Lesson Module Checklist
• Slides
• WB

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

• Lab 7 tested
• Lab X1 tested

• 9V backup battery for microphone
• Backup slides, CCC info, handouts on flash drive

Introductions and Credits

Jim Griffin
• Created this Linux course
• Created Opus and the CIS VLab
• Jim's site: http://cabrillo.edu/~jgriffin/

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

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

CIS 90 - Lesson 8

Cabrillo College est. 1959

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

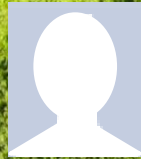Daniel   Riley   Solomon   Curtis   Dillon   Pam
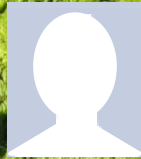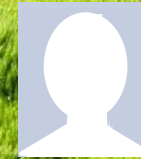
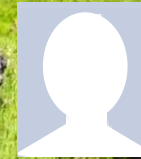Aarron   Liz   Gabe   Lucie   Liam   Michael L.   Ryan   Ben L.   Roger   Ariana
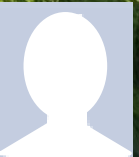
Evan   Alex   Natalia   Perky   Samantha   Paul S.   Hilario   Tyrone   Ben C.   Justin
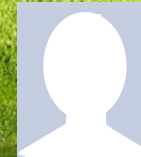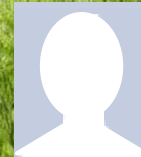
Andrew   Jordan   Mark   Ryan   Greg   MJ   Jay   Rich

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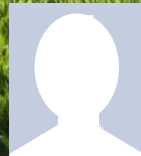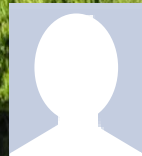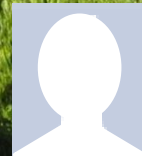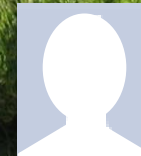
Quiz

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

See electronic white board

email answers to: risimms@cabrillo.edu

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

[ ] Preload White Board with *cis*lesson??*-WB*
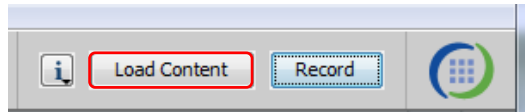
[ ] Connect session to Teleconference

Session now connected to teleconference

Connect Session To Teleconference...
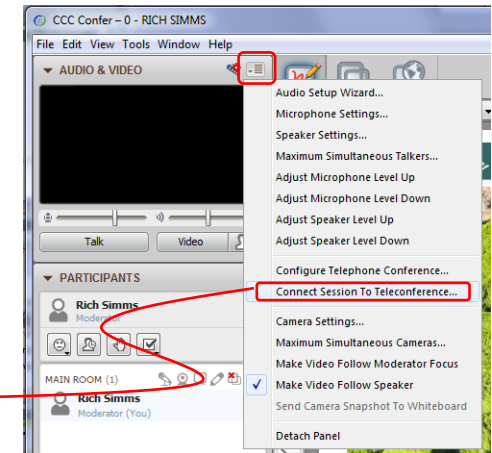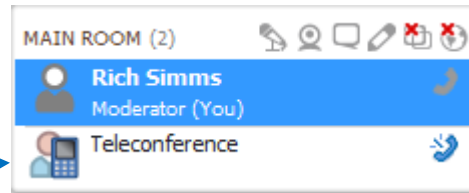
[ ] Is recording on?

Red dot means recording

[ ] Use teleconferencing, not mic

Should be greyed out

5
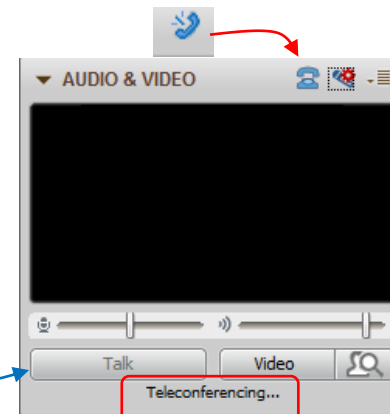
[ ] **Video (webcam) optional**

[ ] **layout and share apps**

[ ] Video (webcam) optional

[ ] Follow moderator

[ ] Double-click on postages stamps

7
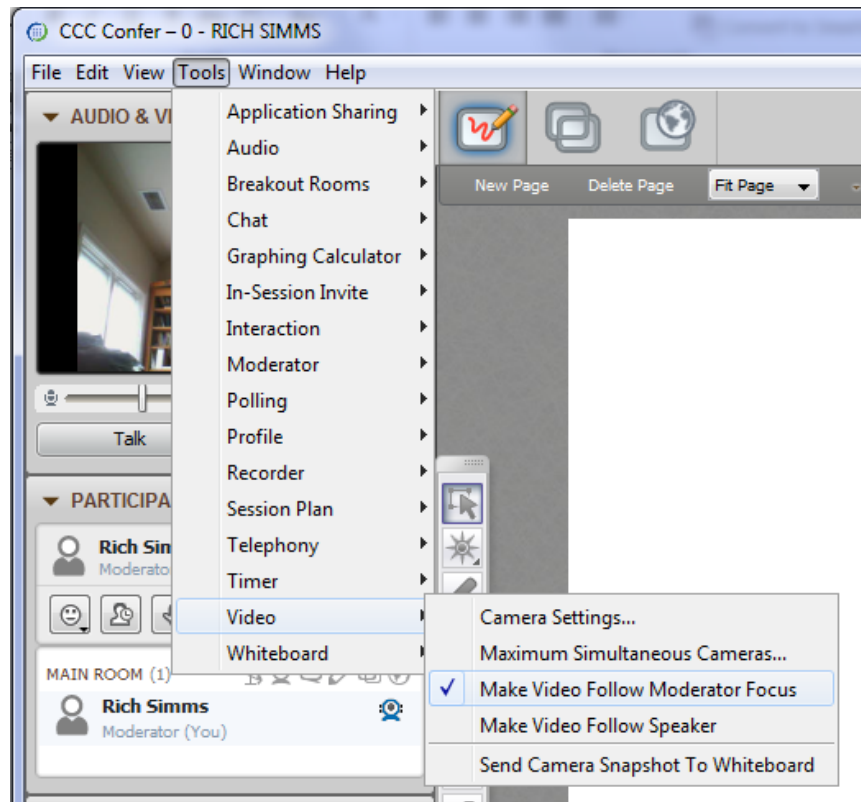
Universal Fix for CCC Confer:

1) Shrink (500 MB) and delete Java cache
2) Uninstall and reinstall latest Java runtime

Control Panel (small icons)

General Tab > Settings…
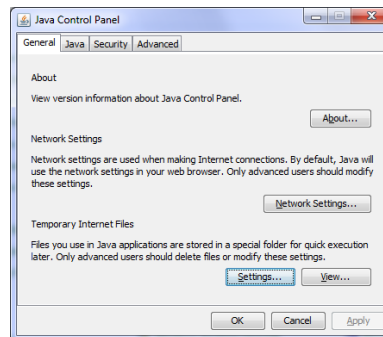
500MB cache size
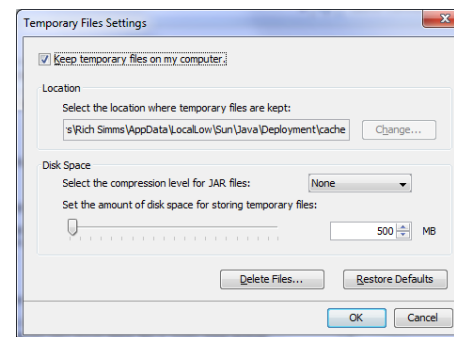
Delete these
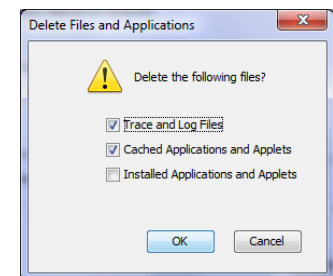
Google Java download

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

9

Questions

Questions

Lesson material?

Labs?

How this course works?

Graded work in home directories

Answers in /home/cis90/answers
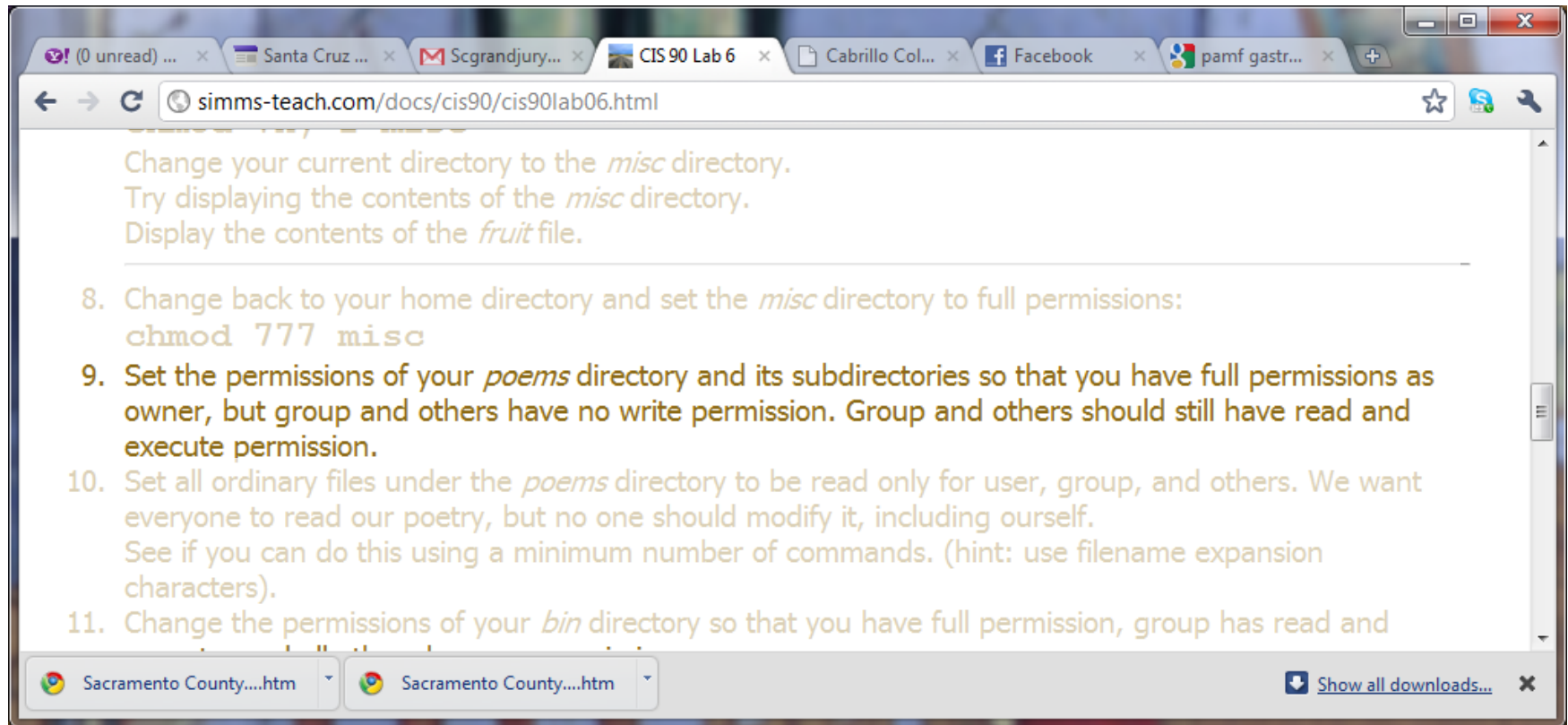
Chinese Proverb	他問一個問題，五分鐘是個傻子，他不問一個問題仍然是一個傻瓜永遠。
	He who asks a question is a fool for five minutes; he who does not ask a question remains a fool forever.

11

Lab 6 Tips

One of the steps in Lab 6
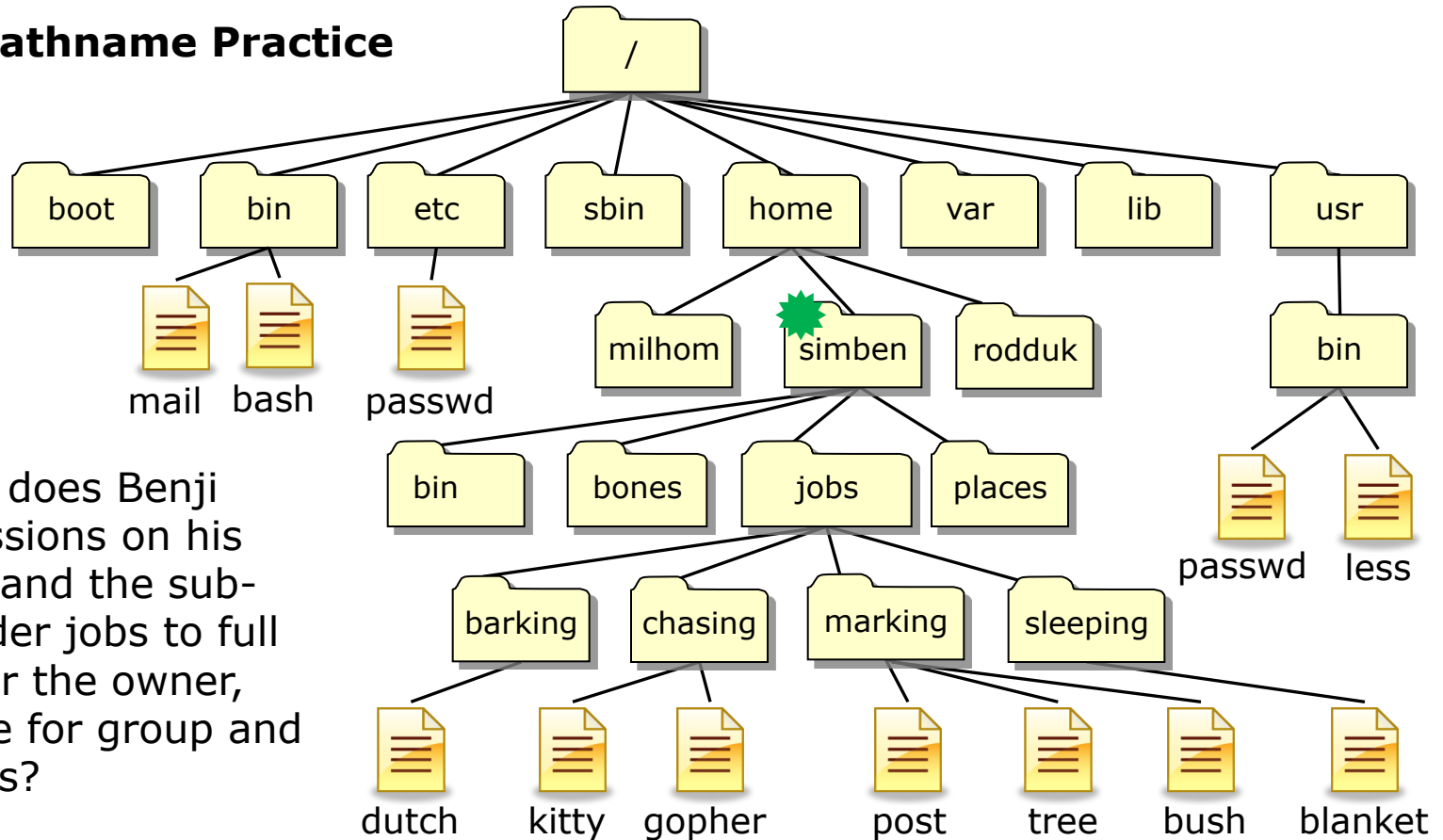
Change your current directory to the *misc* directory.
Try displaying the contents of the *misc* directory.
Display the contents of the *fruit* file.

8. Change back to your home directory and set the *misc* directory to full permissions:
   `chmod 777 misc`
9. Set the permissions of your *poems* directory and its subdirectories so that you have full permissions as owner, but group and others have no write permission. Group and others should still have read and execute permission.
10. Set all ordinary files under the *poems* directory to be read only for user, group, and others. We want everyone to read our poetry, but no one should modify it, including ourself.
    See if you can do this using a minimum number of commands. (hint: use filename expansion characters).
11. Change the permissions of your *bin* directory so that you have full permission, group has read and

File Tree Pathname Practice

From ✳ 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
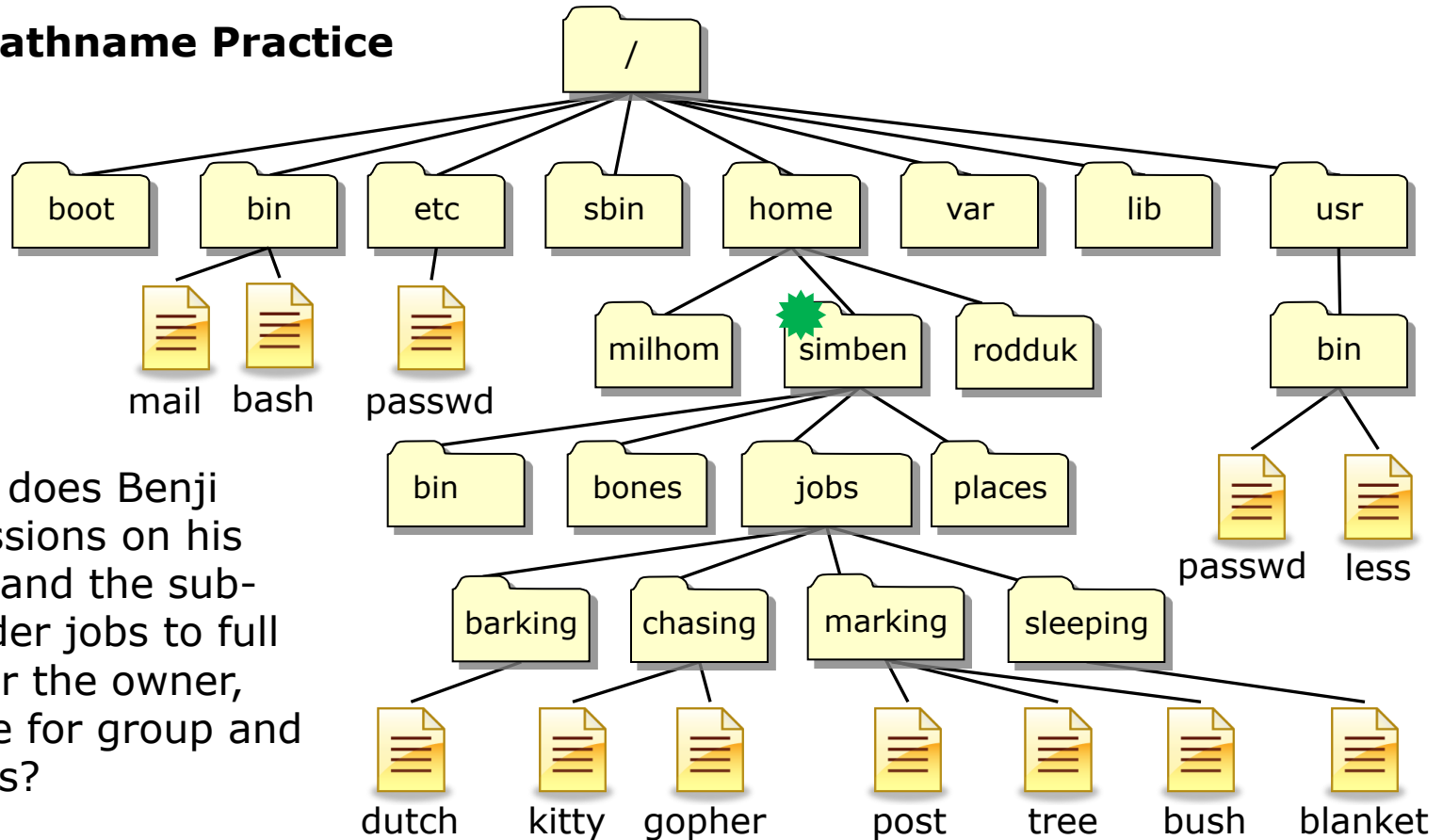
14

