Lesson Module Status

- Slides – draft
- Properties – done
- Flash cards –
- First minute quiz – done
- Web calendar summary – done
- Web book pages – done
- Commands – na
- Lab tested – na
- Supplies – na
- Class PC's – na

- Hide script tested – done
- Practice test uploaded - done
- CCC Confer wall paper – done

- Materials uploaded – done
- Backup headset charged – oops
- Backup slides, CCC info, handouts on flash drive - done

- Check that room headset is charged – done

Instructor: **Rich Simms**

Dial-in: **888-450-4821**

Passcode: **761867**

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

First Minute Quiz

Please close your books, notes, lesson materials, forum and answer these questions **in the order** shown:

- 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

email answers to: `risimms@cabrillo.edu`

CCC Confer

[ ] Has the phone bridge been added?

[ ] Is recording on?

[ ] Does the phone bridge have the mike?

[ ] Share lesson slides, team slide, putty teminals, and Chrome

[ ] Disable spelling on PowerPoint

Review

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

Questions

Previous material and assignment

1. Lab 7 questions?
2. Extra credit Lab questions?
3. Questions on redirection and pipes?
4. Any other material?

Lab 6 Results (Permissions)

Missed questions on Lab 6

03 XXXX
08 XX
09 XXXXXXXXX
10 XXXXXXXXXX
11 XXXX
12 XXX
14 XXXX
15 XXX
16 XXXXXX
17 X
19 XXX
21 XX

Lab 6 Results

2) Do a long listing of the file, /home/rsimms/uhistory. Who owns it?
Can you move the file to your home directory? Why or why not?
Can you copy the file to your home directory? Why or why not?

3) Now that you have copied the file uhistory to your home directory, who owns it? What are the permissions?

Look at the /home/rismms directory and the uhistory file in it

```
/home/cis90ol/simmsben $ ls -ld /home/rsimms
drwxr-x--- 11 rsimms cis90ol 4096 Apr  1 15:13 /home/rsimms
/home/cis90ol/simmsben $ ls -l /home/rsimms/uhistory
-rw-r----- 1 rsimms cis90ol 25895 Mar 23 11:42 /home/rsimms/uhistory
```

Can’t move it (no write permission to /home/rsimms directory)

```
/home/cis90ol/simmsben $ mv /home/rsimms/uhistory .
mv: cannot move `/home/rsimms/uhistory' to `./uhistory': Permission denied
```

Can copy it though, note the owner changes

```
/home/cis90ol/simmsben $ cp /home/rsimms/uhistory .
/home/cis90ol/simmsben $ ls -l uhistory
-rw-r----- 1 simmsben cis90ol 25895 Apr 11 08:57 uhistory
```

Lab 6 Results

2) Do a long listing of the file, /home/rsimms/uhistory. Who owns it? Can you move the file to your home directory? Why or why not? Can you copy the file to your home directory? Why or why not?

3) Now that you have copied the file uhistory to your home directory, who owns it? What are the permissions?

The copy commands some students tried DIDN'T WORK … why?

```
/home/cis90ol/simmsben $ cp /home/rsimms/uhistory
cp: missing destination file operand after `/home/rsimms/uhistory'
Try `cp --help' for more information.
```

No destination specified

```
/home/cis90ol/simmsben $ cp /home/rsimms/uhistory /home/simmsben
cp: cannot create regular file `/home/simmsben': Permission denied
```

Benji's home directory is in /home/cis90ol not /home

```
/home/cis90ol/simmsben $ echo cp /home/rsimms/uhistory
cp /home/rsimms/uhistory
```

Doesn't copy the file, just echoes command line arguments

```
/home/cis90ol/simmsben $ cp /home/rsimms/uhistory simmsben
```

Does the copy, but filename becomes simmsben instead of uhistory

Lab 6 Results

8) Change back to your home directory and set the misc directory to full permissions:

chmod 777 misc

```
/home/cis90ol/simmsben $ chmod 777 misc
```

or

```
/home/cis90ol/simmsben $ chmod 777 misc/
```

Some students did this step, but then clobbered their permissions on misc during the next step (step 9)

Lab 6 Results

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.

Note the use of multiple arguments

```
/home/cis90/roddyduk $ chmod u+rwx,og-w poems/ poems/*
```

*Note, * metacharacter expanded to match all four subdirectories*

or

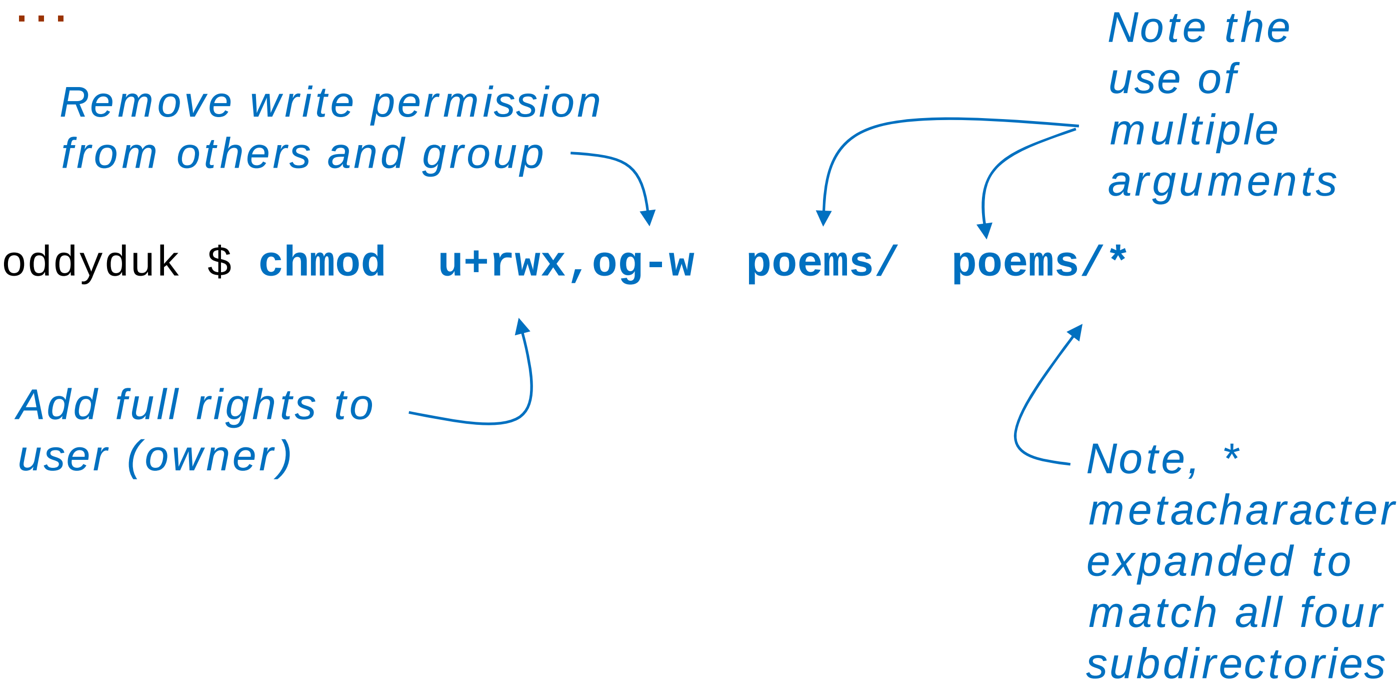
```
/home/cis90/roddyduk $ chmod 755 poems/ poems/*
```

```
/home/cis90/roddyduk $ ls -ld poems/ poems/*
drwxr-xr-x 6 roddyduk cis90 4096 Oct 16 08:21 poems/
drwxr-xr-x 2 roddyduk cis90 4096 Oct 16 08:21 poems/Anon
drwxr-xr-x 2 roddyduk cis90 4096 Jul 20  2001 poems/Blake
drwxr-xr-x 2 roddyduk cis90 4096 Oct 22 16:57 poems/Shakespeare
drwxr-xr-x 2 roddyduk cis90 4096 Oct 21 06:46 poems/Yeats
```

Always check your new permission settings with a long listing!

Lab 6 Results

9) explained ...



Note the use of multiple arguments

Remove write permission from others and group

```
/home/cis90/roddyduk $ chmod  u+rwx,og-w  poems/  poems/*
```

Add full rights to user (owner)

Note, * metacharacter expanded to match all four subdirectories

Use echo to see how the shell will expands the arguments

```
/home/cis90/roddyduk $ echo poems/ poems/*
poems/ poems/Anon poems/Blake poems/Shakespeare poems/Yeats
```

Lab 6 Results

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 yourself. See if you can do this using a minimum number of commands. (hint: use filename expansion characters).

```
/home/cis90/roddyduk/poems $ chmod 444 poems/*/*

/home/cis90/roddyduk $ ls -l poems/*/*
-r--r--r-- 1 roddyduk cis90 237 Aug 26  2003 poems/Anon/ant
-r--r--r-- 1 roddyduk cis90 779 Oct 12  2003 poems/Anon/nursery
-r--r--r-- 1 roddyduk cis90 151 Jul 20  2001 poems/Anon/twister
-r--r--r-- 1 roddyduk cis90 582 Jul 20  2001 poems/Blake/jerusalem
-r--r--r-- 1 roddyduk cis90 115 Jul 20  2001 poems/Blake/tiger
-r--r--r-- 1 roddyduk cis90 614 Jul 20  2001 poems/Shakespeare/sonnet1
-r--r--r-- 1 roddyduk cis90 620 Jul 20  2001 poems/Shakespeare/sonnet10
< snipped >
-r--r--r-- 1 roddyduk cis90 581 Jul 20  2001 poems/Shakespeare/sonnet7
-r--r--r-- 1 roddyduk cis90 620 Jul 20  2001 poems/Shakespeare/sonnet9
-r--r--r-- 1 roddyduk cis90 856 Sep 29 06:15 poems/Yeats/mooncat
-r--r--r-- 1 roddyduk cis90 520 Jul 20  2001 poems/Yeats/old
-r--r--r-- 1 roddyduk cis90 863 Jul 20  2001 poems/Yeats/whitebirds
```

Lab 6 Results

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.

```
/home/cis90/roddyduk $ chmod 750 bin

/home/cis90/roddyduk $ ls -ld bin
drwxr-x--- 2 roddyduk cis90 4096 Mar 26 17:56 bin
```

Lab 6 Results

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

-r-xr-x---

disallowing others outside the group from executing our commands.

```
/home/cis90/roddyduk $ chmod 550 bin/*
/home/cis90/roddyduk $ ls -l bin
total 76
-r-xr-x--- 1 roddyduk cis90  220 Apr 22  2004 app
-r-xr-x--- 1 roddyduk cis90 6160 Aug 28  2003 banner
-r-xr-x--- 1 roddyduk cis90  509 Jun  6  2002 datecal
-r-xr-x--- 1 roddyduk cis90 3388 Sep 11  2005 enlightenment
-r-xr-x--- 1 roddyduk cis90  107 Jul 20  2001 hi
-r-xr-x--- 1 roddyduk cis90  375 Oct 20  2003 I
-r-xr-x--- 1 roddyduk cis90  190 Jul 20  2001 treed
-r-xr-x--- 1 roddyduk cis90  174 Mar  4  2004 tryme
-r-xr-x--- 1 roddyduk cis90   74 Jul 20  2001 zoom
```

Lab 6 Results

14) For the *class* directory set the permissions to 710.
For the *labs* subdirectory, set permissions to 530.
For the *exams* subdirectory, take away all permissions from group and others, leaving full permission for owner.

```
/home/cis90/roddyduk $ chmod 710 class
/home/cis90/roddyduk $ chmod 530 class/labs
/home/cis90/roddyduk $ chmod 700 class/exams

/home/cis90/roddyduk $ ls -ld class/ class/*
drwx--x--- 4 roddyduk cis90 4096 Oct 16 08:18 class/
drwx------ 2 roddyduk users 4096 Oct 16 08:18 class/exams
dr-x-wx--- 2 roddyduk users 4096 Oct 16 08:25 class/labs
```

Lab 6 Results

15) Make all ordinary files under *class/labs* and *class/exams* be:
read-write for owner
read-only for group and
no permission for others.

```
/home/cis90/roddyduk $ chmod 640 class/*/*

/home/cis90/roddyduk $ ls -ld class/*/*
-rw-r----- 1 roddyduk staff     0 Oct 25 08:32 class/exams/test01.graded
-rw-r----- 1 roddyduk staff   143 Sep  9 14:38 class/labs/lab01.graded
-rw-r----- 1 roddyduk staff  1042 Sep 16 19:10 class/labs/lab02.graded
-rw-r----- 1 roddyduk staff 13834 Sep 23 18:07 class/labs/lab03.graded
```

Lab 6 Results

16) For the *edits* directory, give yourself full permission, but no permission for group or others.
For the ordinary files under *edits*, take away read permission from group, leaving everything else as it is.

```
/home/cis90/roddyduk $ chmod 700 edits/
/home/cis90/roddyduk $ chmod g-r edits/*


/home/cis90/roddyduk $ ls -ld edits edits/*
drwx------ 2 roddyduk cis90 4096 Oct 16 08:24 edits/
-rw----r-- 1 roddyduk cis90 1382 Feb  1  2002 edits/better_town
-rw----r-- 1 roddyduk cis90 1580 Nov 16  2004 edits/small_town
-rw----r-- 1 roddyduk cis90  485 Aug 26  2003 edits/spellk
-rw----r-- 1 roddyduk cis90  250 Jul 20  2001 edits/text.err
-rw----r-- 1 roddyduk cis90  231 Jul 20  2001 edits/text.fxd
/home/cis90/roddyduk $
```

Lab 6 Results

17) Add read permission for everyone to all the files in the misc directory.

```
/home/cis90ol/simmsben $ chmod +r misc/*
/home/cis90ol/simmsben $ ls -l misc/
total 60
-rw-r--r-- 1 simmsben cis90ol   148 Jul 20  2001 file.dos
-rw-r--r-- 1 simmsben cis90ol    78 Oct 26  2004 fruit
-rw-r--r-- 2 simmsben cis90ol 10576 Jul 20  2001 manpage
lrwxrwxrwx 1 simmsben cis90ol    20 Feb 17 10:12 mystery ->
../bin/enlightenment
-rw-r--r-- 1 simmsben cis90ol    78 Apr 17  2004 salad
-rw-r--r-- 1 simmsben cis90ol   352 Jul 20  2001 what_am_i
/home/cis90ol/simmsben $
```

Lab 6 Results

19) Create an empty file called old and an empty directory called olddir:

touch old; mkdir olddir

```
/home/cis90ol/simmsben $ touch old; mkdir olddir

/home/cis90ol/simmsben $ ls -ld old olddir
-rw-rw-r-- 1 simmsben cis90ol    0 Apr 11 09:45 old
drwxrwxr-x 2 simmsben cis90ol 4096 Apr 11 09:45 olddir
/home/cis90ol/simmsben $
```

Old should be 664 and olddir should be 775 (because umask is 002)

Lab 6 Results

21) Create an empty file called new and an empty directory called newdir:

touch new; mkdir newdir

```
/home/cis90ol/simmsben $ touch new; mkdir newdir

/home/cis90ol/simmsben $ ls -ld new newdir
-rw-rw-rw- 1 simmsben cis90ol    0 Apr 11 09:51 new
drwxrwxrwx 2 simmsben cis90ol 4096 Apr 11 09:51 newdir
/home/cis90ol/simmsben $
```

Old should be 666 and olddir should be 777 (because umask is 000)

More on I/O

(input/output)

Input and Output

File Redirection

There are 3 standard UNIX file descriptors:

Name	Integer Value
stdin (Standard In)	0
stdout (Standard Out)	1
stderr (Standard Error)	2

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 using these file descriptors:

~~0~~< *filename*

Redirects **stdin**, input will now come from *filename* rather than the keyboard.

~~1~~> *filename*

Redirects **stdout**, output will now go to *filename* instead of the terminal.

2> *filename*

Redirects **stderr**, error messages will now go to *filename* instead of the terminal.

>> *filename*

Redirects **stdout**, output will now be appended to *filename*.

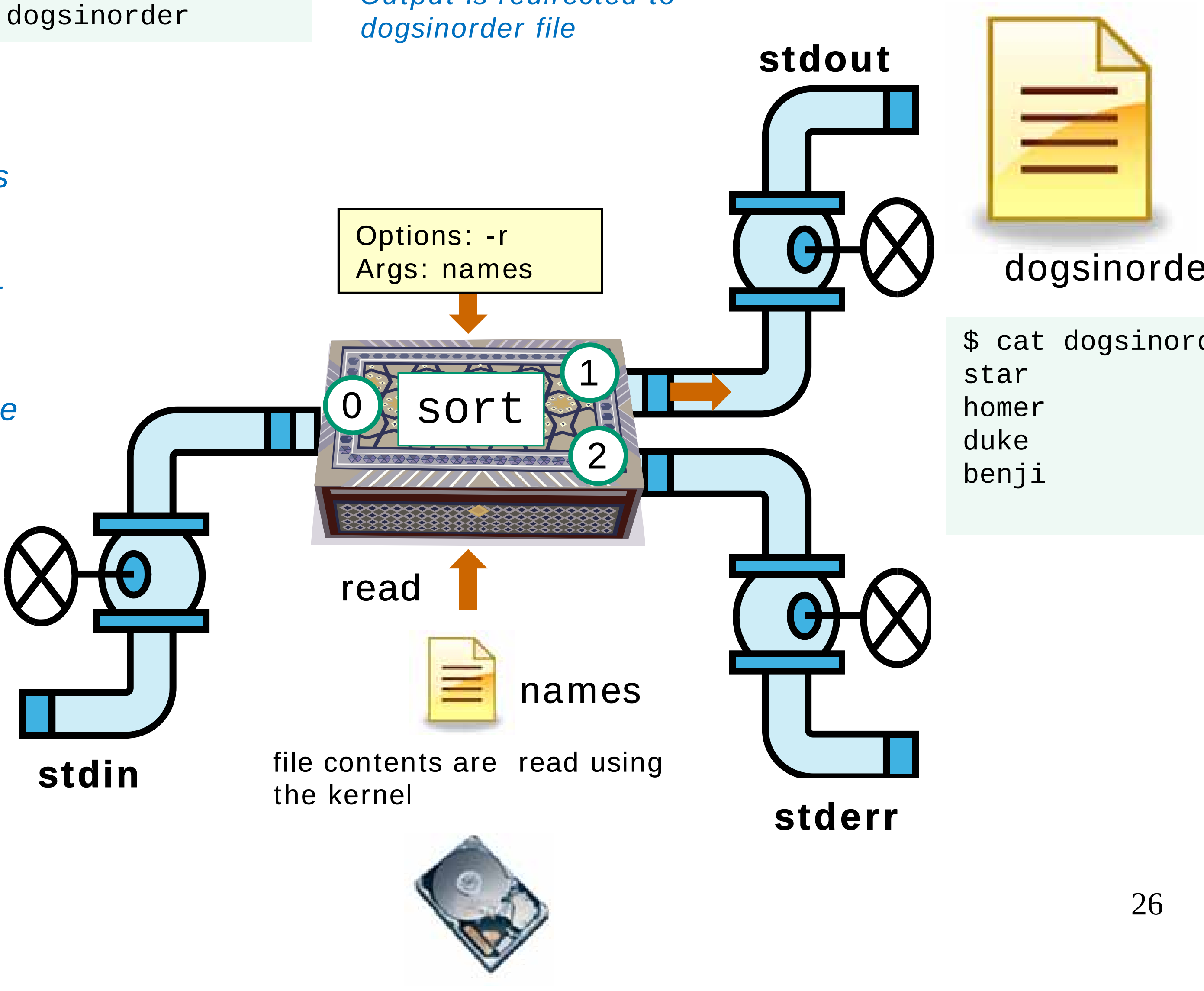
The 0 in 0< is not necessary, just use < to redirect stdin
The 1 in 1> is not necessary, just use > to redirect stdout
The 2 in 2> is necessary, always use 2> to redirect stderr

Example program to process: sort command

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

Output is redirected to dogsinorder file

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 -r option and modifies how it sorts.



file contents are read using the kernel

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

Example C program code

```
[rsimms@opus misc]$ cat simple.c
char question[] = "What is your name stranger? ";
char greeting[] = "Well I'm very pleased to meet you, ";
char buffer[80];
main()
{
        int len;

        write(2, question, sizeof(question));
        len = read(0, buffer, 80);
        write(1, greeting, sizeof(greeting));
        write(1, buffer, len);
}
```

Write to stderr

Read from stdin

Write to stdout

Write to stdout again

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

Compiling simple.c into a binary executable named simple

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

Example C program code

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

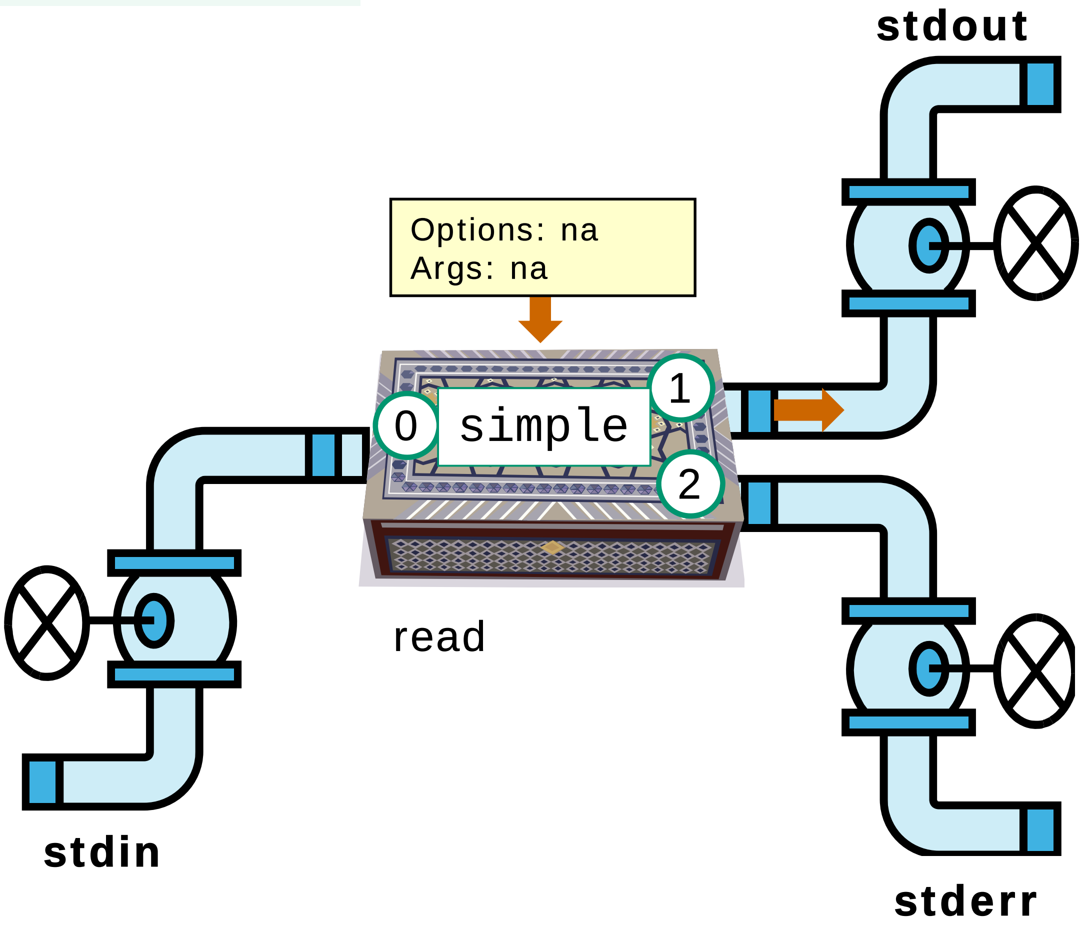
```
[rsimms@opus misc]$ ./simple > myfile
What is your name stranger? Rich
[rsimms@opus misc]$ cat myfile
Well I'm very pleased to meet you, Rich
```

In the second example, output has been redirected to a file named myfile. The simple program has no special knowledge (coding instructions) for a file named myfile. It just writes to stdout and that output will go to wherever stdout had been directed to.

