



Rich's lesson module checklist

Last modified 04/17/2017

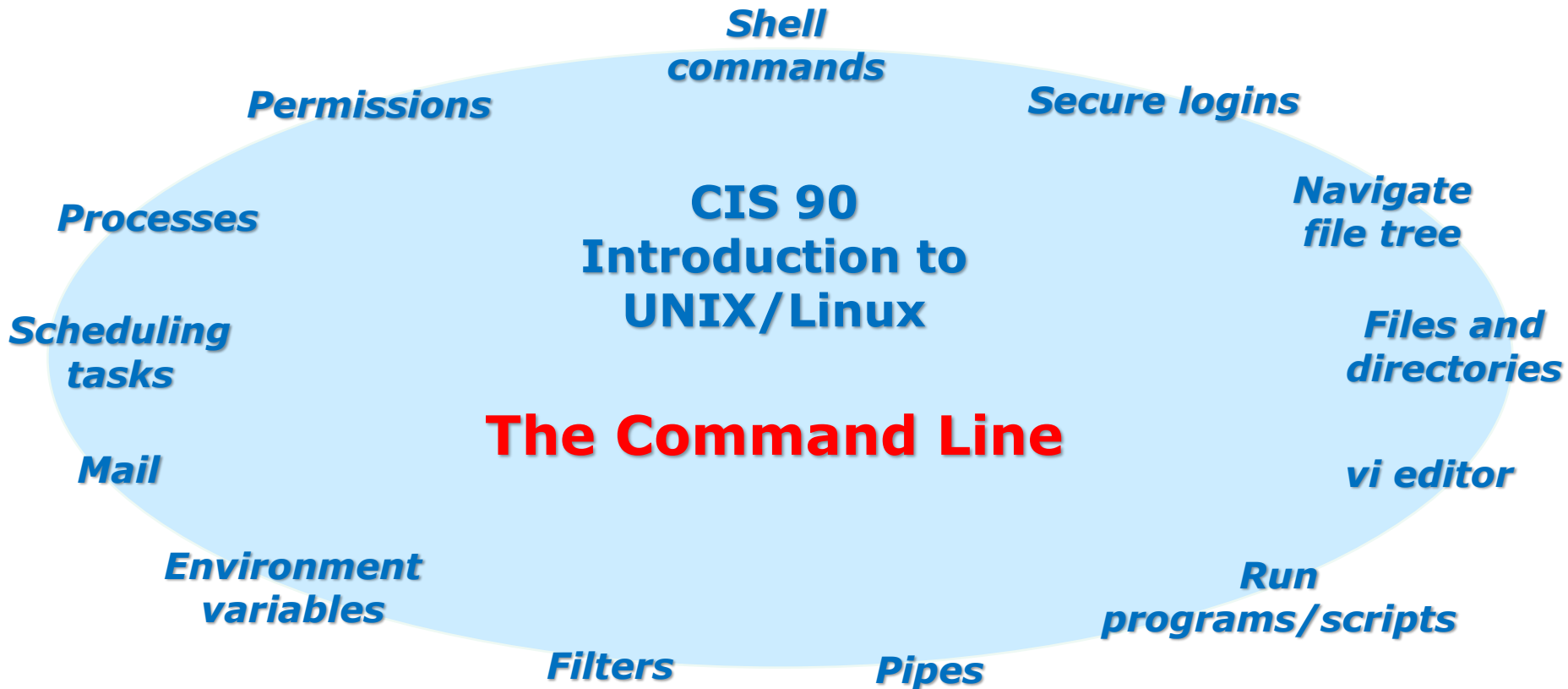
- ☐ Slides, Lab 10 and Project posted
- ☐ WB converted from PowerPoint
- ☐ Print out agenda slide and annotate page numbers

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

- ☐ Lock turnin directory at midnight (scripts/schedule-submit-locks)
- ☐ allscripts updated
- ☐ myscript in depot
- ☐ flowers and riddle in bin
- ☐ sample myscripts for Benji and Homer

- ☐ Backup slides, CCC info, handouts on flash drive
- ☐ Spare 9v battery for mic
- ☐ Key card for classroom door

- ☐ Check CCC Confer and 3C Media videos



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 attending class

simms-teach.com/cis90calendar.php

Rich's Cabrillo College CIS Classes
CIS 90 Calendar

CIS 90 (Fall 2014) Calendar

Course Details: [Calendar](#)

[CIS 90](#)

Lesson	Date	Topics	Link
1	9/2	Class and Lesson Overview - Understand how the course will work - High-level overview of computers, operating systems, and virtual machines - Overview of UNIX/Linux market and architecture - Using SSH for remote network logs - Using terminals and the command line Materials Presentation slides (download) Supplemental - How to #148: Logging into Opus (download) Assignments Student Survey Lab 1 CIS 90 Confer Enter virtual classroom	2.4 9/2-3 9/2-4 (9/2)
2		Quiz 1 Commands	

1. Browse to:
<http://simms-teach.com>
2. Click the **CIS 90** link.
3. Click the **Calendar** link.
4. Locate today's lesson.
5. Find the **Presentation slides** for the lesson and **download** for easier viewing.
6. Click the **Enter virtual classroom** link to join CCC Confer.
7. Log into Opus with Putty or ssh command.

Note: Blackboard Collaborate Launcher only needs to be installed once. It has already been downloaded and installed on the classroom PC's.

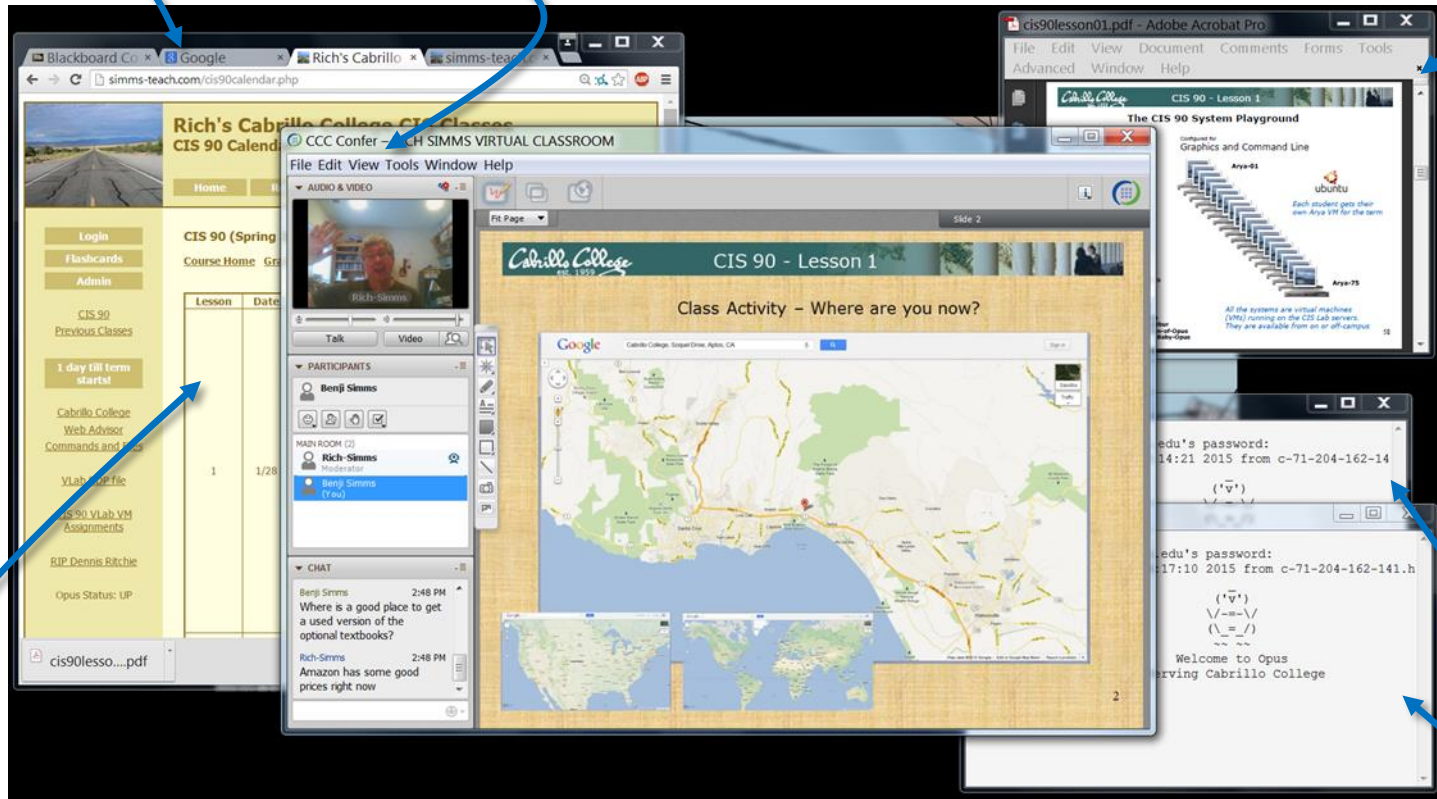


Student checklist for suggested screen layout

☐ Google

☐ CCC Confer

☐ Downloaded PDF of Lesson Slides



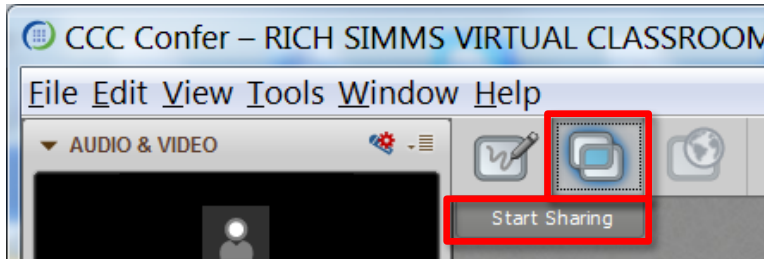
☐ CIS 90 website Calendar page

☐ One or more login sessions to Opus

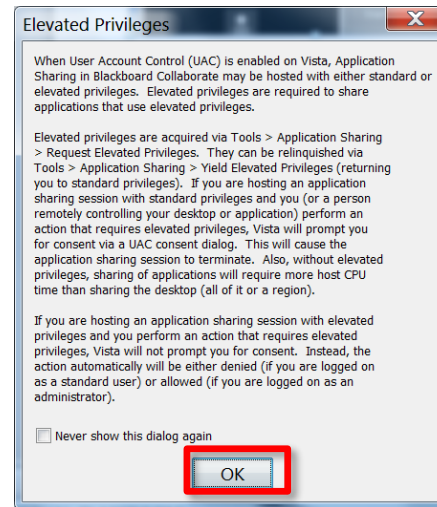


Student checklist for sharing desktop with classmates

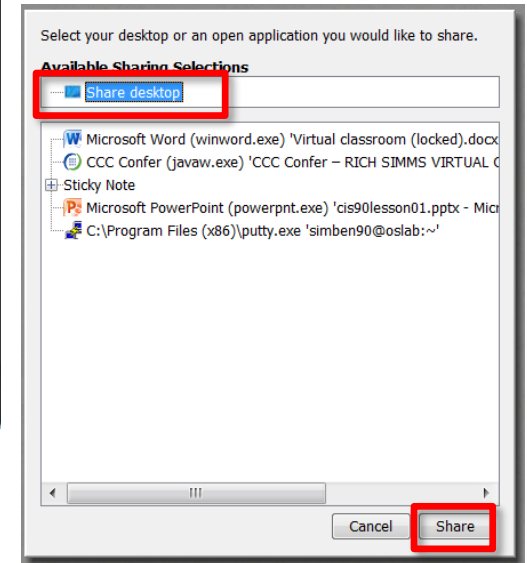
1) Instructor gives you sharing privileges



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



3) Click OK button.



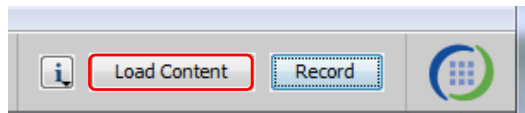
4) Select "Share desktop" and click Share button.



Rich's CCC Confer checklist - setup

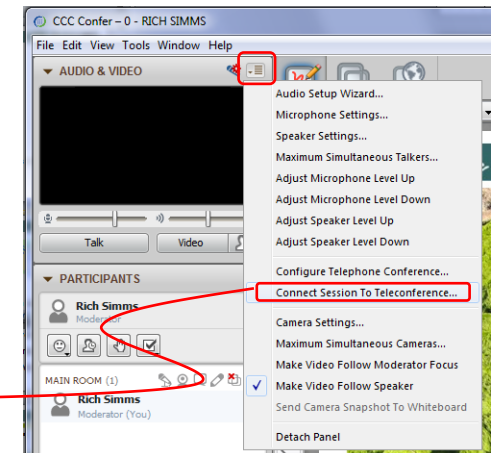
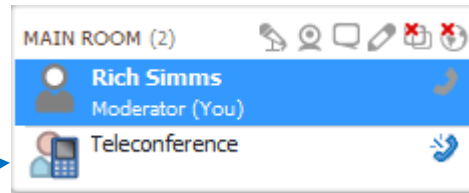


[] Preload White Board

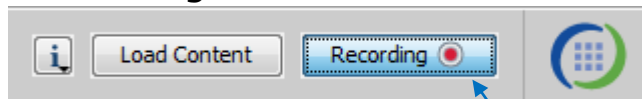


[] Connect session to Teleconference

Session now connected to teleconference



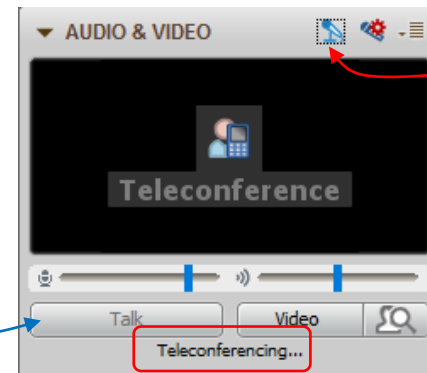
[] Is recording on?



Red dot means recording

[] Use teleconferencing, not mic

Should be grayed out



Should change from phone handset icon to little Microphone icon and the Teleconferencing ... message displayed



Rich's CCC Confer checklist - screen layout



The screenshot displays a Windows desktop environment during a CCC Confer session. The desktop includes the following elements:

- CCC Confer - 0 - RIC...:** A window showing the conferencing interface with a video feed of Rich Simms, a list of participants (Rich Simms, Moderator), and a chat window.
- simms-teach.com/docs/cis90/cis-90-TEST-1-Fall-12.pdf:** A web browser window displaying a quiz titled "Part 1 - Flashc (1 point each)". The quiz questions are:
 - [Q1] What command shows the other users logged in to the computer?
 - [A1]
 - [Q2] What environment variable is used by the shell to determine which directories to search when locating a command?
 - [A2]
- Terminal Window:** A window showing a command prompt session for user simben90@oslab. The session includes a login attempt, password entry, and a list of files in the current directory (boot, bin, etc, sbin, mail, ls).
- vSphere Client:** A window showing the vCenter interface, displaying a list of virtual machines (CIS 192) and their status (Powered on).

Red callout boxes with arrows point to specific elements:

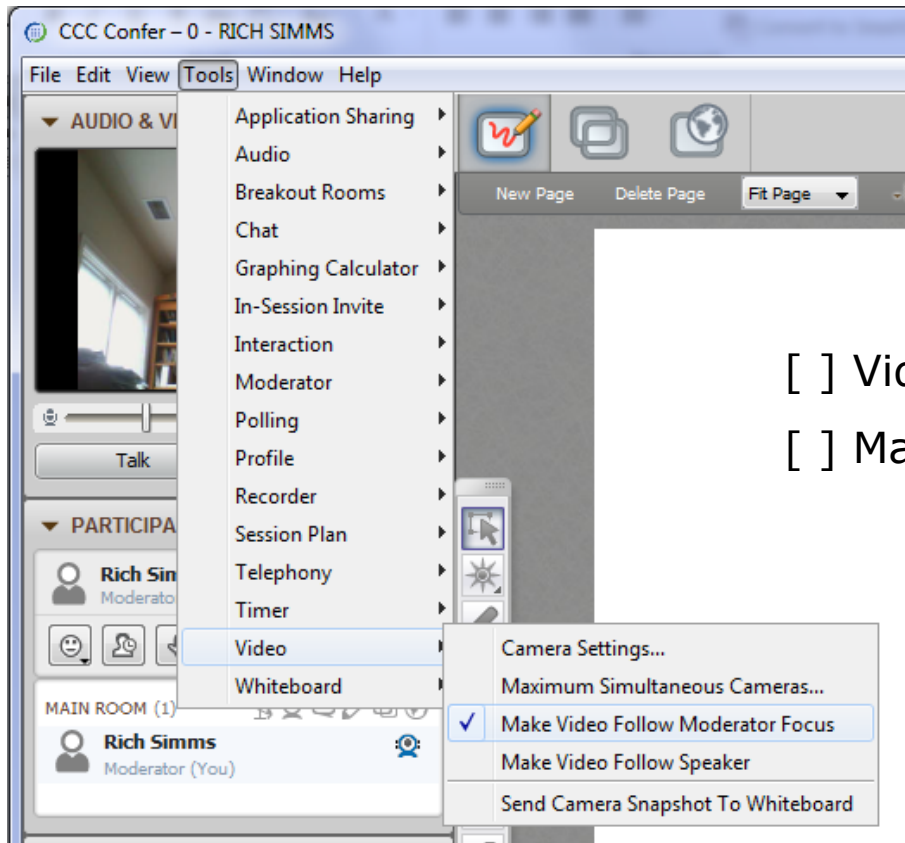
- foxit for slides:** Points to the Foxit Reader window displaying the PDF document.
- chrome:** Points to the web browser window displaying the quiz.
- putty:** Points to the terminal window.
- vSphere Client:** Points to the vCenter interface window.