File Tree Pathname Practice

From ✴ 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

Using relative paths allows us to do the same thing and uses one less command
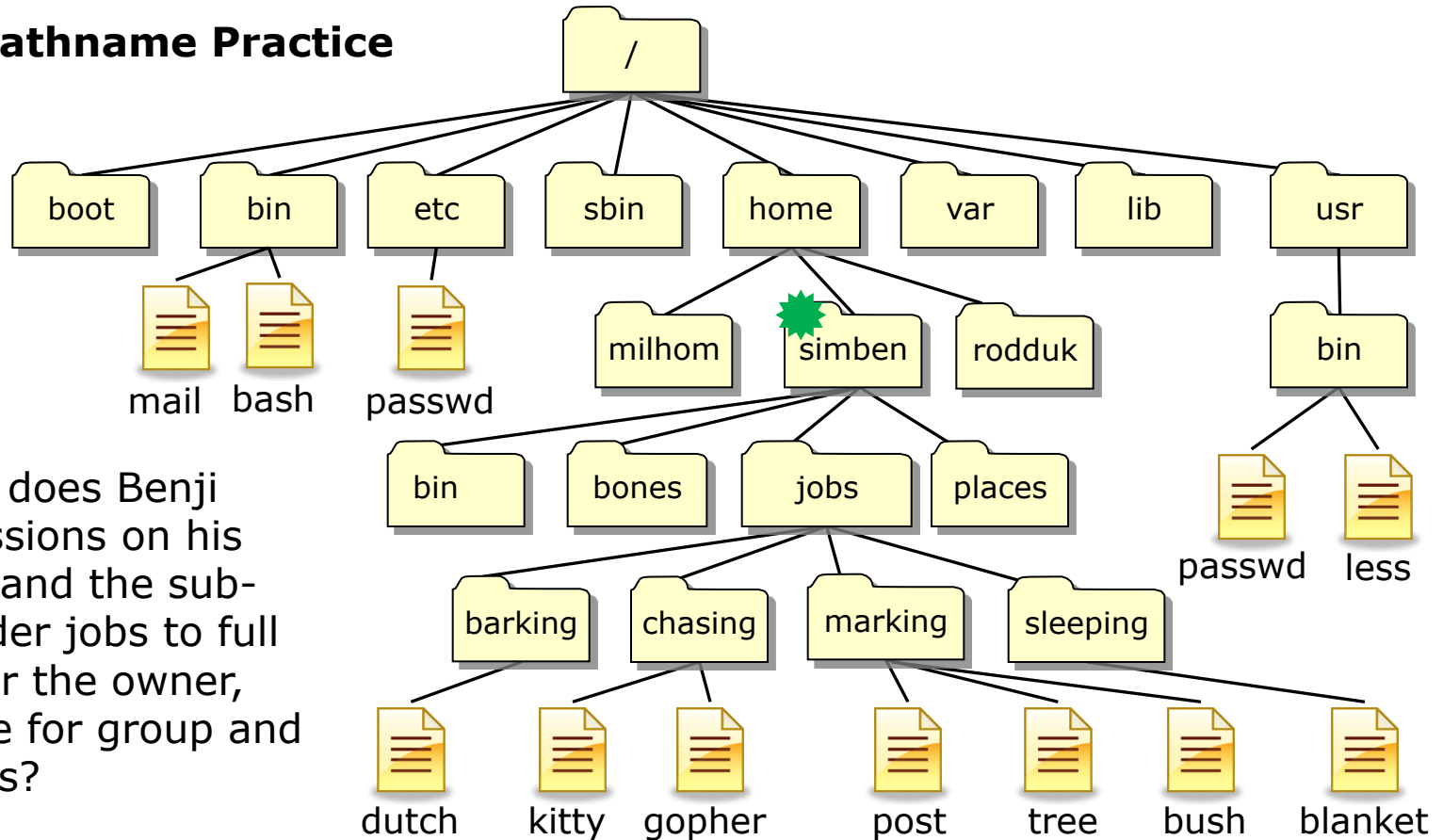
15

File Tree Pathname Practice

From ✳ 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/*

Using relative paths and a filename expansion metacharacter lets us do the same things with only two commands
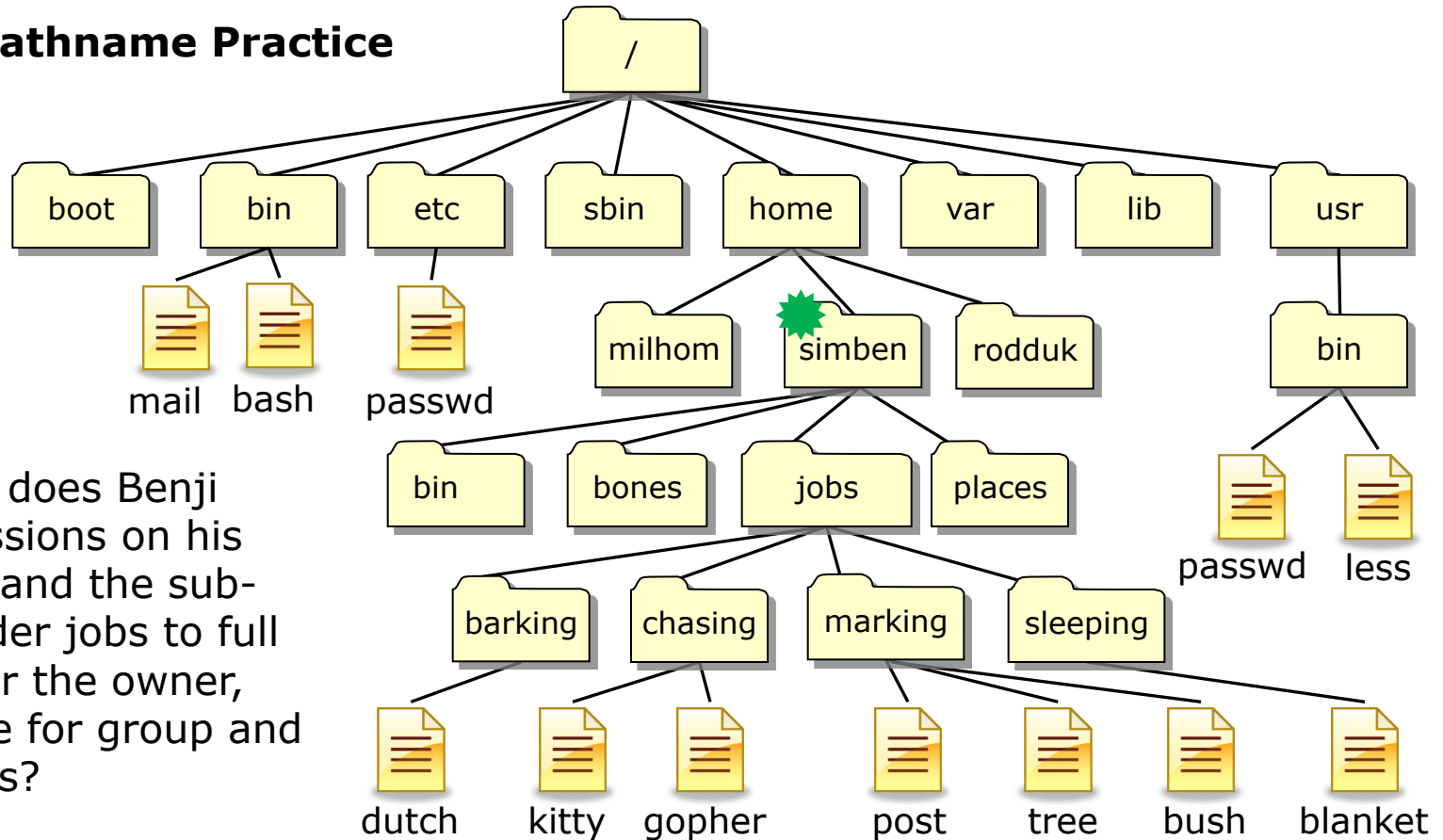
16

File Tree Pathname Practice

From ✳ 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/*

Using relative path, filename expansion metacharacter and putting all the arguments on the command line lets us do the same thing with one command!

17

File Tree Pathname Practice

From ✳ 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?

Using elbow grease:
chmod 750 jobs
cd jobs
chmod 750 barking
chmod 750 chasing
chmod 750 marking
chmod 750 sleeping

Both ways work, the choice is yours!

Leveraging the shell:
chmod 750 jobs jobs/*

18

Another step in Lab 6

simms-teach.com/docs/cis90/cis90lab06.html

```
chmod 777 misc
```

9. Set the permissions of your *poems* directory and its subdirectories so that you have full permissions as owner, but group and others have no write permission. Group and others should still have read and execute permission.

10. Set all ordinary files under the *poems* directory to be read only for user, group, and others. We want everyone to read our poetry, but no one should modify it, including ourself.
See if you can do this using a minimum number of commands. (hint: use filename expansion characters).

11. Change the permissions of your *bin* directory so that you have full permission, group has read and execute, and all others have no permissions.

12. Set the executable files under *bin* to have the following permissions:

```
r-xr-x---
```

disallowing others outside the group from executing our commands.
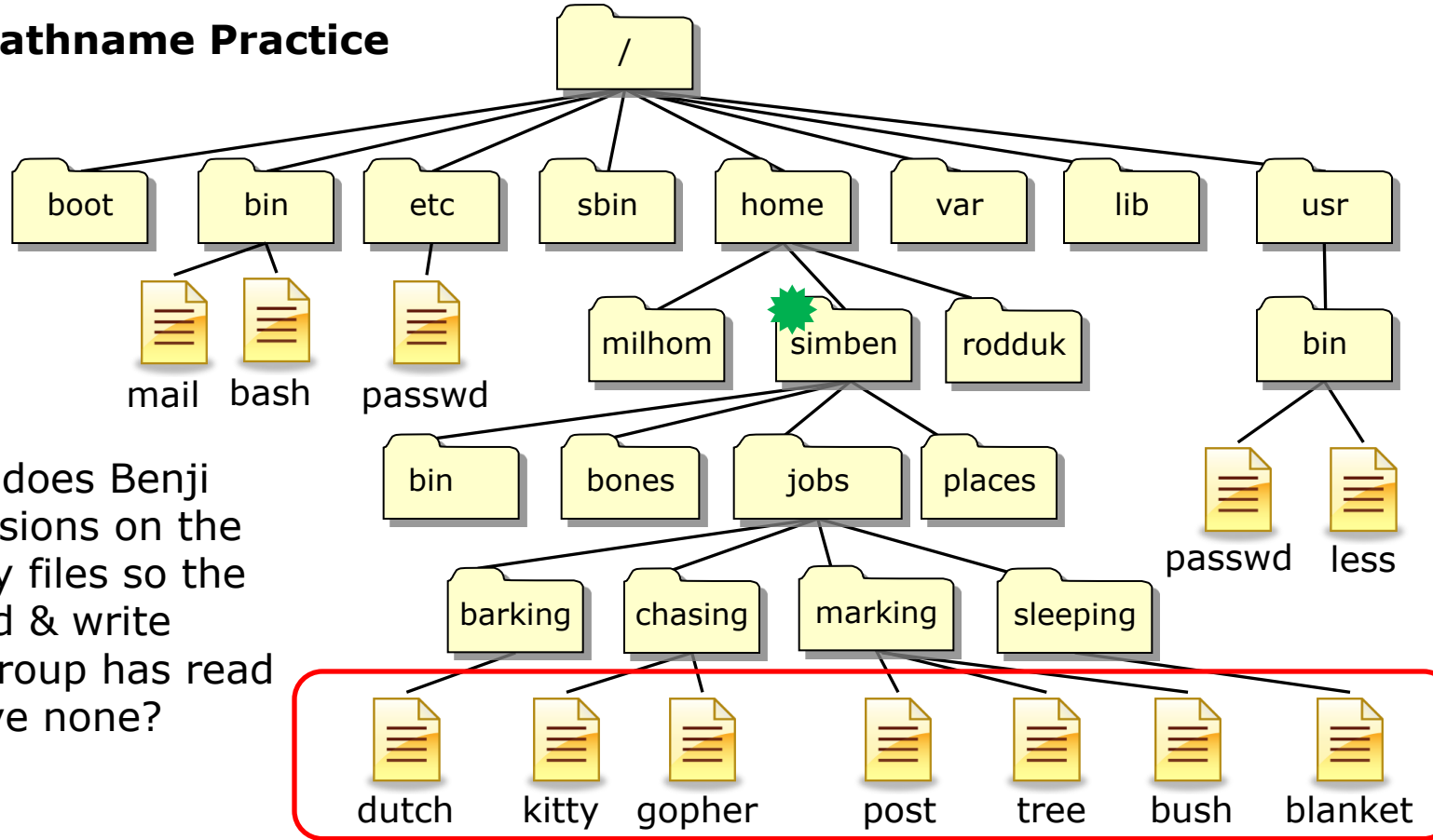
Sacramento County....htm   Sacramento County....htm   Show all downloads...

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**

Elbow grease method takes 16 commands
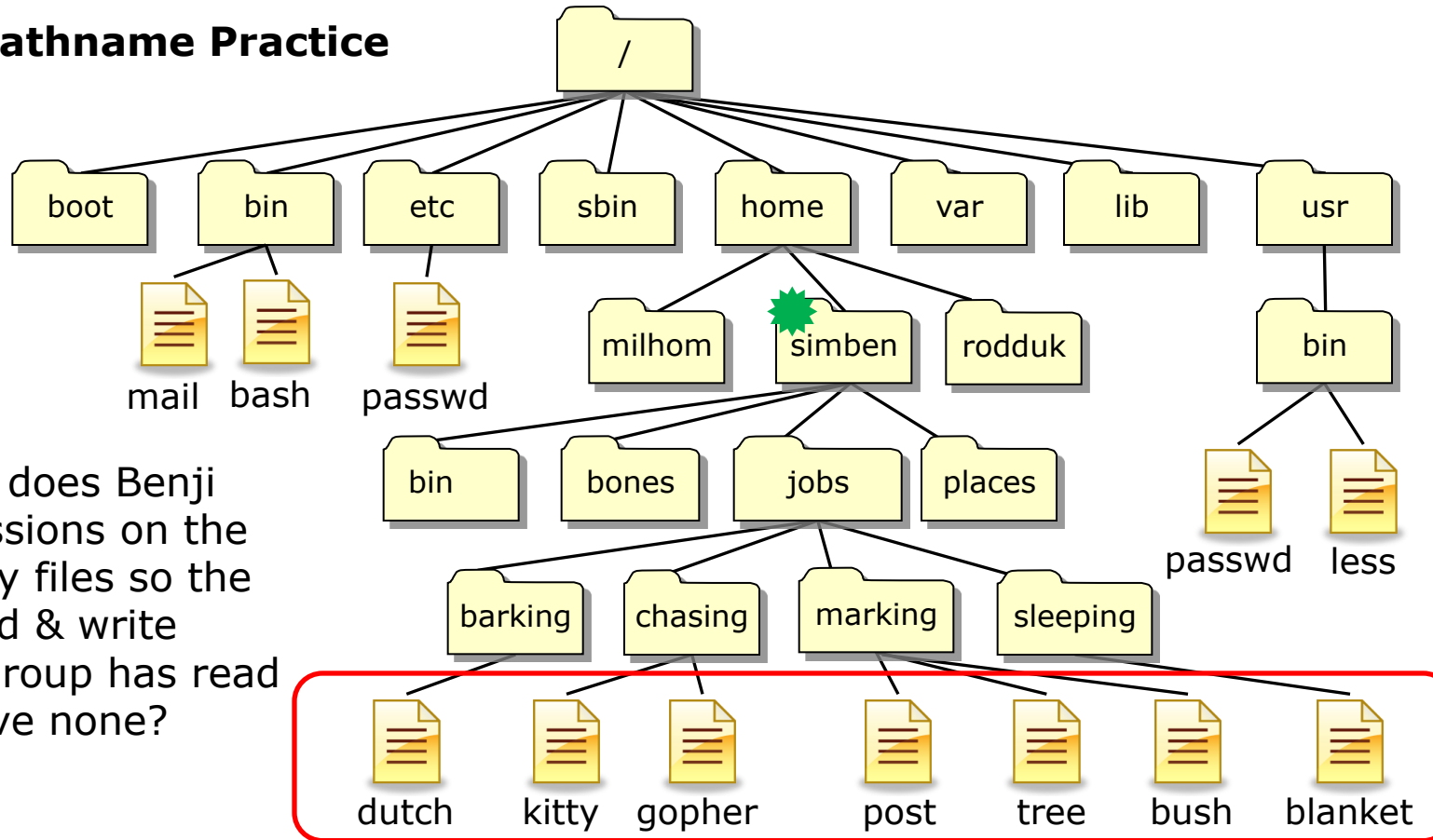
20

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**

*Using multiple arguments:
takes 13 commands*
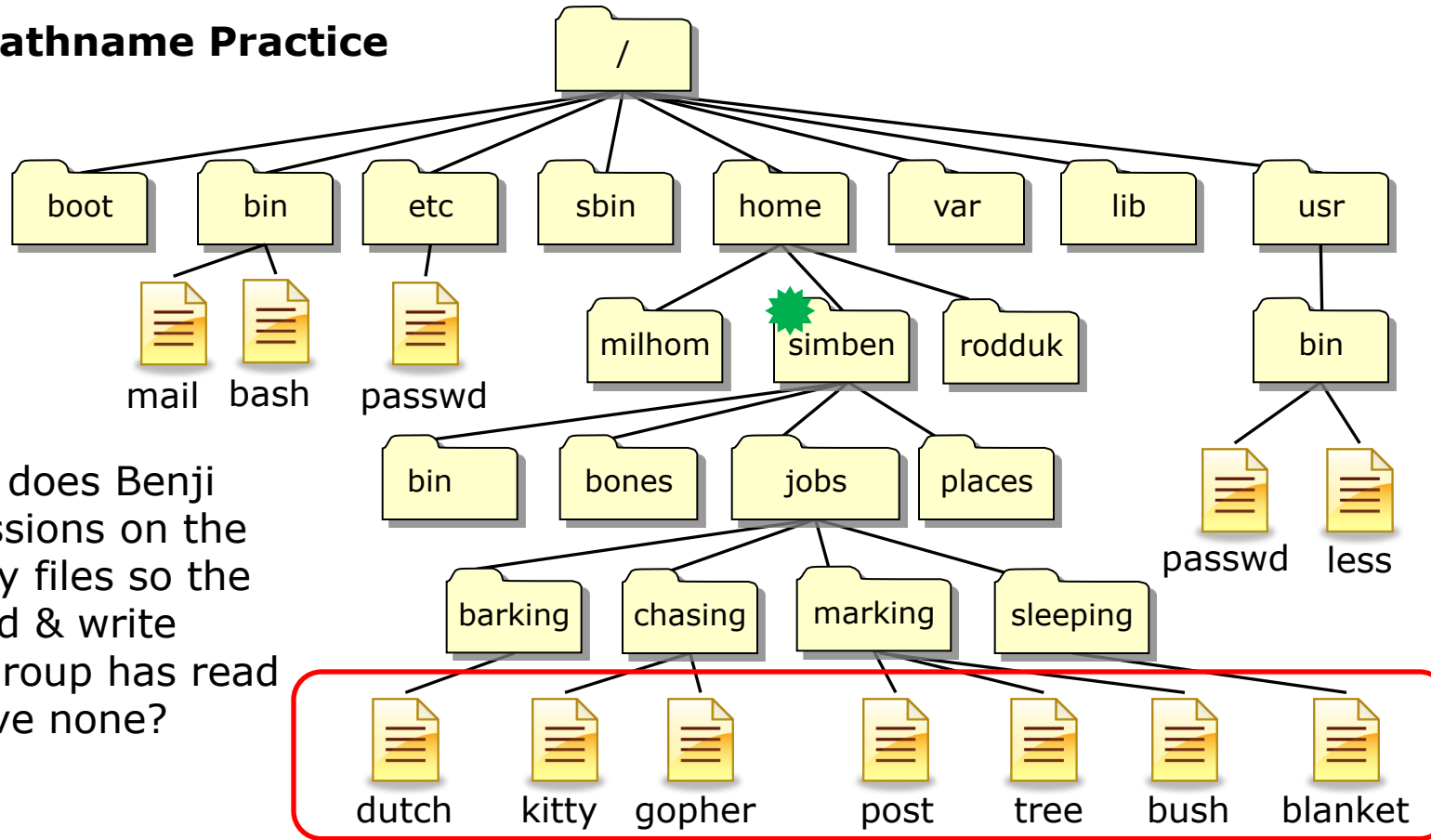
21

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

Using filename expansion metacharacters takes 13 commands
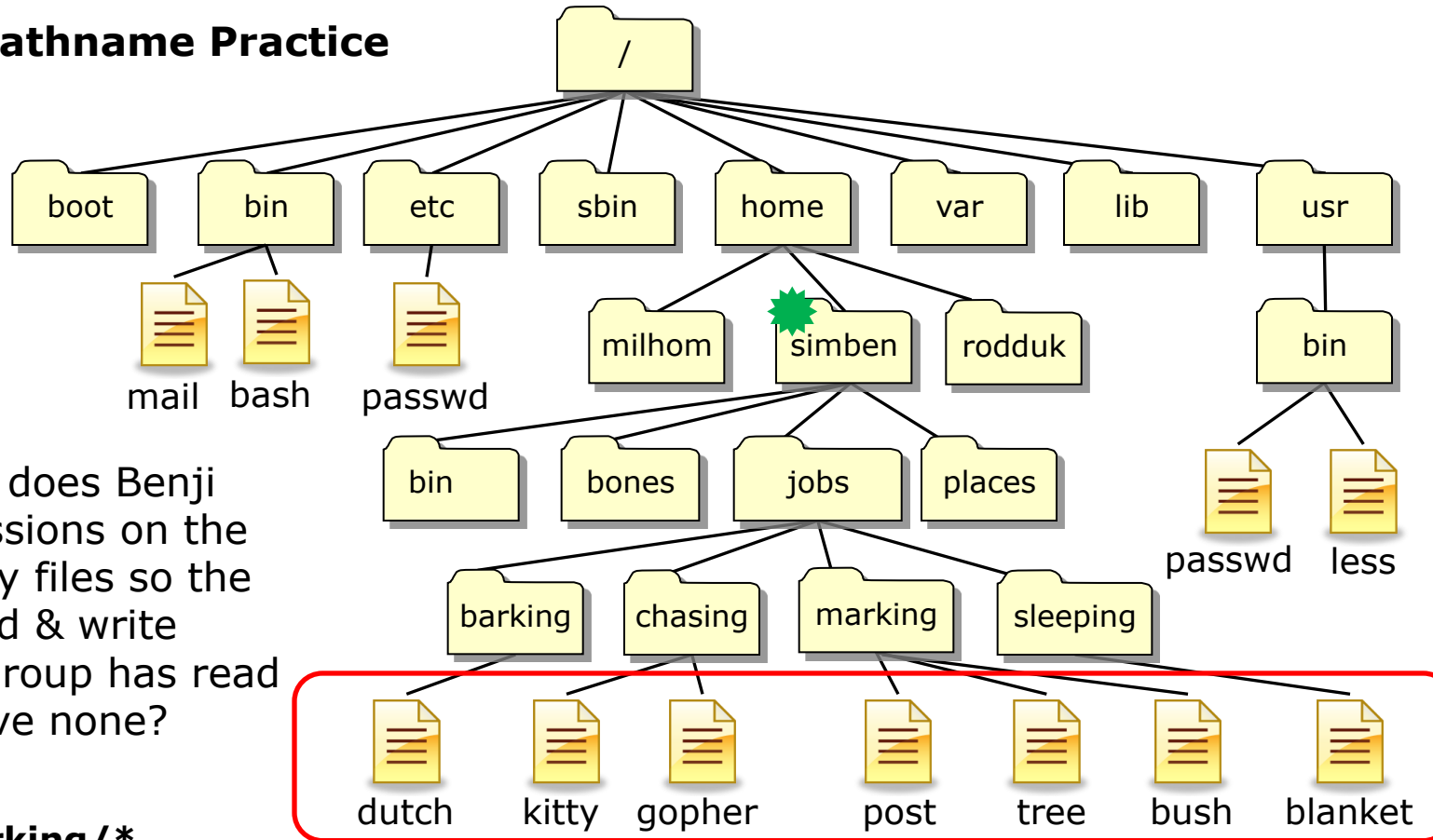
22

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 ..

