Lesson Module Checklist

- Slides posted -
- Lab tested and posted –

- Flash cards -
- First minute quiz -
- Web calendar summary -
- Web book pages –
- Commands -

- CCC Confer room whiteboard posted -

- Bring class roster -
- Bring backup slides, CCC info, materials on flash drive -

- Headset is charged -

[ ] Has the phone bridge been added?

[ ] Is recording on?

[ ] Does the phone bridge have the mike?

[ ] Share slides, putties, Chrome and VLab

[ ] Disable spelling on PowerPoint

CIS 90 - Lesson 2

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

Dieskau	Jonathan							Ana	David
Obie	Dave	Cole	Corey	Nancy	Ryan	Elia	Tasha	Darren	Scott
Devin	Everett	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD
TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD

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:

1. What is the lowest level, inner-most component of a UNIX/Linux Operating System called?

2. What command shows the other users logged in to the computer?

3. What part of UNIX/Linux is both a user interface and a programming language?

email answers to: `risimms@cabrillo.edu`

Commands

Objectives	Agenda
• Understand how the UNIX login operation works. • Meet John the Ripper and learn how vulnerable a poor password is. • Understand basic command syntax and operation. • Understand program files and what happens when they are run. • Understand how the shell works and environment variables. • Understand how to get documentation when online.	• Quiz • Questions and Review • Deep dive on logging in • Passwords • Program files • Running programs/processes • Command line syntax • Environment variables • Life of the shell • Metacharacters • Docs • Wrap up

Questions?

Lab assignment?
Previous Material?

We used at least six computers for Lab 1 !!

(A PC/Mac, Opus, Kate, Not-Opus, Mt-Eko, vmserver4)

A

B

Opus (a RHEL server)

In building 1300

C

vmserver4 (a VMware ESXi server)

In building 1400

F

D

E

CIS-Lab-XX
(a PC in the CIS Lab)

A) Opus via SSH from MrEko
B) Opus via Putty from CIS-Lab-XX station
C) Mr-Eko via RDP and VMware vSphere
D) Not-Opus via RDP and VMware vSphere
E) Kate via RDP and VMware vSphere
F) CIS Lab station

Lab 1 Questions

Are there any questions on these questions?

- On xxxx, what is the prompt string?
- On xxxx, how many current login sessions are there?
- On xxxx, what is the hostname?
- On xxxx, what terminal device did you use?
- On xxxx, what OS kernel is being run?
- On xxxx, which distribution of Linux is being run?
- On xxxx, what is your username and uid (user ID) number?
- On xxxx, what shell is being used?
- On Opus, can you bring up a virtual terminal, like tty2?

Where xxxx = Opus. Not-Opus, Mr-Eko or Kate

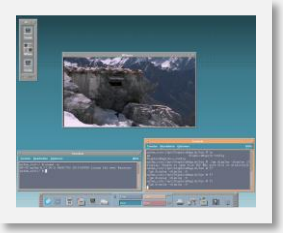
- On any system, does logging off one session log you off all other sessions?"
- On any system, does the history command remember commands for past login sessions?"
- Does the history command remember commands entered on another system?"
- On the same system, is your command history the same for each login session?"

You can resubmit Lab 1 as many times as you want till midnight

*Use **verify** or **cat lab01-submitted** to verify what you submitted*

Review and clarifications

UNIX and Unix-like Operating Systems

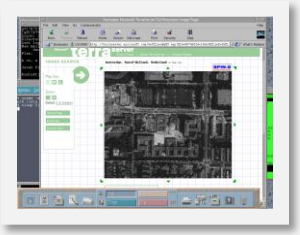


HP-UX

Sun Solaris

SCO UnixWare

SCO



IBM AIX

UNIX

AT&T UNIX
(1969)

BSD UNIX

BSD UNIX

Mac OS X and iOS

iOS 4

Apple operating systems use the Mach Kernel

All Linux distributions use the Linux Kernel

Various GNU/Linux Distributions

Ubuntu

Red Hat

SUSE

Debian

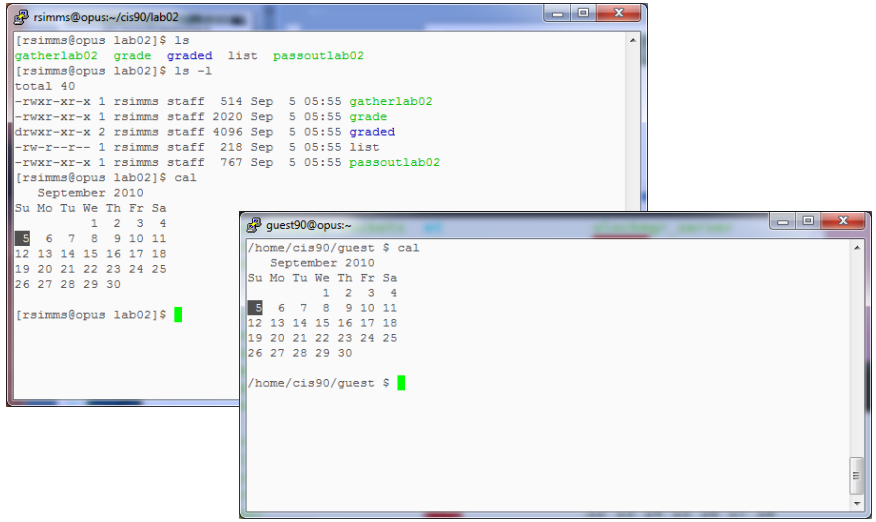
Embedded Linux

TiVo

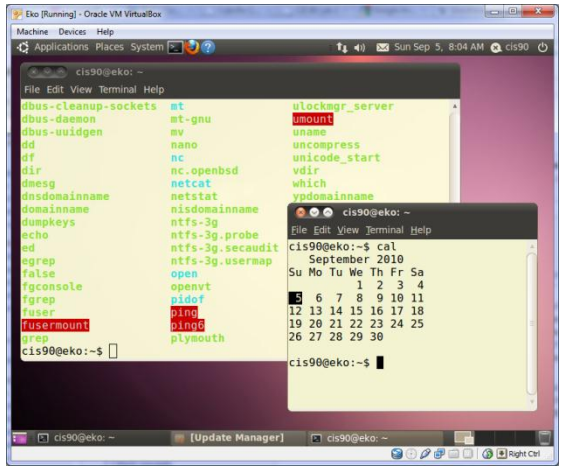
Android

routerboard

Terminals



Terminal emulators like PuTTY (with scroll bars, colors, customizable backgrounds, fonts and sizes) and runs on another computer

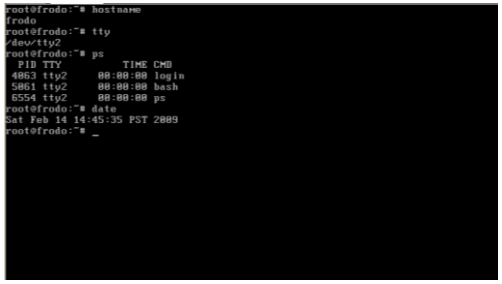


Graphical terminals (with scroll bars, colors, customizable backgrounds, fonts and sizes) available on the graphical desktop

tty = teletype

Terminals were used in the old days to interact with computers.

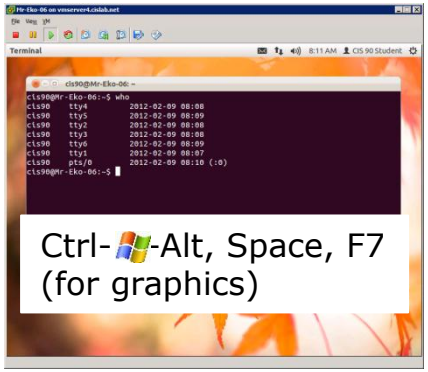
*Today we use **terminal emulators** that are software programs.*

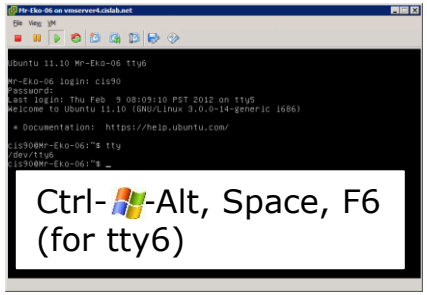


Virtual terminals (use ctrl-alt-fn) (no scroll bar, also called a console)

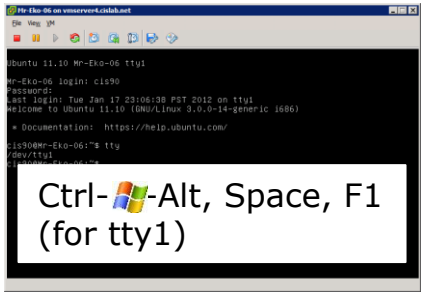
Changing Virtual Terminals using VMware vSphere

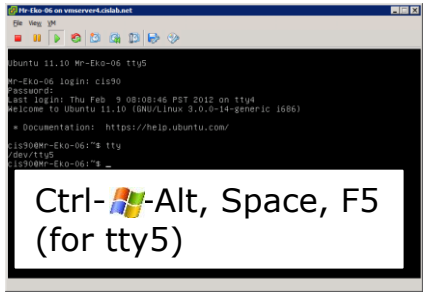
Windows PC Keyboard

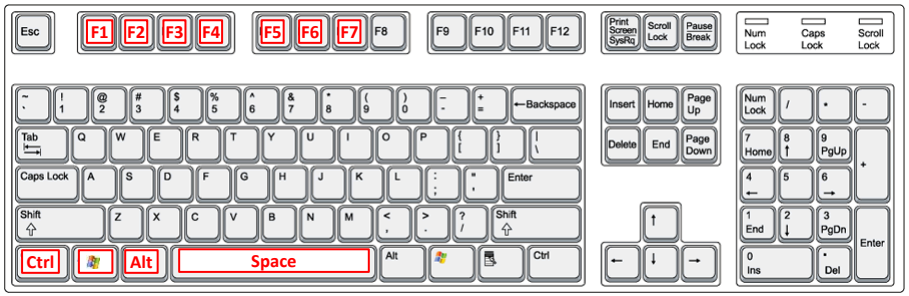


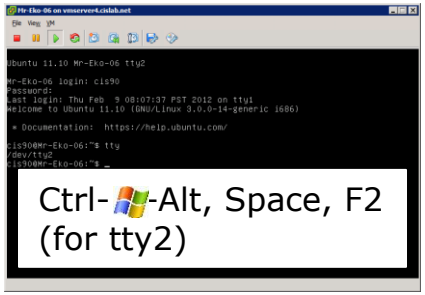


While holding down Crtl-⊞-Alt keys, tap Space, then tap Fn key*

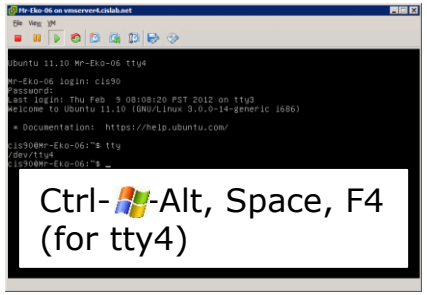


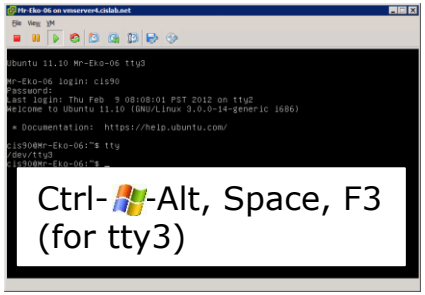






*On some PC keyboards it is not necessary to use the ⊞ key



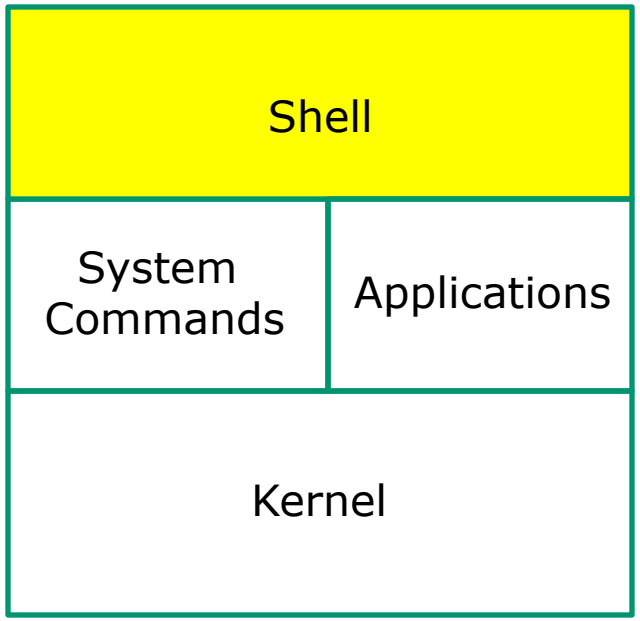


Shell tty command

Running three Putty sessions at the same time to Opus. Note that each session is assigned a different terminal device.

*Use the **tty** command to identify the **terminal device** being used for a session*

The Shell



- Allows users to interact with the computer via a "**command line**".
- **Prompts** for a command, parses the command, finds the right program and gets that program executed.
- Is called a "**shell**" because it hides the underlying operating system.

- Multiple shell programs are available: **sh** (Bourne shell), **bash** (born again shell), **csh** (C shell), **ksh** (Korn shell).
- The shell is a **user interface** and a **programming language** (scripts).
- GNOME and KDE desktops could be called **graphical shells**

In Lab 1 you got to interact with the bash shell on several Linux systems

Commands from last week's lesson and lab

cal	*Prints calendars*
clear	*Clears the screen*
date	*Shows the time and date*
exit	*Exits login session*
history	*Shows previous commands*
hostname	*Shows name of computer being interacted with*
id	*Shows UID's, GID's and SELinux information*
ps	*Shows process information*
ssh	*Initiates connection and login to remote computer*
uname	*Shows name of operating system kernel*
tty	*Shows terminal device being used for session*
who	*Shows all users who are logged in*
who am i	*Like **who**, but only shows your login session*

Note, each of these commands is actually a program residing in the /bin or /usr/bin directories.