[] layout and share apps





Rich's CCC Confer checklist - webcam setup

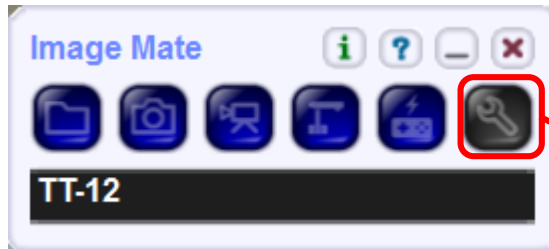


[] Video (webcam)

[] Make Video Follow Moderator Focus



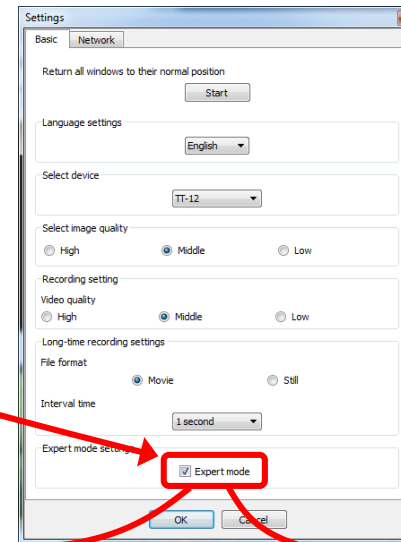
Rich's CCC Confer checklist - Elmo



Elmo rotated down to view side table



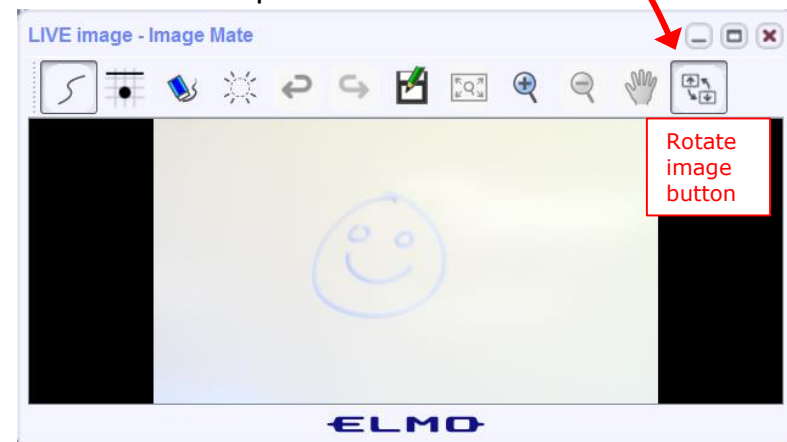
Run and share the Image Mate program just as you would any other app with CCC Confer



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 fixes

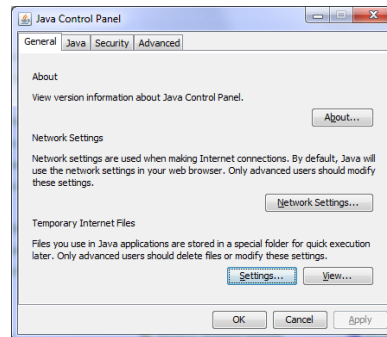
Universal Fix for CCC Confer:

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

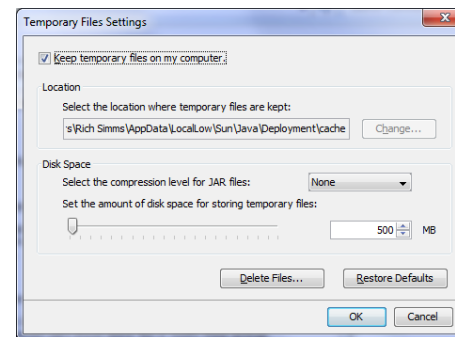
Control Panel (small icons)



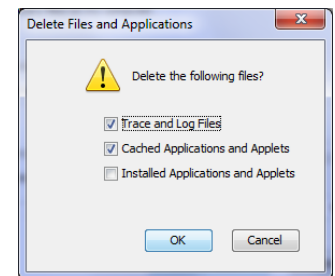
General Tab > Settings...



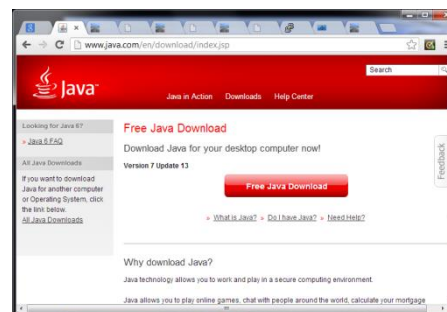
500MB cache size



Delete these

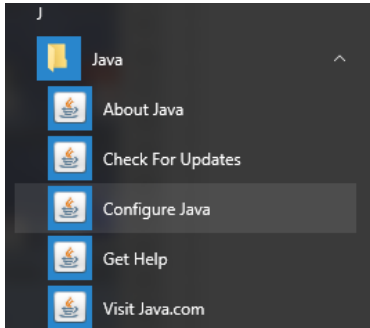


Google Java download

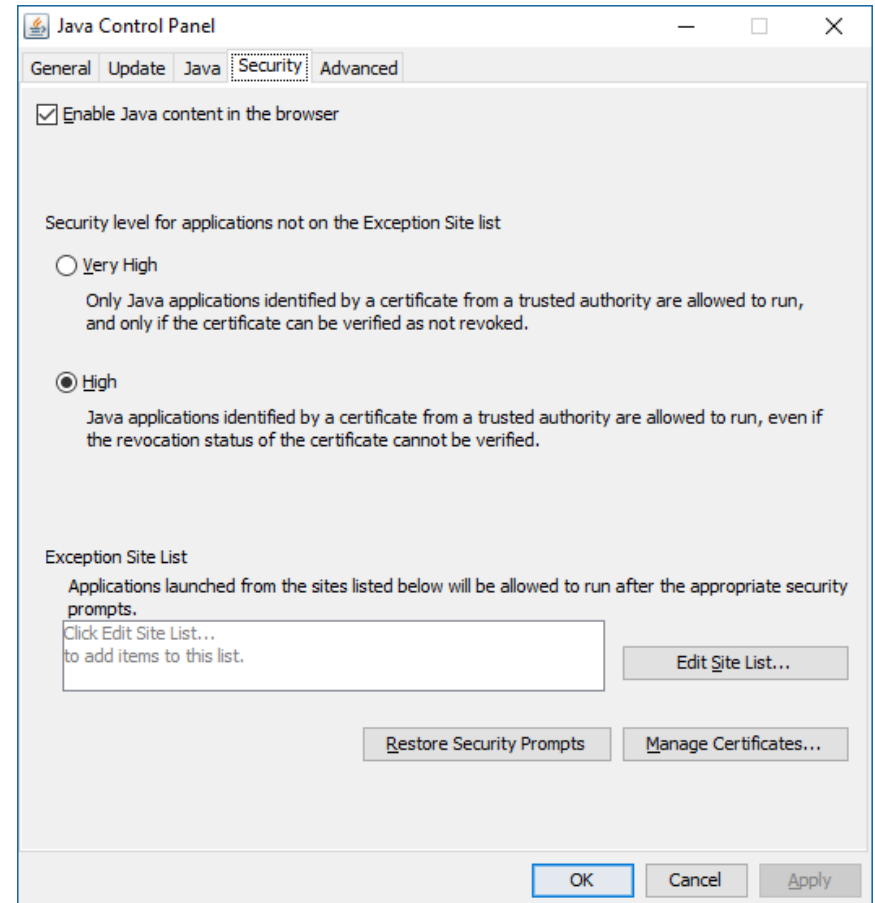




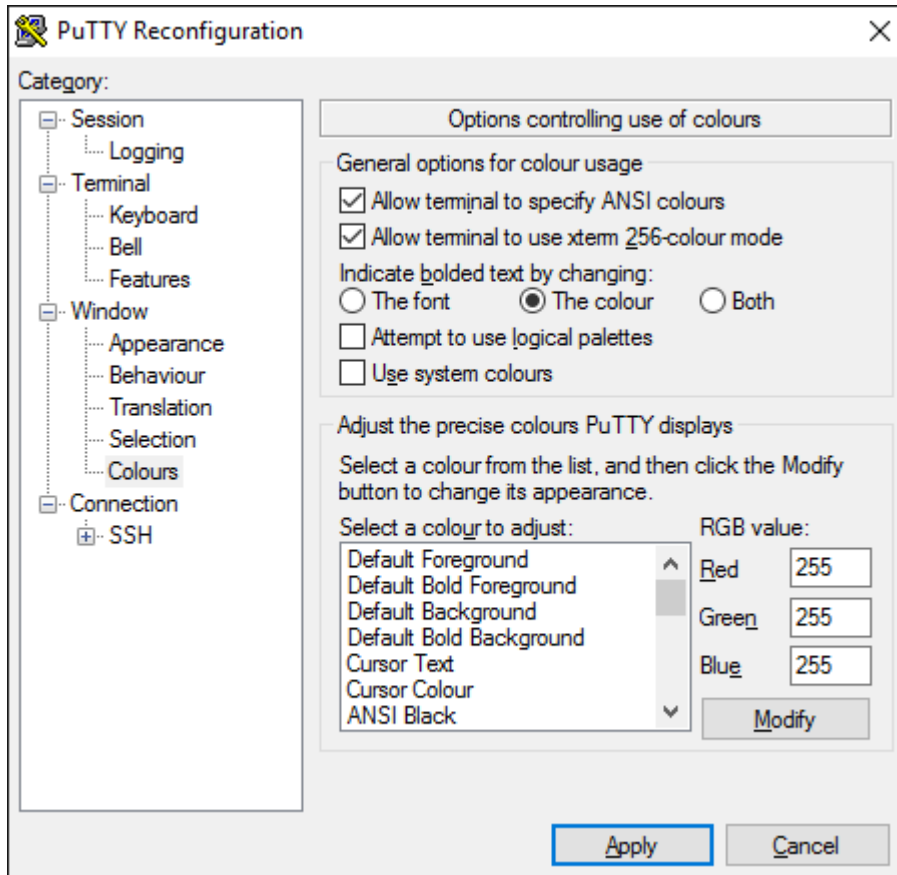
Rich's CCC Confer checklist - digital certificate work around



1. Open the [Java Control Panel](#)
2. Select the **Security** tab
3. Select **Edit Site List...**
4. Select **Add**
5. Click into the white box next to the red exclamation mark and type **<https://na-downloads.illuminate.com>**
6. Press OK
7. Press **Continue** on the pop-up message
8. Press OK
9. Access your session or recording once more



Rich's CCC Confer checklist - Putty Colors



Putty Colors

Default Foreground 255 255 255
 Default Bold Foreground 255 255 255
 Default Background 51 51 51
 Default Bold Background 255 2 85
 Cursor Text 0 0 0
 Cursor Color 0 255 0
 ANSI Black 77 77 77
 ANSI Black Bold 85 85 85
 ANSI Red 187 0 0
 ANSI Red Bold 255 85 85
 ANSI Green 152 251 152
 ANSI Green Bold 85 255 85
 ANSI Yellow 240 230 140
 ANSI Yellow Bold 255 255 85
 ANSI Blue 205 133 63
 ANSI Blue Bold 135 206 235
 ANSI Magenta 255 222 173
 ANSI Magenta Bold 255 85 255
 ANSI Cyan 255 160 160
 ANSI Cyan Bold 255 215 0
 ANSI White 245 222 179
 ANSI White Bold 255 255 255

<http://looselytyped.blogspot.com/2013/02/zenburn-pleasant-color-scheme-for-putty.html>



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.*

Volume

**4 - increase conference volume.*

**7 - decrease conference volume.*

**5 - increase your voice volume.*

**8 - decrease your voice volume.*



Instructor: **Rich Simms**

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

Passcode: **136690**



Melissa



Tess



Philip



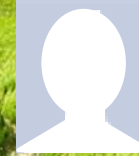
Daniel



Ian J.



Harold



Steven P.



Jasen



Sam



Ryan



Roberto



Hans



Cristian



Nicholas



Nigel



Luis



Stephen L.



Alison



Gracie



Cameron



Julian



Ken



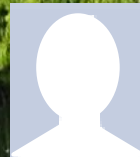
Joshua V.



Samantha



Ian C.



Justin



Dillon

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)

The Shell Environment

Objectives

- Be able to set, view and unset shell variables
- Describe the difference between the set and env commands
- Explain the importance of the export command.
- Describe three actions that are handled by the .bash_profile file
- Define user-defined aliases
- Explain the . (dot) command and the exec command.

Agenda

- Quiz
- Questions
- More on vi
- Submitting Lab 9 & pathnames
- Tangent on spell
- Personal dictionaries
- Lab 9 subtle things
- Housekeeping
- Final project preview
- Variables vs Files
- Shell variables
- Environment variables
- Shell environment
- Variables and child processes
- Aliases
- bash startup files
- .bash_profile
- .bashrc
- . and exec
- Grok this lesson
- 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.



More on vi

Activity

What is the difference between **:q!** and **:!q** commands in vi?

```
18. KEYBOARD:      Whar ya hang the dang keys.
19. SOFTWARE:      Them dang plastic forks and knifs.
20. MOUSE:        Whut eats the grain in the barn.
21. MAINFRAME:    Holds up the barn roof.
:!q
```

```
18. KEYBOARD:      Whar ya hang the dang keys.
19. SOFTWARE:      Them dang plastic forks and knifs.
20. MOUSE:        Whut eats the grain in the barn.
21. MAINFRAME:    Holds up the barn roof.
:q!
```

Write your answer in the chat window

 :!q vs  :q!

```
18. KEYBOARD:      Whar ya hang the dang keys.
19. SOFTWARE:      Them dang plastic forks and knifs.
20. MOUSE:         Whut eats the grain in the barn.
21. MAINFRAME:     Holds up the barn roof.
:!q
```

This will attempt to run a command "q" in the bash shell

```
18. KEYBOARD:      Whar ya hang the dang keys.
19. SOFTWARE:      Them dang plastic forks and knifs.
20. MOUSE:         Whut eats the grain in the barn.
21. MAINFRAME:     Holds up the barn roof.
:q!
```

This will quit vi without saving any changes made

```
simben90@oslab:~/edits
Technology for Mountain Folk
BYTE:      Whut them dang flys do.
CHIP:      Munchies fer the TV.
DOT MATRIX: Old Dan Matrix's wife.
DOWNLOAD:  Gettin the farwood off the truk.
ENTER:     Northerner talk few "C'mon in y'all"
FLOPPY DISC: Whatcha git from tryin to carry too much farwood.
HARD DRIVE: Gettin home in the winter time.
KEYBOARD:  Whar ya hang the dang keys.
LAP TOP:   Whar the kitty sleeps.
LOG OFF:   Don't add no more wood.
LOG ON:    Makin a wood stove hotter.
MAINFRAME: Holds up the barn roof.
MEGA HERTZ: When yer not kerful gettin the farwood.
MICRO CHIP: Whut's in the bottom of the munchie bag.
MODEM:     Whut cha did to the hay fields.
MONITOR:   Keepin an eye on the wood stove.
MOUSE PAD: That hippie talk fer the rat hole.
MOUSE:     Whut eats the grain in the barn.
PORT:      Fancy Flatlander wine.
PROMPT:    Whut the mail ain't in the winter time.
RAM:       That thar thing whut splits the farwood
@
```

Editing vocab in one login session

```
simben90@oslab:~/edits
E325: ATTENTION
Found a swap file by the name ".vocab.swp"
  owned by: simben90   dated: Tue Nov 19 06:34:51 2013
  file name: ~simben90/edits/vocab
  modified: no
  user name: simben90  host name: oslab.cishawks.net
  process ID: 32394 (still running)
While opening file "vocab"
      dated: Sat Nov 16 19:11:16 2013

(1) Another program may be editing the same file.
    If this is the case, be careful not to end up with two
    different instances of the same file when making changes.
    Quit, or continue with caution.

(2) An edit session for this file crashed.
    If this is the case, use ":recover" or "vim -r vocab"
    to recover the changes (see ":help recovery").
    If you did this already, delete the swap file ".vocab.swp"
    to avoid this message.

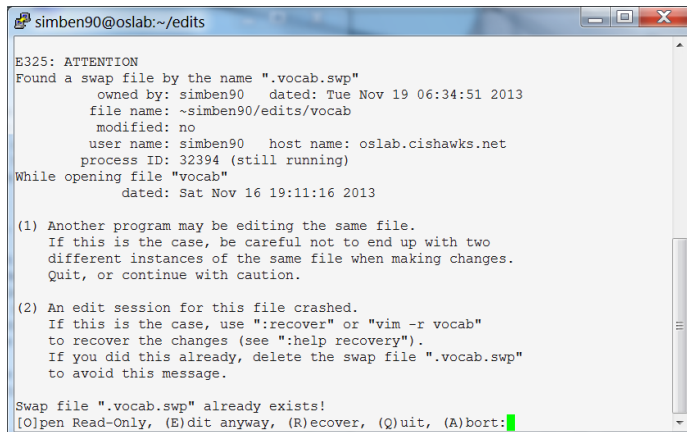
Swap file ".vocab.swp" already exists!
[O]pen Read-Only, (E)dit anyway, (R)ecover, (Q)uit, (A)bort:
```

Attempting to edit vocab in another session before the original edit session was ended

the .swp file for vocab

```
/home/cis90/simben $ cd edits
/home/cis90/simben/edits $ ls -a
.   better_town  small_town  temp      text.fxd  .vocab.swp  words
..  lab09        spellk      text.err  vocab      women
/home/cis90/simben/edits $
```

When you edit a file with vi it copies your original file to a temporary *.swp* file. Any changes made happen to the *.swp* file instead of the original file. The **:w** command updates the contents of the original file with the contents of the *.swp* file.



```
simben90@oslab:~/edits
E325: ATTENTION
Found a swap file by the name ".vocab.swp"
  owned by: simben90   dated: Tue Nov 19 06:34:51 2013
  file name: ~simben90/edits/vocab
  modified: no
  user name: simben90  host name: oslab.cishawks.net
  process ID: 32394 (still running)
While opening file "vocab"
  dated: Sat Nov 16 19:11:16 2013

(1) Another program may be editing the same file.
    If this is the case, be careful not to end up with two
    different instances of the same file when making changes.
    Quit, or continue with caution.

(2) An edit session for this file crashed.
    If this is the case, use ":recover" or "vim -r vocab"
    to recover the changes (see ":help recovery").
    If you did this already, delete the swap file ".vocab.swp"
    to avoid this message.

Swap file ".vocab.swp" already exists!
[O]pen Read-Only, [E]dit anyway, [R]ecover, [Q]uit, [A]bort:
```

