Lesson Module Checklist
- Slides –
- Properties –
- Flash cards –
- First minute quiz –
- Web calendar summary –
- Web book pages –
- Commands –

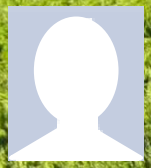
- Lab tested and uploaded –
- Tech file email for Lab 9 ready -
- CCC Confer wall paper made –

- Materials uploaded –
- Backup slides, CCC info, handouts on flash drive –
- Check that backup room headset is charged –
- Spare 9v battery for mic

CIS 90 - Lesson 11

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


Sean C.

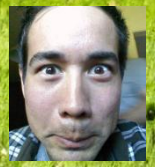

Donald

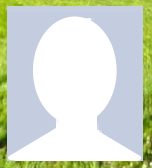

Carlile


Andrew


Sean Fa.


Carter


Sean Fy.


Dajan


Bryn

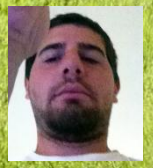

Rita

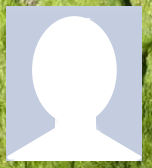

Kelly

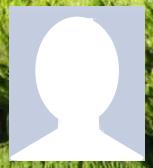

Ben


Ray


Fidel

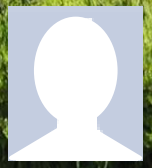

Michael

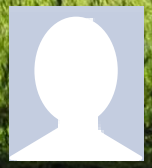

Evan


Josh


Carlos


Gustavo


Jessica


Evie

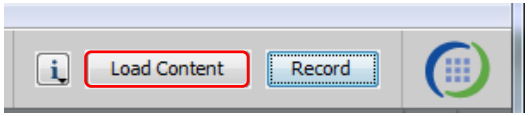

Jacob


Humberto

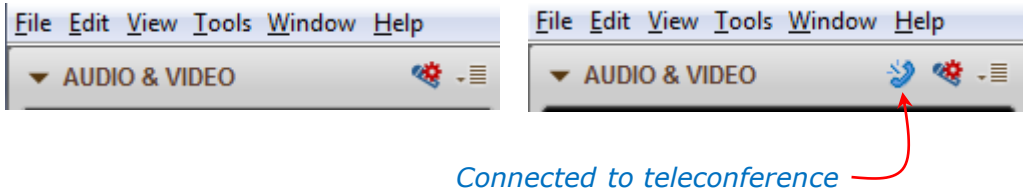

Chad

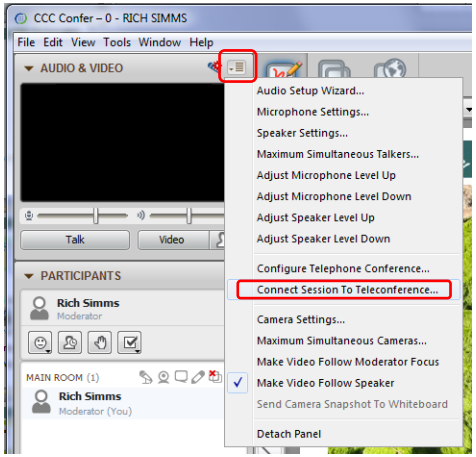
Email me (risimms@cabrillo.edu) a relatively current photo of your face for 3 points extra credit

[ ] Load White Board with *cis*lesson??*-WB*

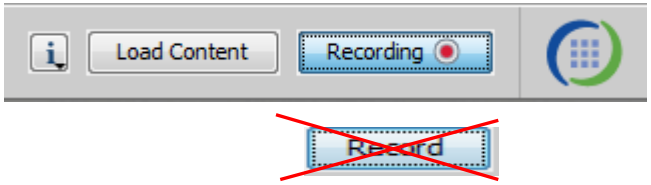


[ ] Connect session to Teleconference





[ ] Is recording on?

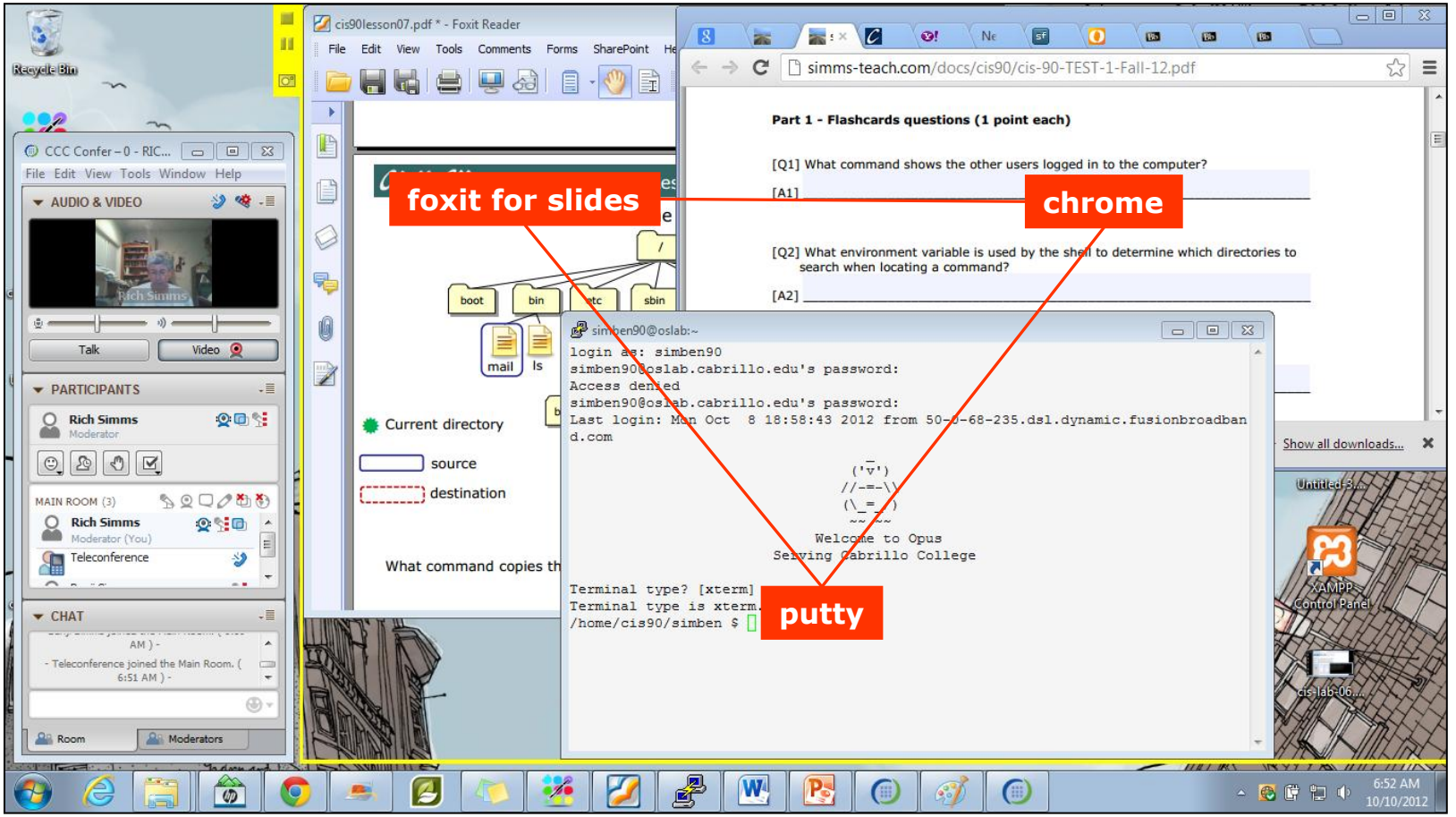


[ ] Toggle Talk button to not use Mic

CCC Confer

[ ] Video (webcam) optional

[ ] layout and share apps



Quiz

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

See electronic white board

email answers to: risimms@cabrillo.edu

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

vi editor

Objectives	Agenda
• Create and modify text files	• Quiz • Questions from last week • more on grep • Review on processes • vi • Wrap up

Questions?

- Test 2?
- Lab 8?
- Previous course material?

Housekeeping

Previous material and assignment

1. Questions?

2. Lab 8 due tonight

Don't wait till midnight tonight to see if this worked! Test with an earlier time.

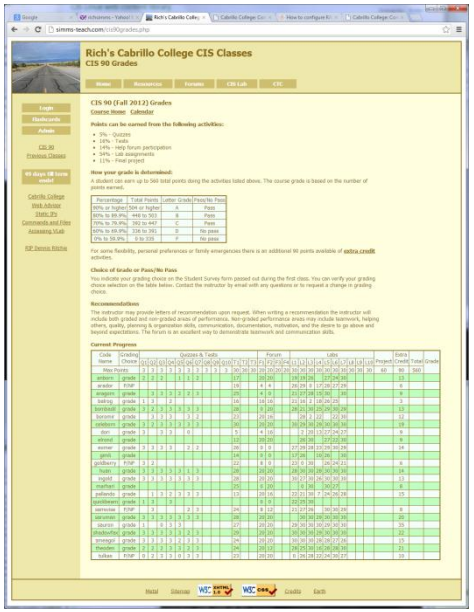
```
at 11:59pm
at> cat files.out bigshell > lab08
at> cp lab08 /home/rsimms/turnin/lab08.$LOGNAME
at> Ctrl-D
```

3. Note: Lab 9 and five posts due next week

4. You can still send me your photo for our class page if you want 3 points extra credit

Managing your grade

Use the web page



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

Use Jesse's checkgrades script

```
anborn: 72% (241 of 331 points)
arador: 59% (198 of 331 points)
aragorn: 67% (225 of 331 points)
balrog: 50% (168 of 331 points)
bombadil: 91% (302 of 331 points)
boromir: 61% (205 of 331 points)
celeborn: 104% (347 of 331 points)
dori: 50% (168 of 331 points)
elrond: 62% (207 of 331 points)
eomer: 83% (275 of 331 points)
gimli: 37% (125 of 331 points)
goldberry: 57% (190 of 331 points)
huan: 109% (364 of 331 points)
ingold: 96% (321 of 331 points)
marhari: 47% (157 of 331 points)
pallando: 74% (248 of 331 points)
quickbeam: 25% (84 of 331 points)
samwise: 70% (234 of 331 points)
saruman: 96% (320 of 331 points)
sauron: 104% (346 of 331 points)
shadowfax: 106% (352 of 331 points)
smeagol: 96% (318 of 331 points)
theoden: 92% (307 of 331 points)
tulkas: 78% (261 of 331 points)
```

As of November 7, 2012

Managing your grade

Percentage	Total Points	Letter Grade	Pass/No Pass
90% or higher	504 or higher	A	Pass
80% to 89.9%	448 to 503	B	Pass
70% to 79.9%	392 to 447	C	Pass
60% to 69.9%	336 to 391	D	No pass
0% to 59.9%	0 to 335	F	No pass

Points gone by
- 7 quizzes - 21 points
- 2 tests - 60 points
- 2 forum periods - 40 points
- 7 labs - 210 points

331 points

Points yet to earn
- 3 quizzes - 9 points
- 1 test - 30 points
- 2 forum periods - 40 points
- 3 labs - 90 points
- 1 final project - 60 points

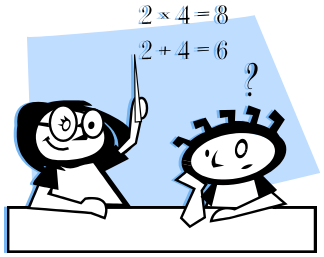
229 points

- Plus extra credit - up to 90 points

Managing your grade

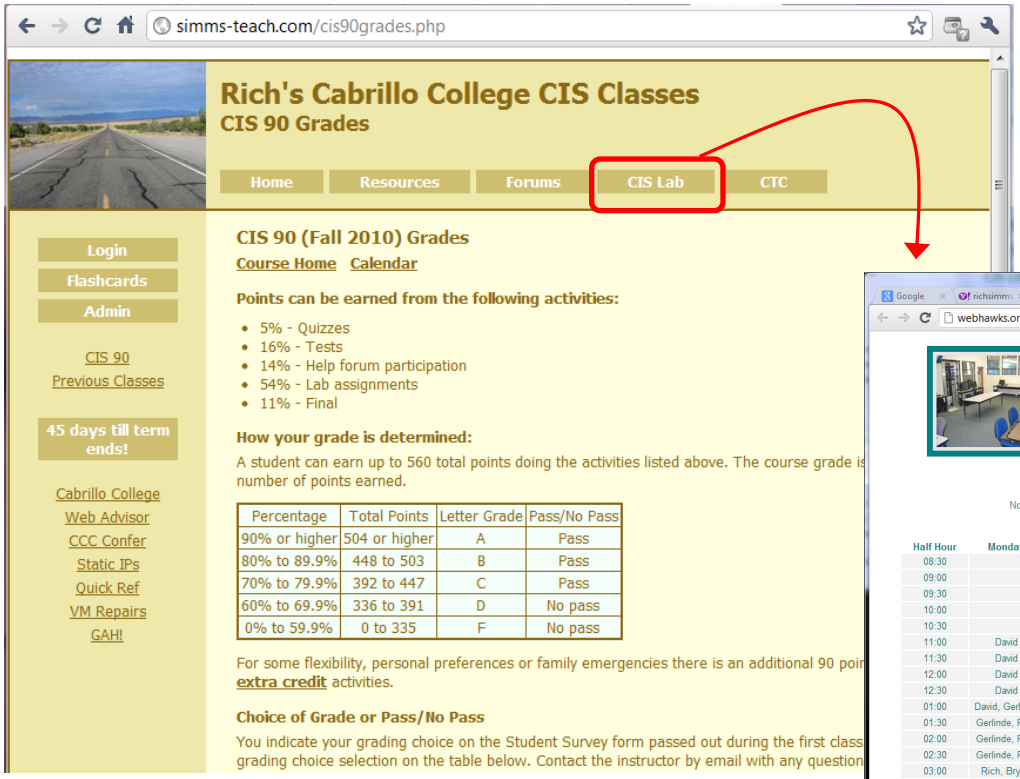
Getting extra help for CIS 90

- Rich's Office Hours Wed 4:20PM-5:10PM in Room 2501 (right after class) or TBA (contact me)
- Ask questions on the Forum at:
  http://opus.cabrillo.edu/forum/



Managing your grade

Getting extra help for CIS 90



Come by the lab and get help from instructors and student assistants

grep workout

Some perfect times to use the **grep** command:

1) To search through the output of a command for some text

```
command | grep "text string"
```

2) To search inside one or more files for some text

```
grep "text string" file1 file2 … filen
```

3) To search (recursively) inside all files in a portion (or all) of the UNIX file tree for some text

```
grep –R "text string" directory
```

grep usage – search output of a command

Is the CUPS daemon (print service) running right now?

```
/home/cis90/simben $ ps -ef | grep cups
root      6251     1  0 Jul31 ?        00:00:04 cupsd -C /etc/cups/cupsd.conf
simben90 27027 26966  0 08:47 pts/3    00:00:00 grep cups
```

Yes it is, with PID=6251

grep practice

- Is the cronjob daemon (crond) running right now?
- Type the crond PID into the chat window

grep usage – search output of a command

Is the Apache web server (httpd) installed?

This shows all installed package names

This searches for package names containing "httpd"

```
/home/cis90/simben $ rpm -qa | grep httpd
httpd-tools-2.2.15-15.el6.centos.1.i686
httpd-2.2.15-15.el6.centos.1.i686
httpd-manual-2.2.15-15.el6.centos.1.noarch
```

Yes, version 2.2.15 has been installed

grep practice

- Has the mysql-server package been installed on Opus?
- If installed on Opus, type the version of mysql in the chat window

grep usage – search output of a command

When were the last 5 times I logged in?

```
/home/cis90/simben $ last | grep $LOGNAME | head -n5
simben90 pts/0        50-0-68-235.dsl. Mon Apr 23 05:39   still logged in
simben90 pts/6        10.64.25.2       Wed Apr 18 12:48 - 16:51  (04:02)
simben90 pts/5        10.64.25.2       Wed Apr 18 12:48 - 16:51  (04:02)
simben90 pts/4        10.64.25.2       Wed Apr 18 12:48 - 16:51  (04:03)
simben90 pts/1        50-0-68-235.dsl. Wed Apr 18 09:06 - 10:23  (01:17)
```

This scans the latest wtmp log file and lists your most recent five logins to Opus

grep practice

- For the time period covered by the current wtmp log file. What was the date of your earliest login?
- Type your earliest login date into the chat window

grep usage – search output of a command

```
[rsimms@oslab ~]$ ls /bin/*sh
/bin/bash  /bin/csh  /bin/dash  /bin/ksh  /bin/rbash  /bin/sh  /bin/tcsh

[rsimms@oslab ~]$ ksh
$ dash
$ sh
sh-4.1$ csh
```

Similar to lab 8. This is how to show which shell uses the most memory when it runs as a process and record that answer in a file

```
[rsimms@oslab ~]$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S   201 27553 27552  0  80   0 -  1308 -      pts/0    00:00:00 bash
0 S   201 27651 27553  0  80   0 -  1376 -      pts/0    00:00:00 ksh
0 S   201 27652 27651  0  80   0 -   517 -      pts/0    00:00:00 dash
0 S   201 27653 27652  0  80   0 -  1307 -      pts/0    00:00:00 sh
0 S   201 27654 27653  0  80   0 -  1458 -      pts/0    00:00:00 csh
0 R   201 27663 27654  0  80   0 -  1214 -      pts/0    00:00:00 ps

[rsimms@oslab ~]$ ps -l | grep csh
0 S   201 27654 27653  0  80   0 -  1458 -      pts/0    00:00:00 csh

[rsimms@oslab ~]$ ps -l | grep csh > bigshell

[rsimms@oslab ~]$ cat bigshell
0 S   201 27654 27653  0  80   0 -  1458 -      pts/0    00:00:00 csh
```

grep practice

- For the bash, dash, ksh, sh and csh shells, which shell process uses the least memory?
- What command that would redirect the line of output for the command using the least amount of memory to the file *smallshell*
- Type the command you use above into the chat window

grep usage – search inside files

How many CIS 90 user accounts are there?

```
/home/cis90/simben $ grep cis90 /etc/passwd | wc -l
29
```

There are 29

grep practice

- How many CIS 172 accounts are there on Opus?
- Type the number of CIS 172 accounts into the chat window

grep usage – search inside files

Example: What is my account information in /etc/passwd?