Class Activity
Command Review

Login to Opus if you haven't already

Now follow along as we review the commands learned last week and new commands for this week

cal command

```
/home/cis90/simben $ cal
   February 2012
Su Mo Tu We Th Fr Sa
           1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29

/home/cis90/simben $ cal 9 2001
   September 2001
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30
/home/cis90/simben $
```

*The **cal** command outputs a calendar*

date command

```
/home/cis90/simben $ date
Mon Feb 13 09:29:00 PST 2012
/home/cis90/simben $
```

*The **date** command outputs the date and time*

clear command

simben90@opus:~

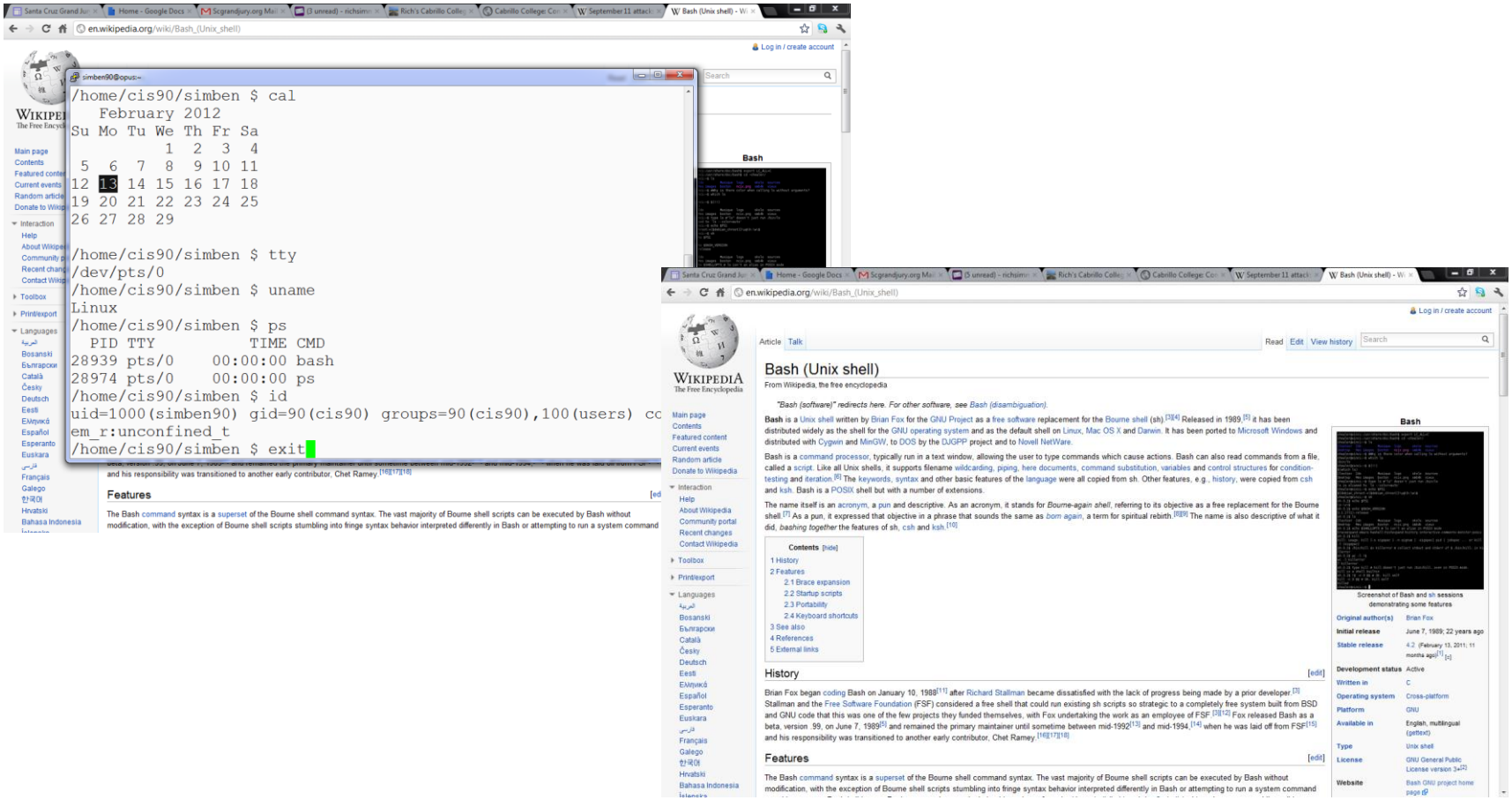
```
/home/cis90/simben $ date
Mon Feb 13 09:32:36 PST 2012
/home/cis90/simben $ cal
   February 2012
Su Mo Tu We Th Fr Sa
           1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29

/home/cis90/simben $ uname
Linux
/home/cis90/simben $ tty
/dev/pts/0
/home/cis90/simben $ hostname
opus.cabrillo.edu
/home/cis90/simben $ clear
```

simben90@opus:~

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

*The **clear** command scrolls previous commands out of sight*

exit command



*The **exit** command ends the session and the terminal window disappears ... POOF!*

history command

```
/home/cis90/simben $ history
    1  hostname
    2  exit
    3  who
    4  who -q
    5  ps -e
< snipped >
  177  cal 9 2001
  178  exit
  179  who
  180  cal
  181  tty
  182  uname
  183  ps
  184  id
  185  exit
  186  history
/home/cis90/simben $
```

*The **history** command outputs commands previously used*

Tip: Use the "Up Arrow" key to use a previous command again!

hostname command

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

*The **hostname** command outputs the name of the computer*

id command

```
/home/cis90/simben $ id
uid=1000(simben90) gid=90(cis90) groups=90(cis90),100(users)
context=user_u:system_r:unconfined_t
/home/cis90/simben $
```

*The **id** command outputs the uid, username, group membership, and SELinux context*

ps command

Process ID numbers

```
/home/cis90/simben $ ps
  PID TTY          TIME CMD
28994 pts/0    00:00:00 bash
29093 pts/0    00:00:00 ps
```

*shell is running and waiting for **ps** command to finish*

***ps** command is running as it outputs this*

Terminal device being used

*The **ps** command outputs the current processes owned by the user.*

ssh command

*The **ssh** command is used to log into another computer*

```
/home/cis90/simben $ ssh simben90@opus.cabrillo.edu
simben90@opus.cabrillo.edu's password:
Last login: Mon Feb 13 10:30:38 2012 from opus.cabrillo.edu

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

Terminal type? [xterm]
Terminal type is xterm.
/home/cis90/simben $ who
simben90 pts/0        2012-02-13 09:50 (dsl.dynamic.tooslow.com)
simben90 pts/5        2012-02-13 10:31 (opus.cabrillo.edu)
/home/cis90/simben $
```

*Note: You can also **ssh** into the same computer you are using already for an additional session*

uname command

```
/home/cis90/simben $ uname
Linux
```

*The **uname** command outputs the name of the operating system kernel*

tty command

```
/home/cis90/simben $ tty
/dev/pts/5
/home/cis90/simben $
```

*The **tty** command outputs the name of the terminal device being used*

who command

```
/home/cis90/simben $ who
simben90 pts/0        2012-02-13 09:50 (dsl.dynamic.tooslow.com)
simben90 pts/5        2012-02-13 10:31 (opus.cabrillo.edu)
```

username

terminal device
(pts/5 = /dev/pts/5)

*The **who** command outputs the other user sessions currently logged into the system*

who am i command

```
/home/cis90/simben $ who
simben90 pts/0        2012-02-13 09:50 (dsl.dynamic.tooslow.com)
simben90 pts/5        2012-02-13 10:31 (opus.cabrillo.edu)
```

```
/home/cis90/simben $ who am i
simben90 pts/5        2012-02-13 10:31 (opus.cabrillo.edu)
/home/cis90/simben $
```

*The **who am i** shows which of the user sessions is your session*

Test your knowledge

What's in a name?

What's the name of the terminal device I'm using right now?

```
login as: simben90
simben90@opus.cabrillo.edu's password:
Last login: Sun Feb 12 19:34:56 2012 from 10.64.25.2

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

Terminal type? [xterm]
Terminal type is xterm.
/home/cis90/simben $
/home/cis90/simben $ tty
/dev/pts/0
/home/cis90/simben $
```

*Use the **tty** command to find out*

Answer: /dev/pts/0

What's in a name?

What type of terminal am I using right now?

```
login as: simben90
simben90@opus.cabrillo.edu's password:
Last login: Sun Feb 12 19:34:56 2012 from 10.64.25.2

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

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

Answer: xterm

We have the answer already!
Note, if xterm had scrolled off the screen, there is a command that shows this information (coming soon)

What's in a name?

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

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

*Use the **hostname** command to find out*

Answer: opus.cabrillo.edu

What's in a name?

What is the name of the OS (operating System) kernel?

```
/home/cis90/simben $
/home/cis90/simben $ uname
Linux
/home/cis90/simben $
```

*Use the **uname** command to find out*

Answer: Linux

What's in a name?

What is the name of the Linux Distribution being run?

```
/home/cis90/simben $
/home/cis90/simben $ cat /etc/*-release
Red Hat Enterprise Linux Server release 5.4 (Tikanga)
/home/cis90/simben $
```

Answer: Red Hat Enterprise Linux

*Use the **cat /etc/*-release***

*Or **cat /etc/issue** command to find out*

What's in a name?

What is my username and uid (user ID number)?

```
/home/cis90/simben $
/home/cis90/simben $ id
uid=1000(simben90) gid=90(cis90) groups=90(cis90),100(users)
 context=user_u:system_r:unconfined_t
/home/cis90/simben $
```

Answer: simben90 (uid=1000)

*Use the **id** command to find out*

What's in a name?

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

```
/home/cis90/simben $
/home/cis90/simben $ ps
  PID TTY          TIME CMD
28237 pts/0    00:00:00 bash
28752 pts/0    00:00:00 ps
/home/cis90/simben $
```

*Use the **ps** command to find out.*

We will soon learn another command for doing this.

Answer: bash

Putty Tips

(Note: tty = teletype)

The Putty program

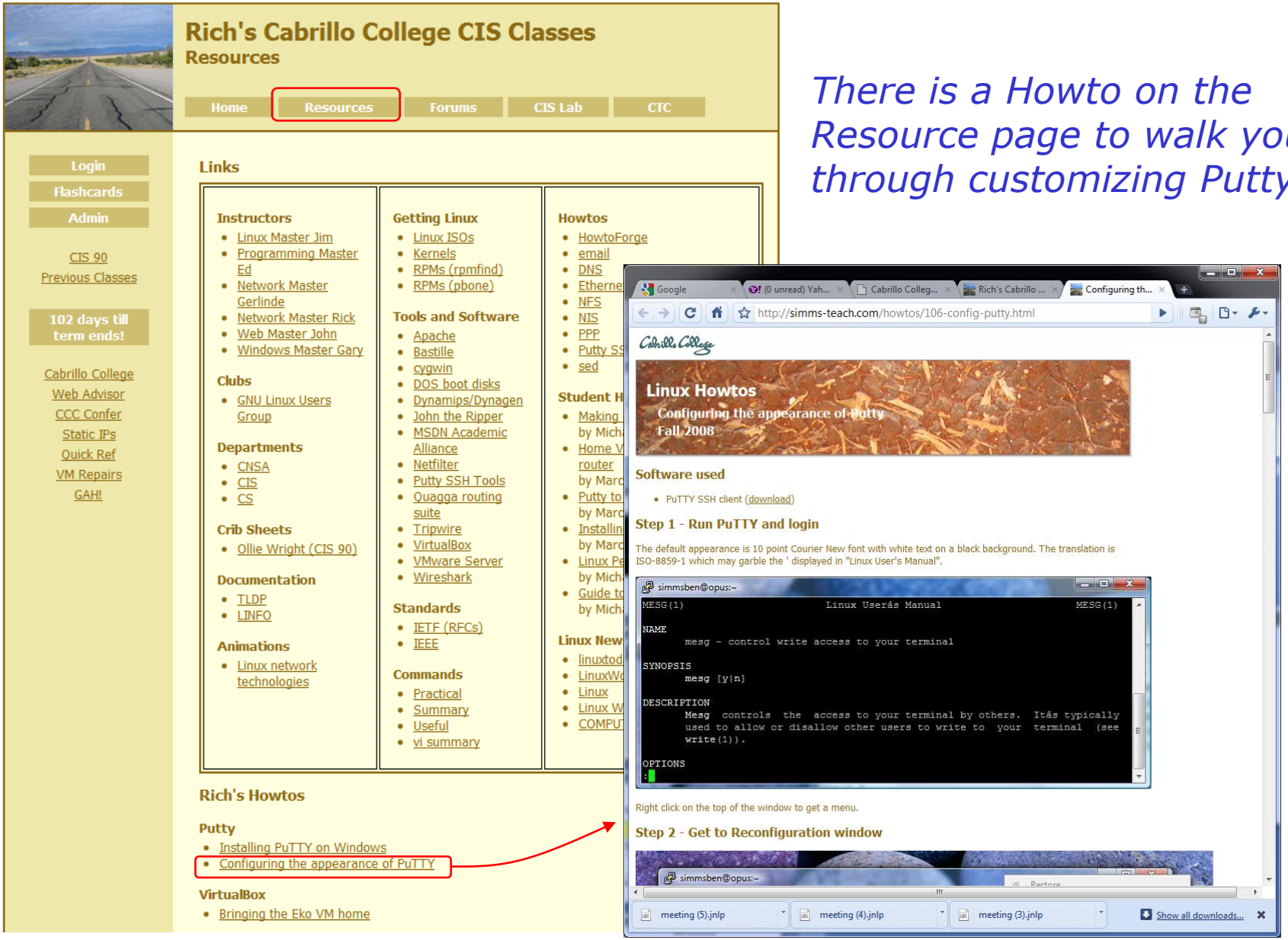
rsimms@server0-01:~

```
[rsimms@server0-01 rsimms]$ ls /bin
arch        cut            fgrep      ls            pwd        sync
ash         date           gawk       mail          r
ash.static  dd             grep       mkdir         r
awk         df             gtar       mknod         r
basename    dmesg          gunzip     mktemp        r
bash        dnsdomainname  gzip       more          r
bash2       doexec         hostname   mount         r
bsh         domainname     igawk      mt            s
cat         dumpkeys       ipcalc     mv            s
chgrp       echo           kbd_mode   netstat       s
chmod       ed             kill       nice          s
chown       egrep          link       nisdomainname s
cp          env            ln         pgawk         s
cpio        ex             loadkeys   ping          s
csh         false          login      ps            s
[rsimms@server0-01 rsimms]$
```

rsimms@nosmo:~/depot/gcal-3.01/src

```
[rsimms@nosmo src]$ ls /bin
alsaunmute  dnsdomainname  kbd_mode   nisdomainname  sync
arch        doexec         keyctl     pgawk          tar
ash         domainname     kill       ping           tcsh
ash.static  dumpkeys       ksh        ping6          touch
awk         echo           link       ps             tracepath
basename    ed             ln         pwd            tracepath6
bash        egrep          loadkeys   red            traceroute
bsh         env            login      rm             traceroute6
cat         ex             ls         rmdir          true
chgrp       false          mail       rpm            umount
chmod       fgrep          mailx      rvi            uname
chown       gawk           mkdir      rview          unicode_start
cp          gettext        mknod      sed            unicode_stop
cpio        grep           mktemp     setfont        unlink
csh         gtar           more       setserial      usleep
cut         gunzip         mount      sh             vi
date        gzip           mt         sleep          view
dd          hostname       mv         sort           ypdomainname
df          igawk          netstat    stty           zcat
dmesg       ipcalc         nice       su
[rsimms@nosmo src]$
```

*Why does Putty sometimes have a **black background** and sometimes a **white background**?*



There is a Howto on the Resource page to walk you through customizing Putty

Logging In (A deep dive)

Logging in

Note: the password is never echoed for security reasons

```
simmsben@opus:~
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

Note: Terminal Type ≠ Terminal Device

Login and Passwords

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

```
CentOS release 4.6 (Final)
Kernel 2.6.9-67.ELsmp on an i686

nosmo login: _
```

```
[root@nosmo ~]# ps t tty1
  PID TTY      STAT   TIME COMMAND
 3545 tty1     Ss+    0:00 /sbin/mingetty tty1
```

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

```
CentOS release 4.6 (Final)
Kernel 2.6.9-67.ELsmp on an i686

nosmo login: rsimms
Password: _
```

```
[root@nosmo ~]# ps t tty1
  PID TTY      STAT   TIME COMMAND
 3545 tty1     Ss+    0:00 /bin/login -
```

3) *If a match then* ***login*** *then starts up the shell specified in the /etc/passwd file*

```
CentOS release 4.6 (Final)
Kernel 2.6.9-67.ELsmp on an i686

nosmo login: rsimms
Password:
Last login: Mon Jul  7 14:25:17 on tty1
[rsimms@nosmo ~]$ _
```

```
[root@nosmo ~]# ps t tty1
  PID TTY      STAT   TIME COMMAND
 4917 tty1     Ss+    0:00 -bash
```

/etc/passwd

This command, which we will learn how to do later, outputs just one line of the /etc/passwd file

```
/home/cis90/simben $ cat /etc/passwd | grep simben
simben90:x:1000:90:Benji Simms:/home/cis90/simben:/bin/bash
```

username

password (just a placeholder now)

User ID (UID)

Group ID (gid)

Comment

Home directory

Shell

Note the field separator used in /etc/passwd is a ":"

/etc/passwd has all the login accounts. Passwords are no longer kept here. Instead the passwords are encrypted and placed in /etc/shadow.

```
/home/cis90/simben $ id
uid=1000(simben90) gid=90(cis90) groups=90(cis90),100(users)
context=user_u:system_r:unconfined_t
/home/cis90/simben $
```

Can you tell where the id command gets (some of) the data that it displays?