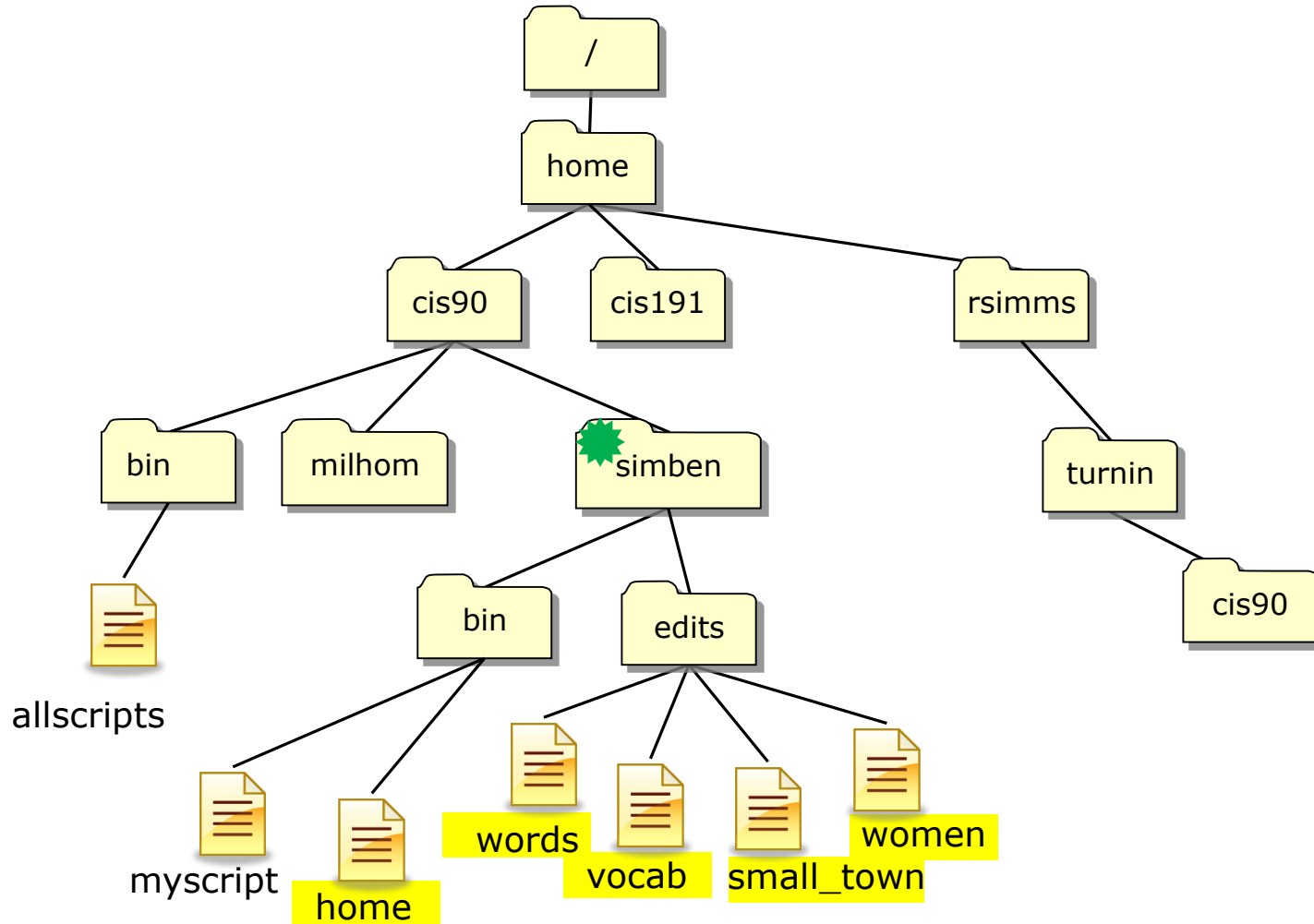
If you get this ATTENTION message it means the temporary *.swp* file still exists. You may be editing the same file in another session or your original editing session was disconnected before finishing. To get rid of this message you need to remove the *.swp* file.



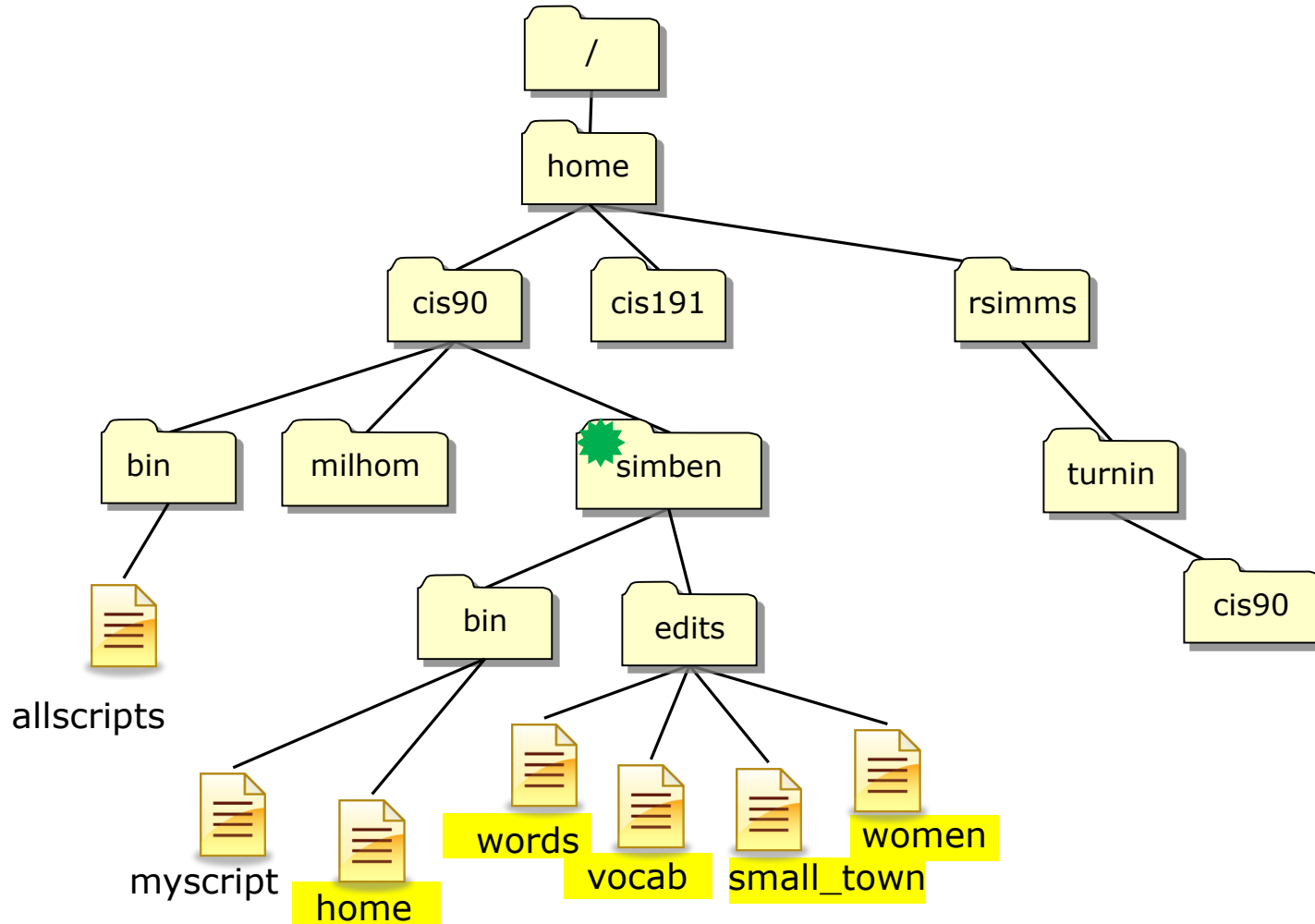
Submitting Lab 9 & Pathnames!

REMINDER

- You must **ALWAYS** use **VALID PATHNAMES** when specifying files as **ARGUMENTS** on a command.
- Pathnames can be relative or absolute.
- A common mistake in the past on Lab 9 is to ignore error messages and not submit all the file content requested.

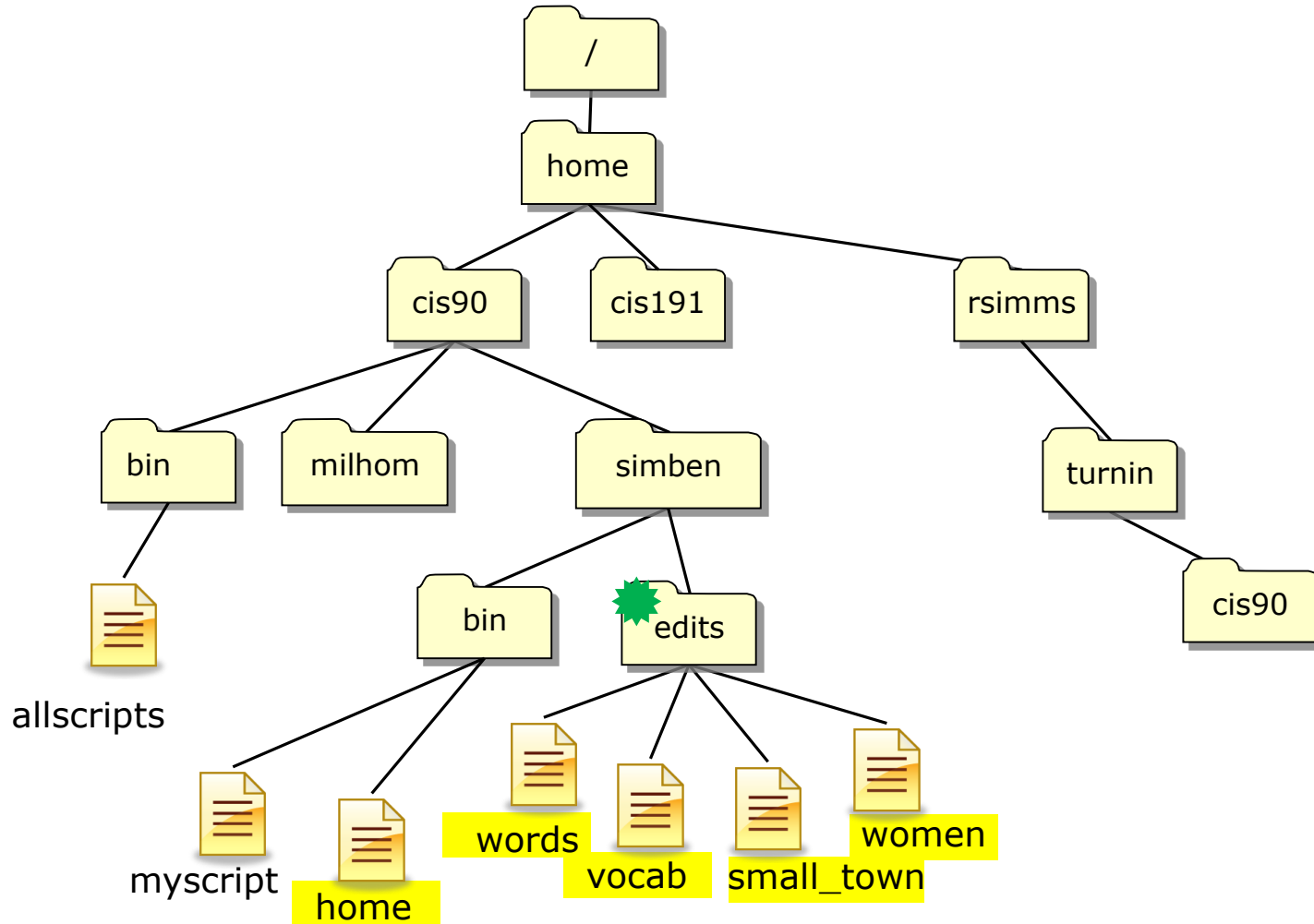


From  how could Benji concatenate the highlighted files into a file named lab09 in his home directory?

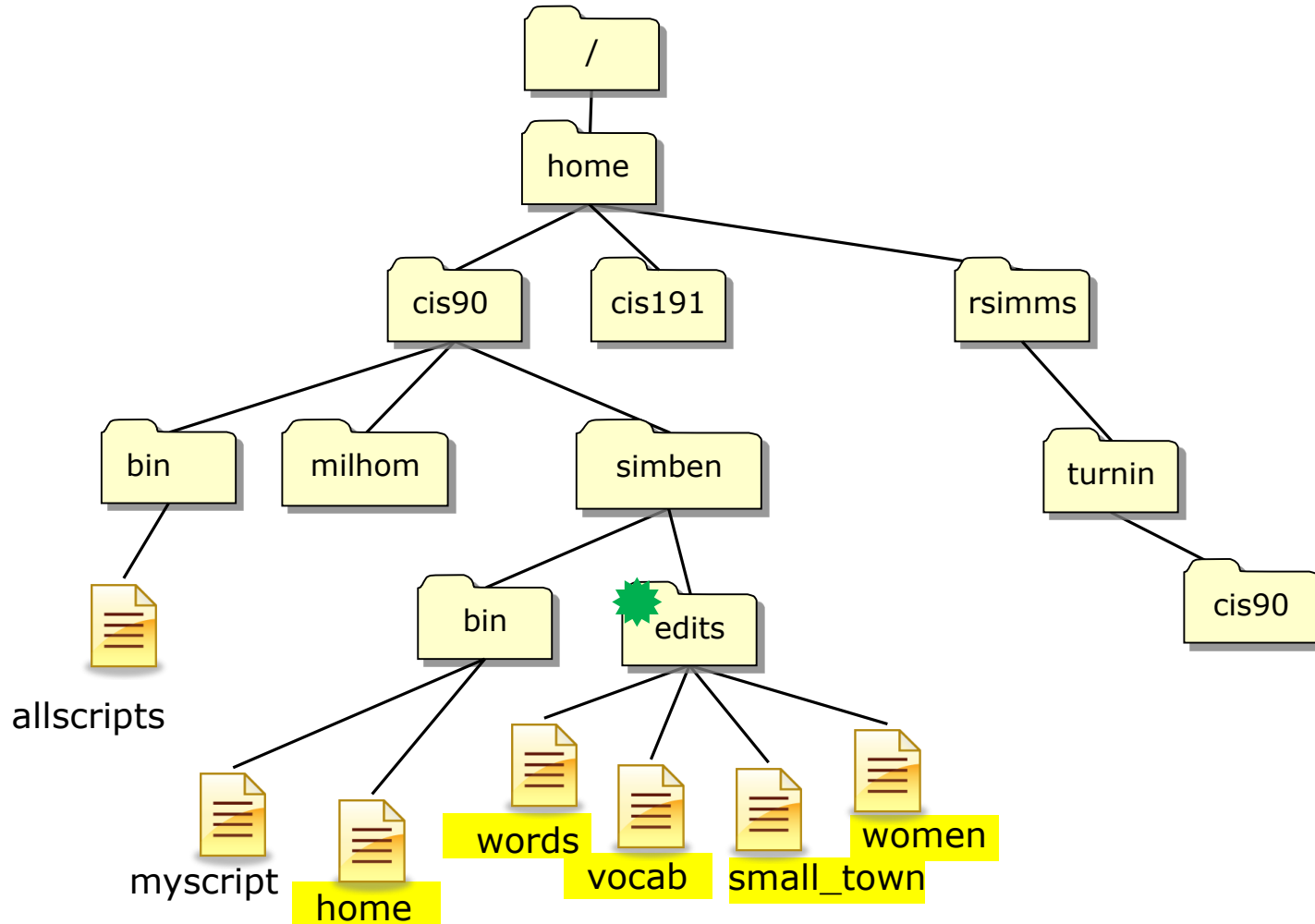


From  how could Benji concatenate the highlighted files into a file named lab09 in his home directory?

```
cat bin/home edits/words edits/vocab edits/small_town edits/women > lab09
```

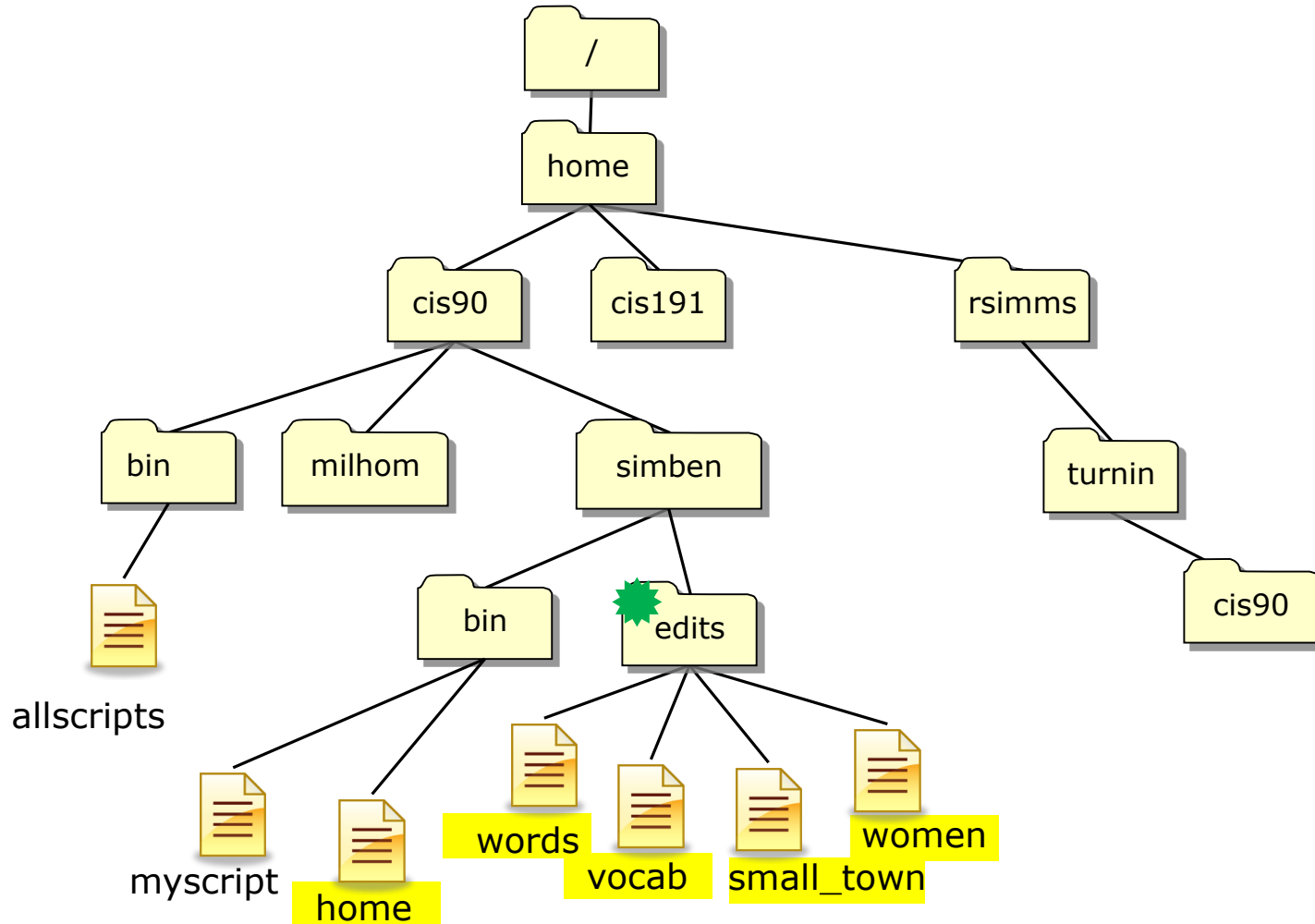


From  how could Benji concatenate the highlighted files into a file named lab09 in his home directory?