Using relative paths and filename expansion characters takes 6 commands
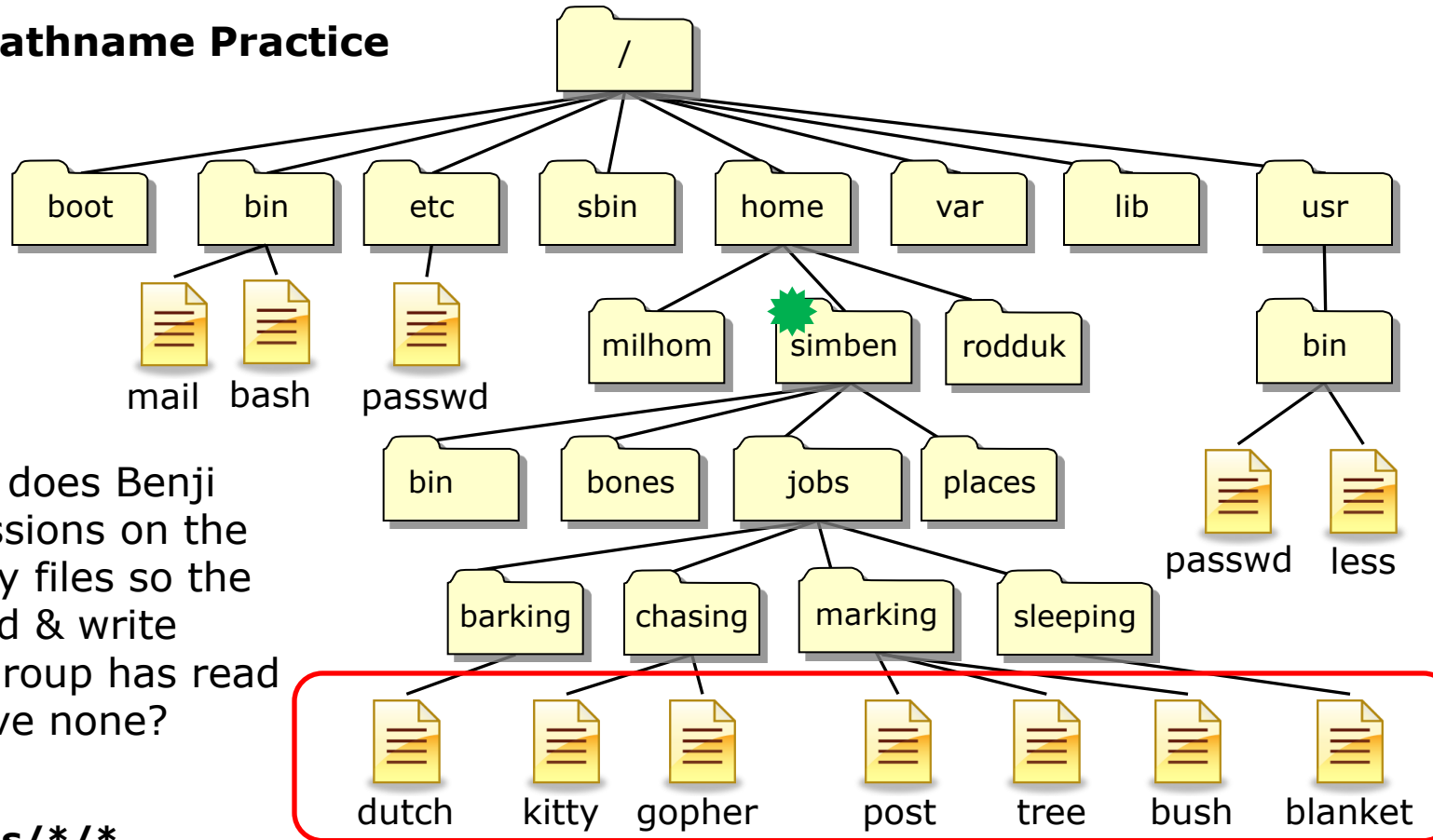
23

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/*/*

Using relative paths, filename expansion characters and combining all arguments on a single command line takes a single command

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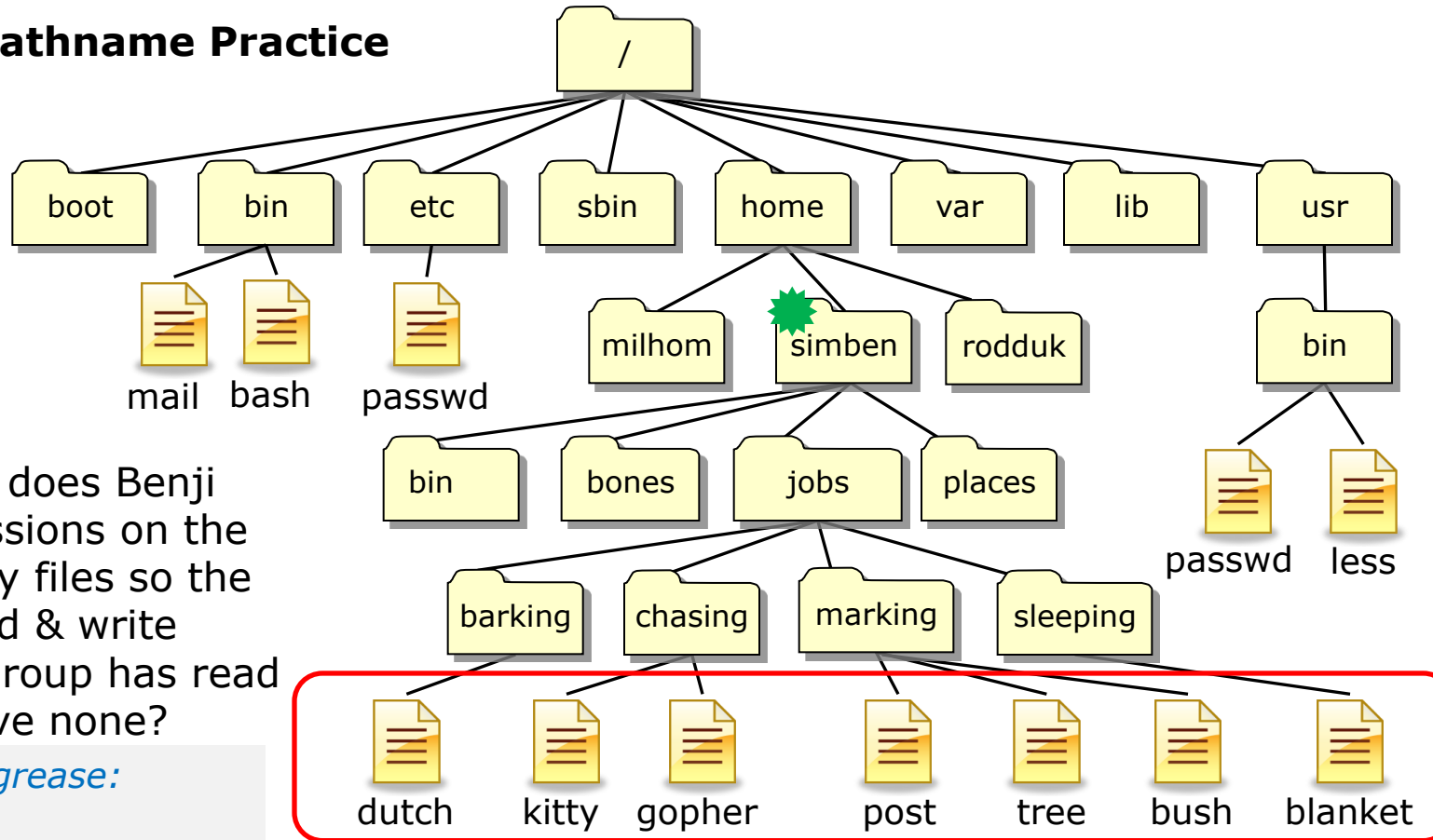
From 💥 how does Benji change permissions on the circled ordinary files so the owner has read & write permissions, group has read and others have none?

Using elbow grease:
```
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
```

Both ways work, the choice is yours!

Leveraging the shell:
chmod 640 jobs/*/*

25

Warmup

File Tree Pathname Practice
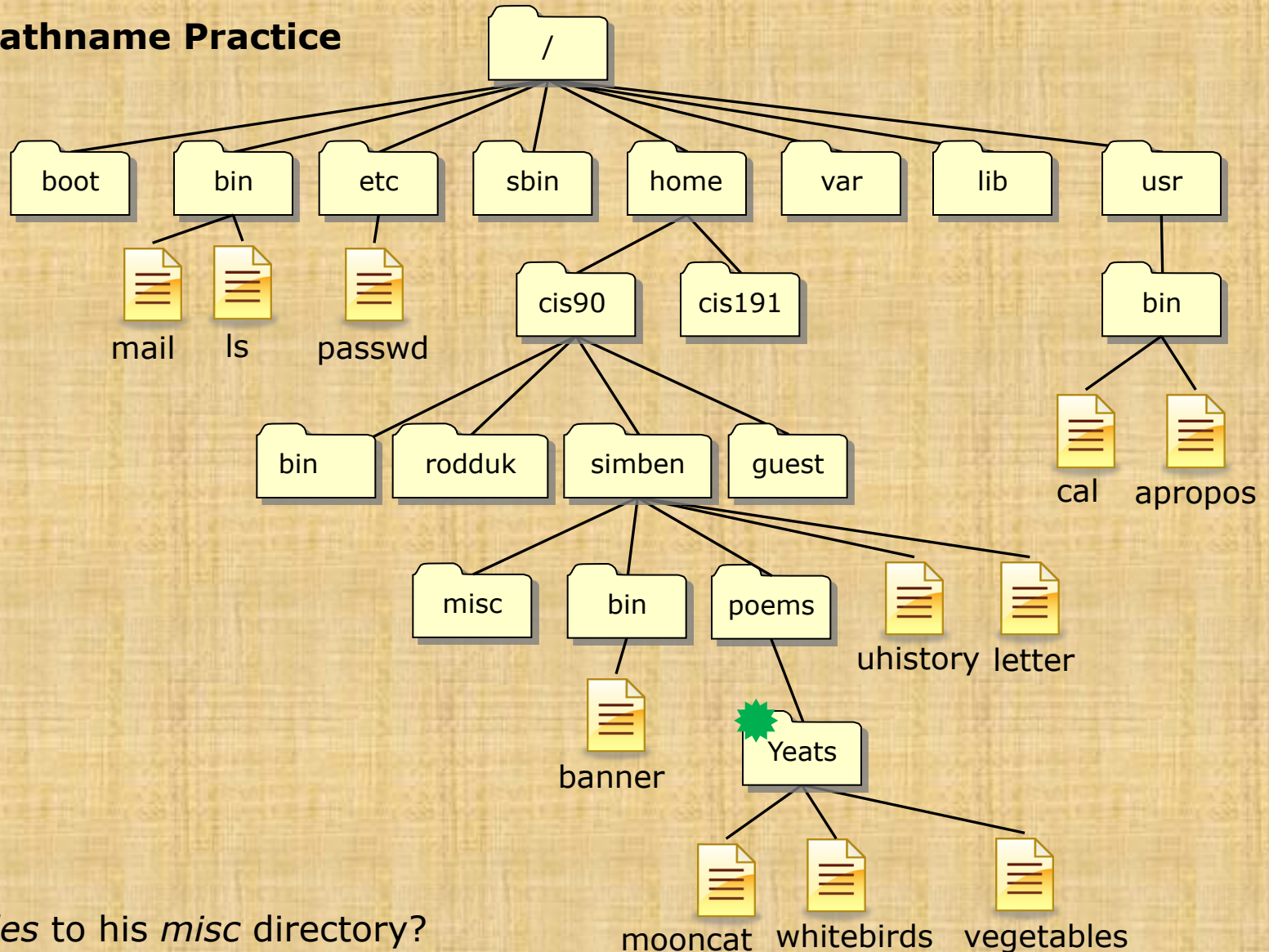
From ✸ how
does Benji:

Move *vegetables* to his *misc* directory?

File Tree Pathname Practice
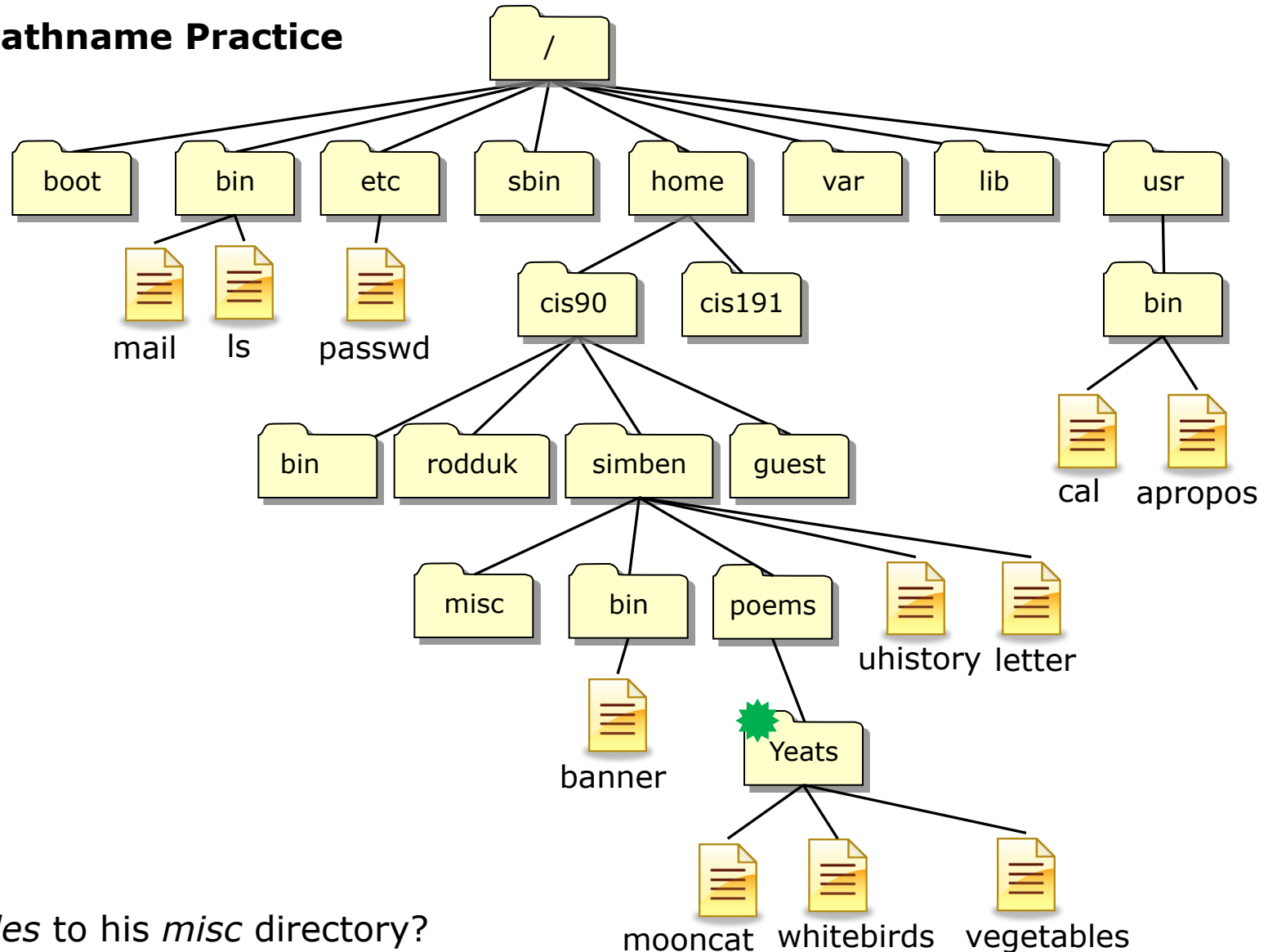
From ✹ how does Benji:

Move *vegetables* to his *misc* directory?

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

File Tree Pathname Practice
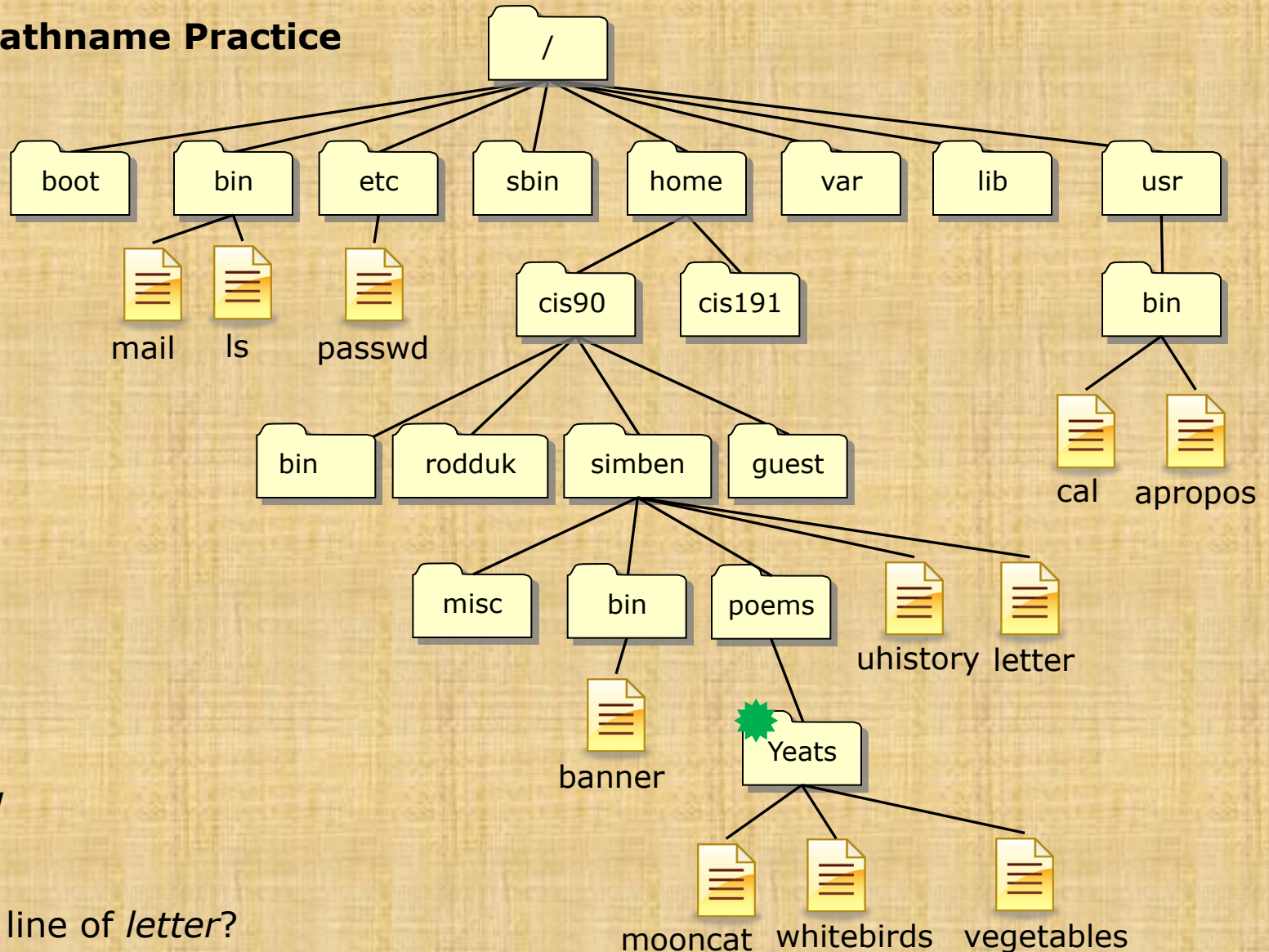
From ✴ how
does Benji:

Print the last line of *letter*?

