



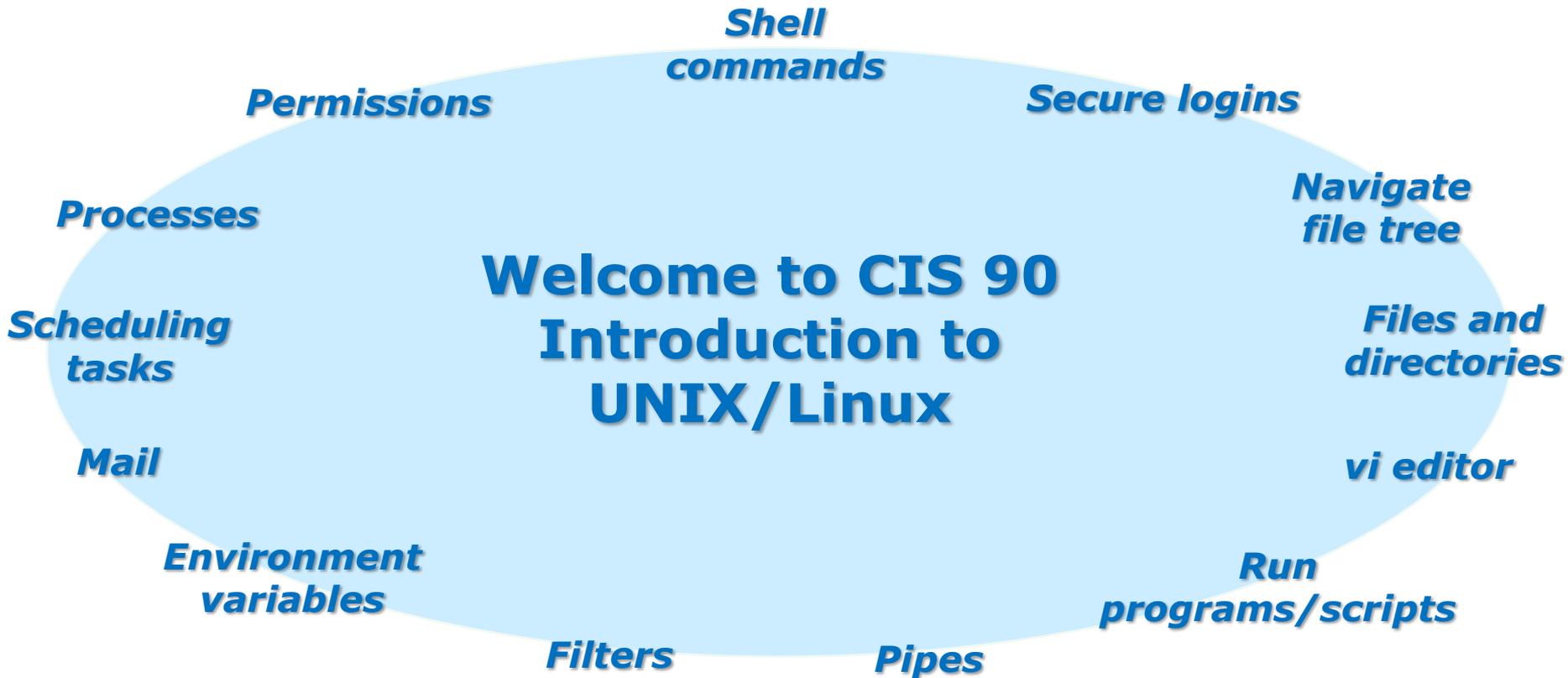
Rich's lesson module checklist

- Slides
- WB converted
- Print out agenda slide

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

- Lab tested and uploaded
- Tech file email for Lab 9 ready
- Lock/unlock turnin directory
- Apache configured for student websites
- Test(s) #2 copied to test archives folder on Blackboard

- Materials uploaded
- Backup slides, CCC info, handouts on flash drive
- Spare 9v battery for mic



Student Learner Outcomes

1. Navigate and manage the UNIX/Linux file system by viewing, copying, moving, renaming, creating, and removing files and directories.
2. Use the UNIX features of file redirection and pipelines to control the flow of data to and from various commands.
3. With the aid of online manual pages, execute UNIX system commands from either a keyboard or a shell script using correct command syntax.

Introductions and Credits



Jim Griffin

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



Rich Simms

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

And thanks to:

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



Student checklist for laying out screen when attending class

- Browse to the CIS 90 website Calendar page
 1. <http://simms-teach.com>
 2. Click CIS 90 link on left panel
 3. Click Calendar link near top of content area
 4. Locate today's lesson on the Calendar

- Download the presentation slides for today's lesson for easier viewing

- Click Enter virtual classroom to join CCC Confer session

- Connect to Opus using Putty or ssh command



Student checklist for laying out screen when attending class

Google

CCC Confer

Downloaded PDF of Lesson Slides

The screenshot displays a virtual classroom interface. On the left is a sidebar with navigation options like 'Login', 'Flashcards', and 'Admin'. The main area shows a 'Rich's Cabrillo College CIS 90 Calendar' with a table of lessons. Overlaid on this is a 'CCC Confer' window showing a video feed of 'Rich Simms' and a list of participants. A 'Google' window shows a map of the San Francisco Bay Area. Another window shows a PDF titled 'cis90lesson01.pdf - Adobe Acrobat Pro' with the title 'The CIS 90 System Playground'. A terminal window at the bottom right shows login prompts for 'Opus' and 'Cabrillo College'.

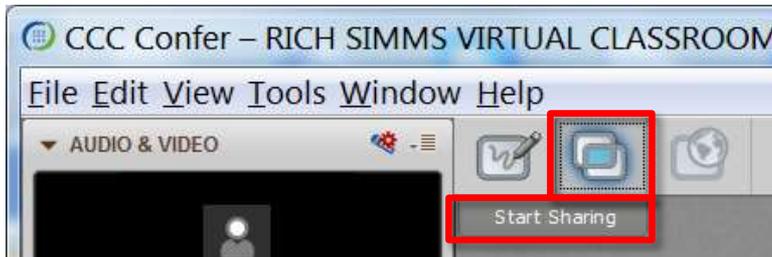
CIS 90 website Calendar page

One or more login sessions to Opus

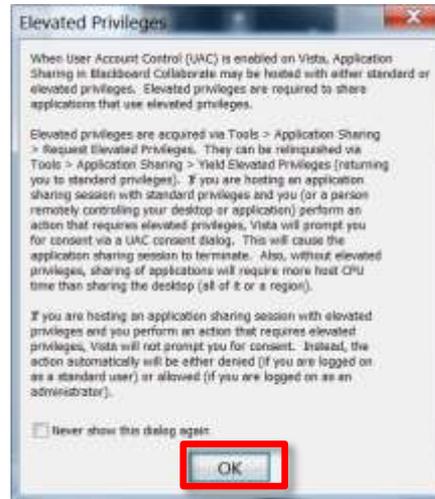


Student checklist for sharing desktop with classmates

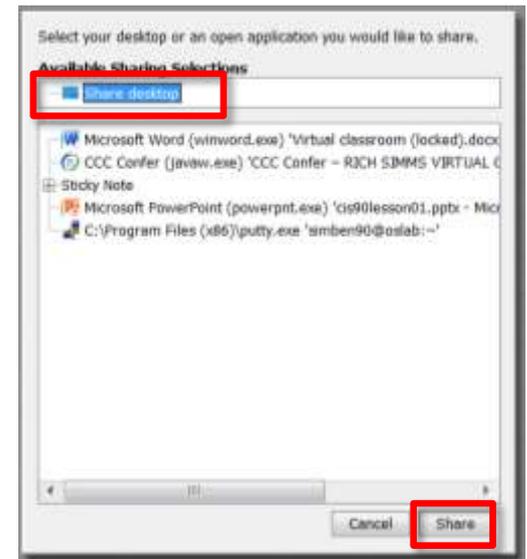
1) Instructor gives you sharing privileges



2) Click overlapping rectangles icon. If white "Start Sharing" text is present then click it as well.



3) Click OK button.



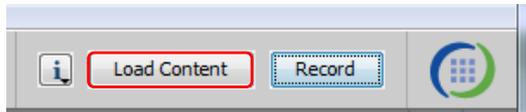
4) Select "Share desktop" and click Share button.



Rich's CCC Confer checklist - setup

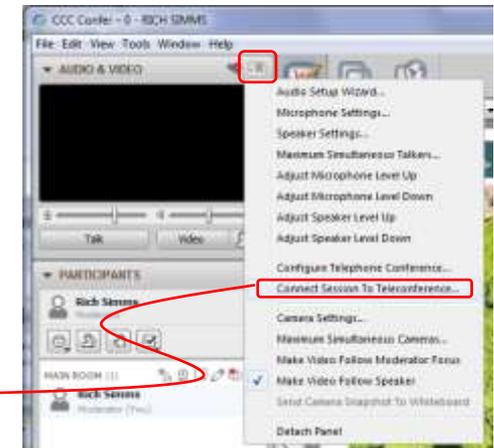


[] Preload White Board

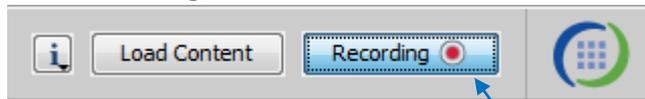


[] Connect session to Teleconference

Session now connected to teleconference



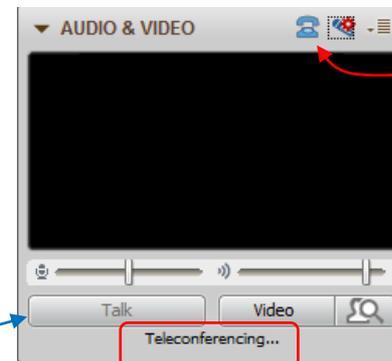
[] Is recording on?



Red dot means recording

[] Use teleconferencing, not mic

Should be greyed out



Should show as this live "off hook" telephone handset icon and the Teleconferencing ... message displayed



Rich's CCC Confer checklist - screen layout and share



Rich's CCC Confer checklist - screen layout and share

foxit for slides

chrome

putty

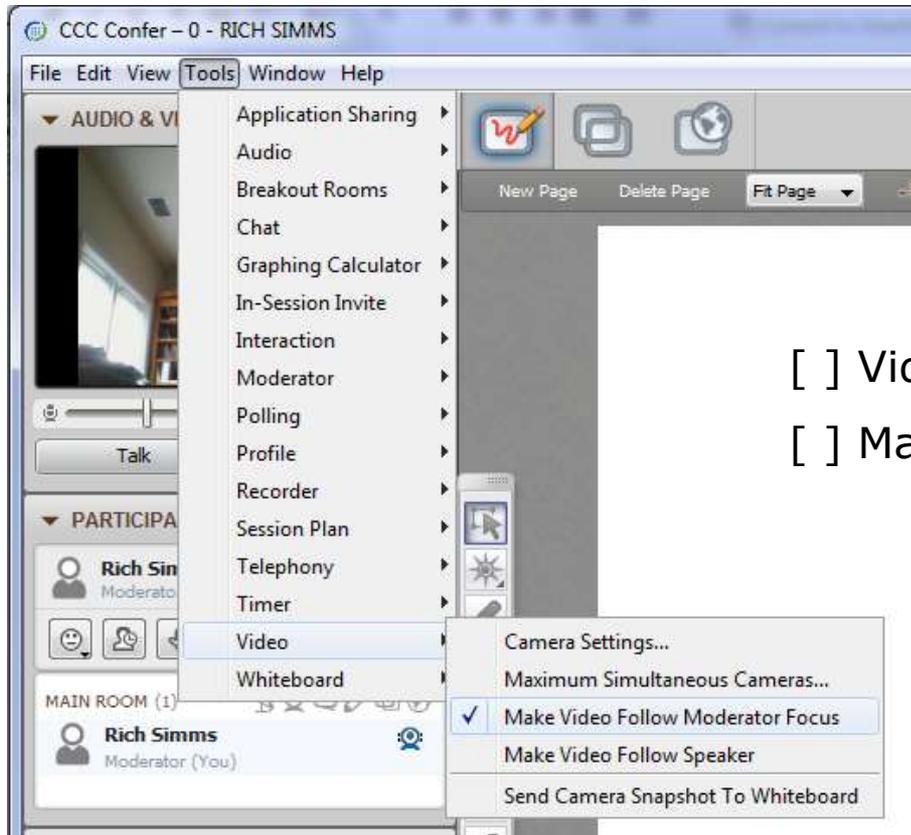
vSphere Client

[] layout and share apps





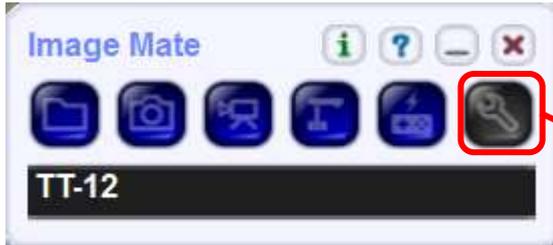
Rich's CCC Confer checklist - webcam setup



- [] Video (webcam)
- [] Make Video Follow Moderator Focus



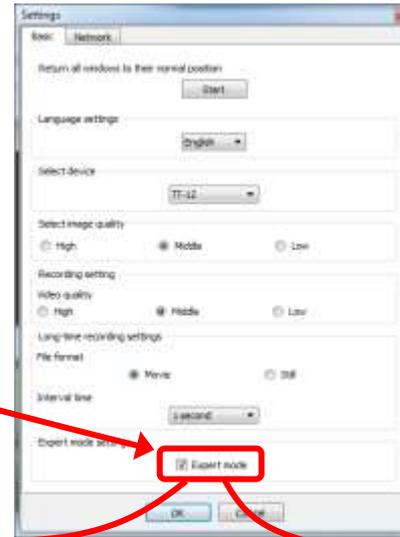
Rich's CCC Confer checklist - Elmo



Elmo rotated down to view side table



Run and share the Image Mate program just as you would any other app with CCC Confer



The "rotate image" button is necessary if you use both the side table and the white board.