Turn OFF the recording

/etc/shadow

This command, which we will learn how to do later, outputs just one line of the /etc/shadow file

```
[root@opus ~]# cat /etc/shadow | grep simben
simben90:$6$f/nw[redacted]ImcS
o7[redacted]AEKf7n0:15382:0:99999:7:::
```

partially redacted

partially redacted

The passwords are now kept in /etc/shadow and they are encrypted

I've redacted (covered up) the encrypted password field to prevent ne-er-do-wells from using password cracking tools like John the Ripper on the password.

Turn ON the recording

Class Activity

Look at /etc/passwd and /etc/shadow files

1. **Login to Opus (if you haven't already)**

2. **cat /etc/passwd**
   - Find your username
   - Compare your home directory with your prompt
   - compare your uid and gid with output from the **id** command

3. **cat /etc/shadow**

What happens when you try to look at /etc/shadow?

Your Opus Password

Your Opus password

- Strong passwords are critical!
- **Botnets** and **ne-er-do-wells** are constantly attempting to break into computers attached to the Internet! (Even my little Frodo VM at home)

They never stop trying

The ne'er-do-wells trying to break in …
this is why you need strong passwords

```
--------------------- SSHD Begin ------------------------

SSHD Killed: 1 Time(s)

SSHD Started: 1 Time(s)

Disconnecting after too many authentication failures for user:
   guest90 : 1 Time(s)


Failed logins from:
    76.254.22.196 (adsl-76-254-22-196.dsl.pltn13.sbcglobal.net): 2 times
    201.7.115.194 (201-7-115-194.spopa302.ipd.brasiltelecom.net.br): 2135 times
    210.240.12.14: 20 times

Illegal users from:
    201.7.115.194 (201-7-115-194.spopa302.ipd.brasiltelecom.net.br): 564 times
    210.240.12.14: 42 times


Users logging in through sshd:
   guest:
      76.254.22.196 (adsl-76-254-22-196.dsl.pltn13.sbcglobal.net): 2 times
   jimg:
      70.132.20.25 (adsl-70-132-20-25.dsl.snfc21.sbcglobal.net): 7 times
   ordazedw:
      76.254.22.196 (adsl-76-254-22-196.dsl.pltn13.sbcglobal.net): 1 time
   root:
      63.249.86.11 (dsl-63-249-86-11.cruzio.com): 3 times
      70.132.20.25 (adsl-70-132-20-25.dsl.snfc21.sbcglobal.net): 1 time
   rsimms:
      63.249.86.11 (dsl-63-249-86-11.cruzio.com): 2 times
```

From a logwatch report showing malicious attempts to break into Opus

They never stop trying

The firewall on Opus slows down but does not end the attacks

```
Failed logins from:
    122.249.183.95 (x183095.ppp.asahi-net.or.jp): 3 times
    218.64.5.131 (131.5.64.218.broad.nc.jx.dynamic.163data.com.cn): 3
times

 Illegal users from:
    78.46.83.76 (static.76.83.46.78.clients.your-server.de): 3 times
    218.4.157.178: 3 times

 pam_succeed_if(sshd:auth): error retrieving information about user
teamspeak : 1 time(s)
 reverse mapping checking getaddrinfo for
131.5.64.218.broad.nc.jx.dynamic.163data.com.cn failed - POSSIBLE
BREAK-IN ATTEMPT! : 3 time(s)
 pam_succeed_if(sshd:auth): error retrieving information about user ts
: 2 time(s)
 pam_succeed_if(sshd:auth): error retrieving information about user
plcmspip : 2 time(s)
 pam_succeed_if(sshd:auth): error retrieving information about user
PlcmSpIp : 1 time(s)
```

We used to get up thousands of attempts every day until we made some changes to the firewall on Opus. Attacks always would come from different computers around the world.

/var/log/wtmp and var/log/btmp

```
[root@opus log]# lastb | sort | cut -f1 -d' ' | grep -v ^$ | uniq -c > bad
[root@opus log]# sort -g bad > bad.sort
[root@opus log]# cat  bad.sort | tail -50
    471 ftp
    472 public
    490 test
    490 tomcat
    498 user
    506 service
    508 mike
    508 username
    524 cyrus
    530 pgsql
    532 test1
    544 master
    554 linux
    554 toor
    576 paul
    584 support
    590 testuser
    604 irc
    610 test
    656 noc
    686 www
    690 postfix
    723 john
    734 testing
    738 adam
    746 alex
    754 info
    798 tester
    832 library
    935 guest
    990 admin
   1002 office
   1022 temp
   1070 ftpuser
   1138 webadmin
   1298 nagios
   1332 web
   1374 a
   1384 student
   1416 postgres
   1690 user
   1858 oracle
   1944 mysql
   2086 webmaste
   5324 test
  10803 root
  10824 admin
  18679 root
  24064 root
[root@opus log]#
```

Top 50 usernames used by the ne'er-do-wells

How to make a strong password

- The longer the better (8 or more characters)
- Not in any dictionary
- Use upper case, lowercase, punctuation, digits
- Something you can remember
- Keep it secret
- Change when compromised

`Wh0le#!!`	(Whole sh'bang)
`KuKu4(co)2`	(Cuckoo for Cocoa Puffs)
`#0p&s@ve`	(shop and save)
`Idl02$da`	(I do laundry on Tuesday)

passwd command

change password

*Use the **passwd** command to change your password*

```
/home/cis90/simmsben $ passwd
Changing password for user simmsben.
Changing password for simmsben
(current) UNIX password:
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
/home/cis90/simmsben $
```

Note, the passwords are not echoed as you type them.

John the Ripper

An open source cracker that tries common passwords first followed by a brute force dictionary attack

john-1.7.9/run/password.lst has most popular passwords to try first

Housekeeping

Housekeeping

1. Student surveys due today
2. Lab 1 submittal due by 11:59PM tonight
3. Last day to add is Saturday 2/18

Turn OFF the recording

Roll Call

Turn recording back ON

CIS 90 – Code Names

Lord of the Rings Characters

Current Progress

Code Name	Grading Choice	Q1	Q2	Q3	Q4
Max Points		3	3	3	3
aragorn	Grade				
arwen	Grade				
balrog	Grade				
boromir	Grade				
denethor	Grade				
dwalin	Grade				
elrond	Grade				
eomer	Grade				
eowyn	Grade				
faramir	Grade				
frodo	Grade				
galadriel	Grade				
gimli	Grade				
glorfindel	Grade				
ioreth	Grade				
legolas	Grade				
lobelia	Grade				
nazgul	Grade				
pippin	Grade				
saruman	Grade				
sauron	Grade				
theoden	Grade				
treebeard	Grade				

Everyone who is enrolled for this course will be assigned a code name.

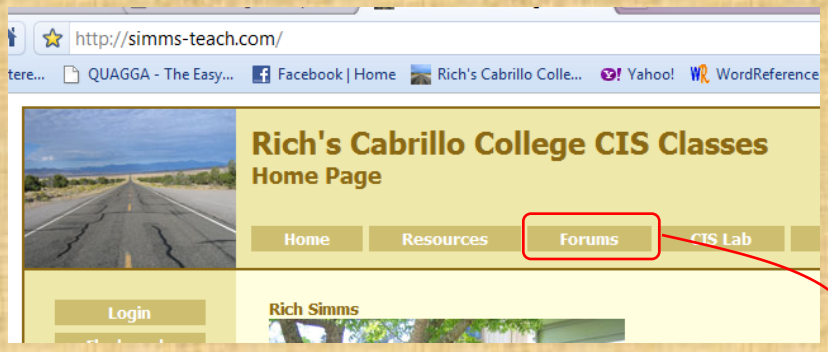
I will use your grading choice on the survey you send me (you can change your mind later)

I'll start sending out code names tomorrow for anyone who has sent me their survey.

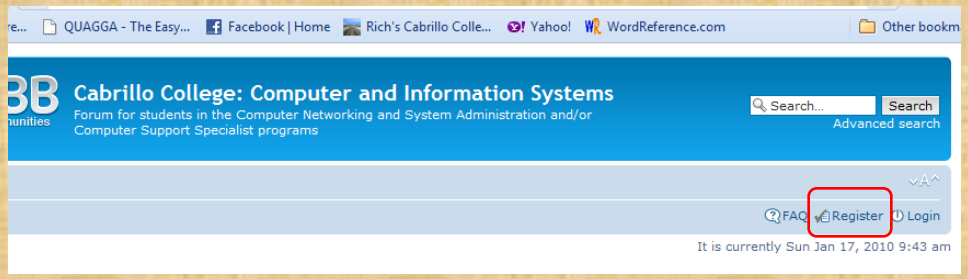
Class Activity

Forum Registration

*There is a Forums link on **simms-teach.com***



*Or browse to **opus.cabrillo.edu/forum***



To Register:

1. Browse to the forum
2. Click on 

3. Review and agree to terms
4. Your **Username** must:
   - Be your **first and last name separated by a space**
   - e.g. Rich Simms
     Not rsimms71 or richsimms

More commands for your toolbox

Introducing some new commands for this lesson

apropos *command*	*Looks up references in the whatis database*
cat *filename*	*print a file (from concatenate)*
cd *path*	*Change to a new directory*
echo *string*	*Print string (on screen)*
file *filename*	*Show additoanal lfile information*
ls *path*	*List files in a directory*
type *command*	*Shows where command resides on the path*
bc	*Binary calculator*

apropos command

apropos - search the whatis database for strings

```
/home/cis90/simben $ apropos uname
oldolduname [obsolete] (2)  - obsolete system calls
olduname [obsolete]  (2)  - obsolete system calls
uname                (1)  - print system information
uname                (1p)  - return system name
uname                (2)  - get name and information about
current kernel
uname                (3p)  - get the name of the current
system
/home/cis90/simben $
```

cat command

concatenate files and print on the standard output

```
/home/cis90/simben $ cat letter
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.

< snipped >

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
/home/cis90/simben $
```

cd command

Change the current directory

```
/home/cis90/simben $ cd

/home/cis90/simben $ ls
bigfile   lab01-submitted      letter         Poems      small_town   timecal
bin       lab01-submitted.bak  log            proposal1  spellk       what_am_i
empty     Lab2.0               Miscellaneous  proposal2  text.err
Hidden    Lab2.1               mission        proposal3  text.fxd

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

*Using **cd** by itself with no argument will return you to your home directory*

echo command

Display a line of text

```
/home/cis90/simben $ echo hello rich
hello rich

/home/cis90/simben $ echo 123
123

/home/cis90/simben $ echo 1 2 3
1 2 3

/home/cis90/simben $ echo earth       wind   fire
earth wind fire

/home/cis90/simben $ echo "earth       wind   fire"
earth           wind   fire
```

file command

Determine file type

```
/home/cis90/simben $ file letter
letter: ASCII English text

/home/cis90/simben $ file Miscellaneous/
Miscellaneous/: directory

/home/cis90/simben $ file timecal
timecal: shell archive or script for antique kernel text
```

ls command

list directory contents

```
/home/cis90/simben $ ls letter
letter

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

/home/cis90/simben $ ls /bin/uname
/bin/uname
```

Regular files show as black, directories show as blue and executable programs/scripts show and green

type command

Locate where a command resides on your path

```
[rsimms@opus run]$ type uname
uname is /bin/uname

[rsimms@opus run]$ type cal
cal is /usr/bin/cal

[rsimms@opus run]$ type uname cal
uname is /bin/uname
cal is /usr/bin/cal
```

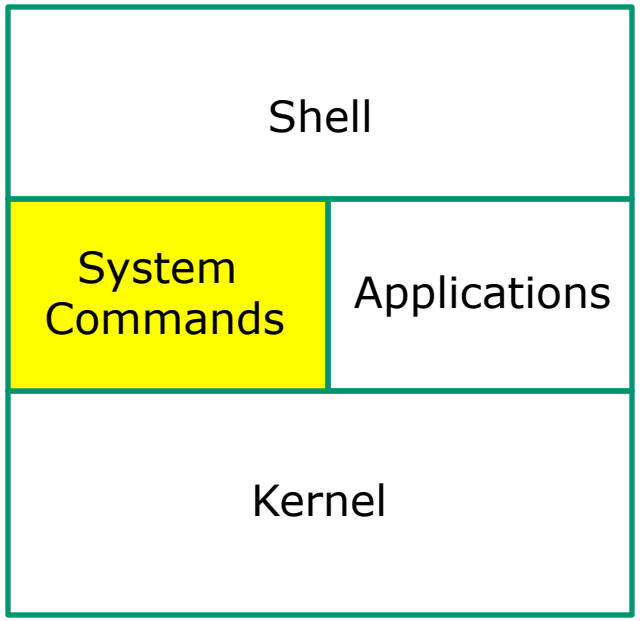
name of the file (command/program)

name of the directory where file is found

Where are the UNIX commands & utilities

UNIX/Linux Architecture

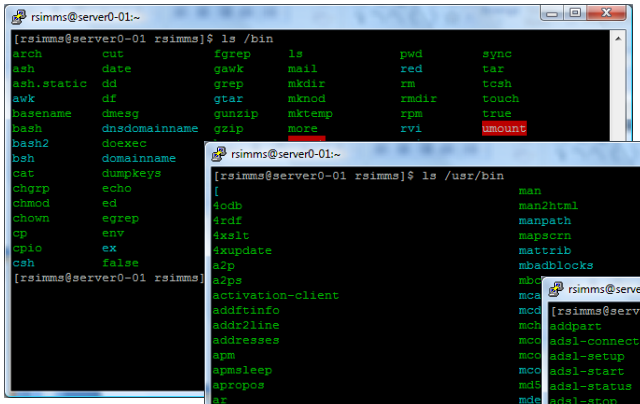
System Commands



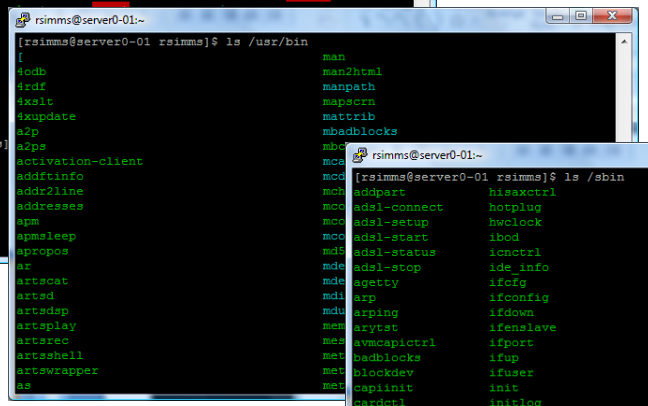
- 100's of system commands and utilities .
- Commands like **ls** (list directories), **cat** (print a file), **rm** (remove a file), … etc.
- Utilities like **vi** (text editor), **sort** (sorts file contents), **find** (searches), … etc.
- Larger utilities like **sendmail** (email), **tar** (backup), **tcpdump** (sniffer), … etc.
- Administrative utilities like **useradd**, **groupadd**, **passwd** (change password), … etc.

Commands and Utilities

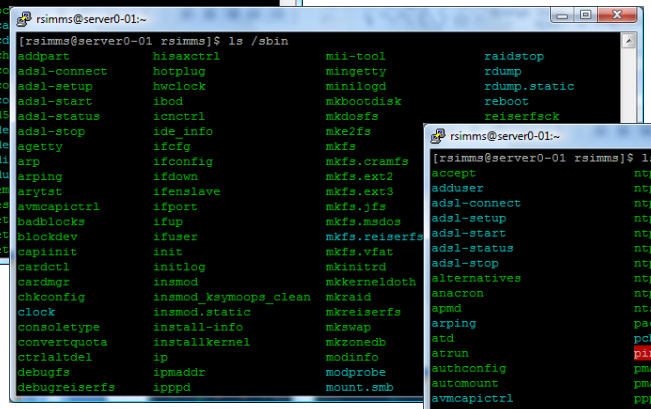
Executable binary code or scripts



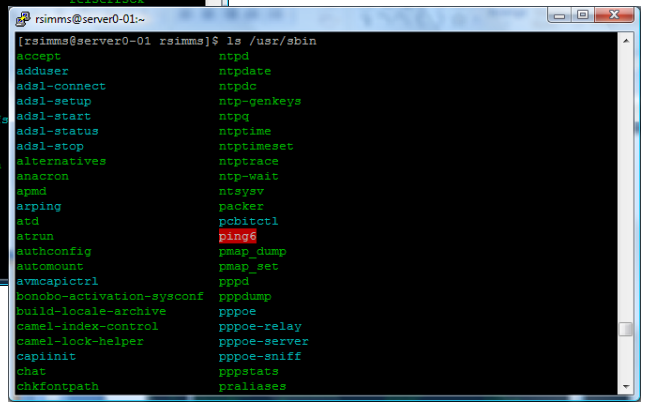
/bin



/usr/bin



/sbin



/usr/sbin

There are lots and LOTS of commands & utilities in the four "bin" (binary) directories

Commands and Utilities

The /bin directory

*Use **ls /bin** to view*