File Tree Pathname Practice
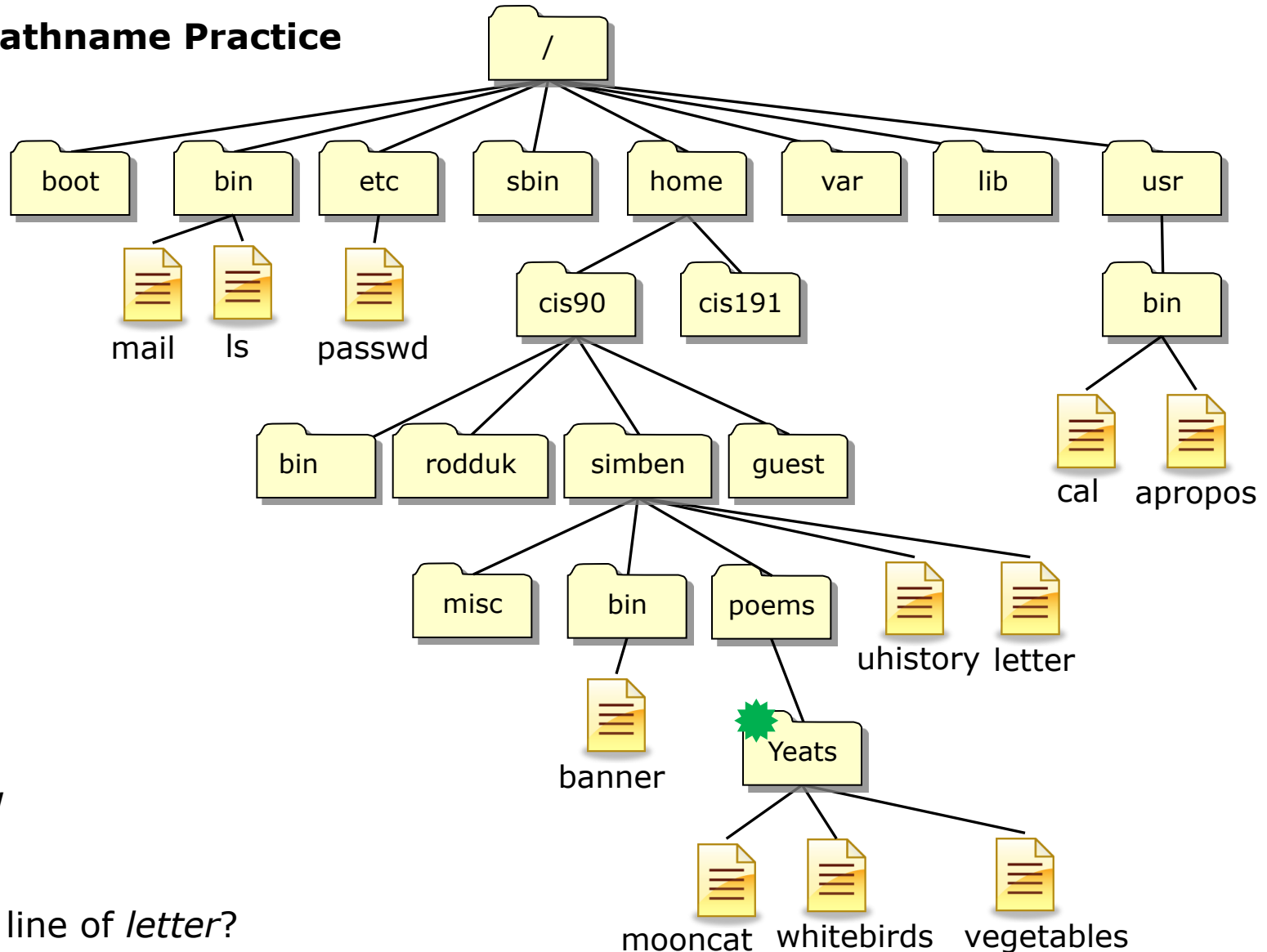
From ✹ how
does Benji:

Print the last line of *letter*?

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

File Tree Pathname Practice
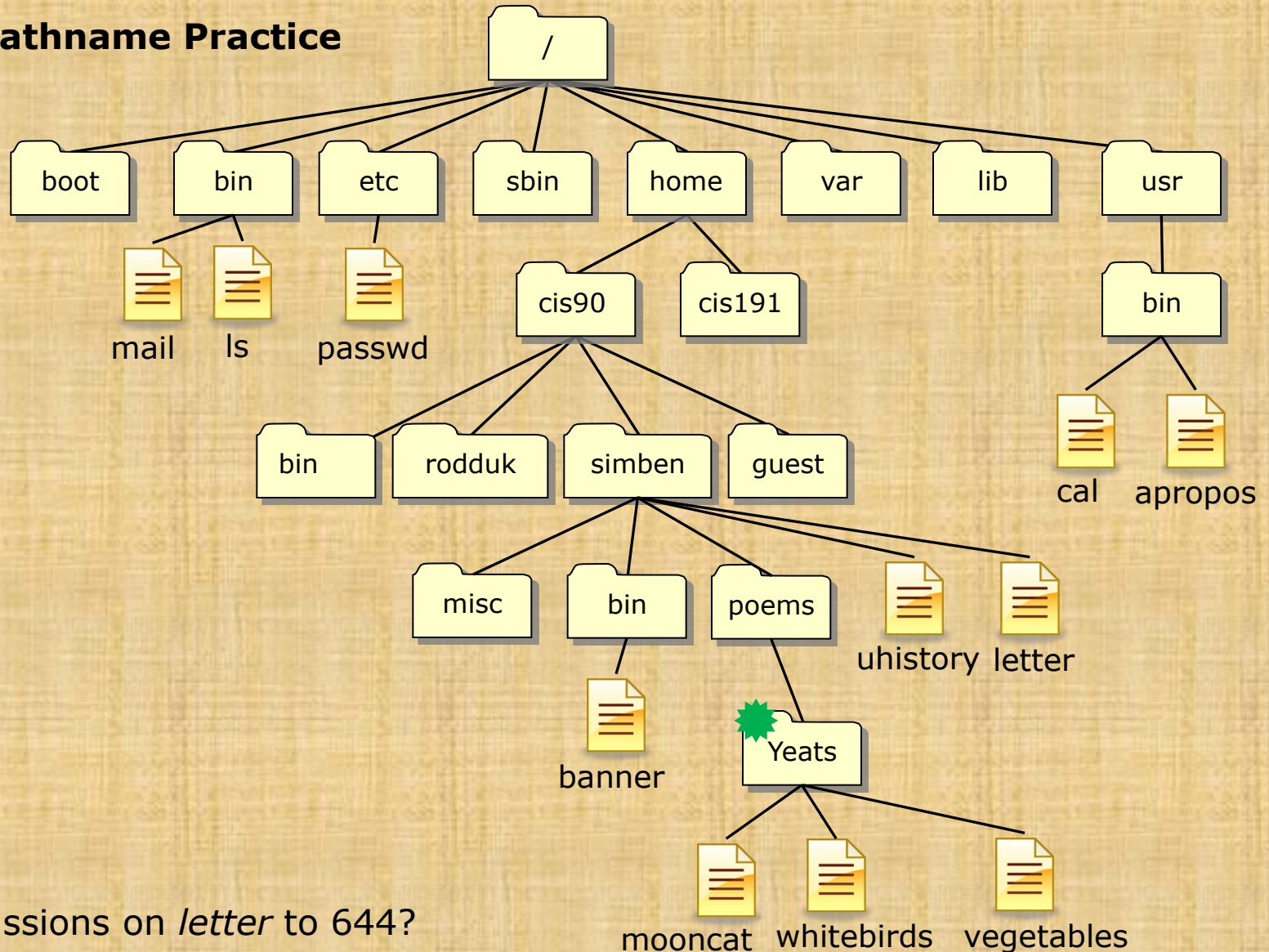
```
                              /
        ┌──────┬──────┬──────┼──────┬──────┬──────┐
      boot    bin    etc   sbin   home   var   lib   usr
              │       │             │                  │
            mail ls passwd    ┌─────┴─────┐          bin
                            cis90       cis191      ┌──┴──┐
                      ┌──────┼──────┐              cal  apropos
                    bin  rodduk simben guest
                             ┌────┼────┬──────┬──────┐
                           misc  bin  poems uhistory letter
                                  │     │
                               banner  Yeats
                                    ┌────┼────┐
                                 mooncat whitebirds vegetables
```

From ✳ how
does Benji:

Change permissions on *letter* to 644?

31

File Tree Pathname Practice
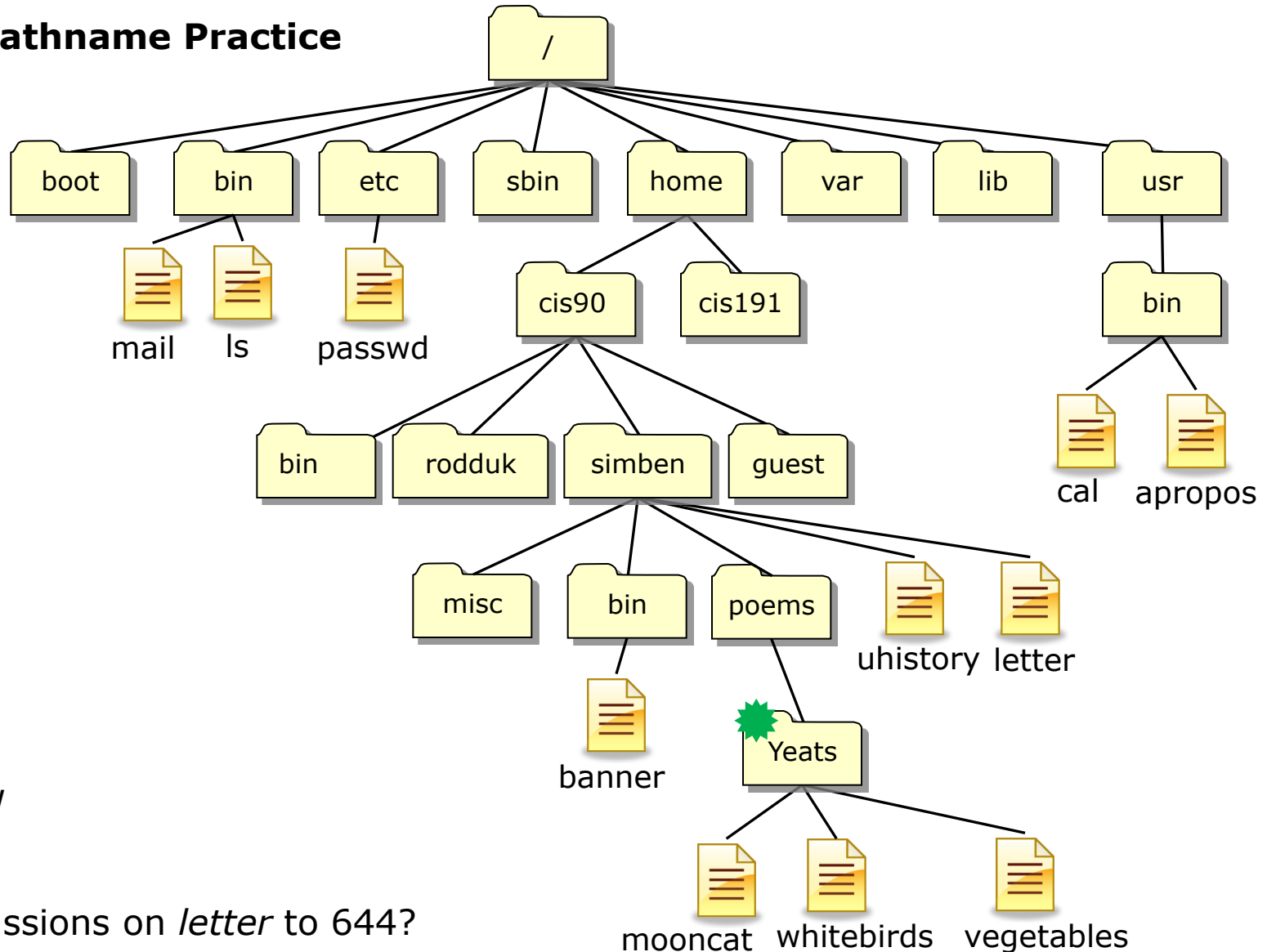
From ✹ how does Benji:

Change permissions on *letter* to 644?

`/home/cis90/simben/poems/Yeats $` **chmod 644 ../../letter**

File Tree Pathname Practice
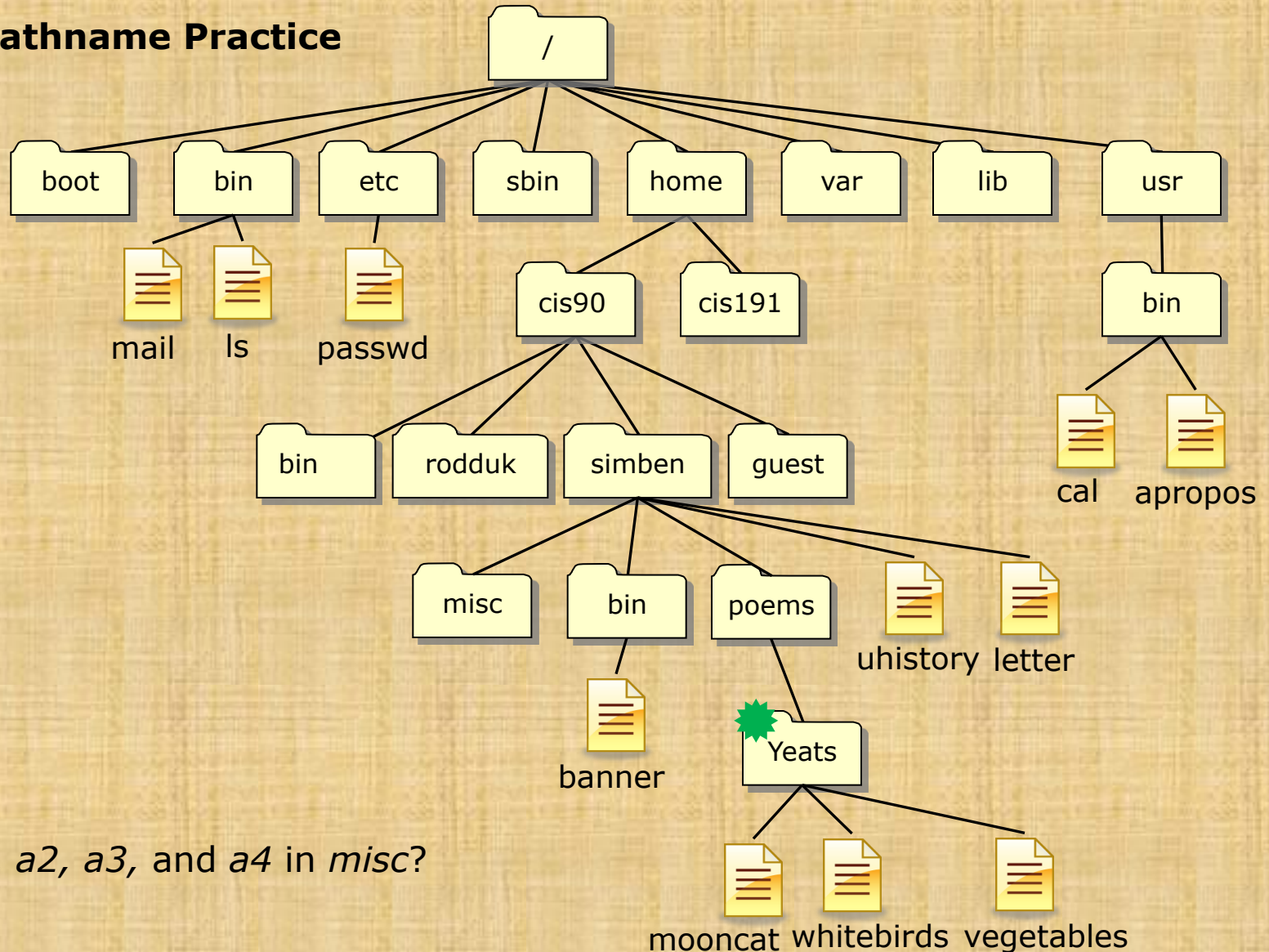
From ✳ how
does Benji:

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

File Tree Pathname Practice
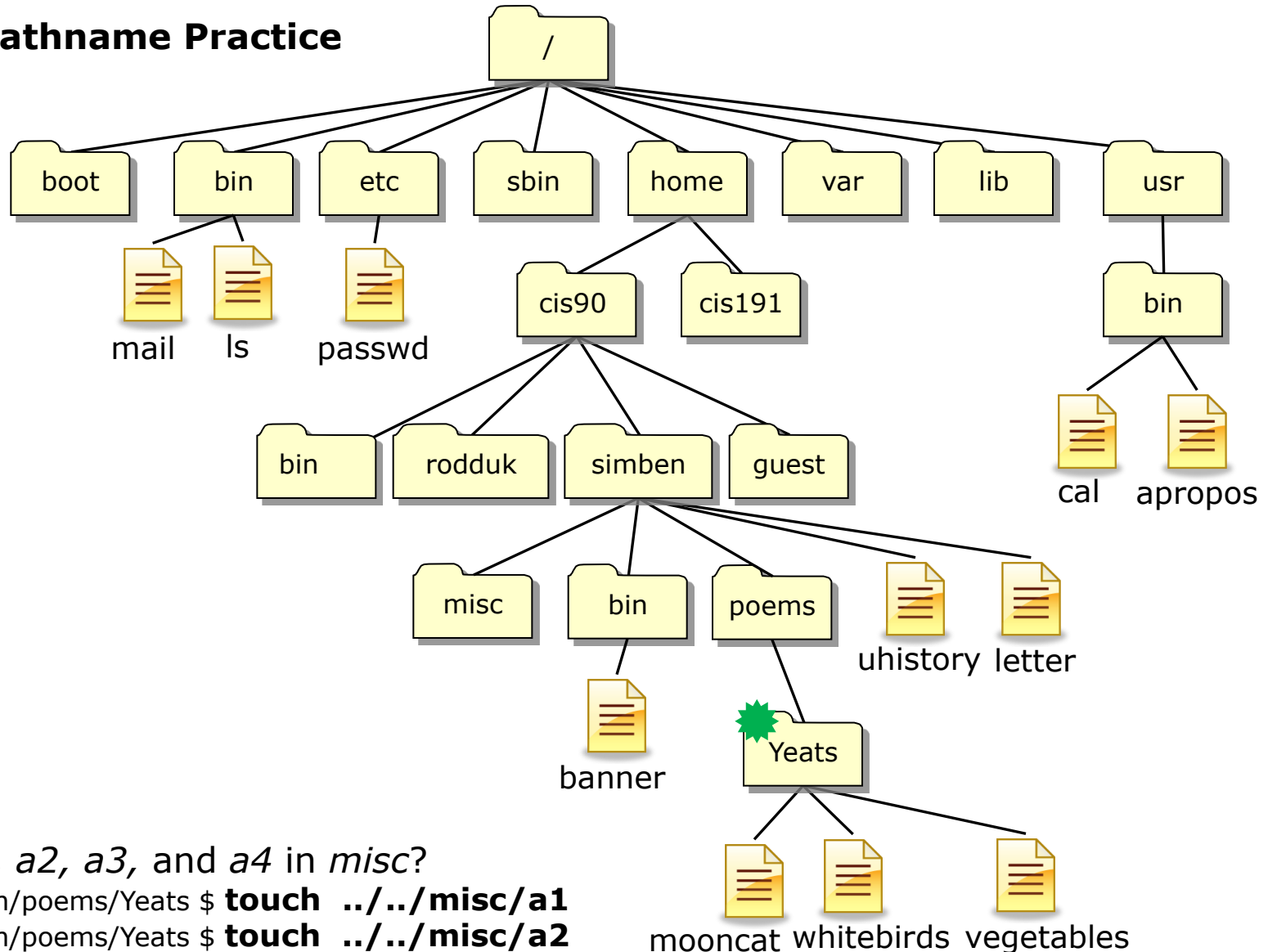
/
boot   bin   etc   sbin   home   var   lib   usr

mail   ls   passwd

cis90   cis191

bin

bin   rodduk   simben   guest

cal   apropos

misc   bin   poems   uhistory   letter

banner   Yeats

mooncat   whitebirds   vegetables

From ✸ how
does Benji:

Create files *a1, a2, a3,* and *a4* in *misc*?
/home/cis90/simben/poems/Yeats $ **touch ../../misc/a1**
/home/cis90/simben/poems/Yeats $ **touch ../../misc/a2**
/home/cis90/simben/poems/Yeats $ **touch ../../misc/a3**
/home/cis90/simben/poems/Yeats $ **touch ../../misc/a4**

34

FYI
only

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

File Tree Pathname Practice
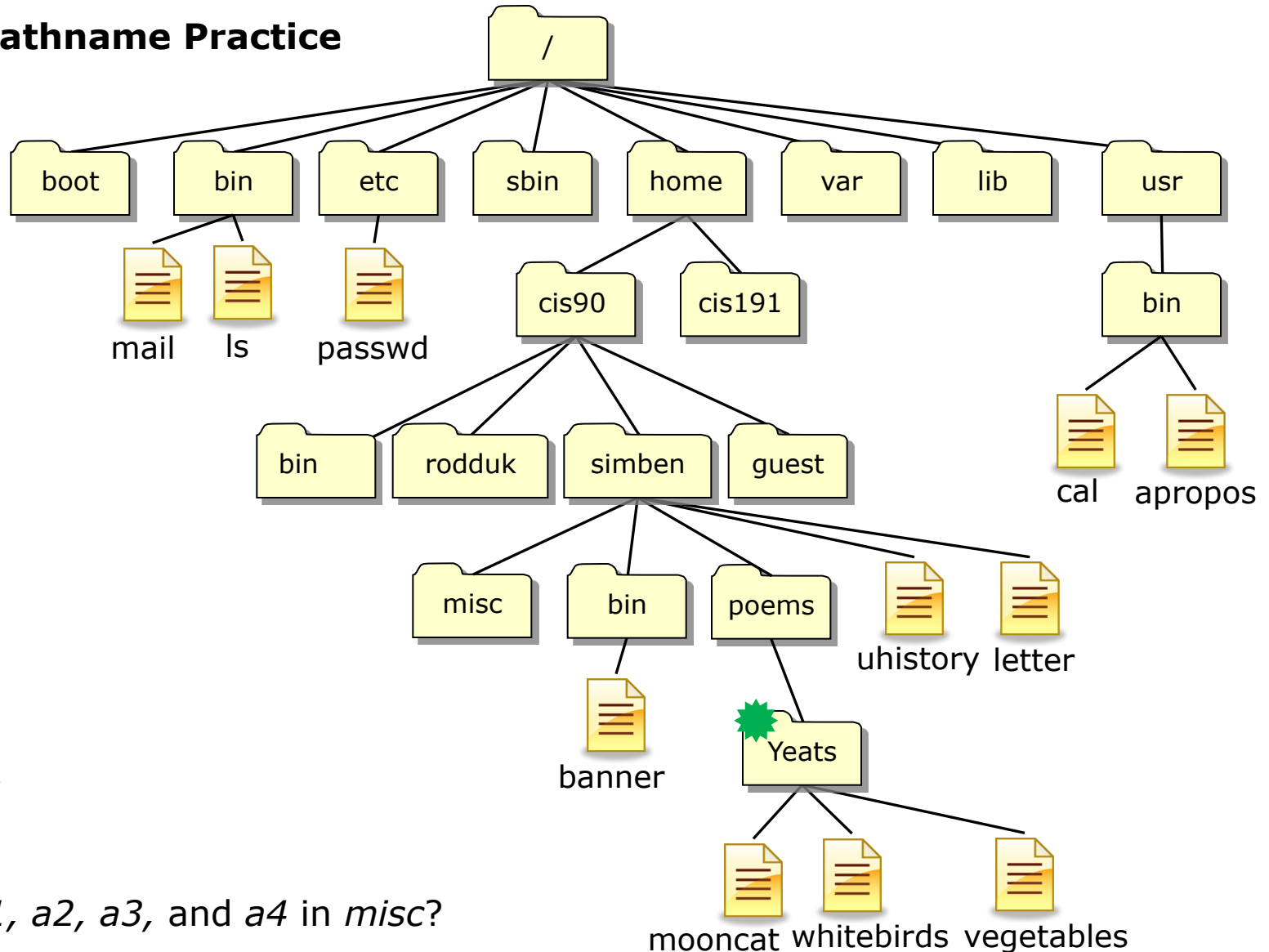
FYI
only

From ✸ how does Benji:

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

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

Permissions
"The rest of the story"

- Special Permissions
- ACLs
- Extended Attributes
- SELinux

FYI only

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

Special Permissions

FYI only

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

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

Special Permissions

FYI only

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

green background with black text

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

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

set sticky bit

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

sticky bit set

green background with black text

39

Special Permissions

FYI only

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

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

red background with gray text

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

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

40

ACLs (Access Control Lists)

FYI
only

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

ACLs (Access Control Lists)

FYI only

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

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

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

Create a file and set permissions to 444

*Use **getfacl** to show ACLs*

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

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

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

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

42

ACLs (Access Control Lists)

FYI only

Let's give special permissions to one user

modify

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

Allow milhom90 to have read/write access

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

Homer can now read the file

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

But not Duke

43

ACLs (Access Control Lists)

FYI only

Let's remove the special permissions to that user

.... *remove all base ACLs*

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

Remove all ACLs on yogi file

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

Now Homer can't read it again

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

Same for Duke

44

Extended File Attributes

FYI
only

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

Extended File Attributes

FYI only

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

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

Create a sample file to work on

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

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

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

!!

Extended File Attributes

FYI only

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

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

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

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

47

FYI only

Extended File Attributes

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

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

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

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

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

48

SELinux context

FYI only

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

SELinux context

FYI only

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

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

user   *role*   *type*   *level*

SELinux context

FYI only

Create two identical web pages with identical permissions

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

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

Use chcon command to change the SELinux context on one file

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

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

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

51

SELinux context

FYI only

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

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

test01.html

test02.html

type = httpd_sys_content_t

type = home_root_t

52
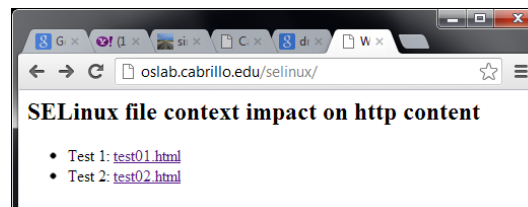
Housekeeping

Previous material and assignment

1. Lab 6 due 11:59PM
   - **check6** script available

**Don't forget to submit
with the submit script!**

2. Five posts due 11:59PM

3. Early preview of Lab X2 is now available

Grades Web Page

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

Please check your:
- Grading Choice
- Quiz points
- Forum points
- Test points
- Lab points
- Extra Credit points

Don't know you secret LOR code name?

… then email me your student survey to get it!

Help with labs

Like some help with labs?

I'm in the CIS Lab Monday afternoons
• See schedule at http://webhawks.org/~cislab/

or see me during office hours

or contact me to arrange another time online

umask

Why umask?

Why umask?

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

*It does not change the permissions on existing files or directories. We already have a command to do that and it's called **chmod***

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

60

CASE 1: When new files are created

1. New files temporarily start with 666 permissions

2. New directories temporarily start with 777 permissions

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

Example: umask 002

Interpret **umask 002** as "From now on, strip write permission from others for all new files and directories created"

```
rw- rw- rw- (666)  starting point for files
--- --- -w- (002)  umask setting
rw- rw- r-- (664)  the permissions a new file will have
```

Prove it to yourself using Opus:

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

/home/cis90/simben $ umask      Note: the mnemonic form
0002                            of 002 is --- --- -w-

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

write permission for others has been stripped off

62

Example: umask 027

Interpret **umask 027** as "From now on, strip write permission from group, and strip all permissions from others for all new files and directories created"

```
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
```

CASE 2: When files are copied

1. When a new file is created by copying, the starting point is NOT 666!

2. When a new directory is created by copying, the starting point is NOT 777!

3. The starting point instead is the set of permissions on the file being copied

cp <sourcefile> <targetfile>

The permissions of the new target file are obtained by applying the umask value to the permissions of the source file

64

Example of umask value on copying files

```
/home/cis90/simben $ touch snap1 snap2
/home/cis90/simben $ chmod 777 snap1
/home/cis90/simben $ chmod 622 snap2
/home/cis90/simben $ ls -l snap*
-rwxrwxrwx. 1 simben90 cis90 0 Oct 14 15:40 snap1
-rw--w--w-. 1 simben90 cis90 0 Oct 14 15:40 snap2

/home/cis90/simben $ umask 222

/home/cis90/simben $ cp snap1 crackle1
/home/cis90/simben $ cp snap2 crackle2
/home/cis90/simben $ ls -l crackle*
-r-xr-xr-x. 1 simben90 cis90 0 Oct 14 15:43 crackle1
-r--------. 1 simben90 cis90 0 Oct 14 15:43 crackle2
```

Create two example files with different permissions

When a file is copied, the umask is applied to the permissions of the source file (instead of 777 or 666)

```
rw- -w- -w- (622)  snap2
-w- -w- -w- (222)  umask
r-- --- --- (400)  crackle2
```

```
rwx rwx rwx (777)  snap1
-w- -w- -w- (222)  umask
r-x r-x r-x (555)  crackle1
```

65

Sample umask test question

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

Tools for your toolbox

sort - sorts input from a file or stdin and writes output to stdout

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)*

70

Input and Output
File Descriptors

Example program: sort command

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

ctrl D

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)

71

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
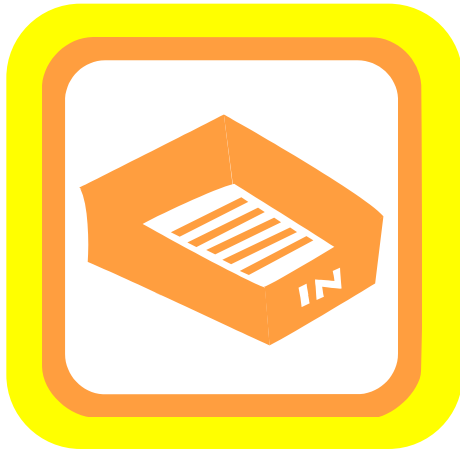
A day in the life of a process

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

74
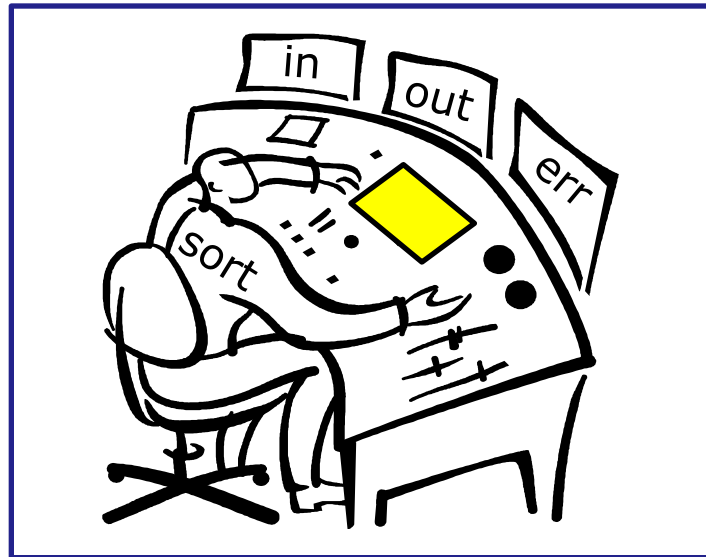
sort process example no args

/home/cis90/simben $ **sort**
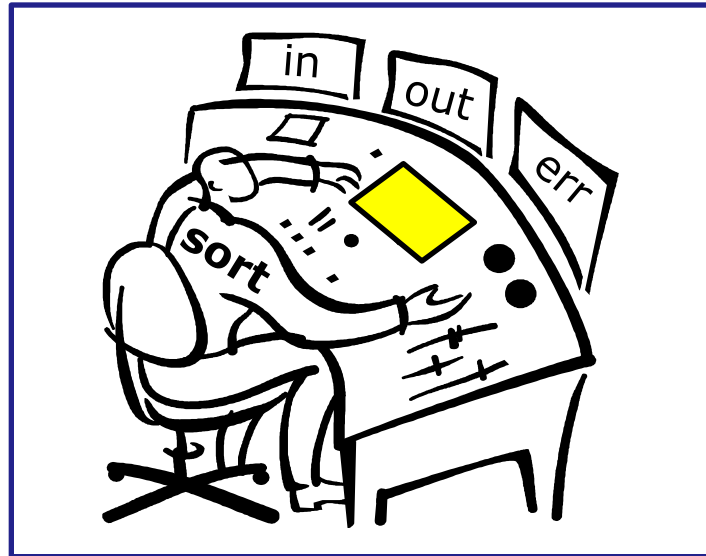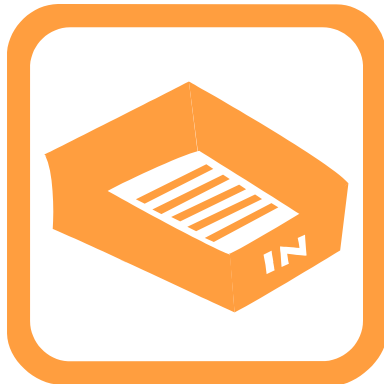
You (the sort process) check your instruction window and see that no options or arguments were given to you to handle. You know (given your internal DNA) that with no arguments you must looks for lines to sort in your in tray, so you reach in to grab the first line to sort.

76

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

charlie benji bella sky kayla

in
out
err

sort

IN

OK   OUT

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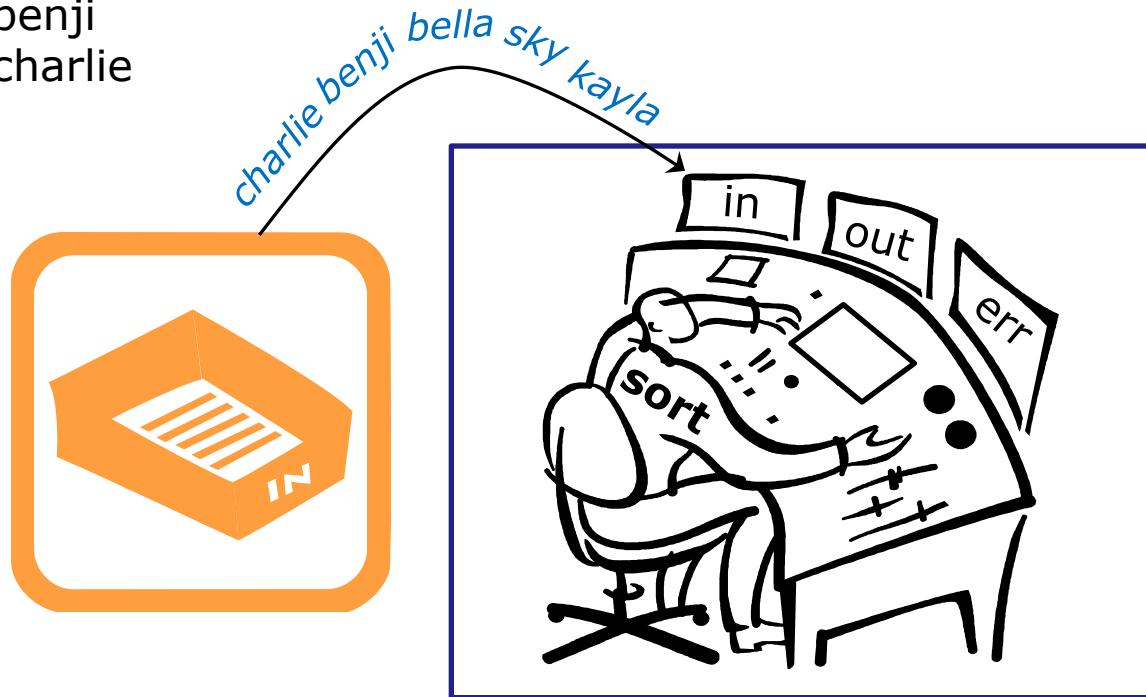
Note: You work hard and fast. Every time your reach into the in tray there is another line for you. They just magically keep appearing from somewhere into your in tray. You have no idea where they are coming from.

77

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

EOF

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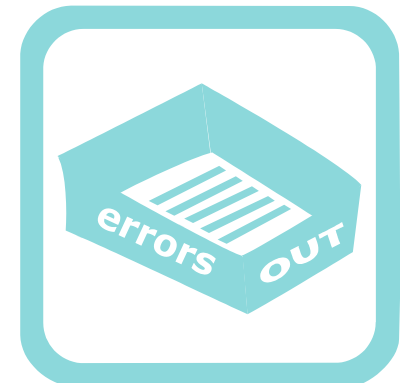
Then suddenly, when you reach into the in tray, instead of another line you find an EOF. You know (your internal DNA code) that this EOF means there are no more lines coming. You must sort what you have collected so far and place them, in order, into your out tray.

78

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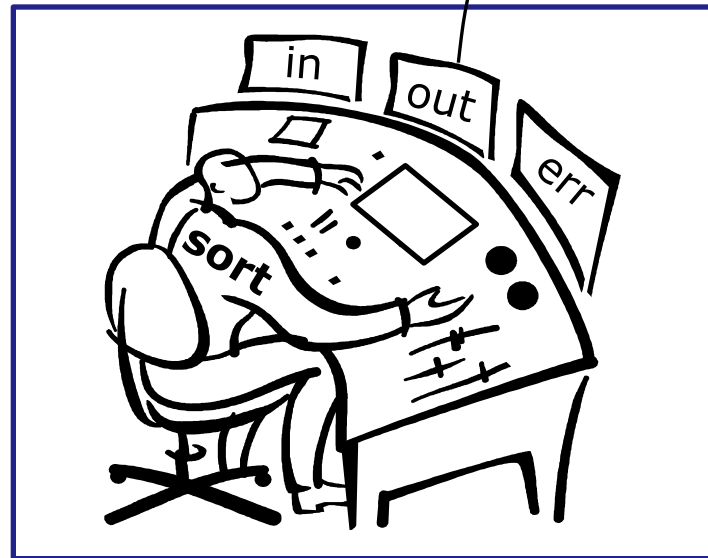
sky kayla charlie benji bella

in

out

err

sort

ok OUT

IN

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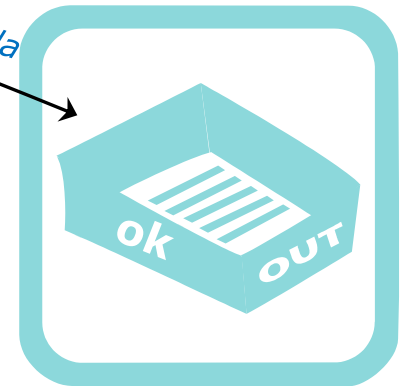
As fast as you can, you sort them, and place then in order in your out tray. They keep getting removed magically from the out tray. You have no idea where they go.

79

sort process example bad arg

/home/cis90/simben $ **sort bogus**
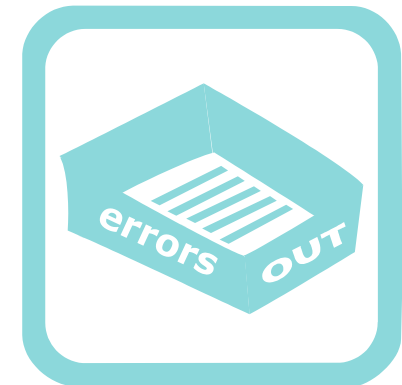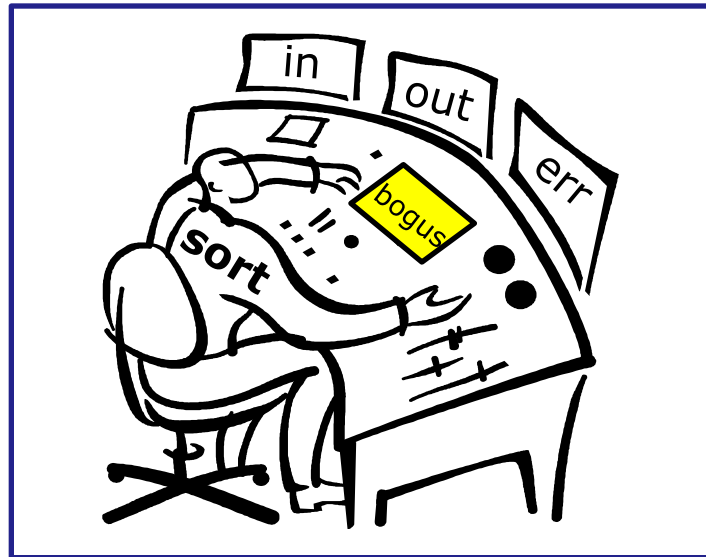
You check your little instruction window and see an argument (bogus). You know (your internal DNA) tells you this must be a file name containing lines to sort.

