simben@opus ~]$ echo 4/0 >> math
[simben@opus ~]$ cat math
2+2
4/0
[simben@opus ~]$ bc < math
4
Runtime error (func=(main), adr=5): Divide by zero
[simben@opus ~]$ bc  < math  > answers  2> errors
[simben@opus ~]$ cat answers
4
[simben@opus ~]$ cat errors
Runtime error (func=(main), adr=5): Divide by zero
[simben@opus ~]$
```

Note: bc reads from stdin which is redirected to math

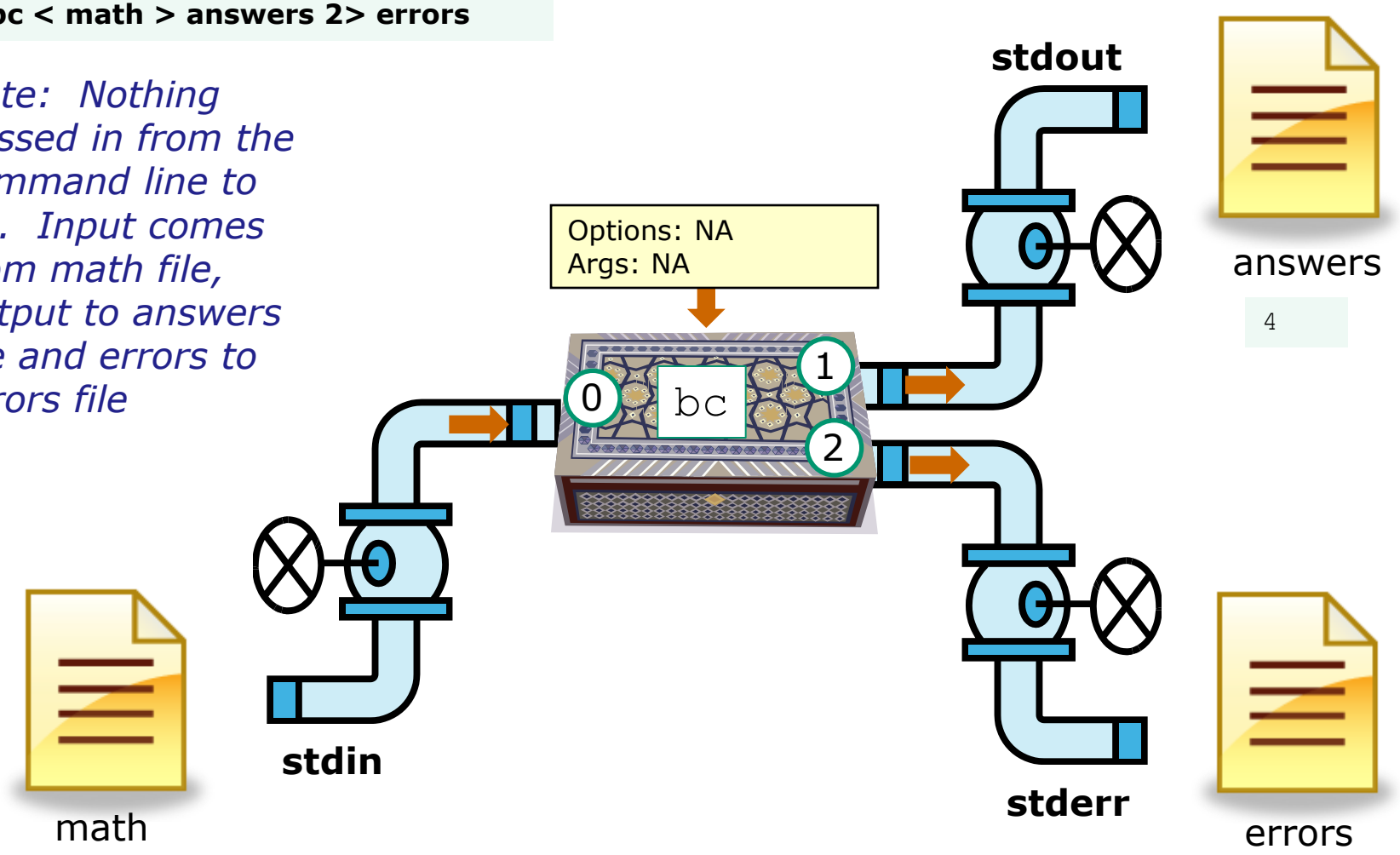
dividing by zero always results in an error

*input from math (via **stdin**), normal output to answers (via **stdout**) and error output to errors (via **stderr**)*

Example program to process: bc command

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

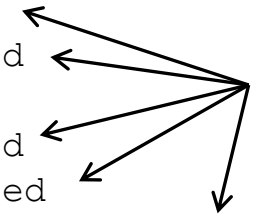
*Note: Nothing passed in from the command line to **bc**. Input comes from math file, output to answers file and errors to errors file*



Input and Output
File Redirection

Introducing the bit bucket

```
[simben@opus ~]$ find . -name sonnet6
find: ./Hidden: Permission denied
./poems/Shakespeare/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

<snipped>

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



Yuck! How annoying is this?

Input and Output

File Redirection

/dev/null AKA the "bit bucket"

Introducing 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 ~]$
```

Much better!

All error messages are redirected to the bit bucket

This is how you can discard output you don't want to see

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.

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 the file users.

who | tee users | sort

Input and Output
Pipelines

Let's count the lines in letter

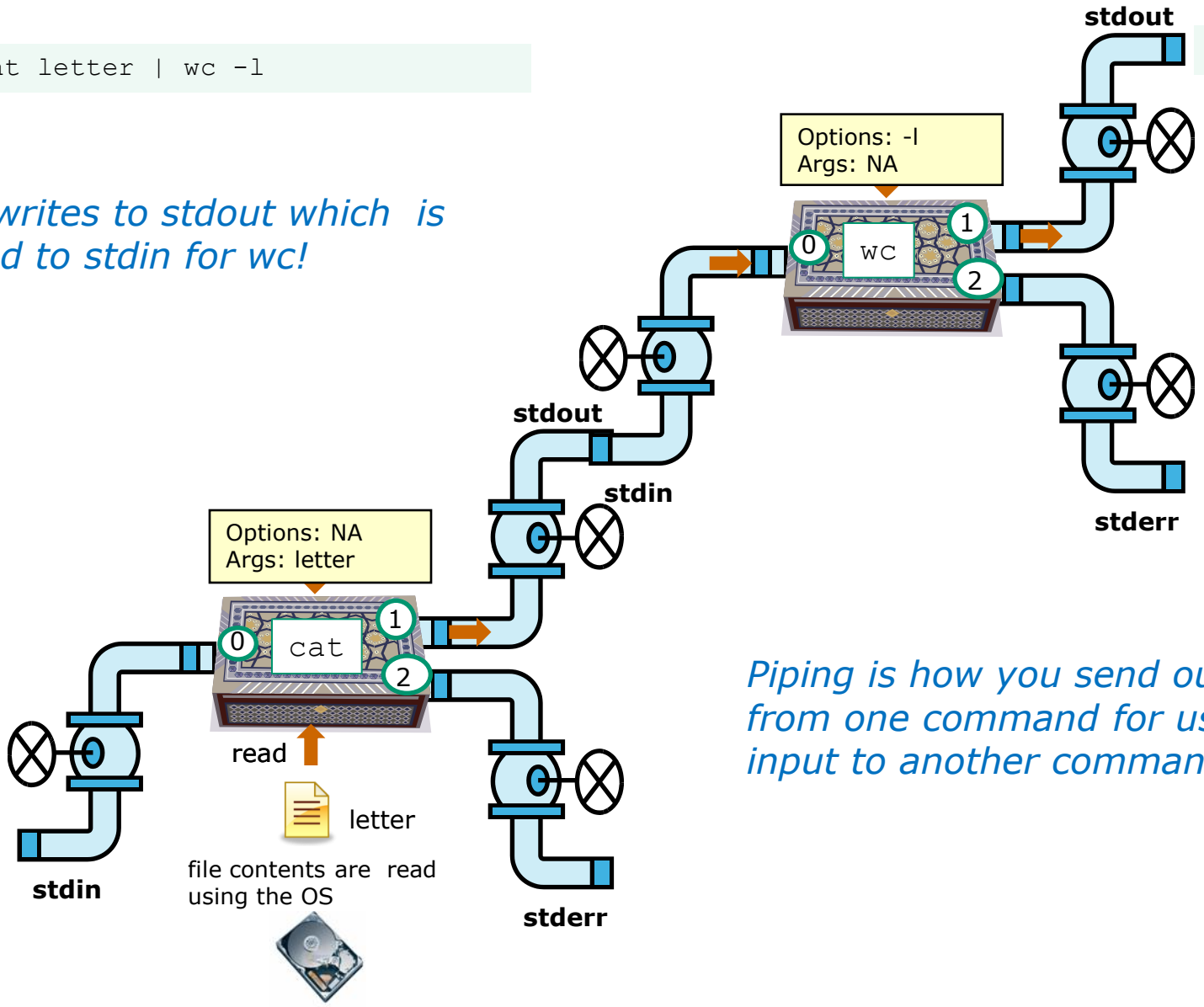
```
[simben@opus ~]$ cat letter | wc -l
28
[simben@opus ~]$
```

Example program to process: cat and wc commands

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

```
28
```

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

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***

Input and Output

Pipelines

Task: I would like to save a sorted list of users and a count of how many users are logged on

```
[simben@opus ~]$ who
simben pts/0        2008-10-19 18:36 (dsl-63-249-103-107.cruzio.com)
simben pts/1        2008-10-19 18:27 (dsl-63-249-103-107.cruzio.com)
rsimms   pts/2        2008-10-20 17:33 (dsl-63-249-103-107.cruzio.com)
bolasale pts/4        2008-10-21 10:43 (dsl-63-249-97-17.cruzio.com)
[simben@opus ~]$ who > tempfile
[simben@opus ~]$ sort tempfile
bolasale pts/4        2008-10-21 10:43 (dsl-63-249-97-17.cruzio.com)
simben pts/0        2008-10-19 18:36 (dsl-63-249-103-107.cruzio.com)
simben pts/1        2008-10-19 18:27 (dsl-63-249-103-107.cruzio.com)
rsimms   pts/2        2008-10-20 17:33 (dsl-63-249-103-107.cruzio.com)
[simben@opus ~]$ sort tempfile > users
[simben@opus ~]$ wc -l users
4 users
[simben@opus ~]$ cat users
bolasale pts/4        2008-10-21 10:43 (dsl-63-249-97-17.cruzio.com)
simben pts/0        2008-10-19 18:36 (dsl-63-249-103-107.cruzio.com)
simben pts/1        2008-10-19 18:27 (dsl-63-249-103-107.cruzio.com)
rsimms   pts/2        2008-10-20 17:33 (dsl-63-249-103-107.cruzio.com)
```


Method I – use temporary files

Input and Output

Pipelines

Task: I would like to save a sorted list of users and a count of how many users are logged on

```
[simben@opus ~]$ who | sort | tee users | wc -l
4
[simben@opus ~]$ cat users
bolasale pts/4        2008-10-21 10:43 (dsl-63-249-97-17.cruzio.com)
simben pts/0        2008-10-19 18:36 (dsl-63-249-103-107.cruzio.com)
simben pts/1        2008-10-19 18:27 (dsl-63-249-103-107.cruzio.com)
rsimms   pts/2        2008-10-20 17:33 (dsl-63-249-103-107.cruzio.com)
[simben@opus ~]$
```

Method II – uses pipes

Input and Output
Pipelines

Let break it down a little to see what's going on …

```
[simben@opus ~]$ who          who is logged in
simben pts/0        2008-10-19 18:36 (dsl-63-249-103-107.cruzio.com)
simben pts/1        2008-10-19 18:27 (dsl-63-249-103-107.cruzio.com)
rsimms   pts/2        2008-10-20 17:33 (dsl-63-249-103-107.cruzio.com)
bolasale pts/4        2008-10-21 10:43 (dsl-63-249-97-17.cruzio.com)
[simben@opus ~]$ who | sort     who is logged in and sorted
bolasale pts/4        2008-10-21 10:43 (dsl-63-249-97-17.cruzio.com)
simben pts/0        2008-10-19 18:36 (dsl-63-249-103-107.cruzio.com)
simben pts/1        2008-10-19 18:27 (dsl-63-249-103-107.cruzio.com)
rsimms   pts/2        2008-10-20 17:33 (dsl-63-249-103-107.cruzio.com)
[simben@opus ~]$ who | sort | wc -l      who is logged in, sorted and counted
4
[simben@opus ~]$ who | sort | tee users | wc -l      who is logged in, sorted, counted
4                                                    and saved in file named users
[simben@opus ~]$ cat users
bolasale pts/4        2008-10-21 10:43 (dsl-63-249-97-17.cruzio.com)
simben pts/0        2008-10-19 18:36 (dsl-63-249-103-107.cruzio.com)
simben pts/1        2008-10-19 18:27 (dsl-63-249-103-107.cruzio.com)
rsimms   pts/2        2008-10-20 17:33 (dsl-63-249-103-107.cruzio.com)
```

Miscellaneous Commands

Input and Output
Miscellaneous Commands

find – Find file or content of a file
grep – "Global Regular Expression Print"
✓ **sort** - sort
spell – spelling correction
✓ **wc** – word count

We will learn how to string commands together shortly using pipelines. The commands above are useful both by themselves and in pipelines.

Lets explore the commands we haven't covered yet then get into pipelines.

Input and Output
Miscellaneous Commands

find – Find file or content of a file

grep – "Global Regular Expression Print"
sort - sort
spell – spelling correction
wc – word count

*The **find** command can be used to search for files from any point in the UNIX file tree and working down from there.*

find command

*The **find** command by itself lists all files from the directory specified and 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
[simben@opus poems]$
```

find command issued in the poems directory

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

find command

Task: How many files (approximately) are on Opus?