Quite interesting that they consider you to be an "expert" in order to use this button!

Elmo rotated up to view white board



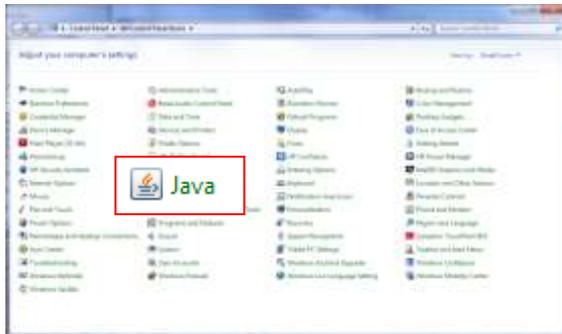


Rich's CCC Confer checklist - universal fix

Universal Fix for CCC Confer:

- 1) Shrink (500 MB) and delete Java cache
- 2) Uninstall and reinstall latest Java runtime
- 3) <http://www.cccconfer.org/support/technicalSupport.aspx>

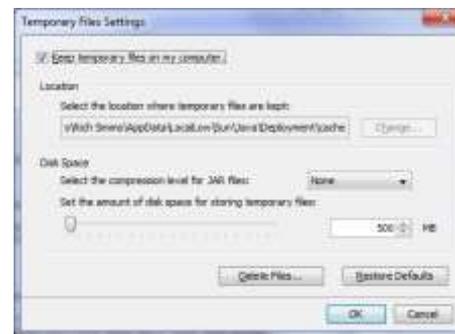
Control Panel (small icons)



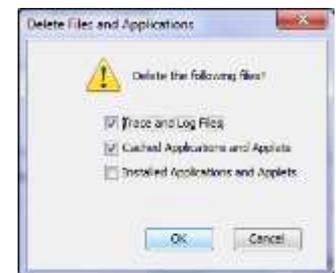
General Tab > Settings...



500MB cache size



Delete these



Google Java download





Start

Sound Check

*Students that dial-in should mute their line using *6 to prevent unintended noises distracting the web conference.*

*Instructor can use *96 to mute all student lines.*



Instructor: **Rich Simms**

Dial-in: **888-886-3951**

Passcode: **136690**



Chris



Jeremy



Jennifer



Cameron



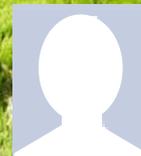
Joseph



Lisa



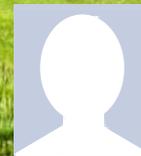
May



Sundance



Charlie



Sean



Brenda



Anthony



Will H.



Josh



Michael



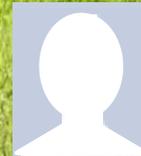
Danny



Vic



William D.



Taylor



Thomas



Stewart



Miguel



Jairo



Tony



Joaquin

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

First Minute Quiz

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

Use CCC Confer White Board

email answers to: risimms@cabrillo.edu

(answers must be emailed within the first few minutes of class for credit) 16

vi editor

Objectives

- Create and modify text files

Agenda

- Quiz
- Questions
- Test 2 Post Mortem
- Housekeeping
- grep workout
- Shell Six Steps (review)
- Signals (review)
- Target Practice
- Using &
- Job control (review)
- Load balancing & scheduling (review)
- Text editors
- vi 101
- vi
- Tangent on spell
- Assignment
- Wrap up



Questions



Questions?

Lesson material?

Labs? Tests?

How this course works?

- Graded work in home directories
- Answers in /home/cis90/answers

Who questions much, shall learn much, and retain much.

- Francis Bacon

If you don't ask, you don't get.

- Mahatma Gandhi

Chinese
Proverb

他問一個問題，五分鐘是個傻子，他不問一個問題仍然是一個傻瓜永遠。

He who asks a question is a fool for five minutes; he who does not ask a question remains a fool forever.



Test 2

Post Mortem

Test 2 – Results

Missed Q26 = 15
Missed Q4 = 14
Missed Q30 = 14
Missed Q29 = 14
Missed Q25 = 13
Missed Q24 = 12
Missed Q28 = 11
Missed Q23 = 11
Missed Q22 = 11
Missed Q21 = 11
Missed Q20 = 11
Missed Q19 = 11
Missed Q17 = 11
Missed Q13 = 11
Missed Q11 = 11

Missed Q9 = 10
Missed Q27 = 10
Missed Q18 = 10
Missed Q3 = 8
Missed Q2 = 8
Missed Q12 = 8
Missed Q15 = 6
Missed Q10 = 6
Missed Q8 = 5
Missed Q7 = 5
Missed Q6 = 5
Missed Q5 = 5
Missed Q14 = 5
Missed Q16 = 4
Missed Q1 = 0

Extra Credit

Missed Q33 = 13
Missed Q32 = 13
Missed Q31 = 14



*17 tests
submitted*



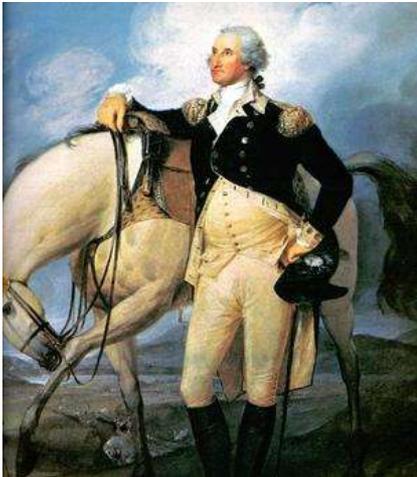
*7 tests not
submitted*





Q16) There is a file in the */etc* directory named *passwd*. This file has information on all user accounts including usernames, UIDs, first and last name, etc. What is the absolute pathname of this file?

Correct answer: */etc/passwd*



<http://www.sodahead.com/united-states/what-color-was-george-washingtons-white-horse/question-636725/>



<http://kids.britannica.com/comptons/art-55428/General-George-Washington-and-his-staff-welcoming-a-provision-train>



<http://www.mountvernon.org/content/revolutionary-war-princeton-white-horse>



Questions today on Test #2?

If not today, there will be a Sunday afternoon workshop in room 828, Nov 15, 2:00 PM till whenever, to work through every question on Test #2 in detail.

The goal is for each attendee to be able to handle any of the test questions in less than 30 seconds.

Housekeeping

1. Lab 8 due tonight

at 11:59pm

```
at> cat files.out bigshell > lab08
```

```
at> cp lab08 /home/rsimms/turnin/cis90/lab08.$LOGNAME
```

```
at> <Ctrl-D>
```

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

2. A **check8** script is available for Lab 8.

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.

5. You can still do the "your name in lights" extra credit activity. See forum for details.

Heads up on Final Exam

Test #3 (final exam) is **MONDAY** Dec 14 1-3:50PM

Monday	12/14	Test #3 (the final exam)	5 posts Lab X1 Lab X2
		Time <ul style="list-style-type: none"> MONDAY 1:00PM - 3:50PM in Room 828 Materials <ul style="list-style-type: none"> Test (blackboard) CCC Confer <ul style="list-style-type: none"> Enter virtual classroom Class archives 	

*Extra credit
labs and
final posts
due by
11:59PM*

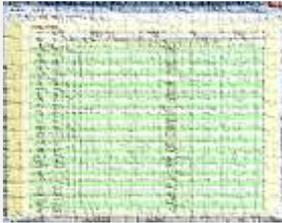
- All students will take the test at the same time. The test must be completed by 3:50PM.
- Working and long distance students can take the test online via CCC Confer and BlackBoard.
- Working students will need to plan ahead to take time off from work for the test.

Where to find your grades

Send me your survey to get your LOR code name.

The CIS 90 website Grades page

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



Points that could have been earned:

7 quizzes:	21 points
7 labs:	210 points
2 tests:	60 points
2 forum quarters:	40 points
Total:	331 points

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

At the end of the term I'll add up all your points and assign you a grade using this table

Or check on Opus

checkgrades *codename*
(where *codename* is your LOR codename)



Written by Jesse Warren a past CIS 90 Alumnus

grades *codename*
(where *codename* is your LOR codename)



Written by Sam Tindell a past CIS 90 Alumnus.
Try his tips, schedule and forums scripts as well!

Would you like some help learning Linux?



If you would like some additional come over to the CIS Lab. There are student lab assistants and instructors there to help you.

*Tess, Michael, and Paul are
CIS 90 Alumni.*

*Mike Matera is the other
Linux instructor.*

I'm in there Mondays.



grip 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 branch 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?

If so, 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-2.2.15-47.el6.centos.i686  
httpd-2.2.15-47.el6.centos.i686  
httpd-manual-2.2.15-47.el6.centos.noarch
```

Yes, version 2.2.15 has been installed

grep practice

Has the **mysql-server** package been installed on Opus?

If so, type the version of mysql-server 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
```

Look familiar? (lab 8) Shows how to compare shells by size and record the biggest one in a file.

```
[rsimms@oslab ~]$ ps -l
```

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
0	S	201	26146	26145	0	80	0	-	1700	-	pts/3	00:00:00	bash
0	S	201	26337	26146	0	80	0	-	1429	-	pts/3	00:00:00	ksh
0	S	201	26343	26337	0	80	0	-	524	-	pts/3	00:00:00	dash
0	S	201	26346	26343	0	80	0	-	1314	-	pts/3	00:00:00	sh
0	S	201	26348	26346	0	80	0	-	1332	-	pts/3	00:00:00	csh
0	R	201	26362	26348	9	80	0	-	1220	-	pts/3	00:00:00	ps

size
↓

```
[rsimms@oslab ~]$ ps -l | grep bash
```

```
0 S 201 26146 26145 0 80 0 - 1700 - pts/3 00:00:00 bash
```

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

```
[rsimms@oslab ~]$ cat bigshell
```

```
0 S 201 26146 26145 0 80 0 - 1700 - pts/3 00:00:00 bash 40
```

grep practice

Instructor note: add write permission to others on Benji's terminal device

- Run **bash**, **dash**, **ksh**, **sh** and **csch** shells and use **ps -l** to see which is the smallest.
- Redirect the line of **ps -l** output for the smallest shell to Benji Simms's terminal: **/dev/pts/??**
- Sign it with **echo "From *first name*" > /dev/pts/??**
- Then **exit** each shell till your are back to just one bash shell running.

grep usage – search inside files

How many CIS 90 user accounts are there?

```
/home/cis90/simben $ grep :190: /etc/passwd | wc -l  
35
```

```
/home/cis90/simben $ grep cis90 /etc/passwd | wc -l  
35
```

```
/home/cis90/simben $ grep "^.\{2,6\}90" /etc/passwd | wc -l  
35
```



There are 35

The third example is a "regular expression". For more information see the Resources page of the website.

grep practice

How many CIS 72 accounts are there on Opus?