```
rsimms@nosmo:~/depot/gcal-3.01/src
[rsimms@nosmo src]$ ls /bin
alsaunmute   dnsdomainname   kbd_mode   nisdomainname   sync
arch         doexec          keyctl     pgawk           tar
ash          domainname      kill       ping            tcsh
ash.static   dumpkeys        ksh        ping6           touch
awk          echo            link       ps              tracepath
basename     ed              ln         pwd             tracepath6
bash         egrep           loadkeys   red             traceroute
bsh          env             login      rm              traceroute6
cat          ex              ls         rmdir           true
chgrp        false           mail       rpm             umount
chmod        fgrep           mailx      rvi             uname
chown        gawk            mkdir      rview           unicode_start
cp           gettext         mknod      sed             unicode_stop
cpio         grep            mktemp     setfont         unlink
csh          gtar            more       setserial       usleep
cut          gunzip          mount      sh              vi
date         gzip            mt         sleep           view
dd           hostname        mv         sort            ypdomainname
df           igawk           netstat    stty            zcat
dmesg        ipcalc          nice       su
[rsimms@nosmo src]$
```

These are core programs used by everyone.

*Can you find the **date**, **hostname**, **passwd**, **ps** and **uname** commands?*

*Can you find the **bash** shell?*

Commands and Utilities

The /usr/bin directory

*Use **ls /usr/bin** to view*

```
rsimms@opus:~
[rsimms@opus ~]$ ls /usr/bin
[
411toppm
a2p
a2ps
ab
ac
aclocal
aclocal-1.4
aclocal-1.5
aclocal-1.6
aclocal-1.7
aclocal-1.9
aconnect
acpi_listen
activation-client
addftinfo
addr2line
afs5log
alacarte
alsamixer
amidi
amixer
amtu
amuFormat.sh
animate
antlr
antlr-java
anytopnm
aplay
aplaymidi
apm
apmsleep
apropos
ar
ar86
arecord
arecordmidi
htdbm
htdigest
htmlview
htpasswd
i386
i386-redhat-linux-c++
i386-redhat-linux-g++
i386-redhat-linux-gcc
icc2ps
icclink
icctrans
iceauth
icontopbm
iconv
id
ident
identify
idn
iecset
ifnames
ilbmtoppm
imake
im-chooser
imgtoppm
import
includeres
indent
indxbib
info
infocmp
infokey
infotocap
infotopam
innochecksum
install
install-catalog
instmodsh
ppmforge
ppmglobe
ppmhist
ppmlabel
ppmmake
ppmmix
ppmnorm
ppmntsc
ppmpat
ppmquant
ppmquantall
ppmrainbow
ppmrelief
ppmrough
ppmshadow
ppmshift
ppmspread
ppmtoacad
ppmtoarbtxt
ppmtobmp
ppmtoeyuv
ppmtogif
ppmtoicr
ppmtoilbm
ppmtojpeg
ppmtoleaf
ppmtolj
ppmtolss16
ppmtomap
ppmtomitsu
ppmtompeg
ppmtoneo
ppmtopcx
ppmtopgm
ppmtopi1
ppmtopict
ppmtopj
```

There are a "ton" of commands (programs) in this directory.

You will need to scroll through a lot of pages to see them all!

*Can you find the **cal, clear**, **id, ssh, tty, and who** commands we used in Lab 1?*

Commands and Utilities

The /sbin directory

*Use **ls /sbin** to view this directory*

```
simben90@opus:~
/home/cis90/simben $ ls /sbin
accton                  installkernel           mpath_wait
addpart                 ip                      multipath
adsl-connect            ip6tables               multipathd
adsl-setup              ip6tables-restore       multipath.static
adsl-start              ip6tables-save          nameif
adsl-status             ipmaddr                 nash
adsl-stop               ipppstats               netplugd
agetty                  iprofd                  netreport
alsactl                 iptables                new-kernel-pkg
arp                     iptables-restore        nologin
arping                  iptables-save           pam_console_apply
audispd                 iptunnel                pam_tally
auditctl                iscsiadm                pam_tally2
auditd                  iscsid                  pam_timestamp_check
aureport                iscsi-iname             parted
ausearch                iscsistart              partprobe
autrace                 isdnctrl                partx
avmcapictrl             isdnlog                 pcbitctl
badblocks               iwconfig                pccardctl
blkid                   iwevent                 pcmcia-check-broken-cis
blockdev                iwgetid                 pcmcia-socket-startup
brcm_iscsiuio           iwlist                  pidof
busybox                 iwpriv                  pivot_root
capiinit                iwspy                   plipconfig
cbq                     kdump                   portmap
cciss_id                kexec                   poweroff
change_console          killall5                pppoe
```

These commands and utilities are typically used by system administrators.

*This is where the **chkconfig**, **ifconfig and modproble** commands are found.*

You will learn how to use these commands in CIS 191 and CIS 192.

Commands and Utilities

The /usr/sbin directory

Use ls /usr/sbin to view this directory

```
simben90@opus:~
/home/cis90/simben $ ls /usr/sbin
accept                          NetworkManager
accton                          newusers
acpid                           nfsstat
adduser                         nhfsgraph
adsl-connect                    nhfsnums
adsl-setup                      nhfsrun
adsl-start                      nhfsstone
adsl-status                     nm-system-settings
adsl-stop                       nscd
alsactl                         nstat
alternatives                    ntpd
anacron                         ntpdate
apachectl                       ntpdc
apmd                            ntp-keygen
arpd                            ntpq
arping                          ntptime
atd                             ntptrace
atrun                           ntp-wait
audit2why                       ntsysv
authconfig                      open_init_pty
authconfig-gtk                  ownership
authconfig-tui                  packer
automount                       pcscd
avahi-autoipd                   ping6
avahi-daemon                    pirut
avahi-dnsconfd                  plainrsa-gen
avcstat                         pluginviewer
```

These commands and utilities are typically used by system administrators.

*This is where commands like **useradd, userdel, tcpdump** are located.*

You will learn how to use these commands in CIS 191 and CIS 192.

Programs

Binary code vs text scripts

All UNIX commands & utilities are executable programs.

A program can be either binary code or text-based scripts:

- Binary machine code is unprintable. A programmer must use hex dumps to examine binary code.
- Binary machine code executes very quickly and is targeted for a specific CPU instruction set.
- The binaries are produced by compiling source code written in a higher level language such as C, or C++.
- A script can be directly viewed and printed.
- A script does not need to be compiled. It is interpreted on the fly and because of that doesn't run as fast as binary code.
- Common scripting languages include bash, perl and python.

programs must have the X (execute) permission bit set to run

Programs

Executable binary code or scripts

Lets take a deep dive on two random commands:

apropos and **cal**

apropos - searches the whatis database for a string of text
cal - prints a calendar

*We will be using the **ls**, **file, cat** and **type** commands to learn more about the **apropos** and **cal** commands*

I'll be using this graphic to indicate a program that has been loaded into memory to be run

apropos

Programs

Executable binary code or scripts

cal

```
/home/cis90/simben $ apropos uname
oldolduname [obsolete] (2)  - obsolete system calls
olduname [obsolete]  (2)  - obsolete system calls
uname                (1)  - print system information
uname                (1p)  - return system name
uname                (2)  - get name and information about current kernel
uname                (3p)  - get the name of the current system
```

*Use **apropos** to look up a reference in the whatis database.*

```
/home/cis90/simben $ cal
   February 2012
Su Mo Tu We Th Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29

/home/cis90/simben $
```

*Use **cal** to print a calendar*

Programs

Executable binary code or scripts

apropos

cal

*After changing into the /usr/bin directory, the **ls** command shows both **apropos** and **cal** are there. They show as green because they are programs (and can be executed).*

```
/home/cis90/simben $ cd /usr/bin
/usr/bin $ ls apropos cal
apropos  cal
```

*Using the **-l** option on the **ls** command prints a "long listing" that shows additional information. The x's indicate the execute permission bits are set.*

```
/usr/bin $ ls -l apropos cal
-rwxr-xr-x 1 root root  1786 Jul 12  2006 apropos
-rwxr-xr-x 1 root root 18764 Jul  3  2009 cal
```

execute permissions set

Programs

Executable binary code or scripts

apropos

cal

*The **file** command shows that **apropos** is a shell script and **cal** is binary code (has been compiled from higher level source code)*

```
/usr/bin $ file apropos
apropos: Bourne shell script text executable
/usr/bin $

/usr/bin $ file cal
cal: 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
/usr/bin $
```

Programs

Executable binary code or scripts

simmsben@opus:/usr/bin

```
/usr/bin $ cat apropos
#!/bin/sh
#
# apropos -- search the whatis database for keywords.
# whatis  -- idem, but match only commands (as whole words).
#
# Copyright (c) 1990, 1991, John W. Eaton.
# Copyright (c) 1994-1999, Andries E. Brouwer.
#
# You may distribute under the terms of the GNU General Public
# License as specified in the README file that comes with the man
# distribution.
#
# apropos/whatis-1.5m aeb 2003-08-01 (from man-1.6d)
#
# keep old PATH - 000323 - Bryan Henderson
# also look in /var/cache/man - 030801 - aeb

program=`basename $0`

# When man pages in your favorite locale look to grep like binary files
# (and you use GNU grep) you may want to add the 'a' option to *grepopt1.
ap
ap
wh
wh
gr
gr
if
then
    echo "usage: $program keyword ..."
    exit 1
fi

manpath=`man --path | tr : '\040'`

if [ "$manpath" = "" ]
then
    echo "$program: manpath is null"
    exit 1
```

*The **cat** command can print the apropos file because it is a readable **ASCII** script*

simmsben@opus:/usr/bin

```
/usr/bin $ cat cal
ELF4tD4(444444090909090400%i9iìÐÐHHH  PåtdÈ6ÈÈQåtd/lib/ld-linux.so.2GNU   libn
curses.so.5__gmon_start___Jv_RegisterClassestgetent_fini_inittputstgetstrlib
c.so.6_IO_stdin_usedstrcpy__printf_chkexit_IO_putcsetlocaleoptindstrrchr__sw
printf_chk__prognamedcgettextstrncpymbstowcs__stack_chk_failputcñêÓi3Ã÷EI*9¨
IK'ÿ¯^o"HU"  dpßC2öFÏñ´F½29öNóÿ°ñÿìñÿ¹ ð`$¤(CEÕi¼Pv¼ÎÊ¬KãÀ8ò0qX³memcpy__strt
ìñÿB@ternalnl_langinfogetenv__q ype_b_locstderr__snprintf_chklocaltime__vfpr
intf_chkwcstombs__sprintf_chÃO ndtextdomain__libc_start_main_edata__bss_star
t_endGLIBC_2.3GLIBC_2.3.4GàR C_2.4GLIBC_2.0libdl.so.2/lib/ld-linux.so.2qFXHÊ
¿³VSFXH QLû.SFXHRB]f9SFX`T £¹i£¡Üÿÿÿ¡ ¡üÿÿÿ°¡Èÿÿÿ¸¡Ðÿÿÿ¸Ô¡ ¡$48¡Øÿÿÿ<¡ÔÿÿÿL¡h
¡¡ôÿÿÿ¬¡Ãÿÿÿ´¡øÿÿÿÔ¡Ìÿÿÿ
$¼3*ÌÐÔØÜàè°    ì-ð°
ô°
 ø°
»°
 $(,04»
```

*The **cat** command "chokes" trying to print the **binary** cal file.*

That's because binary files contain unprintable characters.

Programs

Executable binary code or scripts

From: gcal-3.01.tar.gz

rsimms@nosmo:~/depot/gcal-3.01/src

```
[rsimms@nosmo src]$ head -50 gcal.c
/*
*  gcal.c:  Main part which controls the extended calendar program.
*
*
*  Copyright (c) 1994, 95, 96, 1997, 2000 Thomas Esken
*
*  This software doesn't claim completeness, correctness or usability.
*  On principle I will not be liable for ANY damages or losses (implicit
*  or explicit), which result from using or handling my software.
*  If you use this software, you agree without any exception to this
*  agreement, which binds you LEGALLY !!
*
*  This program is free software; you can redistribute it and/or modify
*  it under the terms of the `GNU General Public License' as published by
*  the `Free Software Foundation'; either version 2, or (at your option)
*  any later version.
*
*  You should have received a copy of the `GNU General Public License'
*  along with this program; if not, write to the:
*
*    Free Software Foundation, Inc.
*    59 Temple Place - Suite 330
*    Boston, MA 02111-1307,  US
*/



static char rcsid[]="$Id: gcal.c


/*
*  Include header files.
*/
#include "tailor.h"
#if HAVE_ASSERT_H
#  include <assert.h>
```

cal

*Note: The **cal** binary code resulted from compiling the original gcal.c source code.*

rsimms@nosmo:~/depot/gcal-3.01/src

```
[rsimms@nosmo src]$ file /usr/bin/cal
/usr/bin/cal: ELF 32-bit LSB executable, Intel 80386, version 1
(SYSV), for GNU/Linux 2.2.5, dynamically linked (uses shared lib
s), stripped
[rsimms@nosmo src]$
```

Because GNU Linux software is licensed under the GPL you can make your own custom version of the commands or the kernel!

Inputs to programs
(commands and scripts)

You will get these questions when you submit Lab 2

Name a UNIX command that gets its input only from the command line?

Name an interactive command that reads its input from the keyboard?

Name a UNIX command that gets its input from the Operating System?

Name a UNIX command that gets its input only from the command line?

```
/home/cis90/simmsben $ echo hello world
hello world
```

*The **echo** command is an example of a command that gets its input from the command line*

Name a UNIX command that gets its input only from the command line?

```
/home/cis90/simmsben $ banner hello world
#      # ####### #        #        #######
#      # #       #        #        #     #
#      # #       #        #        #     #
####### #####   #        #        #     #
#      # #       #        #        #     #
#      # #       #        #        #     #
#      # ####### ####### ####### #######

#      # ####### ######  #        ######
#   #  # #     # #     # #        #     #
#   #  # #     # #     # #        #     #
#   #  # #     # ######  #        #     #
#   #  # #     # #   #   #        #     #
#   #  # #     # #    #  #        #     #
 ## ##  ####### #     # ####### ######
```

*The **banner** command is an example of a command that gets its input from the command line*

Name an interactive command that reads its input from the keyboard?

```
/home/cis90/simmsben $ bc
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
2+2
4
500-200+3
303
sqrt(64)
8
quit
```

*The **bc** (binary calculator) command is an example of an interactive command that reads its input from the keyboard*

Name an interactive command that reads its input from the keyboard?

```
/home/cis90/simmsben $ passwd
Changing password for user simmsben.
Changing password for simmsben
(current) UNIX password:
New UNIX password:
BAD PASSWORD: is too similar to the old one
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
/home/cis90/simmsben $
```

*The **passwd** command is an example of an interactive command that reads its input from the keyboard*

Name a UNIX command that gets its input from the Operating System?