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

or

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

or

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

grep practice

- Does your user ID in */etc/passwd* match the user ID you shown by the **id** command?
- Type your UID into the chat window

grep usage – search inside files in all or part of the file tree

Where does the PS1 "prompt" variable get set?

```
/home/cis90/simben $ grep -R "PS1=" /etc/bash* $HOME 2> /dev/null
/etc/bash_completion.d/git:#        PS1='[\u@\h \W$(__git_ps1 "
(%s)")]\$ '
/etc/bashrc:  [ "$PS1" = "\\s-\\v\\\$ " ] && PS1="[\u@\h \W]\\$ "
/etc/bashrc:  #   PS1="[\u@\h:\l \W]\\$ "
/home/cis90/simben/class/labs/lab04.graded:21) PS1='$PWD $ '
/home/cis90/simben/class/exams/test01.graded:(A32) PS1='\d $ '
/home/cis90/simben/.bash_profile:PS1='$PWD $ '
/home/cis90/simben/lab04.graded:21) PS1='$PWD $ '
/home/cis90/simben/test01.graded:(A32) PS1='\d $ '
```

It is set more than once during login. We will learn in a future lesson that the one in .bash_profile is done last and is what you end up using.

grep practice

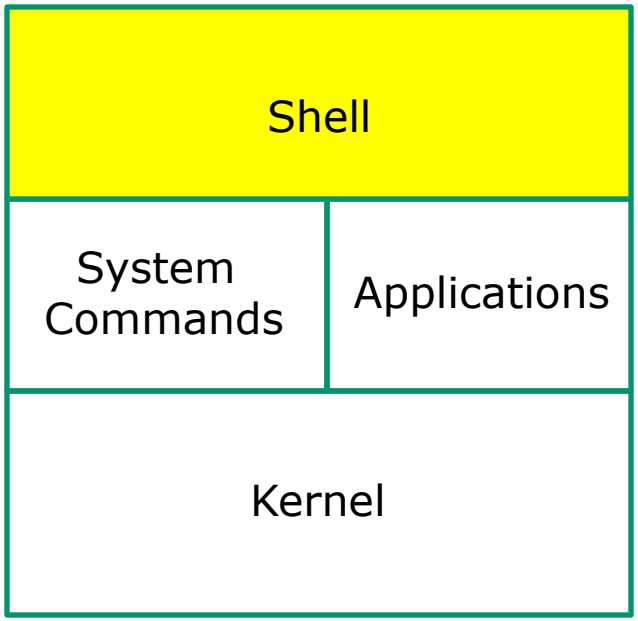
- Find the file in the /usr/lib portion of the file tree that contains "hot pototo dance" (yes, potato is misspelled).
- Type the absolute pathname of the file in the chat window.

Shell six steps (REVIEW)

```
/home/cis90/simben $ find / -name treat* 2> /dev/null
/home/cis90/calsea/treat1
/home/cis90/rawjes/treat1
/home/cis90/ellcar/treat1
/home/cis90/cis/treat1
/home/cis90/davdon/bag/treat1
/home/cis90/davdon/treat1
/home/cis90/noreva/bag/treat1
/home/cis90/noreva/treat1
/home/cis90/libkel/treat1
/home/cis90/rodduk/treat1
/home/cis90/frocar/bag/treat1
/home/cis90/frocar/treat1
/home/cis90/verevi/treat1
 < snipped >
/home/cis90/fyosea/treat1
/home/cis90/ramgus/treat1
/home/cis90/potjos/treat1
/home/cis90/simben/treat1
/home/cis90/wiljac/treat1
/home/cis90/hendaj/treat1
/home/cis90/lyoben/treat1
/home/cis90/marray/bag/mastreats/treat1
/home/cis90/marray/bag/treat1
/home/cis90/marray/treat1
/home/cis90/simben
```

On the next slides we will walk through each of the six steps the shell takes for this command

Prompt Step



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

Prompt Step

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

shell prompt

Benji is using the bash shell which prompts him for a command.

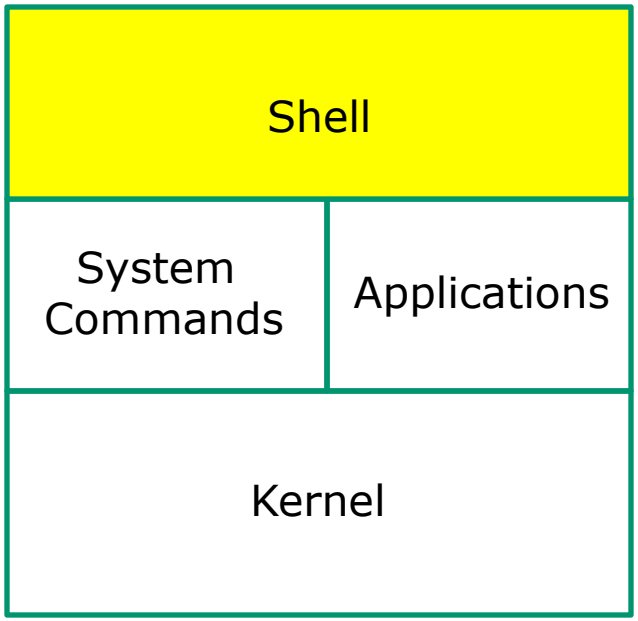
The command prompt is defined by the value of the PS1 variable.

Prompt Step

```
/home/cis90/simben $ find / -name treat* 2> /dev/null
```

Benji types in this find command in response to the shell prompt

Parse Step



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

Parse Step

The shell uses spaces to separate options, arguments and redirection

```
find / -name treat* 2> /dev/null
```

The shell must expand filename expansion characters and variables during the parse step. Note there is an invisible <newline> metacharacter at the end of the command

Parsing RESULTS:

Command: **find**

Options and arguments:
- **/**
- **-name**
- **treat1**

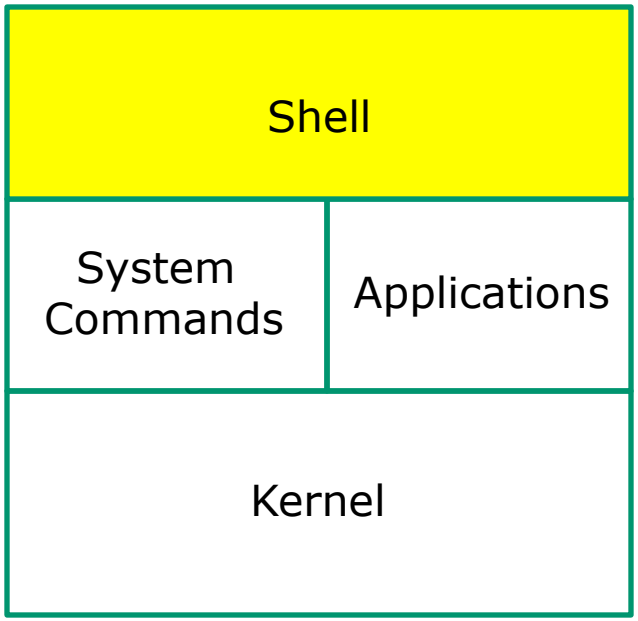
This will be passed to the command (if the command can be located on the path)

Redirection:
Connect **stderr** to **/dev/null** (the "bit bucket")

This will be handled by the shell. The command, if loaded, will not see this

Note: Because Benji had a treat1 file in his home directory, the shell expands treat to treat1*

Search Step



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

Search Step

Command: **find**

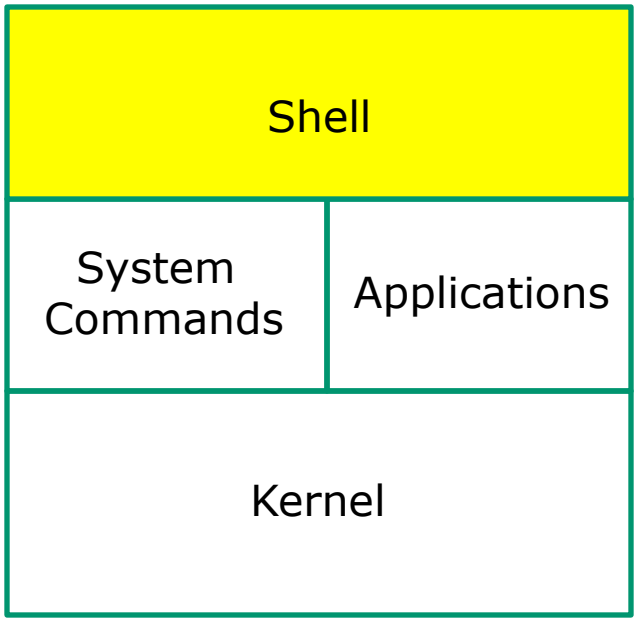
*The shell now must search, in order, every directory on Benji's path to locate the first occurrence of the **find** command.*

Benji's path is defined by the value of his PATH variable

1st directory searched: /usr/lib/qt-3.3/bin
2nd directory searched: /usr/local/bin
3rd directory searched: **/bin**
4th directory searched: /usr/bin
5th directory searched: /usr/local/sbin
6th directory searched: /usr/sbin
7th directory searched: /sbin
8th directory searched: /home/cis90/simben/../bin
9th directory searched: /home/cis90/simben/bin
10th directory searched: .

The shell locates the find command in the /bin directory

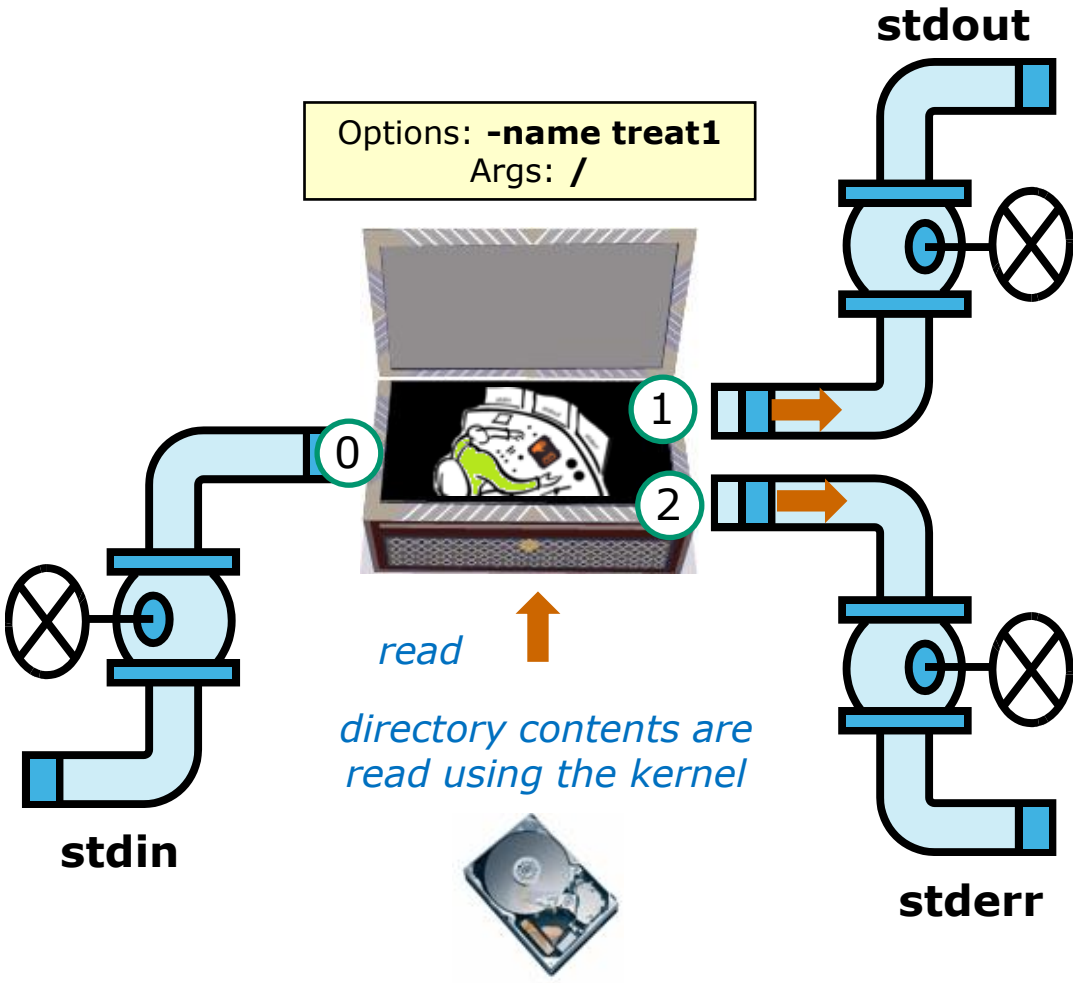
Execute Step



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

Execute Step

```
/home/cis90/simben $ find / -name treat* 2> /dev/null
```




```
/home/cis90/calsea/treat1
/home/cis90/rawjes/treat1
/home/cis90/ellcar/treat1
/home/cis90/cis/treat1
/home/cis90/davdon/bag/treat1
/home/cis90/davdon/treat1
/home/cis90/noreva/bag/treat1
< snipped >
```

```
find: `/lost+found': Permission denied
find: `/var/empty/sshd': Permission denied
find: `/var/log/sssd': Permission denied
< snipped >
```

This is what the find process might look like

A **process:**

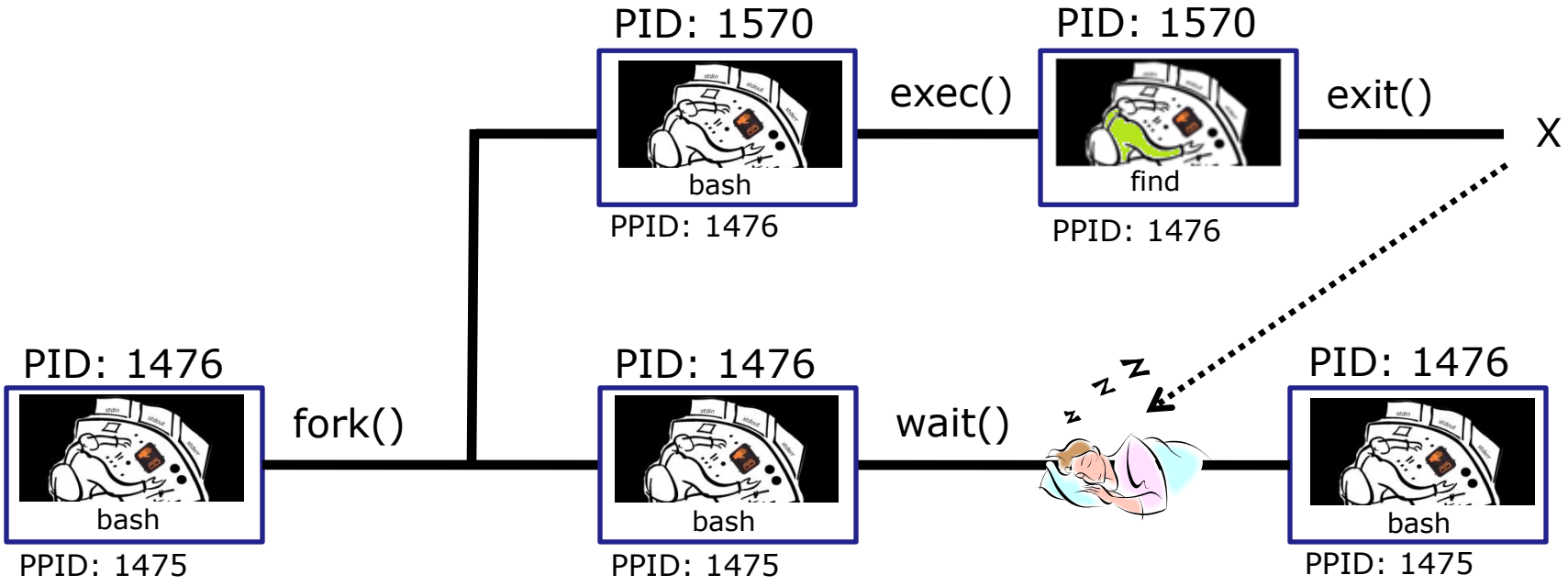
- Is provided with parsed/expanded options and arguments from the shell

- may read from **stdin**
- may write to **stdout**
- may write error messages to **stderr**