start searching in / (the top of the file tree)

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

*use the output of the **find** command as input to the **wc** command to count the number of files*

redirect permission errors into the bit bucket (discard them)

Note, this will not count any files in directories you don't have read permission for. Is there a user on Opus that will get a higher count when using this command?

find command

Task: Find files whose names start with "sonnet" in current home directory (including its sub-directores)

```
[simben@opus ~]$ find -name "sonnet*"
find: ./Hidden: Permission denied
./poems/Shakespeare/sonnet1
./poems/Shakespeare/sonnet2
./poems/Shakespeare/sonnet3
./poems/Shakespeare/sonnet4
./poems/Shakespeare/sonnet5
./poems/Shakespeare/sonnet7
./poems/Shakespeare/sonnet9
./poems/Shakespeare/sonnet10
./poems/Shakespeare/sonnet15
./poems/Shakespeare/sonnet17
./poems/Shakespeare/sonnet26
./poems/Shakespeare/sonnet35
./poems/Shakespeare/sonnet11
./poems/Shakespeare/sonnet6
[simben@opus ~]$
```

Note:

No starting point for the search is specified, so find will start in the current directory which in this example is simben's home directory

***-name "sonnet*"** is an option passed to the find command directing it to only look for files with names starting with "sonnet"*

find command

Task: Find sonnet6 files starting in parent directory

```
[simben@opus ~]$ find .. -name "sonnet6" 2> /dev/null
../guest/Poems/Shakespeare/sonnet6
../simben/poems/Shakespeare/sonnet6
../stanlcha/poems/Shakespeare/sonnet6
../seatocol/poems/Shakespeare/sonnet6
../wrigholi/poems/Shakespeare/sonnet6
../dymesdia/poems/Shakespeare/sonnet6
../lyonsrob/poems/Shakespeare/sonnet6
../ybarrser/poems/Shakespeare/sonnet6
../ybarrser/poems/Sonnets/sonnet6
../valdemar/poems/Shakespeare/sonnet6
../elliokat/poems/Shakespeare/sonnet6
../jessuwes/poems/Shakespeare/sonnet6
../luisjus/poems/Shakespeare/sonnet6
../meyerjas/poems/Shakespeare/sonnet6
../bergelyl/sonnet6
../bergelyl/poems/Shakespeare/sonnet6
../gardnnic/poems/Shakespeare/sonnet6
../mohanchi/poems/Shakespeare/sonnet6
../whitfbob/poems/Shakespeare/sonnet6
../crivejoh/poems/Shakespeare/sonnet6
[simben@opus ~]$
```

Note:

***..** is a relative pathname to the parent directory. This is where the find command will start searching from.*

***-name "sonnet6"** is an option passed to the find command directing it to only look for files named "sonnet6"*

***2> /dev/null** redirects stderr to the "bit bucket" which discards any permission errors*

find command

Find all directories here in my home directory and down

```
[simben@opus ~]$ find . -type d
.
./.mozilla
./.mozilla/extensions
./.mozilla/plugins
./bin
./Hidden
find: ./Hidden: Permission denied
./poems
./poems/Blake
./poems/Shakespeare
./poems/Yeats
./poems/Anon
./olddir
./newdir
./edits
./docs
./etc
./class
./class/labs
./class/exams
./misc
[simben@opus ~]$
```

Note:

***.** is a relative pathname to "here". This is where the find command will start searching from.*

***-type d** is an option passed to the find command directing it to only look for directories*

find command

Task: Find all directories, starting here in my home directory, that start with a capital B, S, Y or A.

start from "here"

specifies directories only

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

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

Task: Find all files starting your current location that contain town

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

find command

Task: Find all ordinary files, starting in the /home directory, containing the word bones.

do not descend into directories on other file systems

ordinary files only. Other types are l (symbolic link), d (directory)

```
$ find /home -mount -type f -exec grep -l "bones" {} \; 2> /dev/null
/home/cis90/simben/stash
$
```

execute this command on what files are found

Input and Output
Miscellaneous Commands

find – Find file or content of a file

grep – "Global Regular Expression Print"

sort - sort
spell – spelling correction
wc – word count

*The **grep** command is used to look for content inside of files*

grep command

Task: Find the word love in Shakespeare's sonnets

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

Looking for love in all the wrong places?

grep command

Task: Find all lines with love and hate

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

grep command

Task: Find simmsben in /etc/passwd

```
[simben@opus poems]$ grep simmsben /etc/passwd
simmsben:x:1160:103:Benji Simms:/home/cis90/simmsben:/bin/bash
```

Task: Now show what line it is on

```
[simben@opus poems]$ grep -n simmsben /etc/passwd
53:simmsben:x:1160:103:Benji Simms:/home/cis90/simmsben:/bin/bash
```

grep command

Background
Apache is the worlds most popular web server and it's installed on Opus. Try it, you can browse to opus.cabrillo.edu.

Every Apache 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 files in Apache's document root directory on Opus?
  *Hint: Use the **ls** command on the document root directory*

Input and Output

Miscellaneous Commands

find – Find file or content of a file
grep – "Global Regular Expression Print"
sort – sort

spell – spelling correction

wc – word count

*The **spell** command is used to check spelling*

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
Daubeny
de
honour
kingdon
Pandulf
Poitou
Poppeley
seneschal
subdeacon
Warin
```

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

Task: Count the number of misspelled words

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

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

Class Exercise
Pipeline Tasks

Background
The cut command can cut a field out of a line of text where each field is delimitated 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
What does this command print? Why?

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

Wrap up

New commands:

find	find files or content
grep	look for text strings
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

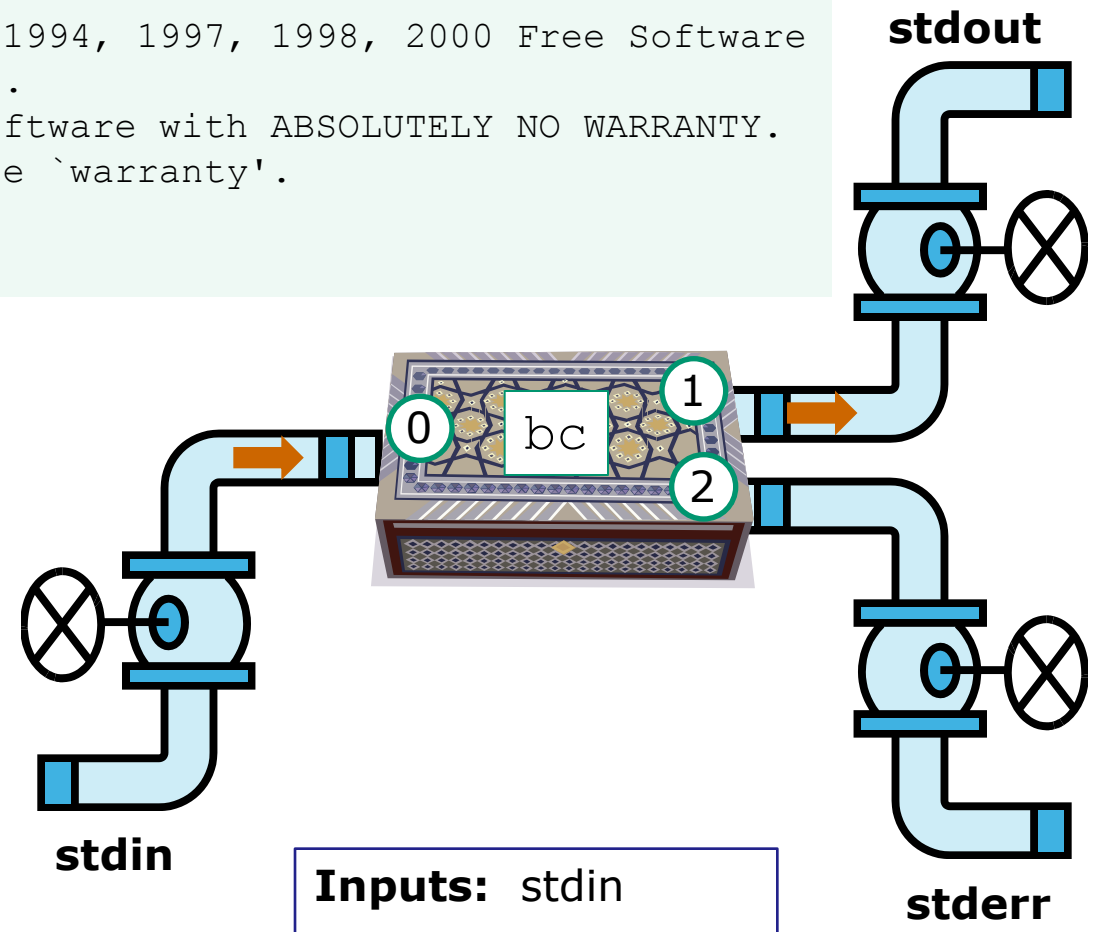
Backup

Example program to process: bc command

```
/home/cis90/simmsben $ bc
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software
Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
2+2
4
```

/dev/pts/1

```
bc 1.06
Copyright 1991-
1994, 1997,
1998, 2000 Free
Software
Foundation, Inc.
This is free
software with
ABSOLUTELY NO
WARRANTY.
For details type
`warranty'.
4
```



/dev/pts/1

2+2

Inputs: stdin

Outputs: stdout

Example program to process: ls command

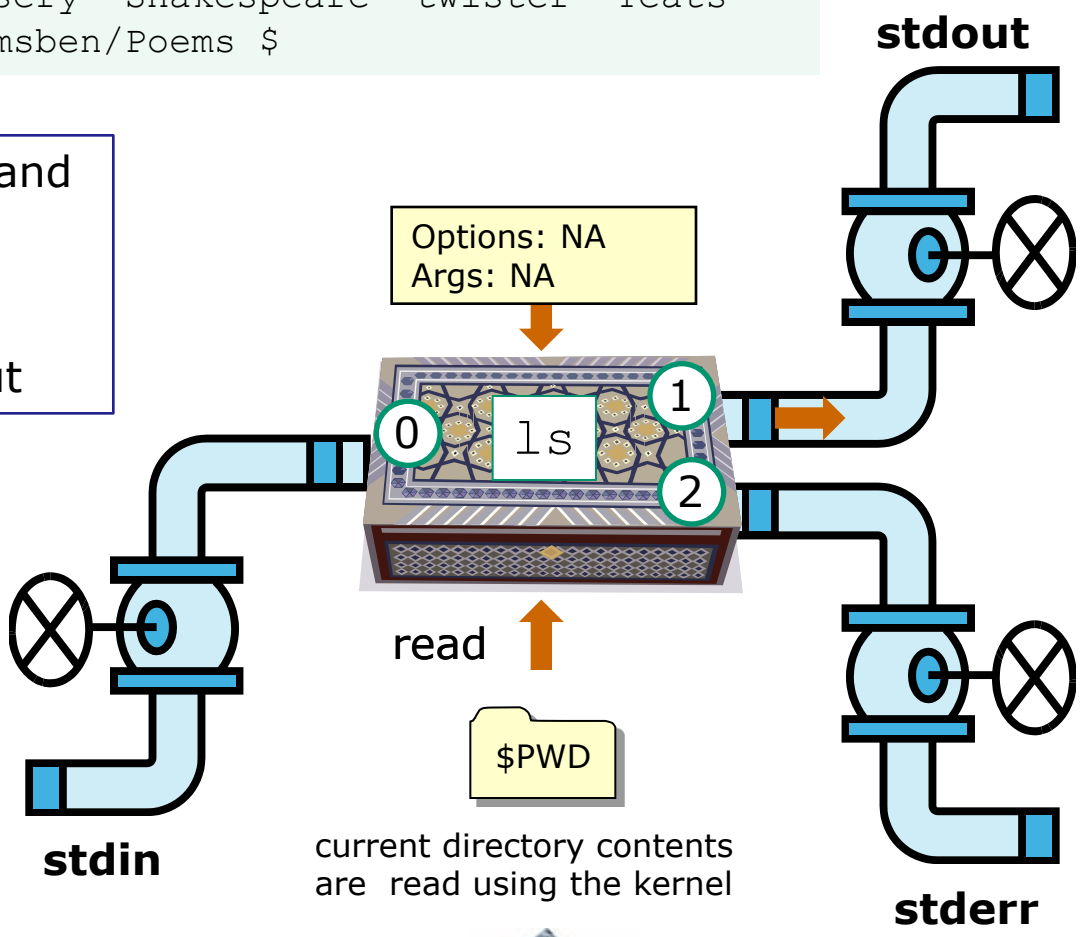
```
/home/cis90/simmsben/Poems $ ls
ant  Blake  nursery  Shakespeare  twister  Yeats
/home/cis90/simmsben/Poems $
```

/dev/pts/1