Type the number of CIS 72 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
```

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 ":"

grep practice

Does your user ID in */etc/passwd* match the uid output by the **id** command?

*Type your answer (yes or no) and your uid from the **id** command into the chat window*

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

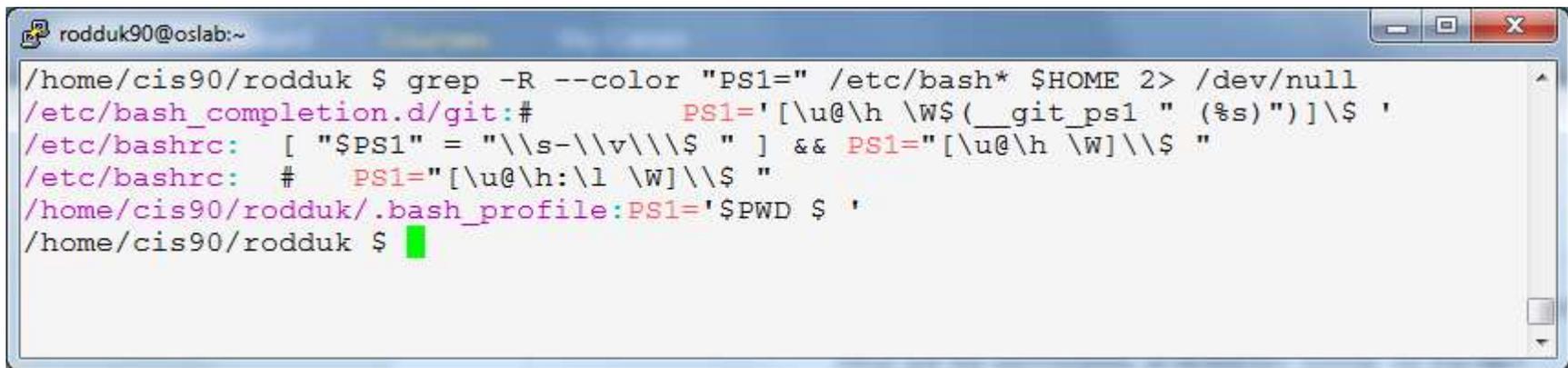
Where does the PS1 "prompt" variable get set?

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

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

This time in color using --color option



```
rodduk90@oslab:~  
/home/cis90/rodduk $ grep -R --color "PS1=" /etc/bash* $HOME 2> /dev/null  
/etc/bash_completion.d/git:#          PS1='[\u@\h \W$(__git_ps1 " (%s)")]\$ '  
/etc/bashrc:  [ "$PS1" = "\s-\v\\\$ " ] && PS1="[\u@\h \W]\$ "  
/etc/bashrc:  #    PS1="[\u@\h:\l \W]\$ "  
/home/cis90/rodduk/.bash_profile:PS1='$PWD $ '  
/home/cis90/rodduk $ █
```

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)

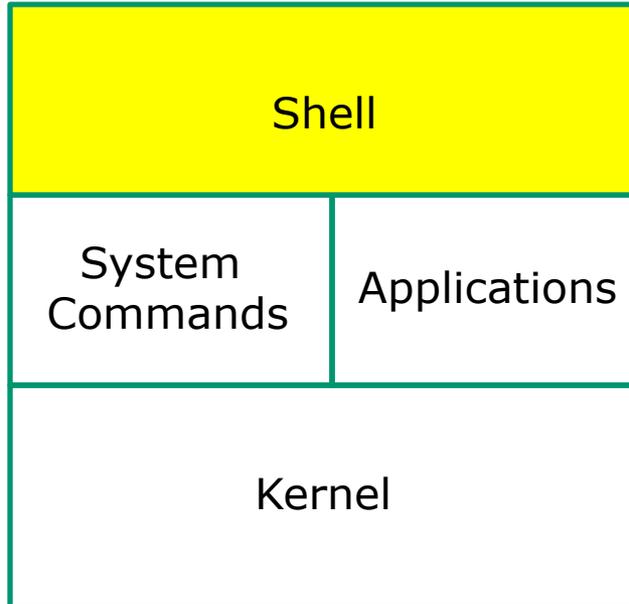
Example Command

```
/home/cis90/simben $ find / -name treat* 2> /dev/null  
/home/cis90/primic/treat1  
/home/cis90/juetay/treat1  
/home/cis90/porjos/treat1  
/home/cis90/beycha/bag/treat1  
/home/cis90/drydan/bag/treat1  
/home/cis90/rodduk/treat1  
/home/cis90/tosbre/treat1  
/home/cis90/remlis/treat1  
/home/cis90/linmay/treat1  
/home/cis90/brevic/treat1  
< snipped >  
/home/cis90/mcgcam/treat1  
/home/cis90/dulste/treat1  
/home/cis90/simben/bag/treat1  
/home/cis90/simben/treat1  
/home/cis90/locjer/treat1  
/home/cis90/neljjoa/treat1  
/home/cis90/johjos/treat1  
/home/cis90/watshe/treat1  
/home/cis90/hipmig/bag/treat1  
/home/cis90/hipmig/treat1  
/home/cis90/seasky/treat1  
/home/cis90/simben $
```

*Note: Benji has a file
named treat1 in his
home directory*



Prompt Step



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





Prompt Step

(uses PS1 variable)

`/home/cis90/simben $`

bash using your PS1 variable creates and outputs your prompt which is written to your terminal device

- Benji is using the bash shell. There are many other shells such as sh, ksh and csh. In `/etc/passwd` the last field in the line for his account determines the shell that is run when logging in.
- The bash program resides in the `/bin` directory.
- The command prompt appearance is defined by the PS1 variable. You can output a prompt yourself using **echo \$PS1**

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

/home/cis90/simben $ ls -l /bin/bash
-rwxr-xr-x. 1 root root 874248 May 10 2012 /bin/bash
```



Prompt Step

*Note there is an invisible
<newline> metacharacter at
the end of the command*

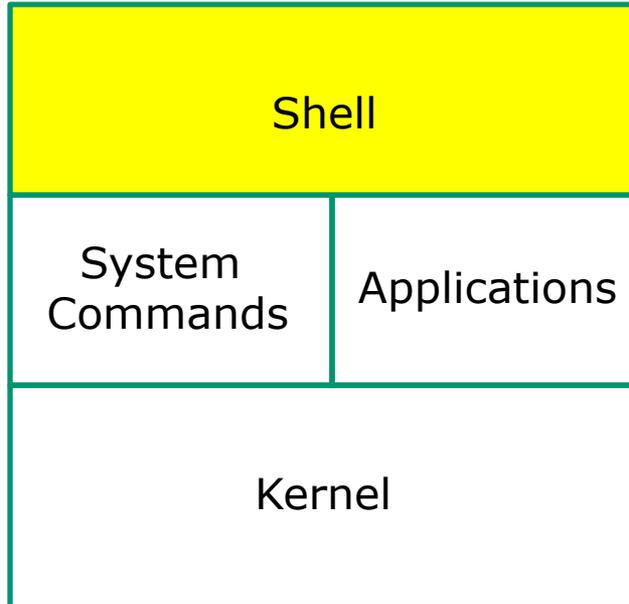
```
/home/cis90/simben $ find / -name treat* 2> /dev/null
```

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

The prompt step is not complete until the user presses the Enter/Return key



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.

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:

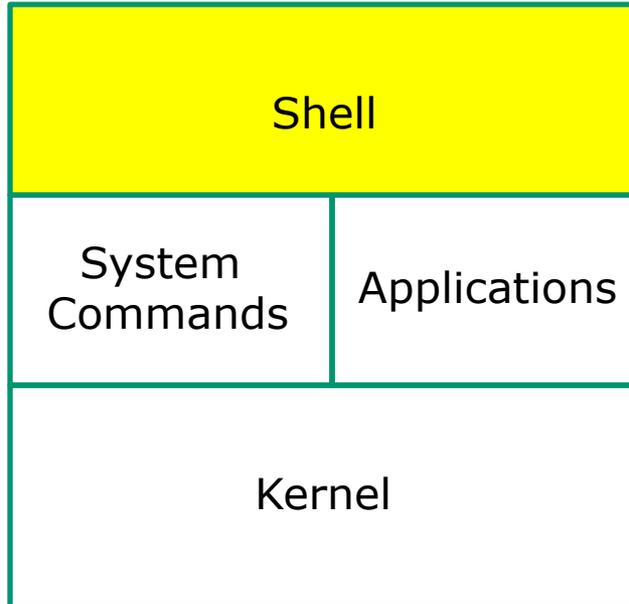
Connect **stderr** to **/dev/null** (the "bit bucket")

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

(uses PATH variable)

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

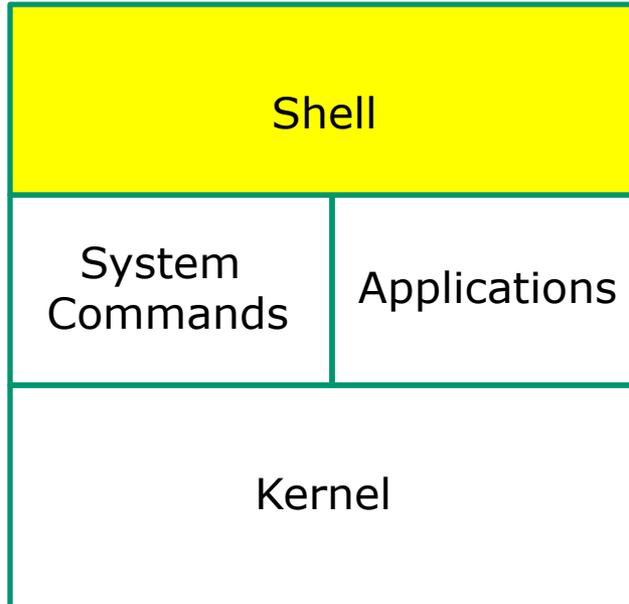
- 1st directory searched: /usr/lib/qt-3.3/bin
- 2nd directory searched: /usr/local/bin
- 3rd directory searched: **/bin**
- 4th directory searched: /usr/bin
- 5th directory searched: /usr/local/sbin
- 6th directory searched: /usr/sbin
- 7th directory searched: /sbin
- 8th directory searched: /home/cis90/simben/../../bin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

The shell locates the find command in the /bin directory

```
/home/cis90/simben $ echo $PATH
/usr/lib/qt-3.3/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/cis90/simben/../../bin:/home/cis90/simben/bin:
/home/cis90/simben $ type find
find is /bin/find
```