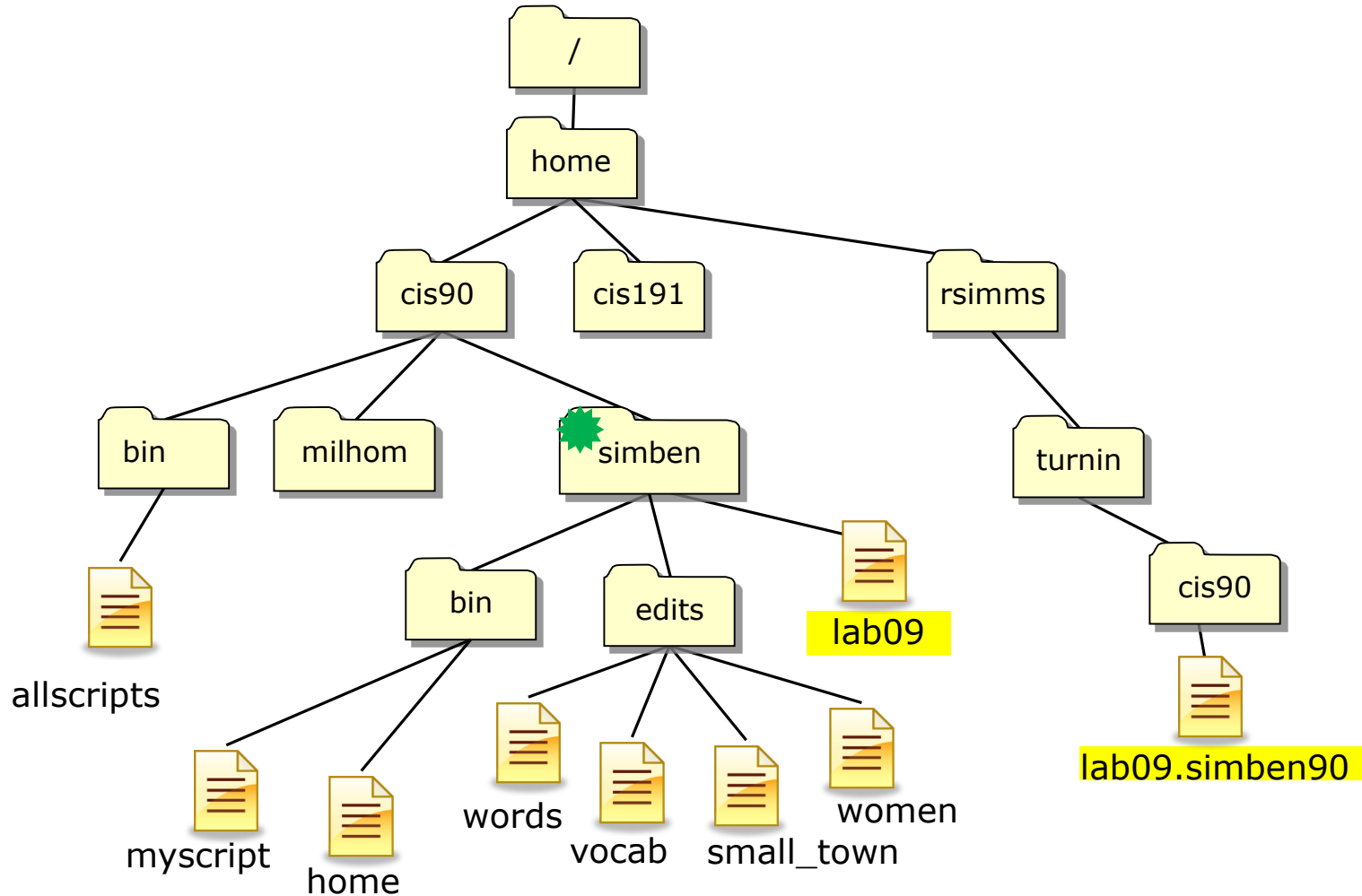
From  how could Benji concatenate the highlighted files into a file named lab09 in his home directory?

```
cat words vocab small_town women ../bin/home > ../lab09
```



From  how could Benji concatenate the highlighted files into a file named lab09 in his home directory?

```
cat words vocab small_town women ~/.bin/home > ~/.lab09
```



From  how could Benji submit his work to Rich's turnin/cis90 directory

```
cp lab09 /home/rsimms/turnin/cis90/lab09.$LOGNAME
```



A Tangent on Spell (from last lesson)

Soquel is not in the UNIX dictionary

```
/home/cis90/simben $ echo Benji lives in Soquel > address  
/home/cis90/simben $ cat address  
Benji lives in Soquel
```

```
/home/cis90/simben $ spell address  
Soquel
```

Question: How can we add Soquel to the UNIX dictionary so it is ignored in future spell checks?

Question: How can we add Soquel to the UNIX dictionary so it is ignored in future spell checks?

```
/home/cis90/simben $ man spell
```

Hmmm. No man page for spell - weird!

No manual entry for spell

```
/home/cis90/simben $ type spell
```

Where is it on our path?

spell is hashed (/usr/bin/spell)

```
/home/cis90/simben $ file /usr/bin/spell
```

So what kind of file is it?

/usr/bin/spell: Bourne shell script text executable

```
/home/cis90/simben $ cat /usr/bin/spell
```

Ah ha, it's a script, so lets look at it ...

#!/bin/sh

aspell list mimicks the standard unix spell program, roughly.

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

*Well ... son of a gun, the
actual command is **aspell**!*

Question: How can we add Soquel to the UNIX dictionary so it is ignored in future spell checks?

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.

<snipped>

--home-dir=<directory>

Directory Location for personal wordlist files.

--per-conf=<file name>

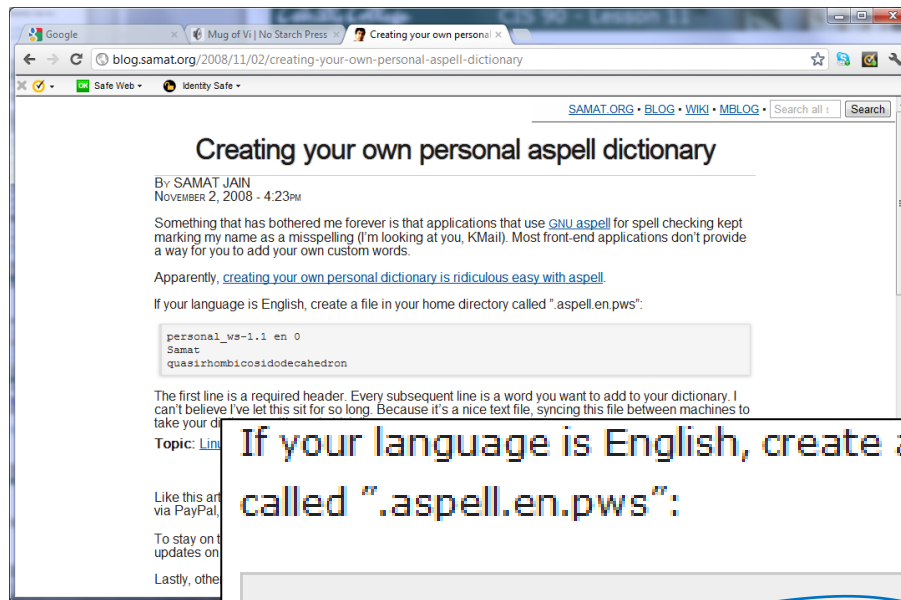
Personal configuration file. This file overrides options found in the global config file.

There must be a way to add Soquel ... the man page indicates it is possible but has no examples ... lets try google instead

Googling "linux aspell personal dictionary"

Bingo! Thank you Samat Jain!

<http://blog.samat.org/2008/11/02/creating-your-own-personal-aspell-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
```

Add this line to the top

Now add any words you wish for the aspell program to ignore when doing spelling checks

Adding words to the UNIX dictionary

```
/home/cis90/simben $ echo "personal_ws-1.1 en 0" > .aspell.en.pws  
/home/cis90/simben $ echo Soquel >> .aspell.en.pws  
  
/home/cis90/simben $ spell address  
/home/cis90/simben $
```

This is how you would add Soquel to your own custom dictionary to be used with the spell command

This is FYI and not required for Lab 9

```
/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*



Ayshire moshpit and personal dictionaries

New Tab x richsimms x Rich's Cabr x Cabrillo Co x esc key - G x

← → ↺ oslab.cishawks.net/forum/viewtopic.php?f=88&t=2524&sid=6491cb07ac419956110ba7cb

phpBB Cabrillo College: Computer and Information Systems
Forum for students in the Computer Networking and System Administration and/or Computer Support Specialist programs

Board index • Cabrillo College Fall 2013 Courses • CIS 90 - Fall 2013

Search: Search
Advanced search

Mag Register Login

Mashpit and Aysire

Forum rules
Be nice to each other and please don't post any username or password information!

POST REPLY | Search this topic | Search

Mashpit and Aysire
(by Marc Caudill • Sat Nov 16, 2012 11:04 am)
Are we supposed to leave mashpit and Aysire misspelled?
Why? Does it add to the country charm of the prose?
This is a mashpit:



Were people drooling their biscuits on Thanksgiving.

This is a mashpit:



A place where red blooded American boys go for fun on Saturday nights.

Here's some playlist fun where you can re-live your youth and make your own mashpit:
<http://mattblairbaum.github.io/mashpits.js/>

Aysire is a small town in Iowa.
Aysire, Iowa

Marc Caudill
Posts: 49
Joined: Tue Sep 03, 2013 1:19 pm

0 posts • Page 1 of 1

moshpit?



1. moshpit

a place at a gig where you can dance with however the ^{bleeped} you want with a bunch of people you don't know. the dancing will often include punches aimed in the air NOT at the person nearest to you however usually results in full contact. can be dangerous however everyone with a ticket should feel welcome in the mosh pit.



mosh pit *noun*

Definition of MOSH PIT

: an area in front of a stage where very physical and rough dancing takes place at a rock concert

 See [mosh pit](#) defined for English-language learners »

First Known Use of MOSH PIT

1988

Ayrshire?

Ayrshire



The Ayrshire breed originated in the County of Ayr in Scotland, prior to 1800. The county is divided into the three districts of Cunningham, in the more northern part, Kyle, which lies in the center, and Carrick, which forms the southern part of the county. During its development, it was referred to first as the Dunlop, then the Cunningham, and finally, the Ayrshire. How the different strains of cattle were crossed to form the breed known as Ayrshire is not exactly known. There is good evidence that several breeds were crossed with native cattle to create the foundation animals of the breed. In *Agriculture, Ancient and Modern*, published in 1866, Samuel Copland describes the native cattle of the region as "diminutive in size, ill-fed, and bad milkers." Prior to 1800 many of the cattle of Ayrshire were black, although by 1775 browns and mottled colors started to appear.

Ayrshires are red and white, and purebred Ayrshires only produce red and white offspring. Actually, the red color is a reddish-brown mahogany that varies in shade from very light to very dark. On some bulls, the mahogany color is so dark that it appears almost black in contrast to the white. There is no discrimination or registry restriction on color patterns for Ayrshires. The color markings vary from nearly all red to nearly all white. The spots are usually very jagged at the edges and often small and scattered over the entire body of the cow. Usually, the spots are distinct, with a break between the red and the white hair. Some Ayrshires exhibit a speckled pattern of red pigmentation on the skin covered by white hair. Brindle and roan color patterns were once more common in Ayrshires, but these patterns are rare today. [\[Oklahoma State University\]](#)

Copyright ©2007, Moocow.com

Add more to your custom word list

```
cd  
echo "moshpit" >> .aspell.en.pws  
echo "Ayshire" >> .aspell.en.pws  
  
spell edits/small_town
```

Note: Please leave Ayshire and moshpit (or mashpit) in your *words* file when you submit Lab 9



Lab 9

Subtle Things

(but very important)

In Lab 9 you create a script named home in your edits/ directory

```
/home/cis90/simben/edits $ cat home
cd
clear
echo This is the home directory of $LOGNAME
echo =====
ls -F
```

From your home directory

```
/home/cis90/simben $ home  
-bash: home: command not found
```

Move home from edits/ to bin/

```
/home/cis90/simben $ mv edits/home bin/
```

Again, from your home directory

```
/home/cis90/simben $ home  
This is the home directory of simben90
```

=====

bag/	etc/	lab07	monster2	snap2
bigfile	expressions	lab07.bak	monster3	tempdir/

< snipped >

*From your home directory,
the script does not work
until it is moved from
edits/ into bin/*

WHY?

QUESTION: From your home directory, why does the home script work only after moving it from the edits/ directory to the bin/ directory?

Answer: The edits/ directory is not on the path but the local bin/ directory is

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

Remember the six steps of the shell

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

```
-bash: home: command not found
```

If the shell is unable to locate the command on the path it prints "command not found"

Because

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

By moving the script into the user's local bin directory, which is on the path, the command can now be run from anywhere on the system

Housekeeping



Housekeeping

1. Lab 9 due 11:59_{PM} tonight.
2. Use **check9** to check your work.
3. Five more posts due 11:59_{PM} tonight.

Reminder:

Only posts in the CIS 90 forum during the most recent posting period are counted. Excess posts in past quarters are not carried forward.

Housekeeping

**Last Withdraw:
Saturday 04/22/17**

Students who are no longer participating in the class (turning in assignments, posting on the forum, taking quizzes or tests) may be dropped by the instructor

Heads up on Final Exam

Test #3 (final exam) is **MONDAY May 15th 7-9:50AM**

Mon	5/15	Test #3 (the final exam) Time - MONDAY 7:00AM - 9:50AM in Room 828 Materials - Test (canvas) CCC Confer Enter virtual classroom Class archives		5 posts Lab X1 Lab X2
------------	------	---	--	--

*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 **9:50AM**.
- Working and long distance students can take the test online via CCC Confer and Canvas.
- Working students will need to plan ahead to arrange time off from work for the test.
- Test #3 is **mandatory** (even if you have all the points you want)

FINAL EXAMINATIONS SCHEDULE: SPRING 2017, May 15 to May 20

Daytime Classes: All times in bold refer to the beginning times of classes. **MW/Daily** means Monday alone, Wednesday alone, Monday and Wednesday or any 3 or more days in any combination. **TTh** means Tuesday alone, Thursday alone, or Tuesday and Thursday. **Classes meeting other combinations of days and/or hours not listed must have a final schedule approved by their Division Dean.**

STARTING CLASS TIME/DAY(S)	EXAM HOUR	EXAM DATE
----------------------------	-----------	-----------

Classes starting between:

6:30 am and 8:55 am, MW/Daily	7:00 am-9:50 am	Wednesday, May 17
9:00 am and 10:15 am, MW/Daily	7:00 am-9:50 am	Monday, May 15
10:20 am and 11:35 am, MW/Daily	10:00 am-12:50 pm	Wednesday, May 17
11:40 am and 12:55 pm, MW/Daily	10:00 am-12:50 pm	Monday, May 15
1:00 pm and 2:15 pm, MW/Daily	1:00 pm-3:50 pm	Wednesday, May 17
2:20 pm and 3:35 pm, MW/Daily	1:00 pm-3:50 pm	Monday, May 15
3:40 pm and 5:30 pm, MW/Daily	4:00 pm-6:50 pm	Monday, May 15
6:30 am and 8:55 am, TTh	7:00 am-9:50 am	Monday, May 15
9:00 am and 10:15 am, TTh	7:00 am-9:50 am	Monday, May 15
10:20 am and 11:35 am, TTh	10:00 am-12:50 pm	Monday, May 15
11:40 am and 12:55 pm, TTh	10:00 am-12:50 pm	Monday, May 15
1:00 pm and 2:15 pm, TTh	1:00 pm-3:50 pm	Monday, May 15
2:20 pm and 3:35 pm, TTh	1:00 pm-3:50 pm	Monday, May 15
3:40 pm and 5:30 pm, TTh	4:00 pm-6:50 pm	Monday, May 15
Friday am	9:00 am-11:50 am	Monday, May 15
Friday pm	1:00 pm-3:50 pm	Monday, May 15
Saturday am	9:00 am-11:50 am	Monday, May 15
Saturday pm	1:00 pm-3:50 pm	Monday, May 15

CIS 90 Introduction to UNIX/Linux

Provides a technical overview of the UNIX/Linux operating system, including hands-on experience with commands, files, and tools. Recommended Preparation: CIS 1L or CIS 72.

Transfer Credit: Transfers to CSU; UC

Section	Days	Times	Units	Instructor	Room
95746	W	9:00AM-12:05PM	3.00	R.Simms	OL
&	Arr.	Arr.		R.Simms	OL

Section 95746 is an ONLINE course. Meets weekly throughout the semester online during the scheduled times by remote technology with an additional 50 min arranged online lab per week. For details, see instructor's web page at go.cabrillo.edu/online.

95747	W	9:00AM-12:05PM	3.00	R.Simms	828
&	Arr.	Arr.		R.Simms	OL

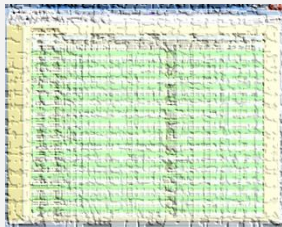
Section 95747 is a Hybrid ONLINE course. Meets weekly throughout the semester at the scheduled times with an additional 50 min online lab per week. For details, see instructor's web page at go.cabrillo.edu/online.

Where to find your grades

Send me your survey to get your LOR code name.