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

sort: open failed: bogus:
No such file or directory
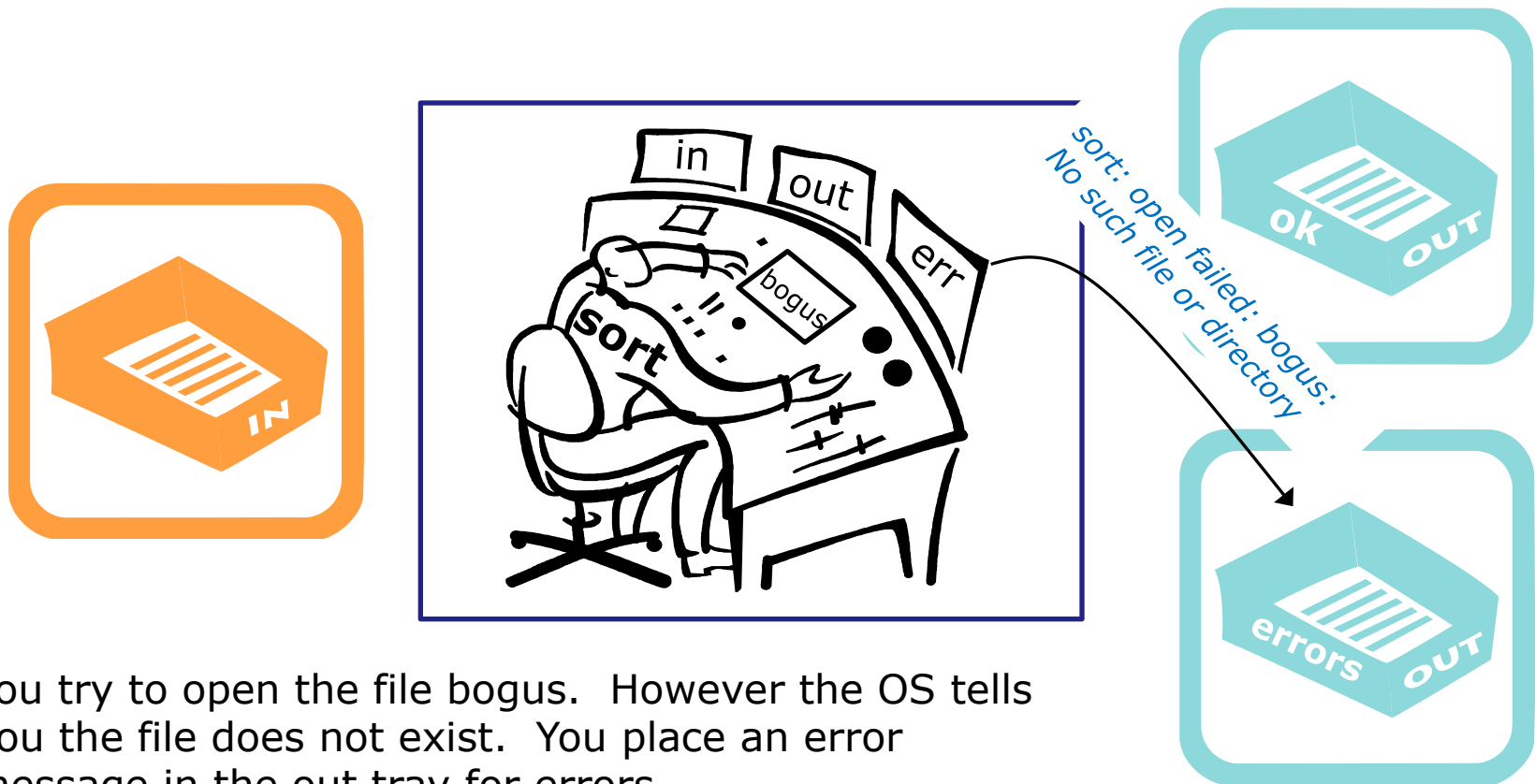
You try to open the file bogus. However the OS tells you the file does not exist. You place an error message in the out tray for errors.

bringing it home

Ok, lets make the visualization a little more realistic

stdout (1)
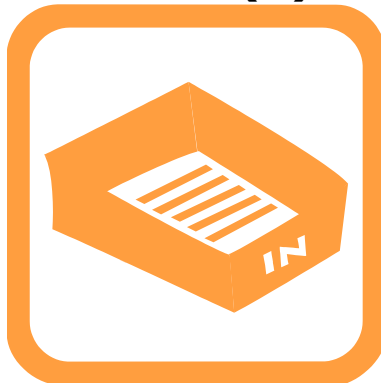
stdin (0)

stderr (2)

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

84

Input and Output
File Descriptors

stdout (1)

Now lets start to show the connections as "pipes".
More on this later.
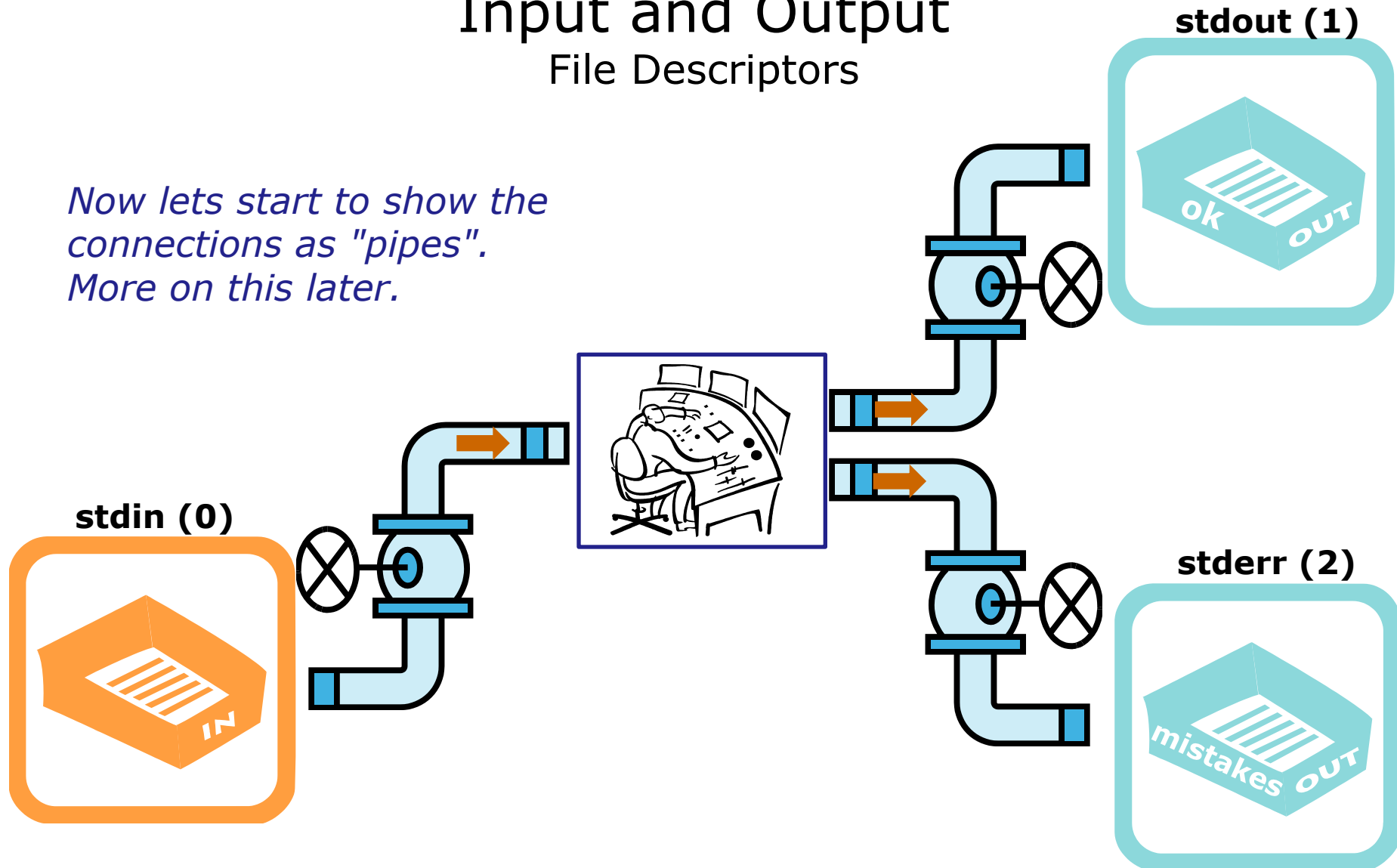
stdin (0)

stderr (2)

Input and Output
File Descriptors

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

stdout

normal output is written to stdout
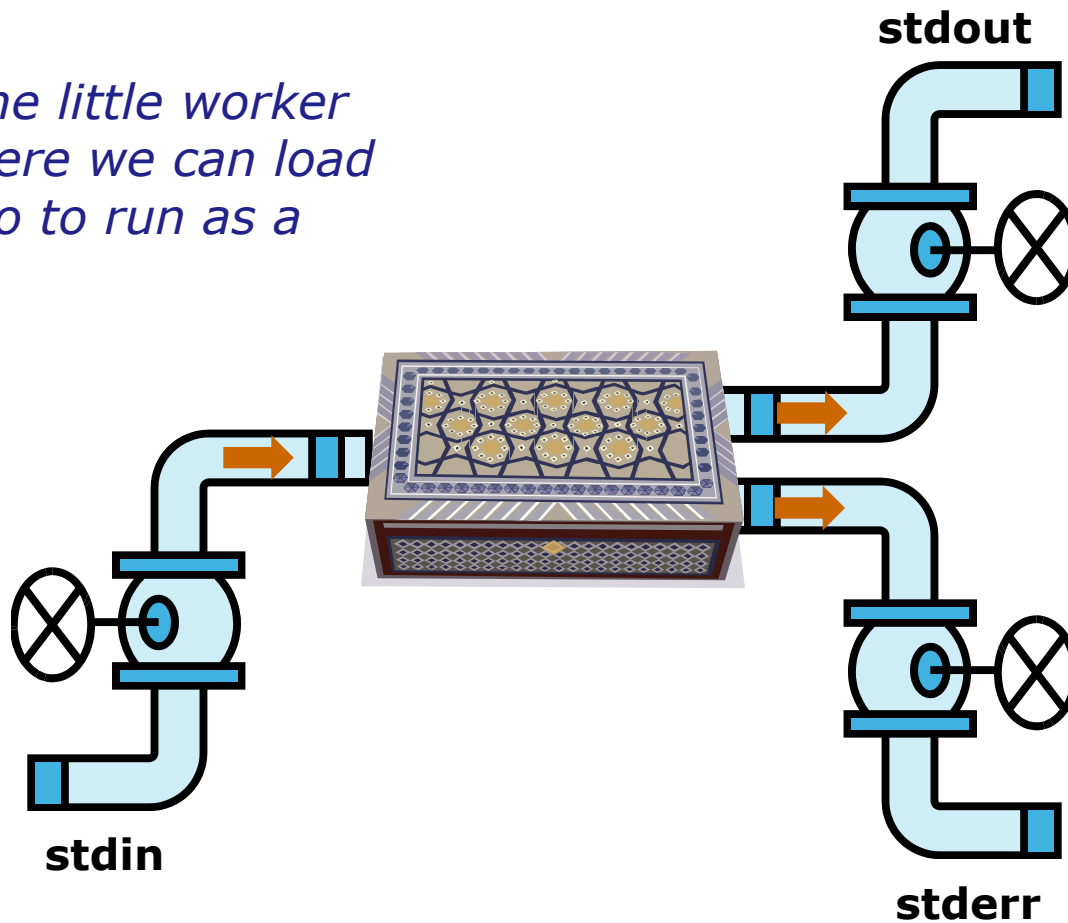
input (if necessary) is read from stdin

stdin

stderr

errors are written to stderr

86

Input and Output
File Descriptors

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

stdout

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

0

1

2

stdin

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

stderr

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

87

Input and Output
File Descriptors

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

stdout

ctrl D

Options: NA
Args: NA

0 sort 1
           2

benji
duke
homer
star
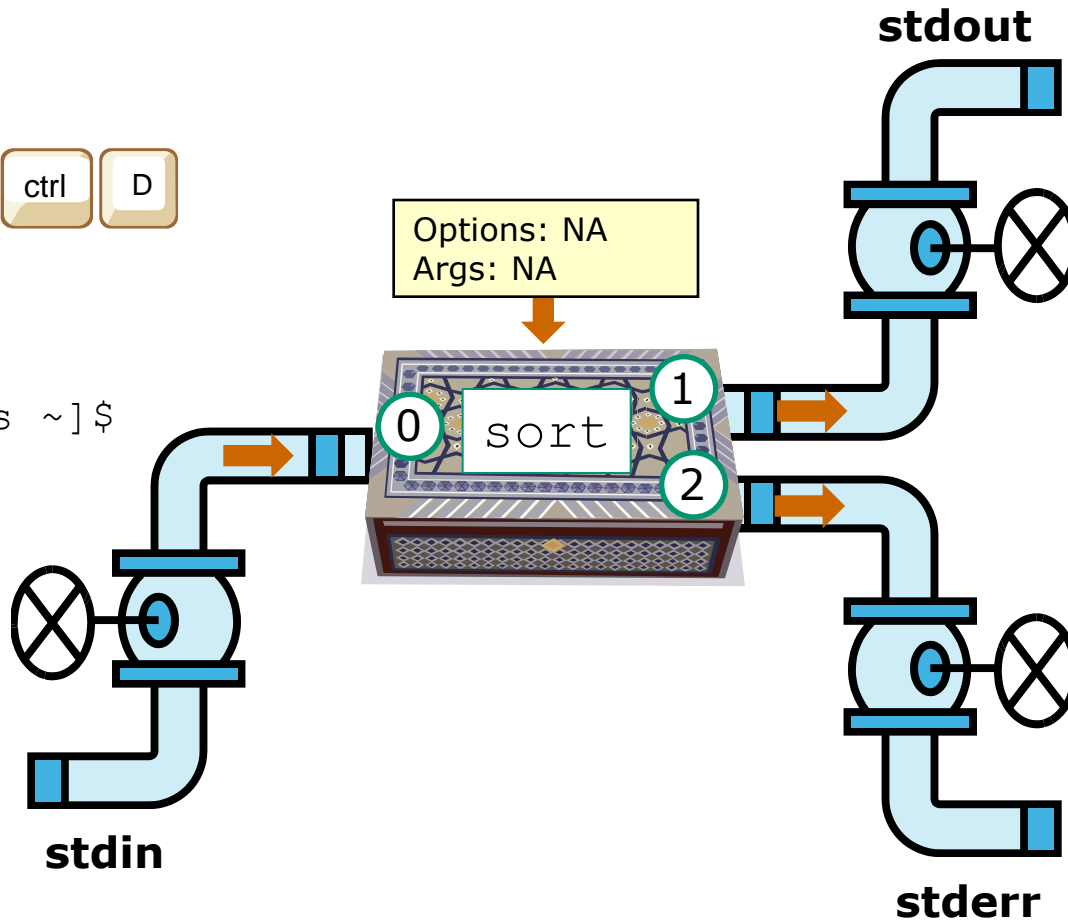
stdin

star
benji
duke
homer

stderr

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

88

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 much more **EXCITING**!*

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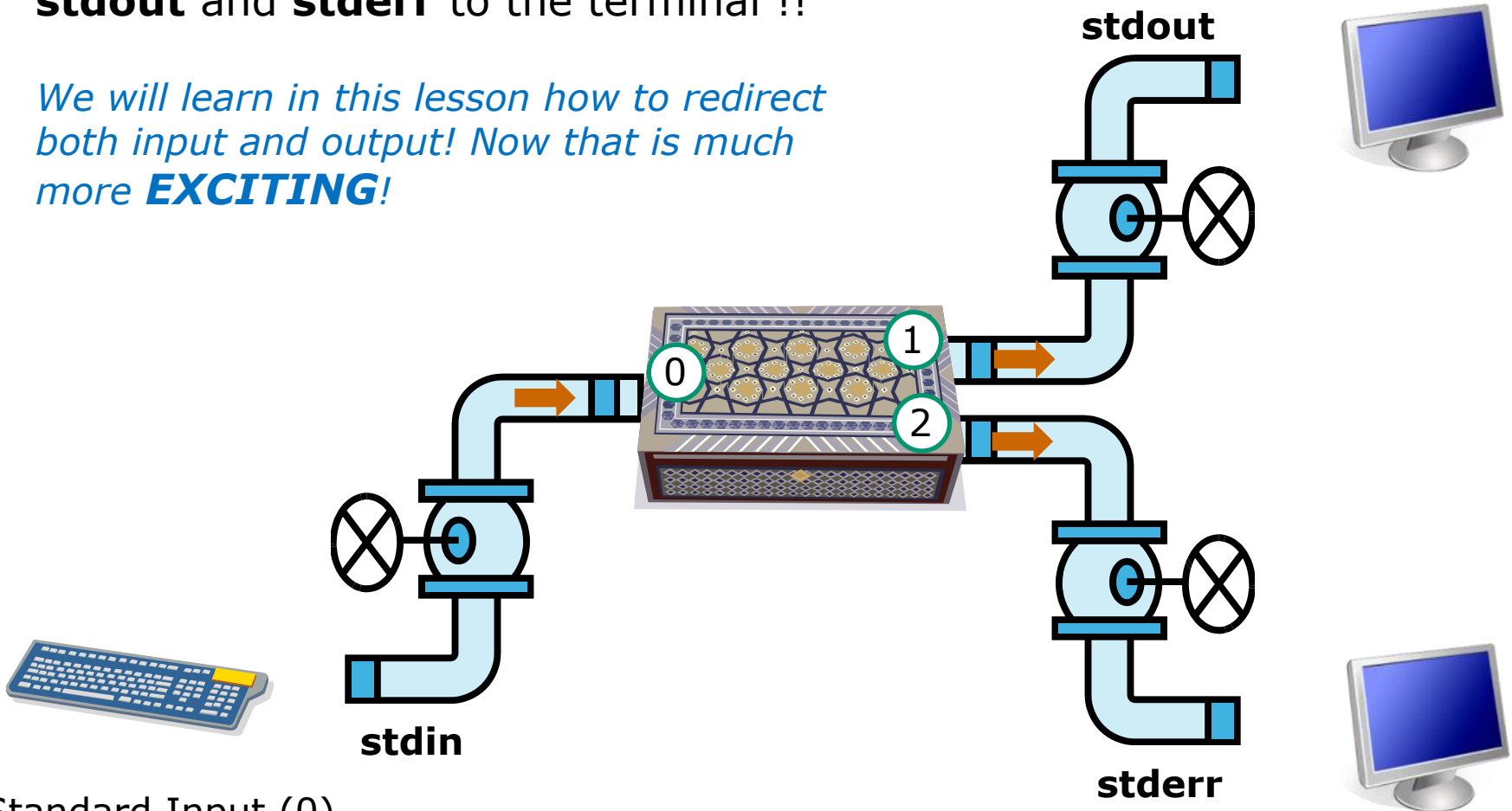
stdout

stdin

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

stderr

Standard Error (2)