- and may get interrupted from time to time by a **signal**

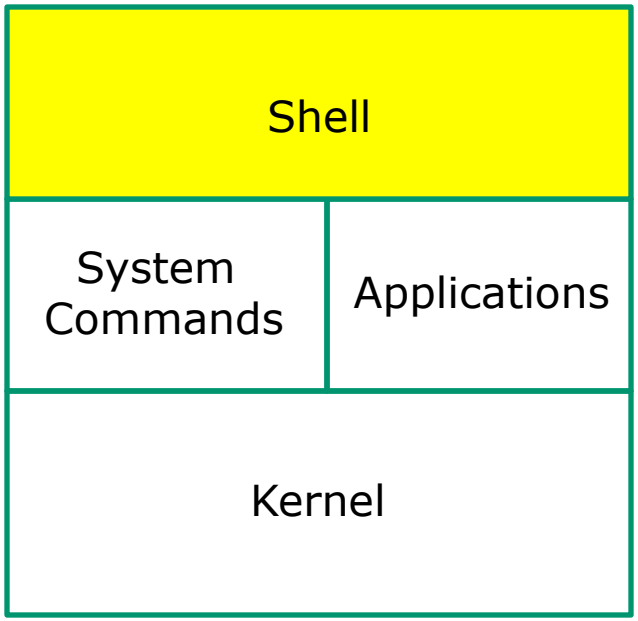
The find process uses the options and arguments passed to it by the shell to perform the find operation

Execute Step



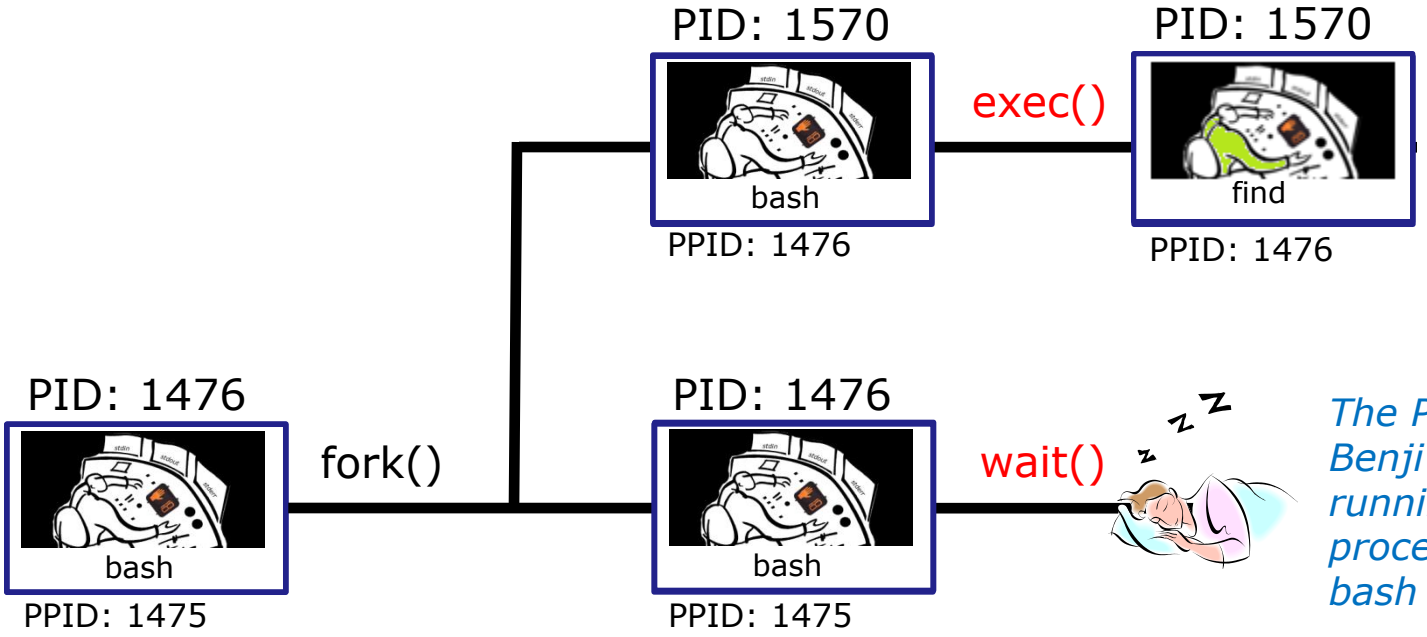
*bash executes the find command by cloning itself with a **fork()** system call to create a new child process. With an **exec()** system call, the new child process is overlaid with the find code instructions. bash sleeps by making a **wait()** system call while the find child process runs. The child process makes an **exit()** system call when it has finished. After that, the parent bash process wakes up and the child process is killed.*

Nap Step



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

Nap Step

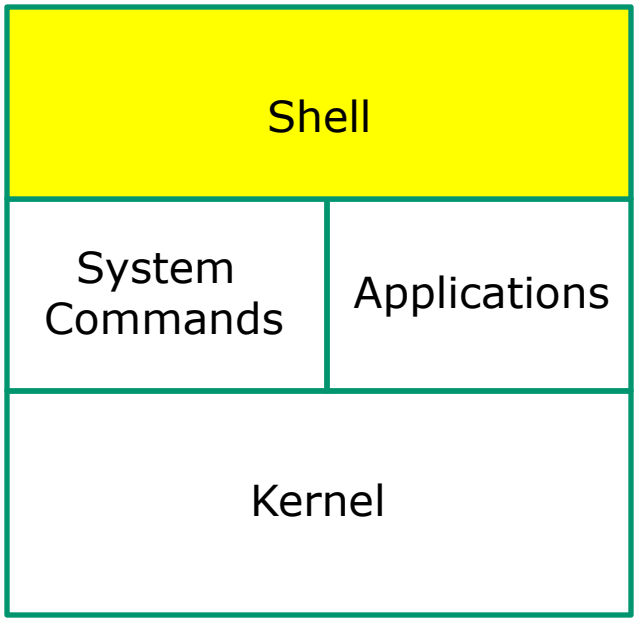


*The PS command shows Benji's **find** command is running as a child process while the parent bash shell sleeps*

```
[rsimms@oslab ~]$ ps -l -u simben90
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
5 S  1001  1475  1470  0  80   0 -  3392 ?      ?        00:00:00 sshd
0 S  1001  1476  1475  0  80   0 -  1308 ?      pts/1    00:00:00 bash
0 R  1001  1570  1476 40  80   0 -  1179 ?      pts/1    00:00:00 find
```

R=Running, S=Sleeping

Repeat Step

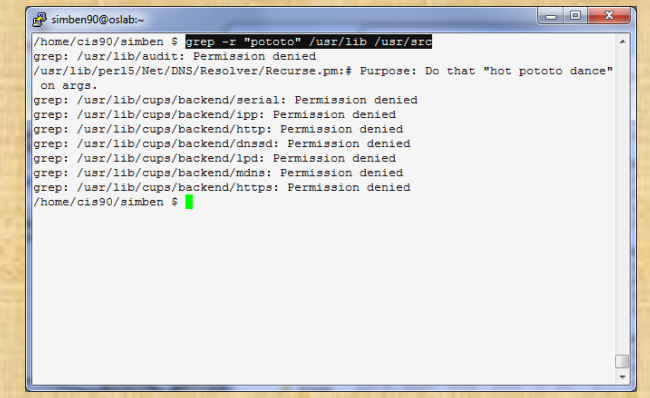


1) Prompt
2) Parse
3) Search
4) Execute
5) Nap
6) **Repeat**

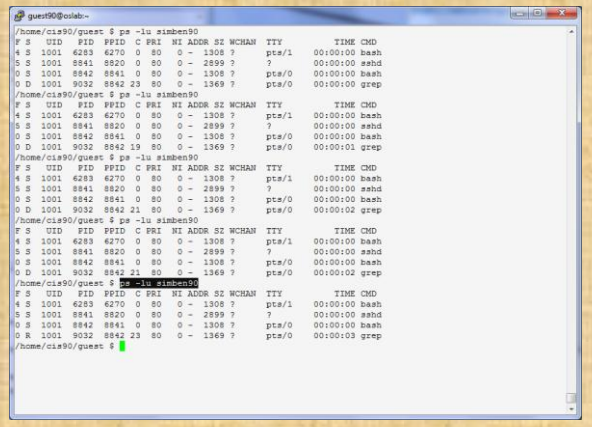
Process activity

- See if you can do a **ps** command that illustrates what happens when a user runs a long **grep** command.
- The **ps** output should show "parent" bash S=Sleeping while the "child" **grep** command is either R=Running or in D=Uninterruptible sleep (IO)
- Use the guest90 account to observe your processes
- Write your grep PID and status into the chat window when done

```
/home/cis90/simben $ grep -r "pototo" /usr/lib /usr/src
```




```
/home/cis90/guest $ ps -lu simben90
```



Review of Signals

Signals

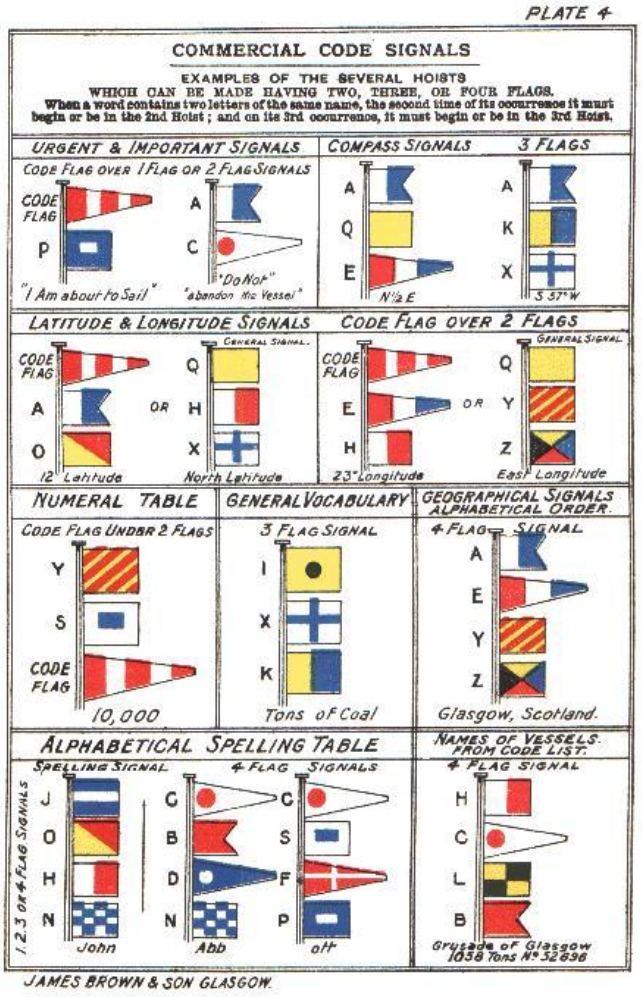


PLATE 4

COMMERCIAL CODE SIGNALS

EXAMPLES OF THE SEVERAL HOISTS WHICH CAN BE MADE HAVING TWO, THREE, OR FOUR FLAGS.
When a word contains two letters of the same name, the second time of its occurrence it must begin or be in the 2nd Hoist; and on its 3rd occurrence, it must begin or be in the 3rd Hoist.

URGENT & IMPORTANT SIGNALS — CODE FLAG OVER 1 FLAG OR 2 FLAG SIGNALS
CODE FLAG P — "I Am about to Sail"
A — C — "Do Not" "abandon the Vessel"

COMPASS SIGNALS — 3 FLAGS
A Q E — N½E
A K X — S 57°W

LATITUDE & LONGITUDE SIGNALS — CODE FLAG OVER 2 FLAGS
CODE FLAG A O — 12° Latitude
OR Q H X — North Latitude
CODE FLAG E H — 23° Longitude
OR Q Y Z — East Longitude

NUMERAL TABLE — CODE FLAG UNDER 2 FLAGS
Y S CODE FLAG — 10,000

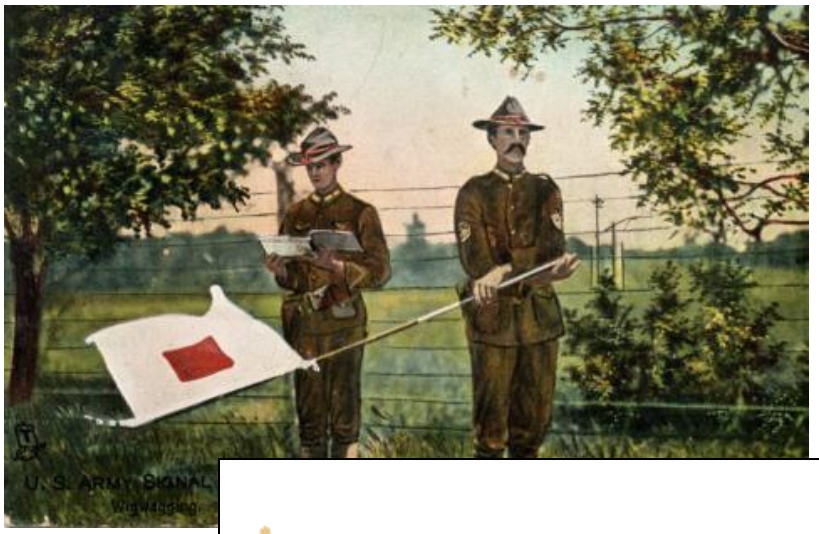
GENERAL VOCABULARY — 3 FLAG SIGNAL
I X K — Tons of Coal

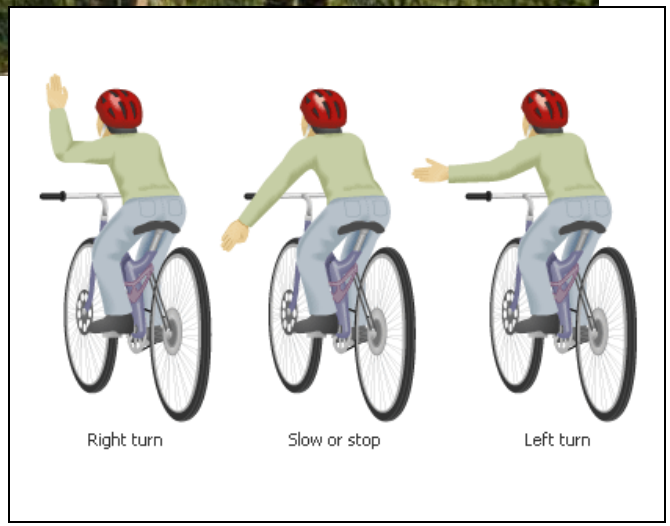
GEOGRAPHICAL SIGNALS ALPHABETICAL ORDER. — 4 FLAG SIGNAL
A E Y Z — Glasgow, Scotland.

ALPHABETICAL SPELLING TABLE — SPELLING SIGNAL — 4 FLAG SIGNALS
1.2.3 OR 4 FLAG SIGNALS
J O H N — John
C B D N — Abb
C S F P — off

NAMES OF VESSELS FROM CODE LIST. — 4 FLAG SIGNAL
H C L B — Gryzade of Glasgow 1058 Tons N° 32896

JAMES BROWN & SON GLASGOW.





This is what a process might look like

A **process:**

- Is provided with parsed/expanded options and arguments from the shell
- may read from **stdin**
- may write to **stdout**
- may write error messages to **stderr**
- and may get interrupted from time to time by a **signal**

*A **process** is a **program** that has been loaded into memory and is either running (executing instructions) or waiting to run*

Signals

The result of sending a signal to a process:

- be ignored
- default action (die)
- execute some predefined function

Signals

SIGHUP	1	Hangup (POSIX)	
SIGINT	2	Terminal interrupt (ANSI)	**Ctrl-C**
SIGQUIT	3	Terminal quit (POSIX)	**Ctrl-**
SIGILL	4	Illegal instruction (ANSI)	
SIGTRAP	5	Trace trap (POSIX)	
SIGIOT	6	IOT Trap (4.2 BSD)	
SIGBUS	7	BUS error (4.2 BSD)	
SIGFPE	8	Floating point exception (ANSI)	
SIGKILL	9	Kill (can't be caught or ignored) (POSIX)	
SIGUSR1	10	User defined signal 1 (POSIX)	
SIGSEGV	11	Invalid memory segment access (ANSI)	
SIGUSR2	12	User defined signal 2 (POSIX)	
SIGPIPE	13	Write on a pipe with no reader, Broken pipe (POSIX)	
SIGALRM	14	Alarm clock (POSIX)	
SIGTERM	15	Termination (ANSI)	

Use kill –l to see all signals

Signals

SIGSTKFLT	16	Stack fault
SIGCHLD	17	Child process has stopped or exited, changed (POSIX)
SIGCONT	18	Continue executing, if stopped (POSIX)
SIGSTOP	19	Stop executing(can't be caught or ignored) (POSIX)
SIGTSTP	20	Terminal stop signal (POSIX) **Ctrl-Z or Ctrl-F**
SIGTTIN	21	Background process trying to read, from TTY (POSIX)
SIGTTOU	22	Background process trying to write, to TTY (POSIX)
SIGURG	23	Urgent condition on socket (4.2 BSD)
SIGXCPU	24	CPU limit exceeded (4.2 BSD)
SIGXFSZ	25	File size limit exceeded (4.2 BSD)
SIGVTALRM	26	Virtual alarm clock (4.2 BSD)
SIGPROF	27	Profiling alarm clock (4.2 BSD)
SIGWINCH	28	Window size change (4.3 BSD, Sun)
SIGIO	29	I/O now possible (4.2 BSD)
SIGPWR	30	Power failure restart (System V)

Use kill –l to see all signals

Signals

Signals are asynchronous messages sent to processes

They can result in one of three courses of action:

1. be ignored,
2. default action (die)
3. execute some predefined function.