```
ant  Blake
nursery
Shakespeare
twister
Yeats
```

Inputs: Command line & Operating System

Outputs: stdout



File Permissions

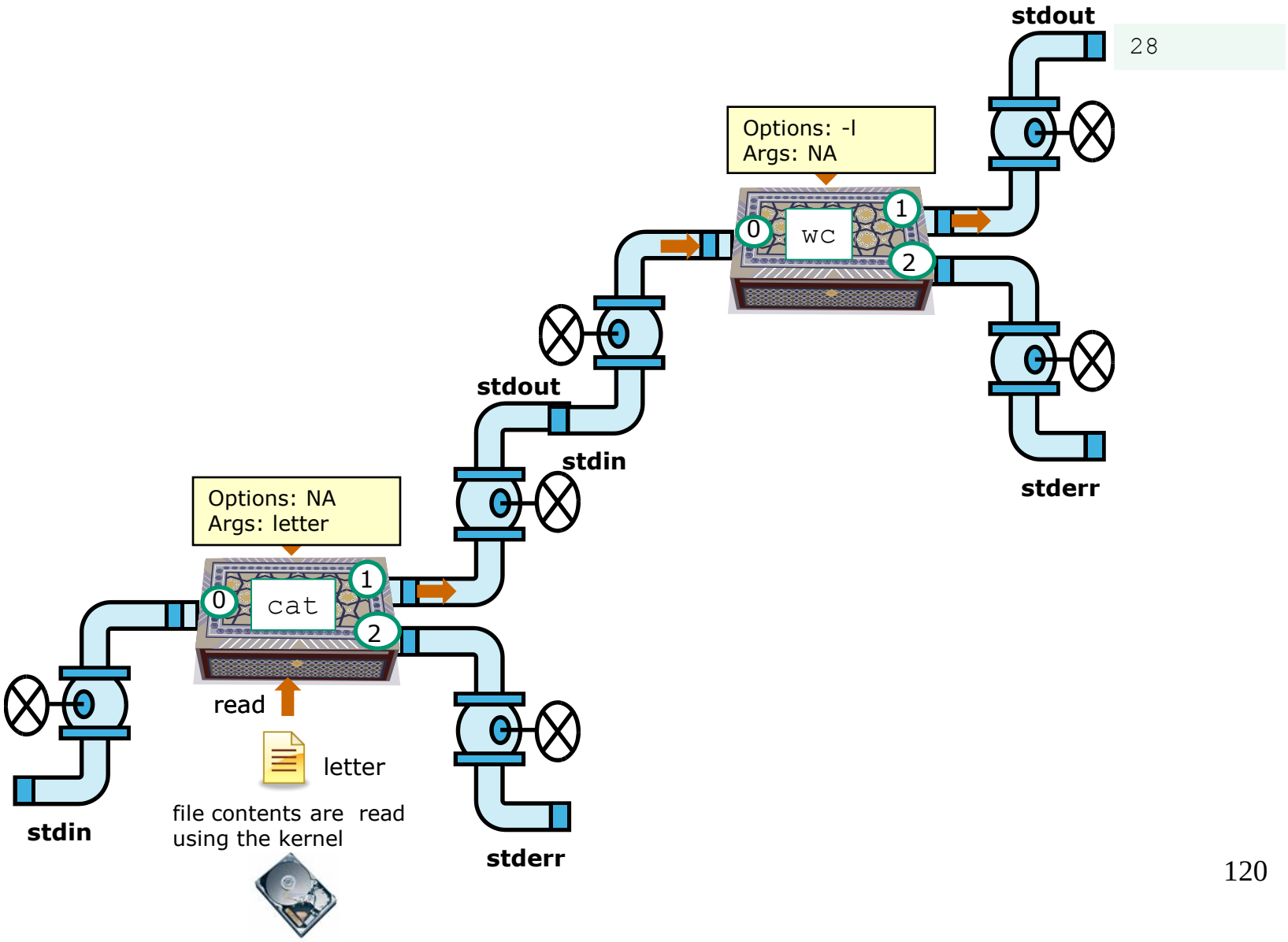
exercise

```
/home/cis90: drwxr-x---
/home/cis90/simmsben: drwxr-xr-x
/home/cis90/simmsben/Directory1: drwxr-x--x

file1: -rw-rw-r--
        owner           __modify        __delete        __read          __execute
        group           __modify        __delete        __read          __execute
        other           __modify        __delete        __read          __execute

file2: -rwxr-xr-x
        owner           __modify        __delete        __read          __execute
        group           __modify        __delete        __read          __execute
        other           __modify        __delete        __read          __execute

file3: -r-xr-xr--
        owner           __modify        __delete        __read          __execute
        group           __modify        __delete        __read          __execute
        other           __modify        __delete        __read          __execute

/home/cis90/simmsben/Directory2: drwxrwxr-x

file1: -rwxr-xr-x
        owner           __modify        __delete        __read          __execute
        group           __modify        __delete        __read          __execute
        other           __modify        __delete        __read          __execute
```

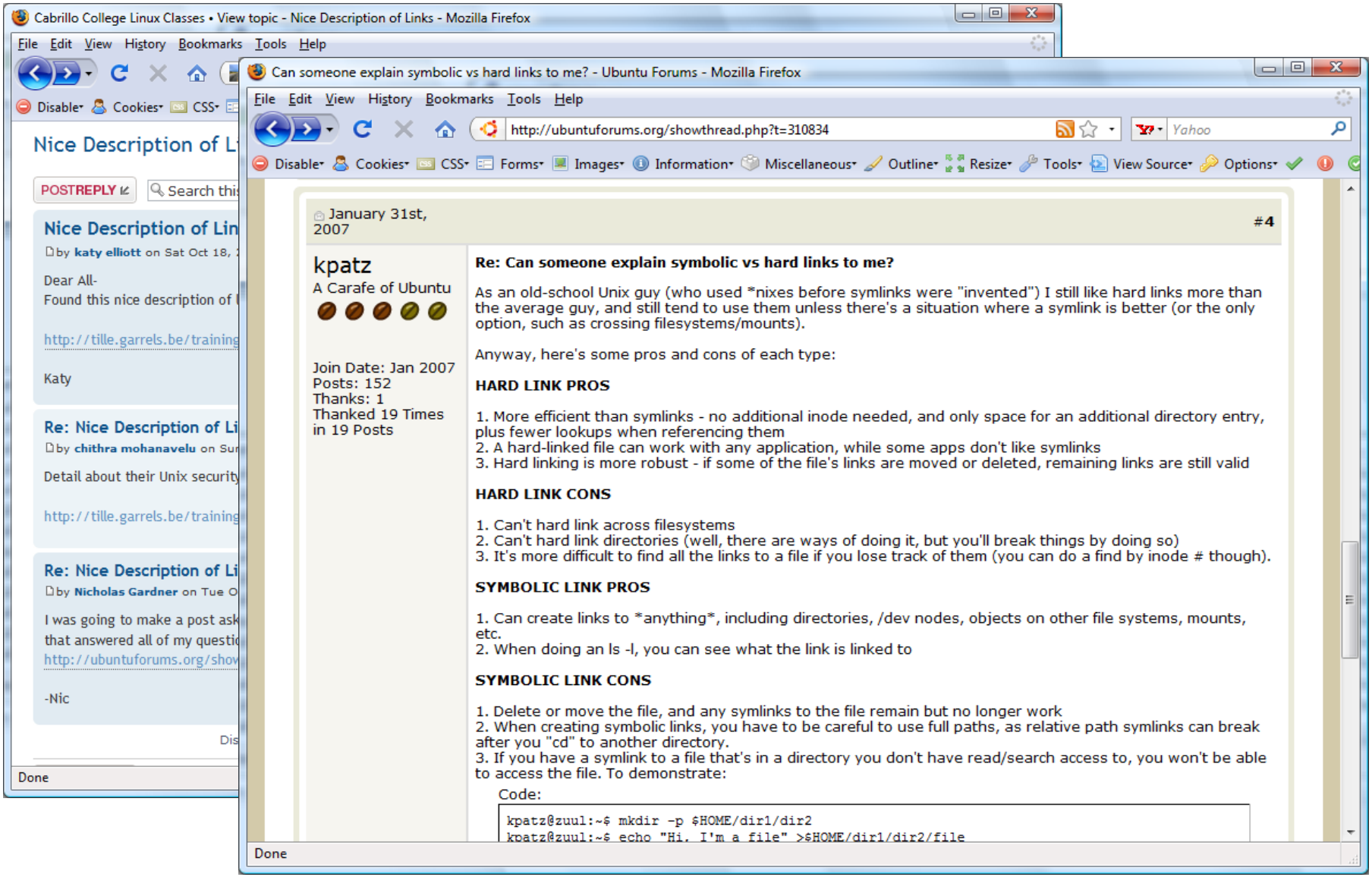
File Permissions

```
/home/cis90: drwxr-x---
/home/cis90/simmsben: drwxr-xr-x
/home/cis90/simmsben/Directory1: drwxr-x—x

file1: -rw-rw-r--
        owner     modify    delete    read    execute
        group     modify    delete    read    execute
        other     modify    delete    read    execute

file2: -rwxr-xr-x
        owner     modify    delete    read    execute
        group     modify    delete    read    execute
        other     modify    delete    read    execute

file3: -r-xr-xr--
        owner     modify    delete    read    execute
        group     modify    delete    read    execute
        other     modify    delete    read    execute

/home/cis90/simmsben/Directory2: drwxrwxr-x

file1: -rwxr-xr-x
        owner     modify    delete    read    execute
        group     modify    delete    read    execute
        other     modify    delete    read    execute
```

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



Hard and Soft Links Forum Posts



Sticky Bit

The Sticky Bit

```
[root@opus /]# chmod 777 temp777
[root@opus /]# chmod 1777 temp777S
[root@opus /]# ls -ld *
drwxr-xr-x   2 root root    4096 Jun 17 16:25 bin
drwxr-xr-x   3 root root    4096 Jun 17 15:00 boot
drwxr-xr-x  11 root root    3660 Sep 16 12:59 dev
drwxr-xr-x  98 root root   12288 Oct 21 04:02 etc
drwxr-xr-x  16 root root    4096 Jun 20 11:07 home
drwxr-xr-x  14 root root    4096 Jun 17 16:22 lib
drwx------   2 root root   16384 Jun 16 08:35 lost+found
drwxr-xr-x   2 root root    4096 Jun 17 15:10 media
drwxr-xr-x   2 root root       0 Sep 10 21:48 misc
drwxr-xr-x   2 root root    4096 Oct 10  2006 mnt
drwxr-xr-x   2 root root    4096 Oct 10  2006 opt
dr-xr-xr-x 123 root root       0 Sep 10 14:48 proc
drwxr-x---  21 root root    4096 Sep 17 17:25 root
drwxr-xr-x   2 root root   12288 Jun 17 16:25 sbin
drwxrwxrwx   2 root root    4096 Oct 22 14:04 temp777
drwxrwxrwt   2 root root    4096 Oct 22 13:59 temp777S
drwxrwxrwt   8 root root    4096 Oct 22 13:52 tmp
drwxr-xr-x  14 root root    4096 Jun 16 15:38 usr
drwxr-xr-x  26 root root    4096 Jun 17 22:16 var
```

A closer look at the /tmp directory

A closer look at the /tmp directory

Sticky Bit

```
[root@opus /]# ls -ld t* bin etc
drwxr-xr-x  2 root root  4096 Jun 17 16:25 bin
drwxr-xr-x 98 root root 12288 Oct 21 04:02 etc
drwxrwxrwx  2 root root  4096 Oct 22 14:21 temp777
drwxrwxrwt  2 root root  4096 Oct 22 13:59 temp777S
drwxrwxrwt  8 root root  4096 Oct 22 13:52 tmp
[root@opus /]#
```

The other directories in / are set to 755 permission. The /tmp is 777 so anyone can view, create and remove files there

```
[simben@opus simmsric]$ cd /temp777
[simben@opus temp777]$ touch duke
[simben@opus temp777]$ echo hi > benji
[simben@opus temp777]$ rm benji
[simben@opus temp777]$
```

sticky bit not set

Without the sticky bit set, one user can delete files belonging to another.

```
[simmsben@opus simmsric]$ cd /temp777
[simmsben@opus temp777]$ touch benji
[simmsben@opus temp777]$ echo hi > duke
[simmsben@opus temp777]$ rm duke
[simmsben@opus temp777]$
```

A closer look at the /tmp directory

Sticky Bit

```
[root@opus /]# ls -ld t* bin etc
drwxr-xr-x  2 root root  4096 Jun 17 16:25 bin
drwxr-xr-x 98 root root 12288 Oct 21 04:02 etc
drwxrwxrwx  2 root root  4096 Oct 22 14:21 temp777
drwxrwxrwt  2 root root  4096 Oct 22 13:59 temp777S
drwxrwxrwt  8 root root  4096 Oct 22 13:52 tmp
[root@opus /]#
```

The other directories in / are set to 755 permission. The /tmp is 777 so anyone can view, create and remove files there

```
[simben@opus temp777S]$ touch duke
[simben@opus temp777S]$ echo hi > benji
[simben@opus temp777S]$ rm benji
rm: cannot remove `benji': Operation not permitted
[simben@opus temp777S]$ rm duke
[simben@opus temp777S]$
```

sticky bit set

With the sticky bit set, a user can delete there own files but not those belonging to another.