Example program to process: simple program

```
$ ./simple
```

simple writes a prompt to stderr, reads input from stdin, then writes to stdout



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

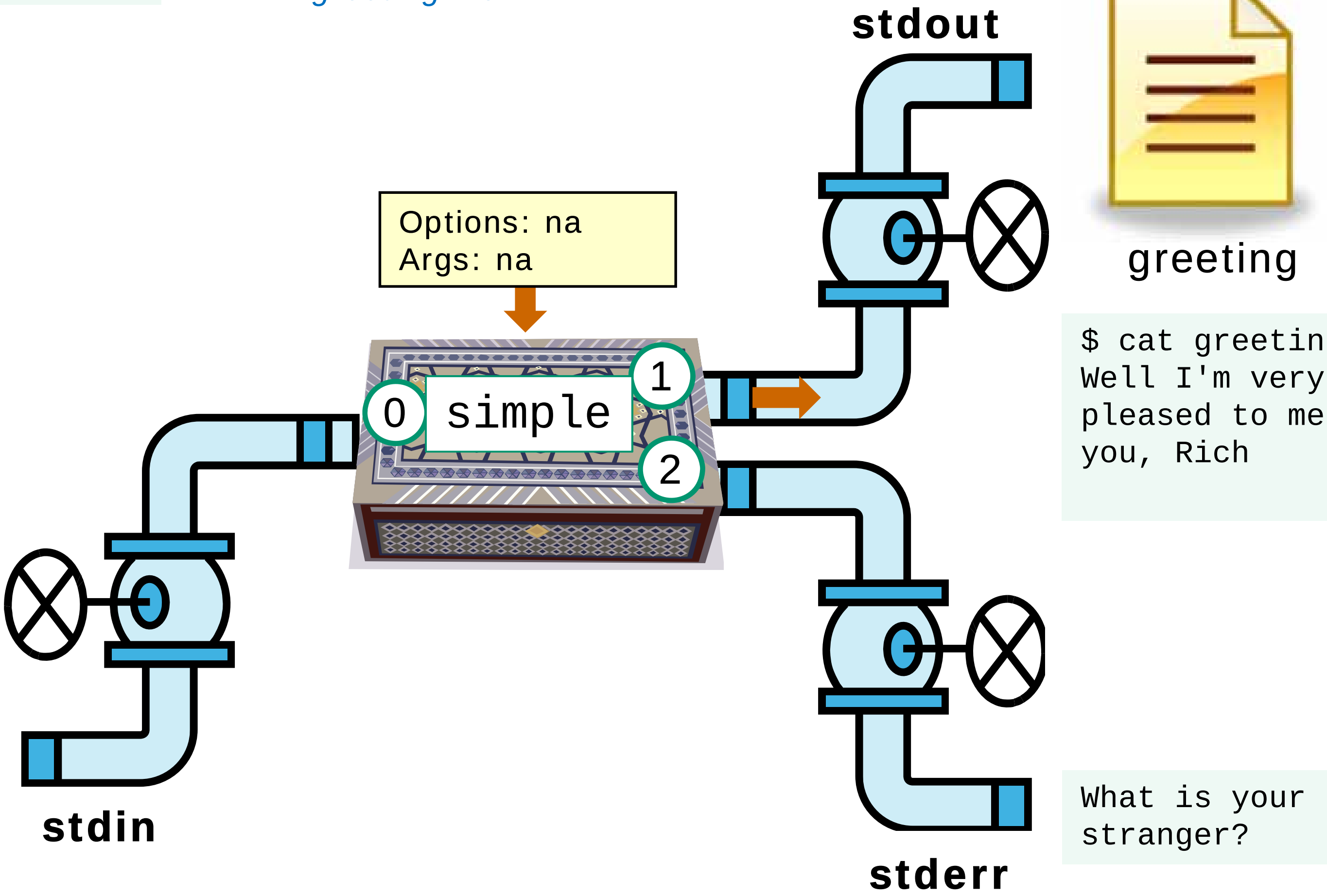
```
Rich
```

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

Example program to process: simple program

```
$ ./simple > greeting
```

Output is redirected to greeting file



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

```
Rich
```

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

More on umask

(input/output)

umask = "user file-creation mask"

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

```
 666
-002
 ---
 664
```

```
/home/cis90/roddyduk/lesson9 $ touch newfile        New file
/home/cis90/roddyduk/lesson9 $ ls -l newfile
-rw-rw-r-- 1 roddyduk cis90 0 Oct 27 07:22 newfile
```

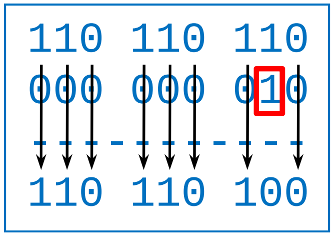
```
 777
-002
 ---
 775
```

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

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

umask = "user file-creation mask"

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

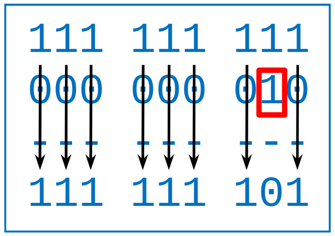


```
110 110 110
000 000 010
110 110 100
```

```
/home/cis90/roddyduk/lesson9 $ touch newfile
/home/cis90/roddyduk/lesson9 $ ls -l newfile
-rw-rw-r-- 1 roddyduk cis90 0 Oct 27 07:22 newfile
```

New file

Start with 666 for new files and apply the mask



```
111 111 111
000 000 010
111 111 101
```

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

New directory

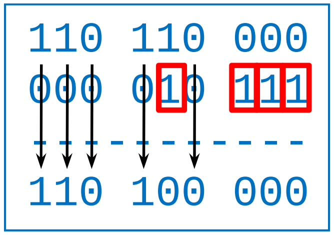
Start with 777 for new directories and apply the mask

*It's not really subtraction, but masking that is being done to create the new file's permissions. Any permission bit in the **mask** will block the **default permission** bit from being set in the new file's permissions.*

umask = "user file-creation mask"

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

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

Copied file



```
110 110 000
000 010 111
-----------
110 100 000
```

Start with original file's permissions and apply the mask

*For new copied files, instead of using the **default permissions** (666 for file and 777 for directory), use the **original file permissions** as the starting point for the mask to be applied to.*

Housekeeping

Housekeeping

1. Lab 7 due today

2. A rude and crude check7 script is available

3. Test #2 next week

Warmup

Egg Hunt

A number of colored eggs have been distributed within your home directory and sub-directories!

1. Can you find them? There should be an obvious one in your home directory. Who is the owner and group for this egg file? The rest are scattered in the various subdirectories you own.

2. Make a new directory named basket in your home directory and see how many egg files you can move into it.

3. Put a Green Check in CCC Confer next to your name when you have collected 3 eggs, raise your hand if you collect all 17.

Test 2 Prep

Teams for today

Debian	Redhat	SUSE	Ubuntu
Emanuel Chris Yu-Chen Dan M Gabriel Jason	Tanner Bobby Terry Geoffrey Jesse	Merrick Craig Tommy Marisol Tajvia	Quinton Jeff Eric Josh Daniel W

4 chocolates will go to 1st place finishers
3 chocolates will go to 2nd place finishers
2 chocolates will go to 3rd place finishers
1 chocolates will go to 4th place finishers

Available in CIS Lab (Mondays 1:30-4:00) or TBA

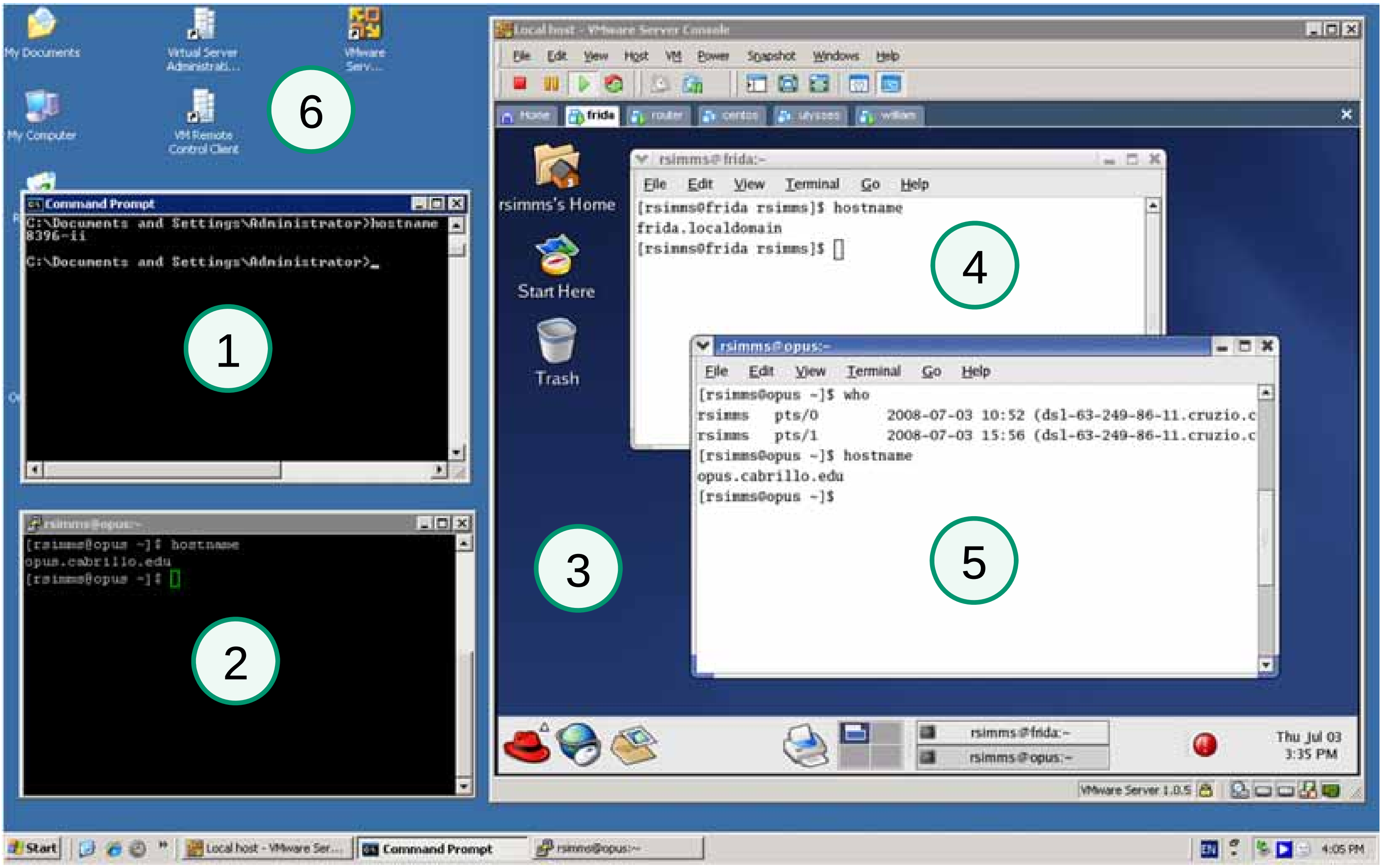
Test #2 Prep

The next test will focus on Lessons 6 - 8 (and related labs)

*… however these lessons build on **all** the material from earlier lessons!*

Base Knowledge

This screen shot shows interaction with three different computers:
8396-II (Win 2003), Frida (RH9) and Opus. **Match the numbers to the computers**



```
C:\Documents and Settings\Administrator>hostname
8396-ii

C:\Documents and Settings\Administrator>_
```

```
[rsimms@opus ~]$ hostname
opus.cabrillo.edu
[rsimms@opus ~]$ []
```

```
[rsimms@frida rsimms]$ hostname
frida.localdomain
[rsimms@frida rsimms]$ []
```

```
[rsimms@opus ~]$ who
rsimms   pts/0        2008-07-03 10:52 (dsl-63-249-86-11.cruzio.c
rsimms   pts/1        2008-07-03 15:56 (dsl-63-249-86-11.cruzio.c
[rsimms@opus ~]$ hostname
opus.cabrillo.edu
[rsimms@opus ~]$
```

8396-II
(Win 2003) ○ ○

Frida
(RH9) ○ ○

Opus
(RHEL5) ○ ○

What terminal device am I using for this session?

```
/home/cis90/simmsben $ tty
/dev/pts/0
```

What is the name of the computer I'm using?

```
/home/cis90/simmsben $ hostname
opus.cabrillo.edu
```

Who else is logged in on this system?

Which one of them is me?

```
/home/cis90/simmsben $ who am i
rsimms   pts/0        2009-04-08 04:43 (dsl-67-105-103-45.cruzio.com)
```

What are my user and group ID's?

```
/home/cis90/simmsben $ id
uid=1001(simmsben) gid=103(cis90) groups=100(users),103(cis90)
context=user_u:system_r:unconfined_t
```

What is the name of the OS on this system?

```
/home/cis90/simmsben $ uname
Linux
```

*Is the command **mail** on my path?*
Where on my path is it located?

```
/home/cis90/simmsben $ type mail
mail is hashed (/bin/mail)
```

What kind of file is /bin/mail?

```
/home/cis90/simmsben $ file /bin/mail
/bin/mail: ELF 32-bit LSB executable, Intel 80386,
version 1 (SYSV), for GNU/Linux 2.6.9, dynamically
linked (uses shared libs), for GNU/Linux 2.6.9, stripped
/home/cis90/simmsben $
```

Can I print the file /bin/mail using commands like cat, head, tail, more or less?

```
/home/cis90/simmsben $ file /bin/mail
/bin/mail: ELF 32-bit LSB executable, Intel 80386,
version 1 (SYSV), for GNU/Linux 2.6.9, dynamically
linked (uses shared libs), for GNU/Linux 2.6.9, stripped
/home/cis90/simmsben $
```

NO, you should only print ASCII text files. Binary files contain unprintable characters.

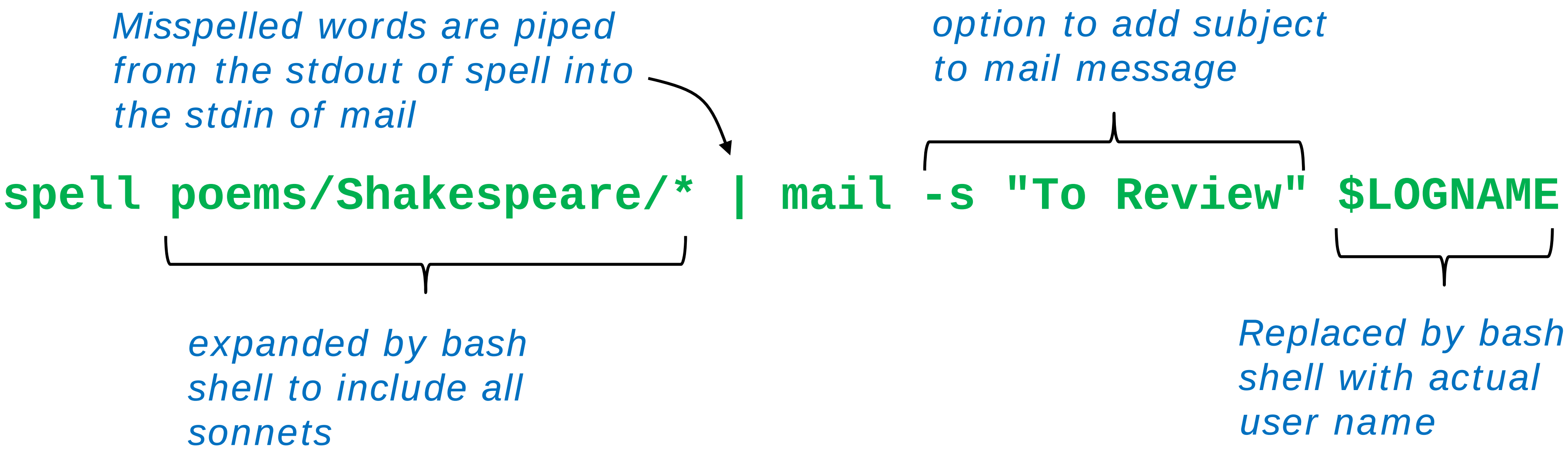
Mail

Q20

From a previous Test #2

Test 2 Q20

20. What single command could be used to mail yourself the misspelled words in all of Shakespeare's sonnets with a subject of “To Review”?



Misspelled words are piped from the stdout of spell into the stdin of mail

option to add subject to mail message

```
spell poems/Shakespeare/* | mail -s "To Review" $LOGNAME
```

expanded by bash shell to include all sonnets

Replaced by bash shell with actual user name

```
$ echo poems/Shakespeare/*
poems/Shakespeare/sonnet1 poems/Shakespeare/sonnet10
poems/Shakespeare/sonnet11 poems/Shakespeare/sonnet15
poems/Shakespeare/sonnet17 poems/Shakespeare/sonnet2
poems/Shakespeare/sonnet26 poems/Shakespeare/sonnet3
poems/Shakespeare/sonnet35 poems/Shakespeare/sonnet4
poems/Shakespeare/sonnet5 poems/Shakespeare/sonnet6
poems/Shakespeare/sonnet7 poems/Shakespeare/sonnet9
poems/Shakespeare/trick2 poems/Shakespeare/words
```

Test 2 Q20 verification

20. What single command could be used to mail yourself the misspelled words in all of Shakespeare’s sonnets with a subject of “To Review”?

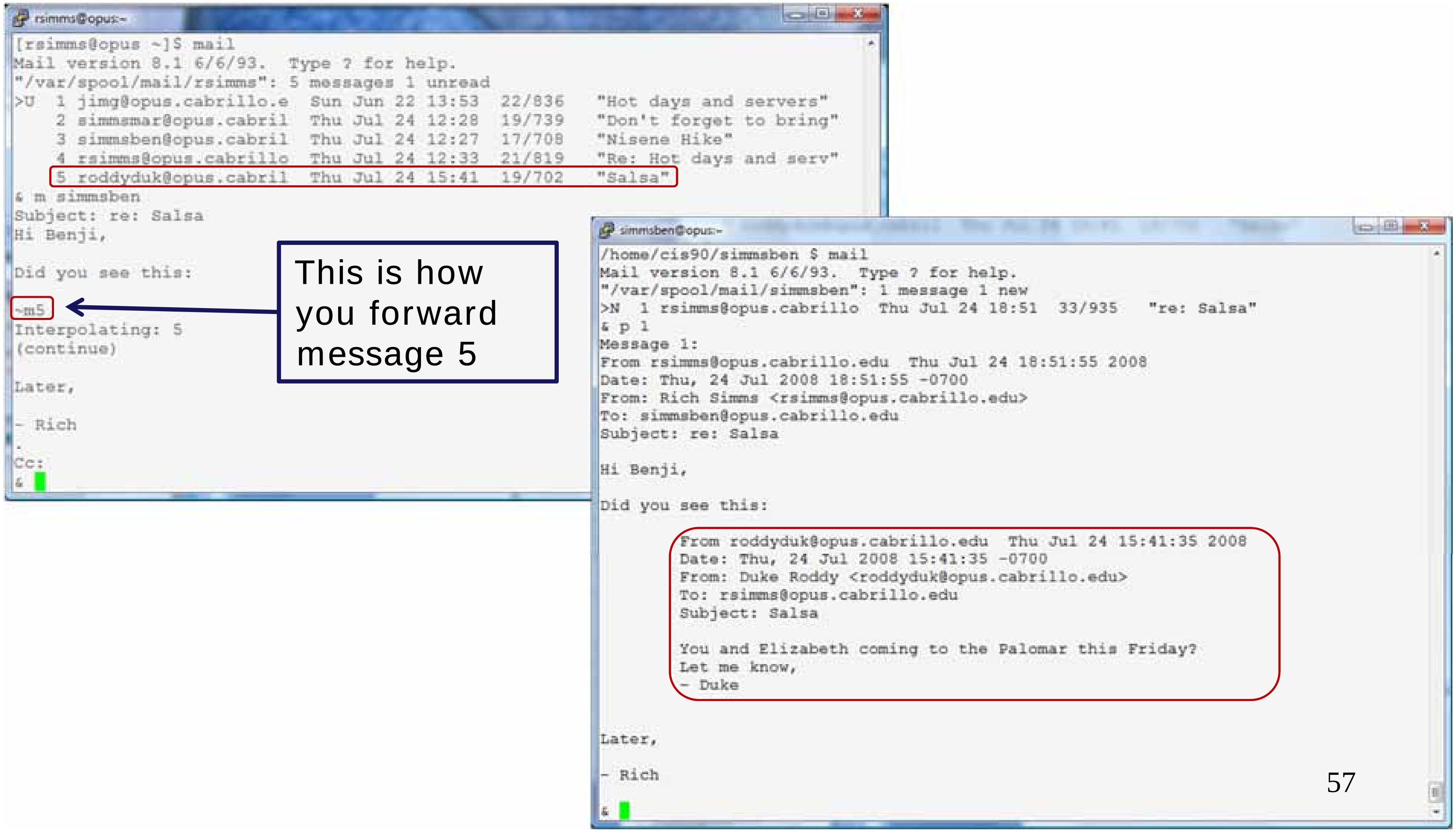
```
/home/cis90/roddyduk $ spell poems/Shakespeare/* | mail -s "To Review" $LOGNAME
You have mail in /var/spool/mail/roddyduk
/home/cis90/roddyduk $ mail
Mail version 8.1 6/6/93.  Type ? for help.
"/var/spool/mail/roddyduk": 1 message 1 unread
>U  1 roddyduk@opus.cabril  Thu Nov  6 11:41  89/1198  "To Review"
& 1
Message 1:
From roddyduk@opus.cabrillo.edu  Thu Nov  6 11:41:24 2008
Date: Thu, 6 Nov 2008 11:41:24 -0800
From: Duke Roddy <roddyduk@opus.cabrillo.edu>
To: roddyduk@opus.cabrillo.edu
Subject: To Review

addeth
ambassage
ask'd
barrenly
bc
bear'st
bedb
belov'd
bestow'st
buriest
check'd
chorl
climb'd
consum'st
convertest
couldst
debateth
departest
deserv'd
diest
distill'd
ec
ee
e&
eeee
Feed'st
feel'st
fill'd
gainstfont
gaz'd
grow'st
happies
highmost
ingraft
issueless
Jz
kill'd
lodg'd
lov'st
makeless
mak'st
metre
murd'rous
nought
o'ersnow
overplus
PJ
possess'd
presententh
reeleth
refigured
rememb'red
renewest
ruinate
scorn'd
shouldst
stick'st
term'd
tomb'd
totter'd
touch'd
unbless
unear'd
Unlook'd
unprovident
unthrift
Unthrifty
unus'd
viewest
will'd
youngly
zb
zd

& x
/home/cis90/roddyduk $
```

To check your answer using Opus, issue the command and then read your mail

font reduced so misspelled words fit on slide

mail command

Forwarding a message with ~m

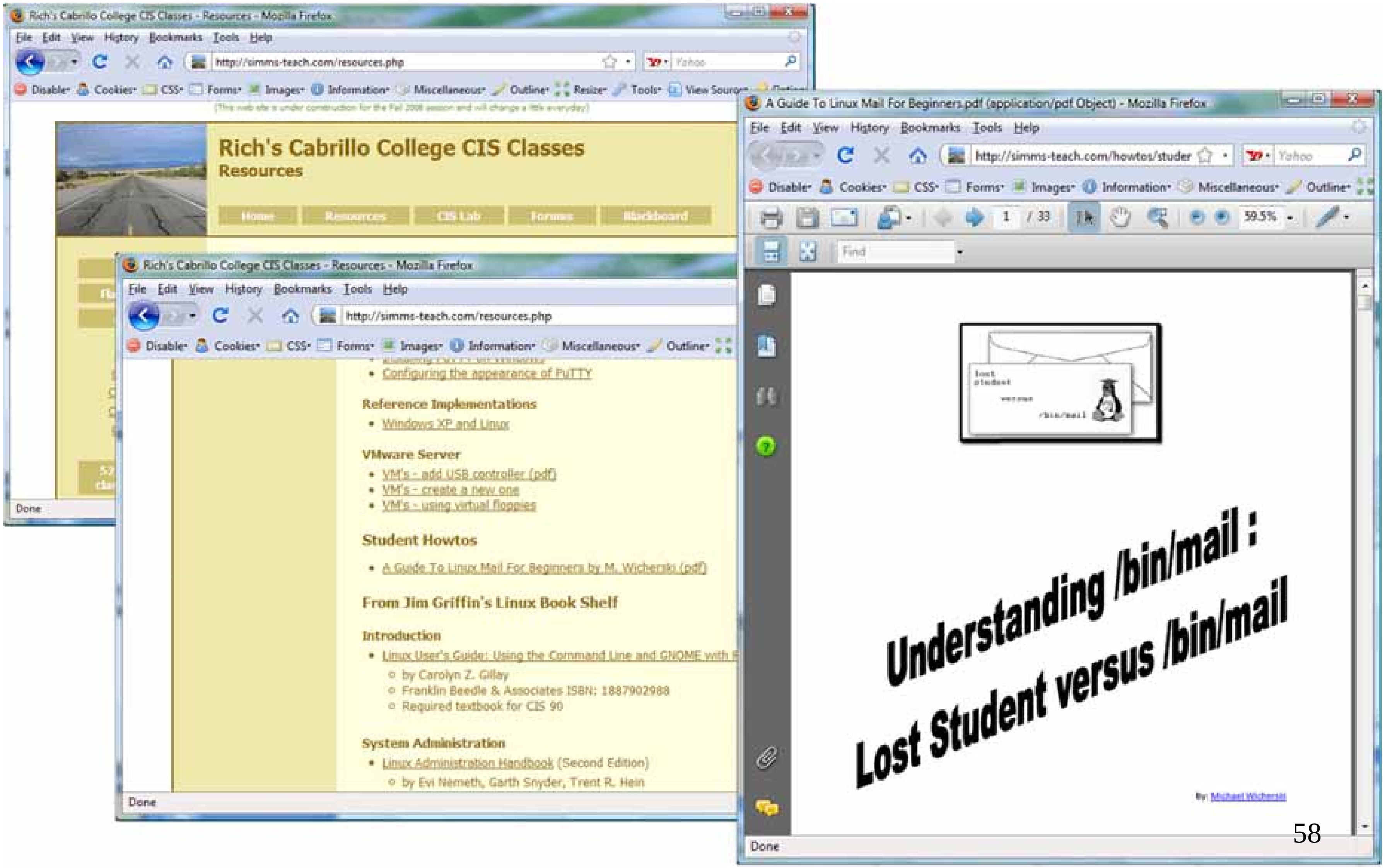


```
[rsimms@opus ~]$ mail
Mail version 8.1 6/6/93.  Type ? for help.
"/var/spool/mail/rsimms": 5 messages 1 unread
>U  1 jimg@opus.cabrillo.e  Sun Jun 22 13:53  22/836   "Hot days and servers"
    2 simmsmar@opus.cabril  Thu Jul 24 12:28  19/739   "Don't forget to bring"
    3 simmsben@opus.cabril  Thu Jul 24 12:27  17/708   "Nisene Hike"
    4 rsimms@opus.cabrillo  Thu Jul 24 12:33  21/819   "Re: Hot days and serv"
    5 roddyduk@opus.cabril  Thu Jul 24 15:41  19/702   "Salsa"
& m simmsben
Subject: re: Salsa
Hi Benji,

Did you see this:

~m5
Interpolating: 5
(continue)

Later,

- Rich
.
Cc:
& 
```

This is how you forward message 5

```
/home/cis90/simmsben $ mail
Mail version 8.1 6/6/93.  Type ? for help.
"/var/spool/mail/simmsben": 1 message 1 new
>N  1 rsimms@opus.cabrillo  Thu Jul 24 18:51  33/935   "re: Salsa"
& p 1
Message 1:
From rsimms@opus.cabrillo.edu  Thu Jul 24 18:51:55 2008
Date: Thu, 24 Jul 2008 18:51:55 -0700
From: Rich Simms <rsimms@opus.cabrillo.edu>
To: simmsben@opus.cabrillo.edu
Subject: re: Salsa

Hi Benji,

Did you see this:

        From roddyduk@opus.cabrillo.edu  Thu Jul 24 15:41:35 2008
        Date: Thu, 24 Jul 2008 15:41:35 -0700
        From: Duke Roddy <roddyduk@opus.cabrillo.edu>
        To: rsimms@opus.cabrillo.edu
        Subject: Salsa

        You and Elizabeth coming to the Palomar this Friday?
        Let me know,
        - Duke


Later,

- Rich

& 
```

More on mail – see the first student Howto



Rich's Cabrillo College CIS Classes - Resources - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://simms-teach.com/resources.php
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source
Home
Resources
CIS Lab
Forums
Blackboard
Done
A Guide To Linux Mail For Beginners.pdf (application/pdf Object) - Mozilla Firefox
http://simms-teach.com/howtos/studer
1 / 33
59.5%
Find
lost student versus /bin/mail

(This web site is under construction for the Fall 2008 session and will change a little everyday)

Rich's Cabrillo College CIS Classes

Resources

- Configuring the appearance of PuTTY

Reference Implementations

- Windows XP and Linux

VMware Server

- VM's - add USB controller (pdf)
- VM's - create a new one
- VM's - using virtual floppies

Student Howtos

- A Guide To Linux Mail For Beginners by M. Wicherski (pdf)

From Jim Griffin's Linux Book Shelf

Introduction

- Linux User's Guide: Using the Command Line and GNOME with F
  - by Carolyn Z. Gillay
  - Franklin Beedle & Associates ISBN: 1887902988
  - Required textbook for CIS 90