```
/home/cis90/simmsben $ who
dycktim  pts/1        2010-09-07 17:07 (nosmo-nat.cabrillo.edu)
root     :0           2009-12-18 17:30
velasoli pts/2        2010-09-07 17:08 (adsl-75-41-114-88.dsl.pltn13.sbcglobal.net)
guest90  pts/3        2010-09-07 16:56 (nosmo-nat.cabrillo.edu)
rsimms   pts/4        2010-09-07 15:54 (dsl-63-249-103-107.dhcp.cruzio.com)
guest90  pts/5        2010-09-07 16:59 (nosmo-nat.cabrillo.edu)
watsohar pts/6        2010-09-07 17:03 (nosmo-nat.cabrillo.edu)
swansgre pts/7        2010-09-07 17:10 (nosmo-nat.cabrillo.edu)
guest90  pts/8        2010-09-07 17:10 (nosmo-nat.cabrillo.edu)
abbenste pts/9        2010-09-07 17:11 (nosmo-nat.cabrillo.edu)
/home/cis90/simmsben $
```

*The **who** command is an example of a command that gets its input from the Operating System*

Name a UNIX command that gets its input from the Operating System?

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

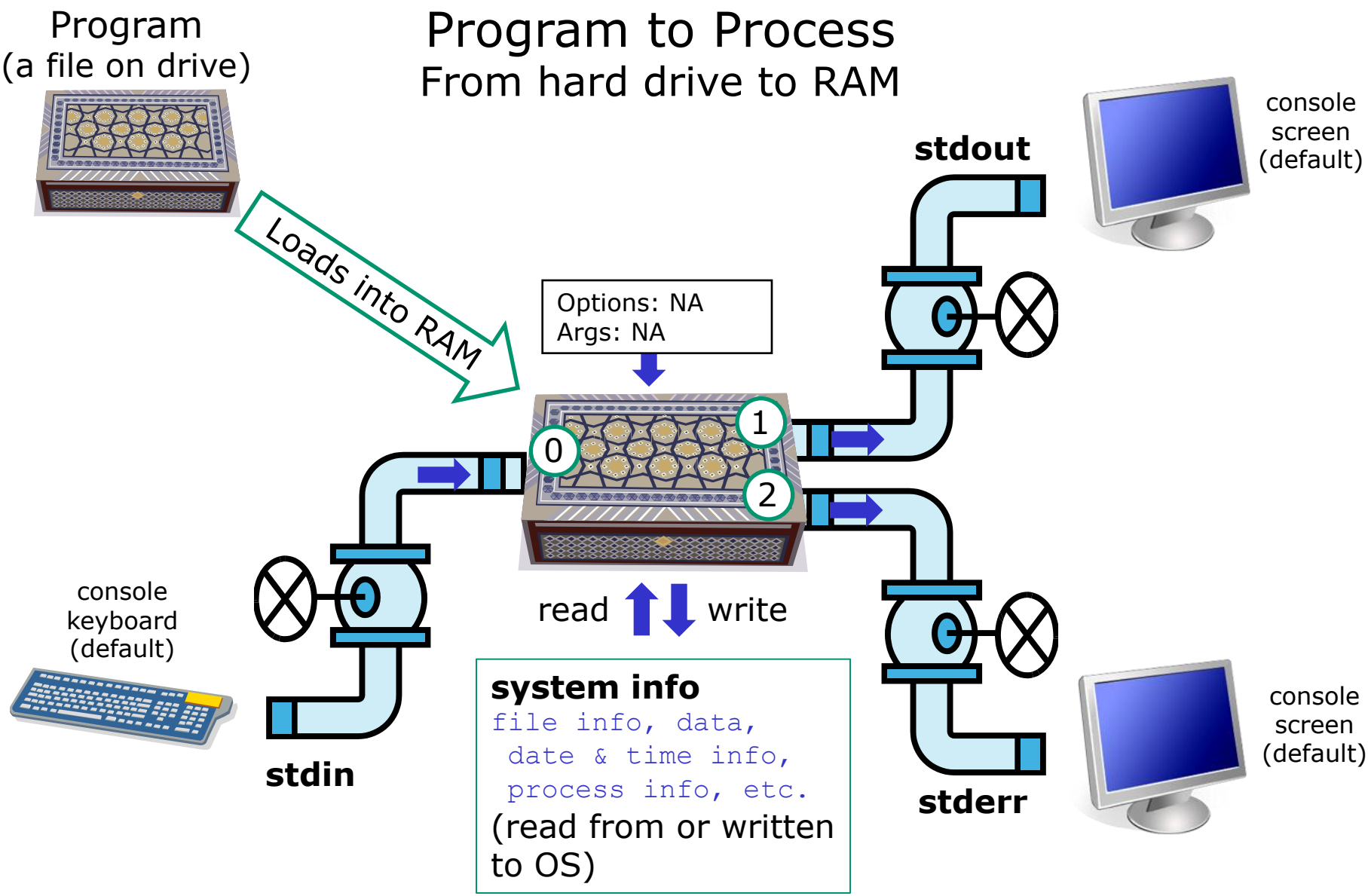
*The **uname** command is an example of a command that gets its input from the Operating System*

Program to Process

The next slides are a preview of future lessons on processes … for now just you don't need to understand all the ins and outs of how this works.

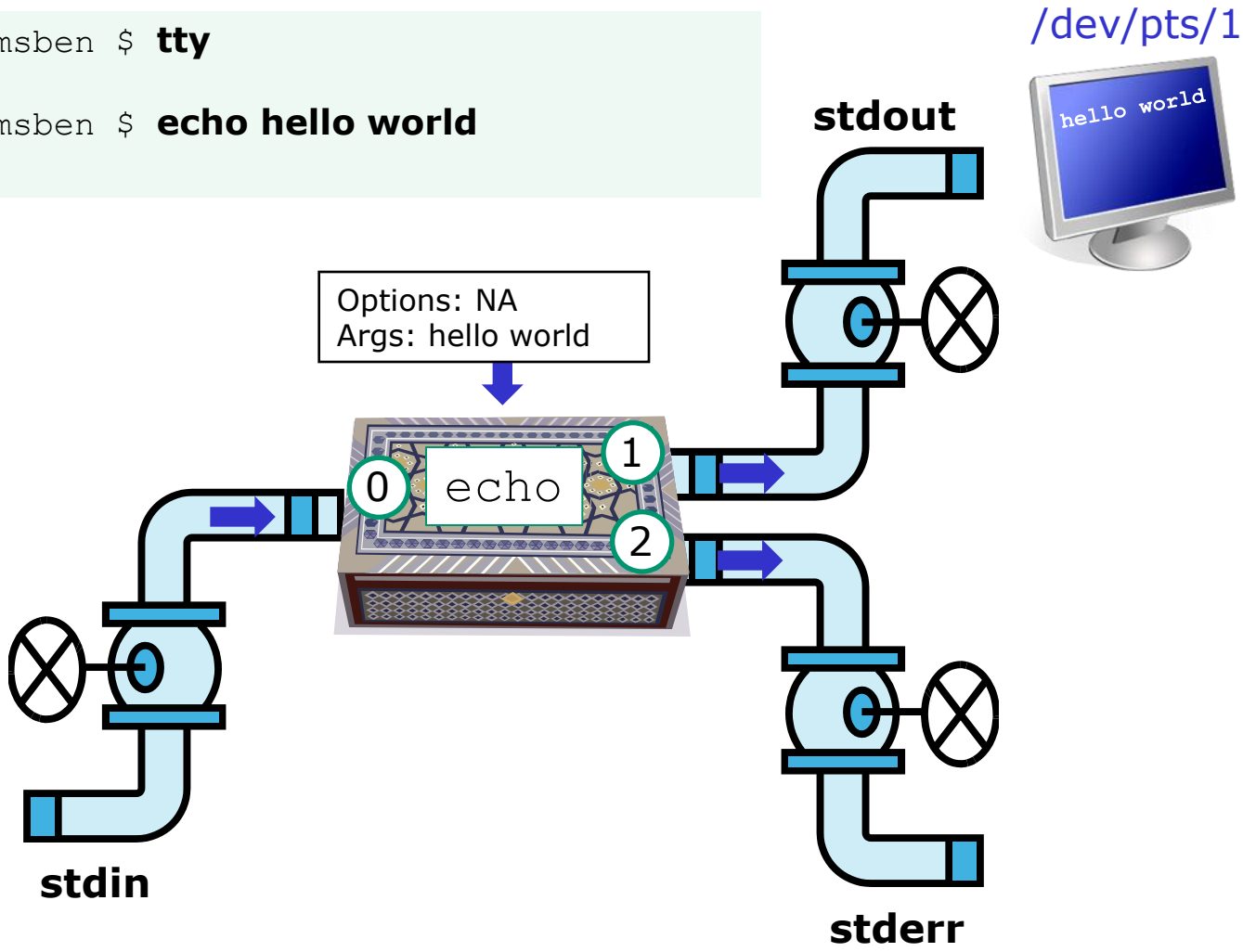
Program to Process

From hard drive to RAM



Example program to process: echo command

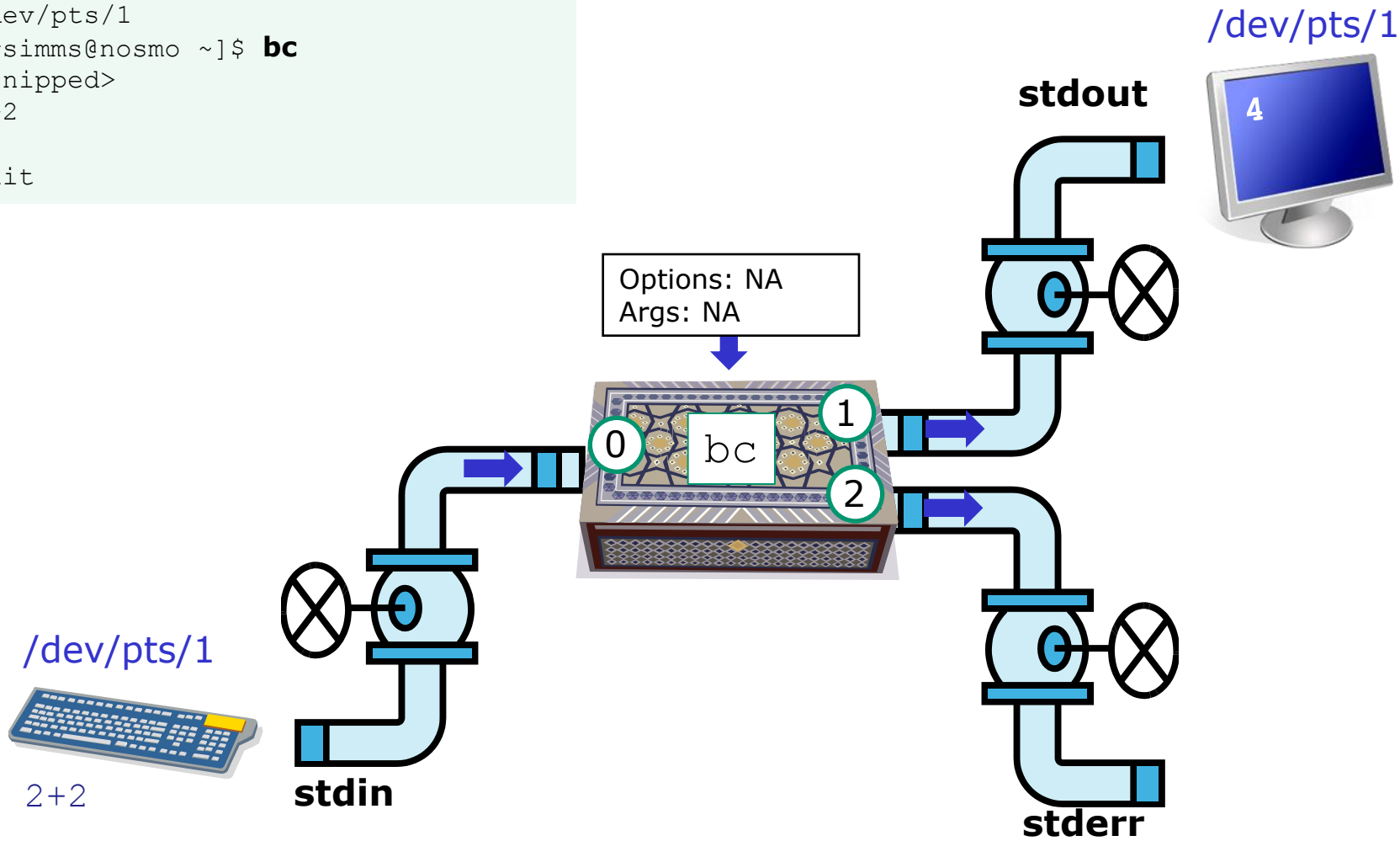
```
/home/cis90/simmsben $ tty
/dev/pts/1
/home/cis90/simmsben $ echo hello world
hello world
```



*The **echo** command is an example of a command that gets its input from the command line*

Example program to process: bc command

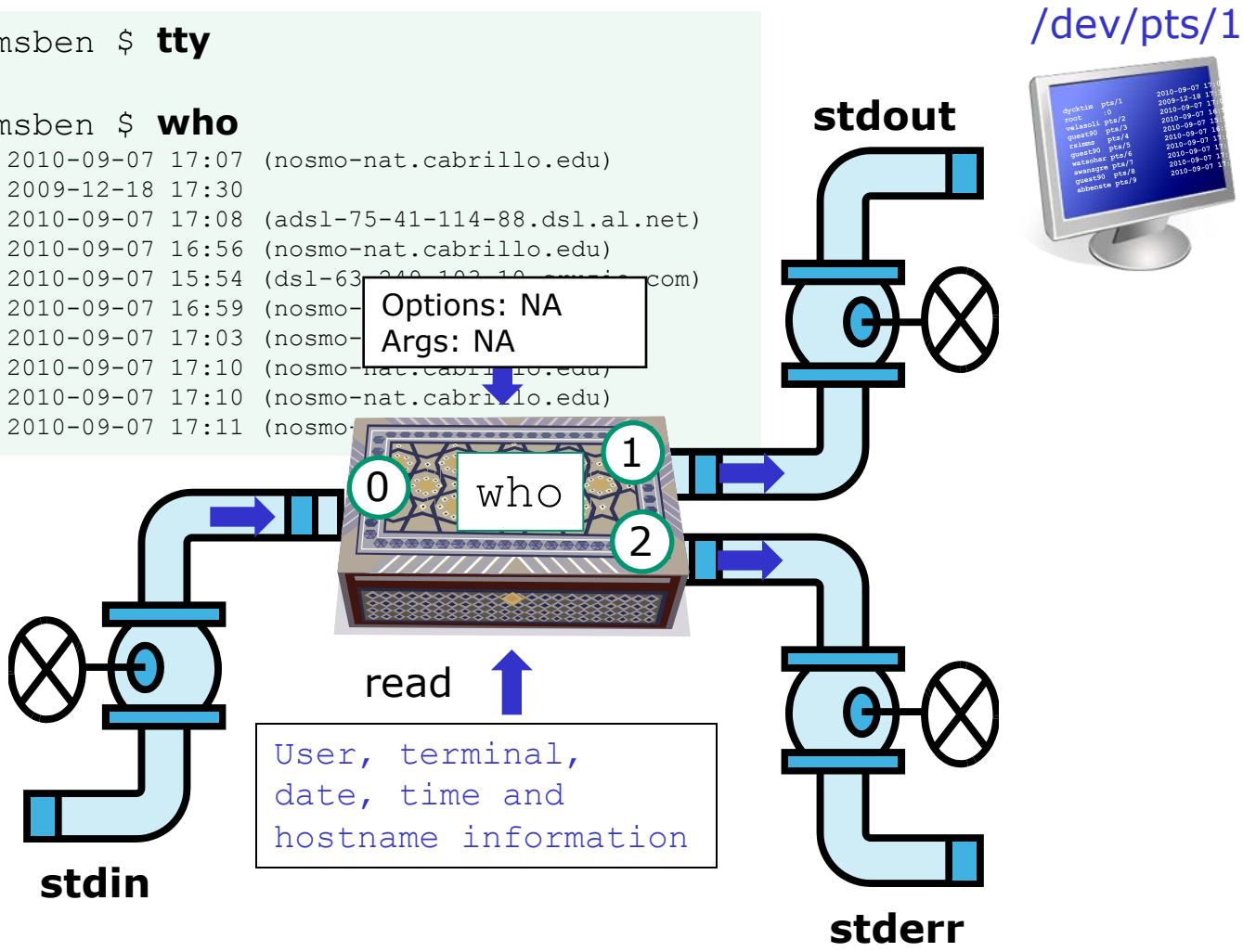
```
[rsimms@nosmo ~]$ tty
/dev/pts/1
[rsimms@nosmo ~]$ bc
<snipped>
2+2
4
quit
```



*The **bc** (binary calculator) command is an example of an interactive command that reads its input from the keyboard*

Example program to process: who command