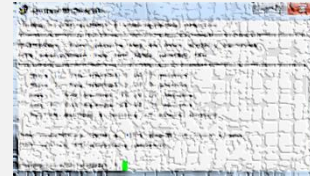
The CIS 90 website Grades page

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



Or check on Opus

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



Written by Jesse Warren a past CIS 90 Alumnus

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

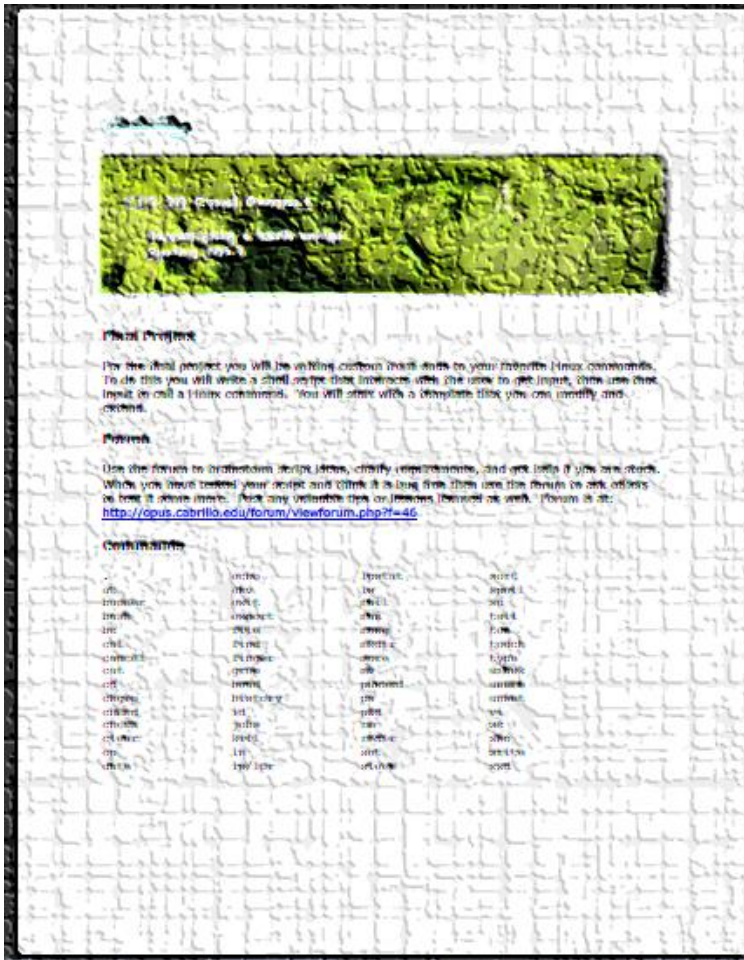
Points that could have been earned:

8 quizzes: 24 points
8 labs: 240 points
2 tests: 60 points
2 forum quarters: 40 points
Total: 364 points



final project *preview*

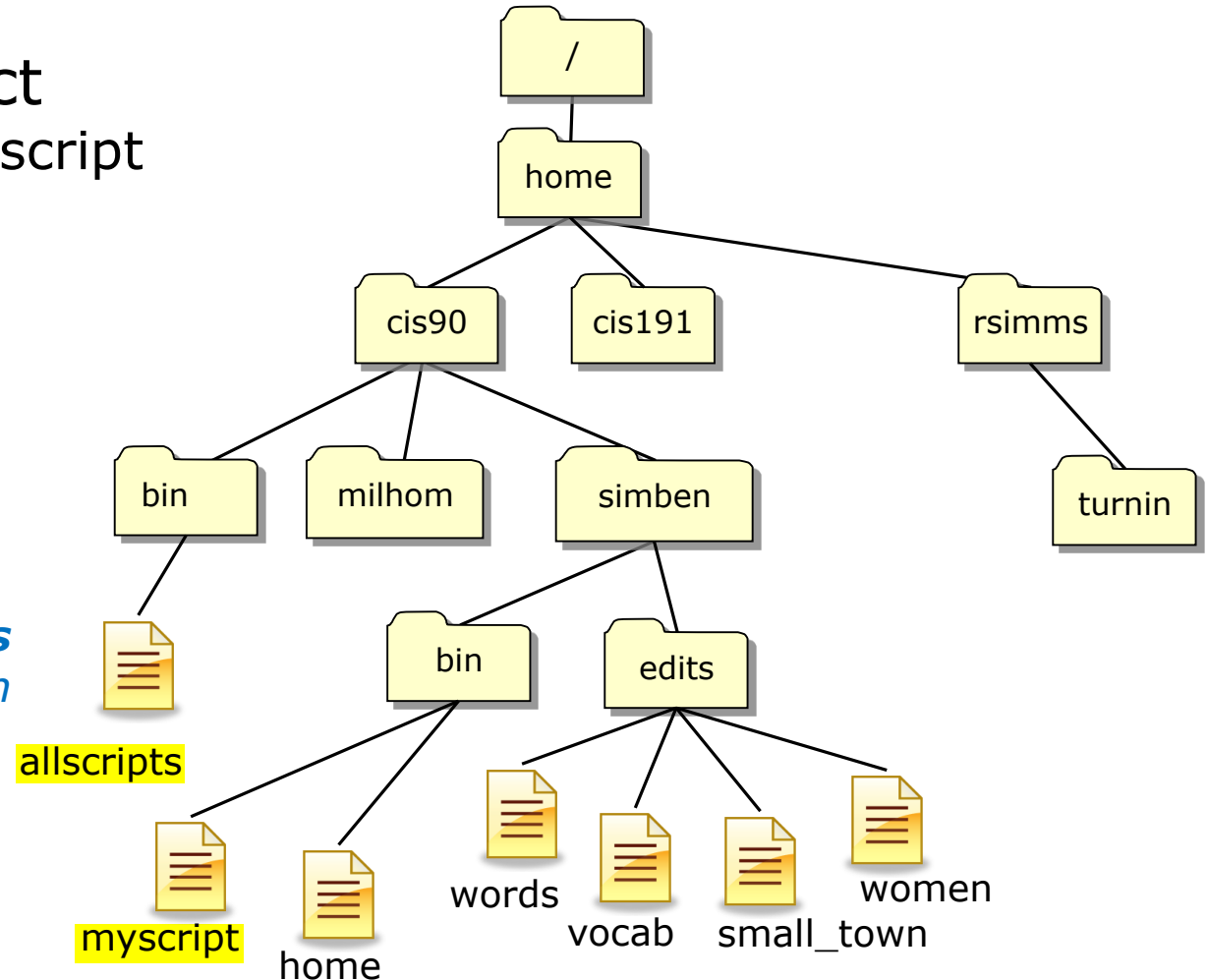
Final Project



*You now have
the necessary
skills to begin
the final project!*

Final Project

allscripts and myscript



*I make the **allscripts** file in /home/cis90/bin*

*You make the **myscript** file in your local bin directory*

```

/home/cis90/simben $ ls -l /home/cis90/bin/allscripts bin/myscript
-rwxr-xr-x 1 simben90 cis90 4296 Nov 13 13:07 bin/myscript
-rwxr-xr-x 1 rsimms    staff 4381 Nov 13 18:17 /home/cis90/bin/allscripts
  
```

The **allscripts** bash script

vi /home/cis90/bin/allscripts

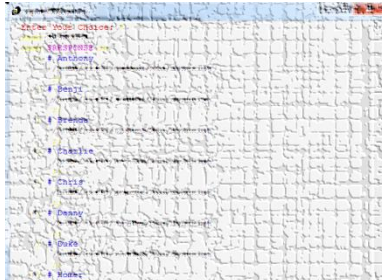
```
simben90@oslab:~
#!/bin/bash
#
# menu: A simple menu template
#
while true
do
    clear
    echo -n "
*****
*      Spring 2017 CIS 90 Online Projects      *
*****
1) Alison
2) Banji
3) Cameron
4) Cristian
5) Daniel H.
6) Dillon
7) Duke
8) Gracie
9) Hans
10) Harold
11) Homer
12) Ian C.
13) Ian J.
14) Jasen
15) Joshua
16) Julian
17) Justin
18) Ken
19) Luis
20) Melissa
21) Nicholas
22) Nigel
23) Phillip
24) Roberto
25) Ryan
26) Samantha
27) Sam
28) Stephen
29) Steven
30) Tess

99) Exit
```

*The first part of **allscripts** uses a long **echo** command to print a selection menu of the CIS 90 students.*

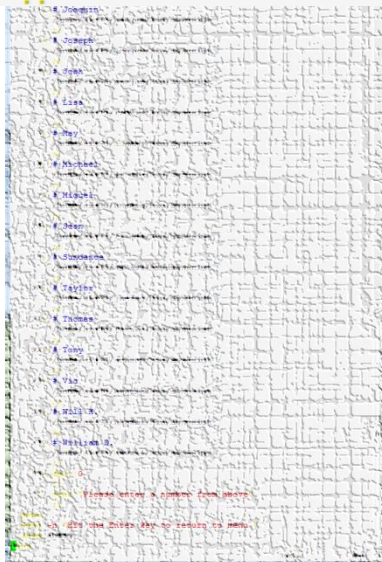
The **allscripts** bash script

```
vi /home/cis90/bin/allscripts
```



The second part of **allscripts** is a long case statement that will run the requested student's **myscript** file located in the student's bin directory.

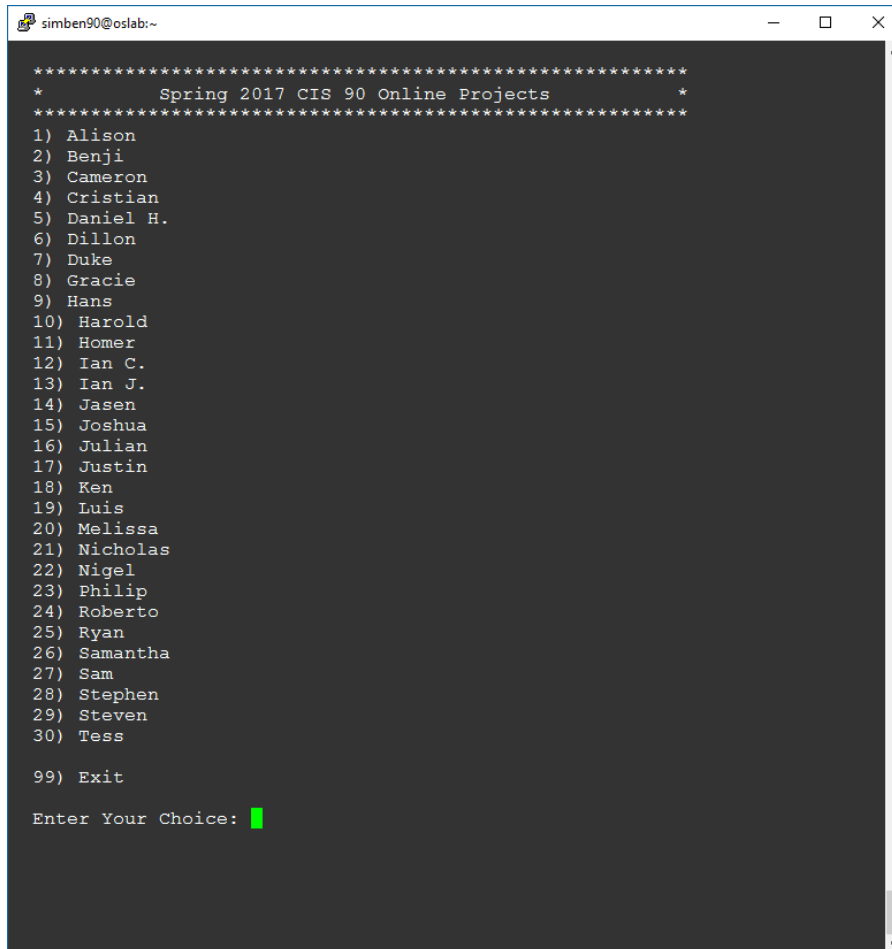
```
2) # Benji
    /home/cis90/simben/bin/myscript
```



Note the use of an absolute path to run each students script

The **allscripts** bash script

Running **allscripts** looks like this

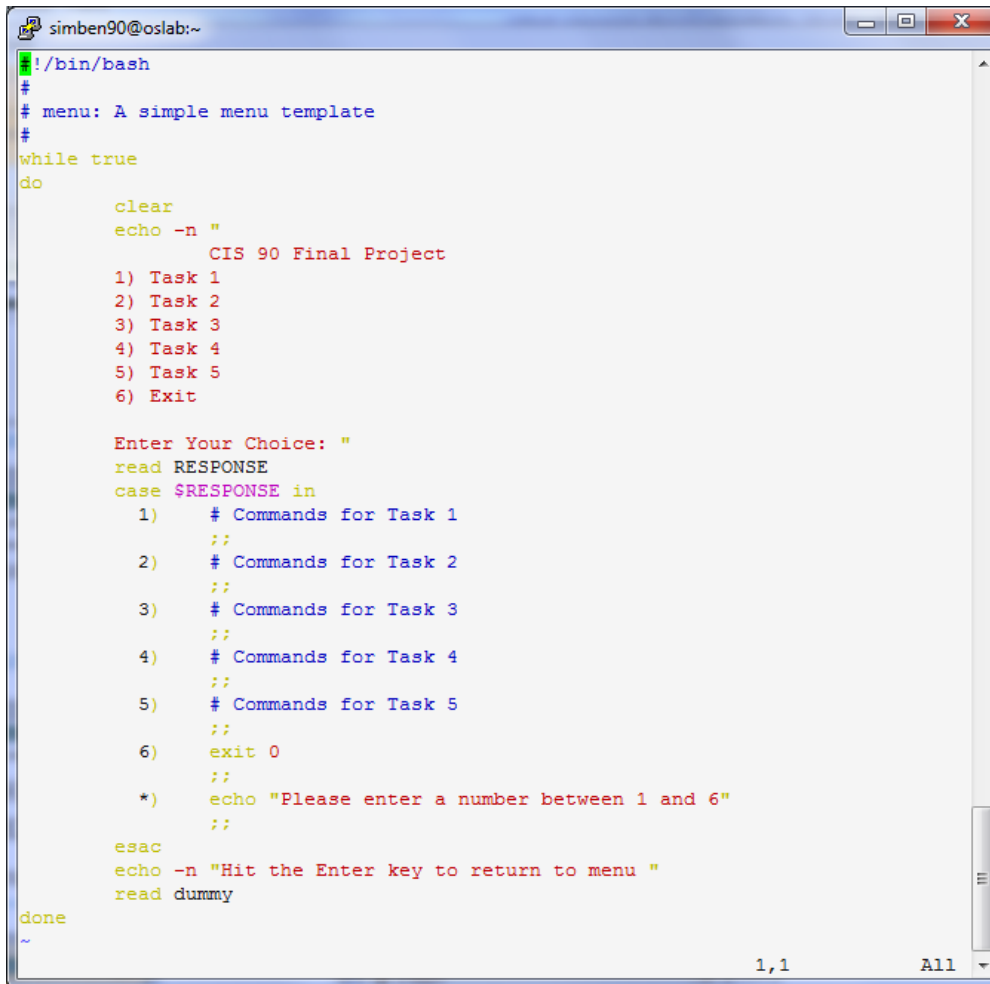


```
simben90@oslab:~  
*****  
*          Spring 2017 CIS 90 Online Projects          *  
*****  
1) Alison  
2) Benji  
3) Cameron  
4) Cristian  
5) Daniel H.  
6) Dillon  
7) Duke  
8) Gracie  
9) Hans  
10) Harold  
11) Homer  
12) Ian C.  
13) Ian J.  
14) Jasen  
15) Joshua  
16) Julian  
17) Justin  
18) Ken  
19) Luis  
20) Melissa  
21) Nicholas  
22) Nigel  
23) Philip  
24) Roberto  
25) Ryan  
26) Samantha  
27) Sam  
28) Stephen  
29) Steven  
30) Tess  
  
99) Exit  
  
Enter Your Choice: █
```

*This script has been updated with everyone's name and pathnames to each student's **myscript** file*

The **myscript** bash script

`vi ~/bin/myscript`



```
simben90@oslab:~
#!/bin/bash
#
# menu: A simple menu template
#
while true
do
    clear
    echo -n "
        CIS 90 Final Project
    1) Task 1
    2) Task 2
    3) Task 3
    4) Task 4
    5) Task 5
    6) Exit

    Enter Your Choice: "
    read RESPONSE
    case $RESPONSE in
        1)    # Commands for Task 1
            ;;
        2)    # Commands for Task 2
            ;;
        3)    # Commands for Task 3
            ;;
        4)    # Commands for Task 4
            ;;
        5)    # Commands for Task 5
            ;;
        6)    exit 0
            ;;
        *)    echo "Please enter a number between 1 and 6"
            ;;
    esac
    echo -n "Hit the Enter key to return to menu "
    read dummy
done
~
```

Every student will be creating a **myscript** file in their bin directory for the final project.

Your initial **myscript** file will look like this in vi

vi understands shell scripts and will use color syntax styling.