Execute Step

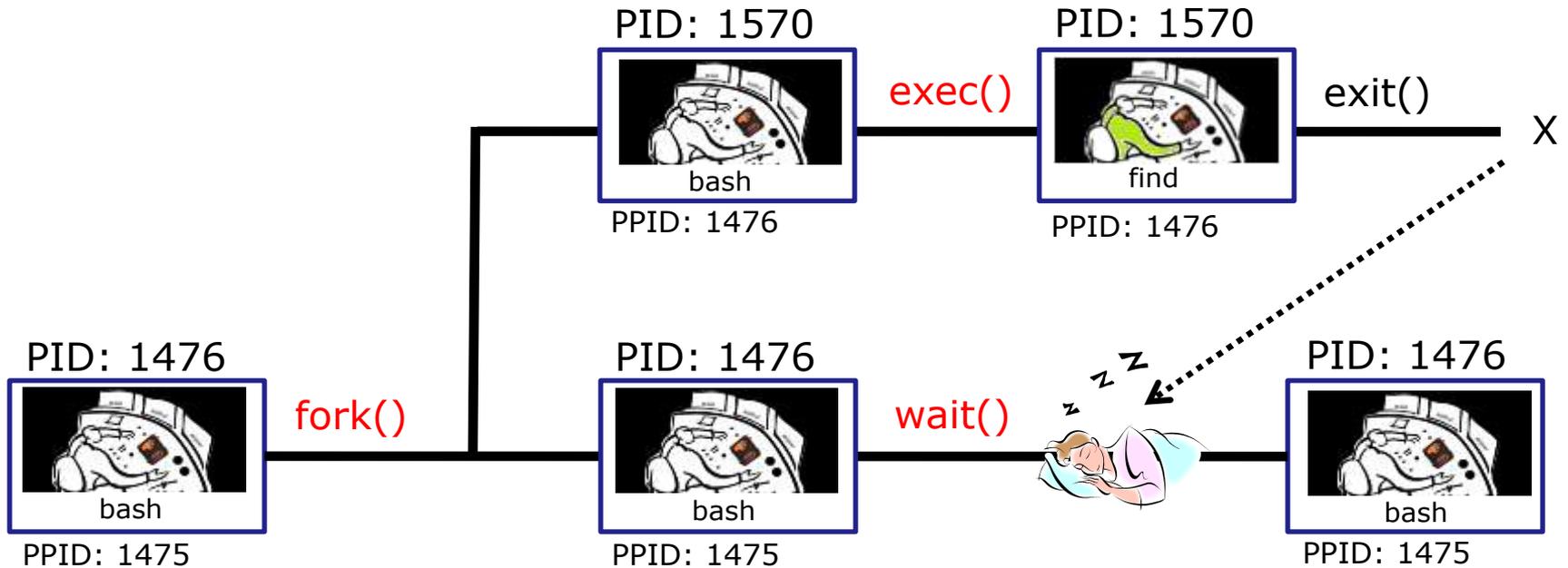


- 1) Prompt
- 2) Parse
- 3) Search
- 4) Execute**
- 5) Nap
- 6) Repeat





Execute Step



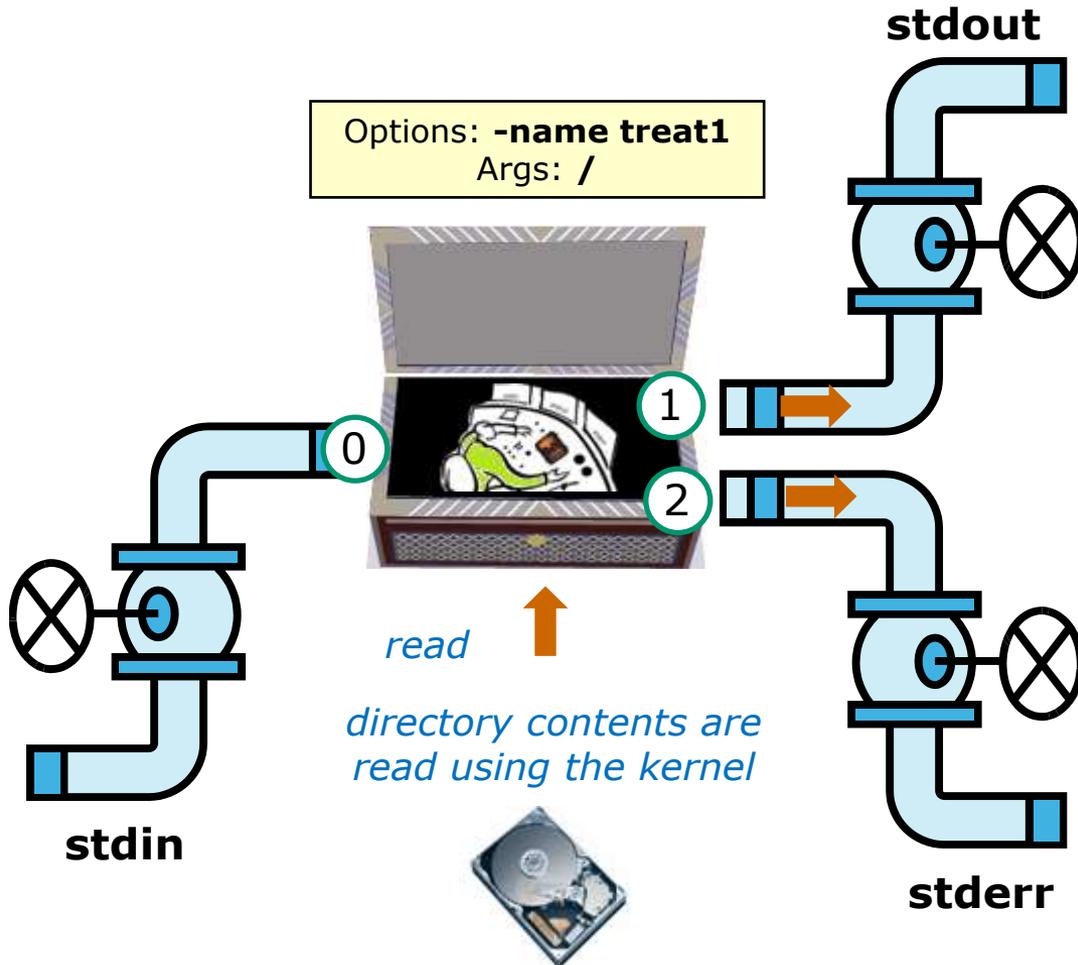
bash executes the **find** command by:

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



Execute Step

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



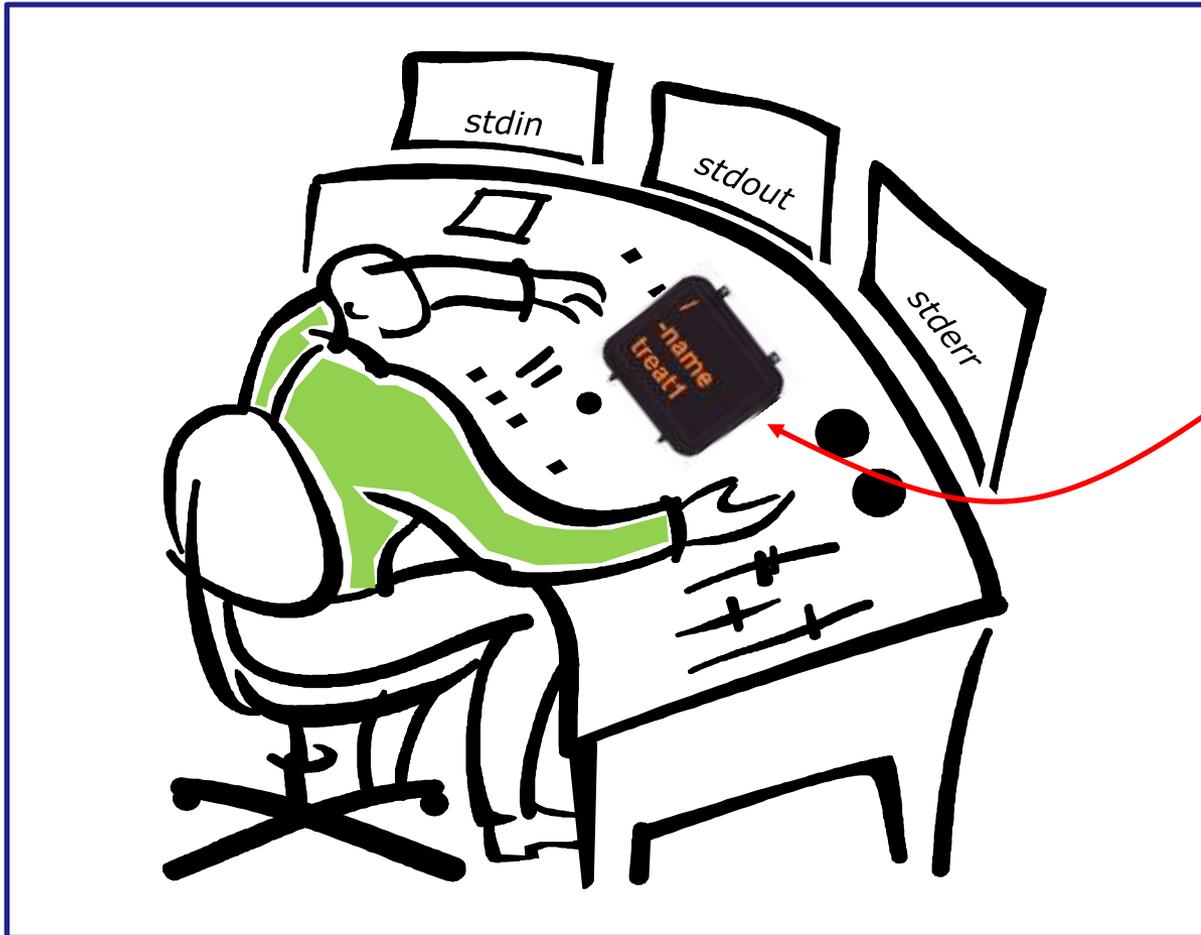
```
/home/cis90/primic/treat1
/home/cis90/juetay/treat1
/home/cis90/porjos/treat1
/home/cis90/beycha/bag/treat1
/home/cis90/drydan/bag/treat1
/home/cis90/rodduk/treat1
/home/cis90/tosbre/treat1
/home/cis90/remlis/treat1
/home/cis90/linmay/treat1
/home/cis90/brevic/treat1
< snipped >
```

/dev/null



```
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



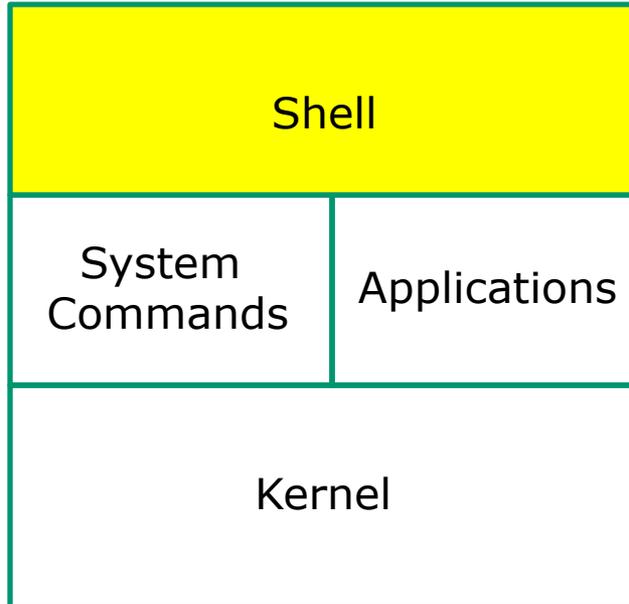
A process:

- 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 is running



Nap Step

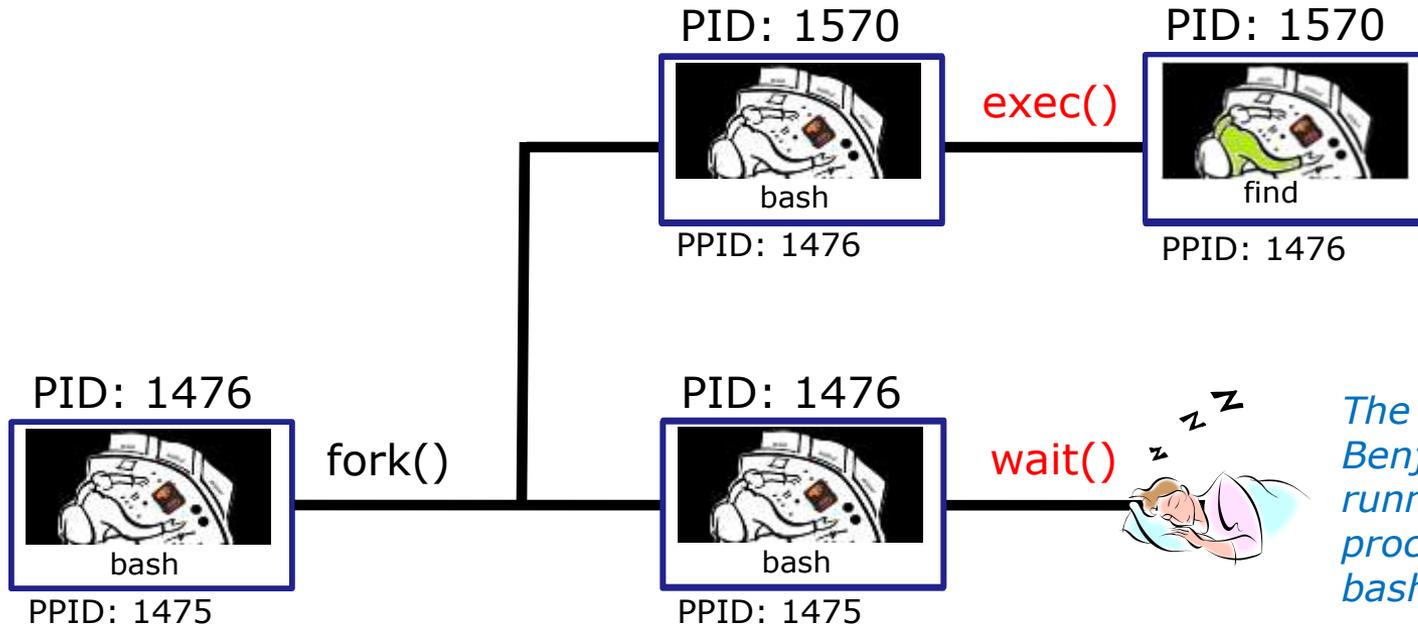


- 1) Prompt
- 2) Parse
- 3) Search
- 4) Execute
- 5) Nap**
- 6) Repeat





Nap Step



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

Sleeping