Signals are sent:

kill command

Using the kill command: **$ kill -# PID**

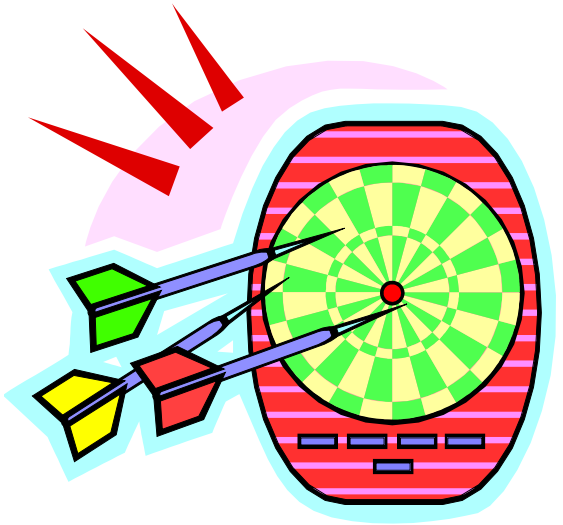
- Where # is the signal number and PID is the process id.
- if no number is specified, SIGTERM (-15) is sent.

Using special keystrokes

- limited to just a few signals
- limited to when you have control of the keyboard

Use kill –l to see all signals

Target Practice



Activity

1) Run the **annoy** program

2) Try sending it a SIGINT with **Ctrl-C**

3) Try sending it a SIGQUIT with **Ctrl-**

4) Bring up another terminal and try signals 1 through 64
   - Use **ps –u $LOGNAME** to find the **annoy** *PID*

   - Try **kill -1 *PID***
   - Try **kill -2 *PID***
   - Try **kill -3 *PID***
   - *and so forth …*

   OR

   - Try **killall -1 annoy**
   - Try **killall -2 annoy**
   - Try **killall -3 annoy**
   - *and so forth …*

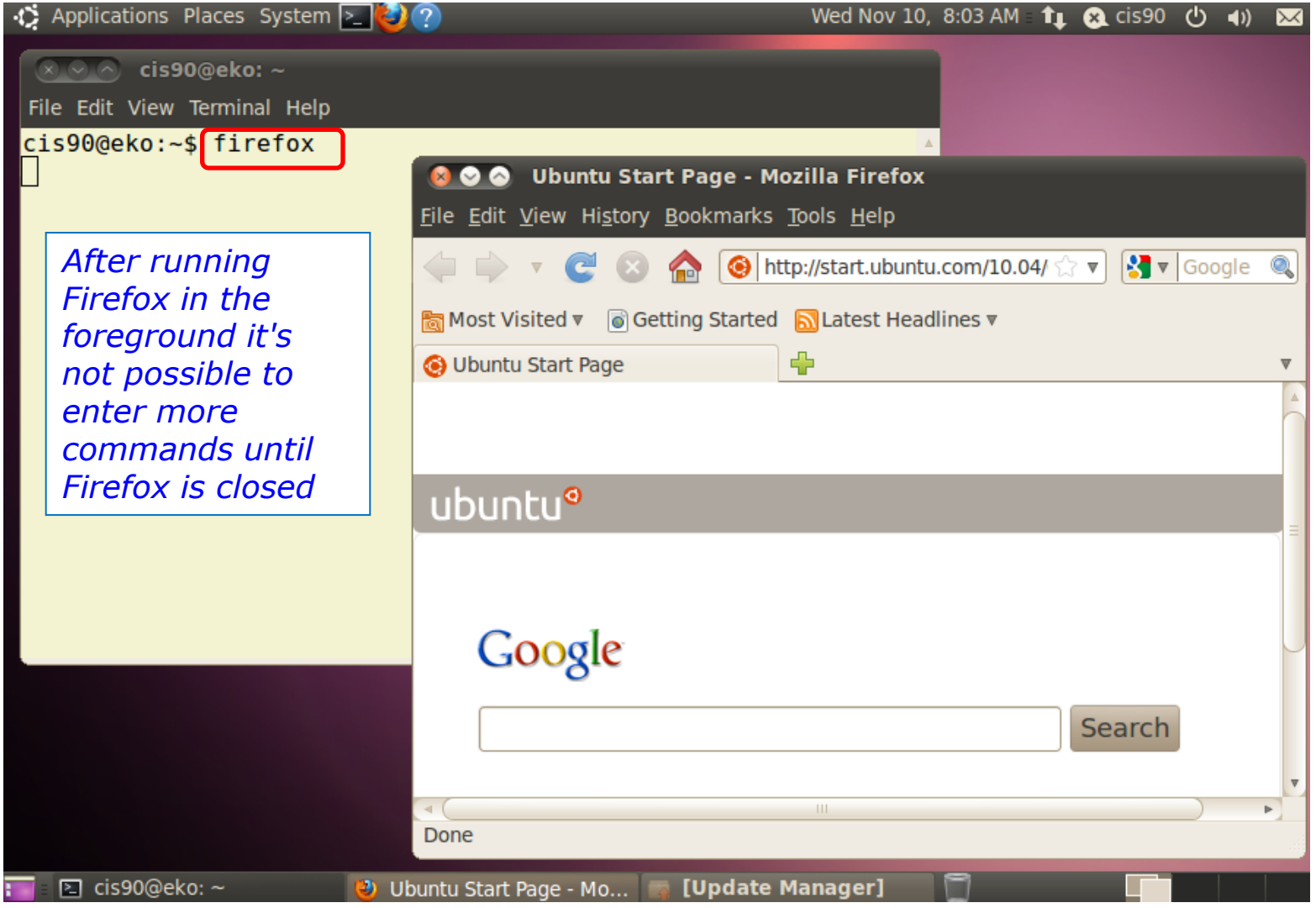
5) Write the signals that kill **annoy** into the chat window

Using &

to run a command in the background

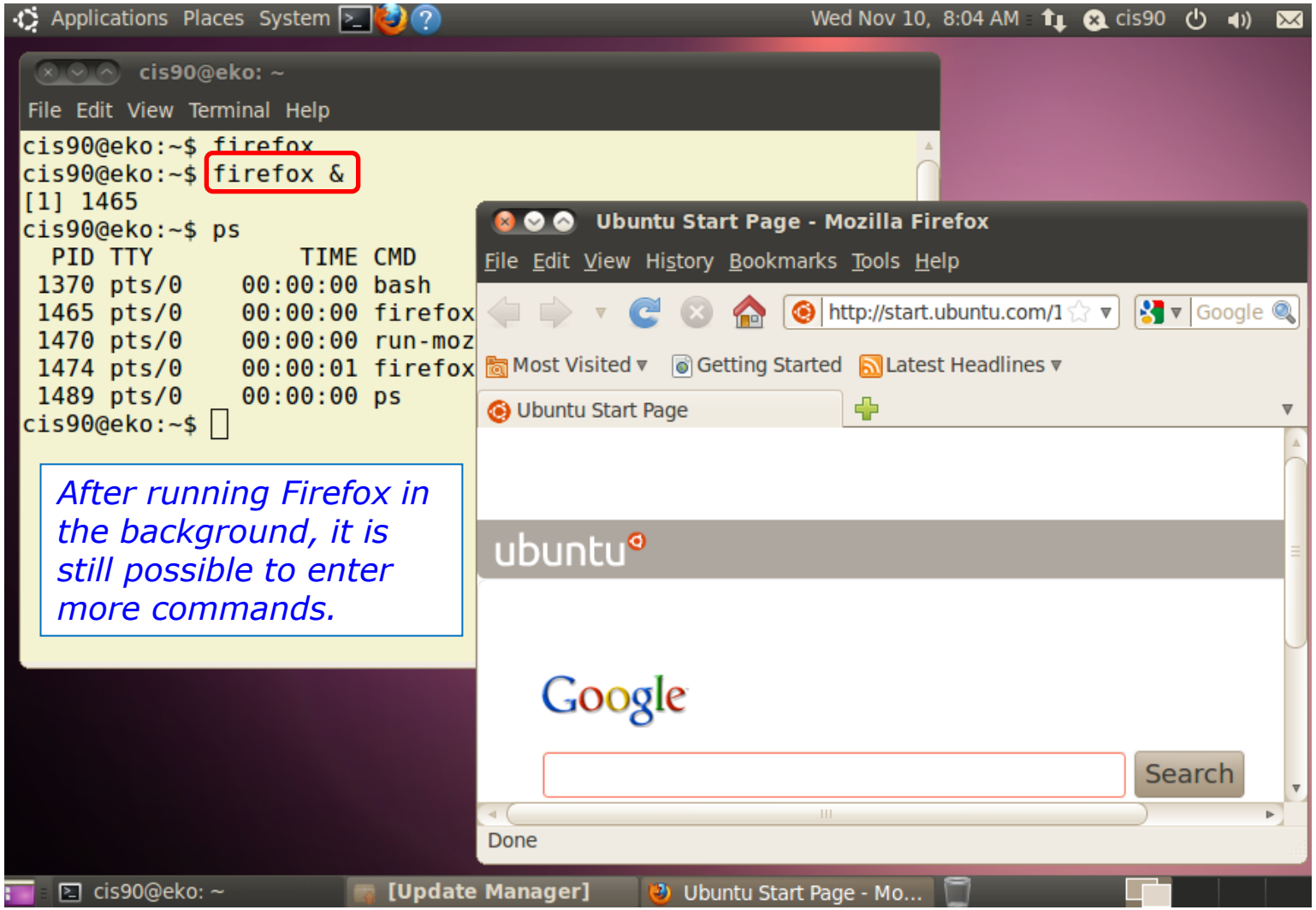
Job Control

Using **&** to run a command in the background



Job Control

Using **&** to run a command in the background



& append to a command to run it in the background

Example 1

```
/home/cis90/simmsben $ find / -user 1200 2> duh | sort > huh
```

No prompt

For long running commands or scripts you must wait for the command to finish before you type more commands

Example 2

```
/home/cis90/simmsben $ find / -user 1200 2> duh | sort > huh &
[1] 11601
/home/cis90/simmsben $ date
Tue Nov  9 14:38:35 PST 2010
```

Hit enter to get the prompt and continue working while the find command runs in the background

Job Control

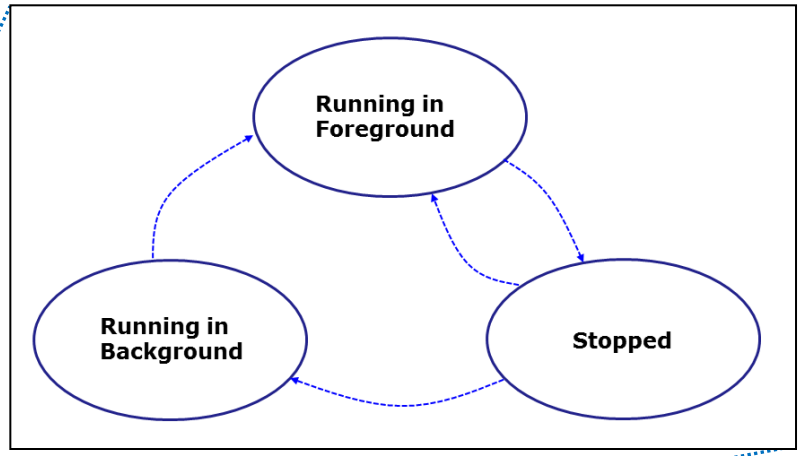
Job Control
A feature of the bash shell

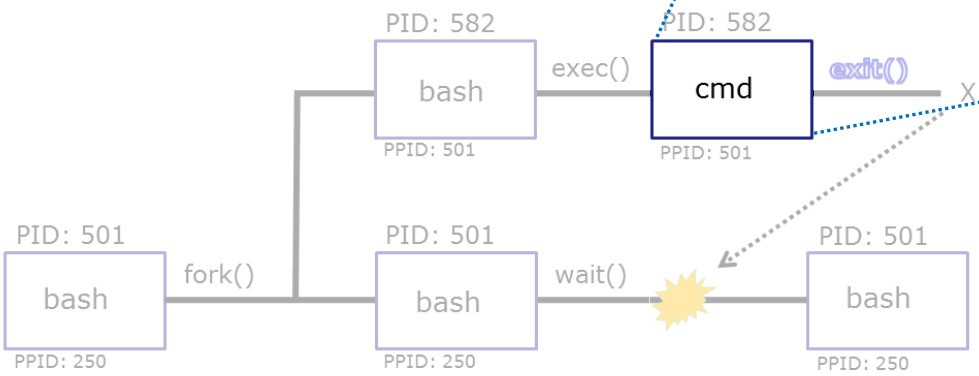
&	Append to a command to run it in the background
bg	Resumes a suspended job in the background
fg	Brings the most recent background process to the foreground
jobs	Lists all background jobs

*Use **jobs**, **bg**, **fg** to list and resume jobs in the foreground or background*

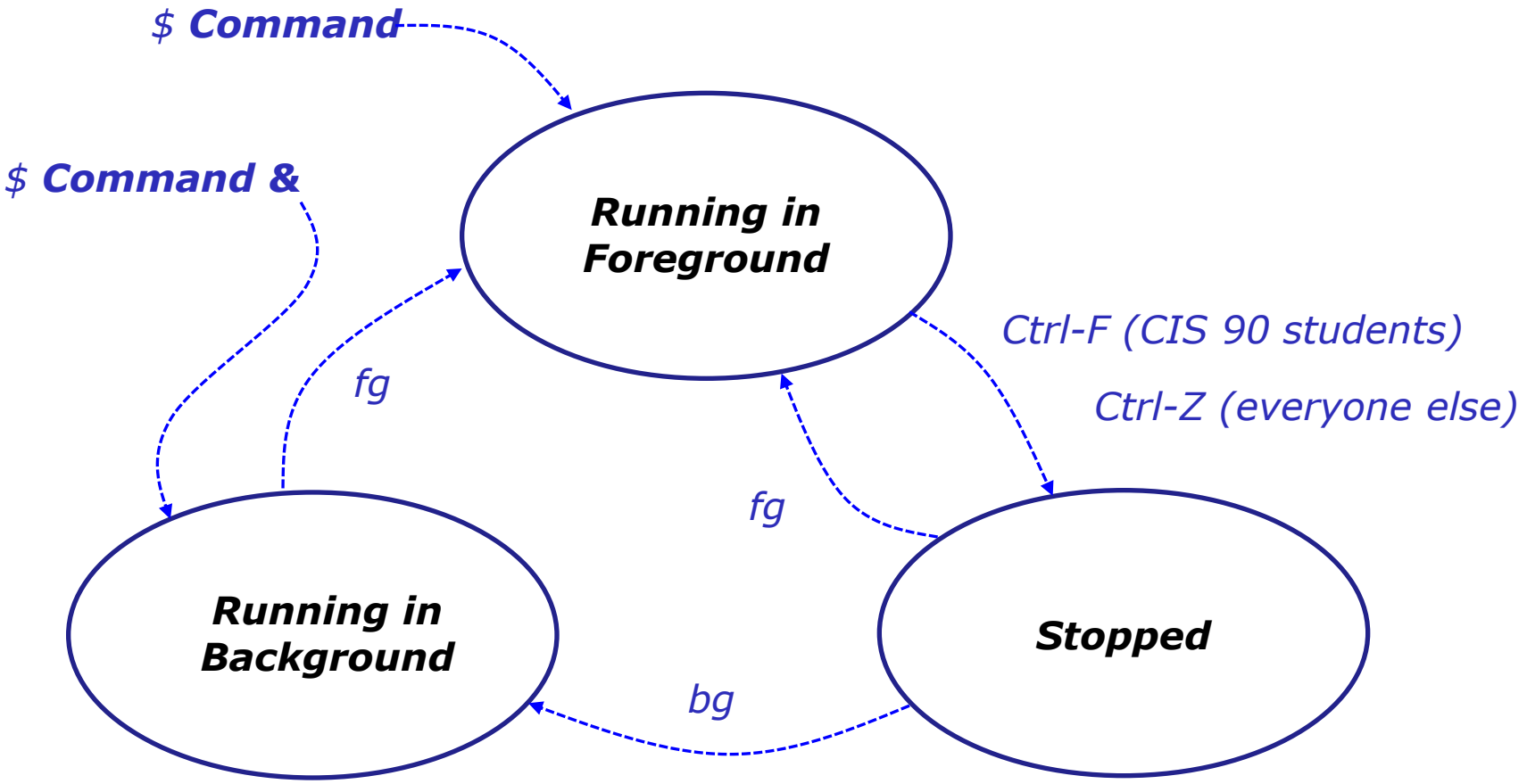
Job Control
A feature of the bash shell

*When a process is **running** (status=R) the user can **stop** it (status=T) and choose whether it runs in the **background** or **foreground***





Job Control
A feature of the bash shell



*Use the **jobs** command to view stopped and background jobs*

Job Control

Find out with keystroke combination is configured to suspend a process

```
/home/cis90ol/simmsben $ stty -a
speed 38400 baud; rows 24; columns 80; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^F; rprnt = ^R;
werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 -hupcl -cstopb cread -clocal -crtscts -cdtrdsr
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff
-iuclc -ixany -imaxbel -iutf8
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt
echoctl echoke
/home/cis90ol/simmsben $
```

In this case it is Ctrl-F that will be used to suspend a process

How is yours configured?