90

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 $
```

ctrl D

End of File

91

Input and Output
File Redirection

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

Read from **stdin**
```
duke
benji
star
homer
```

ctrl D

"End of File"
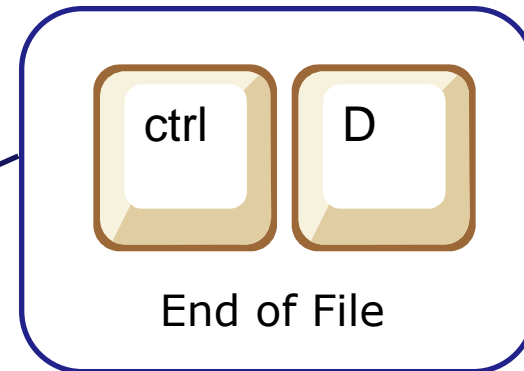
Written to **stdout**
```
benji
duke
homer
star
```
/home/cis90/simben $
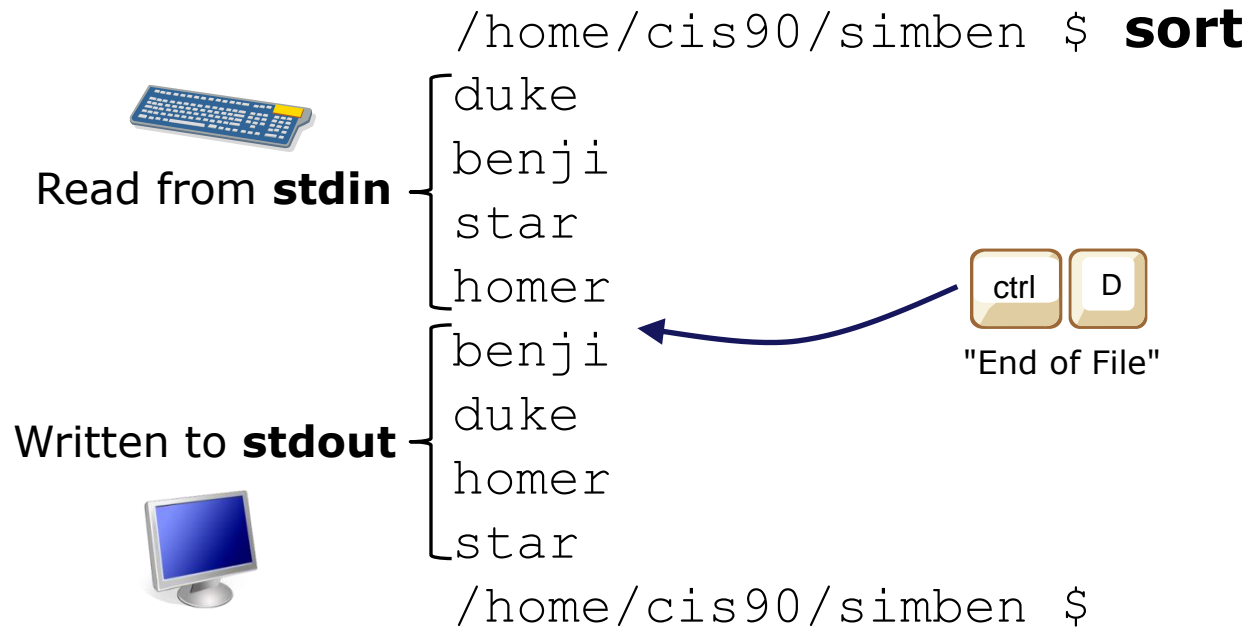
*The sort program reads lines from **stdin** (attached to keyboard), performs the sort, then writes to **stdout** (attached to terminal)*

92

Example program to process: sort command

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

ctrl D

/dev/pts/0

stdout

Options: NA
Args: NA

sort

0 1 2

benji
duke
homer
star

/dev/pts/0

stdin

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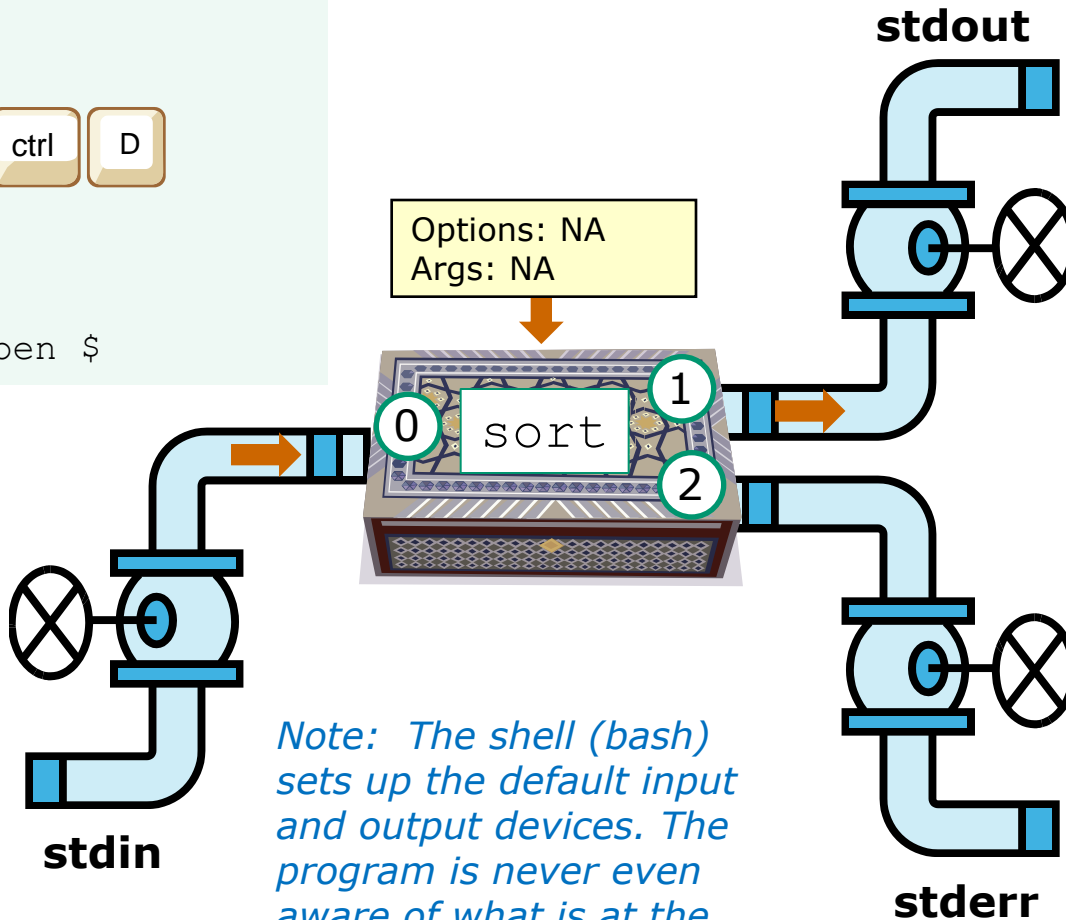
Note: The shell (bash) sets up the default input and output devices. The program is never even aware of what is at the end of the pipes.

stderr

93

Activity

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

Read from **stdin**
```
duke
benji
star
homer
```

ctrl D

"End of File"

Written to **stdout**
```
benji
duke
homer
star
/home/cis90/simben $
```
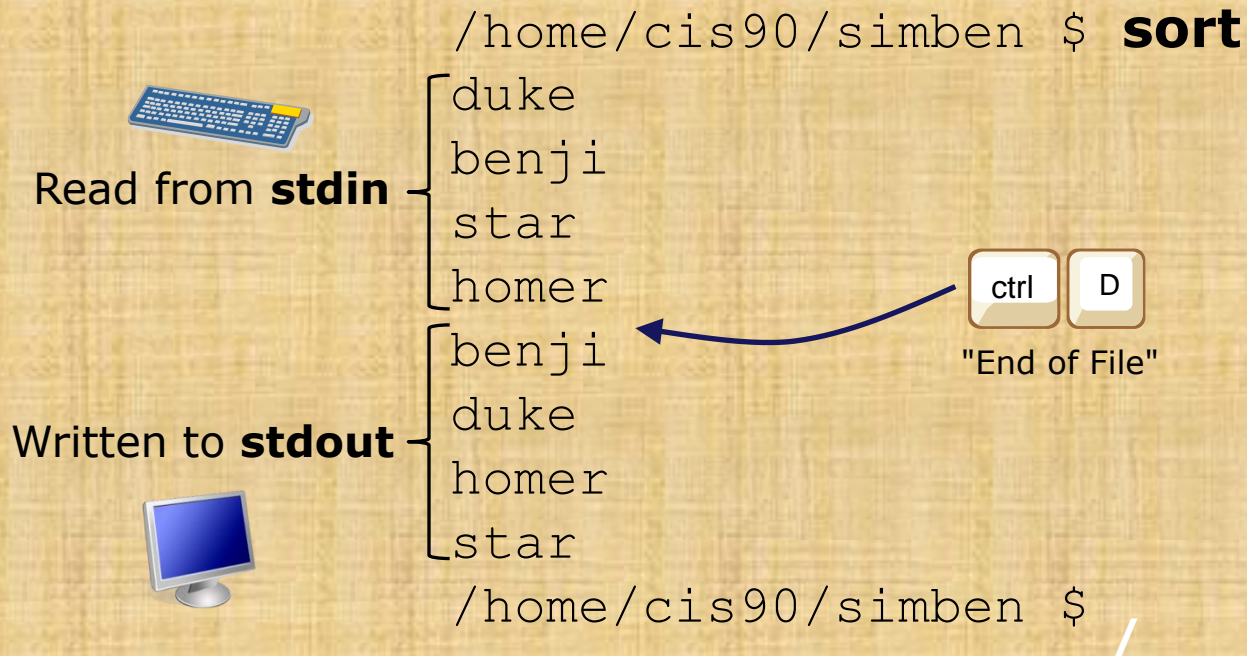
Now you try it with your own list

94

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*

To redirect stdin

1> *filename*

To redirect stdout

2> *filename*

To redirect stderr

>> *filename*

To redirect and append from stdout

95

Input and Output
File Redirection

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

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

The 0 is optional **0< filename**

> Input will now come from *filename* rather than the keyboard.

The 1 is optional **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*.

96

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!

ctrl D

97

Example program to process: sort command

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

ctrl D

stdout

Options: NA
Args: NA

dogsinorder

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

0 sort 1
          2

/dev/pts/0

stdin

stderr

duke
benji
star
homer

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

98

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.
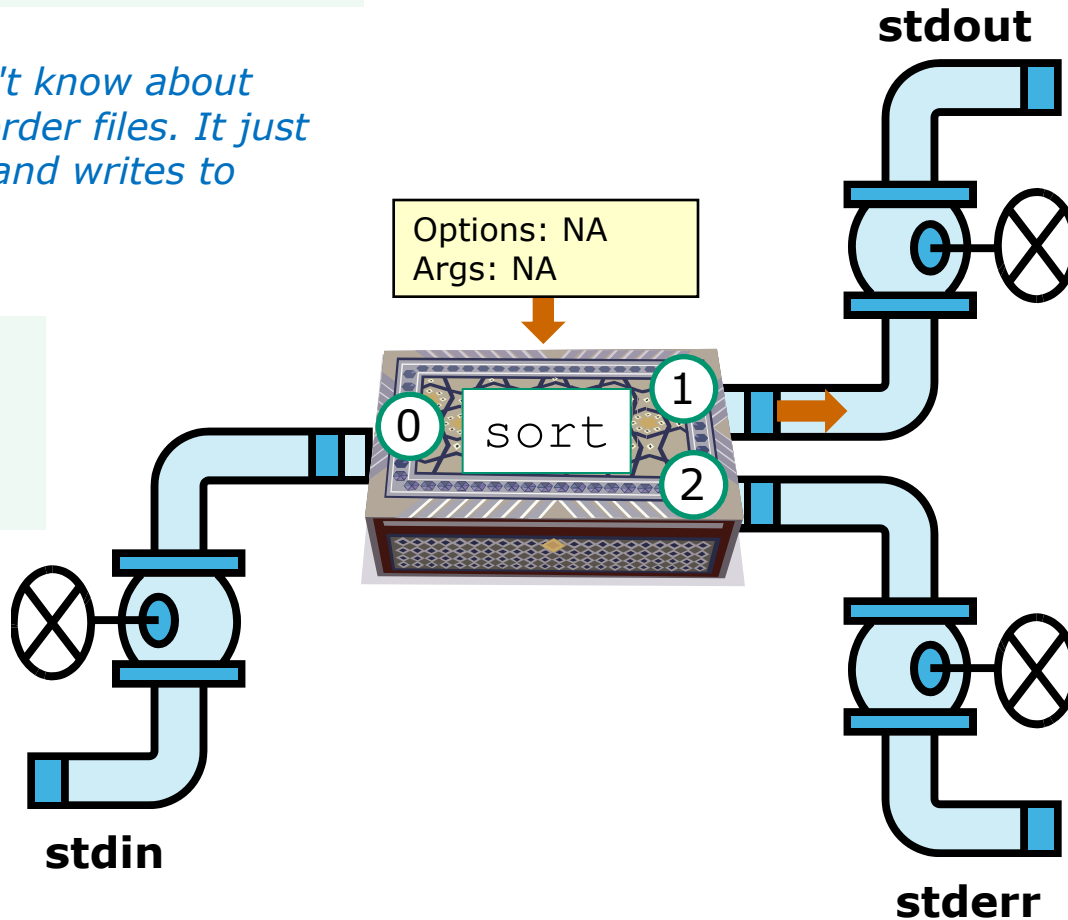
stdout

Options: NA
Args: NA

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


sort

0

1

2

dogsinorder

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

stdin

names

stderr

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

Input and Output
File Redirection

Output is redirected to the file dogsinorder.

*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.*

*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.

102

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.
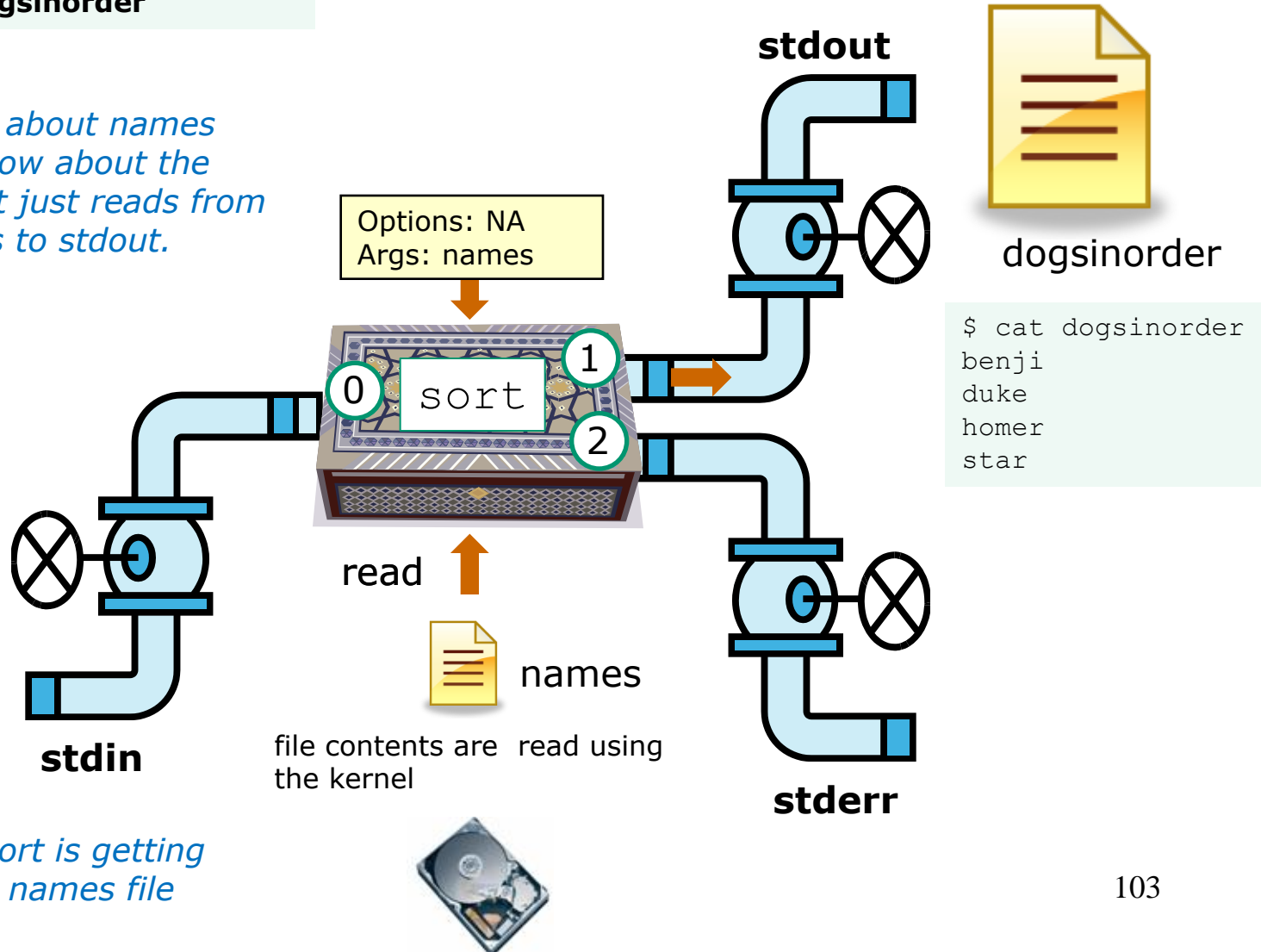
stdout

Options: NA
Args: names

0 sort **1**
2

read

names

file contents are read using
the kernel

stdin

dogsinorder

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

stderr

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

103

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

104

Example program to process: sort command

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

stdout

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.

Options: -r
Args: names

dogsinorder

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

0 sort 1

2

read

names

file contents are read using the kernel

stdin

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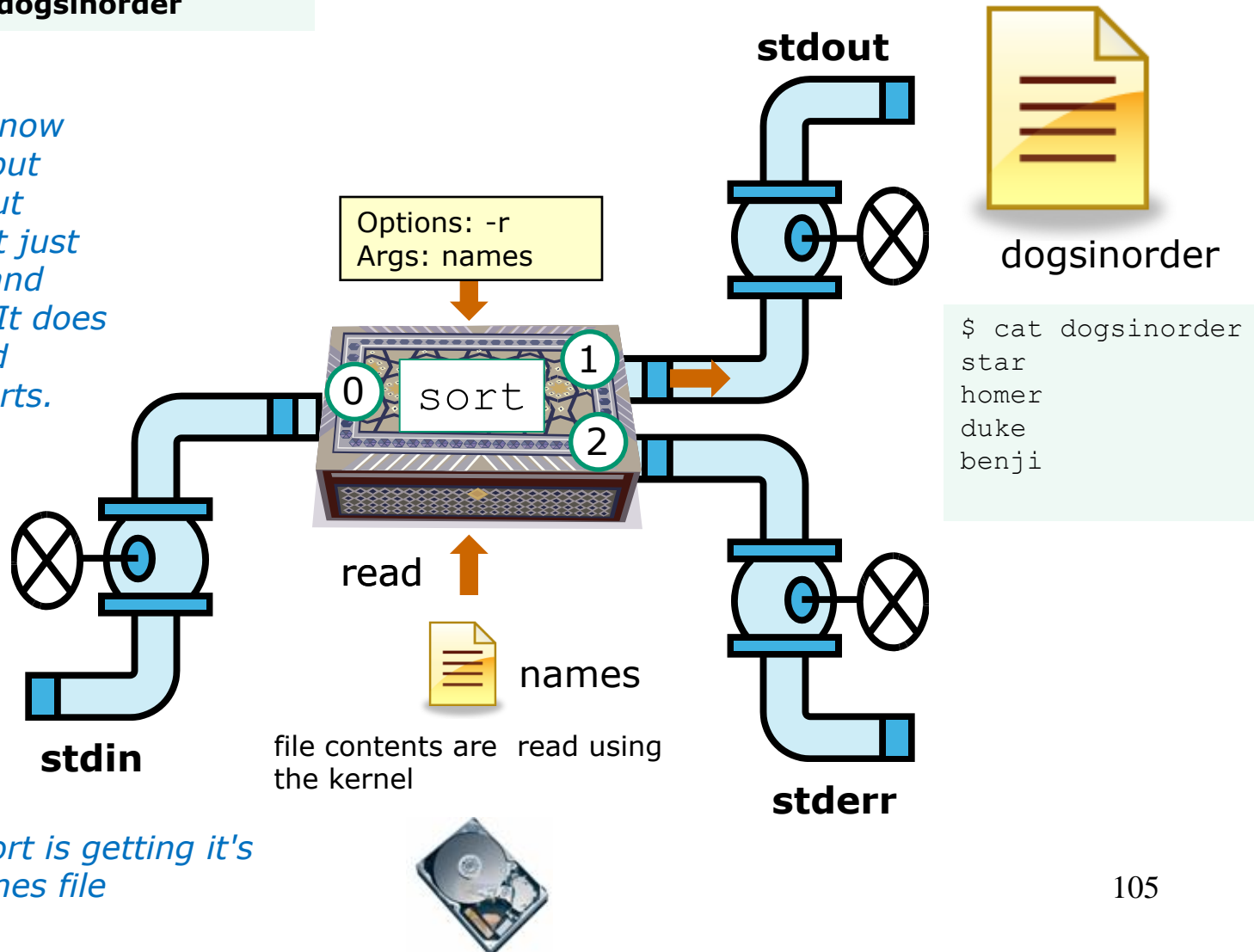
In this example, sort is getting it's input from the names file

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
```

106

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!*

107

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.*

stdout

Options: NA
Args: "Hello World"