```
/home/cis90/simmsben $ tty
/dev/pts/1
/home/cis90/simmsben $ who
dycktim  pts/1        2010-09-07 17:07 (nosmo-nat.cabrillo.edu)
root     :0           2009-12-18 17:30
velasoli pts/2        2010-09-07 17:08 (adsl-75-41-114-88.dsl.al.net)
guest90  pts/3        2010-09-07 16:56 (nosmo-nat.cabrillo.edu)
rsimms   pts/4        2010-09-07 15:54 (dsl-63-249-103-10-[illegible]com)
guest90  pts/5        2010-09-07 16:59 (nosmo-[illegible]
watsohar pts/6        2010-09-07 17:03 (nosmo-[illegible]
swansgre pts/7        2010-09-07 17:10 (nosmo-nat.cabrillo.edu)
guest90  pts/8        2010-09-07 17:10 (nosmo-nat.cabrillo.edu)
abbenste pts/9        2010-09-07 17:11 (nosmo-[illegible]
```



*The **who** command is an example of a command that gets its input from the Operating System*

Class Exercise

Running Programs

1. Use **echo 1 2 3 a b c**
   (this command got its input from the command line)

2. Use **bc** to add 2+2
   (this command read its input from the keyboard)

3. Run the **who** command
   (this command got its input from the operating system)

Command Syntax

(grammar lesson)

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. Multiple arguments are separated by spaces.

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.

Command Syntax

Command	Options	Arguments	Redirection

Command – usually at the beginning of the line

Options – follow the command, usually starts with a dash, may be combined after a single "-" or separated by spaces (`-iad = -i -a -d`)

Arguments – follow the options. Multiple arguments must be separated by spaces.

Redirection – Will be a <, >, >>, 2> or | followed by where the redirection is going or coming from.

Spaces are required between commands, options, arguments and any redirection

Multiple spaces are treated as a single space (unless inside quotes)

from Dictionary.com

parse [pahrs, pahrz] ***verb, parsed, pars·ing***.
verb (used with object)

1. to analyze (a sentence) in terms of grammatical constituents, identifying the parts of speech, syntactic relations, etc.

2. to describe (a word in a sentence) grammatically, identifying the part of speech, inflectional form, syntactic function, etc.

3. Computers . to analyze (a string of characters) in order to associate groups of characters with the syntactic units of the underlying grammar.

Command Syntax

Command	Options	Arguments	Redirection

The command syntax is the underlying grammar used to parse the command line

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

/home/cis90/simben $ uname -o
GNU/Linux

/home/cis90/simben $ ls -ld Poems/
drwxr-xr-x 5 simben90 cis90 4096 Jan 18  2004 Poems/

/home/cis90/simben $ ls -li letter > /dev/null
```

More on redirection in later lessons

Command Syntax

Command	Options	Arguments	Redirection
clear			
hostname			
hostname	-s		
id			
id		root	
ls			
ls	-l		
ls	-l -i	Poems/	
ls	-li	letter log	
ls	-ld	Miscellaneous	> myfile
echo		red blue	
echo		"red blue"	
echo		Hello	>> myfile

More on redirection in later lessons

Command Syntax

Command	Options	Arguments	Redirection

```
/home/cis90/simben $ echo I love Linux
I love Linux
```

Please parse the command line above

Command: echo

Options:
How many: NA
What are they: NA

Arguments:
How many: 3
What are they: I, Love, Linux

Redirection:
How many: NA
What is redirected: NA

Command Syntax

Command	Options	Arguments	Redirection

```
/home/cis90/simben $ ls -ld /bin /usr/bin
drwxr-xr-x 2 root root  4096 Nov 23 13:49 /bin
drwxr-xr-x 2 root root 61440 Nov 23 13:49 /usr/bin
```

Please parse the command line above

Command: ls

Options:
- How many: 2
- What are they: l, d

Arguments:
- How many: 2
- What are they: /bin, /usr/bin

Redirection:
- How many: NA
- What is redirected: NA

Command Syntax

Command	Options	Arguments	Redirection

```
/home/cis90/simben $ ls-ld/bin/usr/bin
-bash: ls-ld/bin/usr/bin: No such file or directory
```

Please parse the command line above

Command: ls-ld/bin/usr/bin

Options:
How many: NA
What are they: NA

Arguments:
How many: NA
What are they: NA

Redirection:
How many: NA
What is redirected: NA

Spaces are required between commands, options, arguments and any redirection

Command Syntax

Command	Options	Arguments	Redirection

```
/home/cis90/simben $ file proposal1 timecal
proposal1: ASCII English text
timecal:   shell archive or script for antique kernel text
```

Please parse the command line above

Command: file

Options:
How many: NA
What are they: NA

Arguments:
How many: 2
What are they: proposal1, timecal

Redirection:
How many: NA
What is redirected: NA

Command Syntax

Command	Options	Arguments	Redirection

```
/home/cis90/simben $ ls -l -i -a /bin Poems/ letter small_town > /dev/null
/home/cis90/simben $
```

Please parse the command line above

Command: ls

Options:
How many: 3
What are they: l, i, a

Arguments:
How many: 4
What are they: /bin, Poems/, letter, small_town

Redirection:
How many: 1
What is redirected: stdout redirected to /dev/null

Command Syntax

Command	Options	Arguments	Redirection

```
/home/cis90/simben $ echo "1  2  3  4  5"
1  2  3  4  5
```

Please parse the command line above

Command: echo

Options:
How many: NA
What are they: NA

Arguments:
How many: 1
What are they: "1 2 3 4 5"

Redirection:
How many: NA
What is redirected: NA

Environment Variables

echo command

echo prints the arguments supplied on the command line

```
[rsimms@opus]$ echo hello
hello
[rsimms@opus]$ echo "My name is Rich"
My name is Rich
[rsimms@opus]$ echo LOGNAME
LOGNAME
[rsimms@opus]$ echo $LOGNAME
rsimms
```

What is the deal with $LOGNAME ???
*… it is something called a **variable***

variables

LOGNAME is a predefined variable that is set by the system to hold your username

$LOGNAME

The $ is a special metacharacter and it means "the value of"

Variables

A little tiny bit of "programming" now

Think of variables as named boxes and the $ in front of a variable name means "the contents of"

```
$ echo $LOGNAME
simmsben

$ echo $HOSTNAME
opus.cabrillo.edu

$ echo $HOME
/home/cis90/simmsben

$ echo $SHELL
/bin/bash
```

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, linux, etc.

Shell (Environment) Variables

common environment variables

Shell Variable	Description
TERM	Type of terminal device , e.g. dumb, vt100, xterm, ansi, linux, etc.

Note the TERM variable gets set every time we log into Opus

Environment variables

Showing the value of a variable

echo $VARIABLENAME

Use echo and a $ in front of the variable to display the contents of that variable

Examples:

```
echo $TERM
echo $PS1
```

Environment variables

Showing the value of a variable

Use **echo $***VARIABLENAME* to show the value of a variable

```
[rsimms@nosmo ~]$ echo $PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/rsimms/bin
```

The is the path used by the shell to locate commands

```
[rsimms@nosmo ~]$ echo $TERM
xterm
```

The is the type of terminal being used

```
[rsimms@nosmo ~]$ echo $HOME
/home/rsimms
```

The is your home directory

```
[rsimms@nosmo ~]$ echo $PS1
[\u@\h \W]\$
```

The defines your shell prompt string

Environment variables

Setting the value of a variable

VARIABLENAME=newvalue

Use = (no spaces, no $ sign) to change the value of a variable

Examples:

TERM=dumb
PS1="Enter command: "

Environment variables

Setting the value of a variable

Changing the value of the TERM variable changes the terminal type

```
/home/cis90/simben $ echo $TERM
xterm


/home/cis90/simben $ TERM=dumb
/home/cis90/simben $ echo $TERM
dumb


/home/cis90/simben $ TERM=xterm
/home/cis90/simben $ echo $TERM
xterm
```

In Lab 2 you will see what happens when the terminal type is changed

Environment variables

Changing your prompt

Changing the value of the PS1 variable changes the prompt

```
/home/cis90/simben $ echo $PS1
$PWD $
/home/cis90/simben $
/home/cis90/simben $ PS1="By your command > "
By your command >
By your command >
By your command > PS1='What can I do for you $LOGNAME? '
What can I do for you simben90?
What can I do for you simben90?
What can I do for you simben90?
What can I do for you simben90? PS1='$PWD $ '
/home/cis90/simben $
/home/cis90/simben $
/home/cis90/simben $
```

Environment variables

Changing the shell prompt

```
What can I do for you rsimms? PS1="[\u@\h \W]\$ "
[rsimms@nosmo ~]$
```

user name | hostname | working directory

There are some special \codes you can use when setting the prompt

Environment variables

Changing the shell prompt

Prompt Code	Meaning
\!	history command number
\#	session command number
\d	date
\h	hostname
\n	new line
\s	shell name
\t	time
\u	user name
\w	entire path of working directory
\W	only working directory
\$	$ or # (for root user)

The PS1 variable (defines the prompt) can have any combination of text, variables and these special codes.

Environment variables

Changing the shell prompt

Prompt string	Result
PS1='$PWD $ '	/home/cis90/simmsben/Poems $
PS1="\w $ "	~/Poems $
PS1="\W $ "	Poems $
PS1="\u@\h $ "	simmsben@opus $
PS1='\u@\h $PWD $ '	simmsben@opus /home/cis90/simmsben/Poems $
PS1='\u@\$HOSTNAME $PWD $ '	simmsben@opus.cabrillo.edu /home/cis90/simmsben/Poems $
PS1='\u \! $PWD $ '	simmsben 825 /home/cis90/simmsben/Poems $
PS1="[\u@\h \W] $ "	[simmsben@opus Poems] $

Important: Use single quotes around variables that change. For example if you use $PWD with double quotes, the prompt will not changes as you change directories! More on this later …

Shell Variables

set command

```
/home/cis90/simmsben/Poems $ set
BASH=/bin/bash
BASH_ARGC=()
BASH_ARGV=()
BASH_ENV=/home/cis90/simmsben/.bashrc
BASH_LINENO=()
BASH_SOURCE=()
BASH_VERSINFO=([0]="3" [1]="2" [2]="25" [3]="1"
[4]="release" [5]="i686-redhat-linux-gnu")
BASH_VERSION='3.2.25(1)-release'
COLORS=/etc/DIR_COLORS.xterm
COLUMNS=80
CVS_RSH=ssh
DIRSTACK=()
EUID=1160
GROUPS=()
G_BROKEN_FILENAMES=1
HISTFILE=/home/cis90/simmsben/.bash_history
HISTFILESIZE=1000
HISTSIZE=1000
HOME=/home/cis90/simmsben
HOSTNAME=opus.cabrillo.edu
HOSTTYPE=i686
IFS=$' \t\n'
IGNOREEOF=10
INPUTRC=/etc/inputrc
LANG=en_US.UTF-8
LESSOPEN='|/usr/bin/lesspipe.sh %s'
LINES=24
LOGNAME=simmsben
LS_COLORS='no=00:fi=00:di=00;34:ln=00;36:pi=40;33:so=00;35
:bd=40;33;01:cd=40;33;01:or=01;05;37;41:mi=01;05;37;41:ex=
00;32:*.cmd=00;32:*.exe=00;32:*.com=00;32:*.btm=00;32:*.ba
t=00;32:*.sh=00;32:*.csh=00;32:*.tar=00;31:*.tgz=00;31:*.a
rj=00;31:*.taz=00;31:*.lzh=00;31:*.zip=00;31:*.z=00;31:*.Z
=00;31:*.gz=00;31:*.bz2=00;31:*.bz=00;31:*.tz=00;31:*.rpm=
00;31:*.cpio=00;31:*.jpg=00;35:*.gif=00;35:*.bmp=00;35:*.x
bm=00;35:*.xpm=00;35:*.png=00;35:*.tif=00;35:'
MACHTYPE=i686-redhat-linux-gnu
MAIL=/var/spool/mail/simmsben
MAILCHECK=60
OLDPWD=/home/cis90/simmsben
OPTERR=1
OPTIND=1
OSTYPE=linux-gnu
PATH=/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/home/
cis90/simmsben/../bin:/home/cis90/simmsben/bin:.
PIPESTATUS=([0]="0")
PPID=26514
PROMPT_COMMAND='echo -ne
"\033]0;${USER}@${HOSTNAME%%.*}:${PWD/#$HOME/~}"; echo -ne
"\007"'
PS1='$PWD $'
PS2='> '
PS4='+ '
PWD=/home/cis90/simmsben/Poems
SHELL=/bin/bash
SHELLOPTS=braceexpand:emacs:hashall:histexpand:ignoreeof:i
nteractive-comments:monitor
SHLVL=1
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
TERM=xterm
UID=1160
USER=simmsben
USERNAME=
_=env
consoletype=pty
```

*The **set** command shows all shell variables including the special environment variables. Users may make their own variables too.*

Shell (Environment) Variables

env command

```
/home/cis90/simmsben/Poems $ env
HOSTNAME=opus.cabrillo.edu
SHELL=/bin/bash
TERM=xterm
HISTSIZE=1000
USER=simmsben
LS_COLORS=no=00:fi=00:di=00;34:ln=00;36:pi=40;33:so=00;35:bd=40;33;01:cd=40;33;01:or=01;05;37;41:mi
=01;05;37;41:ex=00;32:*.cmd=00;32:*.exe=00;32:*.com=00;32:*.btm=00;32:*.bat=00;32:*.sh=00;32:*.csh=
00;32:*.tar=00;31:*.tgz=00;31:*.arj=00;31:*.taz=00;31:*.lzh=00;31:*.zip=00;31:*.z=00;31:*.Z=00;31:*
.gz=00;31:*.bz2=00;31:*.bz=00;31:*.tz=00;31:*.rpm=00;31:*.cpio=00;31:*.jpg=00;35:*.gif=00;35:*.bmp=
00;35:*.xbm=00;35:*.xpm=00;35:*.png=00;35:*.tif=00;35:
USERNAME=
MAIL=/var/spool/mail/simmsben
PATH=/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/home/cis90/simmsben/../bin:/home/cis90/simmsbe
n/bin:.
INPUTRC=/etc/inputrc
PWD=/home/cis90/simmsben/Poems
LANG=en_US.UTF-8
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
SHLVL=1
HOME=/home/cis90/simmsben
BASH_ENV=/home/cis90/simmsben/.bashrc
LOGNAME=simmsben
CVS_RSH=ssh
LESSOPEN=|/usr/bin/lesspipe.sh %s
G_BROKEN_FILENAMES=1
_=/bin/env
OLDPWD=/home/cis90/simmsben
/home/cis90/simmsben/Poems $
```

*The **env** command shows all the special environment variables used by the shell*