```
[simmsben@opus temp777S]$ touch benji
[simmsben@opus temp777S]$ echo hi > duke
[simmsben@opus temp777S]$ rm duke
rm: cannot remove `duke': Operation not permitted
[simmsben@opus temp777S]$ rm benji
[simmsben@opus temp777S]$
```

```
-rw-rw-r-- 1 simmsben cis90 3 Oct 22 14:27 benji
-rw-rw-r-- 1 simben cis90 3 Oct 22 14:26 duke
```

Directory Write Permission

r	w	x
read	write	execute

user	group	others

```
[simmsben@opus ~]$ cd examples/dogs/
[simmsben@opus dogs]$ cp duke duke.bak
[simmsben@opus dogs]$ mv homer Homer
[simmsben@opus dogs]$ rm duke
[simmsben@opus dogs]$ ln benji mydog
[simmsben@opus dogs]$ ls -li
total 32
104704 -rw-r--r-- 2 simmsben cis90 20 Oct 20 08:27 benji
104743 -rw-r--r-- 1 simmsben cis90 20 Oct 20 09:24 duke.bak
104684 -rw-r--r-- 1 simmsben cis90 20 Oct 20 08:27 Homer
104704 -rw-r--r-- 2 simmsben cis90 20 Oct 20 08:27 mydog
[simmsben@opus ~]$ chmod u-w examples/dogs/
[simmsben@opus ~]$ cd examples/dogs/
[simmsben@opus dogs]$ cp duke.bak /tmp
[simmsben@opus dogs]$ cp duke.bak duke
cp: cannot create regular file `duke': Permission denied
[simmsben@opus dogs]$ mv duke.bak duke
mv: cannot move `duke.bak' to `duke': Permission denied
[simmsben@opus dogs]$ rm duke.bak
rm: cannot remove `duke.bak': Permission denied
[simmsben@opus dogs]$ ln duke.bak /tmp/mydog
ln: creating hard link `/tmp/mydog' to `duke.bak': Invalid cross-device link
[simmsben@opus dogs]$ ln Homer herdog
ln: creating hard link `herdog' to `Homer': Permission denied
```

All is well *when the directory has write permission*

All is ***not well*** *without write permission … why?*
Because filenames are stored in a directory. cp, mv, rm and ln commands need to change filenames, therefore they need write access to the directory

Removing directory w permission
- ***cannot*** *cp files into it, can't remove files, can't move files out, can't add links*
- *but you* ***can*** *cp files out*

Directory Execute Permission

r	w	x	r	w	x	r	w	x
read	write	execute	read	write	execute	read	write	execute
user			group			others		

Benji removes x permission on his dogs directory

```
[simmsben@opus ~]$ chmod g-x examples/dogs/
[simmsben@opus ~]$ ls -ld examples/
drwxrwxr-x 4 simmsben cis90 4096 Oct 20 08:27 examples/
[simmsben@opus ~]$ ls -lR examples/
examples/:
total 40
-rw-r--r-- 1 simmsben cis90  237 Oct 20 08:27 ant
drwxr-xr-x 2 simmsben cis90 4096 Oct 20 08:27 birds
drwxr--r-x 2 simmsben cis90 4096 Oct 20 08:27 dogs
-rw-r--r-- 1 simmsben cis90  779 Oct 20 08:27 nursery
-rw-r--r-- 1 simmsben cis90  151 Oct 20 08:27 twister

examples/birds:
total 16
-rw-r--r-- 1 simmsben cis90 24 Oct 20 08:27 abby
-rw-r--r-- 1 simmsben cis90 24 Oct 20 08:27 nibbie

examples/dogs:
total 24
-rw-r--r-- 1 simmsben cis90 20 Oct 20 08:27 benji
-rw-r--r-- 1 simmsben cis90 20 Oct 20 08:27 duke
-rw-r--r-- 1 simmsben cis90 20 Oct 20 08:27 homer
[simmsben@opus ~]$
```

```
[roddyduk@opus ~]$ ls -ld ../simmsben/examples/
drwxrwxr-x 4 simmsben cis90 4096 Oct 20 08:27 ../simmsben/exa
[roddyduk@opus ~]$ ls -lR ../simmsben/examples/
../simmsben/examples/:
total 40
-rw-r--r-- 1 simmsben cis90  237 Oct 20 08:27 ant
drwxr-xr-x 2 simmsben cis90 4096 Oct 20 08:27 birds
drwxr--r-x 2 simmsben cis90 4096 Oct 20 08:27 dogs
-rw-r--r-- 1 simmsben cis90  779 Oct 20 08:27 nursery
-rw-r--r-- 1 simmsben cis90  151 Oct 20 08:27 twister

../simmsben/examples/birds:
total 16
-rw-r--r-- 1 simmsben cis90 24 Oct 20 08:27 abby
-rw-r--r-- 1 simmsben cis90 24 Oct 20 08:27 nibbie

../simmsben/examples/dogs:
total 0
?--------- ? ? ? ?            ? benji
?--------- ? ? ? ?            ? duke
?--------- ? ? ? ?            ? homer
[roddyduk@opus ~]$
```


Now Duke can see filenames but no inode information

Directory Execute Permission

r	w	x	r	w	x	r	w	x
read	write	execute	read	write	execute	read	write	execute
user			group			others		

Benji removes x permission on his dogs directory

```
[simmsben@opus ~]$ chmod g-x examples/dogs/
[simmsben@opus ~]$ ls -ld examples/
drwxrwxr-x 4 simmsben cis90 4096 Oct 20 08:27 examples/
[simmsben@opus ~]$ ls -lR examples/
examples/:
total 40
-rw-r--r-- 1 simmsben cis90  237 Oct 20 08:27 ant
drwxr-xr-x 2 simmsben cis90 4096 Oct 20 08:27 birds
drwxr--r-x 2 simmsben cis90 4096 Oct 20 08:27 dogs
-rw-r--r-- 1 simmsben cis90  779 Oct 20 08:27 nursery
-rw-r--r-- 1 simmsben cis90  151 Oct 20 08:27 twister

examples/birds:
total 16
-rw-r--r-- 1 simmsben cis90 24 Oct 20 08:27 abby
-rw-r--r-- 1 simmsben cis90 24 Oct 20 08:27 nibbie

examples/dogs:
total 24
-rw-r--r-- 1 simmsben cis90 20 Oct 20 08:27 benji
-rw-r--r-- 1 simmsben cis90 20 Oct 20 08:27 duke
-rw-r--r-- 1 simmsben cis90 20 Oct 20 08:27 homer
[simmsben@opus ~]$
```

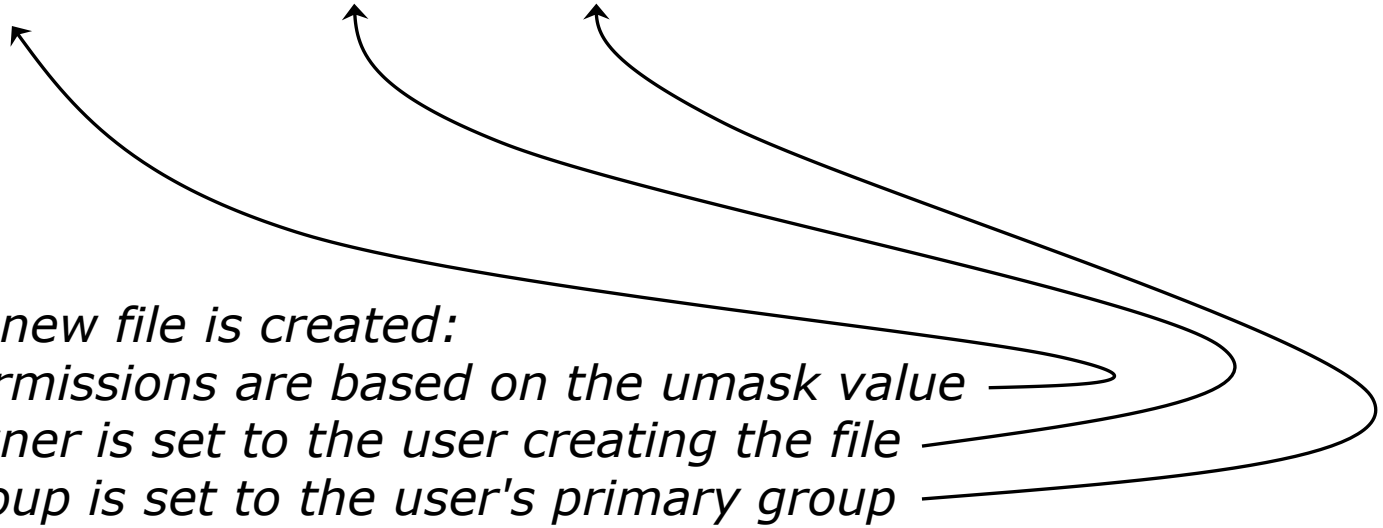
```
[roddyduk@opus ~]$ cd ../simmsben
[roddyduk@opus simmsben]$ cd examples/
[roddyduk@opus examples]$ cd birds
[roddyduk@opus birds]$ cd ..
[roddyduk@opus examples]$ cd dogs/
-bash: cd: dogs/: Permission denied
[roddyduk@opus examples]$
[roddyduk@opus examples]$ cat dogs/duke
cat: dogs/duke: Permission denied
[roddyduk@opus examples]$
```

*Duke **cannot cd** into the directory and he **cannot retrieve any file data** for the files in the directory*

Groups and new files

Groups

```
/home/cis90/roddyduk $ touch mydogs
/home/cis90/roddyduk $ ls -l mydogs
-rw-rw-r-- 1 roddyduk cis90 0 Oct 19 13:16 mydogs
```



When a new file is created:

- *the permissions are based on the umask value*
- *the owner is set to the user creating the file*
- *the group is set to the user's primary group*

Groups

*Use either **id** or **groups** command to determine what groups a user belongs to*

```
[rsimms@opus lab06]$ id roddyduk
uid=1201(roddyduk) gid=90(cis90) groups=90(cis90),100(users)
context=user_u:system_r:unconfined_t
[rsimms@opus lab06]$

[roddyduk@opus ~]$ groups roddyduk
roddyduk : cis90 users
```

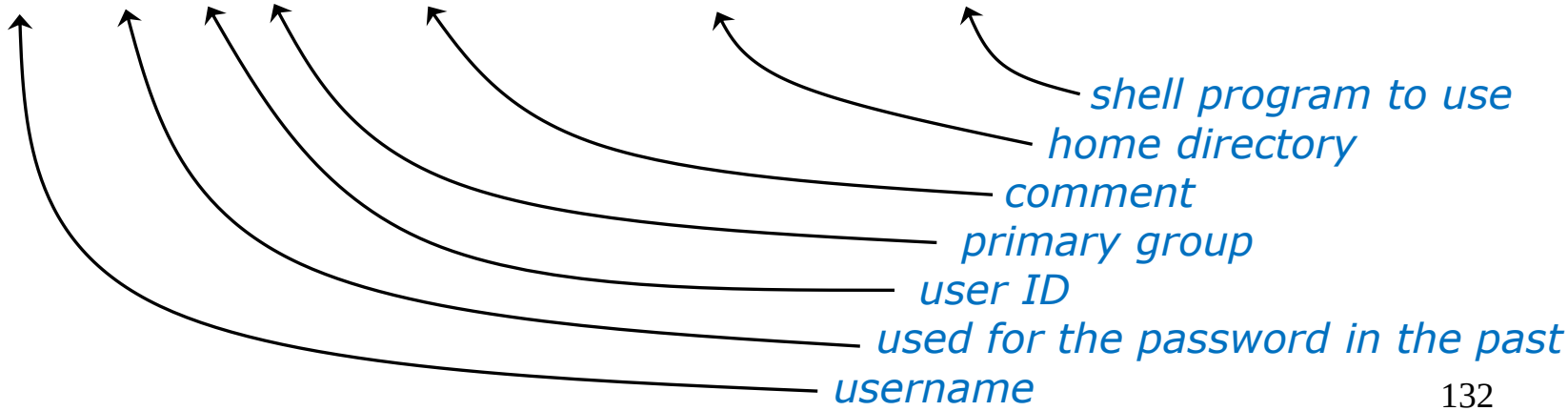
Primary group (gid) is cis90, secondary group is users

Groups

The user's primary group is stored in /etc/passwd (the 4th field)

Excerpt from **/etc/passwd**

```
simmsben:x:1200:90:Benji Simms:/home/cis90/simmsben:/bin/bash
roddyduk:x:1201:90:Duke Roddy:/home/cis90/roddyduk:/bin/bash
clastmax:x:1009:191:Max Clastor:/home/cis191/clastmax:/bin/bash
derriale:x:1202:90:Alex Derrick:/home/cis90/derriale:/bin/bash
garciton:x:1203:90:Tony Garcia:/home/cis90/garciton:/bin/bash
garibjam:x:1204:90:James Garibay:/home/cis90/garibjam:/bin/bash
rochajua:x:1205:90:Juan Rocha:/home/cis90/rochajua:/bin/bash
delfimik:x:1206:90:Mike Delfin:/home/cis90/delfimik:/bin/bash
dingechr:x:1207:90:Christine Dinges:/home/cis90/dingechr:/bin/bash
blacksea:x:1208:90:Sean Black:/home/cis90/blacksea:/bin/bash
antiden:x:1209:90:Dennis Anti:/home/cis90/antiden:/bin/bash
```



Groups

Secondary groups are recorded in /etc/group

Excerpts from **/etc/group**