System Administration

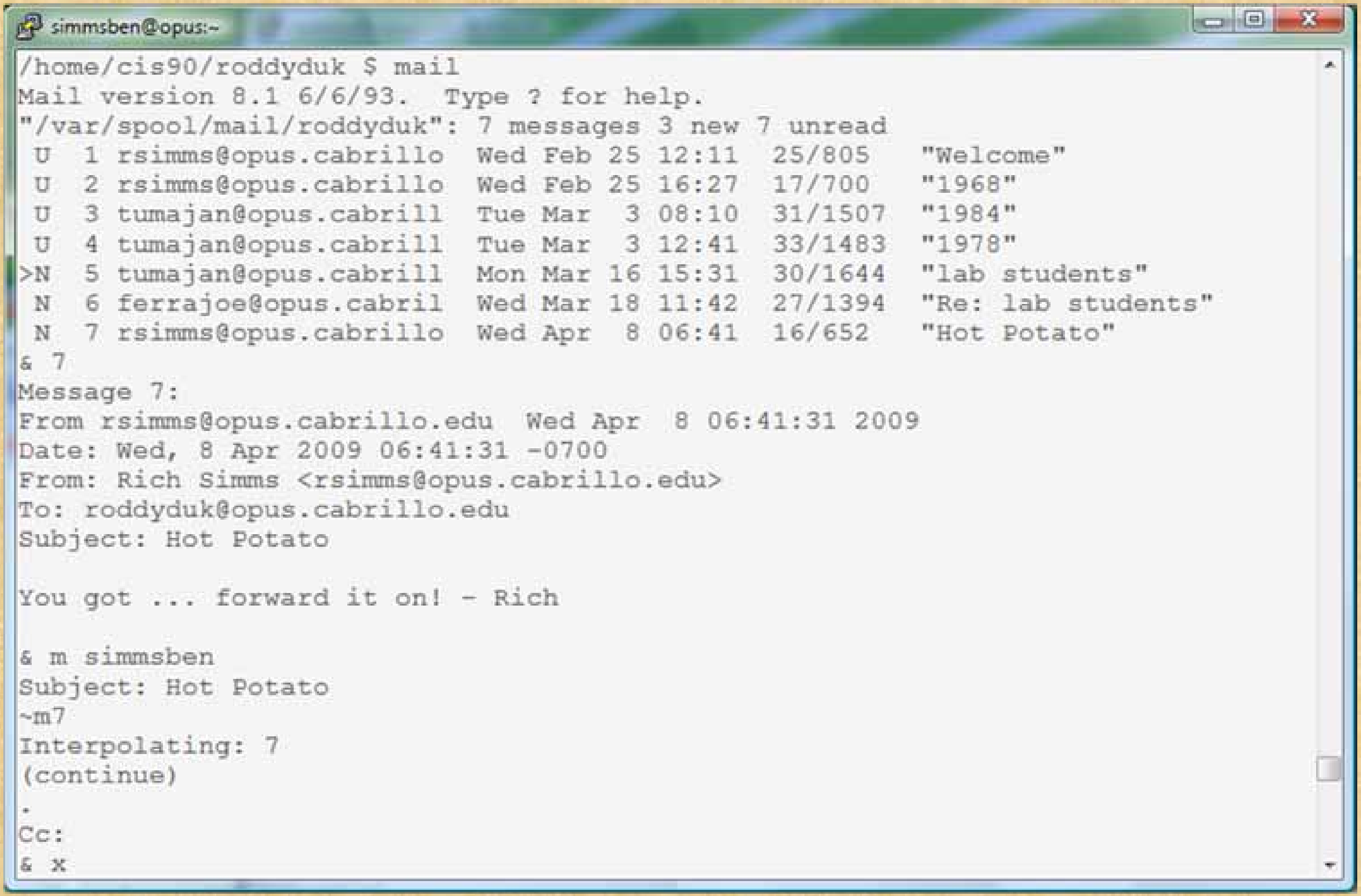
- Linux Administration Handbook (Second Edition)
  - by Evi Nemeth, Garth Snyder, Trent R. Hein

Understanding /bin/mail :
Lost Student versus /bin/mail

By: Michael Wicherski

mail command

Around the room exercise



```
/home/cis90/roddyduk $ mail
Mail version 8.1 6/6/93.  Type ? for help.
"/var/spool/mail/roddyduk": 7 messages 3 new 7 unread
 U  1 rsimms@opus.cabrillo  Wed Feb 25 12:11  25/805   "Welcome"
 U  2 rsimms@opus.cabrillo  Wed Feb 25 16:27  17/700   "1968"
 U  3 tumajan@opus.cabrill  Tue Mar  3 08:10  31/1507  "1984"
 U  4 tumajan@opus.cabrill  Tue Mar  3 12:41  33/1483  "1978"
>N  5 tumajan@opus.cabrill  Mon Mar 16 15:31  30/1644  "lab students"
 N  6 ferrajoe@opus.cabril  Wed Mar 18 11:42  27/1394  "Re: lab students"
 N  7 rsimms@opus.cabrillo  Wed Apr  8 06:41  16/652   "Hot Potato"
& 7
Message 7:
From rsimms@opus.cabrillo.edu  Wed Apr  8 06:41:31 2009
Date: Wed, 8 Apr 2009 06:41:31 -0700
From: Rich Simms <rsimms@opus.cabrillo.edu>
To: roddyduk@opus.cabrillo.edu
Subject: Hot Potato

You got ... forward it on! - Rich

& m simmsben
Subject: Hot Potato
~m7
Interpolating: 7
(continue)
.
Cc:
& x
```

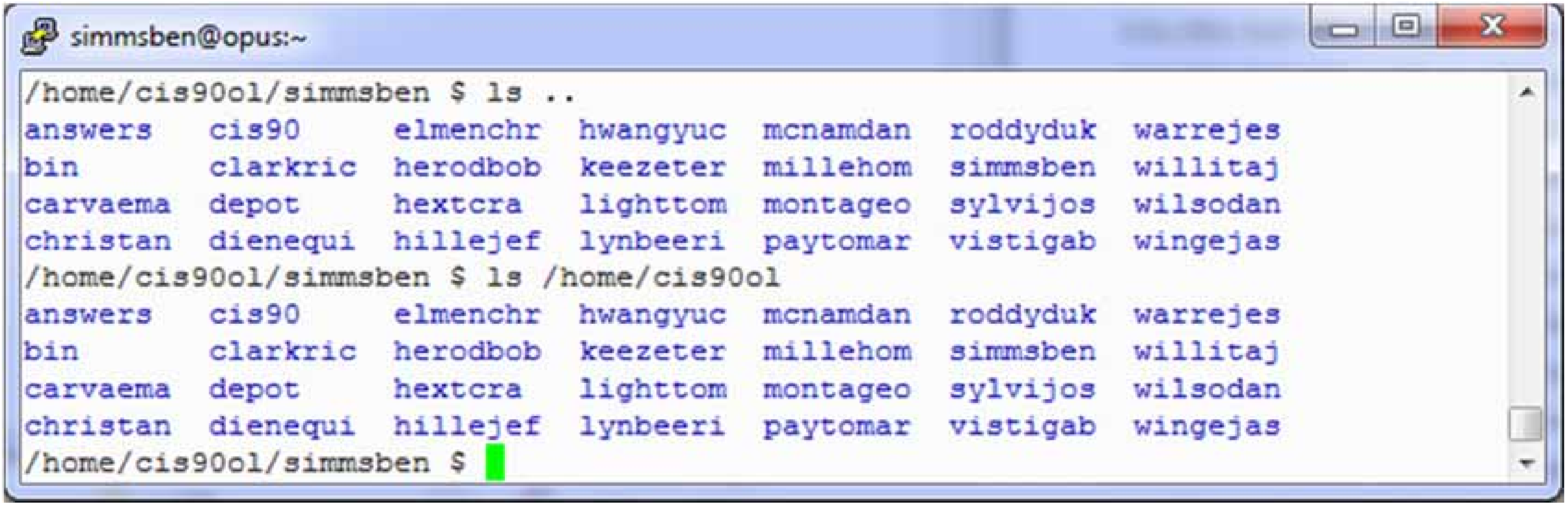
who | sort | cut -f 1 -d "
"

```
carvaema
christan
clarkric
dienequi
elmenchr
herodbob
hextcra
hillejef
hwangyuc
keezeter
lighttom
lynbeeri
mcnamdan
montageo
paytomar
sylvijos
vistigab
warrejes
willitaj
wilsodan
wingejas
```

You have the hot potato - forward it on

tty, who, grep, head, /dev/pts/*, permissions

How can I see all the other home directories for our class



```
/home/cis90ol/simmsben $ ls ..
answers   cis90     elmenchr  hwangyuc  mcnamdan  roddyduk  warrejes
bin       clarkric  herodbob  keezeter  millehom  simmsben  willitaj
carvaema  depot     hextcra   lighttom  montageo  sylvijos  wilsodan
christan  dienequi  hillejef  lynbeeri  paytomar  vistigab  wingejas
/home/cis90ol/simmsben $ ls /home/cis90ol
answers   cis90     elmenchr  hwangyuc  mcnamdan  roddyduk  warrejes
bin       clarkric  herodbob  keezeter  millehom  simmsben  willitaj
carvaema  depot     hextcra   lighttom  montageo  sylvijos  wilsodan
christan  dienequi  hillejef  lynbeeri  paytomar  vistigab  wingejas
/home/cis90ol/simmsben $
```

What kind of pathnames are used above?

Which directories above are not home directories associated with an Opus user account?

What is my terminal?

```
/home/cis90/roddyduk $ tty
/dev/pts/3
```

What are the permissions on my terminal?

```
/home/cis90/roddyduk $ ls -l /dev/pts/3
crw--w---- 1 roddyduk tty 136, 3 Apr  8 08:02 /dev/pts/3
```

How to I change the permissions so others can write to my terminal?

```
/home/cis90/roddyduk $ chmod o+w /dev/pts/3
/home/cis90/roddyduk $ ls -l /dev/pts/3
crw--w--w- 1 roddyduk tty 136, 3 Apr  8 08:06 /dev/pts/3
```

How do I find another user's terminal?

```
/home/cis90/roddyduk $ who | grep simmsben
simmsben pts/2        2009-04-08 07:58 (dsl-63-249-103-107.cruzio.com)
```

How do I write the first four lines of the file letter to another user's terminal?

```
/home/cis90/roddyduk $ head -4 letter > /dev/pts/2
```

Around the room exercise

Duke copies first 4 lines of his file letter to Benji's terminal:

```
[roddyduk@opus ~]$ who | grep simmsben
simmsben pts/1        2008-10-29 14:35
[roddyduk@opus ~]$ head -4 letter > /dev/pts/1
-bash: /dev/pts/1: Permission denied
[roddyduk@opus ~]$ head letter > /dev/pts/1
[roddyduk@opus ~]$
```

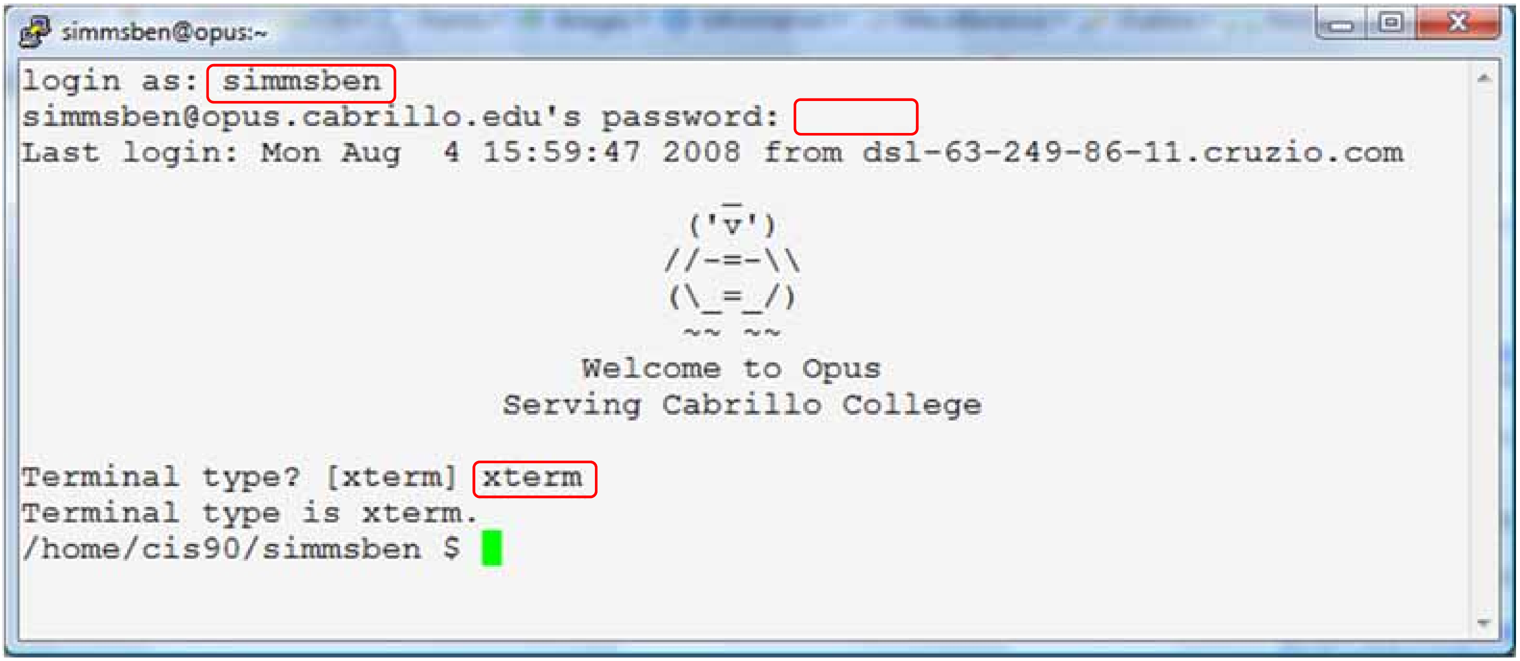
Benji enables his terminal to be written to by others:

```
/home/cis90/simmsben $ tty
/dev/pts/1
/home/cis90/simmsben $ ls -l /dev/pts/1
crw--w---- 1 simmsben tty 136, 1 Oct 29 14:36
/dev/pts/1
/home/cis90/simmsben $ chmod o+w /dev/pts/1
/home/cis90/simmsben $ Hello Mother!  Hello Father!
Here I am at Camp Granada.  Things are very
entertaining,
and they say we'll have some fun when it stops raining.
```

```
who | sort | cut -f 1 -d " "

carvaema
christan
clarkric
dienequi
elmenchr
herodbob
hextcra
hillejef
hwangyuc
keezeter
lighttom
lynbeeri
mcnamdan
montageo
paytomar
sylvijos
vistigab
warrejes
willitaj
wilsodan
wingejas
```

Logging in

Logging in



```
login as: simmsben
simmsben@opus.cabrillo.edu's password:
Last login: Mon Aug  4 15:59:47 2008 from dsl-63-249-86-11.cruzio.com

                                   _
                                 ('v')
                                //-=-\\
                                (\_=_/)
                                 ~~ ~~
                            Welcome to Opus
                        Serving Cabrillo College

Terminal type? [xterm] xterm
Terminal type is xterm.
/home/cis90/simmsben $
```

always requires:

username + password + terminal type

Users and Groups

User and Group Management

Where user and group information resides:

- /etc/passwd
- /etc/shadow

- /etc/group
- /etc/gshadow

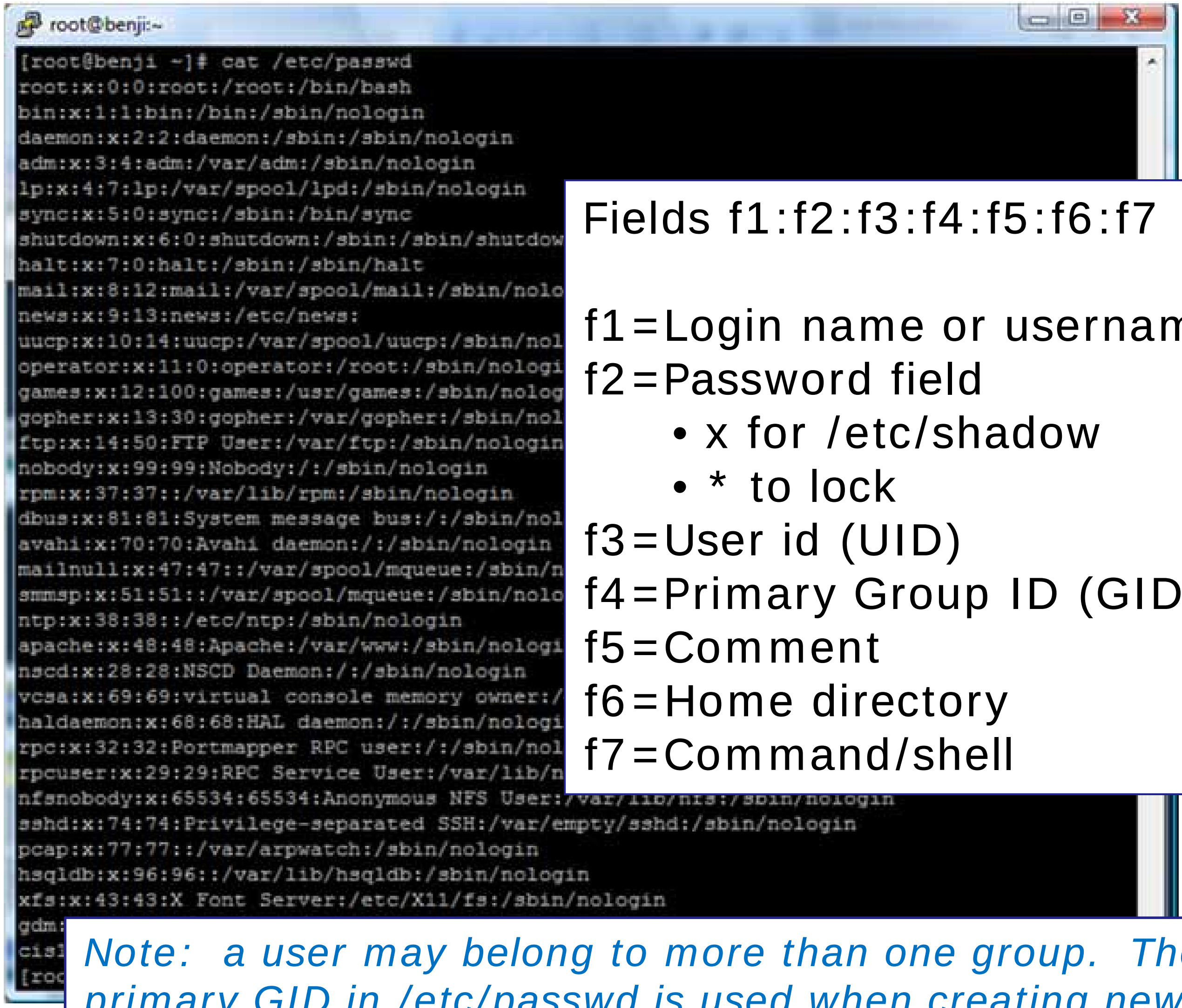
All user accounts are kept in /etc/passwd.

The user passwords are kept in /etc/shadow.

All the groups are kept in /etc/group.

The group passwords are kept in /etc/gshadow

/etc/passwd



```
[root@benji ~]# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdow
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nolo
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nol
operator:x:11:0:operator:/root:/sbin/nologi
games:x:12:100:games:/usr/games:/sbin/nolog
gopher:x:13:30:gopher:/var/gopher:/sbin/nol
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
rpm:x:37:37::/var/lib/rpm:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nol
avahi:x:70:70:Avahi daemon:/:/sbin/nologin
mailnull:x:47:47::/var/spool/mqueue:/sbin/n
smmsp:x:51:51::/var/spool/mqueue:/sbin/nolo
ntp:x:38:38::/etc/ntp:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologi
nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/
haldaemon:x:68:68:HAL daemon:/:/sbin/nologi
rpc:x:32:32:Portmapper RPC user:/:/sbin/nol
rpcuser:x:29:29:RPC Service User:/var/lib/n
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
pcap:x:77:77::/var/arpwatch:/sbin/nologin
hsqldb:x:96:96::/var/lib/hsqldb:/sbin/nologin
xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin
gdm:
cis
[roo
```

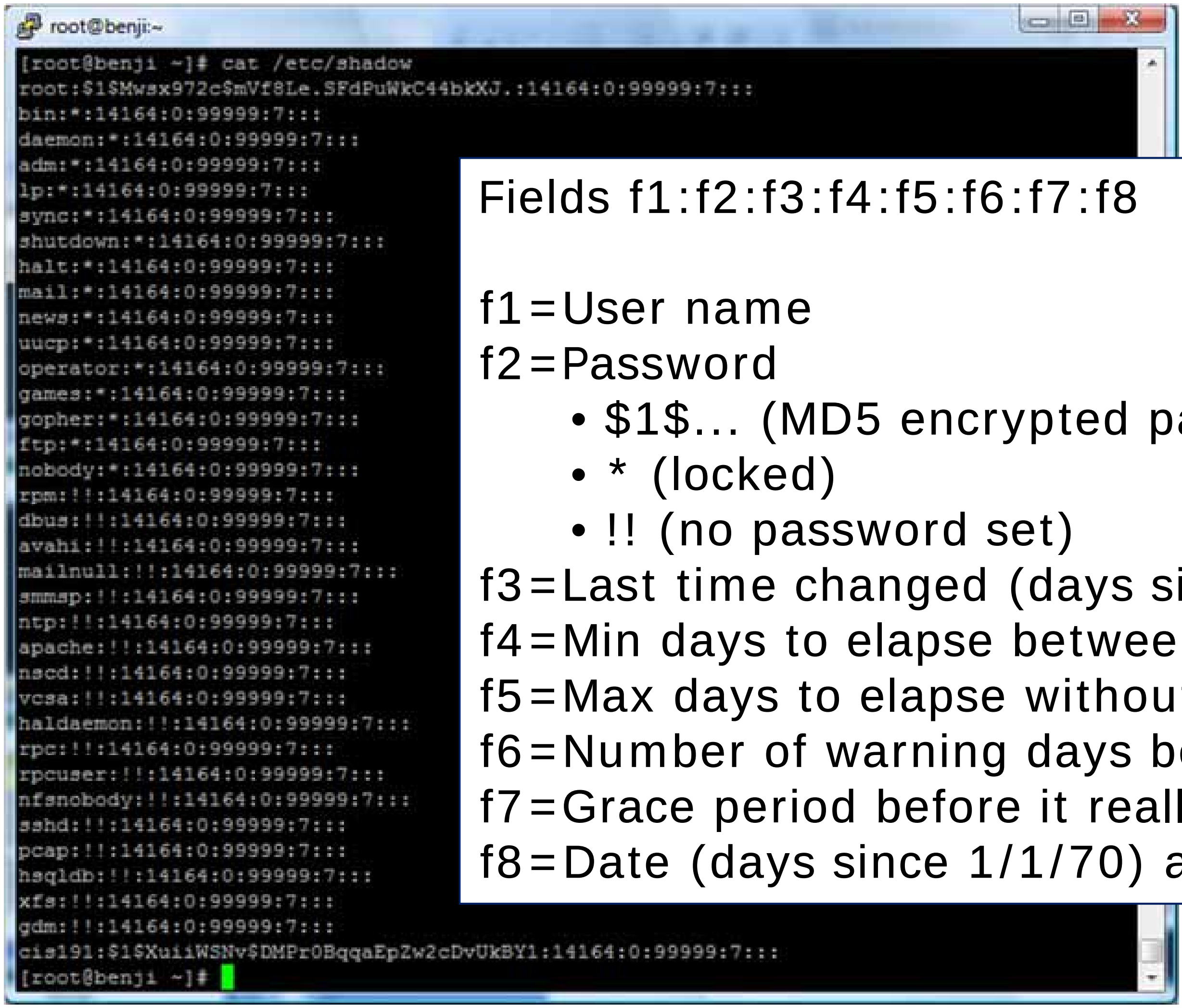
Fields f1:f2:f3:f4:f5:f6:f7

f1=Login name or username (up to 32 chars)
f2=Password field
- x for /etc/shadow
- * to lock

f3=User id (UID)
f4=Primary Group ID (GID)
f5=Comment
f6=Home directory
f7=Command/shell

Note: a user may belong to more than one group. The primary GID in /etc/passwd is used when creating new files

/etc/shadow



```
[root@benji ~]# cat /etc/shadow
root:$1$Mwsx972c$mVf8Le.SFdPuWkC44bkXJ.:14164:0:99999:7:::
bin:*:14164:0:99999:7:::
daemon:*:14164:0:99999:7:::
adm:*:14164:0:99999:7:::
lp:*:14164:0:99999:7:::
sync:*:14164:0:99999:7:::
shutdown:*:14164:0:99999:7:::
halt:*:14164:0:99999:7:::
mail:*:14164:0:99999:7:::
news:*:14164:0:99999:7:::
uucp:*:14164:0:99999:7:::
operator:*:14164:0:99999:7:::
games:*:14164:0:99999:7:::
gopher:*:14164:0:99999:7:::
ftp:*:14164:0:99999:7:::
nobody:*:14164:0:99999:7:::
rpm:!!:14164:0:99999:7:::
dbus:!!:14164:0:99999:7:::
avahi:!!:14164:0:99999:7:::
mailnull:!!:14164:0:99999:7:::
smmsp:!!:14164:0:99999:7:::
ntp:!!:14164:0:99999:7:::
apache:!!:14164:0:99999:7:::
nscd:!!:14164:0:99999:7:::
vcsa:!!:14164:0:99999:7:::
haldaemon:!!:14164:0:99999:7:::
rpc:!!:14164:0:99999:7:::
rpcuser:!!:14164:0:99999:7:::
nfsnobody:!!:14164:0:99999:7:::
sshd:!!:14164:0:99999:7:::
pcap:!!:14164:0:99999:7:::
hsqldb:!!:14164:0:99999:7:::
xfs:!!:14164:0:99999:7:::
gdm:!!:14164:0:99999:7:::
cis191:$1$XuiiWSNv$DMPr0BqqaEpZw2cDvUkBY1:14164:0:99999:7:::
[root@benji ~]#
```