Job Control
Managing jobs

```
/home/cis90ol/simmsben $ sleep 120
Ctrl-Z or Ctrl-F (to suspend process)
[1]+  Stopped                 sleep 120
/home/cis90ol/simmsben $ sleep 110
Ctrl-Z or Ctrl-F (to suspend process)
[2]+  Stopped                 sleep 110
/home/cis90ol/simmsben $ sleep 100
Ctrl-Z or Ctrl-F (to suspend process)
[3]+  Stopped                 sleep 100

/home/cis90ol/simmsben $ jobs
[1]   Stopped                 sleep 120
[2]-  Stopped                 sleep 110
[3]+  Stopped                 sleep 100
```

Lets start up 3 sleep commands and suspend each of them.

Note: The sleep command is a simple way to run a command that will take awhile to finish.

***sleep 120** will last 120 seconds before it is finished.*

Job Control
Managing jobs

```
/home/cis90ol/simmsben $ jobs
[1]   Stopped                 sleep 120
[2]-  Stopped                 sleep 110
[3]+  Stopped                 sleep 100


/home/cis90ol/simmsben $ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1082  5364  5363  0  75   0 -  1168 wait   pts/2    00:00:00 bash
0 T  1082  5452  5364  0  75   0 -   929 finish pts/2    00:00:00 sleep
0 T  1082  5453  5364  0  75   0 -   929 finish pts/2    00:00:00 sleep
0 T  1082  5454  5364  0  75   0 -   929 finish pts/2    00:00:00 sleep
0 R  1082  5459  5364  0  77   0 -  1054 -      pts/2    00:00:00 ps
```

Note, all three processes are sTopped

Job Control
Managing jobs

```
/home/cis90ol/simmsben $ bg 2
[2]- sleep 110 &
/home/cis90ol/simmsben $ jobs
[1]-  Stopped                 sleep 120
[2]   Running                 sleep 110 &
[3]+  Stopped                 sleep 100
```

*Jobs can be resumed in the background using **bg***

```
/home/cis90ol/simmsben $ bg 1
[1]- sleep 120 &
/home/cis90ol/simmsben $ jobs
[1]   Running                 sleep 120 &
[2]-  Running                 sleep 110 &
[3]+  Stopped                 sleep 100
```

*or in the foreground using **fg***

```
/home/cis90ol/simmsben $ fg 3
sleep 100
```

At this point we lose control of the keyboard again until sleep 100 is finished

Job Control
Managing jobs

```
/home/cis90ol/simmsben $ jobs
[1]-  Done
sleep 120
[2]+  Done
sleep 110
```

Background jobs are all done!

Review of Load Balancing

Load Balancing

The **at** command:

- reads from stdin for a list of commands to run
- runs those commands at the specified time
- Any output from those commands will be emailed
- Use **atq** and **atrm** to manage scheduled commands

*Use **at** to schedule commands to run in the future*

Load Balancing
Managing queued jobs

```
at now + 5 minutes
at now + 1 hour
at 7:58AM
at 7:47PM 5/5/2012
at teatime
```

Ways to specify future times

Load Balancing

Managing queued jobs

```
/home/cis90/simben $ atq
25      2011-11-12 14:09 a simben90
28      2011-12-12 03:00 a simben90
27      2011-11-19 12:10 a simben90
26      2011-11-12 16:00 a simben90
24      2011-11-12 12:14 a simben90
```

*The **atq** command lists jobs queued to run in the future*

```
/home/cis90/simben $ atrm24
/home/cis90/simben $ atq
25      2011-11-12 14:09 a simben90
28      2011-12-12 03:00 a simben90
27      2011-11-19 12:10 a simben90
26      2011-11-12 16:00 a simben90
```

*The **atrm** command is used to remove jobs from the queue*

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

*Note: The **jobs** command lists processes running or suspended in the background and is NOT used for **at** commands.*

Load Balancing

Try it yourself with your own terminal device and username:

```
[rsimms@oslab ~]$ tty
/dev/pts/4

[rsimms@oslab ~]$ at now+2 minutes
at> echo "Take Benji for a walk" | mail -s "walk the dog" $LOGNAME
at> echo "Read your mail" > /dev/pts/4
at> <EOT>
job 11 at 2012-11-05 11:02
[rsimms@oslab ~]$ atq
11      2012-11-05 11:02 a rsimms
[rsimms@oslab ~]$
```

These should match

Type what happens in the chat window:

text editors

There are lots of text editors …

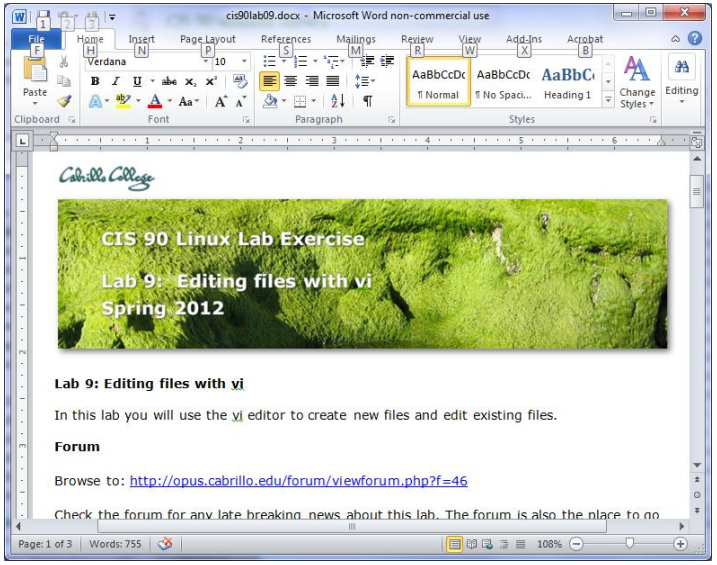
Windows
- notepad
- notepad++
- textpad

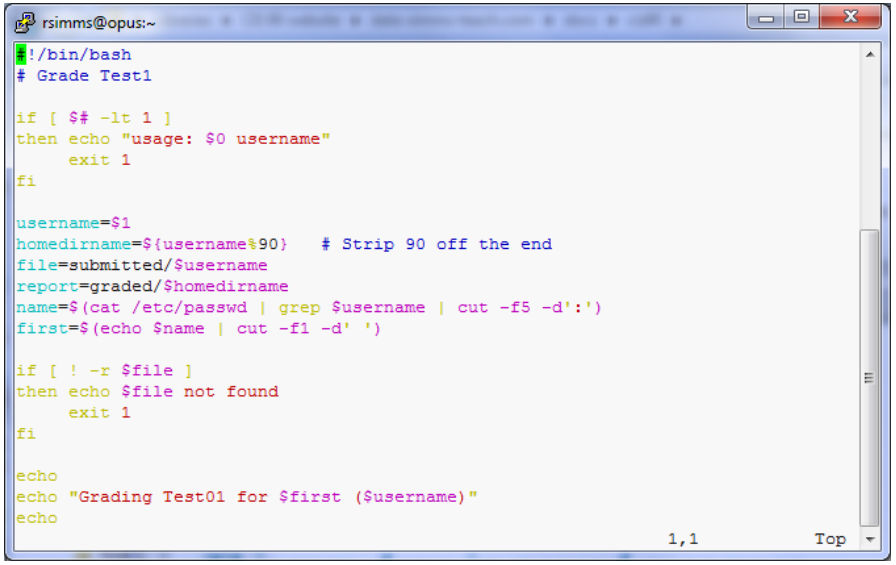
Mac
- TextWrangler

Linux
- gedit
- emacs
- nano
- vi

Text editors and word processors are different!

- *Word processors are used by many different people to create documents containing text and graphics.*
- *Text editors are used by programmers to develop software and web designers to create web sites.*





Word processors *allow a rich set of formatting (fonts, sizes, styles, color) and graphics to be added to documents.*

Text editors *use color to show the language syntax*

vi 101

On Opus we are actually running VIM

```
/home/cis90/simben $ type -a vi
vi is aliased to `vim'
vi is /bin/vi
/home/cis90/simben $ type vim
vim is hashed (/usr/bin/vim)
```

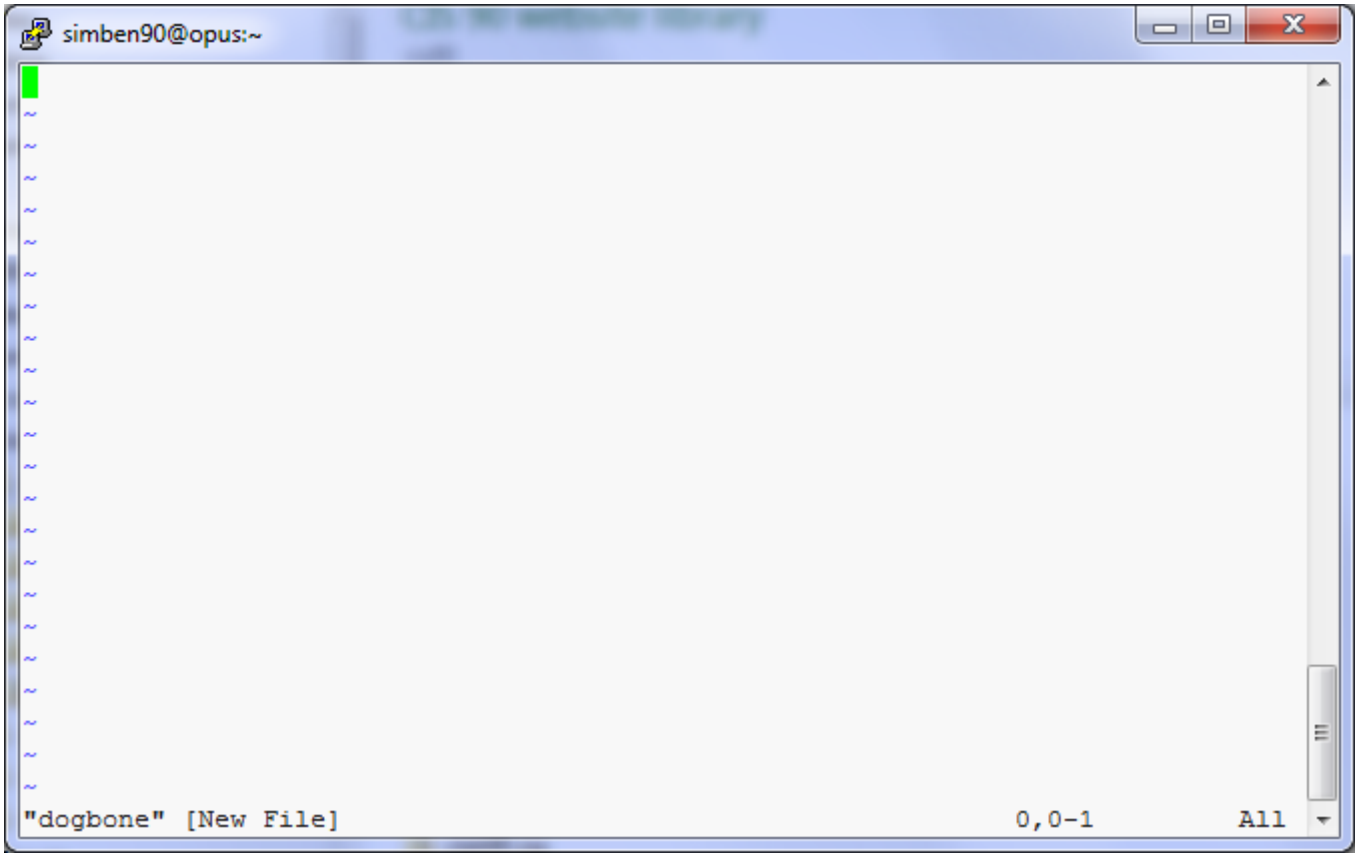
History:

- The original vi code was written by Bill Joy for BSD Unix
- Bill Joy co-founded Sun Microsystems in 1982
- vi (for “visual”)
- vim is an enhanced version of vi

```
/home/cis90/simben $
/home/cis90/simben $ vi dogbone
```

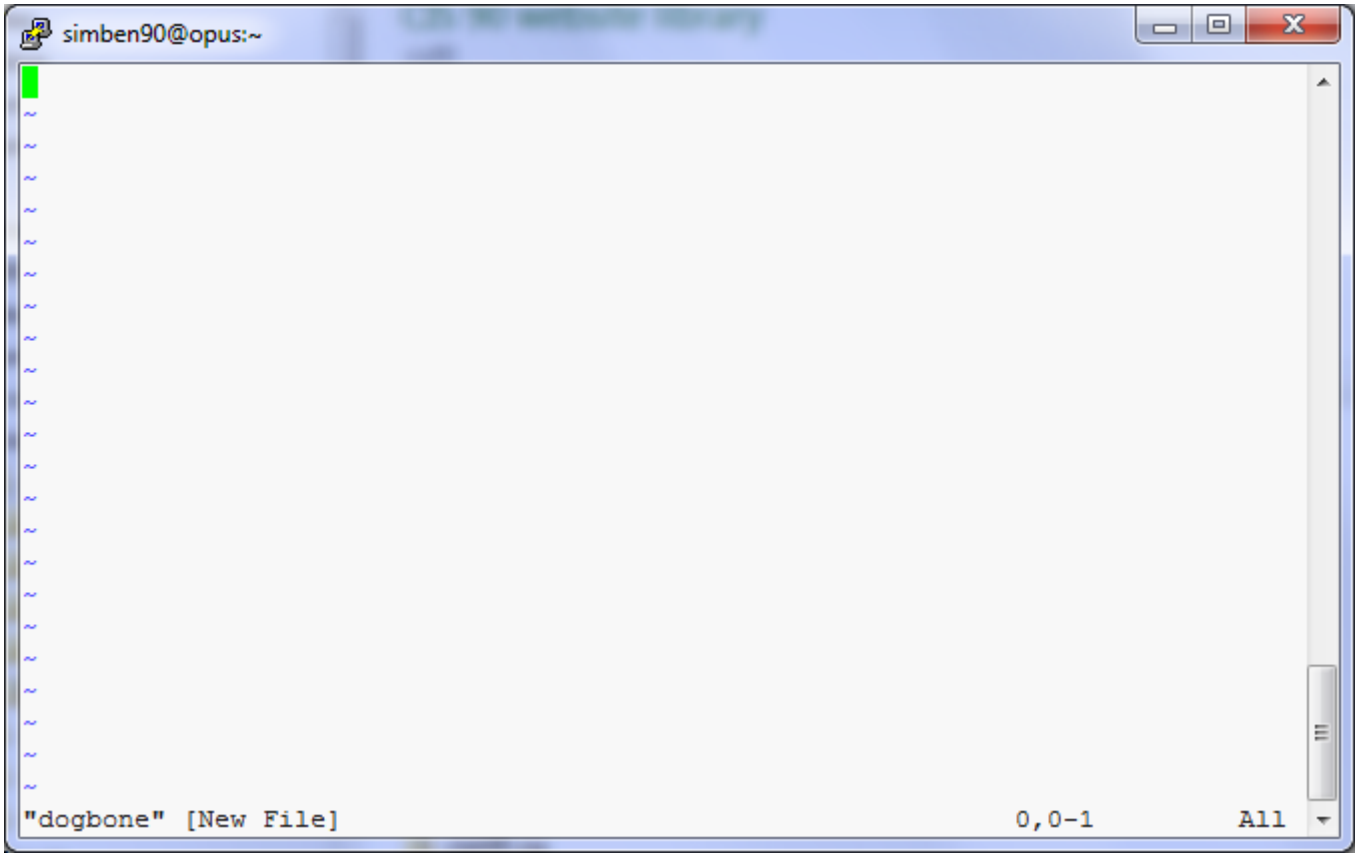
Type this

See this …



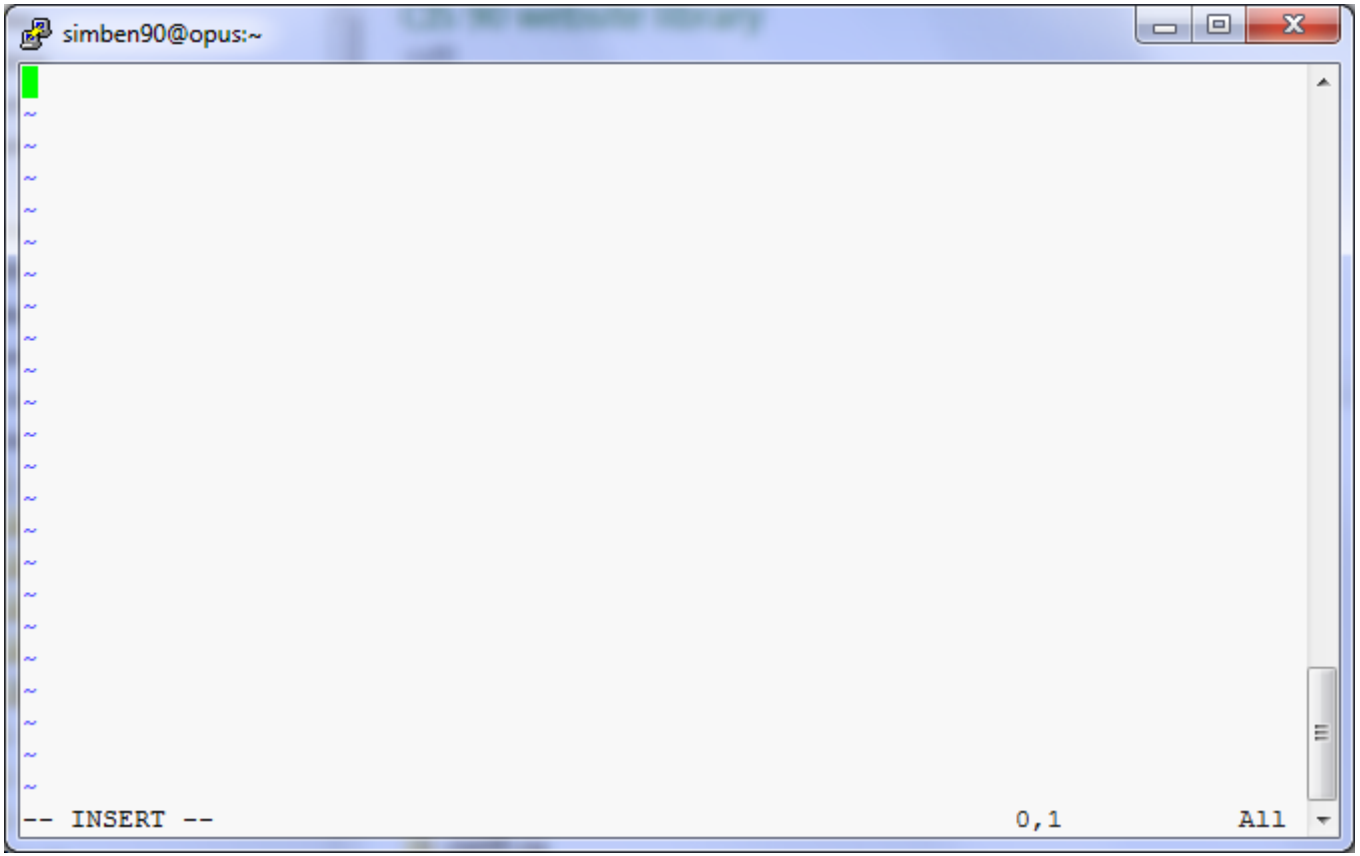
Take your hands OFF THE MOUSE – don't use it in vi!