```
users:x:100:guest,guest90,jimg,abbenste,arltjef,bolasale,bowerjak,dycktim,farreeli,ga
virxim,gilart,gonzaian,goodmthe,hammoste,kotilnat,lenzpat,maganfra,mattimar,mccarmic,
mchalgeo,mezalui,ortegcar,rochaleo,spadymat,starkmic,vasqucar,vistigab,wallgle,watsoh
ar,quintjos,swansgre,archiand,moonecar,orourpat,pantogab,velasoli,simmsben,roddyduk,c
lastmax,derriale,garciton,garibjam,rochajua,delfimik,dingechr,blacksea,antiden,pirkll
au,birmijam,messison,zilissau,plastadr,brownliz,husemat,botoschr,perezrud,palmilar,sa
linjac,hamiljas,pennitan,valadand,woodjan,henrydal,galbrnat,dakkaabd,cardefra,daviesa
r,hrdinste,redmanic,enriqste,dawadast,menafer,orozcmig,srecklau,mottste,fouric,wattsl
uk,dahlicas,velasliv,pitzemik,komicser,parrijen,beltredt,hernaaar,brownbri,castrsal,m
artiant,joossam,ojedavic,millehom,alvesdes,bejarjoh,bergejoh,breitrob,clarkgal,desotm
at,gardnnic,huangyan,leetheri,lewisgre,studetes,lighttho,lindadav,madrista,normasea,p
oncimar,rochever,schreche,schwajoe,tatlojas,velasjos,lukewat,mikedel,seanbla,veracroc
,simmsmar
utmp:x:22:
utempter:x:35:
< snipped >
mikki:x:501:
guest:x:506:
staff:x:503:jimg,rsimms,gerlinde
cis90:x:90:jimg,guest,rsimms
cis130:x:130:jimg,rsimms
```

Groups

Every user is a member of a **primary group** (shown in /etc/passwd) and multiple **secondary groups** (shown in /etc/group)

```
/home/cis90/roddyduk $ groups roddyduk
roddyduk : cis90 users
```

primary

secondary

```
[roddyduk@opus ~]$ id
uid=1201(roddyduk) gid=90(cis90) groups=90(cis90),100(users)
context=user_u:system_r:unconfined_t
```

```
/home/cis90/roddyduk $ touch mydogs
/home/cis90/roddyduk $ ls -l mydogs
-rw-rw-r-- 1 roddyduk cis90 0 Oct 19 13:16 mydogs
```

Note, new files are created using the primary group

Permissions Review

Permissions - Review

r	w	x
read	write	execute

r	w	x
read	write	execute

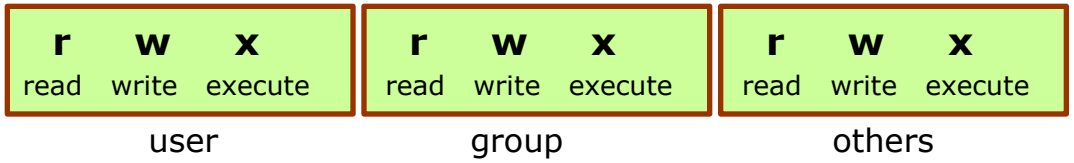
r	w	x
read	write	execute

user group others

```
[rsimms@opus cis90]$ ls -l examples/
total 40
-rw-r--r-- 1 rsimms users  237 Oct 20 07:15 ant
drwxr-xr-x 2 rsimms users 4096 Oct 20 07:16 birds
drwxr-xr-x 2 rsimms users 4096 Oct 20 07:34 dogs
-rw-r--r-- 1 rsimms users  779 Oct 20 07:15 nursery
-rw-r--r-- 1 rsimms users  151 Oct 20 07:16 twister
```

You should now be able to interpret the permissions, user and groups you see on long listings

Permissions - Review



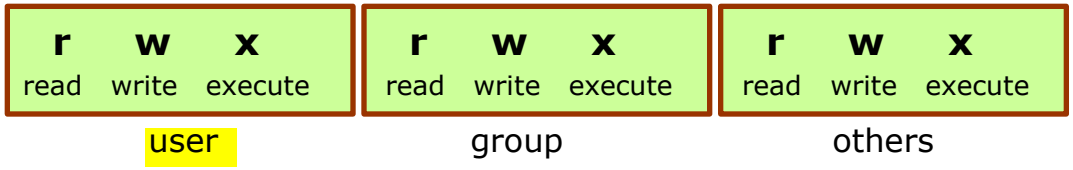
rwx	Binary	Convert	Decimal
_ _ _	0 0 0	0 + 0 + 0	0
_ _ x	0 0 1	0 + 0 + 1	1
_ w _	0 1 0	0 + 2 + 0	2
_ w x	0 1 1	0 + 2 + 1	3
r _ _	1 0 0	4 + 0 + 0	4
r _ x	1 0 1	4 + 0 + 1	5
r w _	1 1 0	4 + 2 + 0	6
r w x	1 1 1	4 + 2 + 1	7

4's column
2's column
1's column

And be able to count in binary

user

Permissions - Practice

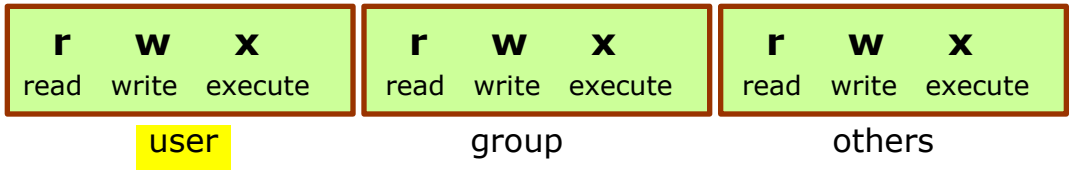



```
/home/cis90/simmsben $ ls -l myfile
-rw-r-x--- 1 simmsben cis90 0 Oct 19 07:12 myfile
/home/cis90/simmsben $
```

r w -

What is this permission in binary?

Permissions - Practice




```
/home/cis90/simmsben $ ls -l myfile
-rw-r-x--- 1 simmsben cis90 0 Oct 19 07:12 myfile
/home/cis90/simmsben $
```

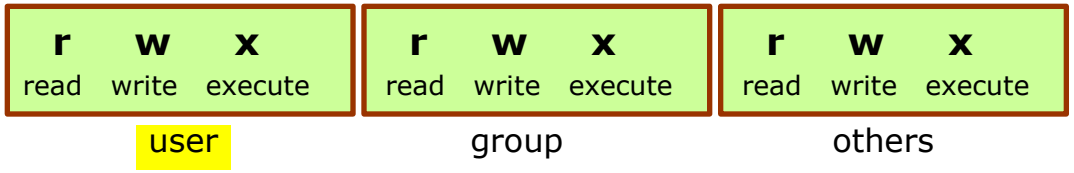
Binary number

r w - = 1 1 0

4's column
2's column
1's column

Now, what is this permission in decimal?

Permissions - Practice




```
/home/cis90/simmsben $ ls -l myfile
-rw-r-x--- 1 simmsben cis90 0 Oct 19 07:12 myfile
/home/cis90/simmsben $
```

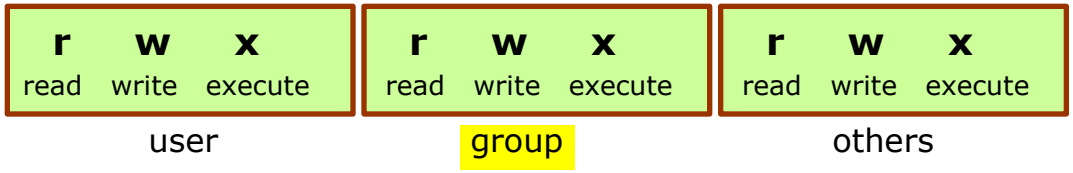
Binary number *Decimal number*

r w - = 1 1 0 = 6 *user is simmsben*

4's column
2's column
1's column

group

Permissions - Practice

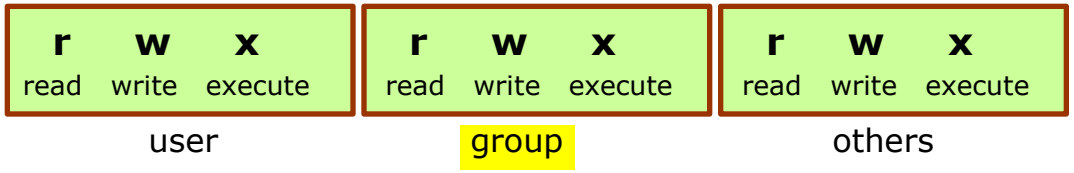



```
/home/cis90/simmsben $ ls -l myfile
-rw-r-x--- 1 simmsben cis90 0 Oct 19 07:12 myfile
/home/cis90/simmsben $
```

r - x

What is this permission in binary?

Permissions - Practice




```
/home/cis90/simmsben $ ls -l myfile
-rw-r-x--- 1 simmsben cis90 0 Oct 19 07:12 myfile
/home/cis90/simmsben $
```

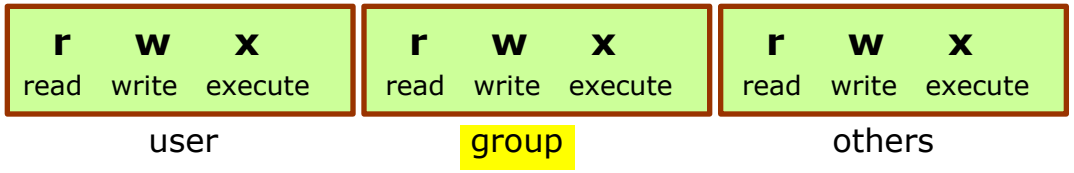
Binary number

r - x = 1 0 1

4's column
2's column
1's column

Now, what is this permission in decimal?

Permissions - Practice




```
/home/cis90/simmsben $ ls -l myfile
-rw-r-x--- 1 simmsben cis90 0 Oct 19 07:12 myfile
/home/cis90/simmsben $
```

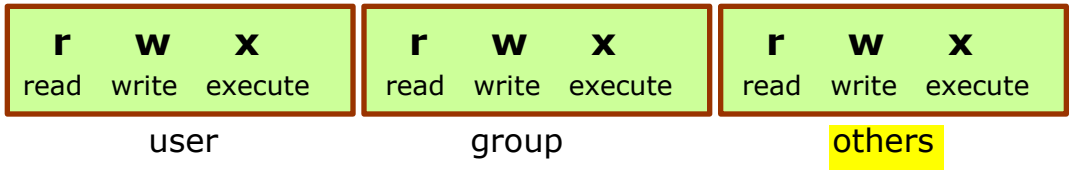
Binary number *Decimal number*

r - x = 1 0 1 = 5 *Group is cis90*

4's column
2's column
1's column

others

Permissions - Practice

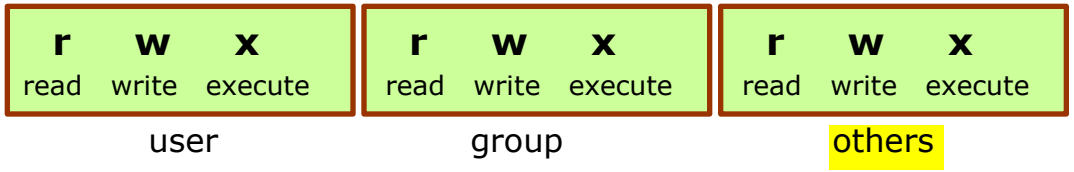



```
/home/cis90/simmsben $ ls -l myfile
-rw-r-x--- 1 simmsben cis90 0 Oct 19 07:12 myfile
/home/cis90/simmsben $
```

\- \- \-

What is this permission in binary?

Permissions - Practice




```
/home/cis90/simmsben $ ls -l myfile
-rw-r-x--- 1 simmsben cis90 0 Oct 19 07:12 myfile
/home/cis90/simmsben $
```

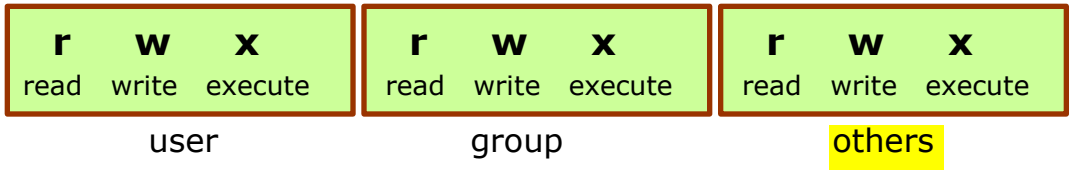
Binary number

- - - = 0 0 0

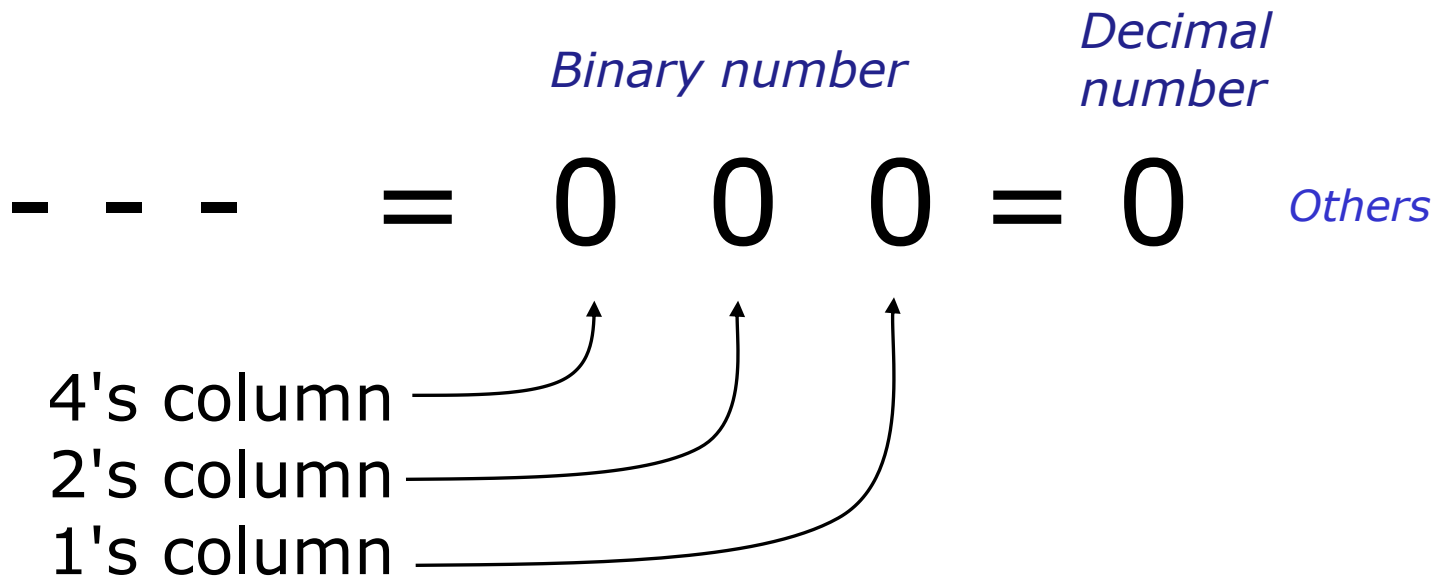
Now, what is this permission in decimal?

4's column
2's column
1's column

Permissions - Practice

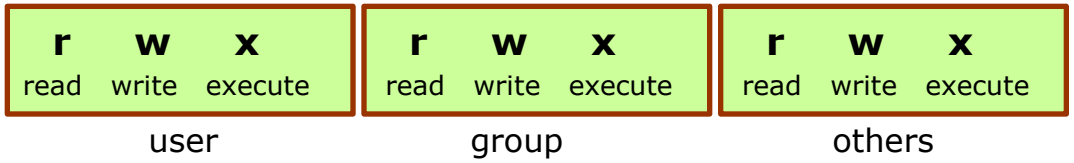