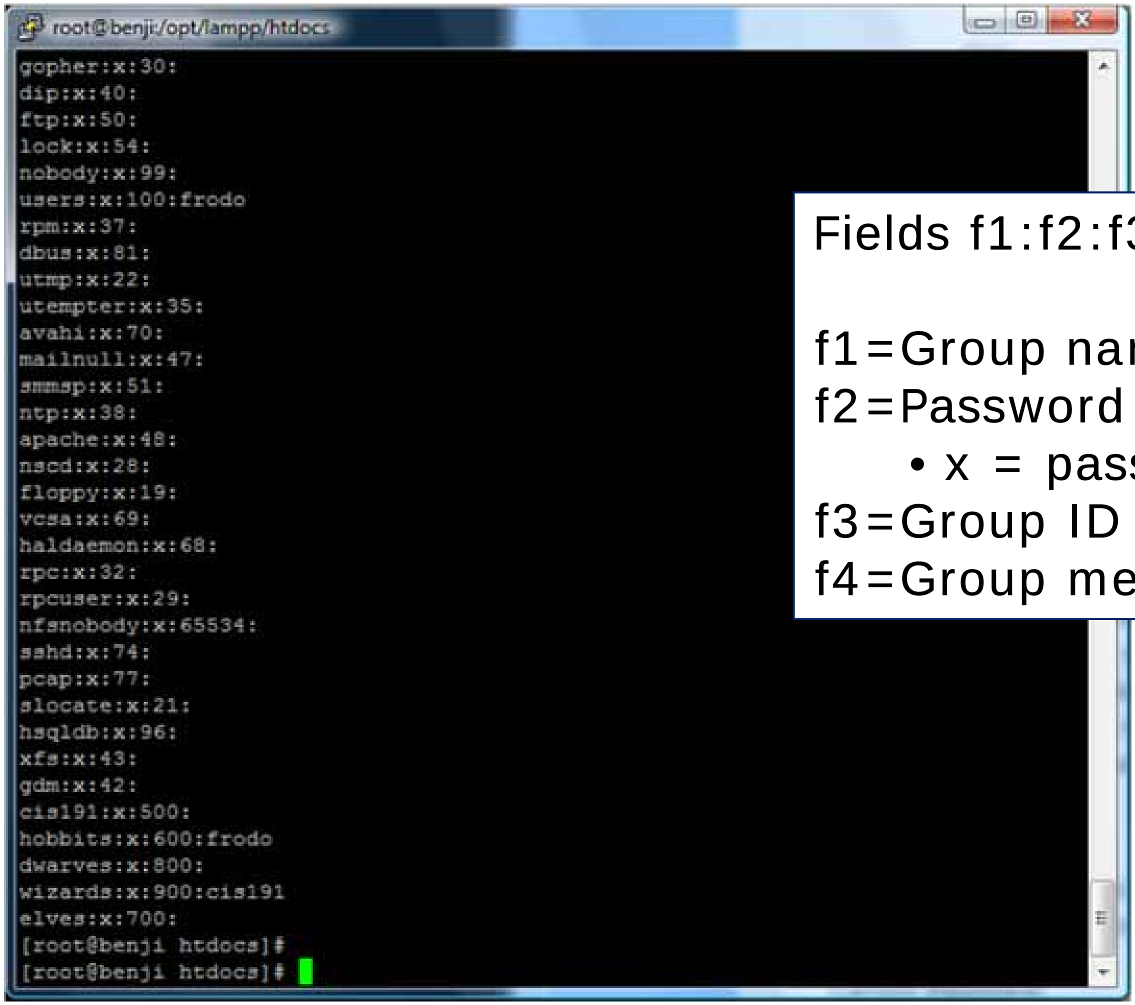
Fields f1:f2:f3:f4:f5:f6:f7:f8

f1=User name
f2=Password

- $1$... (MD5 encrypted password)
- * (locked)
- !! (no password set)

f3=Last time changed (days since 1/1/70)
f4=Min days to elapse between password changes
f5=Max days to elapse without changing password
f6=Number of warning days before expiration
f7=Grace period before it really expires
f8=Date (days since 1/1/70) account will expire

/etc/group



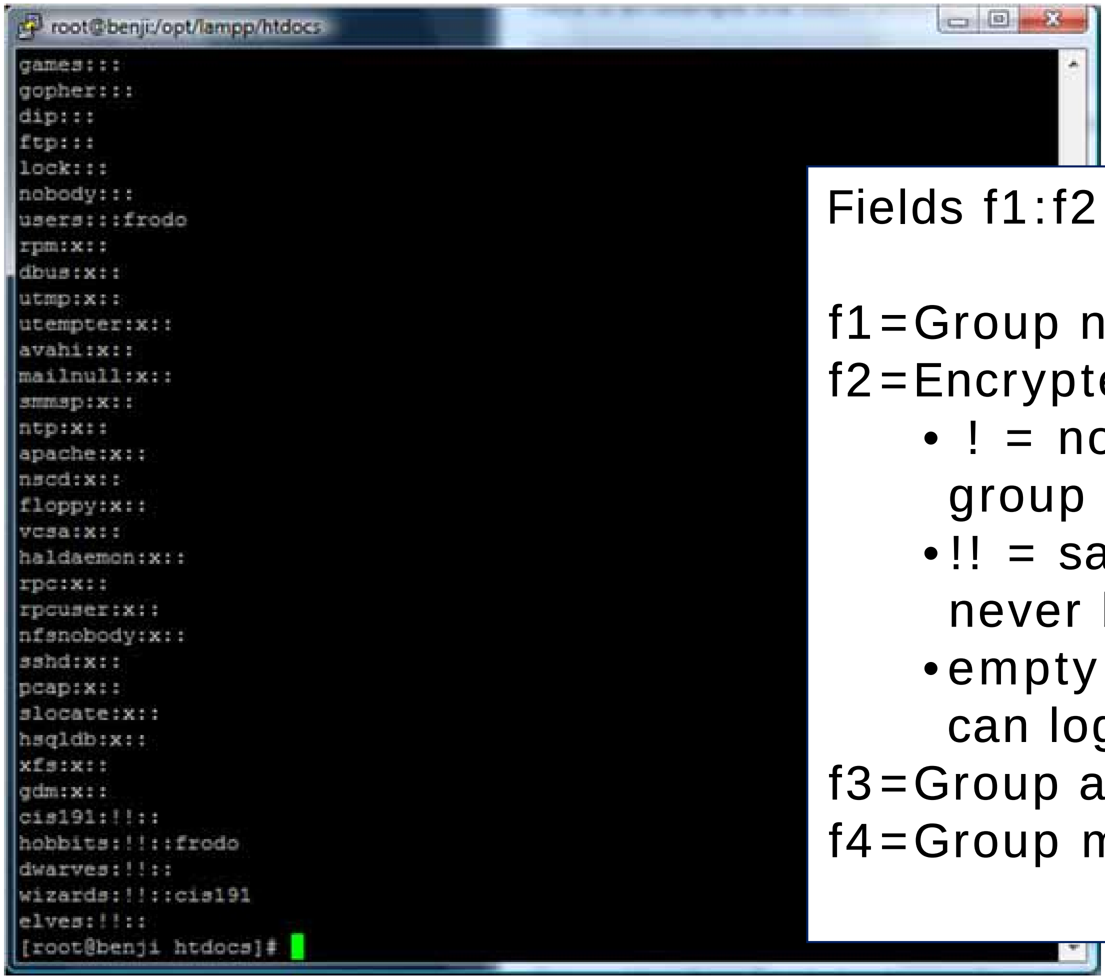
```
gopher:x:30:
dip:x:40:
ftp:x:50:
lock:x:54:
nobody:x:99:
users:x:100:frodo
rpm:x:37:
dbus:x:81:
utmp:x:22:
utempter:x:35:
avahi:x:70:
mailnull:x:47:
smmsp:x:51:
ntp:x:38:
apache:x:48:
nscd:x:28:
floppy:x:19:
vcsa:x:69:
haldaemon:x:68:
rpc:x:32:
rpcuser:x:29:
nfsnobody:x:65534:
sshd:x:74:
pcap:x:77:
slocate:x:21:
hsqldb:x:96:
xfs:x:43:
gdm:x:42:
cis191:x:500:
hobbits:x:600:frodo
dwarves:x:800:
wizards:x:900:cis191
elves:x:700:
[root@benji htdocs]#
[root@benji htdocs]#
```

Fields f1:f2:f3:f4

f1=Group name
f2=Password
- x = password in /etc/gshadow

f3=Group ID
f4=Group members (users)

/etc/gshadow



```
games:::
gopher:::
dip:::
ftp:::
lock:::
nobody:::
users:::frodo
rpm:x::
dbus:x::
utmp:x::
utempter:x::
avahi:x::
mailnull:x::
smmsp:x::
ntp:x::
apache:x::
nscd:x::
floppy:x::
vcsa:x::
haldaemon:x::
rpc:x::
rpcuser:x::
nfsnobody:x::
sshd:x::
pcap:x::
slocate:x::
hsqldb:x::
xfs:x::
gdm:x::
cis191:!!::
hobbits:!!::frodo
dwarves:!!::
wizards:!!::cis191
elves:!!::
[root@benji htdocs]#
```

Fields f1:f2:f3:f4

f1=Group name
f2=Encrypted password

- ! = no user allowed to access group using newgrp command
- !! = same as ! but password has never been set
- empty = only group members can log into the group

f3=Group administrators
f4=Group members

id command

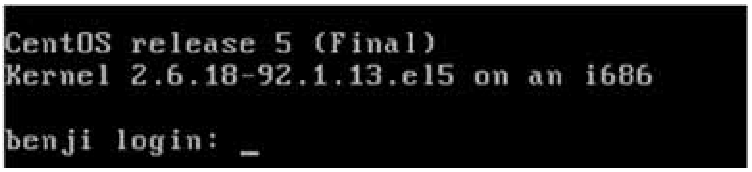
```
[root@benji htdocs]# id cis191
uid=500(cis191) gid=500(cis191)
groups=500(cis191)
context=root:system_r:unconfined_t:SystemLow-
SystemHigh

[root@benji htdocs]# id root
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(d
isk),10(wheel)
context=root:system_r:unconfined_t:SystemLow-
SystemHigh
```

Note: id command in newer distros shows SELinux contexts for users

Shell

The shell is started once you log in

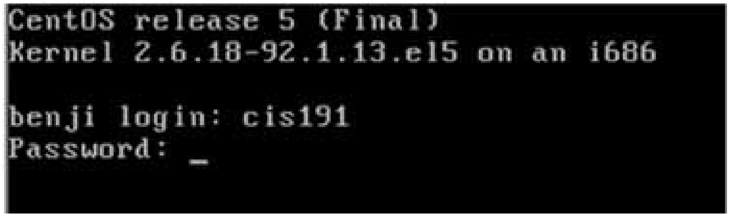
1) init starts up the mingetty process on each terminal which prompts for login username, gets it, then starts login.



```
CentOS release 5 (Final)
Kernel 2.6.18-92.1.13.el5 on an i686

benji login: _
```

```
[cis191@benji ~]$ ps t tty1
  PID TTY      STAT   TIME COMMAND
 3557 tty1     Ss+    0:00 /sbin/mingetty tty1
```

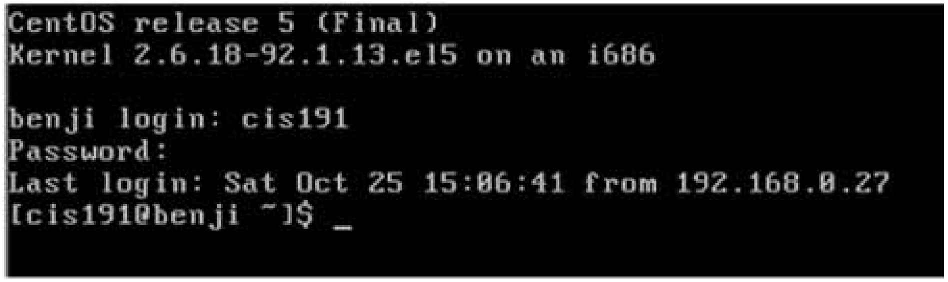
2) login collects the password and checks it with /etc/passwd and /etc/shadow



```
CentOS release 5 (Final)
Kernel 2.6.18-92.1.13.el5 on an i686

benji login: cis191
Password: _
```

```
[cis191@benji ~]$ ps t tty1
  PID TTY      STAT   TIME COMMAND
 3557 tty1     Ss+    0:00 /bin/login –
```

3) login then starts up the shell specified in the /etc/passwd file

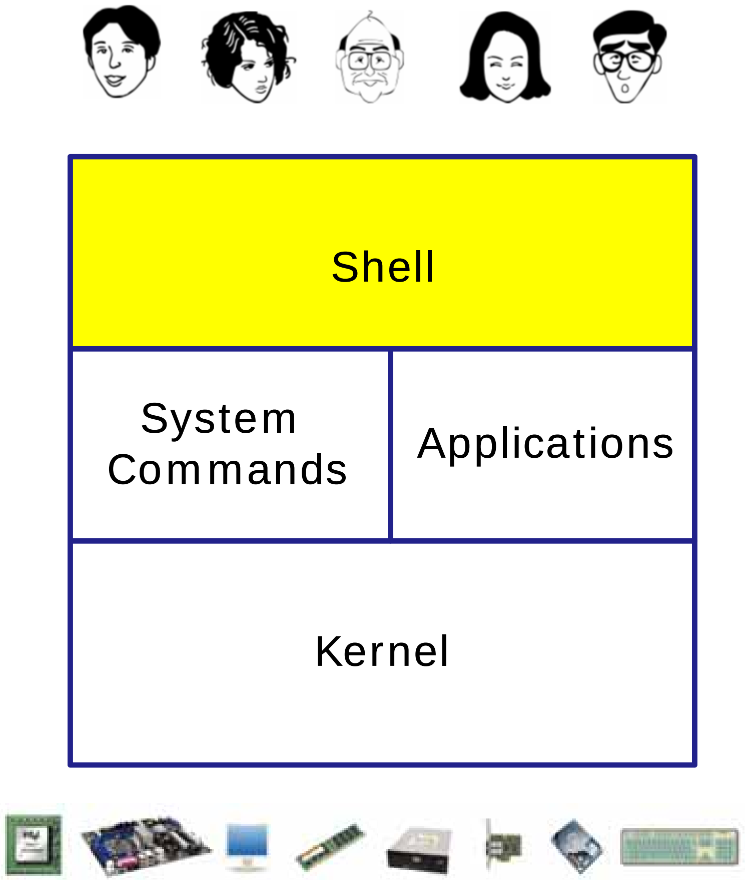


```
CentOS release 5 (Final)
Kernel 2.6.18-92.1.13.el5 on an i686

benji login: cis191
Password:
Last login: Sat Oct 25 15:06:41 from 192.168.0.27
[cis191@benji ~]$ _
```

```
[cis191@benji ~]$ ps t tty1
  PID TTY      STAT   TIME COMMAND
 3603 tty1     Ss+    0:00 –bash
```

This is the point where the shell gets started

Life of the Shell



Shell

System Commands

Applications

Kernel

1) **Prompt** for a command
2) **Parse** (interpret metacharacters, expand file names and dissect command line into options and arguments)
3) **Search** for program (along the path)
4) **Execute** program by loading into memory (becomes a process), hookup input and outputs, and pass along command line options and arguments.
5) **Nap** (wait till process is done)
6) **Repeat**

Prompt

Life of the Shell

1) Prompt user for a command

Note: The shell uses the current setting of the PS1 variable to form the prompt string

Examples:

```
[rsimms@opus work]$
```

To get this prompt, use `PS1='[\u@\h \W]\$ '`

```
/home/cis90/roddyduk $
```

To get this prompt, use `PS1='$PWD $'`

Notes:

- When setting the PS1 variable, use ' (single quotes) to prevent shell from expanding metacharacters.
- To display the prompt variable use echo $PS1
- Some useful PS1 special character codes:
  - \h = hostname
  - \u = user name
  - \W = working directory
  - \$ = $ for normal users, # for root

What environment variable determines my prompt string?

PS1

How do I make my prompt be "Enter command: "

PS1="Enter command: "

How would I make my prompt show my username, the computer I'm using, the current directory, all in [], followed by a $? for example: [rsimms@opus misc]$

```
PS1='[\u@\h \W]\$ '
```

See Lesson 2, Slide 104 or previous slide

How do I make my prompt be the absolute pathname of the current directory?

```
PS1='$PWD $'
```

OS Parse

Life of the Shell

2) Parse command user typed

(analyze and dissect text string into tokens)

- Process all the metacharacters
- Identify the command, the options and arguments to pass to the command
- Determine the I/O needs by looking at pipe (|) and redirection symbols (<, >, >>, 2>).

Note: metacharacters include:

- *filename expansion characters like *, [] and ?*
- *$ for the value of a variable*
- *; for separating commands*
- *Double (") and single (') quotes. Single quoted strings are not expanded further by the shell.*

Parse

Command Syntax

Command	Options	Arguments	Redirection

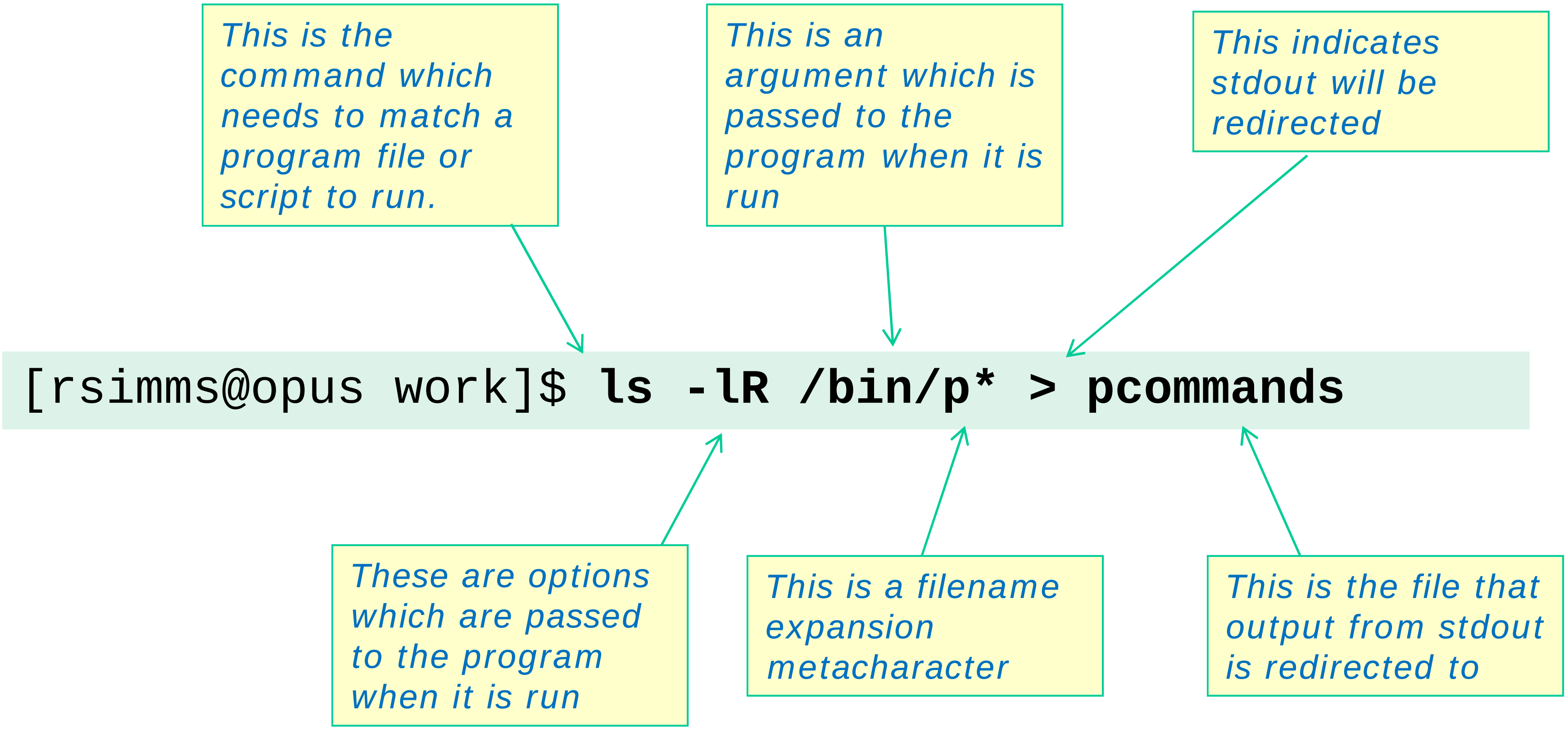
Command – is the name of an executable program file.
Options – various options which control how the program will operate.
Arguments – the objects the command is directed to work upon.
Redirection – The default input stream (stdin) is from the console keyboard, the default output (stdout) and error (stderr) streams go to the console screen. Redirection can modify these streams to other files or devices.

Parse

Life of the Shell

2) Parse command user typed

(analyze and dissect text string into tokens)



This is the command which needs to match a program file or script to run.

This is an argument which is passed to the program when it is run

This indicates stdout will be redirected

```
[rsimms@opus work]$ ls -lR /bin/p* > pcommands
```

These are options which are passed to the program when it is run

This is a filename expansion metacharacter

This is the file that output from stdout is redirected to

Life of the Shell

3) Search for the program file to run

(only look in directories on the PATH)

/bin directory is on the path

```
[rsimms@opus work]$ echo $PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/hom
e/rsimms/bin

[rsimms@opus work]$ type -a ls
ls is aliased to `ls --color=tty'
ls is /bin/ls
[rsimms@opus work]$

[rsimms@opus work]$ ls /bin/ls
/bin/ls
```

***type** command shows that ls is in the /bin directory*

***ls** command lists the ls file and it is executable (green)*

OS Search

What the heck !!@@##

The Shell and the PATH

Four commands: hostname, ps, iptables and ifconfig

```
[rsimms@opus ~]$ ls /bin/hostname /bin/ps
/bin/hostname  /bin/ps
[rsimms@opus ~]$ ls /sbin/iptables /sbin/ifconfig
/sbin/ifconfig  /sbin/iptables
```

Note: We (the humans) can find all four files on the system just by looking in the right directories

Two work and two don‘t:

```
[rsimms@opus ~]$ hostname
opus.cabrillo.edu
[rsimms@opus ~]$ ps
  PID TTY          TIME CMD
14801 pts/0    00:00:00 bash
14902 pts/0    00:00:00 ps
[rsimms@opus ~]$ iptables -L
-bash: iptables: command not found   ⬅ !!@@##
[rsimms@opus ~]$ ifconfig
-bash: ifconfig: command not found   ⬅ !!@@##
```

Why can't bash (the computer) find them?

Search

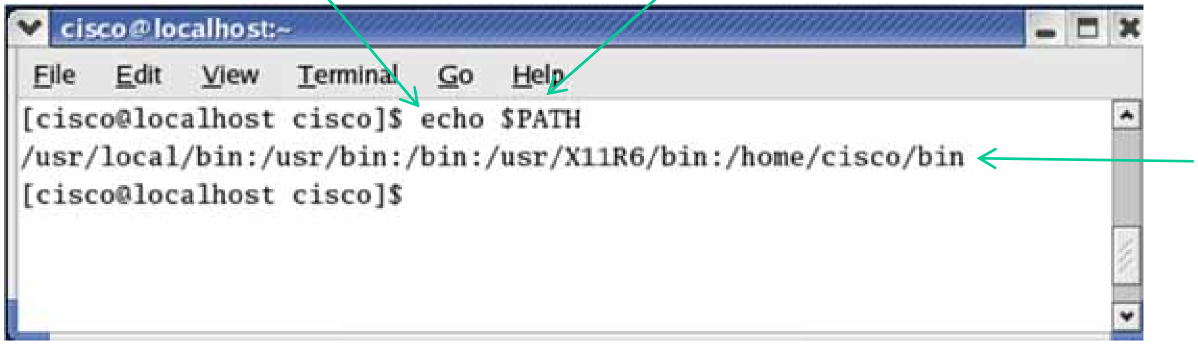
What the heck !!@@##

The Shell and the PATH

- *The shell will only search for commands on the “path”*
- *The path is determined by the environment variable PATH*
- *Use echo $PATH to see your current path*

echo command prints a text string

The $ means “the value of”



```
[cisco@localhost cisco]$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/home/cisco/bin
[cisco@localhost cisco]$
```

This user’s path has the following directories:

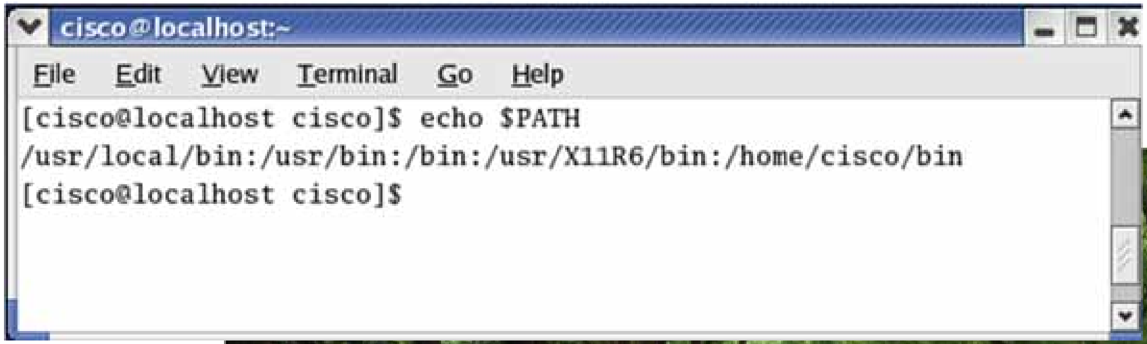
1. /usr/local/bin
2. /usr/bin
3. /bin
4. /usr/X11R6/bin
5. /home/cisco/bin

The order is important as it determines the order in which the directories are searched by the shell for a command

Search

What the heck !!@@##

The Shell and the PATH



```
[cisco@localhost cisco]$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/home/cisco/bin
[cisco@localhost cisco]$
```