1
0 echo
2

message

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

stdin

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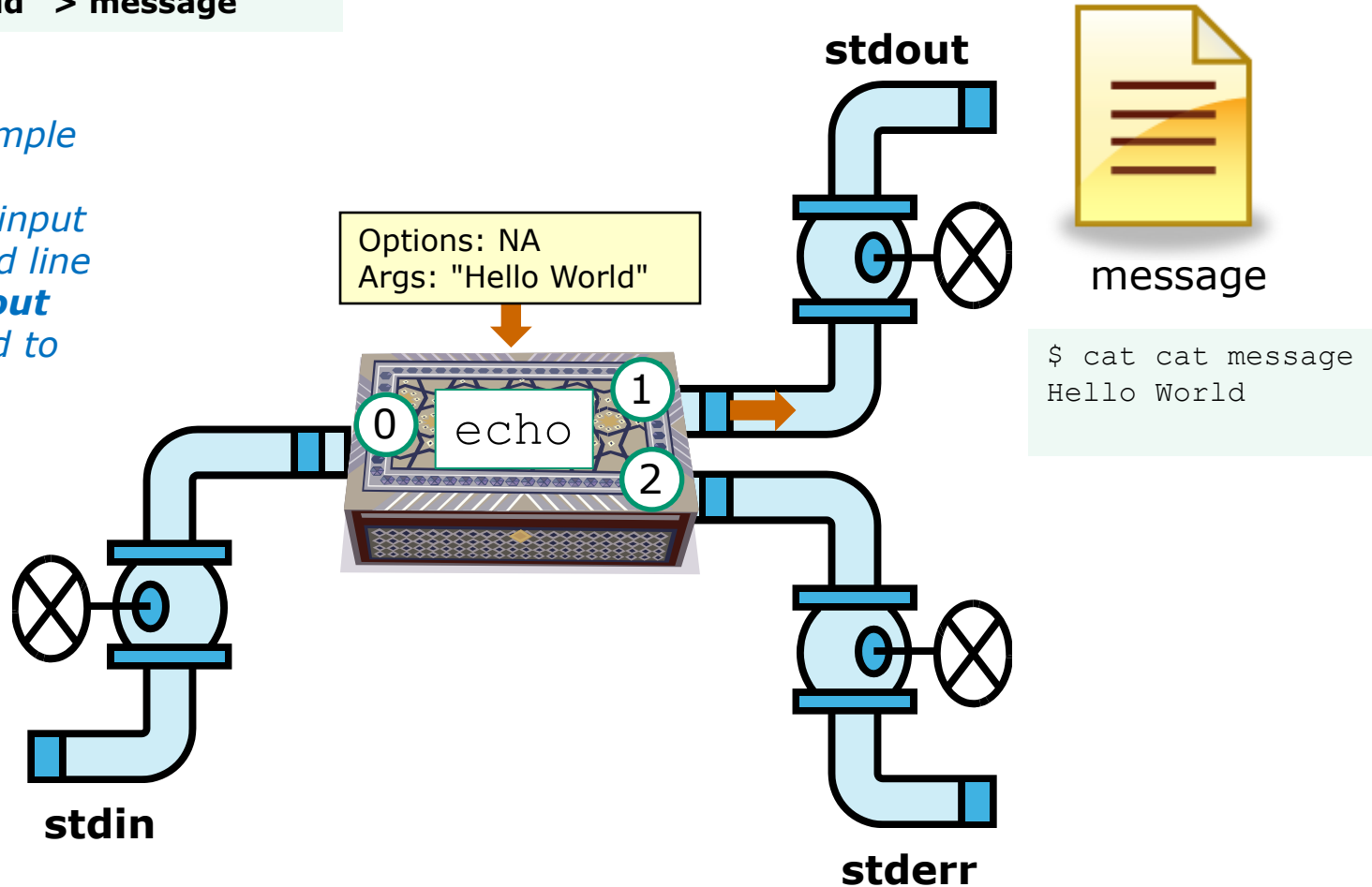
In this example, echo is getting it's input from the command line

108

Input and Output
File Redirection

Another example …

Note: errors are written to stderr, which defaults to the terminal

```
[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 ~]$
```

> redirects stdout to file named snapshot

2> redirects stderr to file named errors

109

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).*

stdout

Options: -lR
Args: NA

```
[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
```

snapshot

`ls`

0 **1** **2**

read

stdin

directory contents are read using the kernel

In this example, ls is getting it's input from the OS

stderr

errors

110

```
$ 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**)*

111

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*

stdout

Options: NA
Args: NA

1

0   bc

2

answers

4

stdin

math

2+2
4/0

stderr

errors

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

112
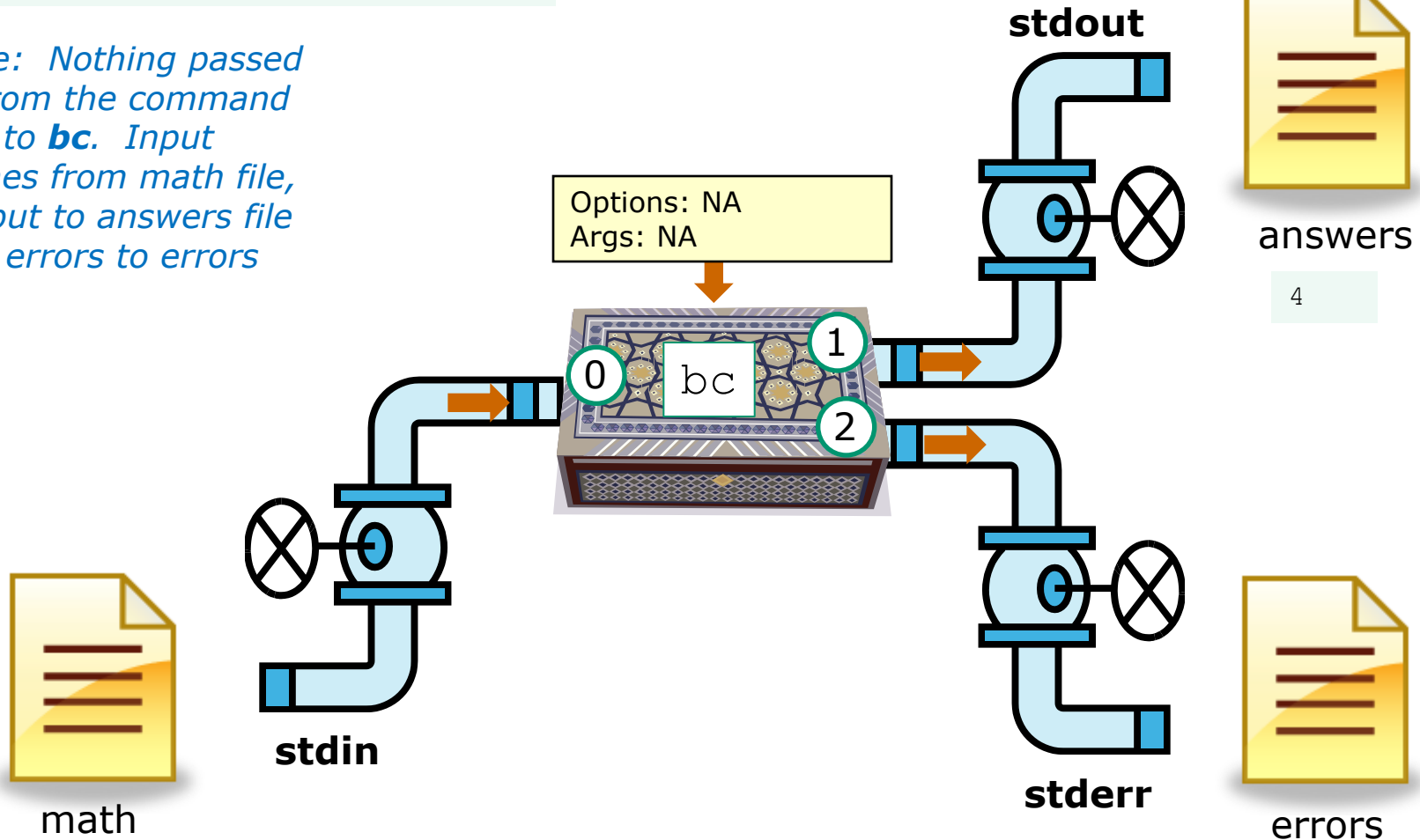
The bit bucket

/dev/null

/dev/null = "bit bucket"

A bit bucket is very handy. You can throw whatever you want into it and never see it again!

http://www.adri
anmouat.com/bi
t-bucket/

http://didyouk
nowarchive.co
m/?p=1755
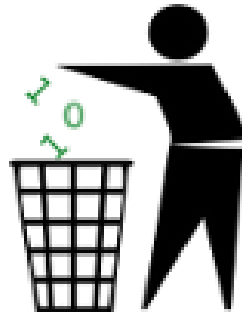
It's like having your own black hole to discard those unwanted bits into!

114

/dev/null = "bit bucket"

Whatever you redirect to the device file above you will never see again

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

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

115

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.

117

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
```

stdout

28

Options: -l
Args: NA

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

0 wc 1
       2

stdout

stdin

stderr

Options: NA
Args: letter

0 cat 1
        2

read

letter

file contents are read using the OS

stdin

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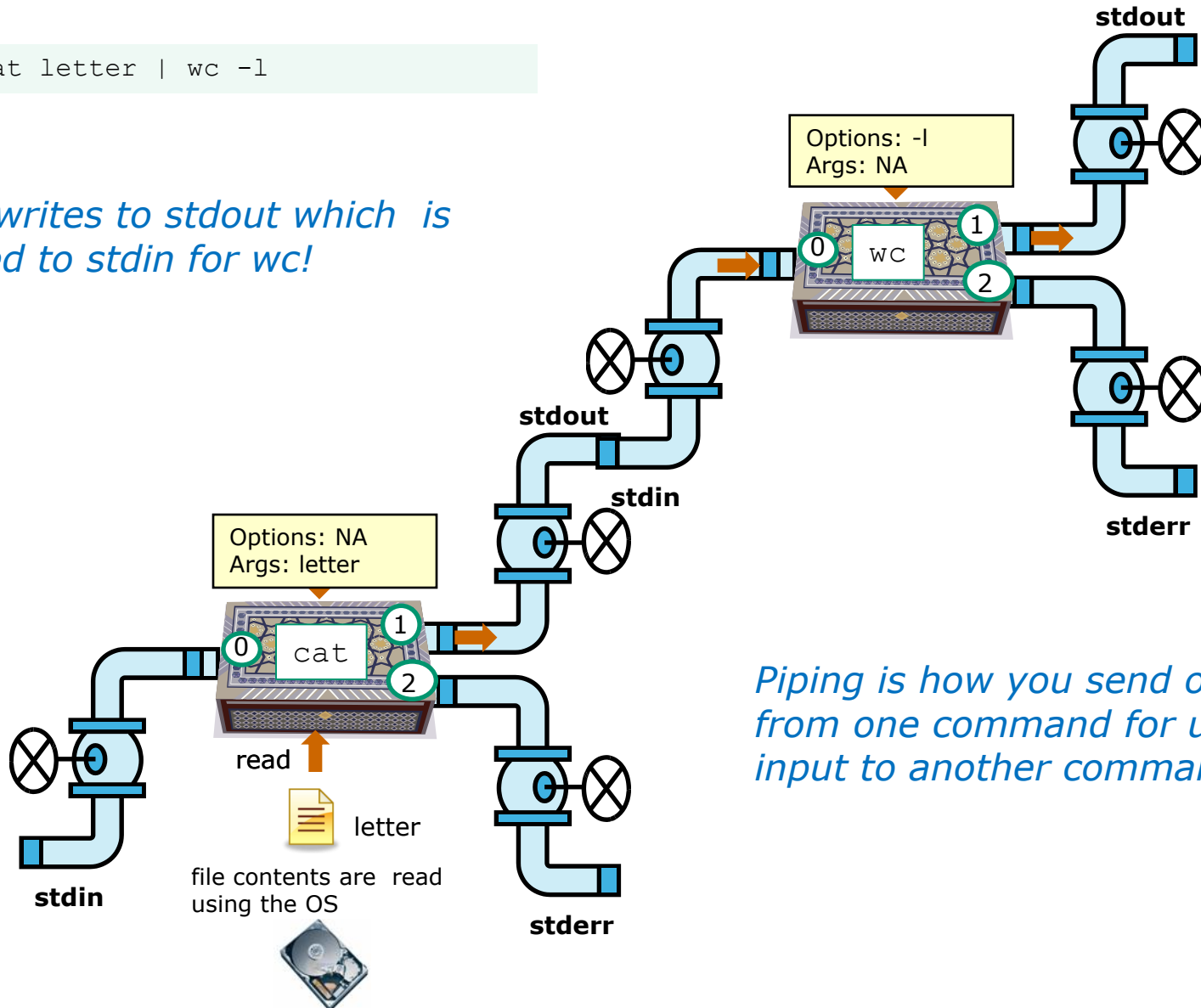
Piping is how you send output from one command for use as input to another command

119

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***

Why pipelines?

Task: Save a sorted list of users and a count of how many users are logged on

Method I - use intermediate temporary files

```
[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)
```

121

Why pipelines?

Method II - uses pipes

```
[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 ~]$
```

*Same result as Method 1 but accomplished on a single line
with no intermediate files to clean up*

Building a pipeline one command at a time

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)
```

123

Miscellaneous Commands

Tools for your toolbox

find - Find file or content of a file

grep - "Global Regular Expression Print"

sort - sort

spell - spelling correction

wc - word count

tee - split output

find command

Find Command

Syntax:

find *<search-directory>* **-name** *<filename>*
                              -type *<filetype>*
                              -user *<username>*
                              -exec *<command>* **{} \;**

*The **find** command can be used to search for files from any point in the UNIX file tree and continue recursively down the tree as far as it goes.*

find command with no options or arguments

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

```
[simben@opus poems]$ find
.
./Blake
./Blake/tiger
./Blake/jerusalem
./Shakespeare
./Shakespeare/sonnet1
./Shakespeare/sonnet2
./Shakespeare/sonnet3
./Shakespeare/sonnet4
./Shakespeare/sonnet5
./Shakespeare/sonnet7
./Shakespeare/sonnet9
./Shakespeare/sonnet10
./Shakespeare/sonnet15
./Shakespeare/sonnet17
./Shakespeare/sonnet26
./Shakespeare/sonnet35
./Shakespeare/sonnet11
./Shakespeare/sonnet6
./Yeats
./Yeats/whitebirds
./Yeats/mooncat
./Yeats/old
./Anon
./Anon/ant
./Anon/nursery
./Anon/twister
[simben@opus poems]$
```

find command issued in the poems directory will list the Blake, Shakespeare and Yeats directories and their contents

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

128

Specifying a starting point as an argument

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

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

find command starting from the /etc/ssh directory

129

Using options for search criteria

The -name option can be used select only matching filenames

```
[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"

130

All those permission errors

An error is printed for every directory lacking read permission!

```
[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?

131

Using find command with the bit bucket

the "bit bucket"

This is why we want a bit bucket

```
[simben@opus ~]$ find  /home/cis90 -name sonnet6  2> /dev/null
/home/cis90/guest/Poems/Shakespeare/sonnet6
/home/cis90/simben/poems/Shakespeare/sonnet6
/home/cis90/stanlcha/poems/Shakespeare/sonnet6
/home/cis90/seatocol/poems/Shakespeare/sonnet6
/home/cis90/wrigholi/poems/Shakespeare/sonnet6
/home/cis90/dymesdia/poems/Shakespeare/sonnet6
/home/cis90/lyonsrob/poems/Shakespeare/sonnet6
/home/cis90/ybarrser/poems/Shakespeare/sonnet6
/home/cis90/ybarrser/poems/Sonnets/sonnet6
/home/cis90/valdemar/poems/Shakespeare/sonnet6
/home/cis90/elliokat/poems/Shakespeare/sonnet6
/home/cis90/jessuwes/poems/Shakespeare/sonnet6
/home/cis90/luisjus/poems/Shakespeare/sonnet6
/home/cis90/meyerjas/poems/Shakespeare/sonnet6
/home/cis90/bergelyl/sonnet6
/home/cis90/bergelyl/poems/Shakespeare/sonnet6
/home/cis90/gardnnic/poems/Shakespeare/sonnet6
/home/cis90/mohanchi/poems/Shakespeare/sonnet6
/home/cis90/whitfbob/poems/Shakespeare/sonnet6
/home/cis90/crivejoh/poems/Shakespeare/sonnet6
[simben@opus ~]$
```

Ahhh … much better!

All the annoying error messages are redirected to the bit bucket

132
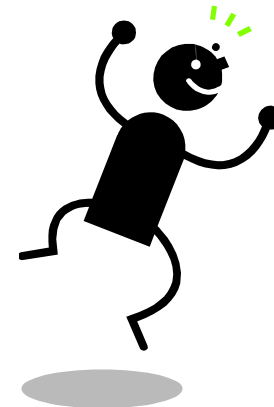
find command

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
```

*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?

133

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

134

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
```

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

135

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

136

find command

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 ~]$
```

137

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

138

grep command

grep command

Syntax

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

grep -R *<options>* "search string" *<startdirectory>*

*The **grep** (Global Regular Expression Print) command searches for content inside of files. The **-R** will search recursively. Some other useful search options are **-i** (case insensitive), **-w** (whole word), **-v** (does not contain)*

140

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?

141

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]$
```

142

grep command

Task: Find simmsben in /etc/passwd

```
/home/cis90/simben $ grep simben90 /etc/passwd
simben90:x:1001:190:Benji Simms:/home/cis90/simben:/bin/bash
```

Task: Now show what line it is on

```
/home/cis90/simben $ grep -n simben90 /etc/passwd
49:simben90:x:1001:190:Benji Simms:/home/cis90/simben:/bin/bash
```

143

grep with the -i option

Look for "so" in sonnet3, sonnet4 and sonnet5

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

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

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

Use the -i option to make searches case insensitive

144

grep with the -w option

Look for "so" in sonnet3, sonnet4 and sonnet5

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

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

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

Use the -w option for whole word only searches

145

grep with the -R option

Search recursively for "kind"

starting in the home directory

discard permission errors

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

Use the -R option to search recursively

146

grep command

Background

Apache is the worlds most popular web server and it's installed on Opus. Try it, you can browse to oslab.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*

147

spell command

spell command

spell - find misspelled words

*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.

spell command

Task: Count the number of misspelled words

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

tee command

tee command

Tee

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.
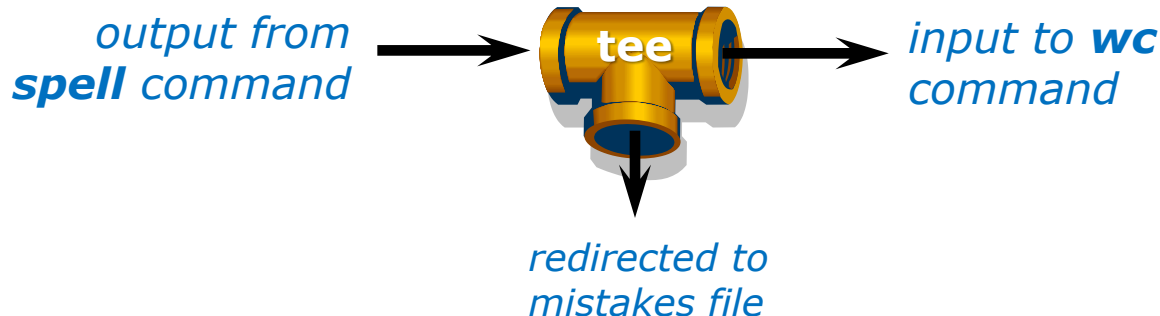
who | tee users | sort

tee command

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

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

This is how you do a spell check , save the misspelled words in a file and count them in a single command

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

*output from **spell** command* → **tee** → *input to **wc** command*

redirected to mistakes file

154

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?

        cat /var/log/wtmp* > logins
        last -f logins | grep $LOGNAME
        last -f logins | grep $LOGNAME | grep "Wed"
        last -f logins | grep $LOGNAME | grep "Wed" | wc -l

On what days do you log in the most? the least?

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
Build up a pipeline, one pipe at a time:

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

What gets printed with the last pipeline?

157

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
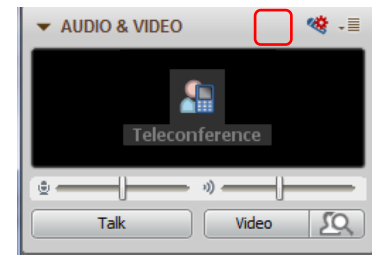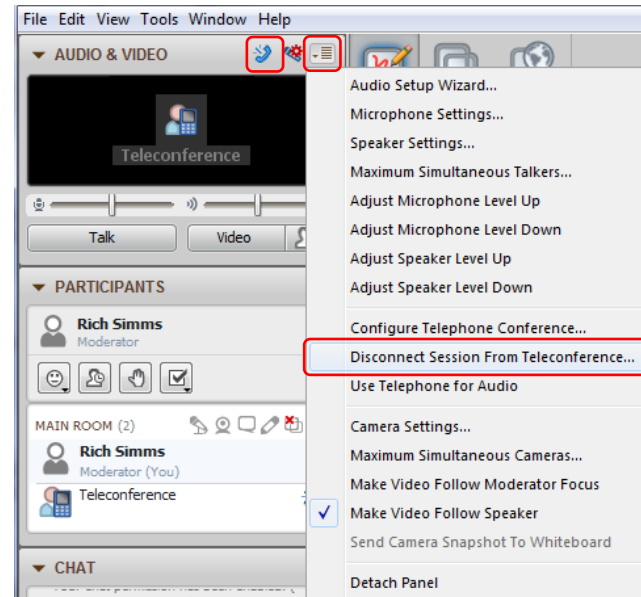web site to see what is due next week.
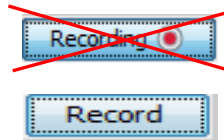
Lab 7

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

[ ] Disconnect session to Teleconference

[ ] Turn recording off

Backup