```
[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

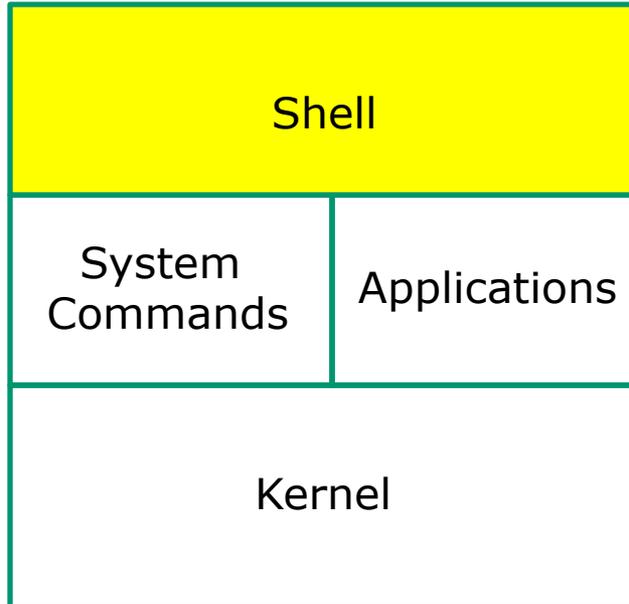
Parent

Child

Running



Repeat Step

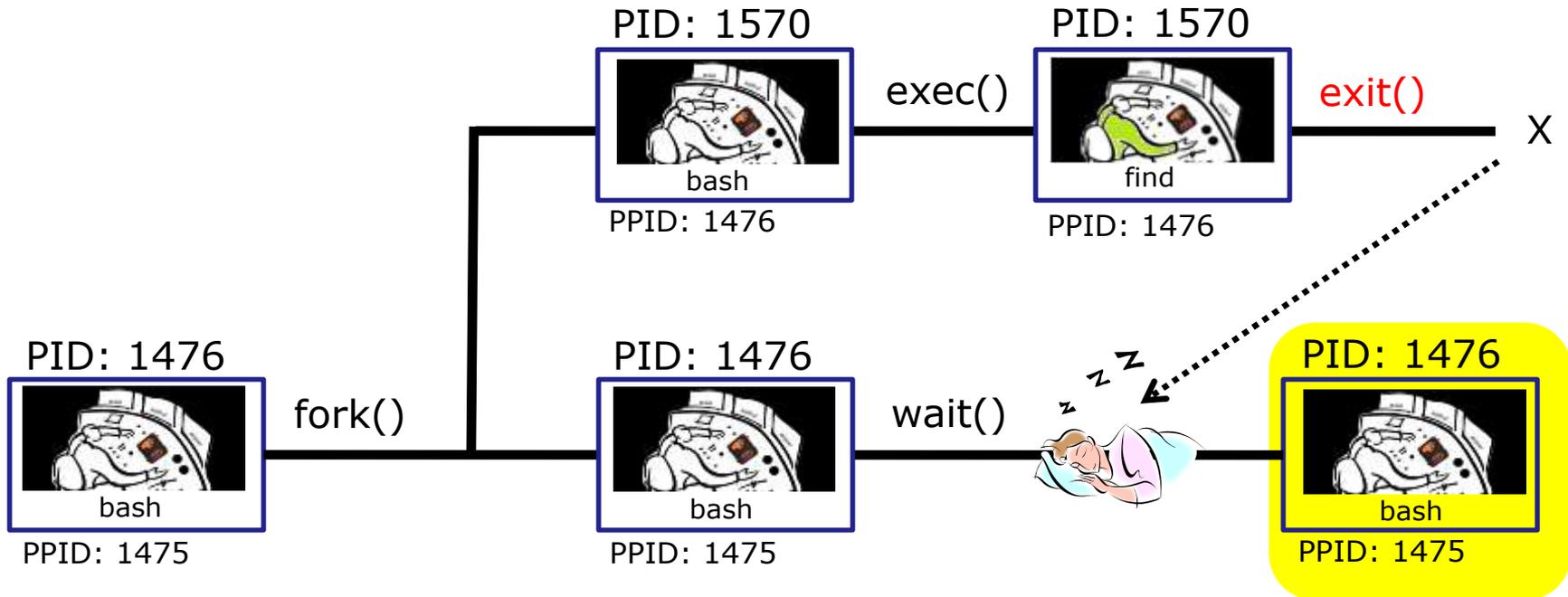


- 1) Prompt
- 2) Parse
- 3) Search
- 4) Execute
- 5) Nap
- 6) Repeat**





Repeat Step



The child process makes an **exit()** system call when it has finished. The parent bash process wakes up, the child process is killed and we are ready to start the process all over again with the next command.

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)
- Start a second login session to observe your processes
- Write your grep PID and status into the chat window when done

/home/cis90/simben \$ **grep -r "pototo" /usr**

```

simben90@oslab:~$
/home/cis90/simben $ grep -r "pototo" /usr/lib /usr/src
grep: /usr/lib/audit: Permission denied
/usr/lib/perl5/Net/DNS/Resolver/Recurse.pm: Purpose: Do that "hot pototo dance"
on args.
grep: /usr/lib/cups/backend/serial: Permission denied
grep: /usr/lib/cups/backend/ipp: Permission denied
grep: /usr/lib/cups/backend/http: Permission denied
grep: /usr/lib/cups/backend/dnsd: Permission denied
grep: /usr/lib/cups/backend/lpd: Permission denied
grep: /usr/lib/cups/backend/mdns: Permission denied
grep: /usr/lib/cups/backend/https: Permission denied
/home/cis90/simben $
  
```

/home/cis90/guest \$ **ps -lu simben90**

```

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

UID	PID	PPID	C	PRI	NI	ADDR	SI	WCHAN	TTY	TIME	CMD	
5	S	1001	8841	8820	0	80	0	-	2899	?	00:00:00	sshd
0	S	1001	8842	8841	0	80	0	-	1308	?	00:00:00	bash
0	D	1001	9032	8842	21	80	0	-	1369	?	00:00:02	grep

```

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

UID	PID	PPID	C	PRI	NI	ADDR	SI	WCHAN	TTY	TIME	CMD	
4	S	1001	6283	6270	0	80	0	-	1308	?	00:00:00	bash
5	S	1001	8841	8820	0	80	0	-	2899	?	00:00:00	sshd
0	S	1001	8842	8841	0	80	0	-	1308	?	00:00:00	bash
0	D	1001	9032	8842	21	80	0	-	1369	?	00:00:02	grep

```

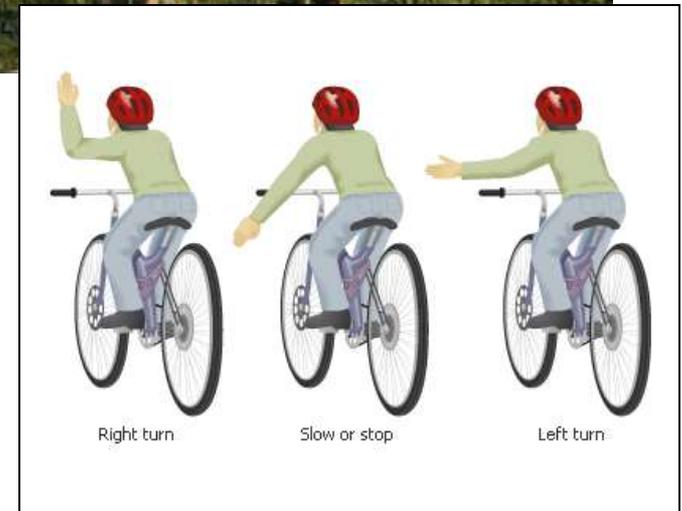
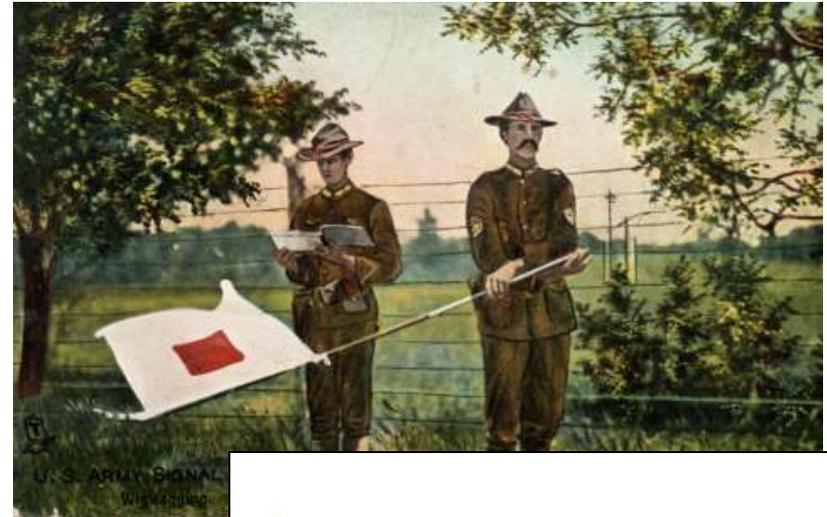
/home/cis90/guest $
  
```

Signals (Review)