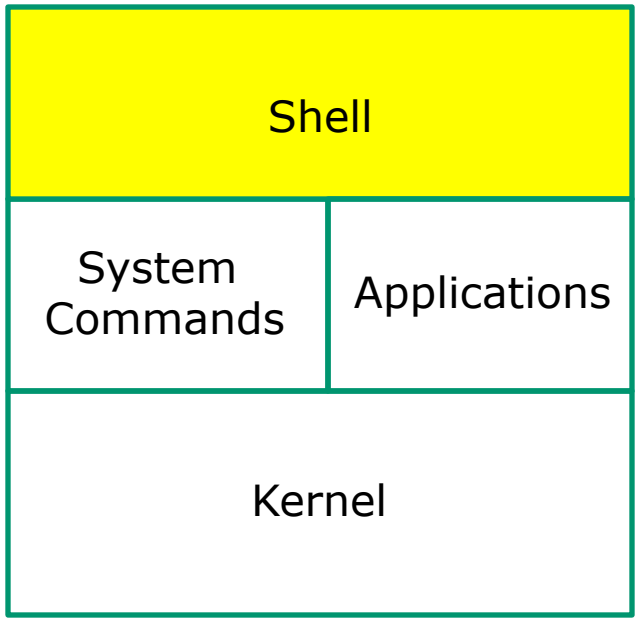
Class Exercise

Environment Variables

1. Change your prompt to "What is your command master? "
2. Use **echo** to show your logname ($LOGNAME)

Shell

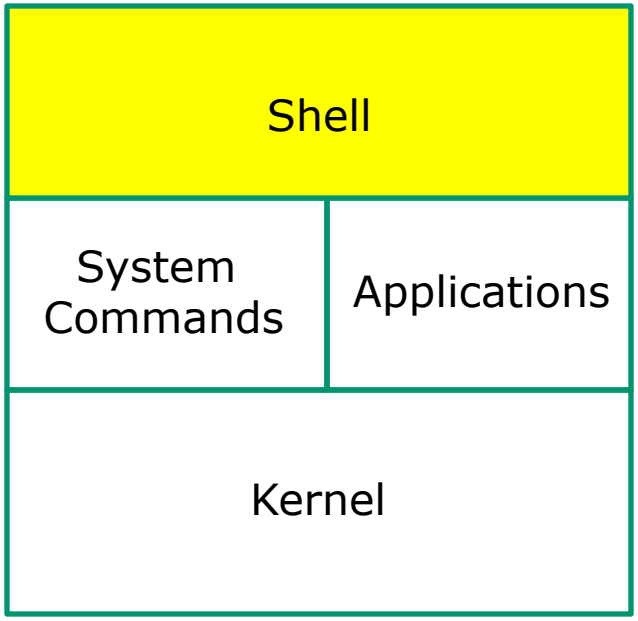
The Shell



- Allows users to interact with the computer via a "**command line**".
- **Prompts** for a command, parses the command, finds the right program and gets that program executed.
- Is called a "**shell**" because it hides the underlying operating system.

- Multiple shell programs are available: **sh** (Bourne shell), **bash** (born again shell), **csh** (C shell), **ksh** (Korn shell).
- The shell is a **user interface** and a **programming language** (scripts).
- GNOME and KDE desktops could be called **graphical shells**

Life of the Shell



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

Life of the Shell

1) Prompt user for a command

(uses the PS1 environment variable)

Examples:

```
[rsimms@opus work]$ echo $PS1
[\u@\h \W]\$
[rsimms@opus work]$
```

Regular Opus prompt for non CIS 90 classes

```
[root@nosmo ~]# echo $PS1
[\u@\h \W]\$
[root@nosmo ~]#
```

Note the change to # when logged on as root

```
/usr/bin $ echo $PS1
$PWD $
/usr/bin $
```

We use this prompt in CIS 90 to show current path

Life of the Shell

2) Parse command user typed
(based on command syntax grammar rules)

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

This is an argument which is expanded and 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
```

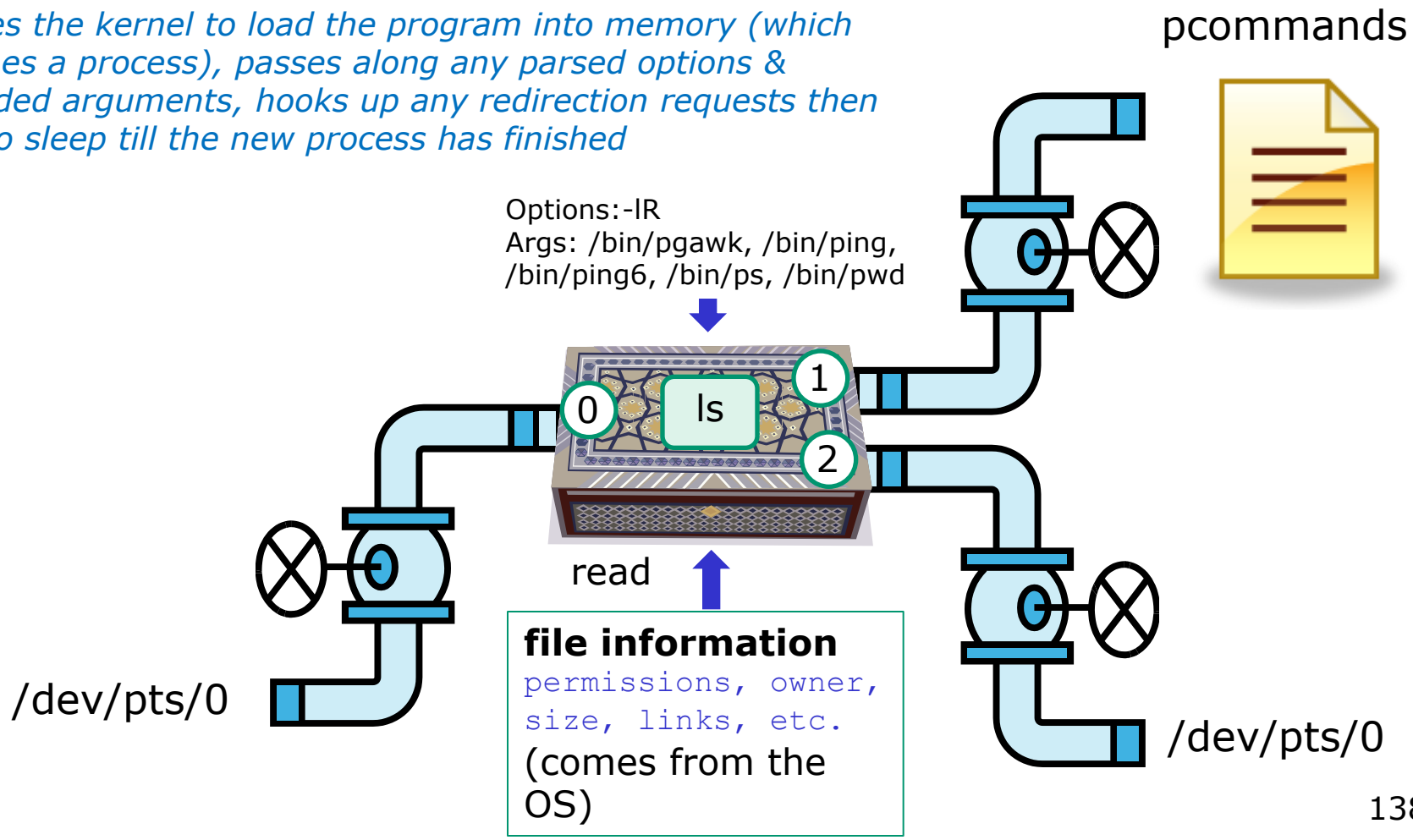
***type** command shows that **ls** is in the /bin directory*

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

Life of the Shell

4) Execute the command

Invokes the kernel to load the program into memory (which becomes a process), passes along any parsed options & expanded arguments, hooks up any redirection requests then goes to sleep till the new process has finished



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)

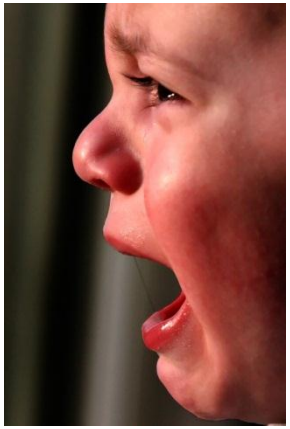
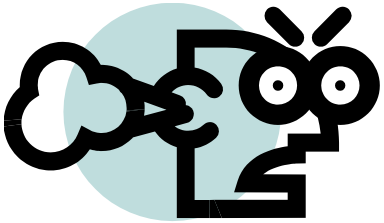
```
[rsimms@opus work]$ ls -lR /bin/p* > pcommands


[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]$
```

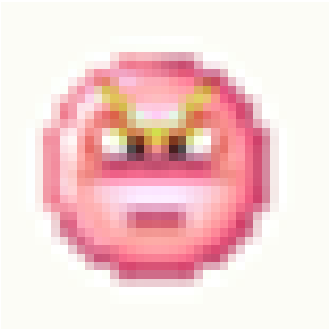
Life of the Shell

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




Command not found



What the heck !!@@##

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
```

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
```

!!@@##

!!@@##

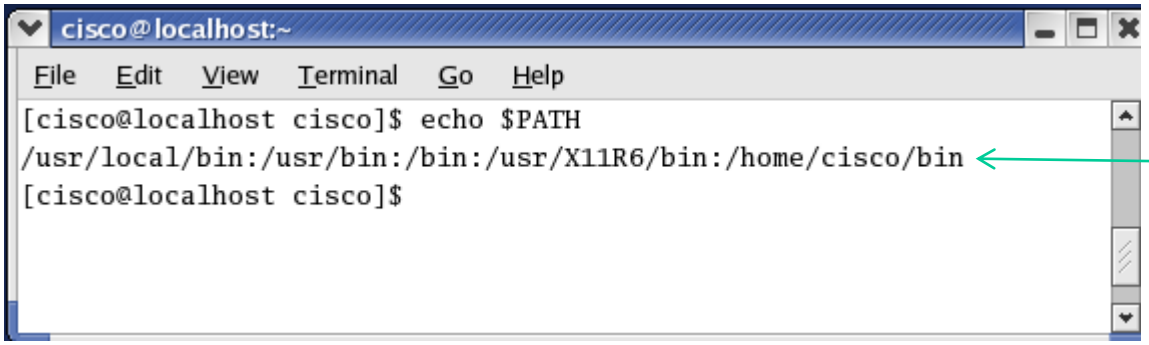
*The **hostname** and **ps** commands work fine. Why do the **iptables** and **ifconfig** commands get the "not found" error?*

… because they are not on the path

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



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

What the heck !!@@##

The Shell and the PATH

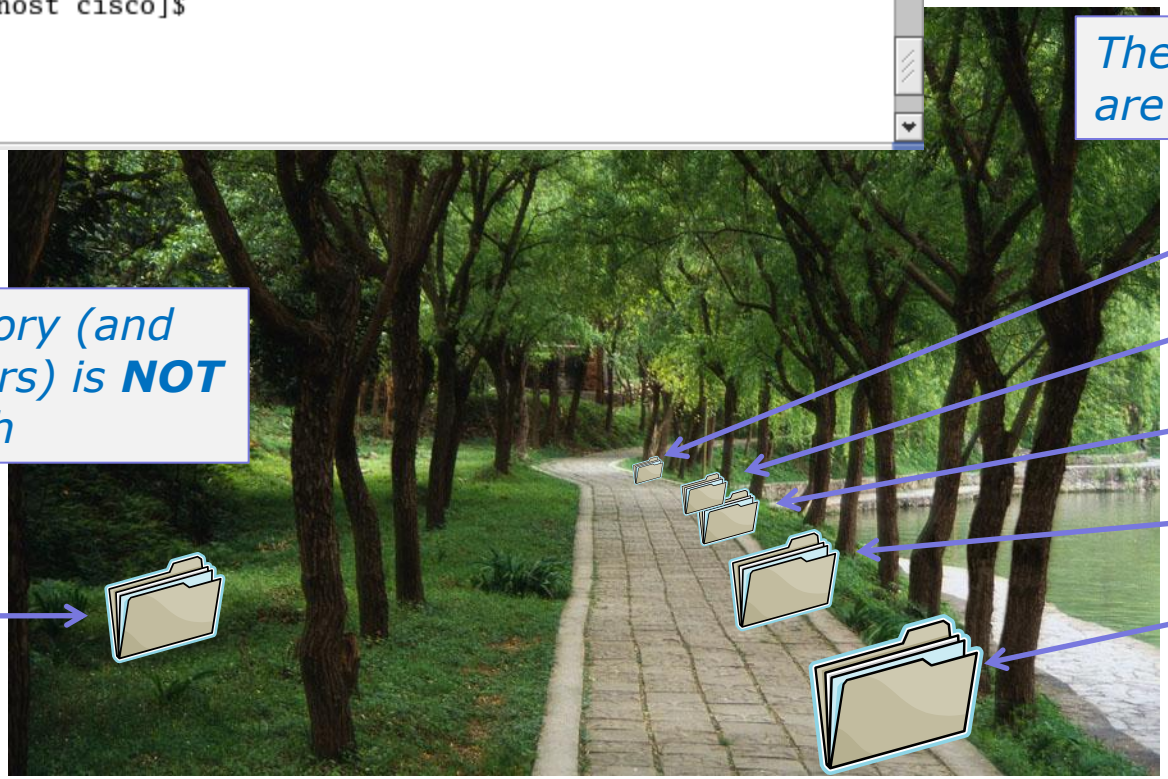
```
cisco@localhost:~
File  Edit  View  Terminal  Go  Help
[cisco@localhost cisco]$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/home/cisco/bin
[cisco@localhost cisco]$
```

Here is the path … well not the actual path but the analogy works!

What the heck !!@@##

The Shell and the PATH

```
cisco@localhost:~
File  Edit  View  Terminal  Go  Help
[cisco@localhost cisco]$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/home/cisco/bin
[cisco@localhost cisco]$
```



*These directories are **on** the path*

/home/cisco/bin

/usr/X11R6/bin

/bin

/usr/bin

/usr/local/bin

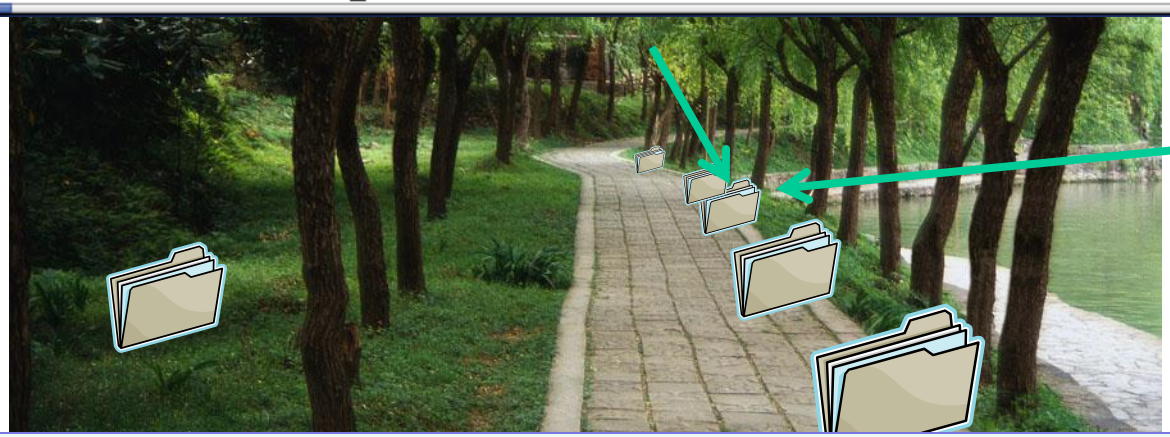
*This directory (and many others) is **NOT** on the path*

/sbin

The Shell and the PATH

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

```
cisco@localhost:/bin
File  Edit  View  Terminal  Go  Help
[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 /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
```

These commands work fine

The Shell and the PATH

cisco@localhost:/sbin

File Edit View Terminal Go Hel

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

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

*The /sbin directory is **NOT** on the path*

*These commands don't work because they were **not found** on the path.*

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

~~!!@@##~~

~~!!@@##~~

OK, makes sense now

Class Exercise

Life of the Shell

1. Use echo $PATH to see your path

2. Use a **hostname** command and a **type hostname** command.
   What happened?

3. use a **iptables –L** command and a **type iptables** command.
   What happened?

Meta-characters

Metacharacters

<cr> (carriage return)

The unprintable carriage return `<cr>` marks the end of a command and lets the shell know to start processing it.