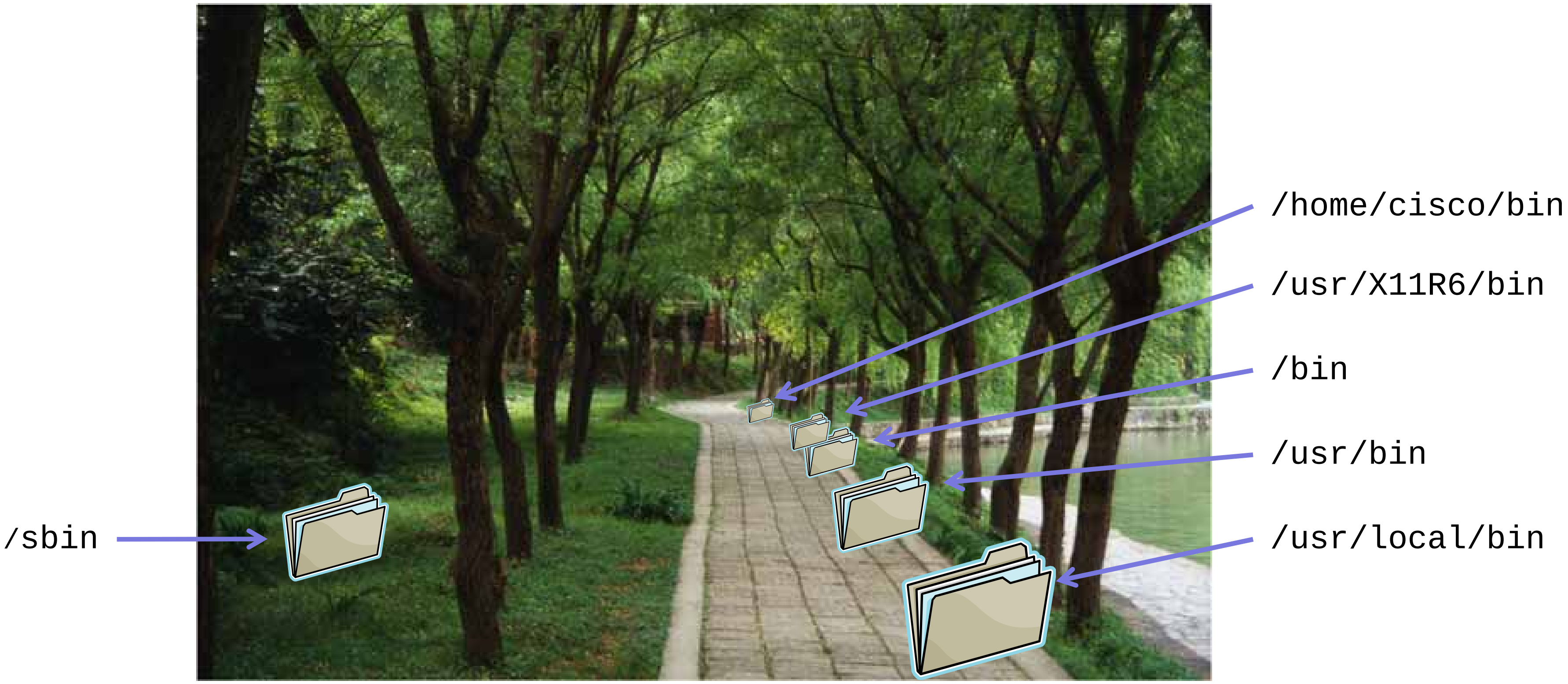
Think of the path like this one

Search

What the heck !!@@##

The Shell and the PATH

Some directories are on the path and some are not

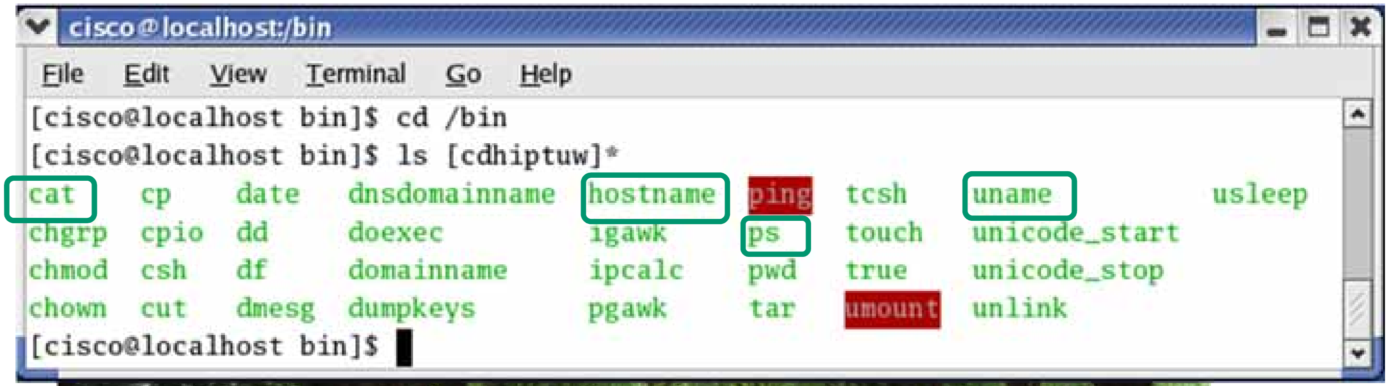


This directory (and many others) is NOT on the path

These directories are on the path

The Shell and the PATH

Search



```
[cisco@localhost bin]$ cd /bin
[cisco@localhost bin]$ ls [cdhiptuw]*
cat    cp    date   dnsdomainname  hostname  ping  tcsh    uname          usleep
chgrp  cpio  dd     doexec         igawk     ps    touch   unicode_start
chmod  csh   df     domainname     ipcalc    pwd   true    unicode_stop
chown  cut   dmesg  dumpkeys       pgawk     tar   umount  unlink
[cisco@localhost bin]$
```

The cat, hostname, ps and uname commands are in the /bin directory

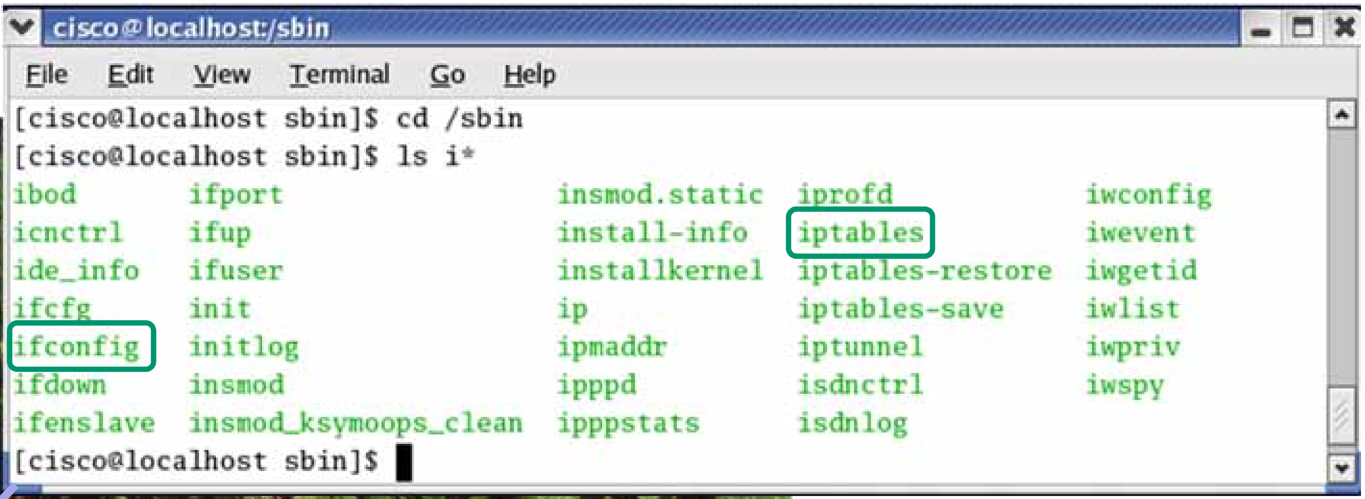
The /bin directory is on the path

```
[rsimms@opus ~]$ hostname
opus.cabrillo.edu
[rsimms@opus ~]$ ps
  PID TTY          TIME CMD
14801 pts/0    00:00:00 bash
14902 pts/0    00:00:00 ps
```

Those commands work just fine

Search

The Shell and the PATH



```
[cisco@localhost sbin]$ cd /sbin
[cisco@localhost sbin]$ ls i*
ibod       ifport                 insmod.static  iprofd            iwconfig
icnctrl    ifup                   install-info   iptables          iwevent
ide_info   ifuser                 installkernel  iptables-restore  iwgetid
ifcfg      init                   ip             iptables-save     iwlist
ifconfig   initlog                ipmaddr        iptunnel          iwpriv
ifdown     insmod                 ipppd          isdnctrl          iwspy
ifenslave  insmod_ksymoops_clean  ipppstats      isdnlog
[cisco@localhost sbin]$ 
```

The ifconfig and iptables commands are in the /sbin directory ...

... and the /sbin directory is NOT on the path

```
[rsimms@opus ~]$ iptables -L
-bash: iptables: command not found
[rsimms@opus ~]$ ifconfig
-bash: ifconfig: command not found
```

~~!!@@##~~

~~!!@@##~~

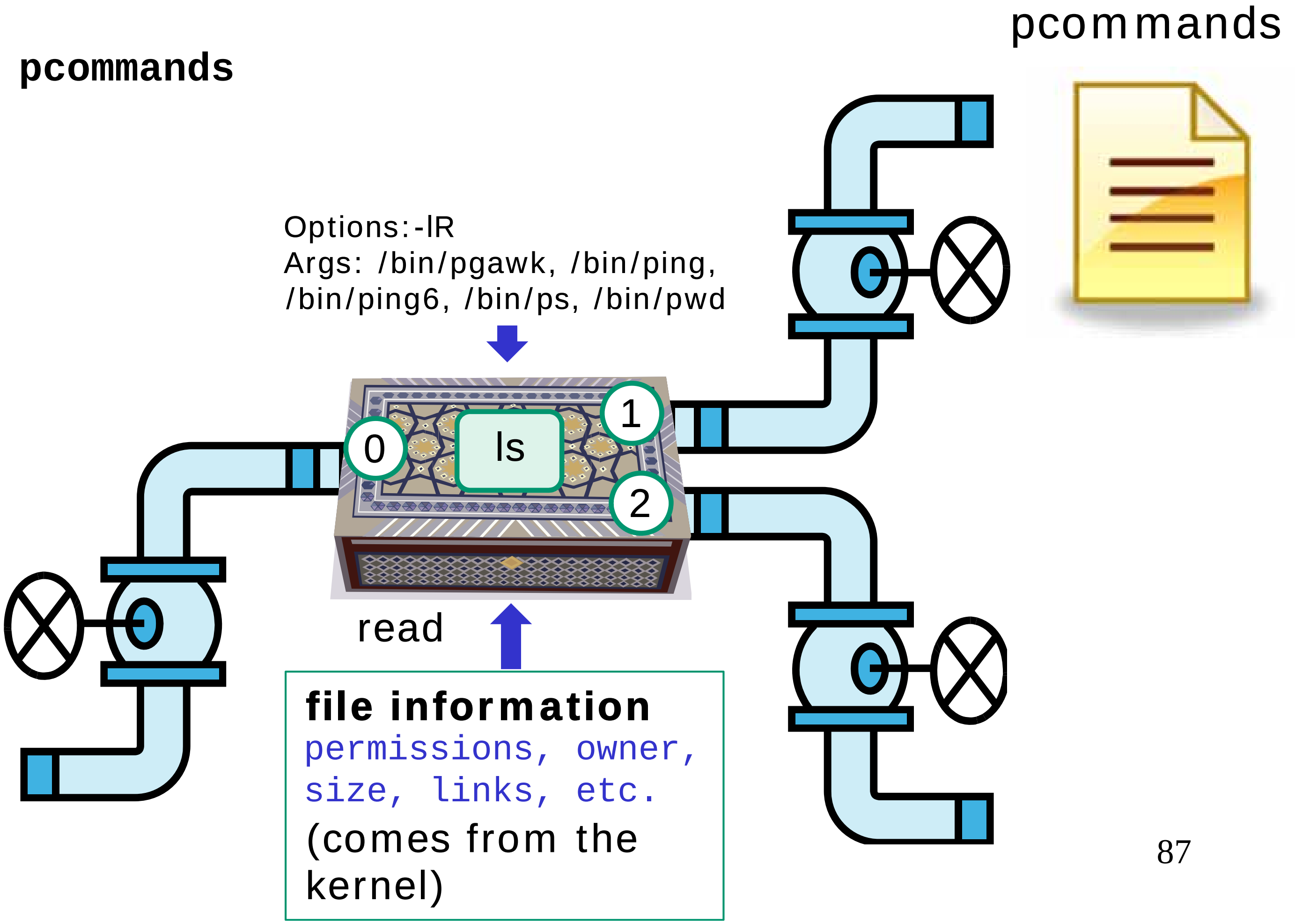
OK, makes sense now

Execute

Life of the Shell

4) Execute the command

```
ls -lR /bin/p* > pcommands
```



file information
permissions, owner, size, links, etc.
(comes from the kernel)

Nap

Life of the Shell

5) Nap while the command (process) runs to completion

(The shell (itself a loaded process) goes into the sleep state and waits till the command process is finished)

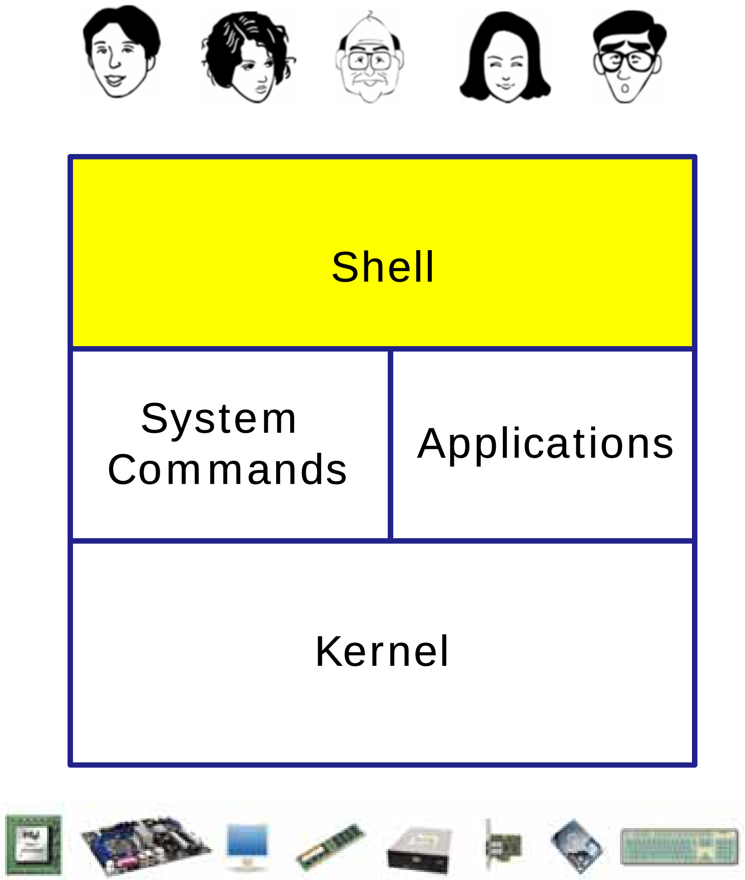
```
[rsimms@opus work]$ ls -lR /bin/p* > pcommands



[rsimms@opus work]$ cat pcommands
-rwxr-xr-x 1 root root 321216 Jan 15  2007 /bin/pgawk
-rwsr-xr-x 1 root root  35864 Dec 21  2006 /bin/ping
-rwsr-xr-x 1 root root  31244 Dec 21  2006 /bin/ping6
-r-xr-xr-x 1 root root  79068 Jan  2  2008 /bin/ps
-rwxr-xr-x 1 root root  22980 Nov 30  2007 /bin/pwd
[rsimms@opus work]$
```

Repeat

Life of the Shell

6) And do it all over again … go to step 1

Life of the Shell



Shell

System Commands

Applications

Kernel

This slide shown again for EMPHASIS!

1) **Prompt** for a command
2) **Parse** (interpret metacharacters, expand file names and dissect command line into options and arguments)
3) **Search** for program (along the path)
4) **Execute** program by loading into memory (becomes a process), hookup input and outputs, and pass along command line options and arguments.
5) **Nap** (wait till process is done)
6) **Repeat**

Life of the Shell

Practice being the Shell

Given:

- user=roddyduk, $PWD=~, hostname=opus
- PS1 is: `'[\u@\h \W]\$'`
- path is: `/bin:/usr/bin:`
- command is: **`ls -lR /bin/p* > pcommands`**

1) Generate the prompt: `[roddyduk@opus ~]$`

2) Parse the command line:
   - command = ls
   - options = lR
   - arguments = /bin/pgawk /bin/ping /bin/ping6 /bin/ps /bin/pwd
   - redirection = stdout redirected to pcommand file

3) Is the command on the path? yes

Metacharacters

Metacharacters

Have special interpretation by the shell

Char	Description
\	Treat the following metacharacter as a plain character. Also called "escaping" the next character.
$	The following text is a shell (environment) variable and the value should be used.
<cr>	Carriage return marks the end of the command
;	Separates multiple commands on one line
'	used to enclose a string that the shell will not do further interpretation
"	Used to enclose a string that the shell will do further interpretation.
>	Redirects stdout
2>	Redirects stderr
*	Matches all non-hidden file names when used alone or zero or more characters when used as prefix, infix or postfix
?	Matches any single character of a file name
[]	Matches any single character contained within the brackets
#	Not an official metacharacter, but any text following the # is ignored by the shell

Metacharacters

Have special interpretation by the shell

```
/home/cis90/simmsben $ #OK lets escape the carriage return in next example
/home/cis90/simmsben $ echo Lets start line 1 here \
> and finish it here
Lets start line 1 here and finish it here
/home/cis90/simmsben $

/home/cis90/simmsben $ #Notice single quoted strings are not interpreted
/home/cis90/simmsben $ echo "I am in $PWD"
I am in /home/cis90/simmsben
/home/cis90/simmsben $ echo 'I am in $PWD'
I am in $PWD
/home/cis90/simmsben $

/home/cis90/simmsben $ #Lets put two commands on one line
/home/cis90/simmsben $ echo "This is my terminal device:"; tty
This is my terminal device:
/dev/pts/2
/home/cis90/simmsben $
```

Filename Expansion Characters

Special characters that your shell recognizes to make it easier to specify file names. (wildcards)

* matches all non-hidden filenames in the current directory when used alone matches zero or more characters when used as a prefix, infix or postfix.

? matches any single character in any of your current directory's filenames.

[] matches any single character contained within the brackets.

Metacharacters

File name expansion characters

```
/home/cis90/simmsben $ #Show all files, hidden and non-hidden
/home/cis90/simmsben $ ls -a
.               bigfile   Lab2.1          .plan          salsa       what_am_i
..              bin       .lesshst        Poems          small_town  .Xauthority
.bash_history   deleteme  letter          proposal1      spellk      .zshrc
.bash_logout    .emacs    mbox            proposal2      text.err
.bash_profile   empty     Miscellaneous   proposal3      text.fxd
.bashrc         Hidden    mission         results-e1     timecal
bcommands       Lab2.0    .mozilla        results-e1a    .viminfo
/home/cis90/simmsben $

/home/cis90/simmsben $ # * matches all non-hidden file names
/home/cis90/simmsben $ echo *
bcommands bigfile bin deleteme empty Hidden Lab2.0 Lab2.1 letter mbox
Miscellaneous mission Poems proposal1 proposal2 proposal3 results-e1 results-
e1a salsa small_town spellk text.err text.fxd timecal what_am_i

/home/cis90/simmsben $ #Show files with a period (differs from DOS)
/home/cis90/simmsben $ echo *.*
Lab2.0 Lab2.1 text.err text.fxd
```

Metacharacters

File name expansion characters

Char	Description
*	Matches all non-hidden file names when used alone or zero or more characters when used as prefix, infix or postfix
?	Matches any single character of a file name
[]	Matches any single character contained within the brackets

```
/home/cis90/simmsben/Poems $ # Using *, ? and []
/home/cis90/simmsben/Poems $ ls -a
.   ..   ant   Blake   nursery   Shakespeare   twister   Yeats
/home/cis90/simmsben/Poems $ echo *
ant Blake nursery Shakespeare twister Yeats
/home/cis90/simmsben/Poems $ echo ../p*
../proposal1 ../proposal2 ../proposal3

/home/cis90/simmsben/Poems $ echo B???e
Blake
/home/cis90/simmsben/Poems $ echo [SB]*
Blake Shakespeare
/home/cis90/simmsben/Poems $
```

All files in current directory

All files in parent directory starting with p

All five letter file names starting with B and ending with e

All files names starting with S or B

Environment Variables

Shell (Environment) Variables

common environment variables

Shell Variable	Description
HOME	Users home directory (starts here after logging in and returns with a cd command (with no arguments)
LOGNAME	User's username for logging in with.
PATH	List of directories, separated by :'s, for the Shell to search for commands (which are program files) .
PS1	The prompt string.
PWD	Current working directory
SHELL	Name of the Shell program being used.
TERM	Type of terminal device , e.g. dumb, vt100, xterm, ansi, etc.

Shell (Environment) Variables

Show and set variable values

Lets look at some of the key environment variables using echo command

```
/home/cis90/simmsben/Poems $ # Print some of the shell variables
/home/cis90/simmsben/Poems $ echo $HOME $LOGNAME $PS1 $PWD $SHELL $TERM
/home/cis90/simmsben simmsben $PWD $ /home/cis90/simmsben/Poems /bin/bash xterm
```

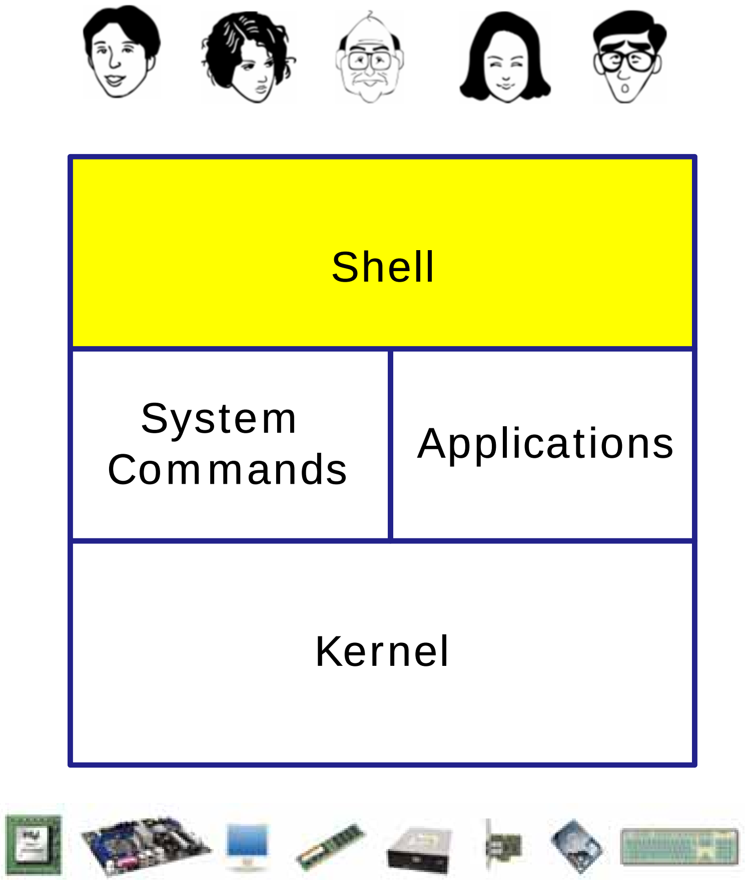
Lets look at our path

```
/home/cis90/simmsben/Poems $ echo $PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/home/cis90/simmsben/../bin:/home/cis90/simm
sben/bin:.
/home/cis90/simmsben/Poems $
```

Lets change a variable

```
/home/cis90/simmsben/Poems $ # Change the prompt variable
/home/cis90/simmsben/Poems $ PS1='[\u@\h \W]\$'
[simmsben@opus Poems]$# Change it back again
[simmsben@opus Poems]$PS1='$PWD $'
/home/cis90/simmsben/Poems $
```

Life of the Shell



Shell

System Commands

Applications

Kernel

1) **Prompt** for a command
2) **Parse** (interpret metacharacters, expand file names and dissect command line into options and arguments)
3) **Search** for program (along the path)
4) **Execute** program by loading into memory (becomes a process), hookup input and outputs, and pass along command line options and arguments.
5) **Nap** (wait till process is done)
6) **Repeat**

Q30

From a previous Test #2

Test 2 Q30

30. Issue the following command:

```
ls -l /bin/p[gws]?* > /dev/null
```

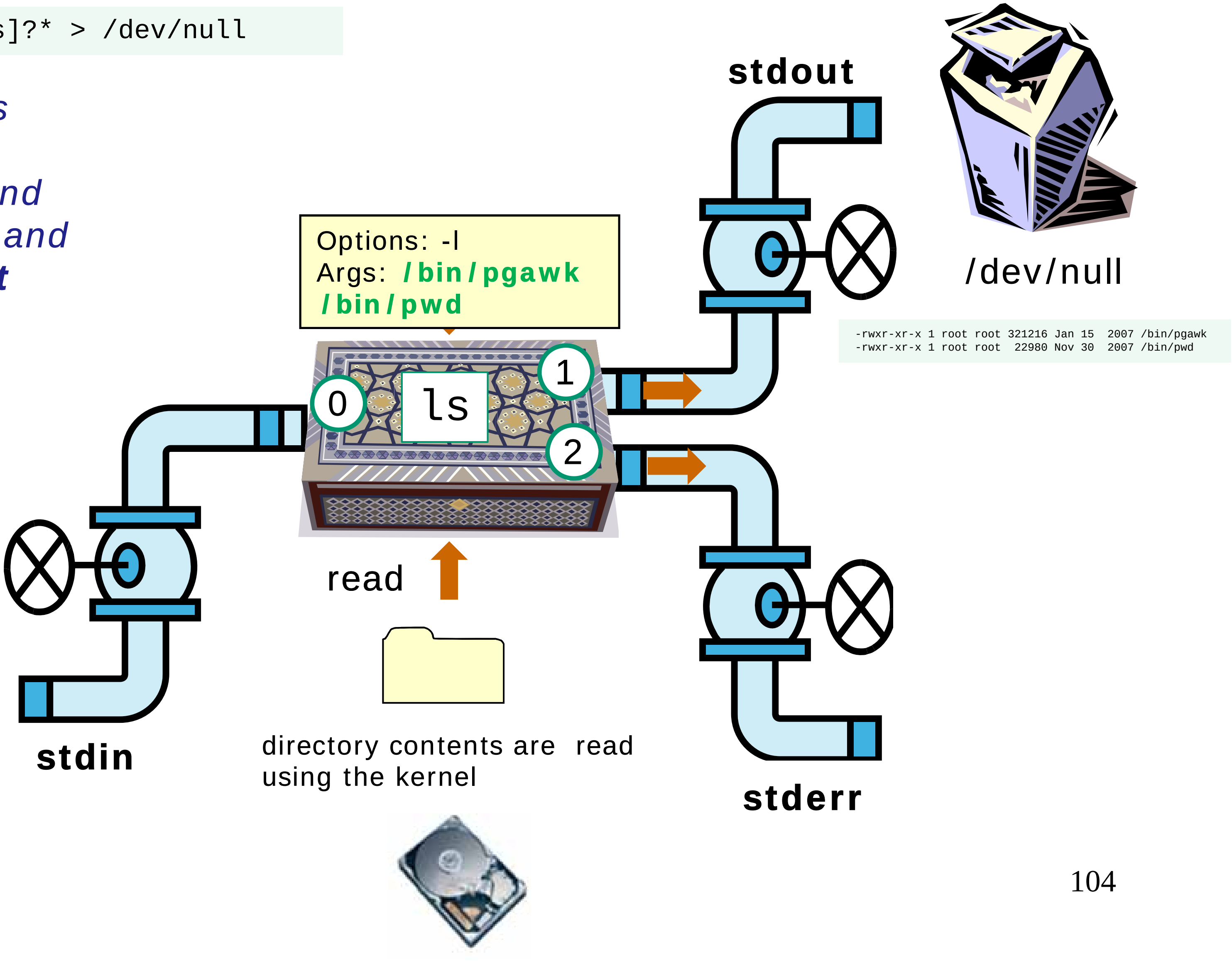
What argument(s) are being passed to the ls command when it is loaded?