*Tap the letter **i** key (for insert)*



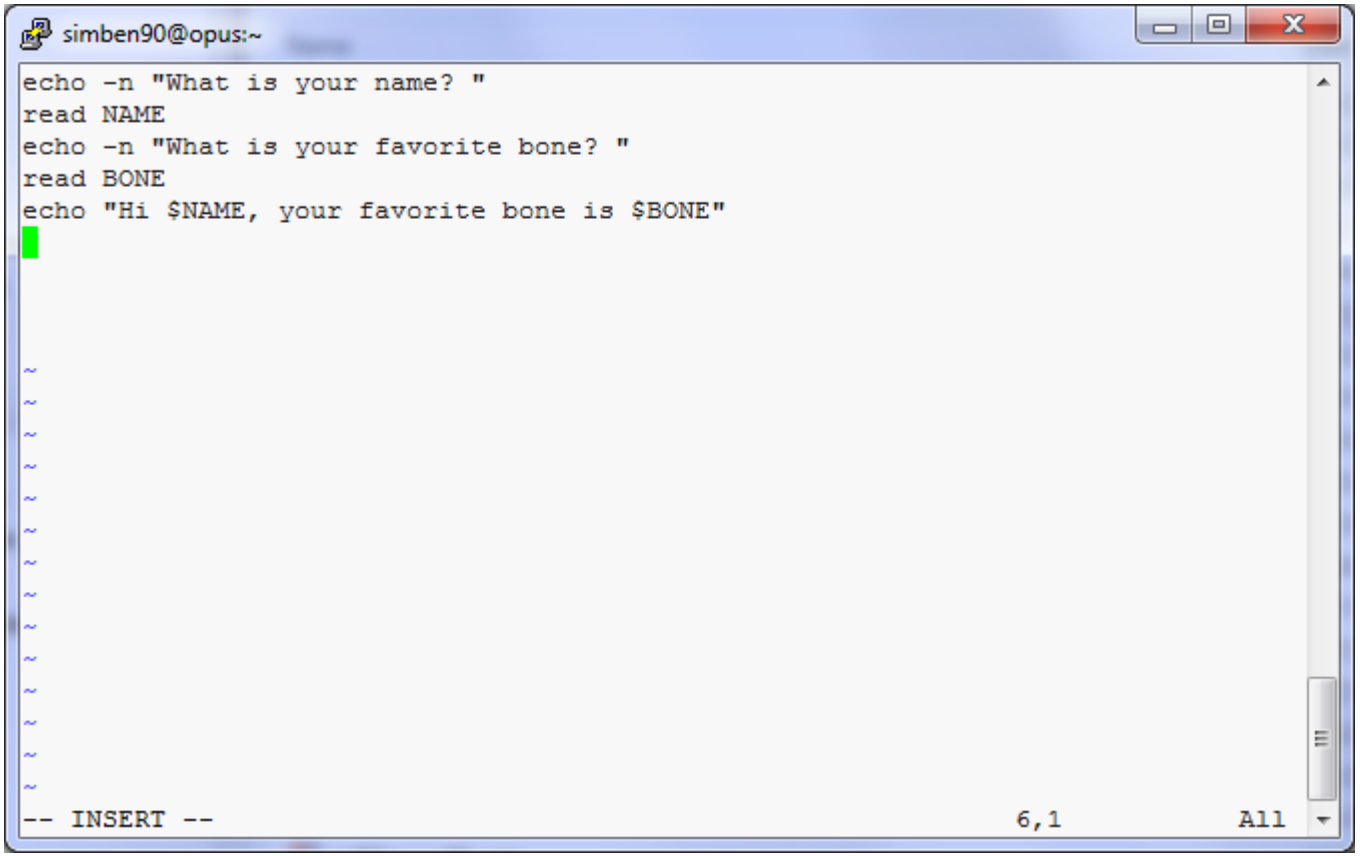
Keep your hands OFF THE MOUSE – don't use it in vi!

See this …



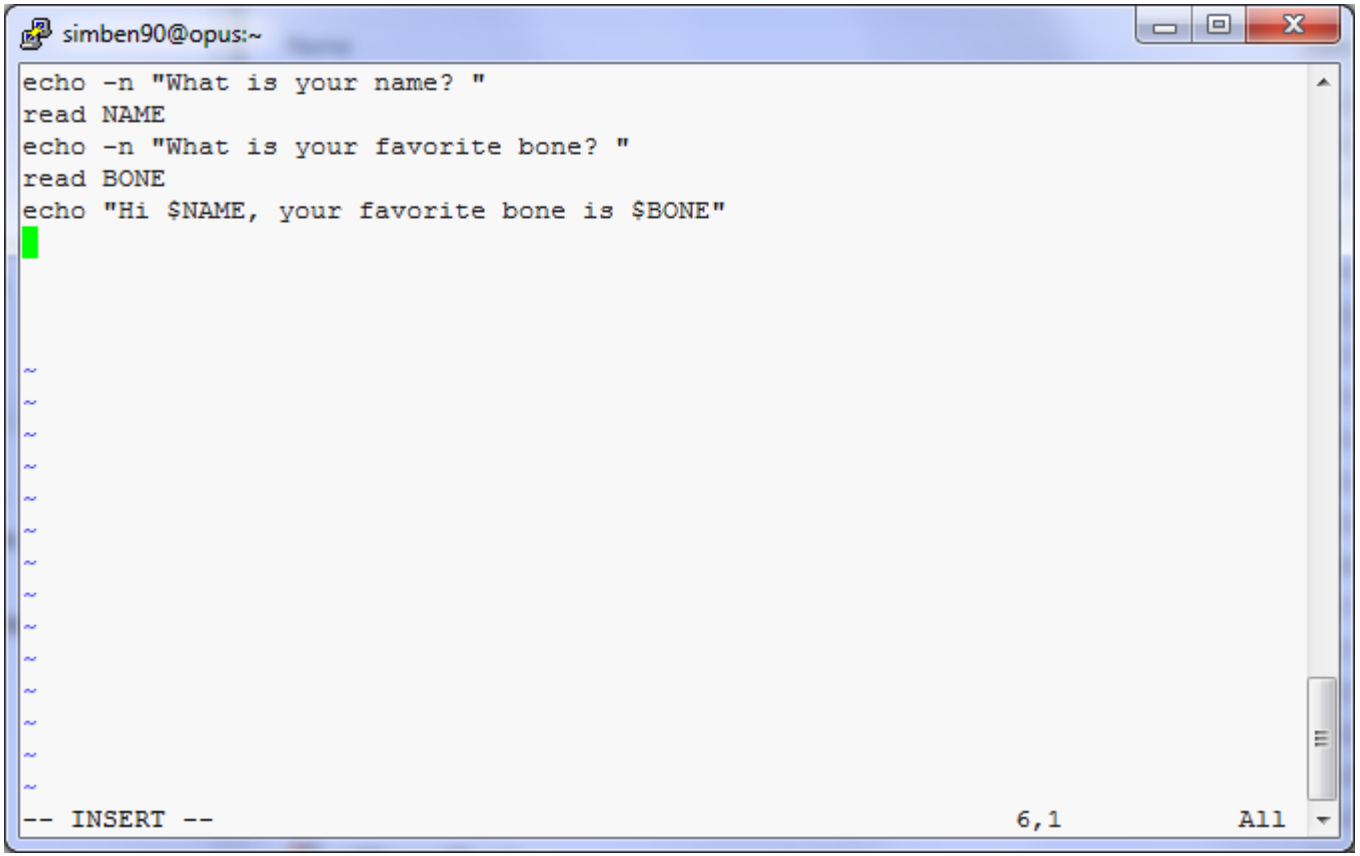
Keep your hands OFF THE MOUSE – don't use it in vi!

Very carefully type these five lines



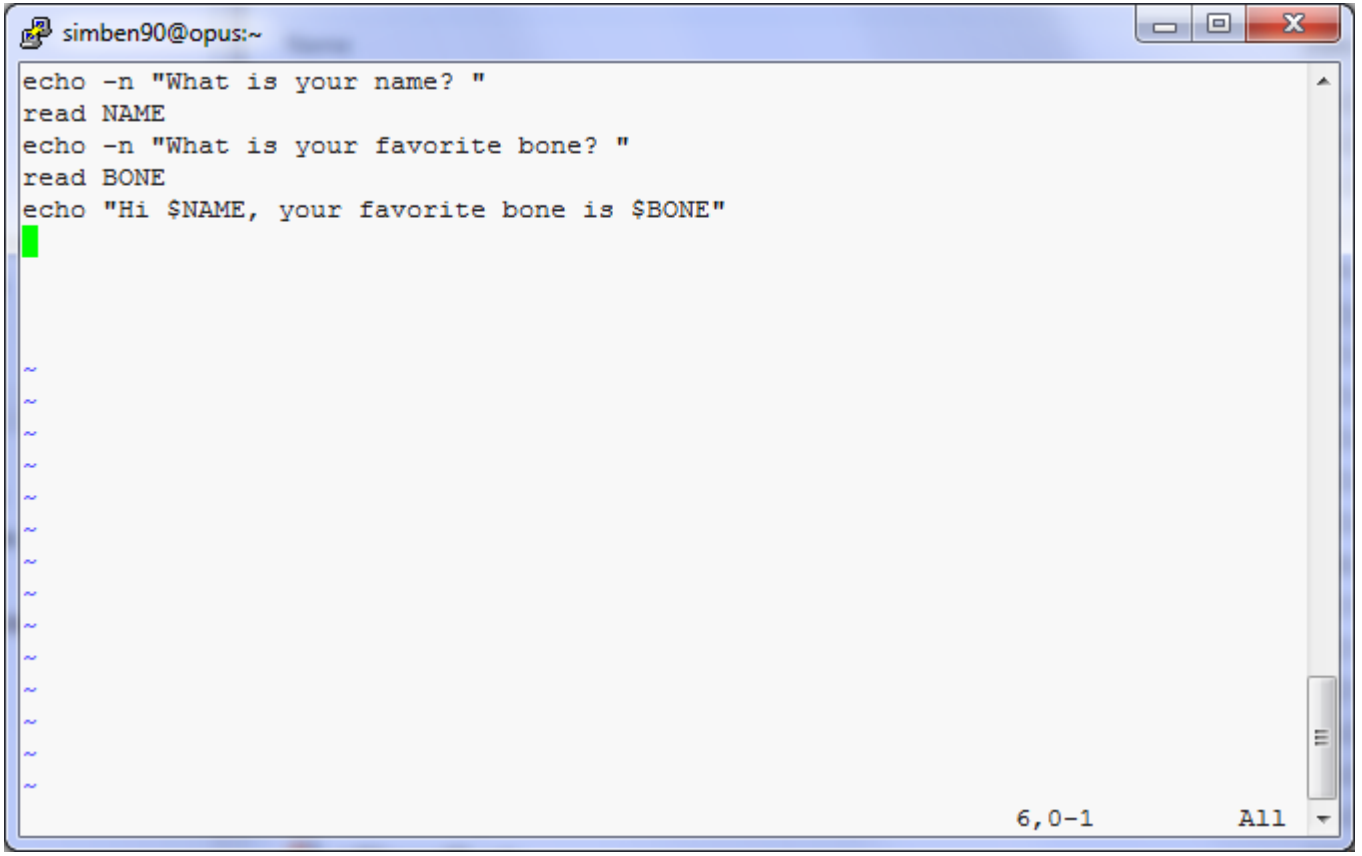
Keep your hands OFF THE MOUSE – don't use it in vi!

Have your neighbor check that your five lines are PERFECT



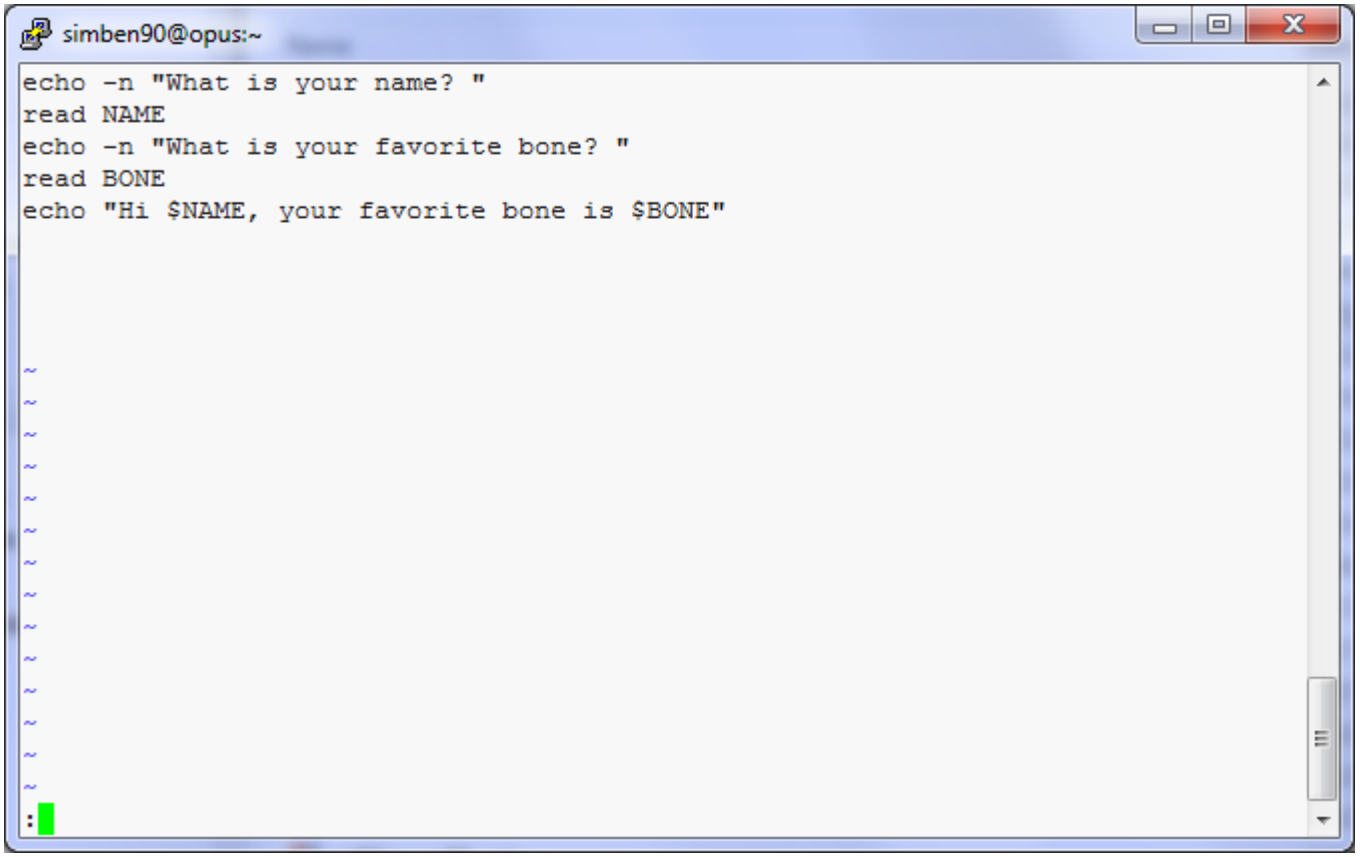
Keep your hands OFF THE MOUSE – don't use it in vi!