```
/home/cis90/simmsben $ ls -l myfile
-rw-r-x--- 1 simmsben cis90 0 Oct 19 07:12 myfile
/home/cis90/simmsben $
```



all

Permissions - Practice




```
/home/cis90/simmsben $ ls -l myfile
-rw-r-x--- 1 simmsben cis90 0 Oct 19 07:12 myfile
```

		Binary number				Decimal number	
r w -	=	1	1	0	=	6	*User is simmsben*
r - x	=	1	0	1	=	5	*Group is cis90*
- - -	=	0	0	0	=	0	*Others*

Permissions Activity

Task: Modify the permissions of the terminal device you are logged in as so the guest90 user has write permission.

Hint: What command shows you the terminal device you are using?
Hint: How do you do a long listing on terminal devices?

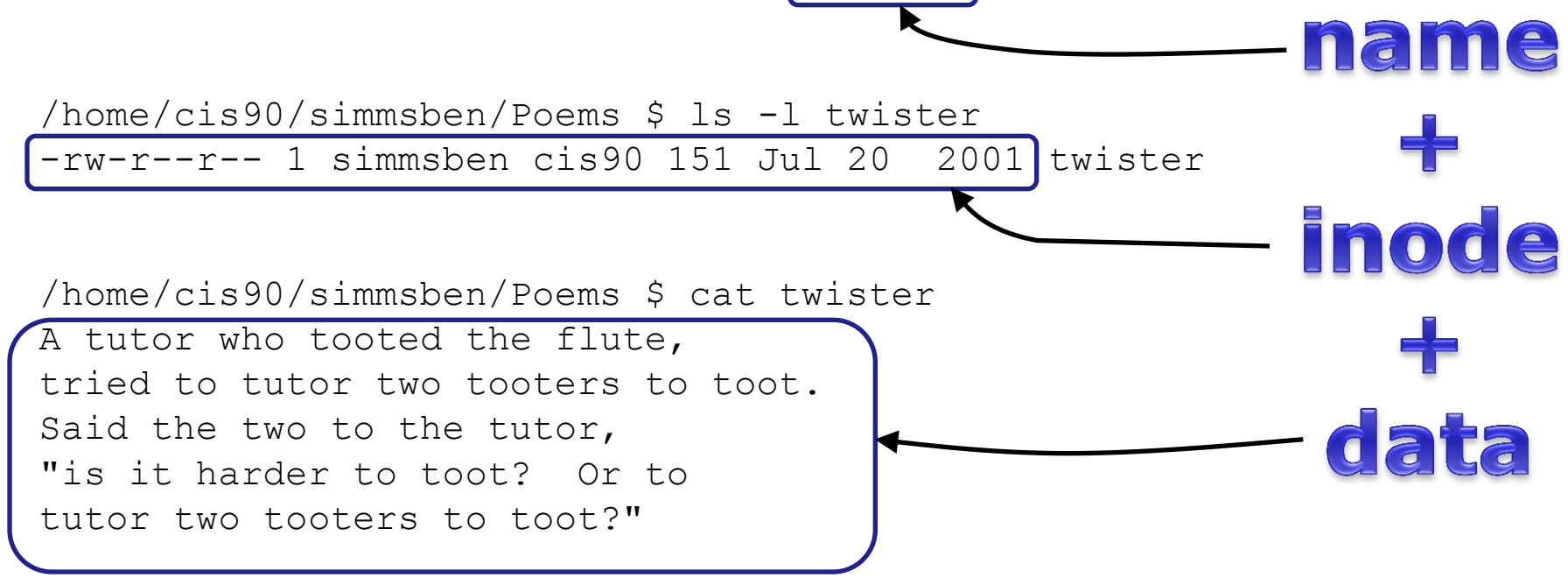
In another Putty session, login as guest90 and write a message to your first session using this command:

banner I did it! > /dev/pts/*xx*
(where ***xx*** is your terminal device)

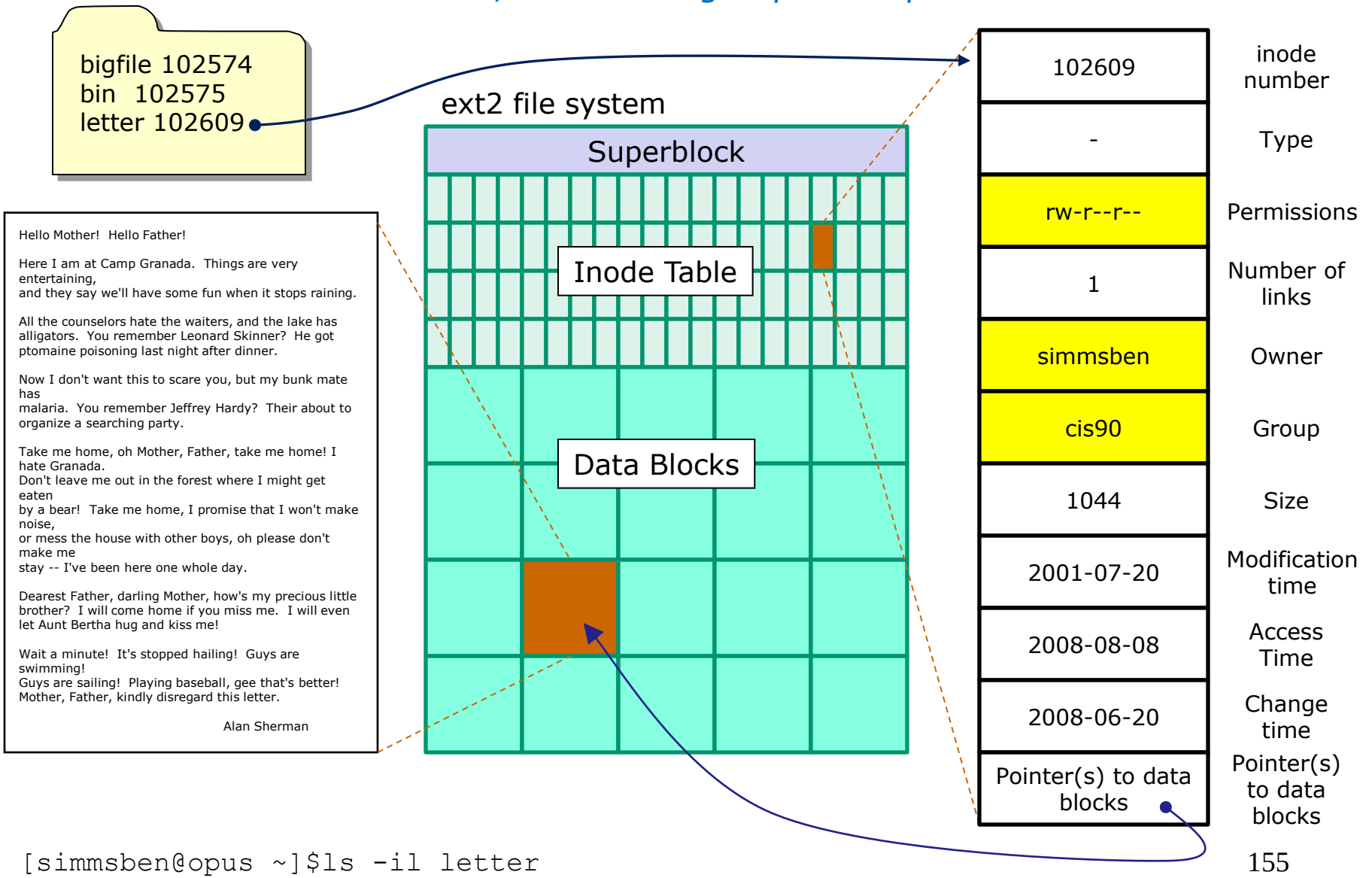
Directory permissions

UNIX Files
The three elements of a file

```
/home/cis90/simmsben/Poems $ ls
ant  Blake  nursery  Shakespeare  twister  Yeats

/home/cis90/simmsben/Poems $ ls -l twister
-rw-r--r-- 1 simmsben cis90 151 Jul 20  2001 twister

/home/cis90/simmsben/Poems $ cat twister
A tutor who tooted the flute,
tried to tutor two tooters to toot.
Said the two to the tutor,
"is it harder to toot?  Or to
tutor two tooters to toot?"
```



Permissions, owner and group are kept in the inode of a file

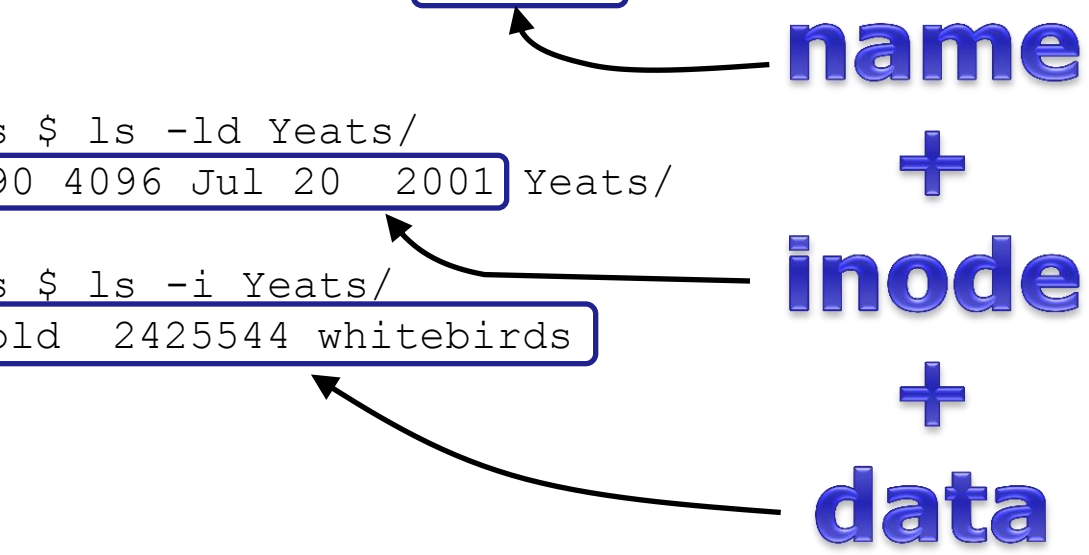


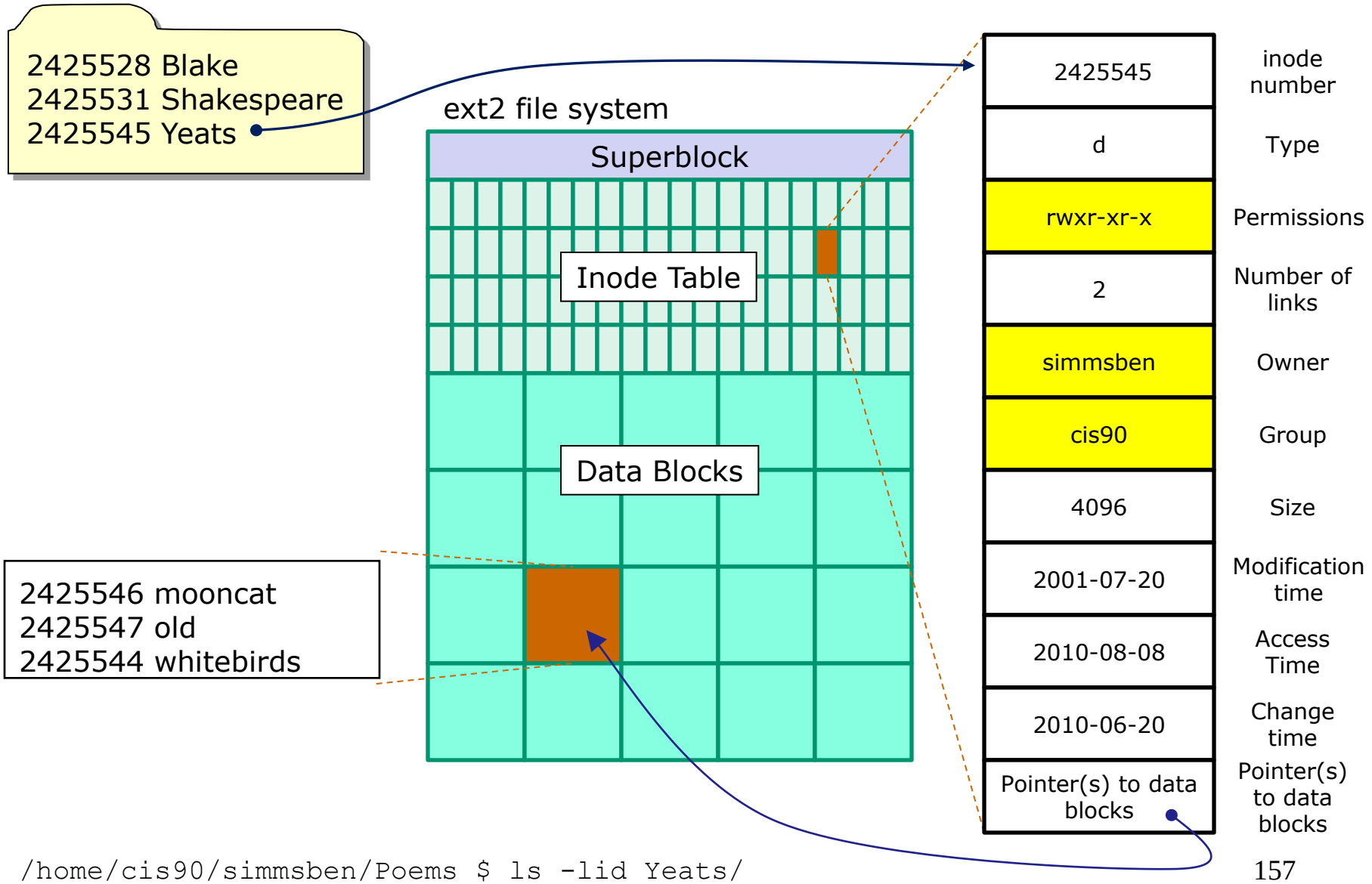