`/bin/pgawk /bin/pwd`

Test 2 Q30 explained

```
$ ls -l /bin/p[gws]?* > /dev/null
```

*Note: ls gets its input from the command line and the OS (kernel) and writes to **stdout** (redirected to /dev/null) and **stderr**.*



Options: -l
Args: **/bin/pgawk /bin/pwd**

```
-rwxr-xr-x 1 root root 321216 Jan 15  2007 /bin/pgawk
-rwxr-xr-x 1 root root  22980 Nov 30  2007 /bin/pwd
```

directory contents are read using the kernel

Test 2 Q30 verification

30. Issue the following command:
`ls -l /bin/p[gws]?* > /dev/null`
What argument(s) are being passed to the `ls` command when it is loaded?

```
/home/cis90/roddyduk $ echo /bin/p[gws]?*
/bin/pgawk /bin/pwd
```

To verify, use the echo command

or

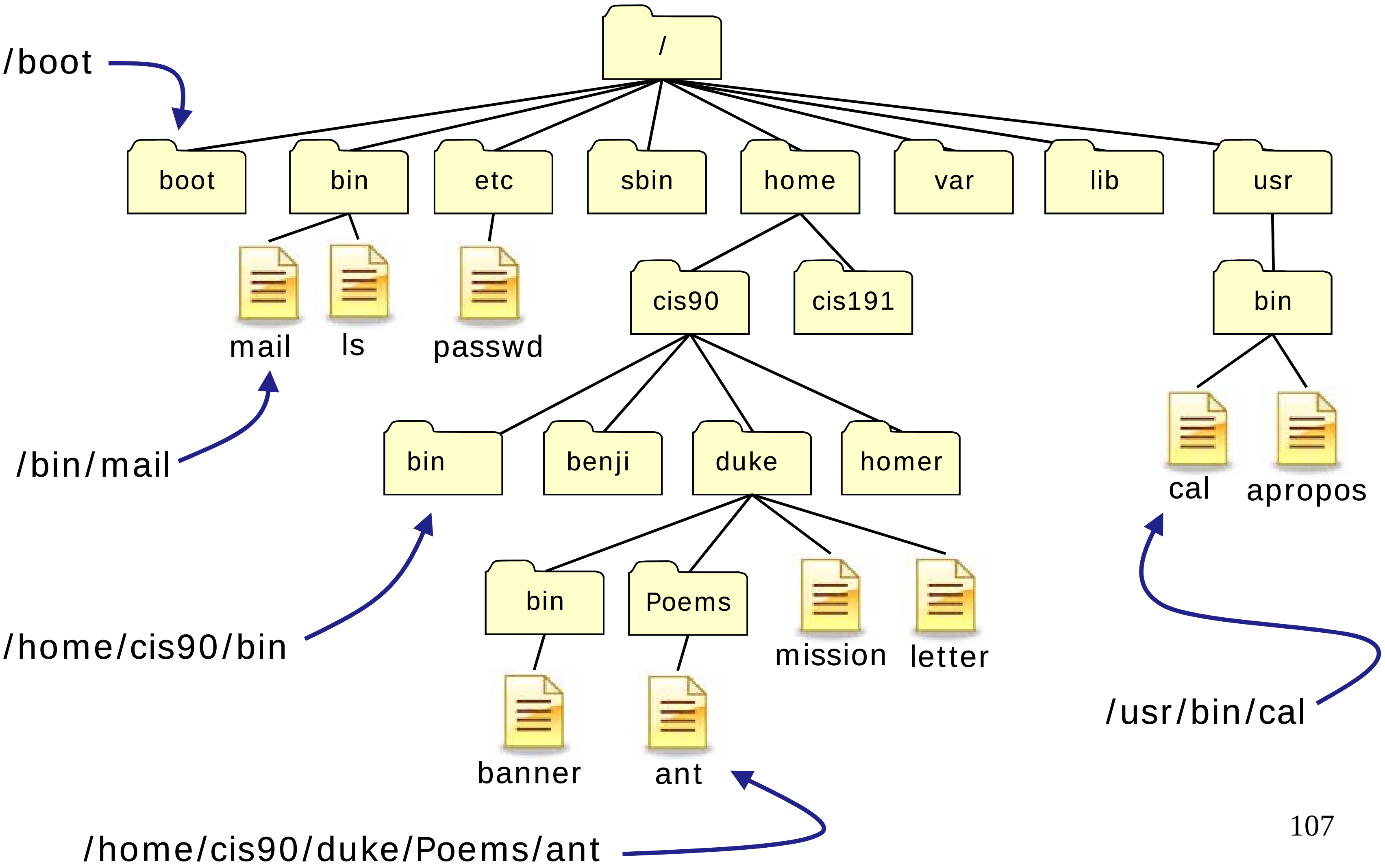
```
/home/cis90/roddyduk $ set -x
++ echo -ne '\033]0;roddyduk@opus:~'

/home/cis90/roddyduk $ ls -l /bin/p[gws]?* > /dev/null
+ ls --color=tty -l /bin/pgawk /bin/pwd
++ echo -ne '\033]0;roddyduk@opus:~'

/home/cis90/roddyduk $
```

Could also turn on bash tracing

File System

Absolute Pathnames

Fully specified names starting with /



/boot

/bin/mail

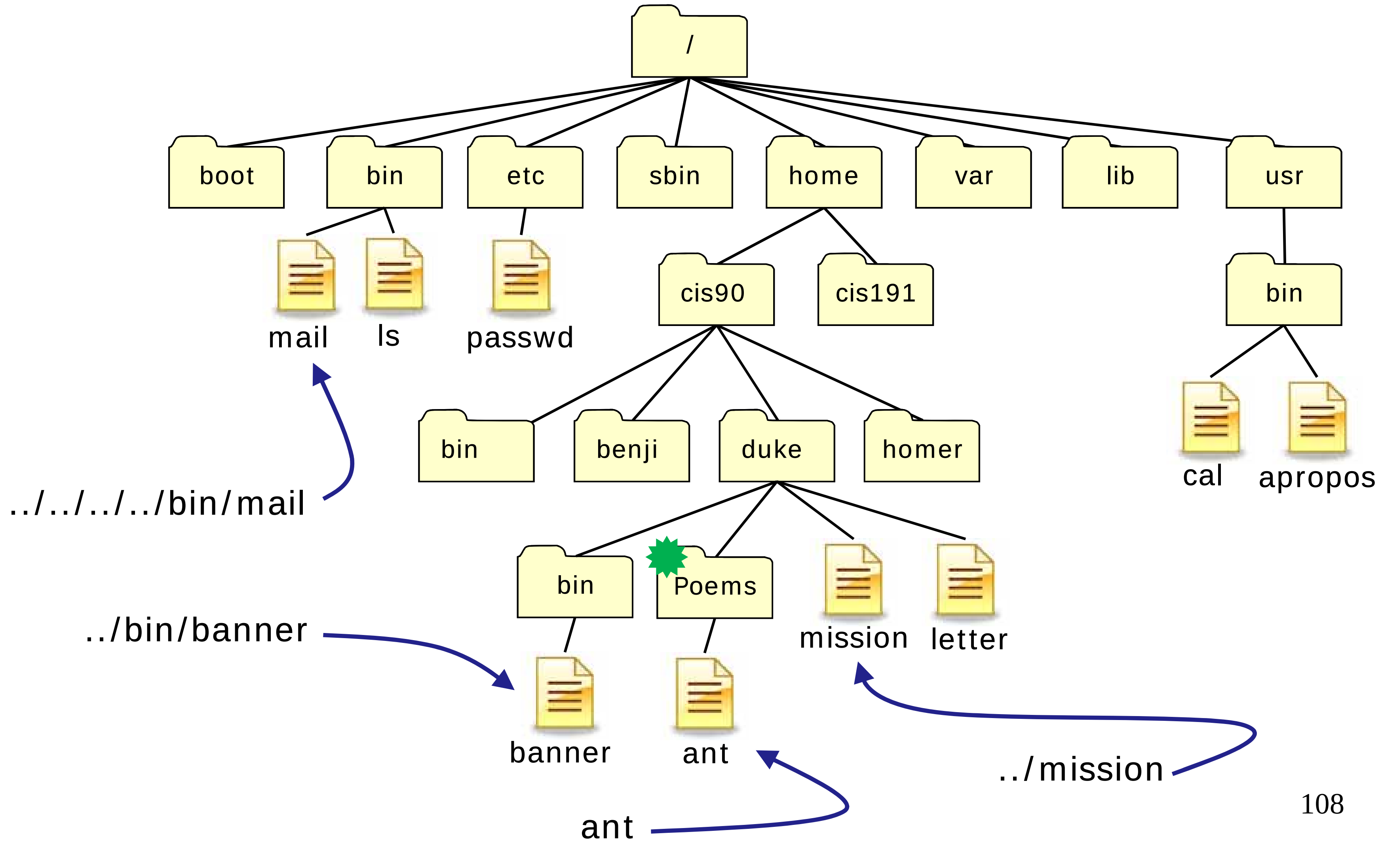
/home/cis90/bin

/usr/bin/cal

/home/cis90/duke/Poems/ant

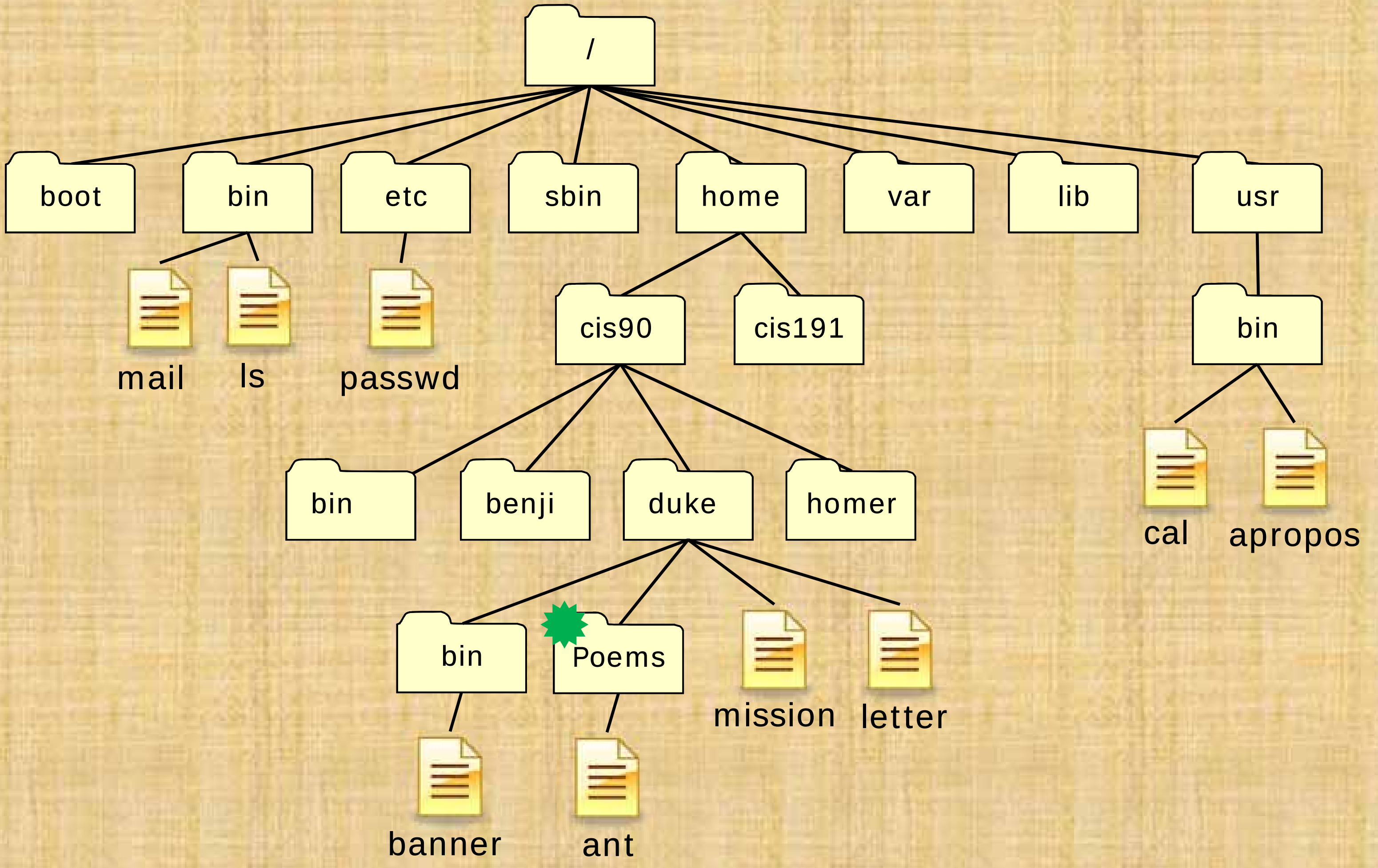
Relative Pathnames

Names that start relative to the current working directory (✸)