*Tap the **esc** key*



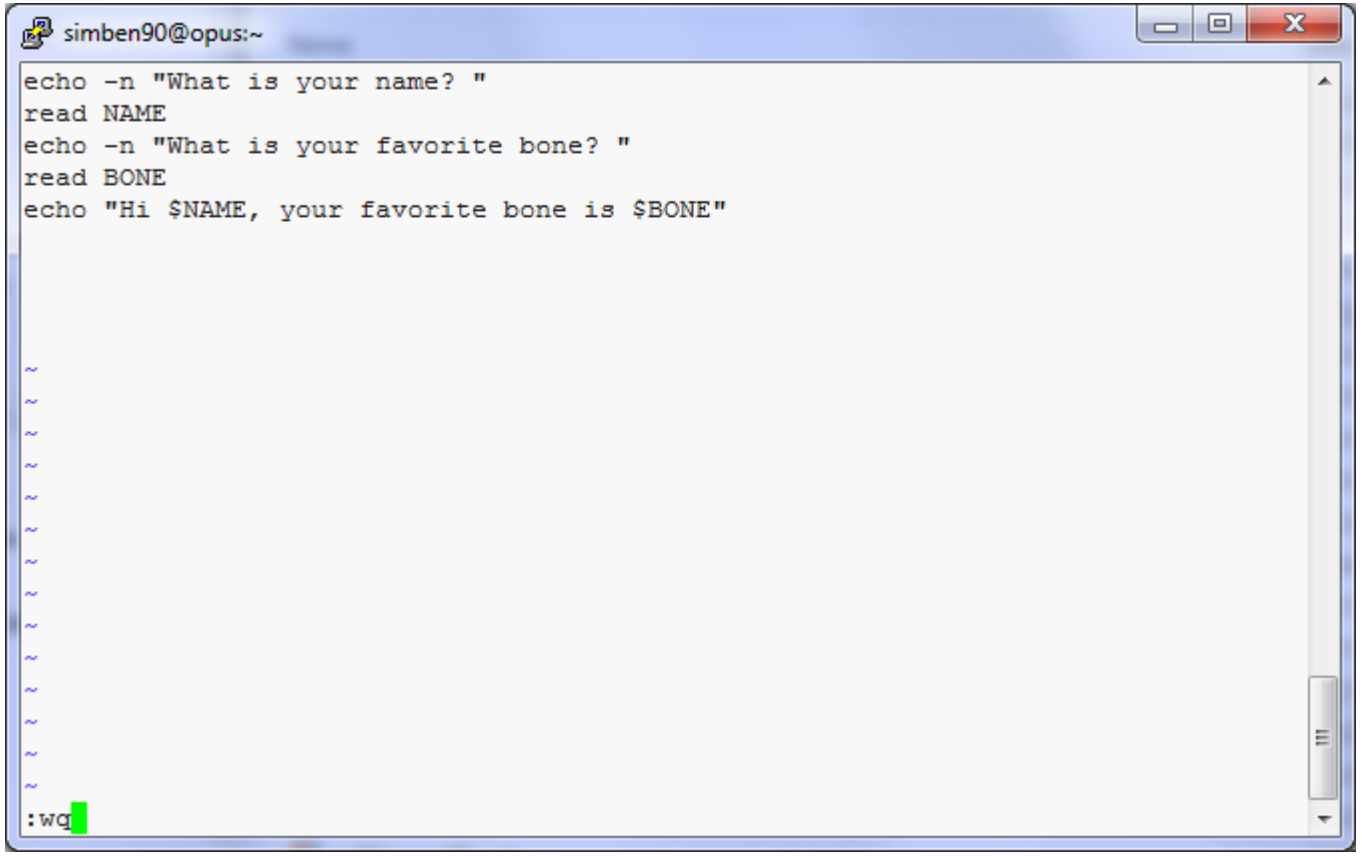
Keep your hands OFF THE MOUSE – don't use it in vi!

Type a **:**



Keep your hands OFF THE MOUSE – don't use it in vi!

*Type **wq***



Keep your hands OFF THE MOUSE – don't use it in vi!

Tap the enter key

```
/home/cis90/simben $ vi dogbone
/home/cis90/simben $
```

Add execute permissions and try your new script

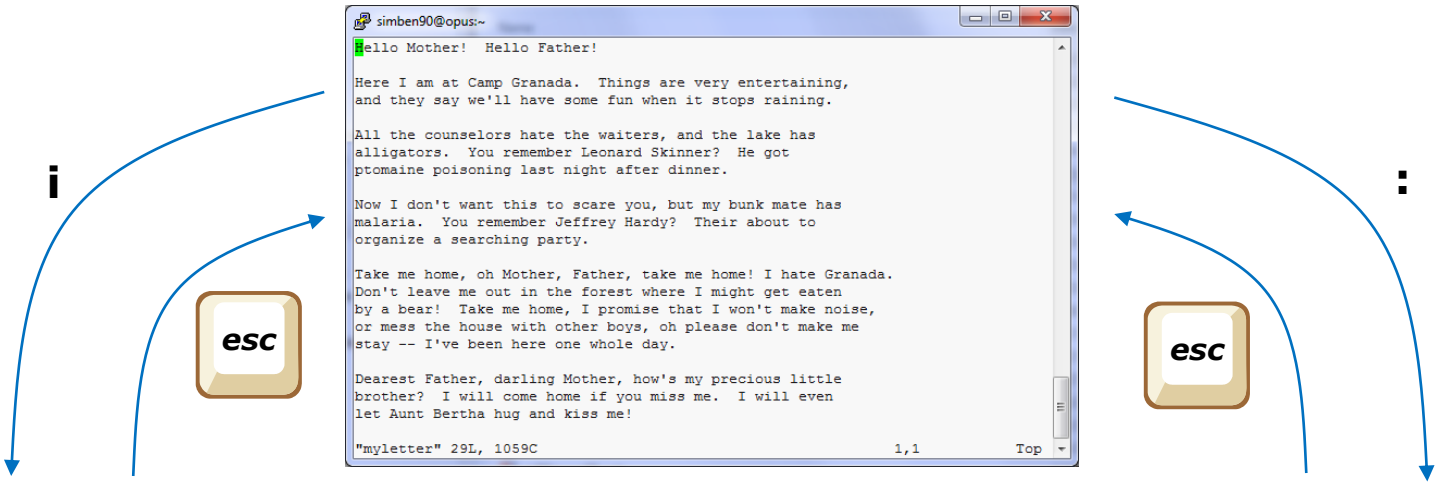
```
/home/cis90/simben $ chmod +x dogbone

/home/cis90/simben $ dogbone
What is your name? Benji
What is your favorite bone? chicken
Hi Benji, your favorite bone is chicken
/home/cis90/simben $
```

vi

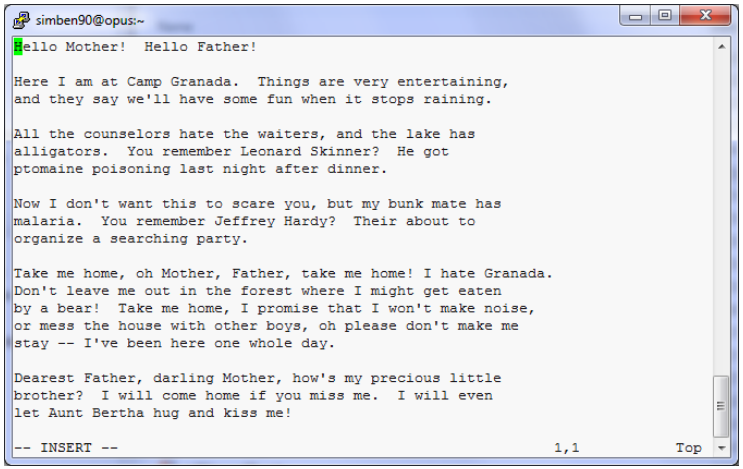
COMMAND mode
INSERT mode
command LINE mode

```
/home/cis90/simben $ cp letter myletter
/home/cis90/simben $ vi myletter
```

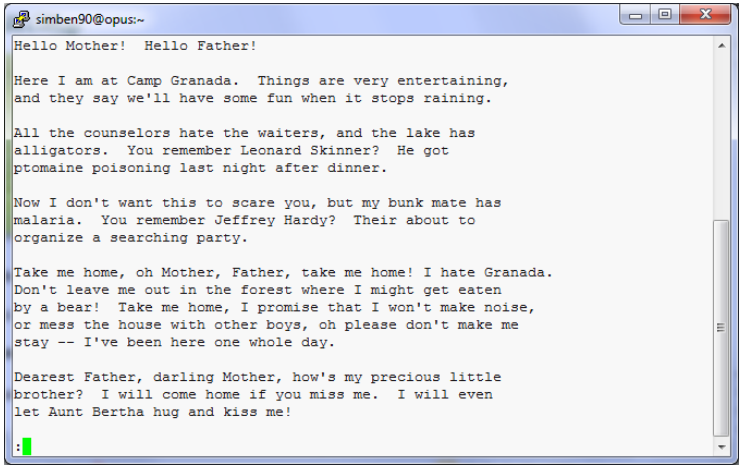
COMMAND mode



INSERT mode



Command LINE mode



vi

Moving around in a file

Use in COMMAND mode

h moves the cursor one character to the left
j moves the cursor down one line
k moves the cursor up one line
l moves the cursor one character to the right

Try typing a number in front of these commands and notice what happens

^d scrolls down 10 lines
^u scrolls up 10 lines
^f page forward one page
^b page back one page

With vim (not vi) you can use arrow and page keys instead of these letter commands

vi

Moving around in a file

Use in COMMAND mode

w moves the cursor one "word" forward
b moves the cursor one "word" back

Try typing a number in front of these commands and notice what happens

0 (zero) moves the cursor to the beginning of the line
$ moves the cursor to the end of the line

G moves the cursor to the last line in the file
1G moves the cursor to the first line in the file
105G moves the cursor to line 105

vi
Saving and Quiting

Use in command LINE mode

:w writes any changes to the file you are editing (like Save)

:q quits vi if you have saved your changes
:q! quits vi even if you haven't saved changes

:wq writes and quits
:wq! writes and quits vi even if you haven't saved changes

vi

Reading in and Writing out files

Use in command LINE mode

:w *filename* saves your file to a new name (like Save As)
:w! *filename* saves your file to a new name overwriting any previous data

:r *filename* reads in the contents of *filename* starting from the cursor position

:e *filename* replaces the current content with the content from *filename*

vi
Entering INSERT mode

From command mode.

i Ready to insert characters immediately before the current cursor position
I Ready to insert characters at the start of the current line

a Ready to append characters immediately after the current cursor position
A Ready to append characters at the end of the current line

o Ready to input characters in a new line that opens up below the cursor
O Ready to input characters in a new line that opens up above the cursor

vi
Cut, Copy, Pasting Commands

Use in command mode

x Deletes the current character
r Replace the current character with the character you type next

dw Deletes the current word
dd Deletes the current line

D Deletes to the end of the line

yy Copies a line to the clipboard buffer
p Pastes whatever is in the clipboard buffer below the current cursor
P Pastes whatever is in the clipboard buffer above the current cursor

vi
Miscellaneous Useful Commands

Use in command mode.

^g Tells you the filename you are editing and what line your cursor is on

u Undoes the last command you executed
^r Undo the undo (redo)

. Repeats the last command you executed

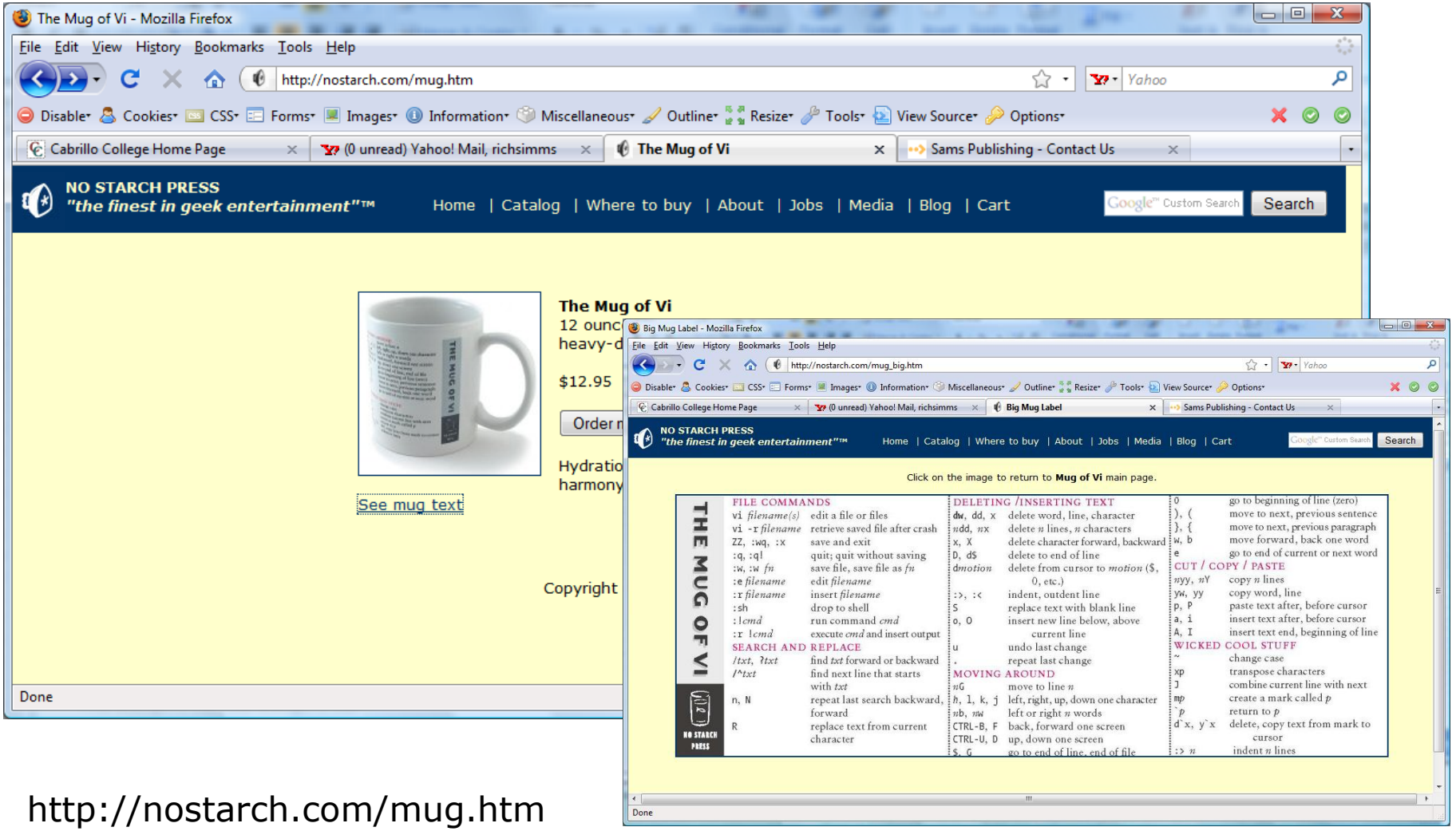
/string Searches for the string of characters in the file
n Finds the next occurrence of the current search string looking down the file
N Finds the next occurrence of the current search string looking up the file

~ Changes the case of the current character

:%s /*string1*/*string2*/g replaces all string1 with string2 in the file

http://vim.wikia.com/wiki/Main_Page

The Mug of vi



http://nostarch.com/mug.htm

/bin/mail and vi

```
/home/cis90/simmsben $ mail roddyduk
Subject: Good bones
Hey Duke,
I really appreciate thatbone you sent me last week.
Let me knwo if you want to go mark some fench posts
this weekend.
Later,
Ben
```

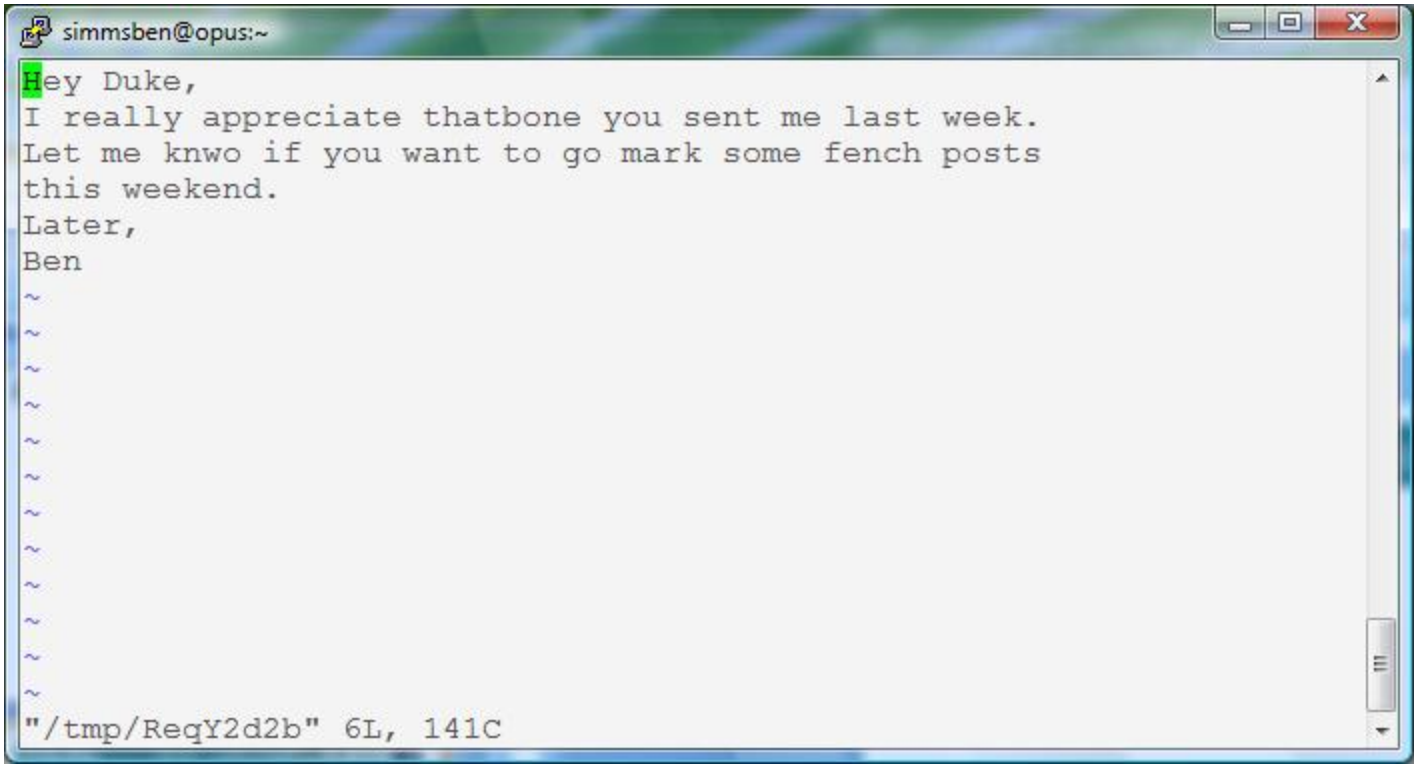
You are composing a message and you spot some typos …
CRUD … what can you do?