Final Project

Getting Started

- 1) On Opus, copy the *myscript* file in the class *depot/* directory to your *bin/* directory:

```
cd
```

```
cp ../depot/myscript bin/
```

- 2) Give your script execute permissions with:

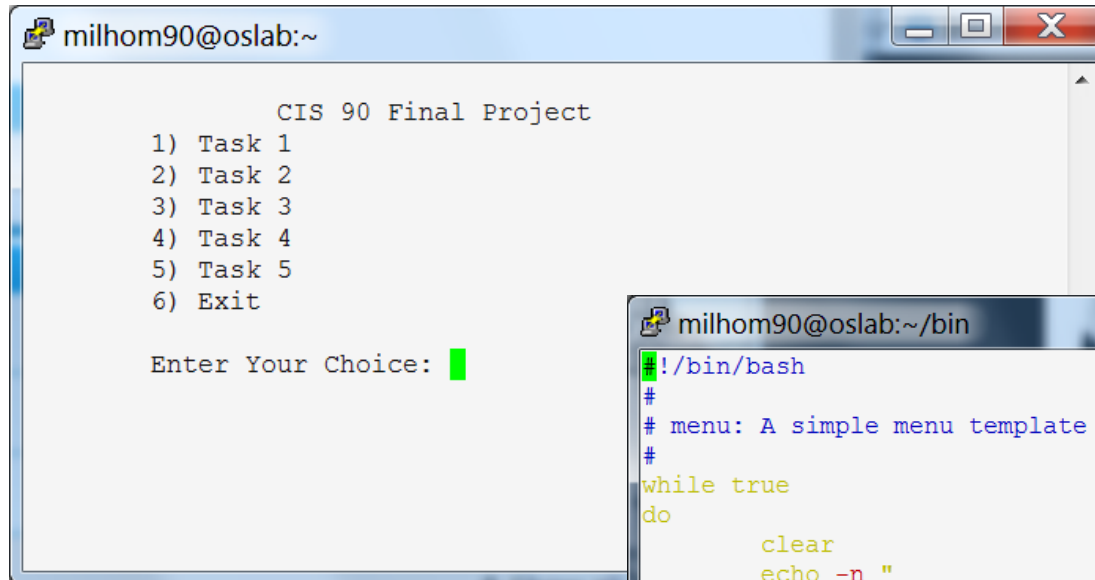
```
chmod +x bin/myscript
```

- 3) Run the script:

```
myscript
```

Final Project

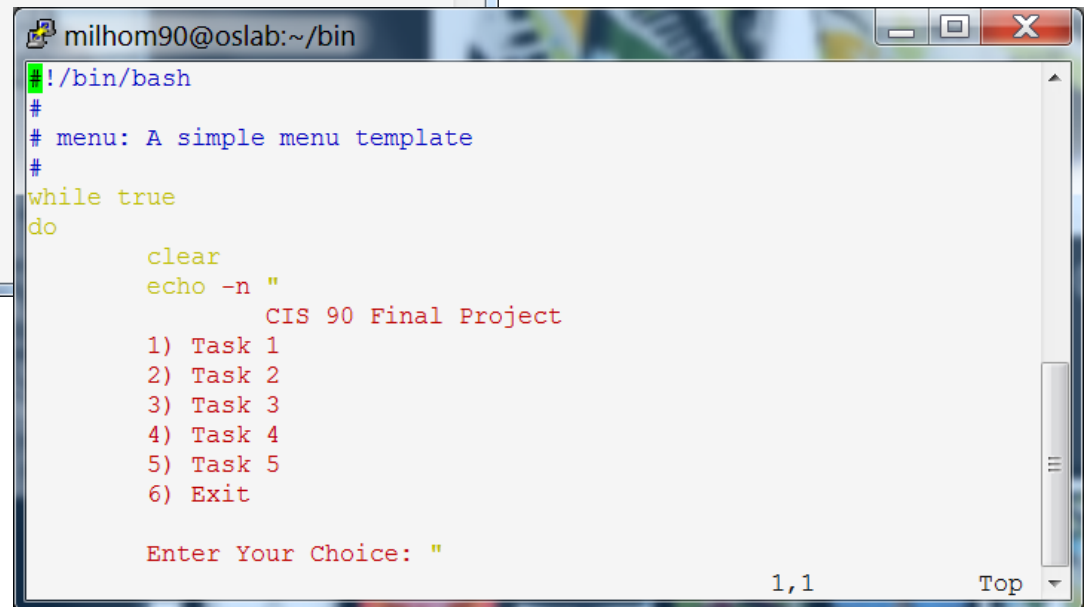
myscript



```
milhom90@oslab:~  
  
CIS 90 Final Project  
1) Task 1  
2) Task 2  
3) Task 3  
4) Task 4  
5) Task 5  
6) Exit  
  
Enter Your Choice: █
```

*Running and viewing
the generic mscript
file*

vi mscript



```
milhom90@oslab:~/bin  
#!/bin/bash  
#  
# menu: A simple menu template  
#  
while true  
do  
    clear  
    echo -n "  
        CIS 90 Final Project  
        1) Task 1  
        2) Task 2  
        3) Task 3  
        4) Task 4  
        5) Task 5  
        6) Exit  
  
    Enter Your Choice: "  
  
1,1 Top
```

Final Project Getting Started

myscript

```
milhom90@oslab:~  
  
      Rumpelstilskin's Final Project  
1) My favorite color  
2) Task 2  
3) Task 3  
4) Task 4  
5) Task 5  
6) Exit  
  
Enter Your Choice: █
```

*Editing the menu title
and option*

vi myscript

```
milhom90@oslab:~/bin  
#!/bin/bash  
#  
# menu: A simple menu template  
#  
while true  
do  
    clear  
    echo -n "  
        Rumpelstilskin's Final Project  
1) My favorite color █  
2) Task 2  
3) Task 3  
4) Task 4  
5) Task 5  
6) Exit  
  
    Enter Your Choice: "  
"myscript" 37L, 567C written    10,21-28    Top
```

Edit the menu title

Edit the first option choice

Final Project Getting Started

myscript

```
milhom90@oslab:~  
  
      Rumpelstilskin's Final Project  
1) My favorite color  
2) Task 2  
3) Task 3  
4) Task 4  
5) Task 5  
6) Exit  
  
Enter Your Choice: 1  
Hit the Enter key to return to menu
```

vi myscript

```
milhom90@oslab:~/bin  
Enter Your Choice: "  
read RESPONSE  
case $RESPONSE in  
  1)    # Commands for Task 1  
        ;;  
  2)    # Commands for Task 2  
        ;;  
  3)    # Commands for Task 3  
        ;;  
  4)    # Commands for Task 4  
        ;;  
  5)    # Commands for Task 5  
        ;;  
  6)    exit 0  
        ;;  
  *)    echo "Please enter a number between 1 and 6"  
        ;;  
;
```

*Running Task 1 doesn't
do anything yet*

Final Project Getting Started

myscript

```
milhom90@oslab:~  
  
          Rumpelstilskin's Final Project  
1) My favorite color  
2) Task 2  
3) Task 3  
4) Task 4  
5) Task 5  
6) Exit  
  
Enter Your Choice: 1  
What is your name? Homer  
What is your favorite color? Green  
Hi Homer, your favorite color is Green!  
Hit the Enter key to return to menu
```

We've implemented a simple task for option 1

vi myscript

```
milhom90@oslab:~/bin  
  
1) My favorite color  
2) Task 2  
3) Task 3  
4) Task 4  
5) Task 5  
6) Exit  
  
Enter Your Choice: "  
read RESPONSE  
case $RESPONSE in  
  1)  # favorite color mini-script  
      echo -n "What is your name? "  
      read name  
      echo -n "What is your favorite color? "  
      read color  
      echo "Hi $name, your favorite color is $color!"  
      ;  
"  
"myscript" 42L, 794C written      26,4-18      36%
```

Add these lines

Modify the comment line

Final Project Getting Started

A new command

```
read RESPONSE
case $RESPONSE in
  1)    # favorite color mini-script
        echo -n "What is your name? "
        read name
        echo -n "What is your favorite color? "
        read color
        echo "Hi $name, your favorite color is $color!"
        ;;
```

another new command

Final Project Getting Started

case statement begins here

```
read RESPONSE
case $RESPONSE in
  1)    # favorite color mini-script
        echo -n "What is your name? "
        read name
        echo -n "What is your favorite color? "
        read color
        echo "Hi $name, your favorite color is $color!"
        ;;
```

*First case ends
here*

*First case of case
statement starts here*

Final Project Getting Started

A variable (\$ means "the value of")

```
read RESPONSE
case $RESPONSE in
  1)    # favorite color mini-script
        echo -n "What is your name? "
        read name
        echo -n "What is your favorite color? "
        read color
        echo "Hi $name, your favorite color is $color!"
        ;;
```

another variable

another variable

Variables (\$ means "the value of")

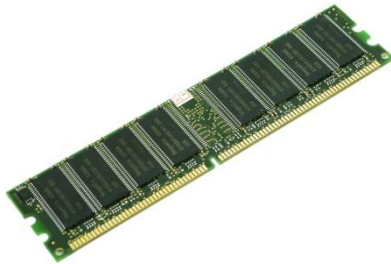
Final Project Getting Started

```
read RESPONSE
case $RESPONSE in
  1)    # favorite color mini-script
        echo -n "What is your name? "
        read name
        echo -n "What is your favorite color? "
        read color
        echo "Hi $name, your fovorite color is $color!"
        ;;
```

Comments begin with a #

Variables vs Files

Variables vs Files



We use **variables** to reference data in memory. For example: PS1, PATH, LOGNAME, color, name



We use **filenames** to reference data on hard drives. For example: */etc/passwd*, *sonnet1*, *letter*



Shell Variables

Shell Variables

SHELL LOGNAME HOME LANG
 SSH_TTY EUID PWD
 BASH_VERSION LINES COLORS PPID
 consoletype IFS
 MAILCHECK BASH_ENV SHELLOPTS HOSTNAME
 USER BASH PS4 TERM PIPESTATUS GROUPS
 HISTFILESIZE OPTIND BASH_VERSINFO
 BASH_ARGV PATH UID PS1
 SHLVL tmpid SSH_CONNECTION HISTFILE
 BASH_ARGC USERNAME OSTYPE
 HISTSIZE BASH_LINENO LESSOPEN
 HOSTTYPE OPTERR SSH_CLIENT
 COLUMNS LS_COLORS CVS_RSH
 INPUTRC BASH_SOURCE _ MACHTYPE
 PROMPT_COMMAND PS2
 DIRSTACK MAIL SSH_ASKPASS G_BROKEN_FILENAMES

Note the convention of using upper case

View all shell variables

/home/cis90/simben/Poems \$ **set | more**

```
BASH=/bin/bash
BASH_ARGC=()
BASH_ARGV=()
BASH_ENV=/home/cis90/simben/.bashrc
BASH_LINENO=()
BASH_SOURCE=()
BASH_VERSINFO=([0]="3" [1]="2" [2]="25" [3]="1" [4]="release"
[5]="i686-redhat-linux-gnu")
BASH_VERSION='3.2.25(1)-release'
COLORS=/etc/DIR_COLORS.xterm
COLUMNS=80
CVS_RSH=ssh
DIRSTACK=()
EUID=1160
GROUPS=()
G_BROKEN_FILENAMES=1
HISTFILE=/home/cis90/simben/.bash_history
HISTFILESIZE=1000
HISTSIZE=1000
HOME=/home/cis90/simben
HOSTNAME=opus.cabrillo.edu
HOSTTYPE=i686
IFS=$' \t\n'
IGNOREEOF=10
INPUTRC=/etc/inputrc
LANG=en_US.UTF-8
LESSOPEN='|/usr/bin/lesspipe.sh %s'
LINES=24
LOGNAME=simben
```

```
LS_COLORS='no=00:fi=00:di=00;34:ln=00;36:pi=40;33:so=00;35
:bd=40;33;01:cd=40;33;01:or=01;05;37;41:mi=01;05;37;41:ex=
00;32:*.cmd=00;32:*.exe=00;32:*.com=00;32:*.btm=00;32:*.ba
t=00;32:*.sh=00;32:*.csh=00;32:*.tar=00;31:*.tgz=00;31:*.a
rj=00;31:*.taz=00;31:*.lzh=00;31:*.zip=00;31:*.z=00;31:*.Z
=00;31:*.gz=00;31:*.bz2=00;31:*.bz=00;31:*.tz=00;31:*.rpm=
00;31:*.cpio=00;31:*.jpg=00;35:*.gif=00;35:*.bmp=00;35:*.x
bm=00;35:*.xpm=00;35:*.png=00;35:*.tif=00;35:'
MACHTYPE=i686-redhat-linux-gnu
MAIL=/var/spool/mail/simben
MAILCHECK=60
OLDPWD=/home/cis90/simben
OPTERR=1
OPTIND=1
OSTYPE=linux-gnu
PATH=/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/home/
cis90/simben/./bin:/home/cis90/simben/bin:.
PIPESTATUS=([0]="0")
PPID=26514
PROMPT_COMMAND='echo -ne
"\033]0;${USER}@${HOSTNAME%%.*}:${PWD/#$HOME/~}"; echo -ne
"\007"'
PS1='$PWD $'
PS2='> '
PS4='+ '
PWD=/home/cis90/simben/Poems
SHELL=/bin/bash
SHELLOPTS=braceexpand:emacs:hashall:histexpand:ignoreeof:i
nteractive-comments:monitor
SHLVL=1
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
TERM=xterm
UID=1160
USER=simben
USERNAME=
_=env
consoletype=pty
```

*The **set** command, with no arguments, will show all shell variables and their values*

Using Shell Variables

- Shell variables names consist of alpha-numeric characters.
- Variables defined by the Operating System are uppercase, e.g. TERM, PS1, PATH
- The **set** command will display all the shell's current variables and their values.
- Shell variables are initialized using the assignment operator:
For example: **TERM=vt100**
Note: Quotes must be used for white space: **VALUE="any value"**
- Variables may be viewed using the echo command:
e.g. **echo \$TERM**
The \$ in front of a variable name denotes the value of that variable.
- To remove a variable, use the unset command: **unset PS1**
- Shell variables hold their values for the duration of the session i.e. until the shell is exited

Showing the values of variables

Use: **echo \$varname**

Think of the \$ metacharacter as "the value of"

Example 1

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

Example 2

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

Example 3

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

Example 4

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

Setting the values of variables

Use: **varname=value**

Do NOT use the \$ when setting a variable

(no spaces please around the =)

Example 1

```
[rsimms@nosmo ~]$ PS1="By your command >"
By your command >
By your command >PS1="What can I do for you $LOGNAME? "
What can I do for you rsimms?
What can I do for you rsimms?
```

Example 2

```
/home/cis90/simben/bin $ river="The Amazon"
/home/cis90/simben/bin $ echo $river
The Amazon
/home/cis90/simben/bin $ echo river
river
```

Creating Shell Variables

1

```
/home/cis90/simmen/bin $ echo $defrost $ac $fan
```

```
/home/cis90/simmen/bin $
```

the value of a variable that has not been created is null

2

```
/home/cis90/simmen/bin $ defrost=on
```

```
/home/cis90/simmen/bin $ ac=off
```

```
/home/cis90/simmen/bin $ fan=medium
```

create some new shell variables and assign values

3

```
/home/cis90/simmen/bin $ echo $defrost $ac $fan  
on off medium
```

*print the **values** of the shell variables*

```
/home/cis90/simmen/bin $ echo defrost ac fan  
defrost ac fan
```

*print the **names** of the shell variables*

Shell Variables

```
/home/cis90/simben $ defrost=on
/home/cis90/simben $ ac=off
/home/cis90/simben $ fan=medium
/home/cis90/simben $ set
```

*Note: Any new variables you initialize will show up in the output of the **set** command*

```
BASH=/bin/bash
BASH_ARGC=()
BASH_ARGS=( )
BASH_ENV=/home/cis90/simben/.bashrc
BASH_LINENO=()
BASH_SOURCE=()
BASH_VERSION='[0]="3" [1]="2" [2]"25" [3]="%" [4]"release" [5]"1686-redhat-linux-gnu"'
COLORS="/etc/dirs_colors.xterm"
COLUMN=44
CVS_RSH=ssh
DIRSTACK=()
EUID=1156
GROUPS=()
C BROKEN_FILENAMEES=1
HISTFILE=/home/cis90/simben/.bash_history
HISTFILESIZE=100
HISTSZIE=1000
HOME=/home/cis90/simben
HOSTNAME=npsn.cabrillo.edu
HOSTTYPE=i686
IFS=$'\t\n'
IGNOREREOF=1
INPUTRC=/etc/inputrc
LANG=en_US.UTF-8
LESSOPEN=| /usr/bin/lesspipe.sh %s
LINES=39
LOGNAME=simben
LC_CTYPE=C;_GLIBC__24:_GLIBCVER=2.14;_POSIX_VERS=2013;_POSIX_XSI=2013;_XBSI=2013;_GNU=2013;_GNU_LIBC_VER=2.14;_GNU_LIBC_MAJOR=2;_GNU_LIBC_MINOR=16;_GNU_LIBC_PATCHLEVEL=86;-redhat-linux-gnu
MAIL=/var/spool/mail/simben
MAILCHECK=60
OLDPWD=/home/cis90/simben/edits
OPTERR=1
OPTARG=
OPTIND=1
OPTSPEC=:Linux-gnu
PATH=/usr/katberos/bin:/usr/local/bin:/bin:/usr/bin/home/cis90/simben/.:/bin:/home/cis90/simben/bin.:
PAGER=pag -O="-G"
PID=7254
PPID? COMMAND="echo -ne "[03]o$USER)[8$(HOSTNAME%.*)]<(FMD:(MOS&&-))"; echo -ne "[00T"]
PS1="$PS0 $ "
PS2="$ "
PWD=/home/cis90/simben
SHELL=/bin/bash
shellsExpand=>macos:haskell:hietexpand;ignoreeof:interactive-comments:monitor
shlvl=1
SSH_ASKPASS=/usr/libexec/openssh/gnome-smb-askpass
SSH_CLIENT="63.249.103.107 19509 22"
SSH_CONNECTIO="63.249.103.107 19509 207.62.186.9 22"
SSH_TTY=/dev/ppts/r
TERM=xterm
UID=1156
USER=simben
USERNAME=
```

command

font reduced for the other variables to fit on slide

font reduced for the other variables to fit on slide

ac=off

```
defrost=on
```

fan=medium

Shell Variables

Using grep to find a variable in the output of the set command

```
/home/cis90/simben $ set | grep defrost  
defrost=on
```

The output of the set command is piped to the grep command which displays only lines containing "defrost"

Class Activity

Create and initialize three new variables:

defrost=on

ac=off

fan=medium

Show the names of the variables:

echo defrost ac fan

Show the values of the variables:

echo \$defrost \$ac \$fan

Display all variables and locate yours:

set

set | grep defrost

set | grep "^ac"

set | grep fan

*The ^ means look for ac
starting in column 1 only*

*Paste the output from **set | grep fan** in the chat window*

Removing Shell Variables

To remove a variable, use the unset command: **unset PS1**

```
/home/cis90/simben $ echo $defrost $ac $fan      show values  
on off medium
```

```
/home/cis90/simben $ unset defrost  
/home/cis90/simben $ echo $defrost $ac $fan      remove one of the  
off medium                                     variables
```

```
/home/cis90/simben $ unset ac fan                remove remaining  
/home/cis90/simben $ echo $defrost $ac $fan      variables
```

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

Class Exercise

Delete your three new variables:

```
unset defrost
```

```
unset ac fan
```

Show the names of the variables:

```
echo defrost ac fan
```

Show the values of the variables:

```
echo $defrost $ac $fan
```

```
echo "defrost=$defrost"
```

*Paste the output from **echo "defrost=\$defrost"** into the chat window*

Shell Variables

Variables are often used in scripts when you need a temporary placeholder to store some data

1

```
/home/cis90/simben $ vi funscript
/home/cis90/simben $ cat funscript
#!/bin/bash
echo -n "Turn the Air Conditioning on or off? "
read ac
echo "Air Conditioning set to $ac"
exit
```

Create a script that uses a variable named "ac" to hold the status of an air conditioner.