```
[simmsben@opus ~]$ls -il letter
102609 -rw-r--r-- 1 simmsben cis90 1044 Jul 20  2001 letter
```

UNIX Files
The three elements of a file

Directors are files as well. The data portion of a directory contains filename/inode pairs

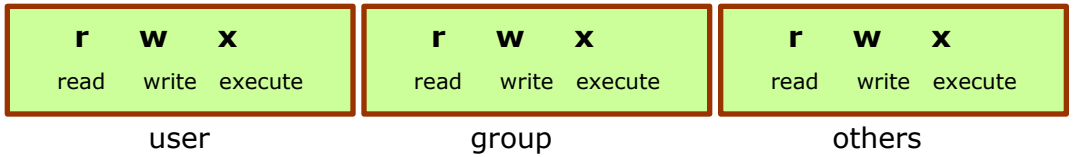
```
/home/cis90/simmsben/Poems $ ls
ant   Blake   nursery   Shakespeare   twister   Yeats

/home/cis90/simmsben/Poems $ ls -ld Yeats/
drwxr-xr-x 2 simmsben cis90 4096 Jul 20   2001 Yeats/

/home/cis90/simmsben/Poems $ ls -i Yeats/
2425546 mooncat   2425547 old   2425544 whitebirds
```






```
/home/cis90/simmsben/Poems $ ls -lid Yeats/
2425545 drwxr-xr-x 2 simmsben cis90 4096 Jul 20  2001 Yeats/
```

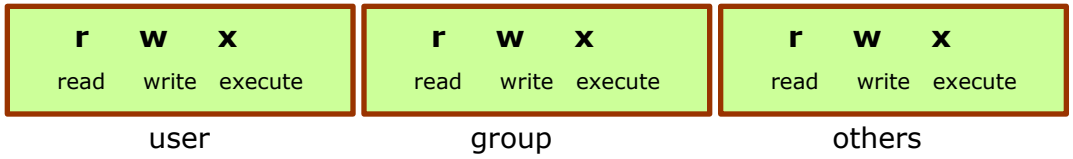
Directory Read Permission



Setup for the next examples:

```
/home/cis90/roddyduk $ mkdir examples
/home/cis90/roddyduk $ cd examples/
/home/cis90/roddyduk/examples $ mkdir birds dogs
/home/cis90/roddyduk/examples $ cd birds
/home/cis90/roddyduk/examples/birds $ echo "Tweet tweet" > abby
/home/cis90/roddyduk/examples/birds $ echo "Tweet tweet" > nibbie
/home/cis90/roddyduk/examples/birds $ cd ../dogs
/home/cis90/roddyduk/examples/dogs $ echo "Woof woof" > benji
/home/cis90/roddyduk/examples/dogs $ echo "Woof woof" > duke
/home/cis90/roddyduk/examples/dogs $ echo "Woof woof" > homer
```

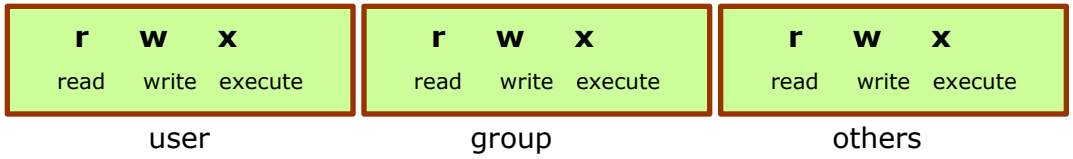
Directory Read Permission



Tree view of examples directory:

```
/home/cis90/roddyduk $ tree examples
examples
|-- birds
|   |-- abby
|   `-- nibbie
`-- dogs
    |-- benji
    |-- duke
    `-- homer

2 directories, 5 files
```

Directory Read Permission



Long listing showing directory and contents:

```
/home/cis90/roddyduk $ ls -ld examples/
drwxrwxr-x 5 roddyduk cis90 4096 Oct 19 13:49 examples/
```

The directory itself (use the -d option)

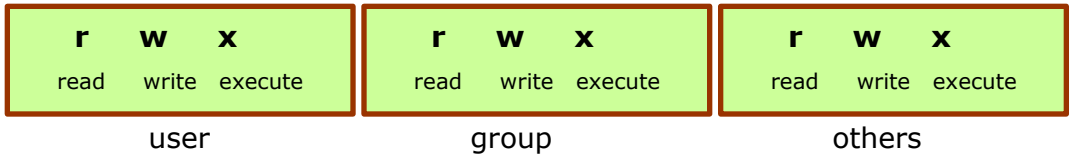
```
/home/cis90/roddyduk $ ls examples/
birds  dogs
```

The contents of the directory

```
/home/cis90/roddyduk $ ls -i examples/
2525532 birds  2525533 dogs
```

The contents of the directory with inodes (use the -i option)

Directory Read Permission



Long listing showing directory and contents:

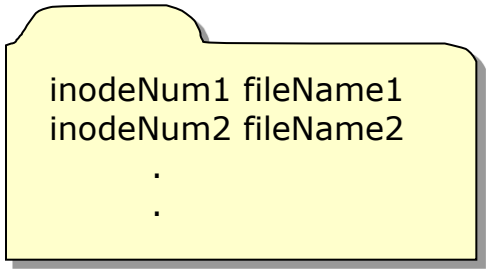
```
/home/cis90/roddyduk $ ls -ld examples/
drwxrwxr-x 5 roddyduk cis90 4096 Oct 19 13:49 examples/

/home/cis90/roddyduk $ ls -lR examples/
examples/:
total 16
drwxrwxr-x 2 roddyduk cis90 4096 Oct 19 13:50 birds
drwxrwxr-x 2 roddyduk cis90 4096 Oct 19 13:51 dogs

examples/birds:
total 16
-rw-rw-r-- 1 roddyduk cis90 12 Oct 19 13:50 abby
-rw-rw-r-- 1 roddyduk cis90 12 Oct 19 13:50 nibbie

examples/dogs:
total 24
-rw-rw-r-- 1 roddyduk cis90 10 Oct 19 13:51 benji
-rw-rw-r-- 1 roddyduk cis90 10 Oct 19 13:51 duke
-rw-rw-r-- 1 roddyduk cis90 10 Oct 19 13:51 homer
```

Use the -R option to recursively show contents of all subdirectories

Directory permissions READ

Directory Read Permission

rwx

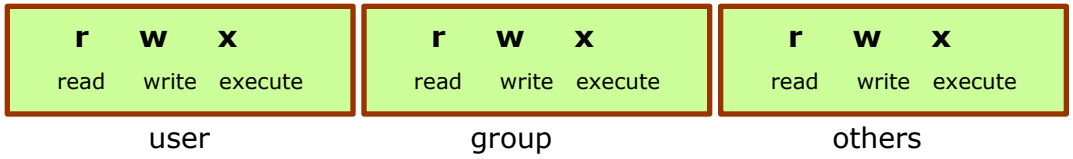


rwx

Permission	File	Directory
Read (4)	cat, more, file, head, tail, cp	ls
Write (2)	vi, saving mail	cp (into), mv, rm, ln
Execute (1)	$ command	cd, ls -l, find

*Use the **ls** command to read the contents of a directory.*
Note, having read permission is required!

Directory Read Permission



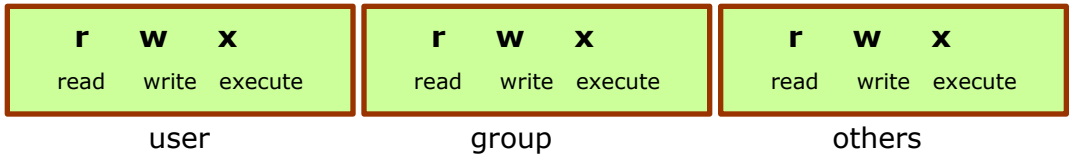
Start with normal directory permissions:

```
/home/cis90/roddyduk $ ls -ld examples/
drwxrwxr-x 5 roddyduk cis90 4096 Oct 19 13:49 examples/

/home/cis90/roddyduk $ ls -i examples/
2525532 birds  2525533 dogs
```

If read permission is removed from the directory …

Can we still list the directory contents?

Directory Read Permission



Remove read permission and confirm it's gone

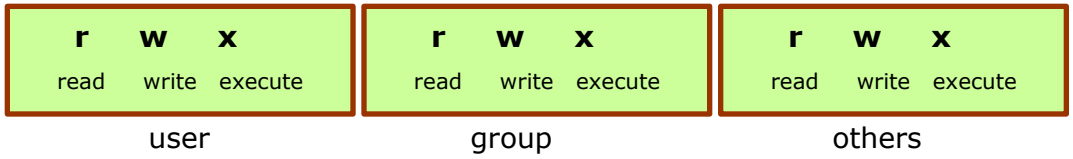
```
/home/cis90/roddyduk $ chmod u-r examples
/home/cis90/roddyduk $ ls -ld examples
d-wxrwxr-x 4 roddyduk cis90 4096 Oct 19 13:59 examples
```

Can we still list the directory contents?

```
/home/cis90/roddyduk $ ls -l examples/
ls: examples/: Permission denied
/home/cis90/roddyduk $
```

NO!

Directory Read Permission



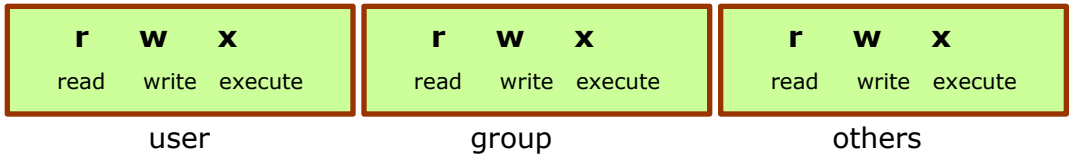
Start with normal directory permissions:

```
/home/cis90/roddyduk $ ls -ld examples/
drwxrwxr-x 5 roddyduk cis90 4096 Oct 19 13:49 examples/

/home/cis90/roddyduk $ ls -i examples/
2525532 birds  2525533 dogs
```

If read permission is removed from the directory …

*Can we still **cd** into the directory?*

Directory Read Permission



Remove read permission and confirm it's gone

```
/home/cis90/roddyduk $ chmod u-r examples
/home/cis90/roddyduk $ ls -ld examples
d-wxrwxr-x 4 roddyduk cis90 4096 Oct 19 13:59 examples
```

*Can we still **cd** into the directory?*