```
[rsimms@opus ~]$ ps
  PID TTY          TIME CMD
19015 pts/0    00:00:00 bash
19378 pts/0    00:00:00 ps

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

[rsimms@opus ~]$ echo "Use <cr> to end the command"
Use <cr> to end the command
```

Pressing the Enter key here generates an invisible `<cr>`

Metacharacters
$ (the value of)

Use `$` for the "value" of a variable

Analogy: Each variable is a named location. The contents of any location is the "value" of that variable.

```
$ echo $LOGNAME
simmsben

$ echo HOME
HOME

$ echo $HOME
/home/cis90/simmsben

$ echo $SHELL
/bin/bash

$ echo $HOSTNAME
opus.cabrillo.edu
```

Metacharacters

' " (single and double quotes)

- *One or more blanks between arguments is treated as a single blank*
- *Use " (double) or ' (single) quotes for preserving blanks*

```
[rsimms@opus ~]$ echo 1 2 3
1 2 3

[rsimms@opus ~]$ echo 1        2                 3
1 2 3

[rsimms@opus ~]$ echo "1        2                 3"
1        2                 3

[rsimms@opus ~]$ echo "1 2 3"
1 2 3

[rsimms@opus ~]$ echo '"1 2 3"'
"1 2 3"
```

The blanks in these commands are used to separate the arguments

Use quotes if blanks are important and are needed for spacing

Use single and double quotes together to show quote marks

Metacharacters

' " (quotes)

Note that strings in " (double) quotes allow the $ metacharacter to be interpreted by the shell

```
[simmsben@opus Poems]$ echo Hello   $LOGNAME
Hello simmsben

[simmsben@opus Poems]$ echo "Hello   $LOGNAME"
Hello   simmsben

[simmsben@opus Poems]$ echo 'Hello   $LOGNAME'
Hello   $LOGNAME
```

Not so with strings in ' (single) quotes

The use of a single quote will prevent the shell from interpreting the $ metacharacter

Metacharacters

\ (don't interpret next metacharacter)

Use \ (back slash) to not interpret the next metacharacter

```
[rsimms@opus ~]$ echo a b c
a b c

[rsimms@opus ~]$ echo a b c \
> d e f
a b c d e f

[rsimms@opus ~]$ echo $PS1
[\u@\h \W]\$

[rsimms@opus ~]$ echo \$PS1
$PS1

[rsimms@opus ~]$ echo "Hello World"
Hello World

[rsimms@opus ~]$ echo \"Hello World\"
"Hello World"
```

Do not interpret the invisible <cr> at the end of the line (from the Enter key)

Do not interpret the $ (which shows the value of the variable)

Do not interpret the double quote marks

Metacharacters

; (command separator)

Use ; to put multiple commands on one line

```
[simmsben@opus Poems]$ hostname; uname; echo $LOGNAME; ls
opus.cabrillo.edu
Linux
simmsben
ant  Blake  nursery  Shakespeare  twister  Yeats
```

More on the Command Line

Handy Shortcuts

- Use up and down arrows to "retype" previous commands
- Left and right arrow for editing current command
- Use <tab> to complete filenames automatically

```
[simmsben@opus Poems]$ hostname; name; echo $LOGNAME; ls Blake/
opus.cabrillo.edu
bash: name: command not found
simmsben
jerusalem  tiger

[simmsben@opus Poems]$ hostname; uname; echo $LOGNAME; ls
Blake/
opus.cabrillo.edu
Linux
simmsben
jerusalem  tiger
```

Press <tab> after the B and the shell fills in the remaining "lake/"

Press up arrow and the shell retypes the previous command

Use the left arrow to backup and fix the typo (uname instead of name)

Docs

Using man (manual) pages

*Type the **man** command followed by the name of the command you want documentation on.*

Example: **man bc**

```
simmsben@opus:~
/home/cis90/simmsben $
/home/cis90/simmsben $ man bc
bc(1)                                                                   bc(1)

NAME
       bc - An arbitrary precision calculator language

SYNTAX
       bc [ -hlwsqv ] [long-options] [  file ... ]

VERSION
       This man page documents GNU bc version 1.06.

DESCRIPTION
       bc  is a language that supports arbitrary precision numbers with inter-
       active execution of statements.  There are  some  similarities  in  the
       syntax  to  the  C  programming  language.   A standard math library is
       available by command line option.  If requested, the  math  library  is
       defined before processing any files.  bc starts by processing code from
       all the files listed on the command line in the  order  listed.   After
       all  files  have been processed, bc reads from the standard input.  All
       code is executed as it is read.  (If a file contains a command to  halt
       the processor, bc will never read from the standard input.)
```

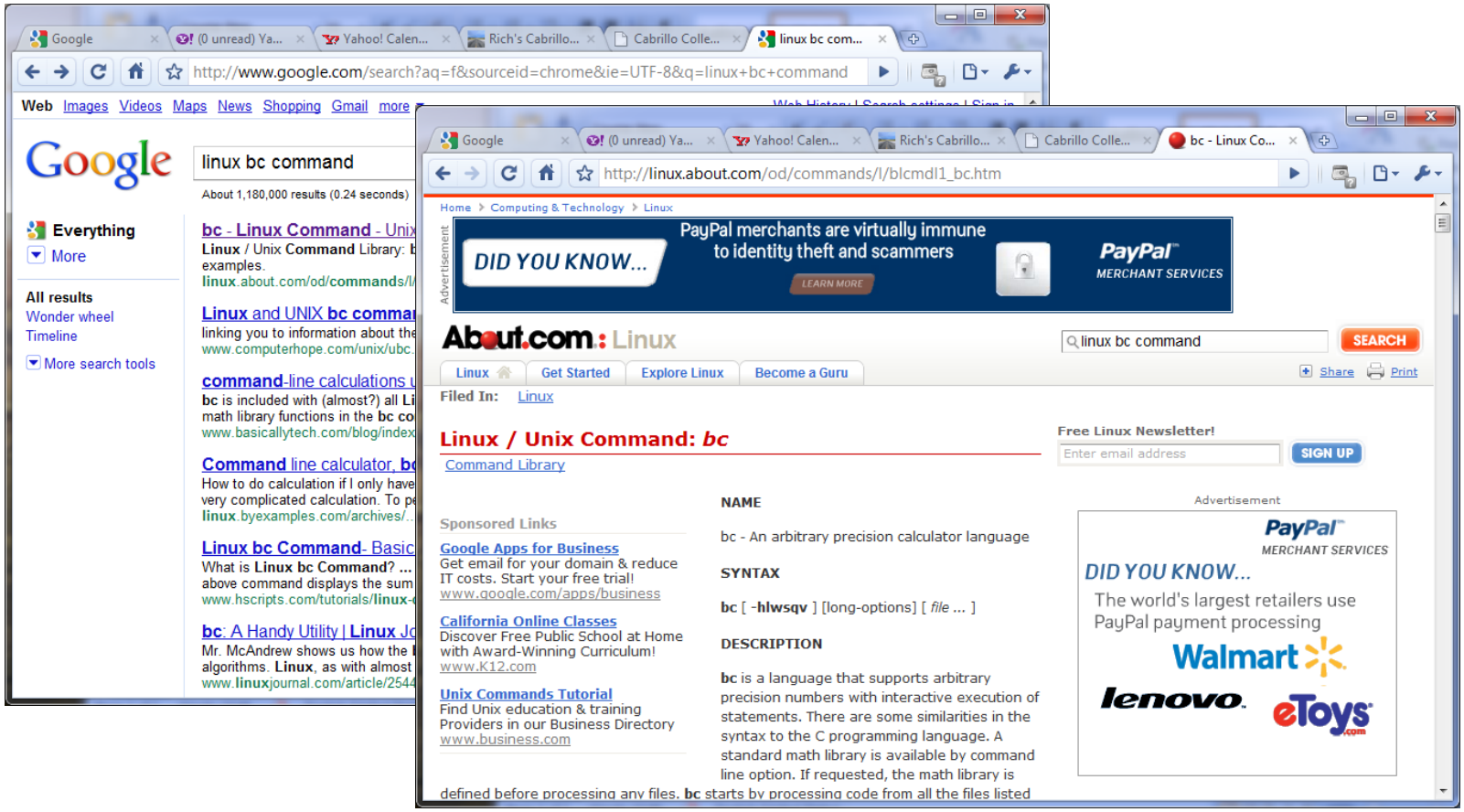
Use these keys to scroll

Use q key to quit

Using Google

Do a Google search on "linux xxx command" where xxx is the command you want documentation for.

Example: google linux bc command



Other Documentation

- **whatis** *command* *same as the **man –f** command*
- **apropos** *command* *same as the **man –k** command*
- **info** *command*

Documentation examples

Example: **whatis ls**

```
simmsben@opus:~
/home/cis90/simmsben $ whatis ls
ls                    (1)   - list directory contents
ls                    (1p)   - list directory contents
/home/cis90/simmsben $
```

whatis *searches the whatis database for a complete word. Same as the* ***man -f*** *command .*

Documentation examples

Example: **apropos kernel**

```
simmsben@opus:~
/home/cis90/simmsben $ apropos kernel
/proc/slabinfo [slabinfo] (5)  - Kernel slab allocator statistics
IPPROTO_ICMP [icmp]  (7)  - Linux IPv4 ICMP kernel module
add_key              (2)  - Add a key to the kernel's key management facility
adjtimex             (2)  - tune kernel clock
arp                  (7)  - Linux ARP kernel module
audit               (rpm) - User space tools for 2.6 kernel auditing
auditctl             (8)  - a utility to assist controlling the kernel's audit s
ystem
bootparam            (7)  - Introduction to boot time parameters of the Linux ke
rnel
curs_set [curs_kernel] (3x)  - low-level curses routines
def_prog_mode [curs_kernel] (3x)  - low-level curses routines
def_shell_mode [curs_kernel] (3x)  - low-level curses routines
dmesg                (8)  - print or control the kernel ring buffer
elksemu              (1)  - Embedded Linux Kernel Subset emulator
exports              (5)  - NFS file systems being exported (for Kernel based NF
S)
get_kernel_syms      (2)  - retrieve exported kernel and module symbols
getkeycodes          (8)  - print kernel scancode-to-keycode mapping table
getkeycreatecon      (3)  - get or set the SELinux security context used for cre
ating a new kernel keyrings
getsyx [curs_kernel] (3x)  - low-level curses routines
glGetConvolutionFilter (3gl)  - get current 1D or 2D convolution filter kernel
```

apropos *searches the whatis database for a string of text. Same as the* ***man -k*** *command .*

Documentation examples

Example: **info ls**

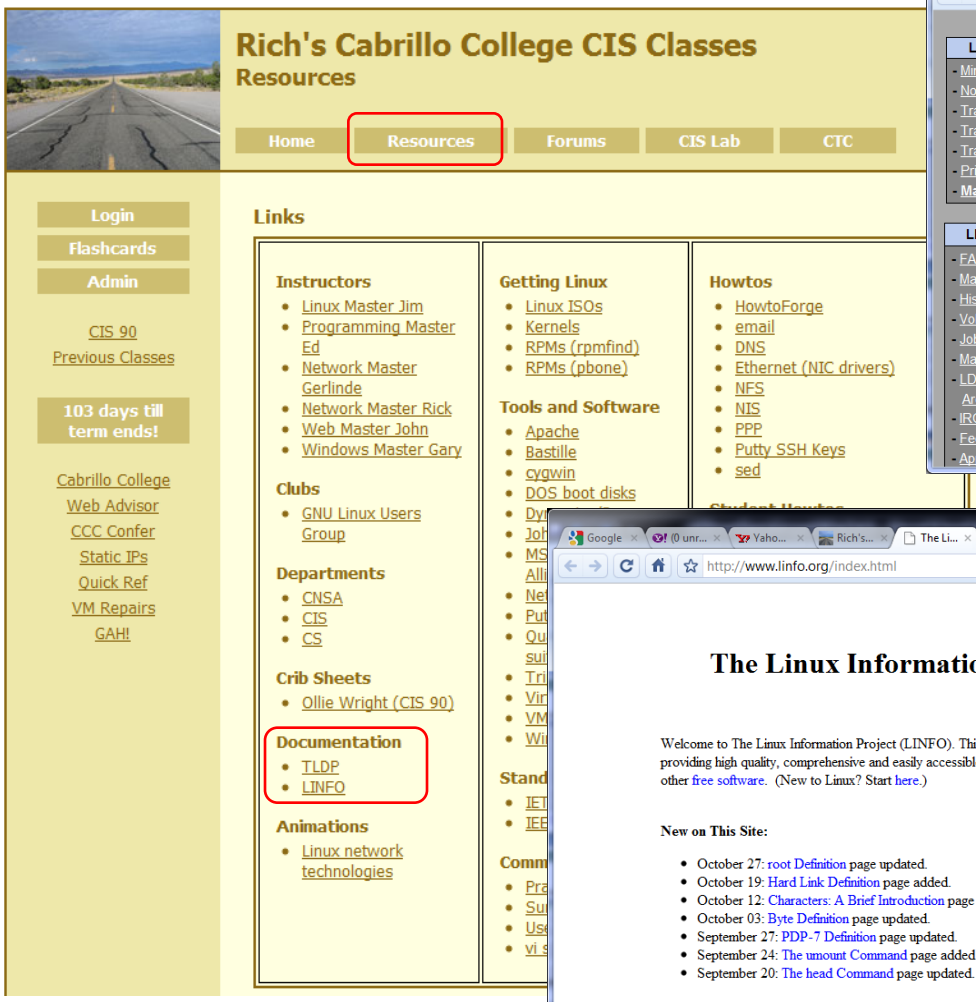
```
simmsben@opus:~
File: coreutils.info,  Node: ls invocation,  Next: dir invocation,  Up: Directo\
ry listing

10.1 `ls': List directory contents
==================================

The `ls' program lists information about files (of any type, including
directories).  Options and file arguments can be intermixed
arbitrarily, as usual.

   For non-option command-line arguments that are direc
default `ls' lists the contents of directories, not rec
omitting files with names beginning with `.'.  For othe
arguments, by default `ls' lists just the file name.  I
argument is specified, `ls' operates on the current dir
as if it had been invoked with a single argument of `.'

   By default, the output is sorted alphabetically, acc
locale settings in effect.(1) If standard output is a t
output is in columns (sorted vertically) and control ch
output as question marks; otherwise, the output is list
and control characters are output as-is.
--zz-Info: (coreutils.info.gz)ls invocation, 54 lines --Top----------------------
Welcome to Info version 4.8. Type ? for help, m for menu item.
```

Navigating info pages:

- Enter to follow links (*'s)
- n or <space> for next page
- p for previous page
- u for up tree
- l for last page
- q to quit

Documentation

Two of my favorite documentation links



The Linux Documentation and Information Projects

Class Exercise
Documentation

Use the man command on itself:
- **man man**

Research the **ls** command using:
- The **whatis** command
- The **man** command
- The **info** command
- Google

Wrap up

New commands:

apropos	- search for string in whatis database
bc	- binary calculator
cat	- print file(s)
cd	- change directory
echo	- print text
env	- show shell environment variables
info	- online documentation with hot links
file	- show file information
ls	- show directory contents
passwd	- change password
set	- show (or set) shell variables
type	- show command location in path
man	- manual page for a command
whatis	- command summary

New Files and Directories:

/etc/passwd	- user accounts
/etc/shadow	- encrypted passwords
/bin	- directory of commands
/sbin	- directory of superuser commands
/usr/bin	- directory of commands, tools and utilities
/usr/sbin	- directory of superuser commands, tools and utilities

Next Class

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

Lab #2

Quiz questions for next class:

- Name four directories where one can find commands?
- How do you show your path?
- What is the command to print the manual page for a command?

Backup