/bin/mail and vi

```
/home/cis90/simmsben $ mail roddyduk
Subject: Good bones
Hey Duke,
I really appreciate thatbone you sent me last week.
Let me knwo if you want to go mark some fench posts
this weekend.
Later,
Ben
~v
```

Well … you could try the ~v command

/bin/mail and vi



The message is loaded into vi where changes or additions can be made. :wq is used to save and quit vi

/bin/mail and vi

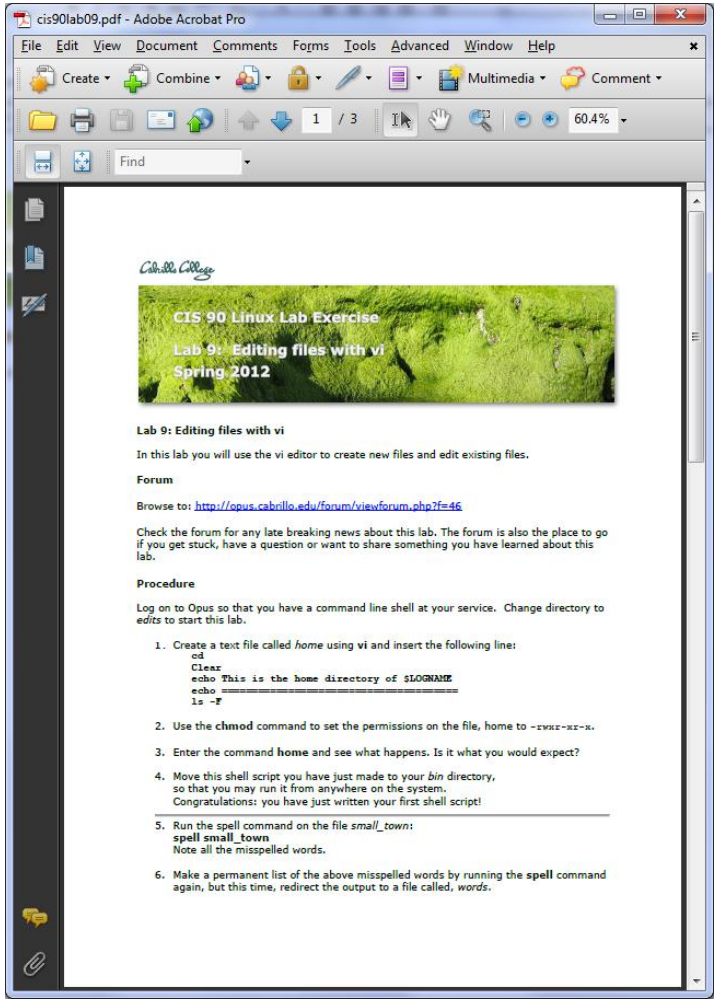
```
/home/cis90/simmsben $ mail roddyduk
Subject: Good bones
Hey Duke,
I really appreciate thatbone you sent me last week.
Let me knwo if you want to go mark some fench posts
this weekend.
Later,
Ben
~v
(continue)
.
Cc:
/home/cis90/simmsben $
```

The earlier text with typos is still showing, however the corrected version is what is actually sent.

/bin/mail and vi

```
/home/cis90/roddyduk $ mail
Mail version 8.1 6/6/93.  Type ? for help.
"/var/spool/mail/roddyduk": 1 message 1 unread
>U  1 simmsben@opus.cabril  Mon Nov 10 20:25  22/782   "Good bones"
& 1
Message 1:
From simmsben@opus.cabrillo.edu  Mon Nov 10 20:25:32 2008
Date: Mon, 10 Nov 2008 20:25:32 -0800
From: Benji Simms <simmsben@opus.cabrillo.edu>
To: roddyduk@opus.cabrillo.edu
Subject: Good bones

Hey Duke,
I really appreciate that bone you sent me last week.
Let me know if you want to go mark some fence posts
this weekend.
Later,
Ben

&
```

The message Duke reads has all the typos fixed.



Lab 9 will help you start building your vi skills!

Instructor: remember to mail students the tech file!

A Tangent on Spell

spell command

```
/home/cis90/roddyduk/edits $ cat text
Welcome to the CIS 90 class !!

/home/cis90/roddyduk/edits $ spell text
CIS
```

spell *command flags CIS as misspelled word.*

How can we add CIS to the dictionary?

spell command

```
/home/cis90/roddyduk/edits $ cat text
Welcome to the CIS 90 class !!
/home/cis90/roddyduk/edits $ spell text
CIS
```

How can we add CIS to the dictionary?

```
/home/cis90/roddyduk/edits $ man spell
No manual entry for spell
/home/cis90/roddyduk/edits $ type spell
spell is hashed (/usr/bin/spell)
/home/cis90/roddyduk/edits $ file usr/bin/spell
/usr/bin/spell: Bourne shell script text executable
/home/cis90/roddyduk/edits $ cat /usr/bin/spell
#!/bin/sh

# aspell list mimicks the standard unix spell program, roughly.

cat "$@" | aspell list --mode=none | sort -u

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

Hmmm. No man page for spell ???????????

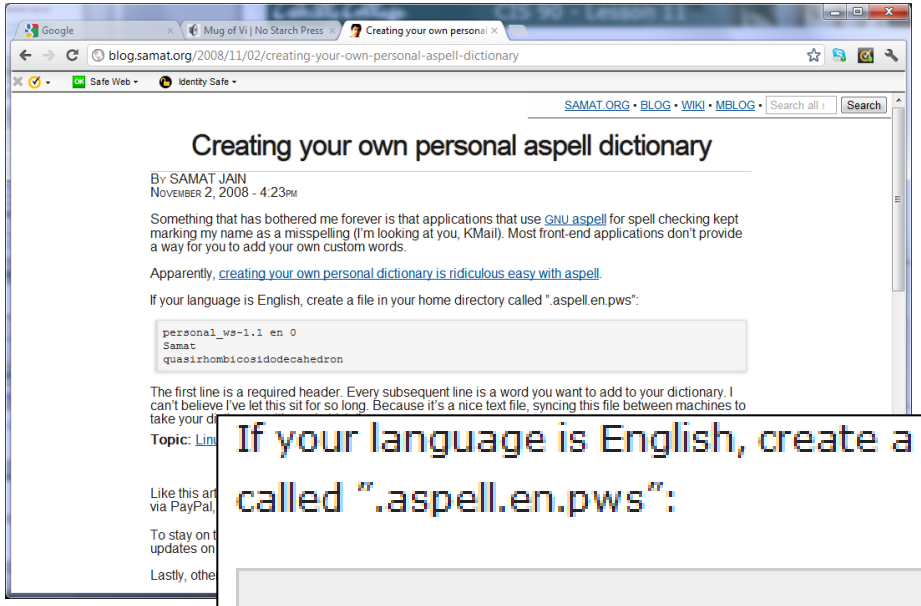
*OK, the actual command is **aspell***

spell command

```
ASPELL(1)                Aspell Abbreviated User's Manual                ASPELL(1)

NAME
       aspell - interactive spell checker

SYNOPSIS
       aspell [options] <command>

DESCRIPTION
       aspell  is  a utility that can function as an ispell -a replacement,
       as an independent spell checker, as  a  test  utility  to  test  out
       Aspell features, and as a utility for managing dictionaries.

COMMANDS
       <command> is one of:

       -?,help
              display the help message

       -c,check file
              to spell-check a file
```

There must be a way to add CIS …. but … lets try google

spell command



How to add words to your dictionary

If your language is English, create a file in your home directory called ".aspell.en.pws":

```
personal_ws-1.1 en 0
Samat
quasirhombicosidodecahedron
```

Googling "linux aspell personal dictionary" yields this page

Bingo! Thank you Samat Jain

spell command

```
/home/cis90/roddyduk/edits $ cd
/home/cis90/roddyduk $ echo "personal_ws-1.1 en 0" >  .aspell.en.pws
/home/cis90/roddyduk $ echo "CIS" >> .aspell.en.pws
/home/cis90/roddyduk $ cd edits/
/home/cis90/roddyduk/edits $ spell text
```

This is how you would add your own custom dictionary to be used with spell checks

Wrap up

New commands:

vi	Run vi editor

New Files and Directories:

na	na

Next Class

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

Lab 9 Five Posts

Quiz questions for next class:

- How do you send a SIGKILL to one of your own processes?
- What vi command is used to exit vi without saving any of the changes you made?
- What vi commands are used for copy and paste?

Backup

The **mystery** of Ctrl-Z vs Ctrl-F

Signals
Special keystrokes

```
/home/cis90/roddyduk $ stty -a
speed 38400 baud; rows 26; columns 78; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^F; rprnt = ^R;
werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
```

```
[rsimms@opus ~]$ stty -a
speed 38400 baud; rows 39; columns 84; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>; eol2 = <undef>;
swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W;
lnext = ^V; flush = ^O; min = 1; time = 0;
```

Why does the keystroke to send a Suspend (SIGTSTP or 20) signal differ between roddyduk (^F or Ctrl-F) and rsimms (^Z or Ctrl-Z)?

Job Control
A feature of the bash shell

Ctrl-Z or Ctrl-F (sends SIGTSTP 20 signal)
- Stops (suspends) a foreground process

```
[rsimms@opus ~]$ sleep 5

[1]+  Stopped                 sleep 5
```

Ctrl-Z is tapped which stops the sleep command

PID 7728 is stopped

```
[rsimms@opus ~]$ ps -l  -u rsimms
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
5 S   201  5368  5365  0  75   0 -  2460 -      ?        00:00:00 sshd
0 S   201  5369  5368  0  76   0 -  1165 wait   pts/0    00:00:00 bash
5 S   201  6203  6200  0  75   0 -  2491 -      ?        00:00:00 sshd
0 S   201  6204  6203  0  75   0 -  1165 -      pts/6    00:00:00 bash
0 T   201  7728  6204  0  75   0 -   926 finish pts/6    00:00:00 sleep
0 R   201  7730  5369  0  78   0 -  1062 -      pts/0    00:00:00 ps
[rsimms@opus ~]$
```

Job Control

A feature of the bash shell

bg command

- Resumes a suspended job in the background

```
[rsimms@opus ~]$ sleep 5

[1]+  Stopped                 sleep 5
[rsimms@opus ~]$ bg
[1]+ sleep 5 &
[rsimms@opus ~]$
```

bg resumes the sleep command

PID 7728 is gone

```
[rsimms@opus ~]$ ps -l  -u rsimms
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
5 S   201  5368  5365  0  75   0 -  2460 -      ?        00:00:00 sshd
0 S   201  5369  5368  0  76   0 -  1165 wait   pts/0    00:00:00 bash
5 S   201  6203  6200  0  75   0 -  2491 -      ?        00:00:00 sshd
0 S   201  6204  6203  0  75   0 -  1165 -      pts/6    00:00:00 bash
0 R   201  7742  5369  0  78   0 -  1061 -      pts/0    00:00:00 ps
[rsimms@opus ~]$
```

Signals
Jim's app script

rsimms@opus:/home/cis90/depot

```
#!/bin/sh
#
# app - script to demostrate use of signals
#
# Usage:  run app with no options or parameters
#
# Send signals to it with keystrokes or kill command
#
# Notes:
# stty -echo stop the display of characters typed
# stty echo makes typed characters visible again
# stty susp ^Z sets suspend keystroke to Ctlr-Z (to stop forground processes)
# stty susp @ sets suspend character to @ (to stop foreground processes)
#
trap '' 2  #Ignore SIGINT
trap 'echo -n quit it!' 3 #Handle SIGQUIT
trap 'stty echo susp ^Z;echo ee; echo cleanup;exit' 15 #Handle SIGTERM
clear
banner testing
stty -echo susp @
sleep 1
echo  one
sleep 1
echo two
sleep 1
echo -n thr
while :
do sleep 1
done
~
                                                          13,1          All
```

This is why Cntl-F (suspend) stopped working and we had to use Ctrl-Z

Tangent on bg and SIGCONT

Signals

What is signal 18?

Signals

SIGSTKFLT	16	Stack fault
SIGCHLD	17	Child process has stopped or exited, changed (POSIX)
SIGCONT	18	Continue executing, if stopped (POSIX)
SIGSTOP	19	Stop executing(can't be caught or ignored) (POSIX)
SIGTSTP	20	Terminal stop signal (POSIX) ***Ctrl-Z or Ctrl-F***
SIGTTIN	21	Background process trying to read, from TTY (POSIX)
SIGTTOU	22	Background process trying to write, to TTY (POSIX)
SIGURG	23	Urgent condition on socket (4.2 BSD)
SIGXCPU	24	CPU limit exceeded (4.2 BSD)
SIGXFSZ	25	File size limit exceeded (4.2 BSD)
SIGVTALRM	26	Virtual alarm clock (4.2 BSD)
SIGPROF	27	Profiling alarm clock (4.2 BSD)
SIGWINCH	28	Window size change (4.3 BSD, Sun)
SIGIO	29	I/O now possible (4.2 BSD)
SIGPWR	30	Power failure restart (System V)

Signal 18 continues a stopped process ... isn't that what bg does?

The bg command is used to resume a stopped process

```
/home/cis90/roddyduk $ sleep 60
Ctrl-F (or Ctrl-Z) typed here
[1]+  Stopped                 sleep 60
/home/cis90/roddyduk $ bg
[1]+ sleep 60 &
/home/cis90/roddyduk $ jobs
[1]+  Running                 sleep 60 &
/home/cis90/roddyduk $ jobs
[1]+  Running                 sleep 60 &
/home/cis90/roddyduk $ jobs
[1]+  Done                    sleep 60
/home/cis90/roddyduk $
```

bg resumed the stopped process which runs till it is finished

*Instead of using **bg** to resume a stopped process in the background, lets try a SIGCONT (signal 18) instead*

```
/home/cis90/roddyduk $ sleep 60
Ctrl-F (or Ctrl-Z) typed here
[1]+  Stopped                 sleep 60
/home/cis90/roddyduk $ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000 10705 10704  0  76   0 -  1165 wait   pts/0    00:00:00 bash
0 T  1000 10743 10705  0  75   0 -   926 finish pts/0    00:00:00 sleep
0 R  1000 10744 10705  0  78   0 -  1051 -      pts/0    00:00:00 ps
/home/cis90/roddyduk $ jobs
[1]+  Stopped                 sleep 60
/home/cis90/roddyduk $ kill -18 10743
/home/cis90/roddyduk $ jobs
[1]+  Running                 sleep 60 &
/home/cis90/roddyduk $ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000 10705 10704  0  75   0 -  1165 wait   pts/0    00:00:00 bash
0 S  1000 10743 10705  0  85   0 -   926 322800 pts/0    00:00:00 sleep
0 R  1000 10746 10705  0  77   0 -  1050 -      pts/0    00:00:00 ps
/home/cis90/roddyduk $ jobs
[1]+  Running                 sleep 60 &
/home/cis90/roddyduk $ jobs
[1]+  Running                 sleep 60 &
/home/cis90/roddyduk $ jobs
[1]+  Done                    sleep 60
```

Note sending a 18 signal or using the bg command will resume a stopped process