```
/home/cis90/roddyduk $ cd examples/
/home/cis90/roddyduk/examples $ ls
ls: .: Permission denied
/home/cis90/roddyduk/examples $ ls birds
abby  nibbie
```

Yes, but …

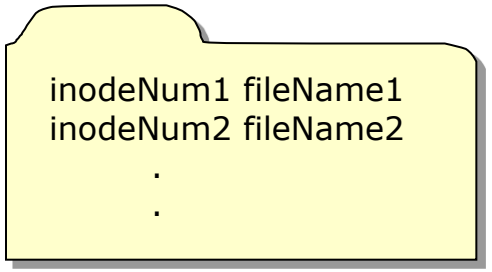
- *we still can't list the contents,*
- *yet we can still access anything in the directory!*

It's like walking into a pitch black room. You can't see anything, but if you know where things are you can still use them.

Directory permissions WRITE

Directory Write Permission

rwx

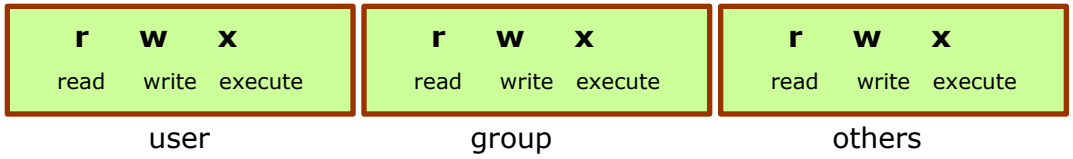


rwx

Permission	File	Directory
Read (4)	cat, more, file, head, tail, cp	ls
Write (2)	vi, saving mail	cp, mv, rm, ln
Execute (1)	$ command	cd, ls -l, find

Removing directory w permission
- *can't cp files to it,*
- *can't remove files,*
- *can't move files out,*
- *can't add links*

Directory Write Permission



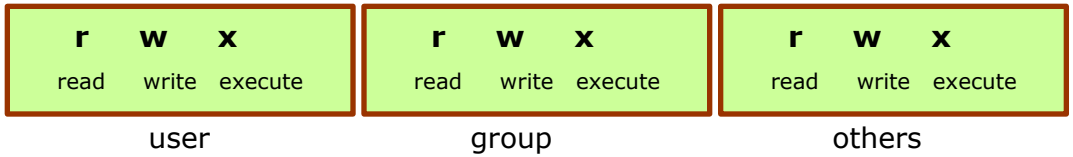
Start with normal directory permissions:

```
/home/cis90/roddyduk $ ls -ld examples/
drwxrwxr-x 5 roddyduk cis90 4096 Oct 19 13:49 examples/

/home/cis90/roddyduk $ ls -i examples/
2525532 birds  2525533 dogs
```

If write permission is removed from the directory …

Can we remove files from the directory?

Directory Write Permission



Remove write permission and confirm it's gone

```
/home/cis90/roddyduk $ chmod u-w examples
/home/cis90/roddyduk $ ls -ld examples
dr-xrwxr-x 4 roddyduk cis90 4096 Oct 19 13:59 examples/
```

Can we remove files form the directory?

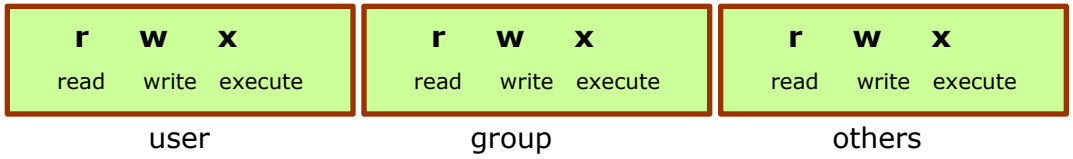
```
/home/cis90/roddyduk/examples $ rmdir dogs
rmdir: dogs: Permission denied
```

NO!

Yet we can cd into and list directory contents

```
/home/cis90/roddyduk $ cd examples/
/home/cis90/roddyduk/examples $ ls
birds  dogs
```

Directory Write Permission



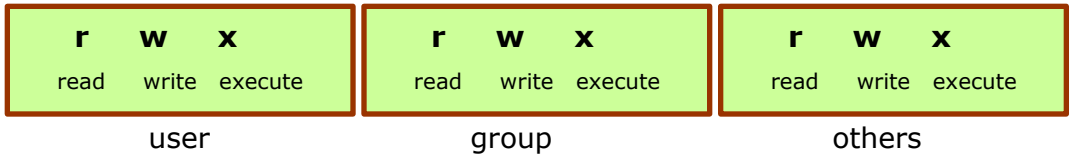
Start with normal directory permissions:

```
/home/cis90/roddyduk $ ls -ld examples/
drwxrwxr-x 5 roddyduk cis90 4096 Oct 19 13:49 examples/

/home/cis90/roddyduk $ ls -i examples/
2525532 birds  2525533 dogs
```

If write permission is removed from the directory …

Can we create new files or copy/move files into the directory?

Directory Write Permission



Remove write permission and confirm it's gone

```
/home/cis90/roddyduk $ chmod u-w examples
/home/cis90/roddyduk $ ls -ld examples
dr-xrwxr-x 4 roddyduk cis90 4096 Oct 19 13:59 examples/
```

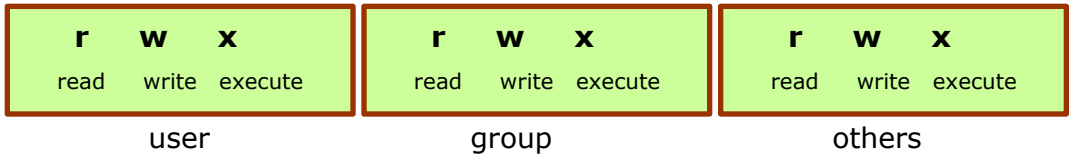
Can we create new files or copy/move files into the directory?

```
/home/cis90/roddyduk $ cp letter examples/
cp: cannot create regular file `examples/letter': Permission denied
/home/cis90/roddyduk $ mv letter examples/
mv: cannot move `letter' to `examples/letter': Permission denied
/home/cis90/roddyduk $ touch examples/newfile
touch: cannot touch `examples/newfile': Permission denied
/home/cis90/roddyduk $
```

NO!

To change the contents of a directory (either add or remove files) requires write permission

Directory Write Permission



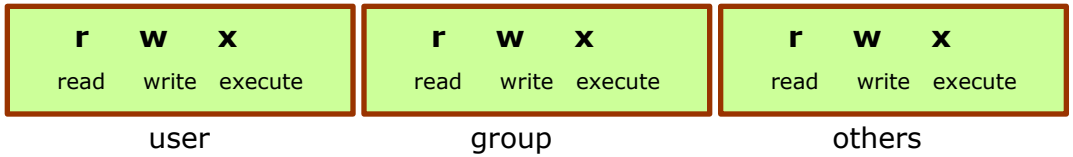
Start with normal directory permissions:

```
/home/cis90/roddyduk $ ls -ld examples/
drwxrwxr-x 5 roddyduk cis90 4096 Oct 19 13:49 examples/

/home/cis90/roddyduk $ ls -i examples/
2525532 birds  2525533 dogs
```

If write permission is removed from the directory …

Can we move files out of the directory?

Directory Write Permission



Remove write permission and confirm it's gone

```
/home/cis90/roddyduk $ chmod u-w examples
/home/cis90/roddyduk $ ls -ld examples
dr-xrwxr-x 4 roddyduk cis90 4096 Oct 19 13:59 examples/
```

Can we move files out of the directory?

```
/home/cis90/roddyduk $ mv examples/birds .
mv: cannot move `examples/birds' to `./birds': Permission denied
```

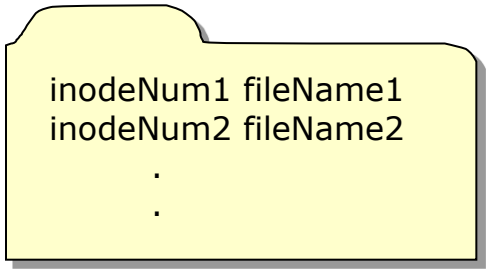
NO!

To change the contents of a directory (either add or remove files) requires write permission

Directory permissions EXECUTE

Directory Execute Permission

rwx



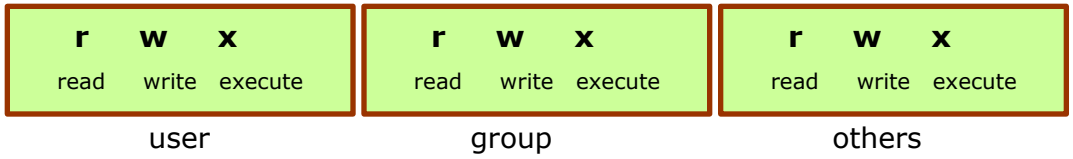
rwx

Permission	File	Directory
Read (4)	cat, more, file, head, tail, cp	ls
Write (2)	vi, saving mail	cp, mv, rm, ln
Execute (1)	$ command	cd, ls -l, find

Removing directory x permission

- ***cannot*** *retrieve inode information (ls –l)*
  (which means no file data either)
- ***cannot*** *cd into directory*

Directory Execute Permission



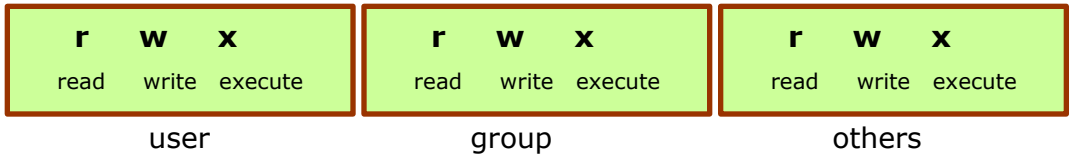
Start with normal directory permissions:

```
/home/cis90/roddyduk $ ls -ld examples/
drwxrwxr-x 5 roddyduk cis90 4096 Oct 19 13:49 examples/

/home/cis90/roddyduk $ ls -i examples/
2525532 birds  2525533 dogs
```

If execute permission is removed from the directory …

Can we change into (cd) the directory?

Directory Execute Permission



Remove execute permission and confirm it's gone

```
/home/cis90/roddyduk $ chmod u-x examples
/home/cis90/roddyduk $ ls -ld examples
drw-rwxr-x 4 roddyduk cis90 4096 Oct 19 13:59 examples/
```

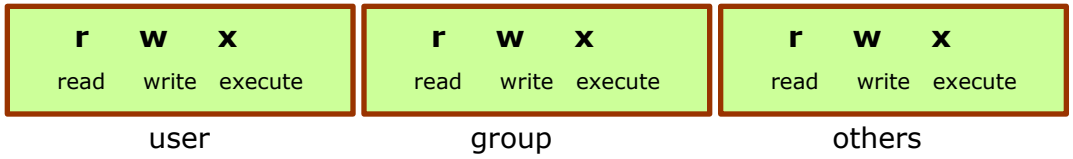
Can we change into (cd) the directory?

```
/home/cis90/roddyduk $ cd examples/
-bash: cd: examples/: Permission denied
/home/cis90/roddyduk $
```

NO!

Execute permission is required to change into a directory or to get inode based information for any of the files in the directory. Note, without inode information you can't get to a file's data.

Directory Execute Permission



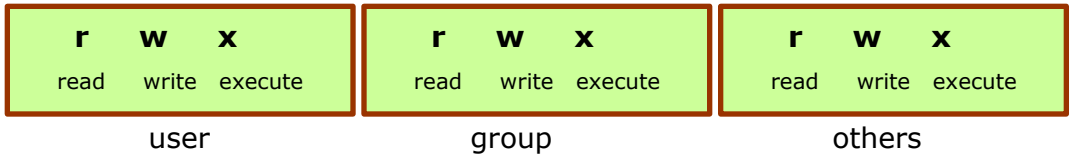
Start with normal directory permissions:

```
/home/cis90/roddyduk $ ls -ld examples/
drwxrwxr-x 5 roddyduk cis90 4096 Oct 19 13:49 examples/

/home/cis90/roddyduk $ ls -i examples/
2525532 birds  2525533 dogs
```

If execute permission is removed from the directory …

Can we list directory contents?

Directory Execute Permission



Remove execute permission and confirm it's gone

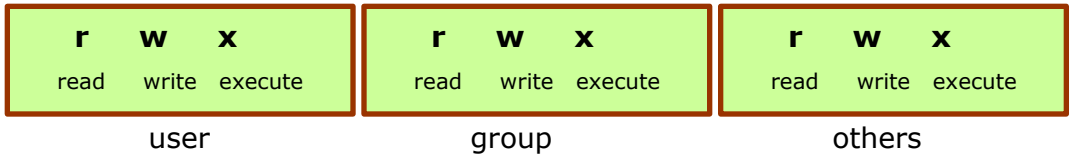
```
/home/cis90/roddyduk $ chmod u-x examples
/home/cis90/roddyduk $ ls -ld examples
drw-rwxr-x 4 roddyduk cis90 4096 Oct 19 13:59 examples/
```

Can list directory contents?

```
/home/cis90/roddyduk $ ls examples/
birds  dogs
```

Yes

Directory Execute Permission



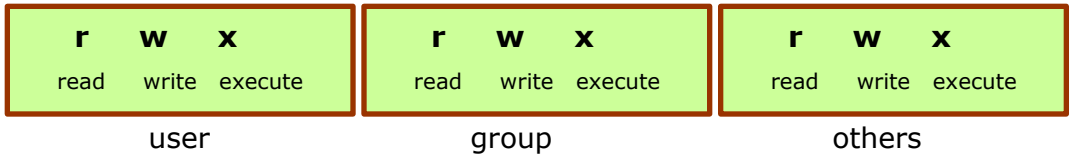
Start with normal directory permissions:

```
/home/cis90/roddyduk $ ls -ld examples/
drwxrwxr-x 5 roddyduk cis90 4096 Oct 19 13:49 examples/

/home/cis90/roddyduk $ ls -i examples/
2525532 birds  2525533 dogs
```

If execute permission is removed from the directory …

Can we do a long listing of the directory?

Directory Execute Permission



Remove execute permission and confirm it's gone

```
/home/cis90/roddyduk $ chmod u-x examples
/home/cis90/roddyduk $ ls -ld examples
drw-rwxr-x 4 roddyduk cis90 4096 Oct 19 13:59 examples/
```

Can we do a long listing (show inode information) of the directory?

```
/home/cis90/roddyduk $ ls -l examples/
total 0
?--------- ? ? ? ?            ? birds
?--------- ? ? ? ?            ? dogs
```

Incomplete! Only file names. No information kept in the file's inode is shown!

We can read the filenames, but without execute permission we can't retrieve information from the inode