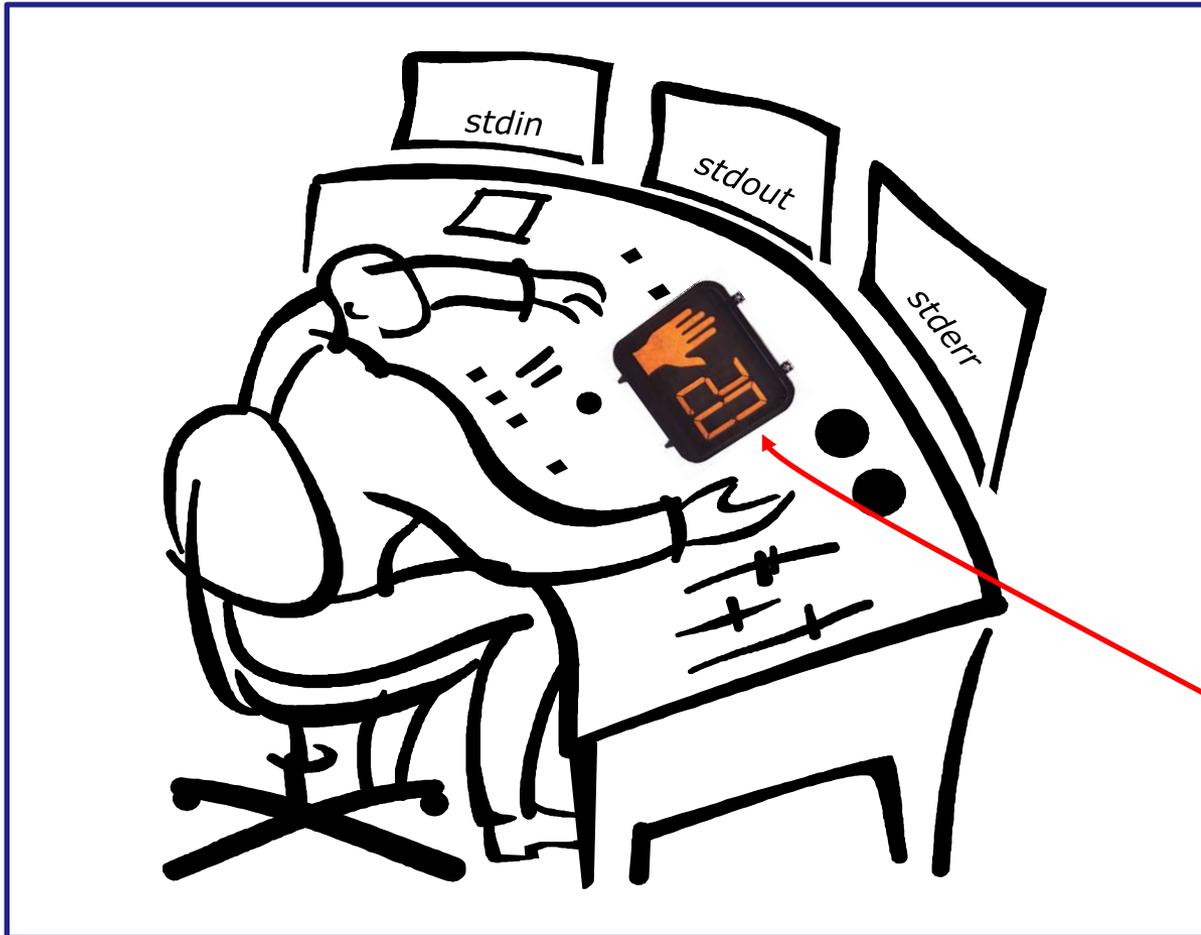
Signals

PLATE 4

COMMERCIAL CODE SIGNALS		
<p>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.</p>		
URGENT & IMPORTANT SIGNALS		COMPASS SIGNALS 3 FLAGS
<p>CODE FLAG OVER 1 FLAG OR 2 FLAG SIGNALS</p> <p>CODE FLAG P: "I Am about to Sail"</p> <p>A: "Do Not"</p> <p>C: "abandon the Vessel"</p>		<p>A: N 1/2 E</p> <p>K: S 3/4 W</p> <p>X: S 3/4 W</p>
LATITUDE & LONGITUDE SIGNALS		CODE FLAG OVER 2 FLAGS
<p>CODE FLAG A: 12° Latitude North Latitude</p> <p>O: 12° Latitude North Latitude</p> <p>Q: General Signal</p> <p>H: OR H</p> <p>X: OR X</p>		<p>CODE FLAG E: 23° Longitude East Longitude</p> <p>H: OR H</p> <p>Z: OR Z</p> <p>Q: General Signal</p> <p>Y: OR Y</p>
NUMERAL TABLE	GENERAL VOCABULARY	GEOGRAPHICAL SIGNALS ALPHABETICAL ORDER
<p>CODE FLAG UNDER 2 FLAGS</p> <p>Y: 10,000</p> <p>S: 10,000</p> <p>CODE FLAG: 10,000</p>	<p>3 FLAG SIGNAL</p> <p>I: Tons of Coal</p> <p>X: Tons of Coal</p> <p>K: Tons of Coal</p>	<p>4 FLAG SIGNAL</p> <p>A: Glasgow, Scotland.</p> <p>E: Glasgow, Scotland.</p> <p>Y: Glasgow, Scotland.</p> <p>Z: Glasgow, Scotland.</p>
ALPHABETICAL SPELLING TABLE		NAMES OF VESSELS FROM CODE LIST
<p>SPELLING SIGNAL</p> <p>J: John</p> <p>O: John</p> <p>H: John</p> <p>N: John</p> <p>C: Abb</p> <p>B: Abb</p> <p>D: Abb</p> <p>N: Abb</p> <p>C: at</p> <p>S: at</p> <p>F: at</p> <p>P: at</p>		<p>4 FLAG SIGNAL</p> <p>H: Graysa of Glasgow</p> <p>C: Graysa of Glasgow</p> <p>L: Graysa of Glasgow</p> <p>B: Graysa of Glasgow</p>



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:



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.



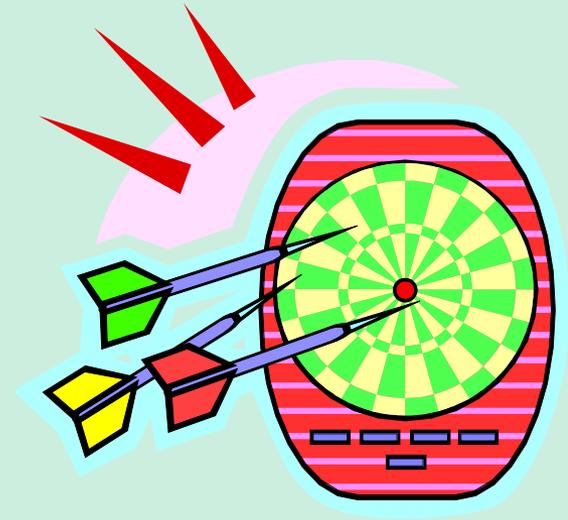
Using special keystrokes

- 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 ...*

*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

The screenshot shows a Linux desktop environment. In the foreground, a terminal window titled 'cis90@eko: ~' is open. The command 'firefox' is entered at the prompt and is highlighted with a red box. To the left of the terminal, a yellow text box contains the following text: *After running Firefox in the foreground it's not possible to enter more commands until Firefox is closed*. In the background, a Mozilla Firefox browser window is open, displaying the 'Ubuntu Start Page' at the URL 'http://start.ubuntu.com/10.04/'. The browser window shows the Ubuntu logo, the Google logo, and a search bar. The system tray at the bottom of the screen shows the terminal window, the browser window, and the Update Manager icon.

Job Control

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

The screenshot displays a terminal window and a Firefox browser window. In the terminal, the command `firefox &` is entered and highlighted with a red box. The output of the `ps` command is shown below:

```

cis90@eko:~$ ps
  PID TTY          TIME CMD
 1370 pts/0    00:00:00 bash
 1465 pts/0    00:00:00 firefox
 1470 pts/0    00:00:00 run-moz
 1474 pts/0    00:00:01 firefox
 1489 pts/0    00:00:00 ps
cis90@eko:~$

```

A blue text box contains the following text: *After running Firefox in the background, it is still possible to enter more commands.*

The Firefox browser window shows the Ubuntu Start Page with the URL `http://start.ubuntu.com/1` and a search bar.

& append to a command to run it in the background

Example 1

```
/home/cis90/simben $ grep -r pototo /usr /opt 2> /dev/null
```

 **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/simben $ grep -r pototo /usr /opt 2> /dev/null &
```

```
[1] 21175
```

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

```
Tue Apr 15 14:43:09 PDT 2014
```

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



Job Control (Review)

Job Control

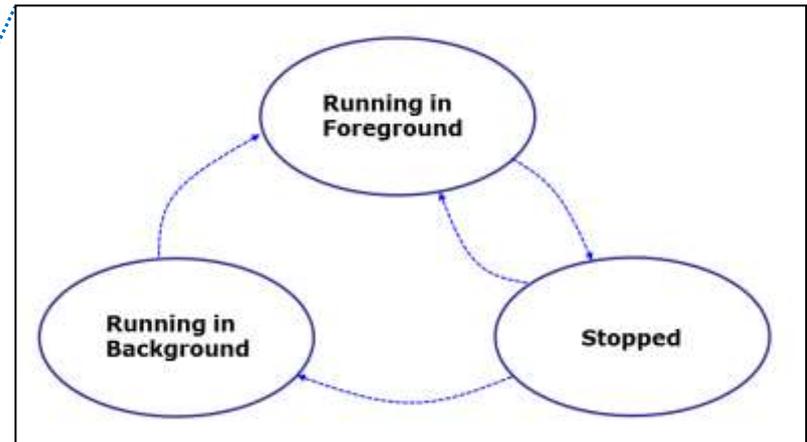
A feature of the bash shell

&	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

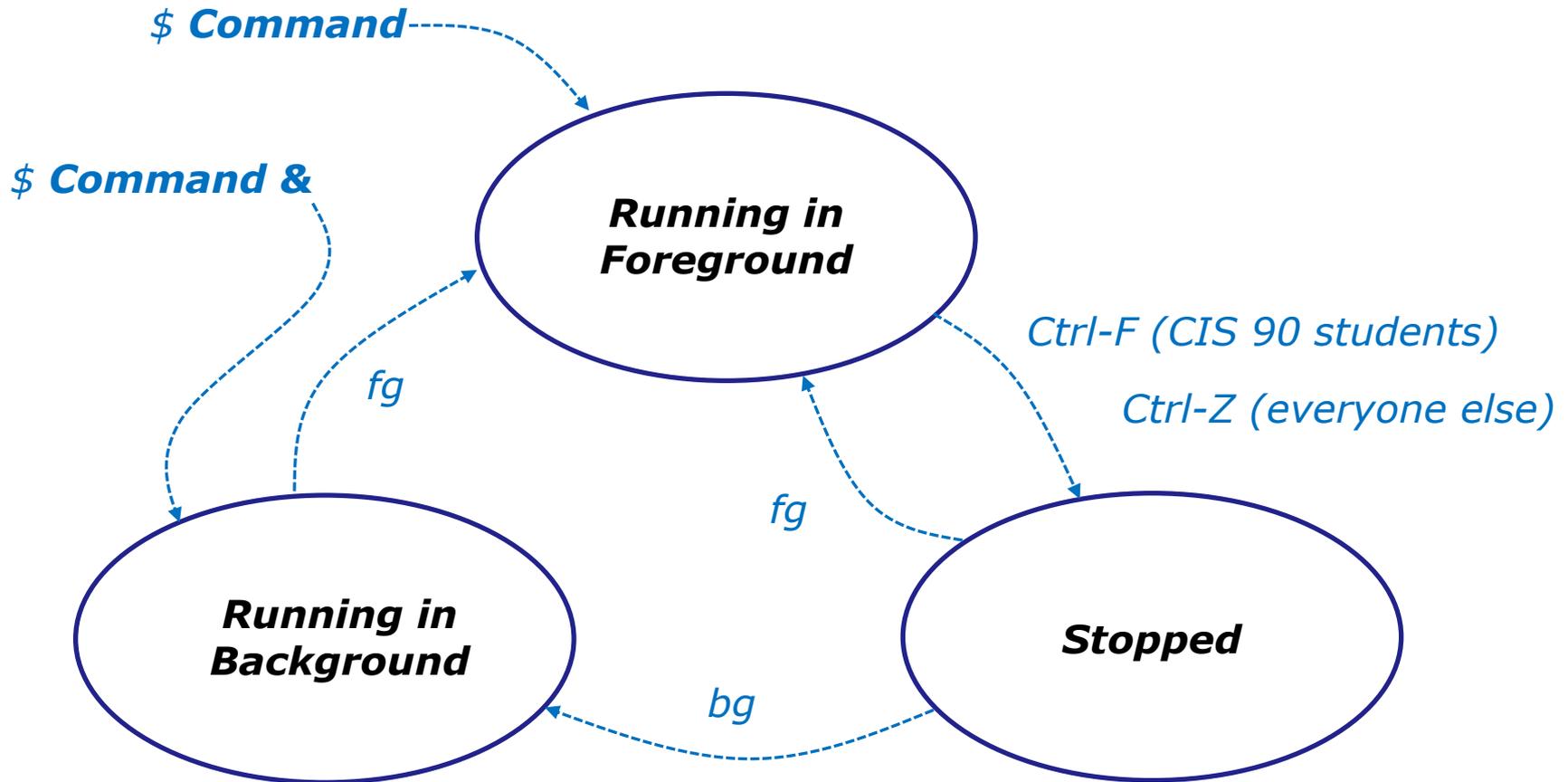
*Use **jobs**, **bg**, **fg** to list and resume jobs in the foreground or background*

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 s**T**opped*

Job Control Managing jobs

```
/home/cis90ol/simmsben $ bg 2 Let's resume job 2 in the background
```

```
[2]- sleep 110 &
```

```
/home/cis90ol/simmsben $ jobs
```

```
[1]- Stopped sleep 120
```

```
[2] Running sleep 110 &
```

```
[3]+ Stopped sleep 100
```

```
/home/cis90ol/simmsben $ bg 1 Let's resume job 1 in the background
```

```
[1]- sleep 120 &
```

```
/home/cis90ol/simmsben $ jobs
```

```
[1] Running sleep 120 &
```

```
[2]- Running sleep 110 &
```

```
[3]+ Stopped sleep 100
```

```
/home/cis90ol/simmsben $ fg 3 Let's resume job 1 in the foreground
```

```
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!*



Load Balancing & Scheduling (Review)