Prompt the user and input what they type into the this variable.

2

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

Add execute permissions so the script can be run

3

```
/home/cis90/simben $ ./funscript
Turn the Air Conditioning on or off? off
Air Conditioning set to off
```

Run the script

Class Exercise

Now make this little user dialog script:

```
vi funscript
```

insert the following lines then save

```
#!/bin/bash  
echo -n "Turn the Air Conditioning on or off? "  
read ac  
echo "Air Conditioning set to $ac"  
exit
```

```
chmod +x funscript
```

```
./funscript
```

Do a long listing on funscript and paste the output into the chat window



Environment Variables

Environment Variables

SHELL **SSH_TTY** **LOGNAME** **HOME** **LANG**
 BASH_VERSION EUID **PWD**
 MAILCHECK consoletype IFS LINES COLORS PPID
USER BASH **BASH_ENV** **HOSTNAME**
 HISTFILESIZE PS4 **TERM** PIPESTATUS GROUPS
 BASH_ARGV **PATH** UID BASH_VERSINFO PS1
SHLVL tmpid **SSH_CONNECTION** HISTFILE
 BASH_ARGC **USERNAME** OSTYPE
HISTSIZE BASH_LINENO **LESSOPEN**
 HOSTTYPE OPTERR **SSH_CLIENT**
 COLUMNS **LS_COLORS** **CVS_RSH**
 PROMPT_COMMAND **INPUTRC** BASH_SOURCE _ MACHTYPE
 DIRSTACK **MAIL** **SSH_ASKPASS** **G_BROKEN_FILENAMES**
 PS2

Use the **env** to see which of the shell variables have been exported and therefore are environment variables (shown in bold/green above)

View all Environment (exported) Variables

```
[simben@opus ~]$ env
```

```
HOSTNAME=opus.cabrillo.edu
SHELL=/bin/bash
TERM=xterm
HISTSIZE=1000
SSH_CLIENT=63.249.103.107 20807 22
SSH_TTY=/dev/pts/0
USER=simben
LS_COLORS=no=00;fi=00;di=00;34:ln=00;36:pi=40;33:so=00;35:bd=40;33;01:cd=40;33;01:or=01;05;37;41:mi=01;05;37;41:ex=00;32:*.cmd=00;32:*.exe=00;32:*.com=00;32:*.btm=00;32:*.bat=00;32:*.sh=00;32:*.csh=00;32:*.tar=00;31:*.tgz=00;31:*.arj=00;31:*.taz=00;31:*.lzh=00;31:*.zip=00;31:*.z=00;31:*.Z=00;31:*.gz=00;31:*.bz2=00;31:*.bz=00;31:*.tz=00;31:*.rpm=00;31:*.cpio=00;31:*.jpg=00;35:*.gif=00;35:*.bmp=00;35:*.xbm=00;35:*.xpm=00;35:*.png=00;35:*.tif=00;35:
USERNAME=
PATH=/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/home/cis90/simben/../../bin:/home/cis90/simben/bin:
MAIL=/var/spool/mail/simben
PWD=/home/cis90/simben
INPUTRC=/etc/inputrc
LANG=en_US.UTF-8
fan=medium
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
HOME=/home/cis90/simben
SHLVL=2
BASH_ENV=/home/cis90/simben/.bashrc
LOGNAME=simben
CVS_RSH=ssh
SSH_CONNECTION=63.249.103.107 20807 207.62.186.9 22
LESSOPEN=|/usr/bin/lesspipe.sh %s
G_BROKEN_FILENAMES=1
_=/bin/env
```

The env command by itself will list all the environment (exported) variables

View all Environment (exported) Variables

```
[simben@opus ~]$ export
```

```
declare -x BASH_ENV="/home/cis90/simben/.bashrc"
declare -x CVS_RSH="ssh"
declare -x G_BROKEN_FILENAMES="1"
declare -x HISTSIZE="1000"
declare -x HOME="/home/cis90/simben"
declare -x HOSTNAME="opus.cabrillo.edu"
declare -x INPUTRC="/etc/inputrc"
declare -x LANG="en_US.UTF-8"
declare -x LESSOPEN="|/usr/bin/lesspipe.sh %s"
declare -x LOGNAME="simben"
declare -x
LS_COLORS="no=00:fi=00:di=00;34:ln=00;36:pi=40;33:so=00;35:bd=40;33;01:cd=40;33;01:or=01;05;37;41:mi=01;05;37;41:ex=00;32:*.cmd=00;32:*.exe=00;32:*.com=00;32:*.btm=00;32:*.bat=00;32:*.sh=00;32:*.csh=00;32:*.tar=00;31:*.tgz=00;31:*.arj=00;31:*.taz=00;31:*.lzh=00;31:*.zip=00;31:*.z=00;31:*.Z=00;31:*.gz=00;31:*.bz2=00;31:*.bz=00;31:*.tz=00;31:*.rpm=00;31:*.cpio=00;31:*.jpg=00;35:*.gif=00;35:*.bmp=00;35:*.xbm=00;35:*.xpm=00;35:*.png=00;35:*.tif=00;35:"
declare -x MAIL="/var/spool/mail/simben"
declare -x OLDPWD
declare -x
PATH="/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/home/cis90/simben/../../bin:/home/cis90/simben/bin:."
declare -x PWD="/home/cis90/simben"
declare -x SHELL="/bin/bash"
declare -x SHLVL="2"
declare -x SSH_ASKPASS="/usr/libexec/openssh/gnome-ssh-askpass"
declare -x SSH_CLIENT="63.249.103.107 20807 22"
declare -x SSH_CONNECTION="63.249.103.107 20807 207.62.186.9 22"
declare -x SSH_TTY="/dev/pts/0"
declare -x TERM="xterm"
declare -x USER="simben"
declare -x USERNAME=""
```

*The **export** command by itself will list all the exported (environment) variables.*

*Similar to **env** command but different output format*



Using Environment (exported) Variables

- Environment variables are a special subset of the shell variables.
- Environment variables are shell variables that have been *exported*.
- The **env** command will display the current environment variables and their values. Using the **export** command with no arguments will also show all the environment variables.
- The **export** command is used to make a shell variable into an environment variable.

dog=benji; export dog
or **export dog=benji**

- The **export -n** command is used to make an environment variable back into a normal shell variable. E.g. **export -n dog** makes dog back into a regular shell variable.
- **Child processes are provided copies of the parent's environment variables.**
- **Any changes made by the child will not affect the parent's copies.**

Shell (Environment) Variables

export command - show all exported variables

To create your own environment variable use the **export** command

1

```
/home/cis90/simben $ env | wc -l
29
/home/cis90/simben $ export | wc -l
29
```

There are currently 29
environment (exported)
variables

2

```
/home/cis90/simben $ fan=medium
/home/cis90/simben $ export fan
```

Create a new shell variable named
fan and export it so it becomes an
environment variable

3

```
/home/cis90/simben $ env | wc -l
30
/home/cis90/simben $ export | wc -l
30
```

Now there are 30
environment variables

4

```
[simben@opus ~]$ export | grep fan
declare -x fan="medium"
[simben@opus ~]$ env | grep fan
fan=medium
[simben@opus ~]$ set | grep fan
fan=medium
```

use grep to show fan is an environment
(exported) shell variable

use grep to show fan is a
shell variable

Class Exercise

Recreate the variable named fan:

fan=high

Show that fan is now one of your shell variables:

set | grep fan

Show that fan is not exported:

env | grep fan

Now export fan:

export fan

env | grep fan

*Paste the output from **env | grep fan** into the chat window*



Shell Environment



The Shell Environment

- The shell environment can be customized using the environment variables.
- Commands in the shell environment can be customized using aliases.
- Aliases and environment variable settings can be made permanent using the hidden *.bash_profile* and *.bashrc* files in the users home directory.
- Environment variables can be exported so they are available to child processes.

Shell (Environment) Variables

Some famous environment variables

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

Class Exercise

Echo three environment variables as follows:

echo "I'm in \$PWD using \$SHELL and my username is \$LOGNAME"

Paste the output you get into the chat window

bash shell tip

changing the prompt

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

The prompt string can have any combination of text, variables and these codes.

Customizing the shell prompt with PS1

PS1 settings	Result
<code>PS1='\$PWD \$'</code>	<code>/home/cis90/simben/Poems \$</code>
<code>PS1="\w \$"</code>	<code>~/Poems \$</code>
<code>PS1="\W \$"</code>	<code>Poems \$</code>
<code>PS1="\u@\h \$"</code>	<code>simben90@opus \$</code>
<code>PS1='\u@\h \$PWD \$'</code>	<code>simben90@opus /home/cis90/simben/Poems \$</code>
<code>PS1='\u@\\$HOSTNAME \$PWD \$'</code>	<code>simben90@opus.cabrillo.edu /home/cis90/simben/Poems \$</code>
<code>PS1='\u \! \$PWD \$'</code>	<code>simben90 825 /home/cis90/simben/Poems \$</code>
<code>PS1="\d [\u@\h \W/] \\$ "</code>	<code>Mon Nov 16 [simben90@oslab Poems/] \$</code>
<code>PS1="Enter command: "</code>	<code>Enter command:</code>

Important: Use single quotes around variables that change. For example if you use `$PWD` with double quotes, the prompt will **not** change as you change directories!

Class Exercise

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

Make a new prompt using one or more of the special prompt codes:

PS1="make your own prompt here"

Paste your new prompt into the chat window

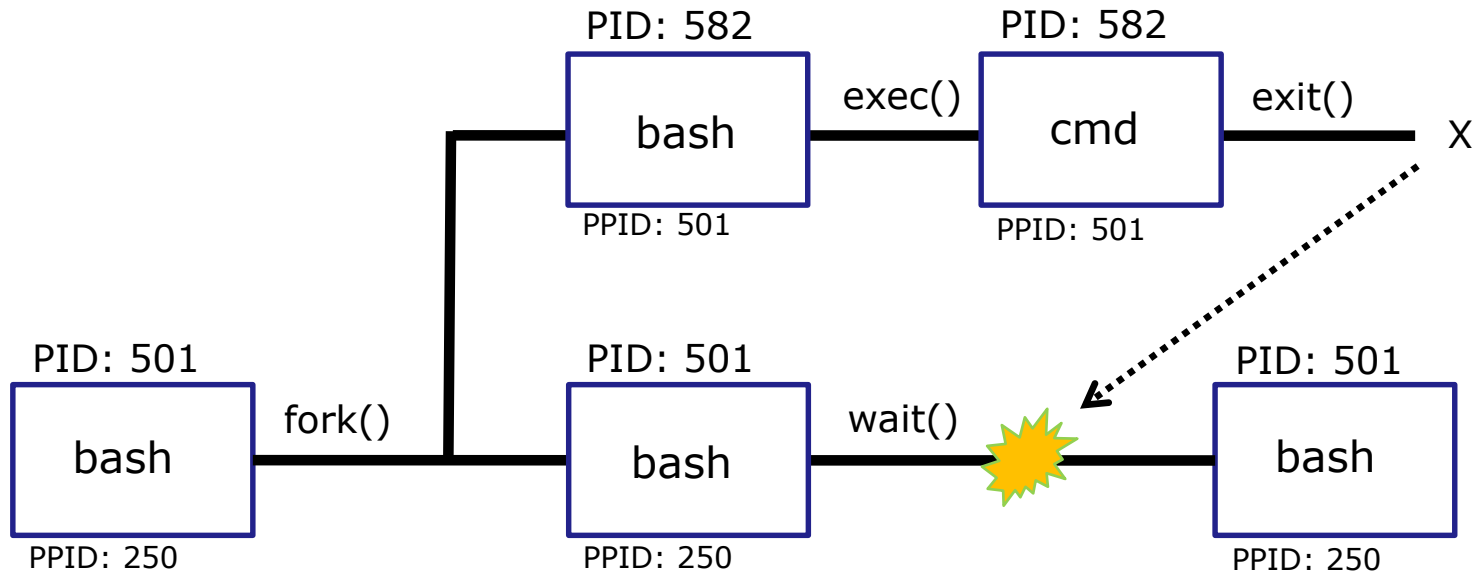


Variables and child processes

The rules of the road for variables

1. When a shell forks a child, only copies of exported variables are made available to the child.
2. A child can modify the variables it receives but those modifications will not change the parent's variables.

exporting variables



- When a shell forks a child, only copies of exported variables are made available to the child.
- A child can modify the variables it receives but those modifications will not change the parent's variables.

The rules of the road for variables

1. When a shell forks a child, only copies of exported variables are made available to the child.
2. A child can modify the variables it receives but those modifications will not change the parent's variables.

Only exported variables are available to the child

- 1

parent

```

/home/cis90/simben $ window=down
/home/cis90/simben $ echo $window $LOGNAME
down simben90

```

*Create a new variable
named window*

- 2

parent

```

/home/cis90/simben $ env | grep window
/home/cis90/simben $ set | grep window
window=down

/home/cis90/simben $ env | grep LOGNAME
LOGNAME=simben90
/home/cis90/simben $ set | grep LOGNAME
LOGNAME=simben90

```

*window is a shell variable
that has **not** been
exported.*

*LOGNAME is an
environment variable that
has been exported.*

- 3

child

```

/home/cis90/simben $ bash
[simben@opus ~]$ echo $window $LOGNAME
simben90
[simben@opus ~]$ exit
exit

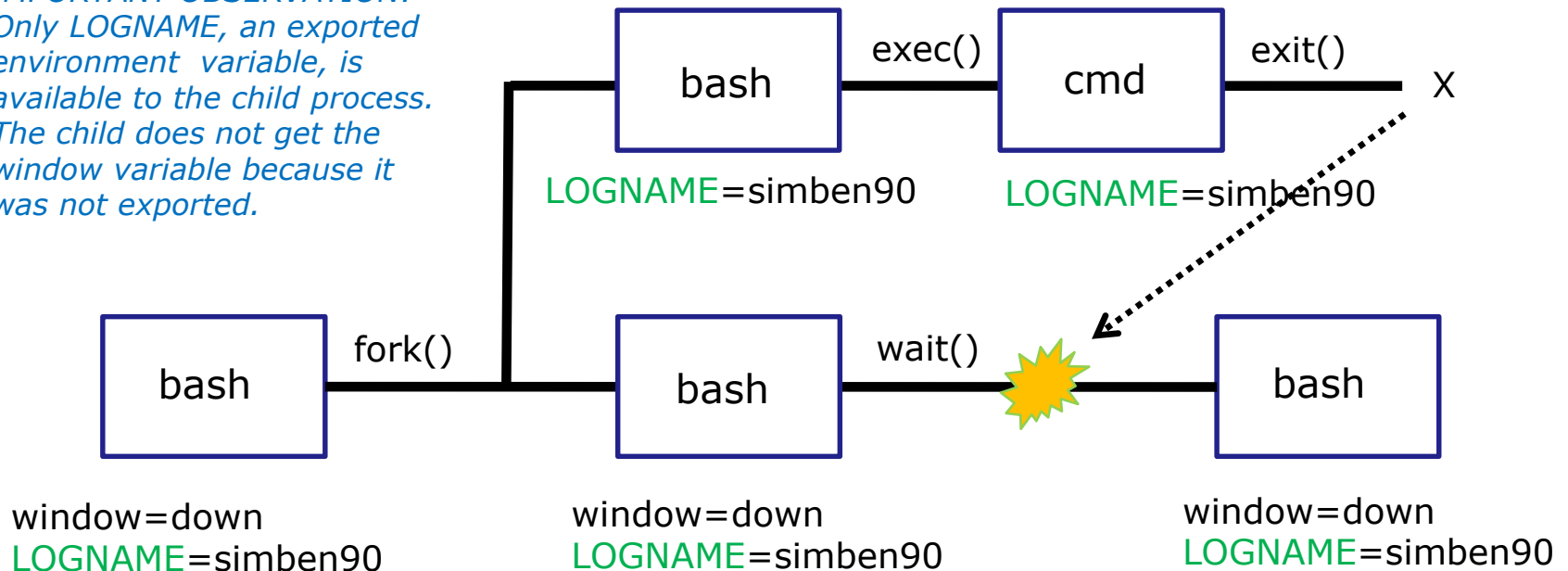
```

*Running the bash command
starts another bash process
as a child of the current bash
process. LOGNAME has a
value, but there is no window
variable.*

IMPORTANT OBSERVATION: Only LOGNAME, an exported environment variable, is available to the child process. The child does not get the window variable because it was not exported.

Only exported variables are available to the child

*IMPORTANT OBSERVATION:
Only LOGNAME, an exported
environment variable, is
available to the child process.
The child does not get the
window variable because it
was not exported.*



- When a shell forks a child, not all of the variables are passed on to the child.
- Only copies of the parent's exported variables are passed to the child.