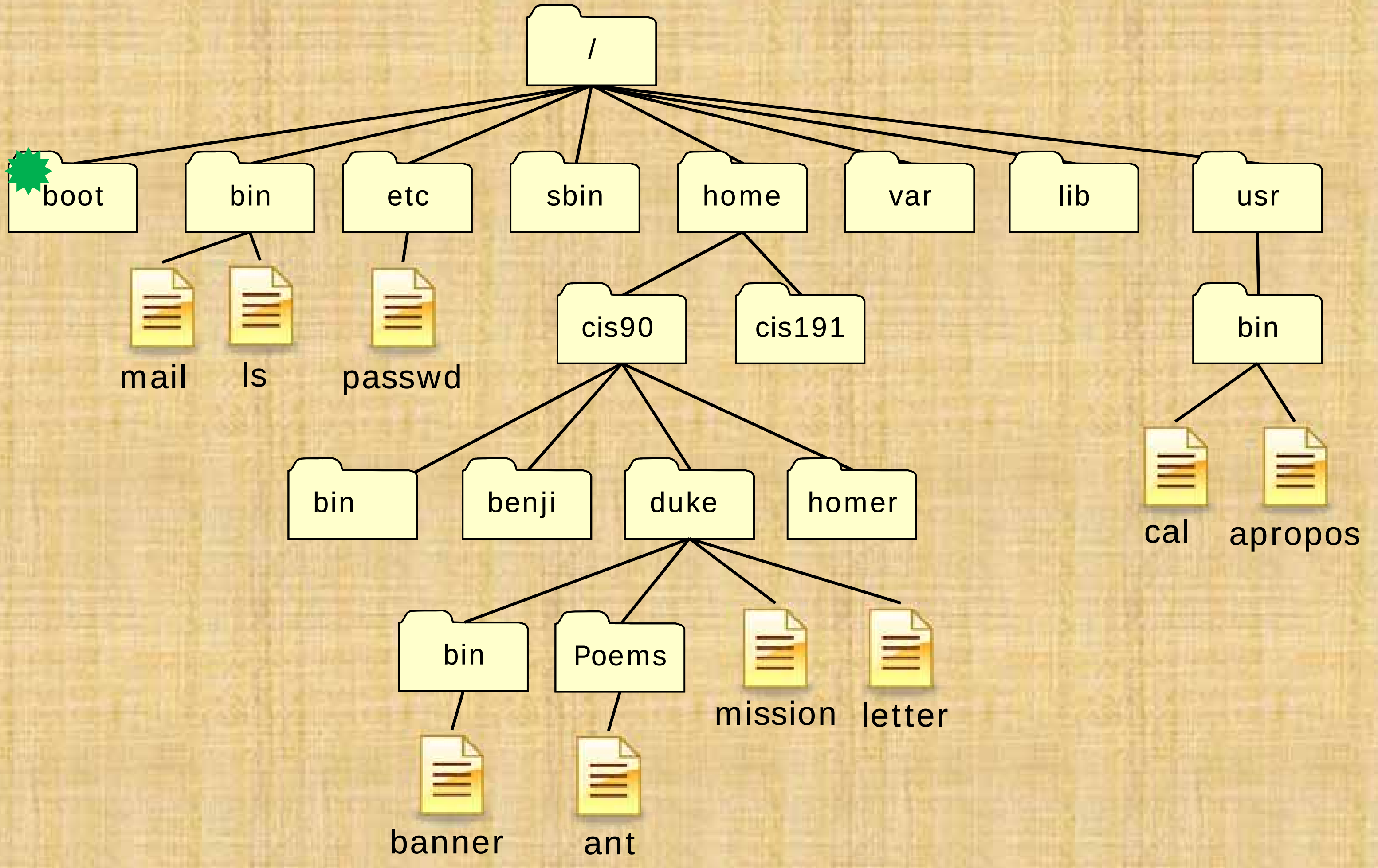
Pathname Practice

Current working directory shown by ✸



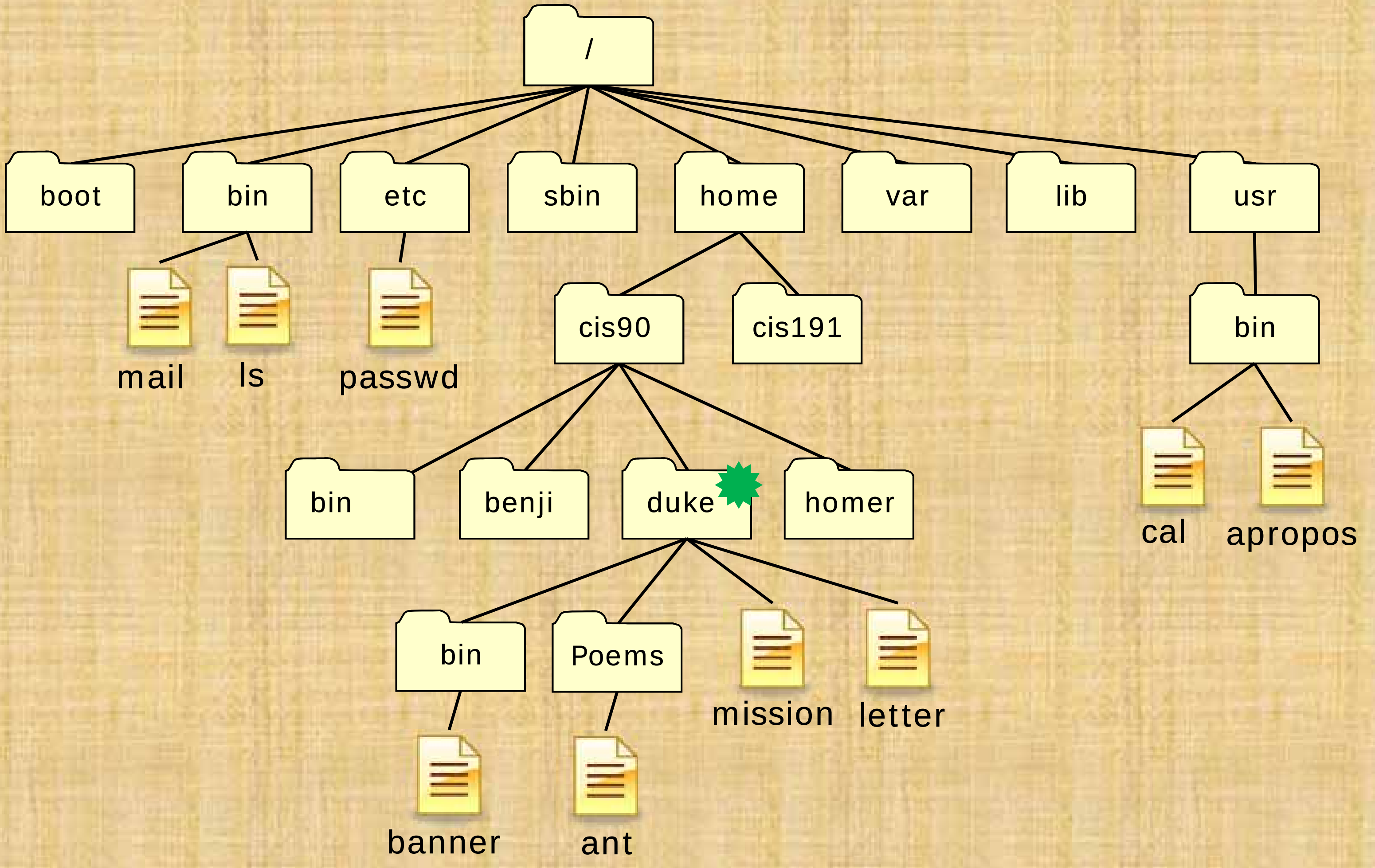
Pathname Practice

Current working directory shown by ✸



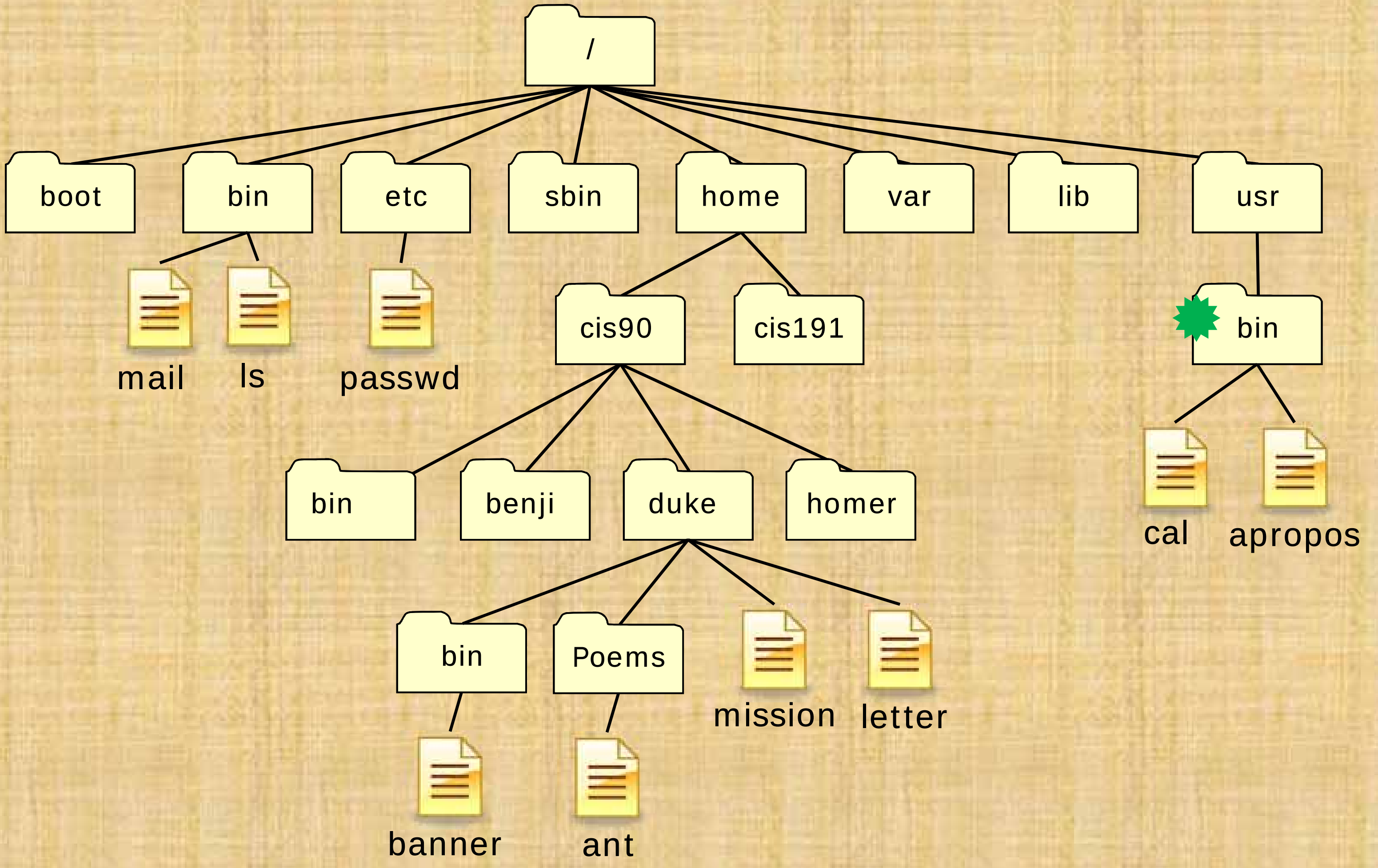
Pathname Practice

Current working directory shown by ✸



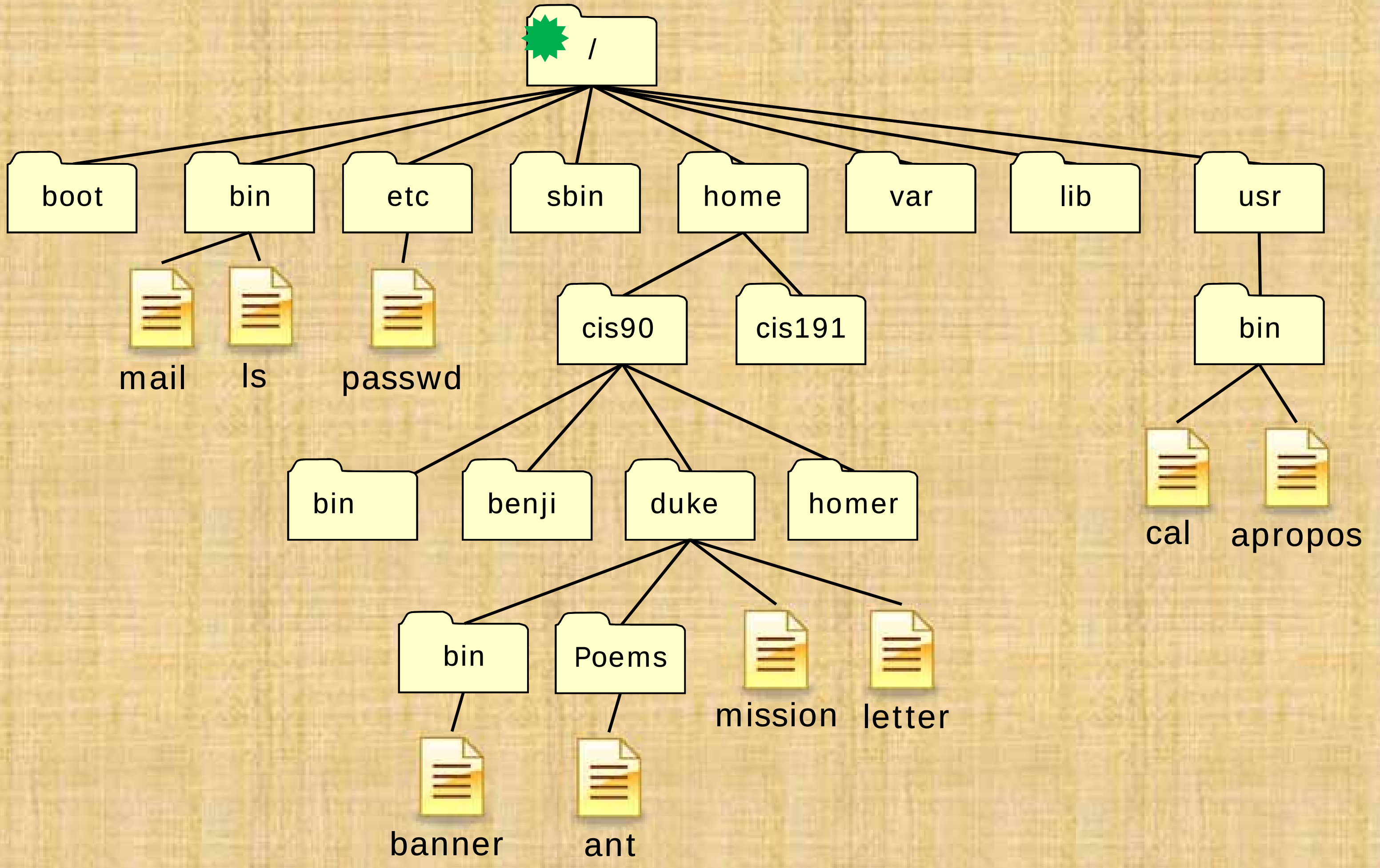
Pathname Practice

Current working directory shown by ✸



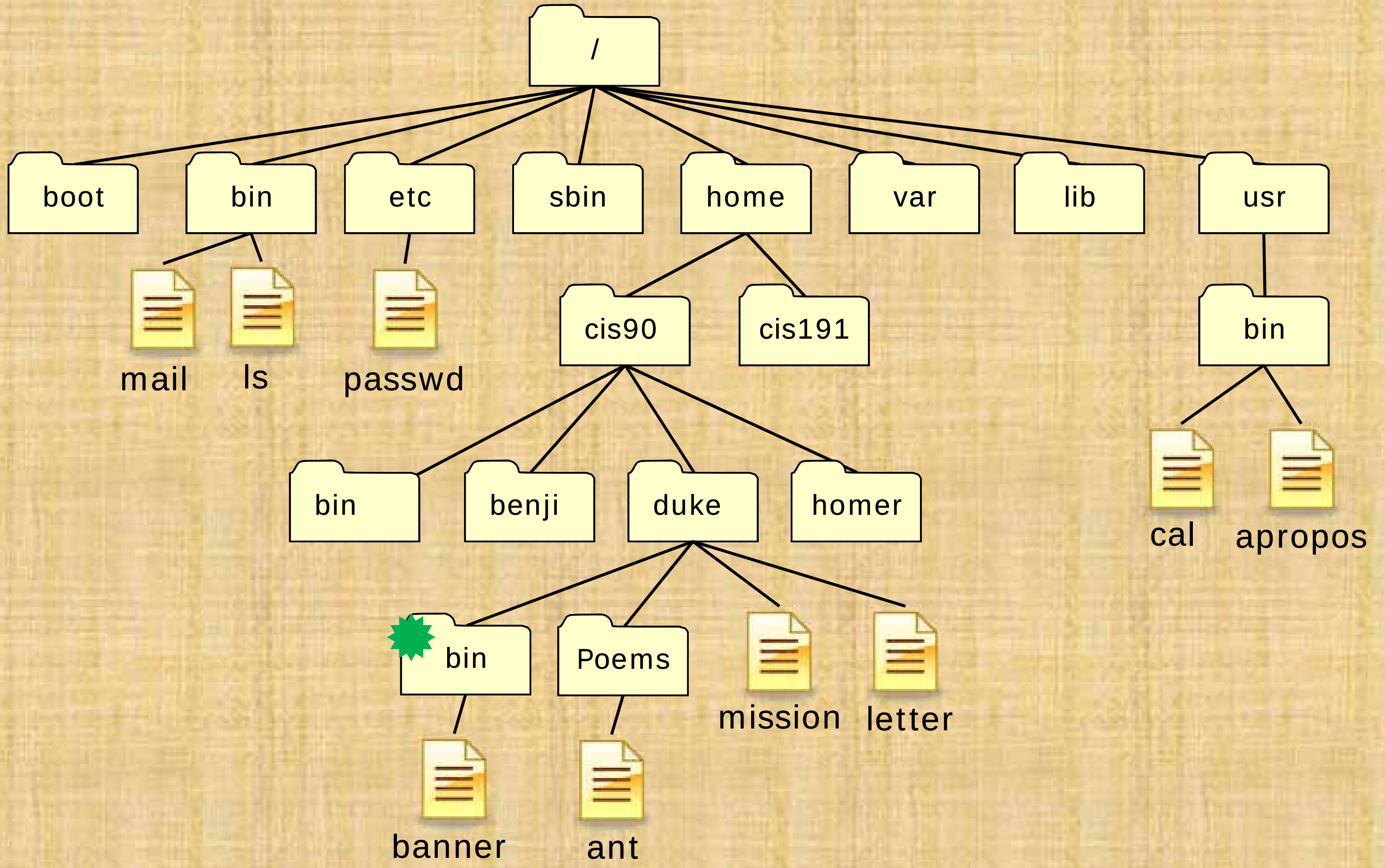
Pathname Practice

Current working directory shown by ✸



Pathname Practice

Current working directory shown by ✹



Managing Files

Managing the UNIX/Linux File System

Objectives

- Name the three elements of a Unix file, and where each is stored.

- Be able to distinguish between text, data, programs, and directory files.

- Know how the xxd command can be used to look at data files.

- Be able to manage the files in your home directory using:

  mkdir
  cp
  mv
  rmdir
  rm
  ln

Managing the UNIX/Linux File System

Filename Expansion Characters

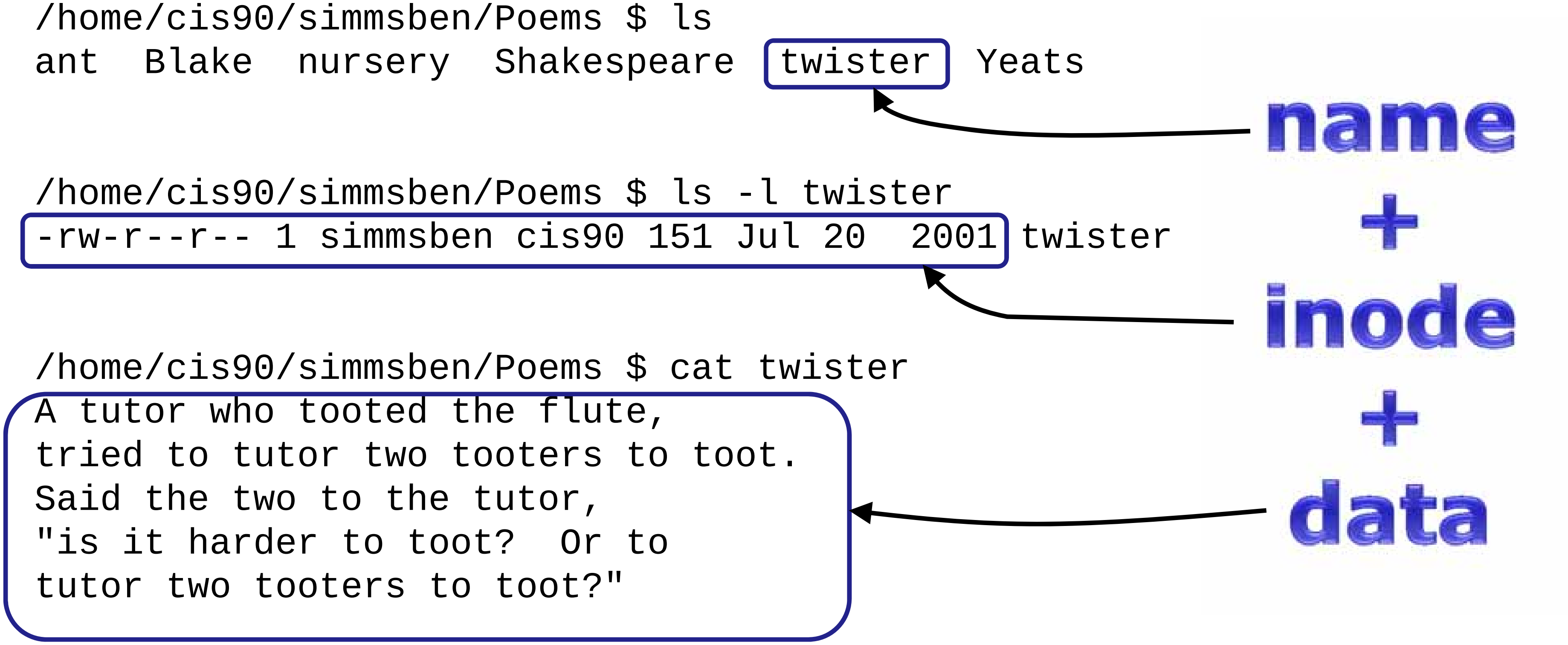
special characters that your shell recognizes to make it easier to specify file names. (wildcards)*

* matches all non-hidden filenames in the current directory when used alone, matches zero or more characters when used as a prefix, infix or postfix.

? matches any single character in any of your current directory's filenames.

[ ] matches any single character contained within the brackets

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?"
```

name

+

inode

+

data

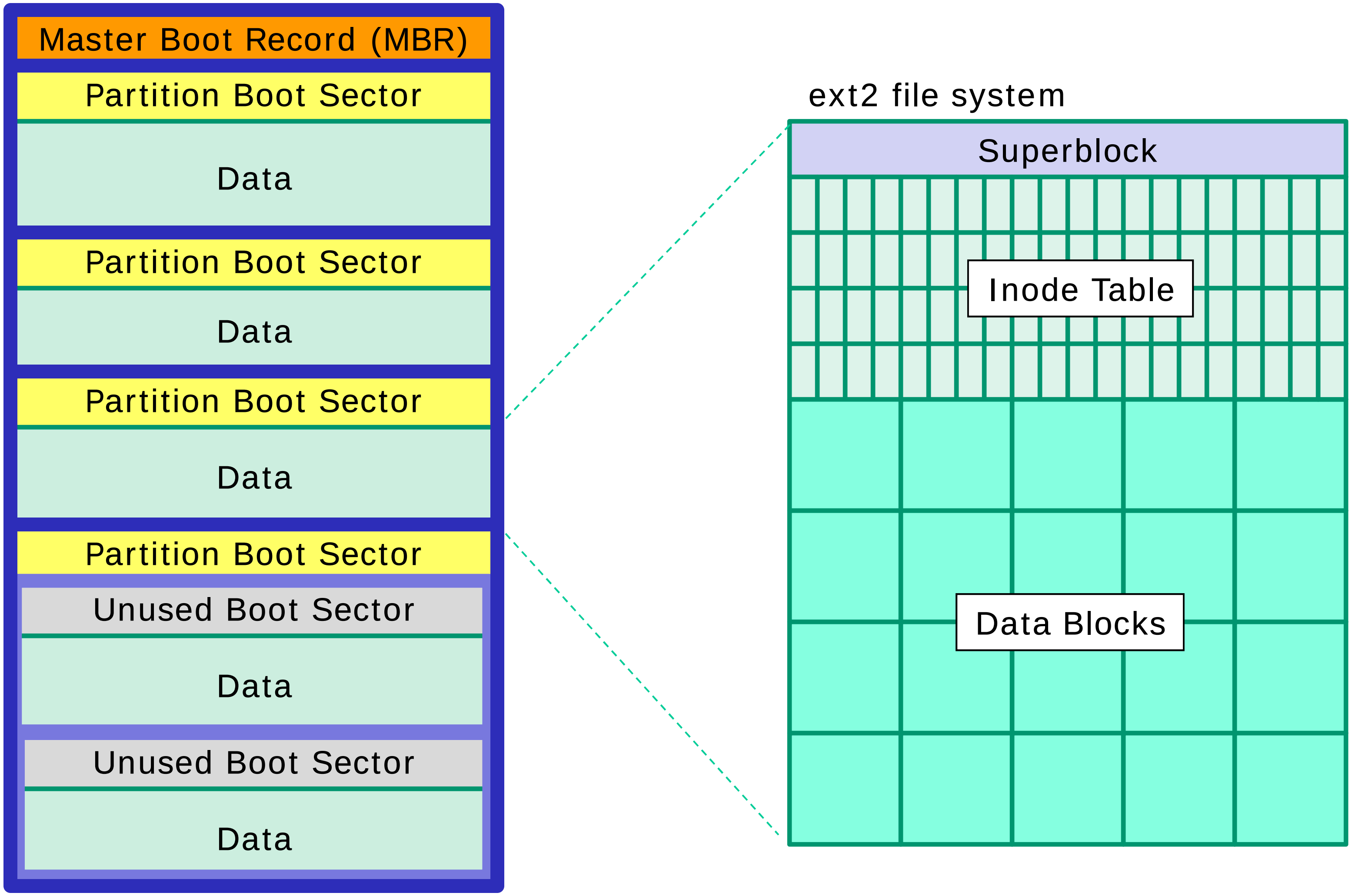
File Types and Commands

Long listing code (ls –l)	Type	How to make one
d	directory	mkdir
-	regular • Programs • Text • Data (binary)	touch
l	symbolic link	ln -s
c	special character device files	mknod
b	special block device files	mknod

Note: Other files types includes sockets (s) and named pipes (p)

File Systems

Linux



bigfile 102574
bin 102575
letter 102609

Hello Mother! Hello Father!

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

All the counselors hate the waiters, and the lake has alligators. You remember Leonard Skinner? He got ptomaine poisoning last night after dinner.

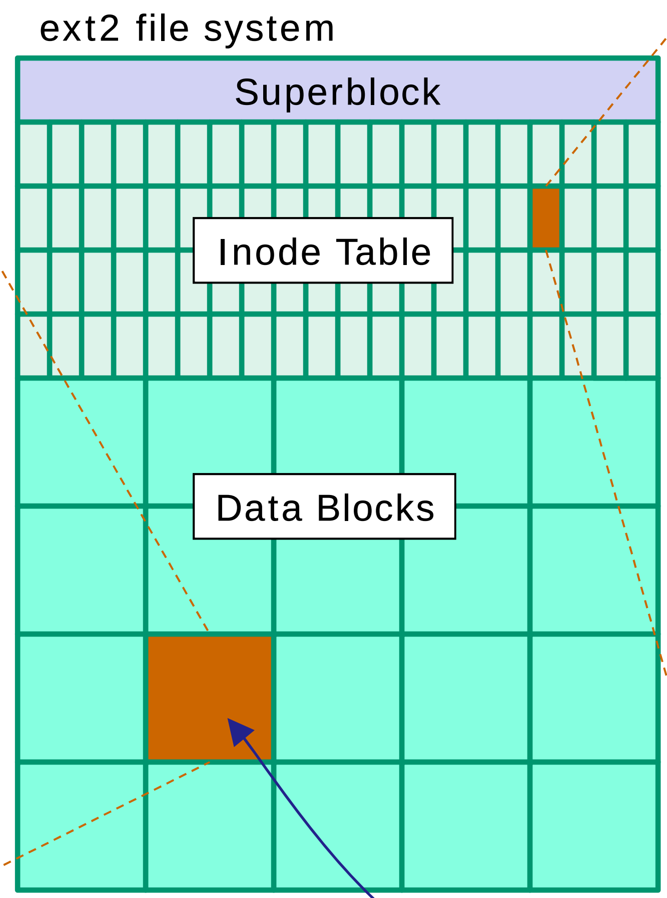
Now I don't want this to scare you, but my bunk mate has
malaria. You remember Jeffrey Hardy? Their about to organize a searching party.

Take me home, oh Mother, Father, take me home! I hate Granada.
Don't leave me out in the forest where I might get eaten
by a bear! Take me home, I promise that I won't make noise,
or mess the house with other boys, oh please don't make me
stay -- I've been here one whole day.

Dearest Father, darling Mother, how's my precious little brother? I will come home if you miss me. I will even let Aunt Bertha hug and kiss me!

Wait a minute! It's stopped hailing! Guys are swimming!
Guys are sailing! Playing baseball, gee that's better!
Mother, Father, kindly disregard this letter.

Alan Sherman



102609	inode number
-	Type
rw-r—r--	Permissions
1	Number of links
simmsben	User
cis90	Group
1044	Size
2001-07-20	Modification time
2008-08-08	Access Time
2008-06-20	Change time
Pointer(s) to data blocks	Pointer(s) to data blocks

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

inode

*Note, except for the filename, all other information shown on a **long listing** comes from the inode.*

Filenames are not kept in inodes, they are kept in ______________?

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

102609	inode number
-	Type
rw-r—r--	Permissions
1	Number of links
simmsben	User
cis90	Group
1044	Size
2001-07-20	Modification time
2008-08-08	Access Time
2008-06-20	Change time
Pointer(s) to data blocks	Pointer(s) to data blocks

Viewing files

ASCII (text), binary data

```
[roddyduk@opus ~]$ file /usr/bin/* | grep python | tail -5
/home/cis90ol/simmsben $ file /usr/bin/* | grep python | tail -5
/usr/bin/urlgrabber:                              python script text executable
/usr/bin/xml2po:                                  python script text executable
/usr/bin/xmlproc_parse:                           python script text executable
/usr/bin/xmlproc_val:                             python script text executable
/usr/bin/yum:                                     python script text executable
[roddyduk@opus ~]$
```

If you see the word text or ASCII as output from the file command it is safe to view with cat, head, tail, more or less

```
[roddyduk@opus ~]$ head /usr/bin/yum
#!/usr/bin/python
import sys
try:
    import yum
except ImportError:
    print >> sys.stderr, """\
There was a problem importing one of the Python modules
required to run yum. The error leading to this problem was:

   %s
[roddyduk@opus ~]$
```

Managing the UNIX/Linux File System

Creating

Commands:

`touch`

- creates an empty ordinary file(s), or if the file already exists, it updates the time stamp.

`mkdir`

- creates an empty directory(s)
- options: -p

`echo "string" > newfile`

- Creates or overwrites a text file

Managing the UNIX/Linux File System

Copying

Commands:

cp *<source file> <target file>*
or
cp <source file> <target directory>
or
cp <source file> <source file> <target directory>

options: -i –r

i = warns before overwriting
r = recursive (copies all sub folders)

Managing the UNIX/Linux File System

Moving

Commands:

```
mv  <source file> <target file>
   or
mv  <source file> <target directory>
   or
mv  <source file> <source file> <target directory>
```

options: `-i`

`i` = warns before overwriting

Managing the UNIX/Linux File System

Renaming

Commands:

```
mv  <original name> <new name>
```

Managing the UNIX/Linux File System

Removing

Commands:

```
rm <filename>...
    options:  -i -r –f
      i = prompt before overwrite
      r = recursive (delete subdirectories)
      f = force (never prompt)
```

```
rmdir <directory name>
    Directories must be empty for this to work
```

Managing the UNIX/Linux File System

Linking

Commands:

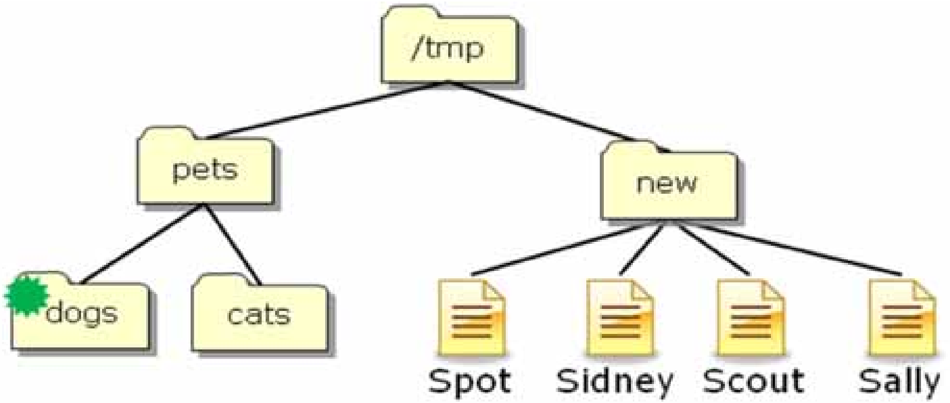
```
ln <existing-name> <new-name>
    options: -s

      s = symbolic link (like Windows shortcut)
```

Q19

From a previous Test #2

Test 2 Q19

19. Given this directory structure:



If your current working directory is *dogs*, what single command using filename expansion characters would move just the files *Scout* and *Sally* to the *dogs* directory?

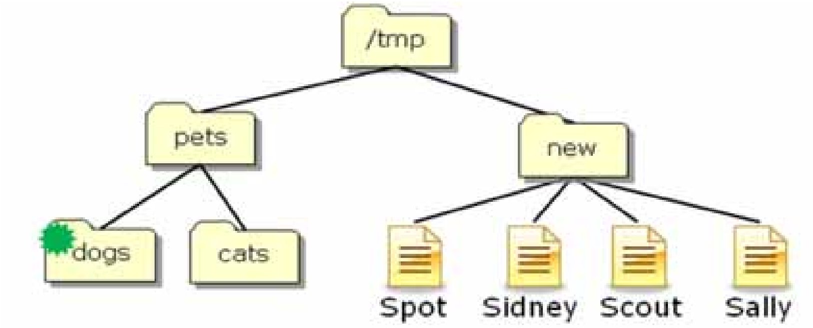
The shell replaces this with:
/tmp/new/Scout and /tmp/new/Sally

```
mv /tmp/new/S[ca]* .
```

here

Test 2 Q19 verification

```
/home/cis90/roddyduk $ cd /tmp
/tmp $ mkdir -p pets pets/dogs pets/cats new
/tmp $ cd new; touch Spot Sidney Scout Sally; cd ..
/tmp $ ls -R pets new
new:
Sally  Scout  Sidney  Spot

pets:
cats  dogs

pets/cats:

pets/dogs:
/tmp $ cd pets/dogs
/tmp/pets/dogs $ mv /tmp/new/S[ca]* .
/tmp/pets/dogs $ ls
Sally  Scout
/tmp/pets/dogs $
```



To verify your answer using Opus, create the same directory structure and test your command

```
# Turning on bash tracing
/tmp/pets/dogs $ set -x
++ echo -ne '\033]0;roddyduk@opus:/tmp/pets/dogs'

/tmp/pets/dogs $ mv /tmp/new/S[ca]* .
+ mv /tmp/new/Sally /tmp/new/Scout .
++ echo -ne '\033]0;roddyduk@opus:/tmp/pets/dogs'

/tmp/pets/dogs $
```

Permissions

Q18

From a previous Test #2

Test 2 Q18 answer

18. What permission is lacking that prevents you from viewing */boot/grub/grub.conf*?

r (read) permission for others

```
/home/cis90/roddyduk $ ls -l /boot/grub/grub.conf
-rw------- 1 root root 865 Jun 17 16:53 /boot/grub/grub.conf
/home/cis90/roddyduk $
```

Test 2 Q18 verification

18. What permission is lacking that prevents you from viewing */boot/grub/grub.conf*?

r (read) permission for others

```
/home/cis90/roddyduk $ cat /boot/grub/grub.conf
cat: /boot/grub/grub.conf: Permission denied
/home/cis90/roddyduk $ touch grub.conf
/home/cis90/roddyduk $ ls -l grub.conf /boot/grub/grub.conf
-rw------- 1 root     root  865 Jun 17 16:53 /boot/grub/grub.conf
-rwxrw-r-- 1 roddyduk cis90   0 Nov 10 07:54 grub.conf
/home/cis90/roddyduk $ chmod u-r grub.conf
/home/cis90/roddyduk $ cat grub.conf /boot/grub/grub.conf
cat: grub.conf: Permission denied
cat: /boot/grub/grub.conf: Permission denied
/home/cis90/roddyduk $ chmod u+r grub.conf
/home/cis90/roddyduk $ cat grub.conf /boot/grub/grub.conf
cat: /boot/grub/grub.conf: Permission denied
/home/cis90/roddyduk $
```

To check your answer using Opus, create your own grub.conf and verify by removing and adding r permission.

I/O

Parse

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:

~~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

Pipelines using the | operator

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.

Example

```
who | tee users | sort
```

Important! Redirection sends output to another file. Pipes send output to another process.

Input and Output

Pipelines

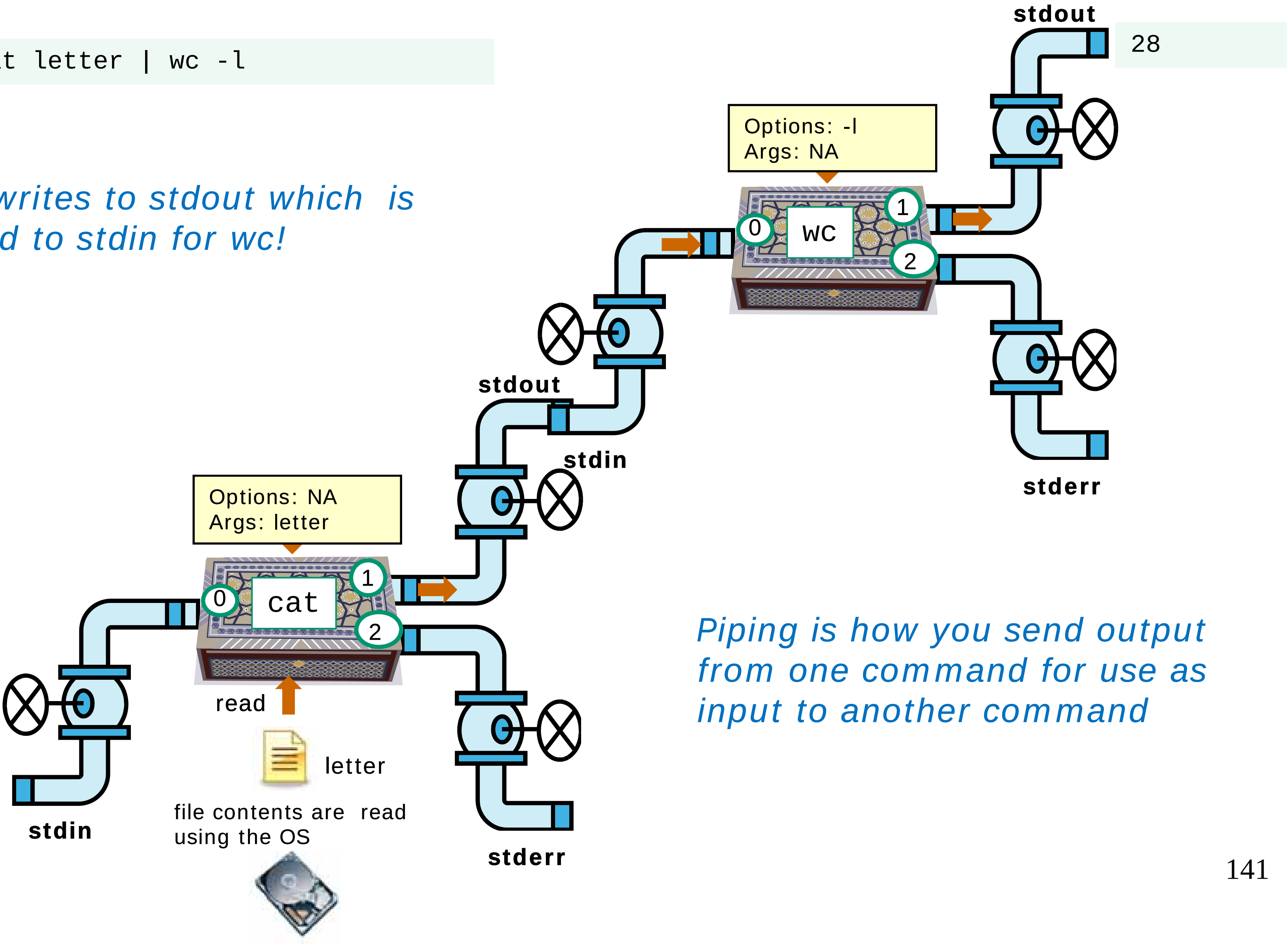
Let's count the lines in letter

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

Example program to process: cat and wc commands

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

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



```
28
```

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***

Q13

From a previous Test #2

Test 2 Q13

13. What complete command (with no “;”s) counts all the files belonging to you on the system, places a sorted list of them in the file *allmine*, and redirects error messages to the bit bucket?

Limits the files listed to just those owned by the user. The shell replaces $LOGNAME with the actual username.

The tee send the sorted files to both the file allmine and to the stdin of the wc command

```
find / -user $LOGNAME 2> /dev/null | sort | tee allmine | wc –l
```

find will list all files starting at / on the UNIX file tree

Permission errors are thrown away (from trying to list or traverse directories you don't have read and execute permission)

Use Opus to verify your answer

Q28

From a previous Test #2

Test 2 Q28

28. Given the file *problems* contains:

    2+2
    5/0

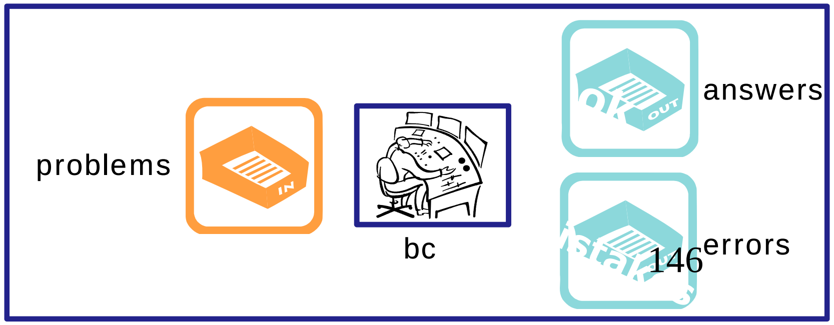
What complete command using bc would input the math problems in *problems*, **append** the calculated answers to the file *answers* and write any errors to the file *errors*?

stdin redirected from keyboard to file problems

stderr redirected from terminal to file errors

```
bc < problems >> answers 2> errors
```

stdout redirected from terminal to append to file answers



Test 2 Q28 verification

28. Given the file *problems* contains:

```
2+2
5/0
```

What complete command using bc would input the math problems in *problems*, append the calculated answers to the file *answers* and write any errors to the file *errors*?