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 11/25/2014

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 $ atrm 24
/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/x
```

These should match

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

Type what happens in the chat window:



text editors

There are lots of text editors ...

Windows

notepad
notepad++
textpad

Text editors and word processors are different!

Mac

TextWrangler

- *Word processors are used by many different people to create documents containing text and graphics.*

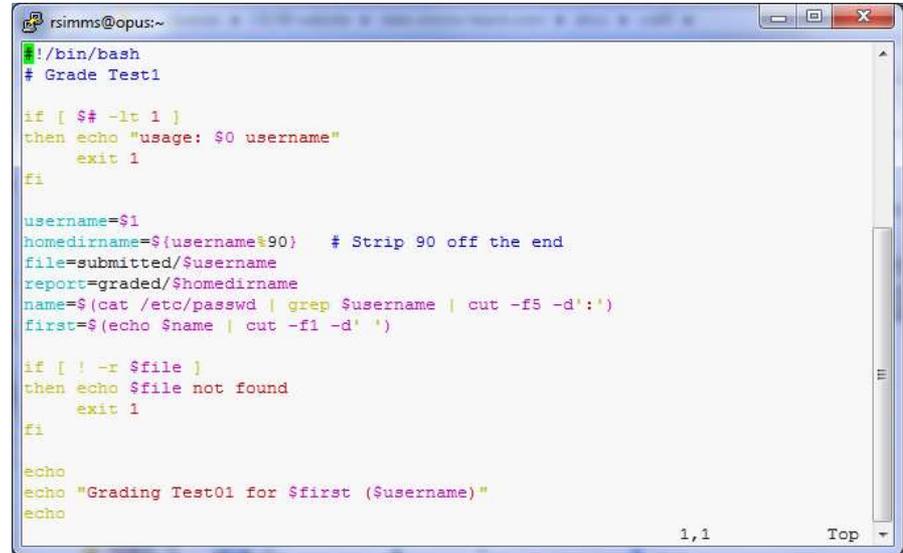
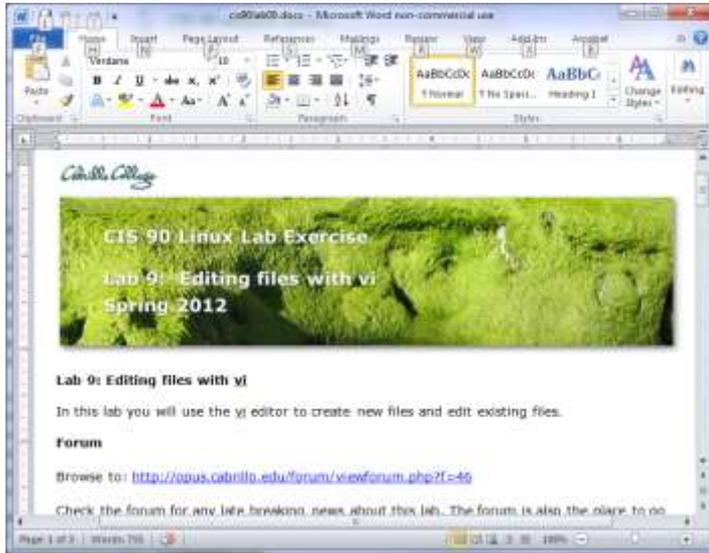
Linux

gedit
emacs
nano
vi
jove

- *Text editors are used by programmers to develop software and web designers to create web sites.*



Thanks Maria!



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:

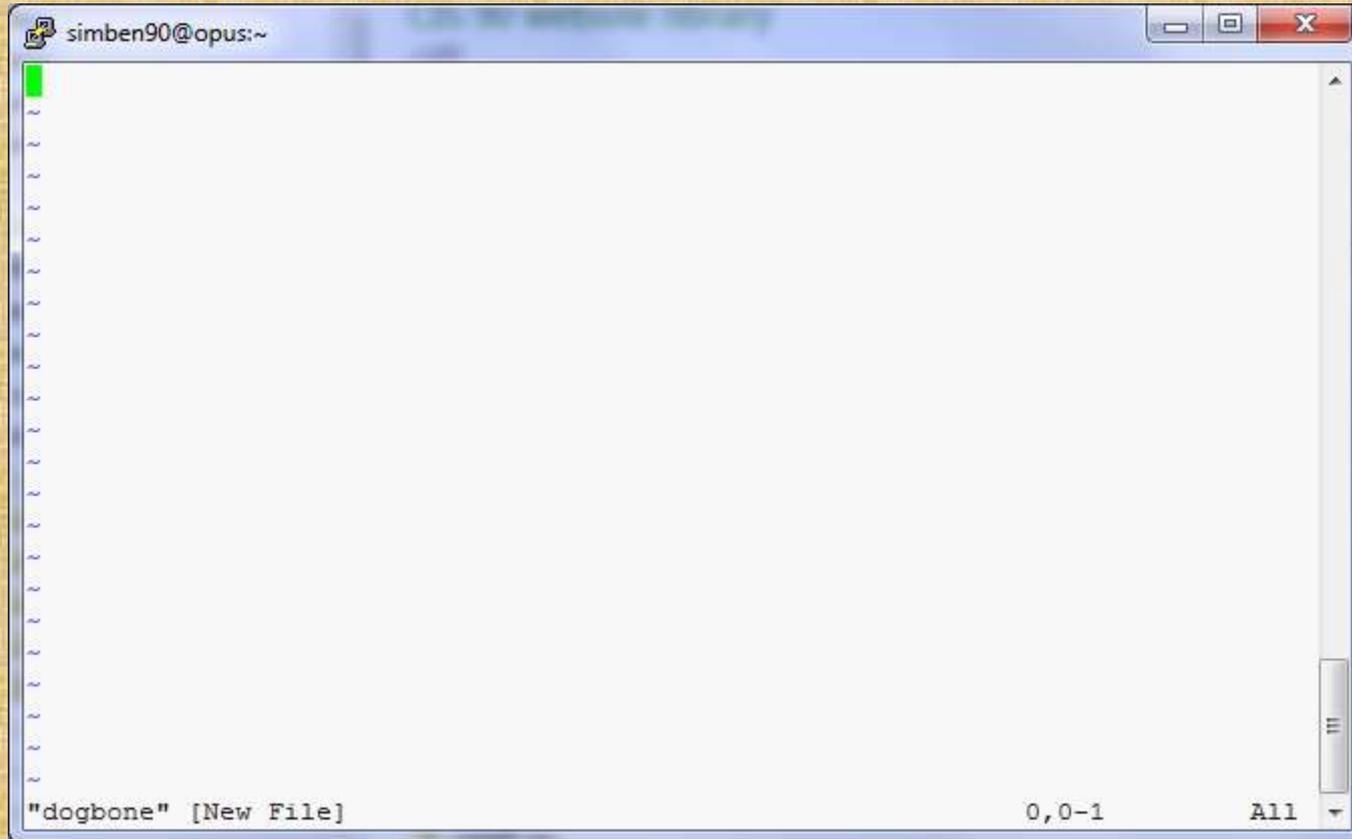
- The original vi code was written by Bill Joy for BSD Unix
- Bill Joy co-founded Sun Microsystems in 1982
- vi (for "visual")
- vim is an enhanced version of vi

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

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

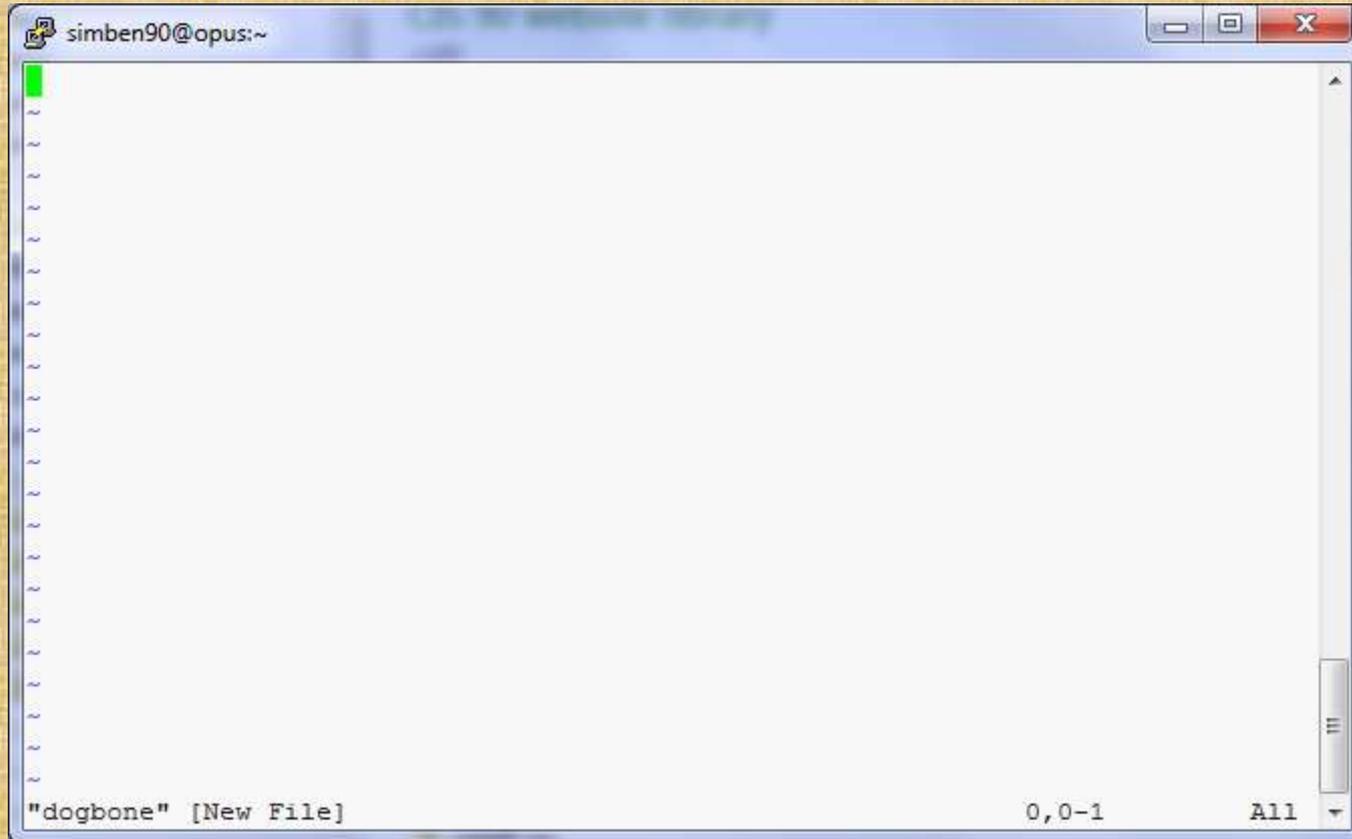
Type this

See this ...



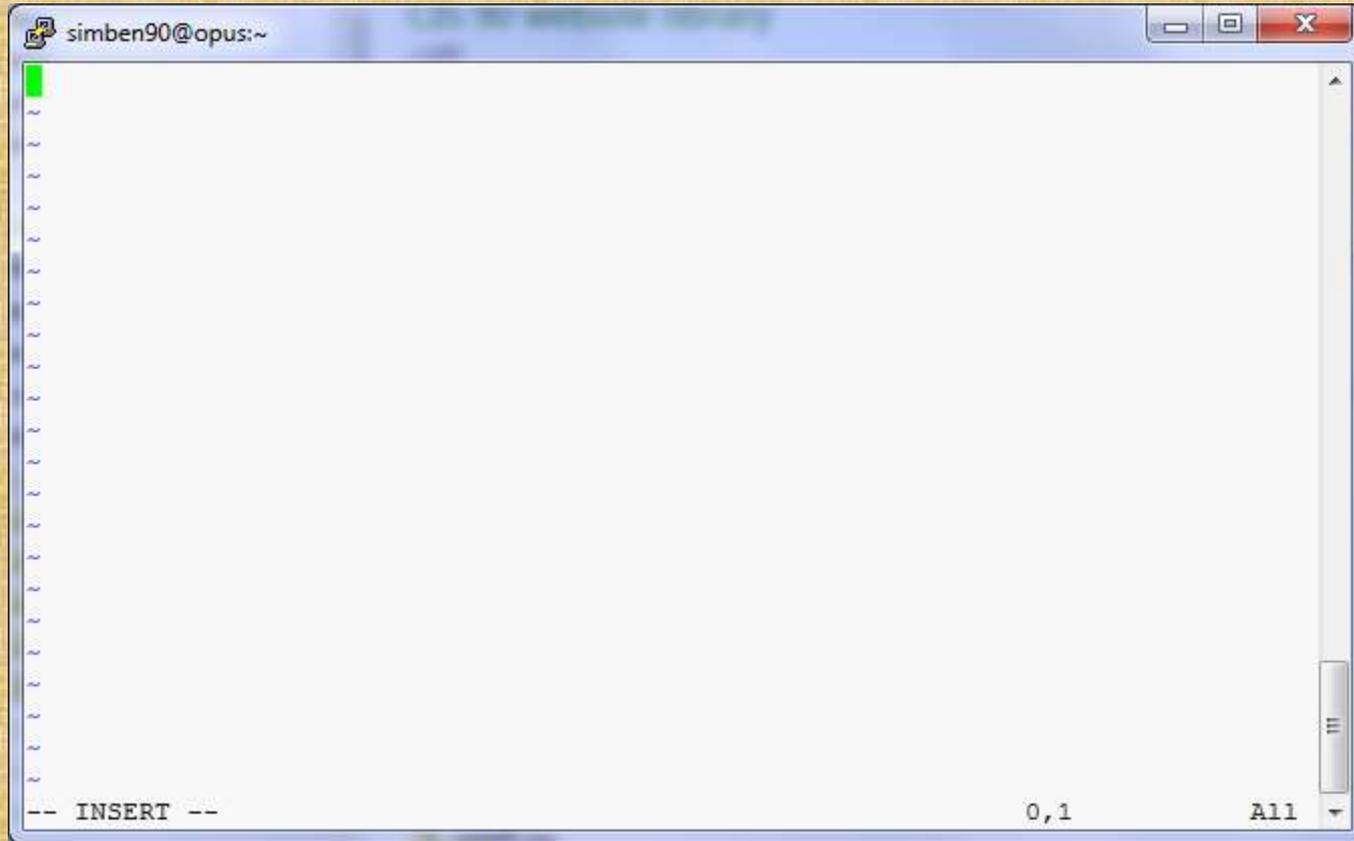
Take your hands OFF THE MOUSE – don't use it in vi!

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!



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



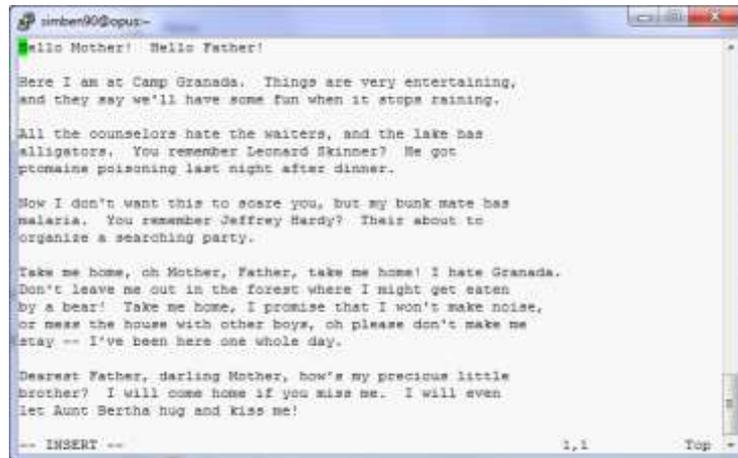
i

esc

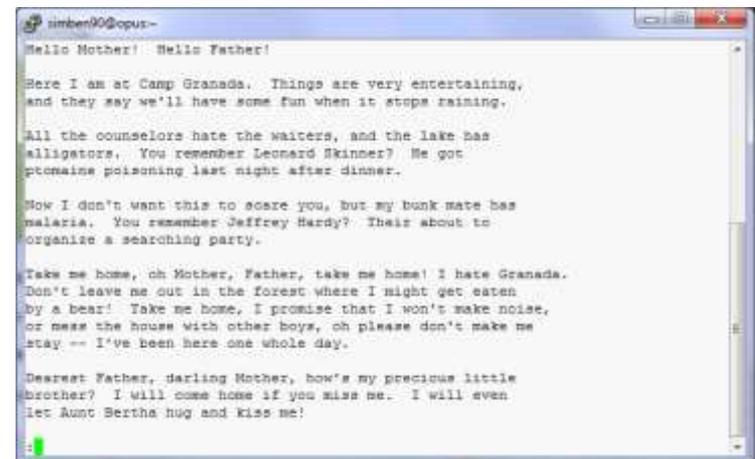
:

esc

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

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

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

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 Quitting

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*

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

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

Use vi to edit your *edits/text.err* file