```
/home/cis90/roddyduk $ echo 2+2 > problems
/home/cis90/roddyduk $ echo 5/0 >> problems
/home/cis90/roddyduk $ bc < problems >> answers 2> errors
/home/cis90/roddyduk $ cat answers errors
4
Runtime error (func=(main), adr=5): Divide by zero
/home/cis90/roddyduk $
```

To verify your answer on Opus, create the problems file to test your answer

Wrap up

Next Class

No Quiz

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

- Format:
  - 5 questions from flashcards lessons 6-8
  - 10 operational questions using Opus.
  - Open book, open notes, open computer
  - No help from others, you must answer all the questions by yourself.
- Recommended preparation:
  - Review Lessons 6-8 slides and Labs 5-7
  - Try doing some or all of Lab X2 (pathnames)
  - Scan previous Lessons so you know where to find things if needed
  - Take the practice test
  - Collaborate with others on the forum to compare answers!
  - Collaborate with others on the forum to compare answers!!
  - Collaborate with others on the forum to compare answers!!!

Jim's Summary Pages

Jim has some really good summary information on Lessons 6-8 on his web site:

Lesson 6 - Managing Files
http://cabrillo.edu/~jgriffin/CIS90/files/lecture5.html

Lesson 7 - File Permissions
http://cabrillo.edu/~jgriffin/CIS90/files/lecture6.html

Lesson 8 - Input/Output Processing
http://cabrillo.edu/~jgriffin/CIS90/files/lecture7.html

Backup

Given:

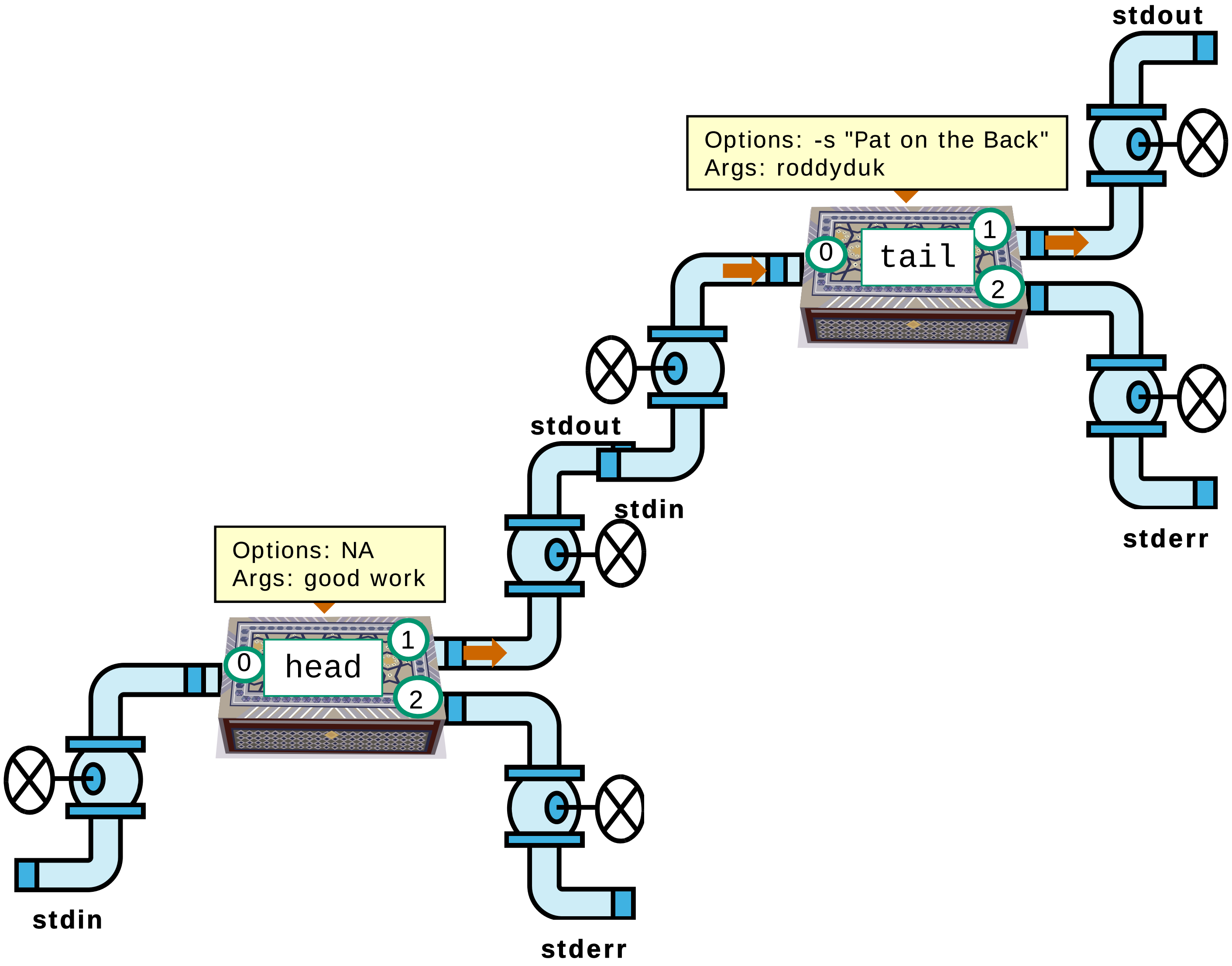
- PS1 is: `'\u likes $SHELL: '`
- path is: `/bin:/usr/bin:/home/cis90/bin:`
- command is:

```
banner Good Work | mail -s "Pat on the Back" $LOGNAME
```

1) Generate the prompt:

2) Parse the command line:
   - command(s) =
   - options =
   - arguments =
   - redirection =

3) Are the command(s) on the path?

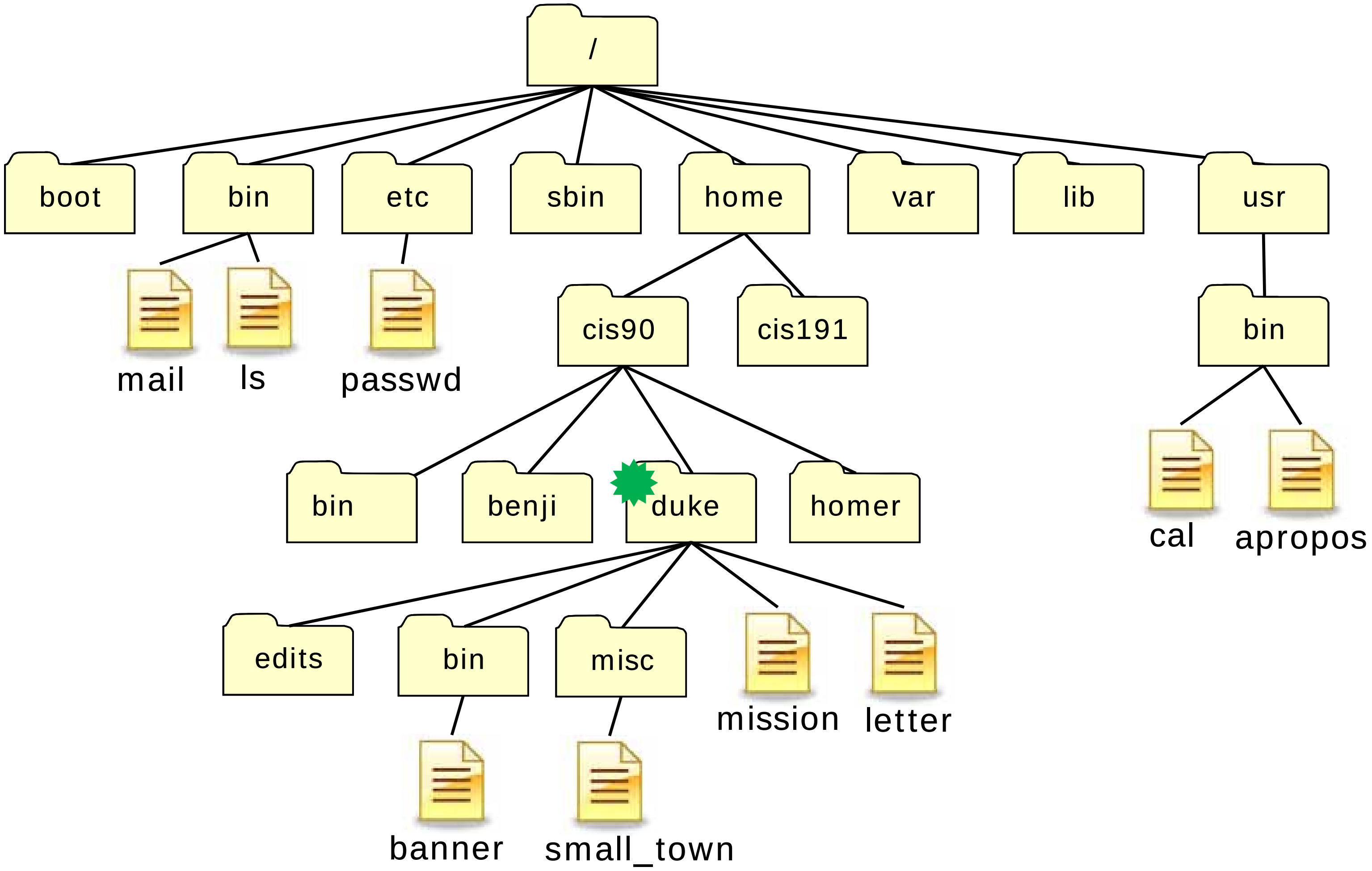


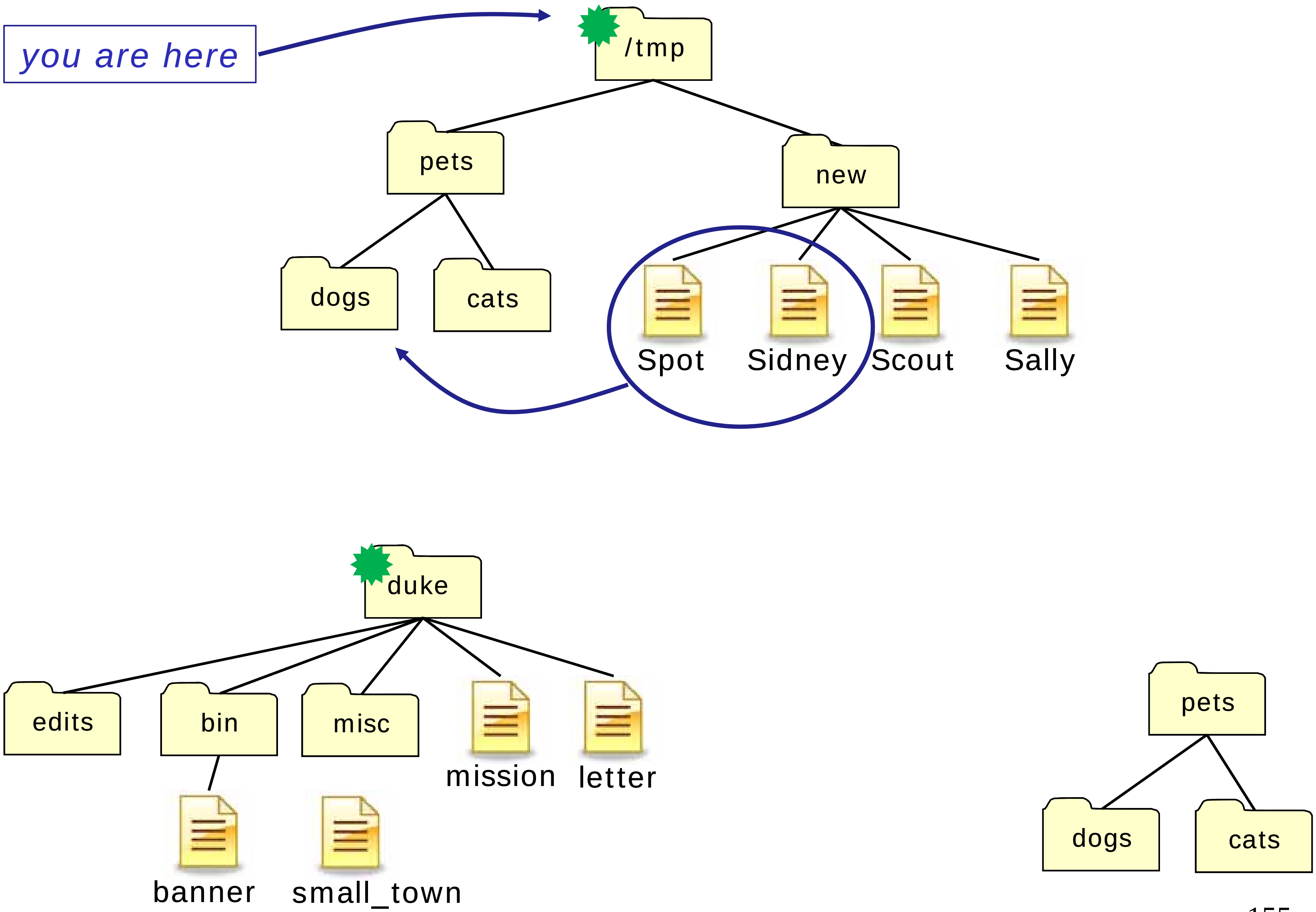
Options: NA
Args: good work

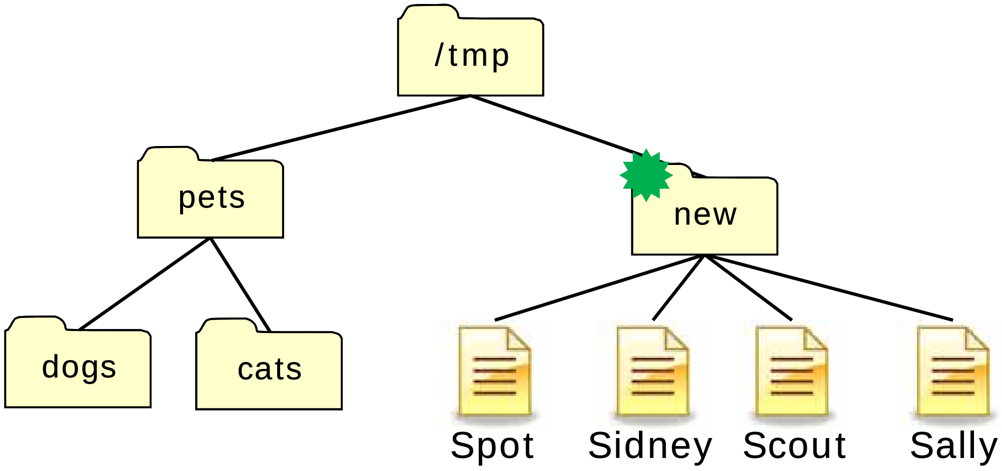
Options: -s "Pat on the Back"
Args: roddyduk

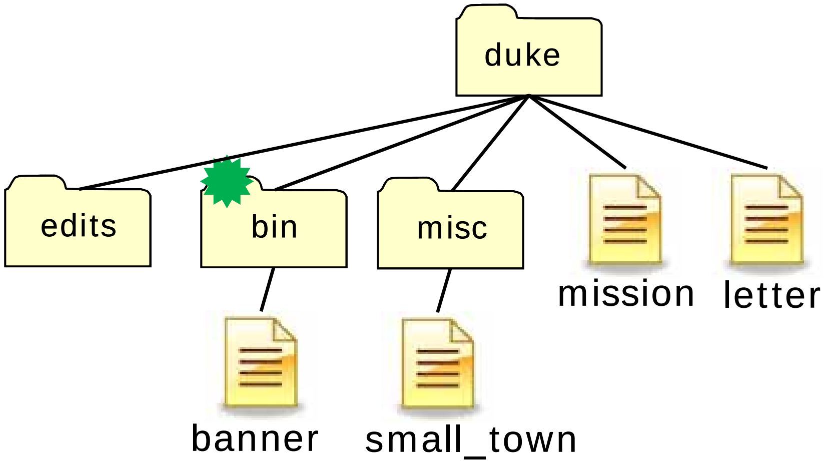
Pathnames

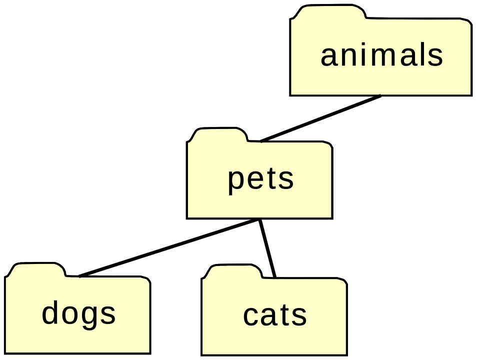
Current working directory shown by ✸

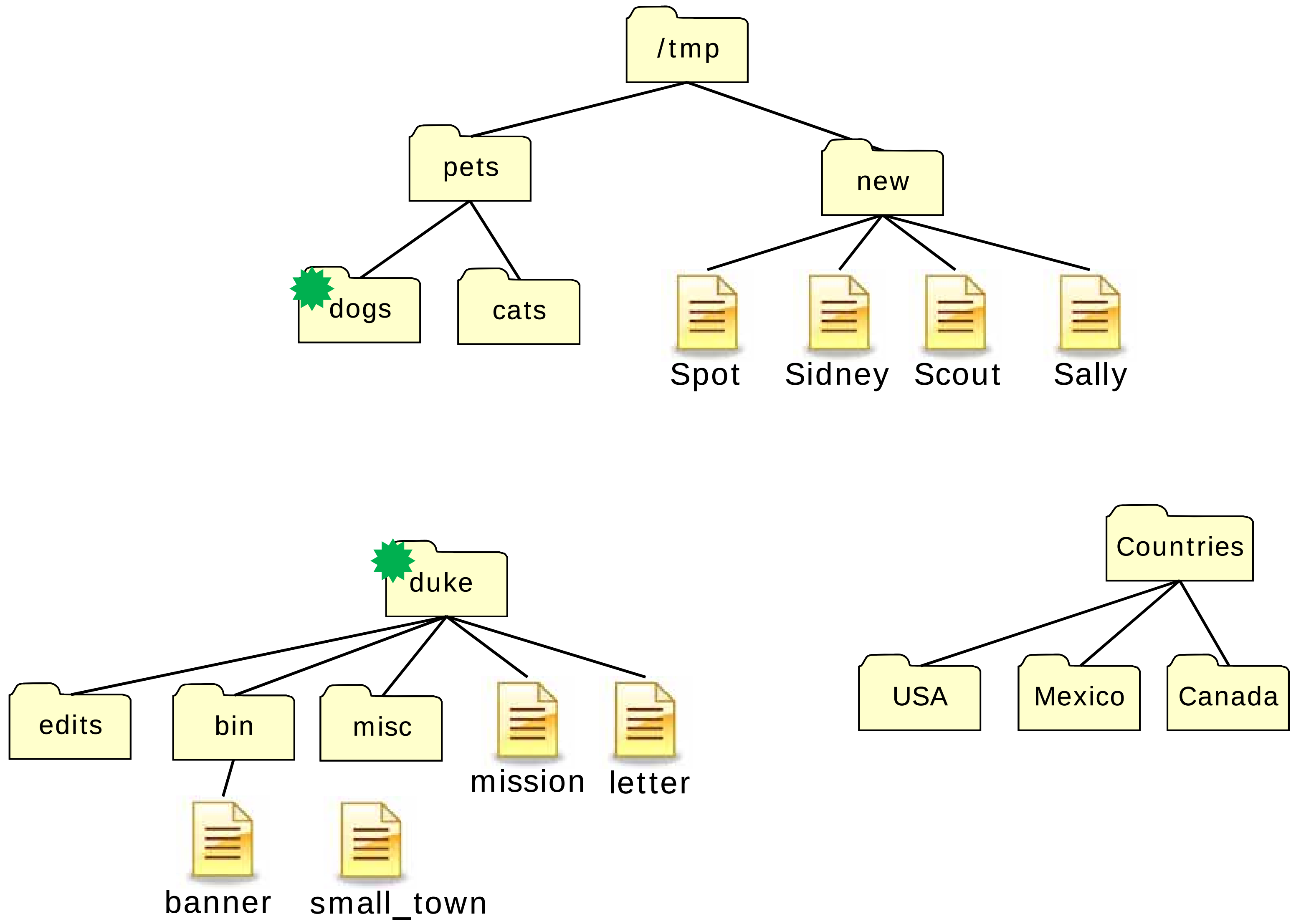



you are here
/tmp
pets
new
dogs
cats
Spot
Sidney
Scout
Sally
duke
edits
bin
misc
mission
letter
banner
small_town
pets
dogs
cats


/tmp
pets
new
dogs
cats
Spot
Sidney
Scout
Sally


duke
edits
bin
misc
mission
letter
banner
small_town


animals
pets
dogs
cats



/tmp

pets

new

dogs

cats

Spot

Sidney

Scout

Sally

duke

edits

bin

misc

mission

letter

banner

small_town

Countries

USA

Mexico

Canada

Life of the Shell

Practice being the Shell

Team 1

Given:

- PS1 is: `'\u in $PWD: '`
- path is: `/bin:/usr/bin:`
- command is: `cp -i /usr/sha*/gr?b/i386-*/stage[15] $LOGNAME`

1) Generate the prompt:

2) Parse the command line:
   - command =
   - options =
   - arguments =
   - redirection =

3) Is the command on the path?

Life of the Shell

Practice being the Shell

Team 2

Given:

- PS1 is: `'$LOGNAME in $PWD > '`
- path is: `/bin:/usr/bin:`
- command is: `iptables –l; head -21 [bB]igfi?? | sort > /dev/null`

1) Generate the prompt:

2) Parse the command line:
   - command =
   - options =
   - arguments =
   - redirection =

3) Are the command(s) on the path?

Life of the Shell

Practice being the Shell

Team 3

Given:

- PS1 is: `"prompt > "`
- path is: `/bin:/usr/bin:`
- command is: `> demo; head -10 l[ea]??er | tail -1 >> demo`

1) Generate the prompt:

2) Parse the command line:
   - command =
   - options =
   - arguments =
   - redirection =

3) Are the command(s) on the path?

Life of the Shell

Practice being the Shell

Team 4

Given:

- PS1 is: `'$SHELL<>$LOGNAME: '`
- path is: `/bin:/usr/bin:/sbin`
- command is: `modprobe; chmod g+w,g-w -c  po*/S*/s* 2> errors`

1) Generate the prompt:

2) Parse the command line:
    - command(s) =
    - options =
    - arguments =
    - redirection =

3) Are the command(s) on the path?

Life of the Shell

Practice being the Shell

Team 5

Given:

- PS1 is: `'\u likes $SHELL: '`
- path is: `/bin:/usr/bin:/sbin`
- command is:

```
find /etc -type d -name '*c[123456]*' 2> /dev/null | grep 2 >> list; cat list
```

1) Generate the prompt:

2) Parse the command line:
   - command(s) =
   - options =
   - arguments =
   - redirection =

3) Are the command(s) on the path?