```
This is line number1.  
This is line number 1.  
Thi sis line line number 2.  
his is line number3.line number3.  
This is This is line #4.  
this number5 is line .  
Here is line number      6.  
This is lamw number      7.  
Thi is line number9.  
This is line  
number10.
```



```
This is line number 1.  
This is line number 2.  
This is line number 3.  
This is line number 4.  
This is line number 5.  
This is line number 6.  
This is line number 7.  
This is line number 8.  
This is line number 9.  
This is line number 10.
```

Copy your corrected file into the chat window when finished

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

The screenshot shows a web browser window displaying the main page of the Vim Tips wiki. The browser's address bar shows the URL http://vim.wikia.com/wiki/Main_Page. The page layout includes a top navigation bar with options like 'Create a new wiki', 'Log in', and 'Create an account'. Below this is a green header with 'Edit this page', 'History', and 'Share this article' buttons, along with 'Article' and 'Discussion' tabs. The main content area is titled '(Redirected from Main Page)' and features a large banner for the TV show 'happy town' with the text 'Don't let the name fool you.' and 'PREMIERES TONIGHT 10|9c abc'. To the left of the main content is a sidebar with a search box, navigation links (Home, Community portal, To do, Top content, Folding category, Searching category), and a 'Latest activity' section showing 1,592 articles on this wiki. The right sidebar contains an 'Information' section with links to documentation, scripts, and news, and a 'Featured tip for April' section with the text: 'We all know what a tab page is – each tab holds a different file. Actually, Vim tabs are *not* like that. Instead, tab pages in Vim offer much more flexibility, and provide many features that are unavailable with standard editors.'

The Mug of vi

The image shows a Mozilla Firefox browser window displaying the website <http://nostarch.com/mug.htm>. The page features a white mug with the text "THE MUG OF VI" and a list of vi editor commands. A smaller window shows a detailed view of the mug's text, which includes:

THE MUG OF VI

FILE COMMANDS	DELETING / INSERTING TEXT	MOVING AROUND
<code>vi filename(s)</code> edit a file or files	<code>dd, dd, k</code> delete word, line, character	<code>0</code> go to beginning of line (next)
<code>vi -r filename</code> review saved file after crash	<code>dd, nx</code> delete n lines, n characters	<code>), (</code> move to next, previous sentence
<code>ZZ, ZZ, OX</code> save and exit	<code>x, X</code> delete character forward, backward	<code>}, {</code> move to next, previous paragraph
<code>q, q!</code> quit; quit without saving	<code>D, d\$</code> delete to end of line	<code>W, B</code> move forward, back one word
<code>q, :w, :wq</code> save file, save file as file	<code>db</code> delete from cursor to window (\$, 0, etc.)	<code>e</code> go to end of current or next word
<code>re filename</code> edit filename	<code>>>, <<</code> indent, outdent line	CUT / COPY / PASTE
<code>! filename</code> insert filename	<code>S</code> replace text with blank line	<code>yy, yy</code> copy n lines
<code>!sh</code> drop to shell	<code>a, o</code> insert new line below, above current line	<code>P, P</code> paste next after, before cursor
<code>!cmd</code> run command cmd	<code>u</code> undo last change	<code>A, I</code> insert text after, beginning of line
<code>!cmd</code> execute cmd and insert output	<code>r</code> repeat last change	WICKED COOL STUFF
SEARCH AND REPLACE	<code>~</code> change case	<code>xp</code> transpose characters
<code>/pat, /Pat</code> find pat forward or backward	<code>!&</code> combine current line with next	<code>PP</code> create a mark called p
<code>?pat</code> find next line that starts with pat	<code>o</code> return to p	<code>q</code> delete, copy text from mark to cursor
<code>n, N</code> repeat last search backward, forward	<code>A, I, k, j</code> left, right, up, down one character	<code>!&w, y'x</code> delete, copy text from mark to cursor
<code>R</code> replace text from current character	<code>SB, sb</code> left or right n words	<code>!&n</code> indent n lines
	<code>CTRL-B, F</code> back, forward one screen	
	<code>CTRL-U, D</code> up, down one screen	
	<code>!&, G</code> go to end of line, end of file	

<http://nostarch.com/mug.htm>

/bin/mail and vi

```
/home/cis90/simben $ mail milhom90
```

```
Subject: Good Bones
```

```
Hey Homer,
```

```
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/simben $ mail milhom90
```

```
Subject: Good Bones
```

```
Hey Homer,
```

```
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

```
/home/cis90/simben $ mail milhom90
Subject: Good Bones
Hey Homer,
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)
.
EOT
/home/cis90/simben $
```

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

/bin/mail and vi

```
/home/cis90/milhom $ mail
Heirloom Mail version 12.4 7/29/08.  Type ? for help.
"/var/spool/mail/milhom90": 157 messages 5 new 155 unread
>N157 Benji Simms          Mon Nov 10 14:05  25/952  "Good Bones"
& 157
Message 157:
From simben90@oslab.cis.cabrillo.edu  Mon Nov 10 14:05:20 2014
Return-Path: <simben90@oslab.cis.cabrillo.edu>
From: Benji Simms <simben90@oslab.cis.cabrillo.edu>
Date: Mon, 10 Nov 2014 14:05:20 -0800
To: milhom90@oslab.cis.cabrillo.edu
Subject: Good Bones
User-Agent: Heirloom mailx 12.4 7/29/08
Content-Type: text/plain; charset=us-ascii
Status: R
```

```
Hey Homer,
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,
Benji
```

The message Homer reads has all the typos fixed.

&

Fix an email message before sending

```
/home/cis90/simben/edits $ mail rsimms
Subject: test of vi
sdkfjas;dfkjas;lkdfj
~v
(continue)
.
EOT
/home/cis90/simben/edits $
```

In vi:

- Use i to enter insert mode
- make changes
- save with <Esc>:wq



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
```

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

aspell list mimicks the standard unix spell program, roughly.

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

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

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

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

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

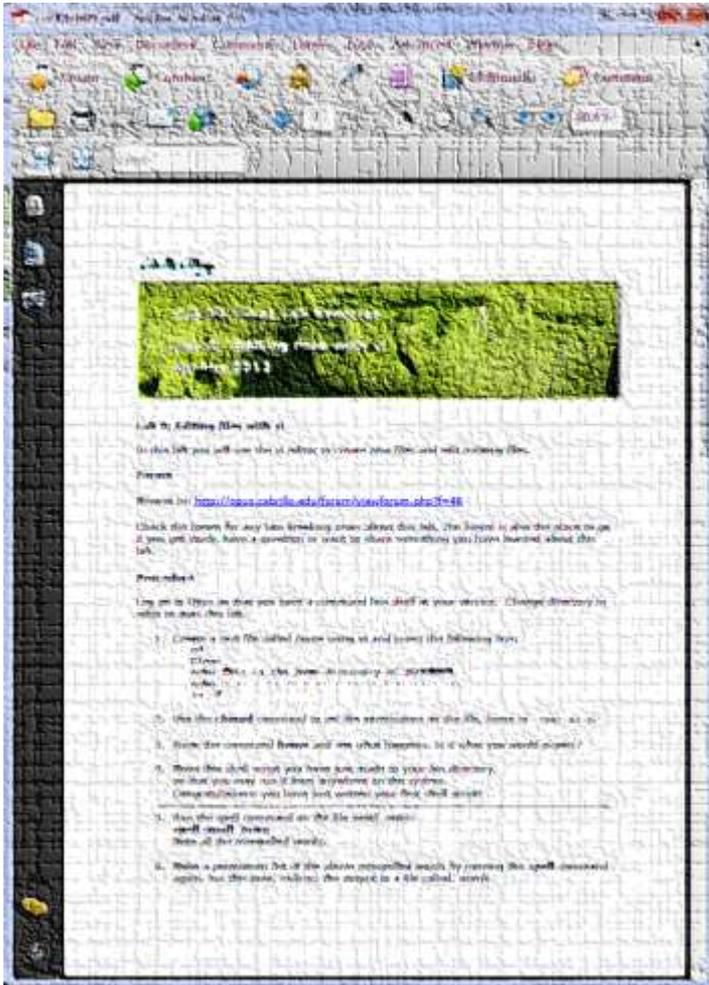
```
Eye halve a spelling chequer  
It came with my pea sea  
It plainly marques four my revue  
Miss steaks eye kin knot sea.  
Eye strike a key and type a word  
And weight four it two say  
Weather eye am wrong oar write  
It shows me strait a weigh.  
As soon as a mist ache is maid  
It nose bee fore two long  
And eye can put the error rite  
Its rare lea ever wrong.  
Eye have run this poem threw it  
I am shore your pleased two no  
Its letter perfect awl the weigh  
My chequer tolled me sew.
```

```
/home/cis90/simben $ spell edits/spellk  
chequer
```

How would you add "chequer"
(the British spelling) to your
personal dictionary?

*Copy the commands used into
the chat window when finished*

Assignment



Lab 9 will help
you start building
your vi skills!

Instructor: remember to mail students the tech file!

~/cis90/lab09/mail-tech-all

or

at 4pm

at> /home/rsimms/cis90/lab09/mail-tech-all

at> <Ctrl-d>



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 demonstrate 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 Ctrl-Z (to stop foreground 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
~
```

This is why Ctrl-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