The rules of the road for variables

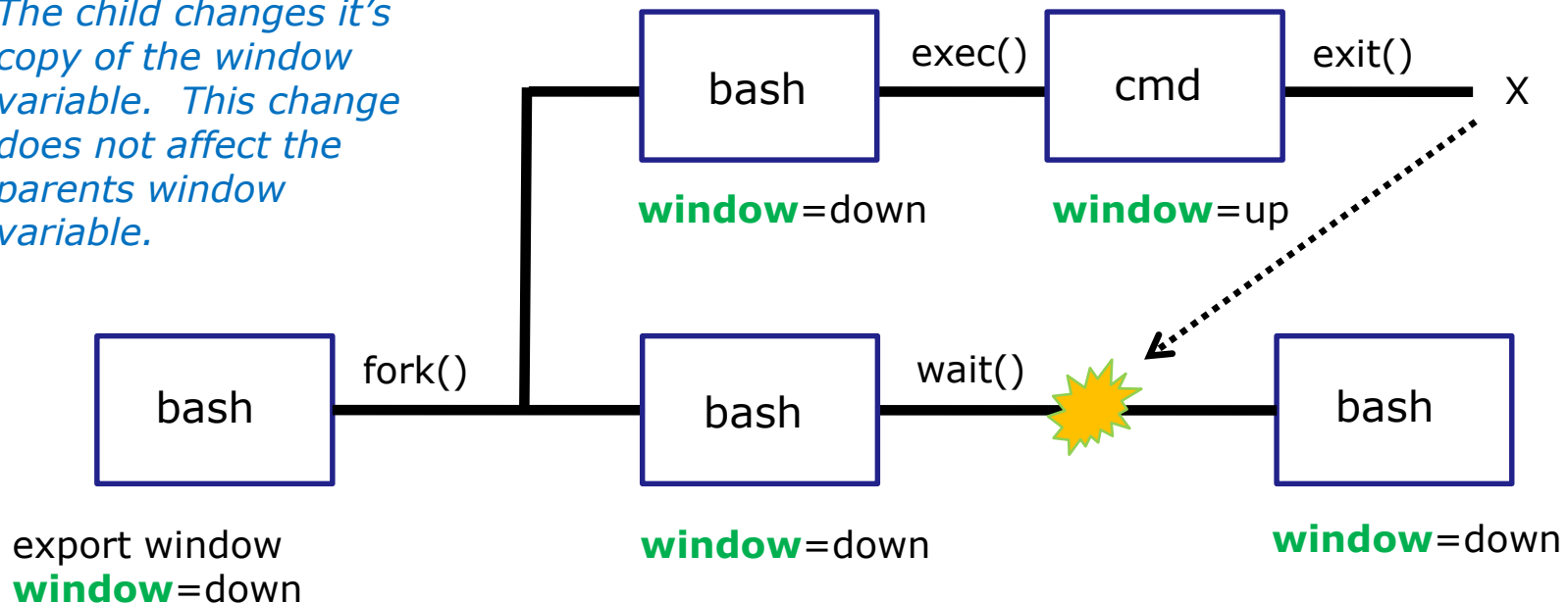
1. When a shell forks a child, only copies of exported variables are made available to the child.
2. A child can modify the variables it receives but those modifications will not change the parent's variables.

Changes made by the child do not affect the parent

- | | | | |
|---|--------|--|---|
| 1 | parent | <pre>/home/cis90/simben \$ echo \$window down /home/cis90/simben \$ export window</pre> | <p><i>export window so it is available to children</i></p> |
| 2 | child | <pre>/home/cis90/simben \$ bash [simben@opus ~]\$ echo \$window down</pre> | <p><i>a copy of window is now available to the child process</i></p> |
| 3 | child | <pre>[simben@opus ~]\$ window=up [simben@opus ~]\$ echo \$window up [simben@opus ~]\$ exit exit</pre> | <p><i>the child modifies the window variable</i></p> |
| 4 | parent | <pre>/home/cis90/simben \$ echo \$window down</pre> | <p><i>The modifications made by the child do not affect the parent's variable</i></p> |

Changes made by the child do not affect the parent

The child changes its copy of the window variable. This change does not affect the parents window variable.



- A child can modify the variables it receives but those modifications will not change the parent's variables.

Class Exercise

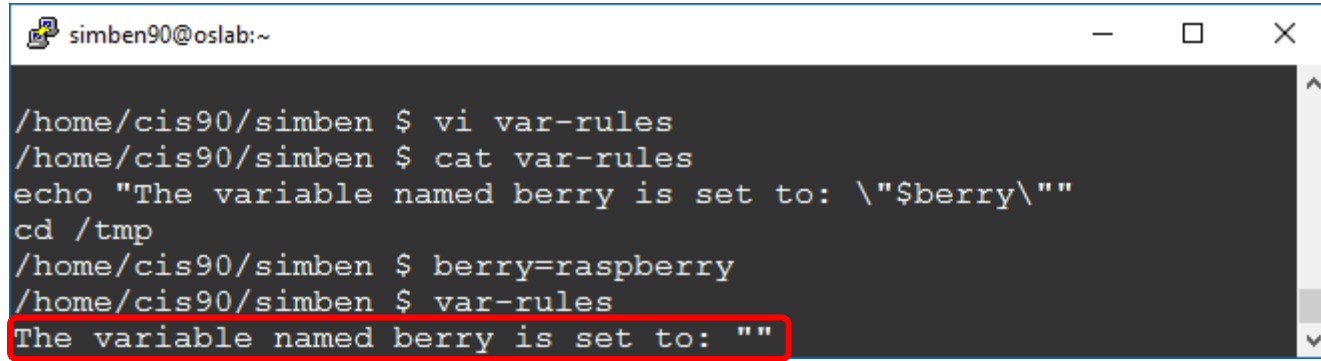
Look at the commands in this executable script:

```
/home/cis90/simben $ chmod +x var-rules  
/home/cis90/simben $ cat var-rules  
echo "The variable named berry is set to: \"$berry\""  
cd /tmp
```

What would be the output of running the script as follows:

```
berry=raspberry  
var-rules
```

Paste your answer into the chat window



```
simben90@oslab:~  
/home/cis90/simben $ vi var-rules  
/home/cis90/simben $ cat var-rules  
echo "The variable named berry is set to: \"${berry}\""  
cd /tmp  
/home/cis90/simben $ berry=raspberry  
/home/cis90/simben $ var-rules  
The variable named berry is set to: ""
```

Child can only see variables the parent exports and berry was NOT exported

Class Exercise

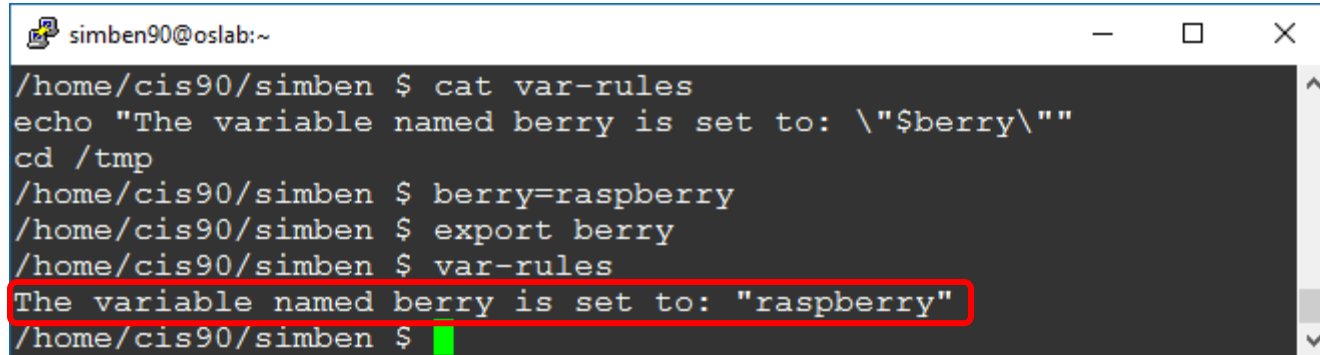
Look at the commands in this executable script:

```
/home/cis90/simben $ chmod +x var-rules  
/home/cis90/simben $ cat var-rules  
echo "The variable named berry is set to: \"$berry\""  
cd /tmp
```

What would be the output of running the script as follows:

```
berry=raspberry  
export berry  
var-rules
```

Paste your answer into the chat window



```
simben90@oslab:~  
/home/cis90/simben $ cat var-rules  
echo "The variable named berry is set to: \"${berry}\""  
cd /tmp  
/home/cis90/simben $ berry=raspberry  
/home/cis90/simben $ export berry  
/home/cis90/simben $ var-rules  
The variable named berry is set to: "raspberry"  
/home/cis90/simben $
```

Child can only see variables the parent exported and berry was exported.

Class Exercise

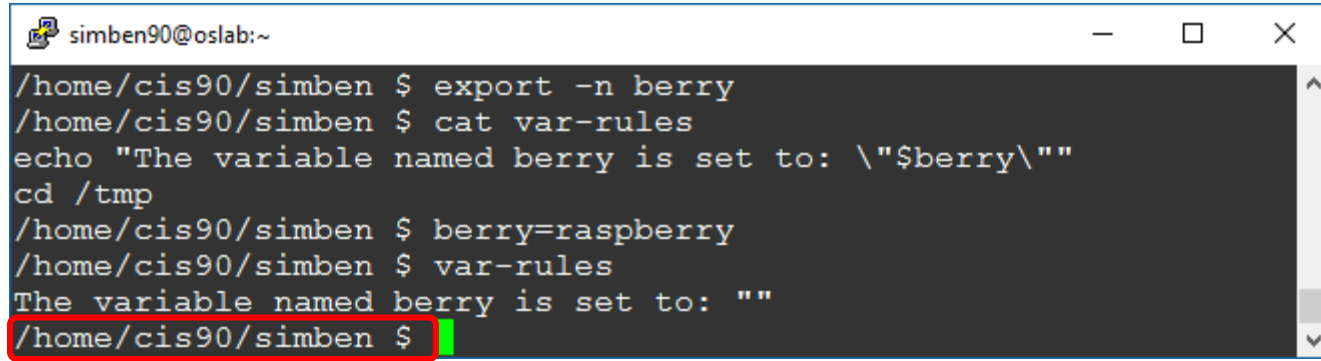
Look at the commands in this executable script:

```
/home/cis90/simben $ chmod +x var-rules  
/home/cis90/simben $ cat var-rules  
echo "The variable named berry is set to: \"$berry\""  
cd /tmp
```

What directory would you be in after running the script as follows:

```
berry=raspberry  
var-rules
```

Paste your answer into the chat window



```
simben90@oslab:~  
/home/cis90/simben $ export -n berry  
/home/cis90/simben $ cat var-rules  
echo "The variable named berry is set to: \"$berry\""  
cd /tmp  
/home/cis90/simben $ berry=raspberry  
/home/cis90/simben $ var-rules  
The variable named berry is set to: ""  
/home/cis90/simben $
```

Child cannot change parent's variables, like PWD

Aliases

alias command

(a shell builtin)

```
alias [-p] [name[=value] ...]
```

Alias with no arguments or with the `-p` option prints the list of aliases in the form `alias name=value` on standard output. When arguments are supplied, an alias is defined for each name whose value is given. A trailing space in value causes the next word to be checked for alias substitution when the alias is expanded. For each name in the argument list for which no value is supplied, the name and value of the alias is printed. Alias returns true unless a name is given for which no alias has been defined.

Note aliases are not expanded by default in non-interactive shell, and it can be enabled by setting the `expand_aliases` shell option using `shopt`.

Now you can give your own name to commands!

alias command

Example: Make a new name for the cp command

1 /home/cis90/simben \$ **alias copy=cp**
/home/cis90/simben \$ **copy lab09 /home/rsimms/turnin/cis90/lab09.\$LOGNAME**
/home/cis90/simben \$

2 /home/cis90/simben \$ **type copy**
copy is aliased to `cp`
/home/cis90/simben \$

*The **type** command shows that copy is an alias*

3 /home/cis90/simben \$ **alias copy**
alias copy='cp'
/home/cis90/simben \$

*The **alias** command (without an "=" sign) shows what the alias is*

4 /home/cis90/simben \$ **unalias copy**
/home/cis90/simben \$ **alias copy**
-bash: alias: copy: not found

*Use **unalias** command to remove an alias*

alias command

Example: Make an alias, called s, that prints the first 5 lines of small_town

1

```
/home/cis90/simben $ alias s="clear; head -n5 ~/edits/small_town"  
/home/cis90/simben $ s  
HOW SMALL IS SMALL?
```

```
YOU KNOW WHEN YOU'RE IN A SMALL TOWN WHEN...
```

```
The airport runaway is terraced.
```

```
The polka is more popular than a moshpit on Saturday night.
```

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

2

```
/home/cis90/simben $ type s  
s is aliased to `clear; head -n5 ~/edits/small_town'  
/home/cis90/simben $ alias s  
alias s='clear; head -n5 ~/edits/small_town'
```

*The **type** and **alias** commands show that s is an alias*

3

```
/home/cis90/simben $ unalias s  
/home/cis90/simben $
```

*Use **unalias** command to remove an alias*

alias an alias

Yes, an alias can be made using another alias

1

```
/home/cis90/simben $ alias show=cat
/home/cis90/simben $ alias mira=show
```

Make **show** an alias of **cat**
Make **mira** an alias of **show**

```
/home/cis90/simben $ show letter
```

hello mother! hello father!
here I am at camp mumsade. things are very entertaining,
and they say we'll have some fun when it stops raining.
all the mosquitoes hate the weather, and the lake has
dried up. no summer lemons either! so get
promote something last night after dinner.
how I don't want this to mean you, but my bunk mate has
broke. no summer, no way! they say? they want to
organize a reading party.
take me home, oh mother, father, take me home! I don't want
don't leave me out in the forest when I might get eaten
by a bear! Take me home, I promise that I don't have mosquitoes,
it says the house will have bugs, oh please don't make me
stay -- I'll have some one make my.
dearest father, dearest mother, how's my precious little
brother? I will come home if you like me. I will come.
let Aunt Martha hug and kiss me!
well a minute! it's stopped raining! My are wonderful!
My are wonderful! Playing baseball, get that's better!
Mother, Father, kindly disregard this letter.
Alan Sherman

reduced size to fit on page

2

```
/home/cis90/simben $ mira letter
```

hello mother! hello father!
here I am at camp mumsade. things are very entertaining,
and they say we'll have some fun when it stops raining.
all the mosquitoes hate the weather, and the lake has
dried up. no summer lemons either! so get
promote something last night after dinner.
how I don't want this to mean you, but my bunk mate has
broke. no summer, no way! they say? they want to
organize a reading party.
take me home, oh mother, father, take me home! I don't want
don't leave me out in the forest when I might get eaten
by a bear! Take me home, I promise that I don't have mosquitoes,
it says the house will have bugs, oh please don't make me
stay -- I'll have some one make my.
dearest father, dearest mother, how's my precious little
brother? I will come home if you like me. I will come.
let Aunt Martha hug and kiss me!
well a minute! it's stopped raining! My are wonderful!
My are wonderful! Playing baseball, get that's better!
Mother, Father, kindly disregard this letter.
Alan Sherman

reduced size to fit on page

Now, either **show letter** or
mira letter will cat out the
letter file

3

```
/home/cis90/simben $ unalias show
/home/cis90/simben $ alias mira
alias view='show'
/home/cis90/simben $ mira letter
-bash: show: command not found
/home/cis90/simben $
```

It can be broken too

single and double quotes (very subtle)

You can control whether bash does filename expansion when you create the alias or ... when the alias is used

```
$ ac=on
$ fan=medium
$ defrost=off
```

double

single

① `$ alias p="echo $ac $fan $defrost"`
`$ alias p`

```
alias p='echo on medium off'
```

`$ alias p='echo $ac $fan $defrost'`
`$ alias p`

```
alias p='echo $ac $fan $defrost'
```

② `$ p`
on medium off

`$ p`
on medium off

③ `$ ac=off`

`$ ac=off`

④ `$ p`
on medium off

`$ p`
off medium off

Note: using single quotes prevents bash from expanding the variables when creating up the alias

Class Exercise

Make some aliases

Make an alias named **showpath** that shows the shell path:

```
alias showpath="echo $PATH"  
showpath
```

Make an alias named **whereonpath** that shows where on the path a command is:

```
alias whereonpath="type -a"  
whereonpath ls  
whereonpath tty  
whereonpath bogus
```

*Paste the output of **whereonpath tty** into the chat window*

bash startup files

bash startup files

Only
executed
when
logging in

/etc/profile (system wide)

- adds root's special path

/etc/profile.d/*.sh (system wide)

- kerberos directories added to path
- adds color, vi aliases
- language, character sets

.bash_profile or **.profile** (user specific)

- set up your path, prompt and other environment variables

.bashrc (user specific)

- add your new aliases here

*Edit these files to
customize your shell
environment*

/etc/bashrc (system wide)

- changes umask to 0002 for regular users
- sets final prompt string



.bash_profile

(Red Hat family)

.profile

(Debian family)

.bash_profile

- The *.bash_profile* is a shell script that sets up a user's shell environment.
- This script is executed each time the user logs in.
- The *.bash_profile* is used for initializing shell variables and running basic commands like `umask` or `set -o` options.
- This script also runs the user's *.bashrc* file

The Debian family uses `.profile` instead of `.bash_profile`

.bash_profile for CIS 90 (runs only at login)

```
[simben@opus ~]$ cat .bash_profile
# .bash_profile
```

```
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc sources the .bashrc file
fi
```

*Appends the
CIS 90 bin,
the user's bin
and the
"current"
directories to
the path*

```
# User specific environment and startup programs
```

```
PATH=$PATH:$HOME/../../bin:$HOME/bin:.
```

```
BASH_ENV=$HOME/.bashrc
```

```
USERNAME=""
```

```
PS1='$PWD $ ' The special prompt used for CIS 90 students is specified
```

```
export USERNAME BASH_ENV PATH variables are exported
```

*umask value
is set*

```
umask 002
```

```
set -o ignoreeof EOF's are ignored
```

```
stty susp ^F Suspend character redefined from Z to F
```

*Terminal type is
requested and
set*

```
eval `tset -s -m vt100:vt100 -m : \?${TERM:-ansi} -r -Q`
```

```
[simben@opus ~]$
```

.bashrc

.bashrc

- The *.bashrc* is a shell script that is executed during user login and whenever a new shell is invoked
- Good place to add user defined aliases

.bashrc

The *.bashrc* is a shell script that is executed during user login and whenever a new shell is invoked. This file usually contains the user defined aliases.

```
[simben@opus ~]$ cat .bashrc
```

```
# .bashrc
```

```
# User specific aliases and functions
```

```
# Source global definitions
```

```
if [ -f /etc/bashrc ]; then
```

```
    . /etc/bashrc
```

sources the /etc/bashrc file

```
fi
```

```
alias print="echo -e"
```

creates a print alias, the -e option enables interpretation of backslash escapes

```
[simben@opus ~]$
```

Class Exercise

Modify .bashrc

Add a new permanent alias to your bash environment

```
alias me="finger $LOGNAME"
```

When finished logout and login again and verify the alias is permanent.



■ and exec

. and exec

In normal execution of a UNIX command, shell-script or binary, the child process is unable to affect the login shell environment.

Sometimes it is desirable to run a shell script that will initialize or change shell variables in the parent environment. To do this, the shell (bash) provides a **.** (dot) or **source** command, which instructs the shell to execute the shell script itself, without spawning a child process to run the script, and then continue on where it left off.

. myscript
source myscript } *equivalent*

In this example, the commands in the file script are run by the parent shell, and therefore, any changes made to the environment will last for the duration of the login session.

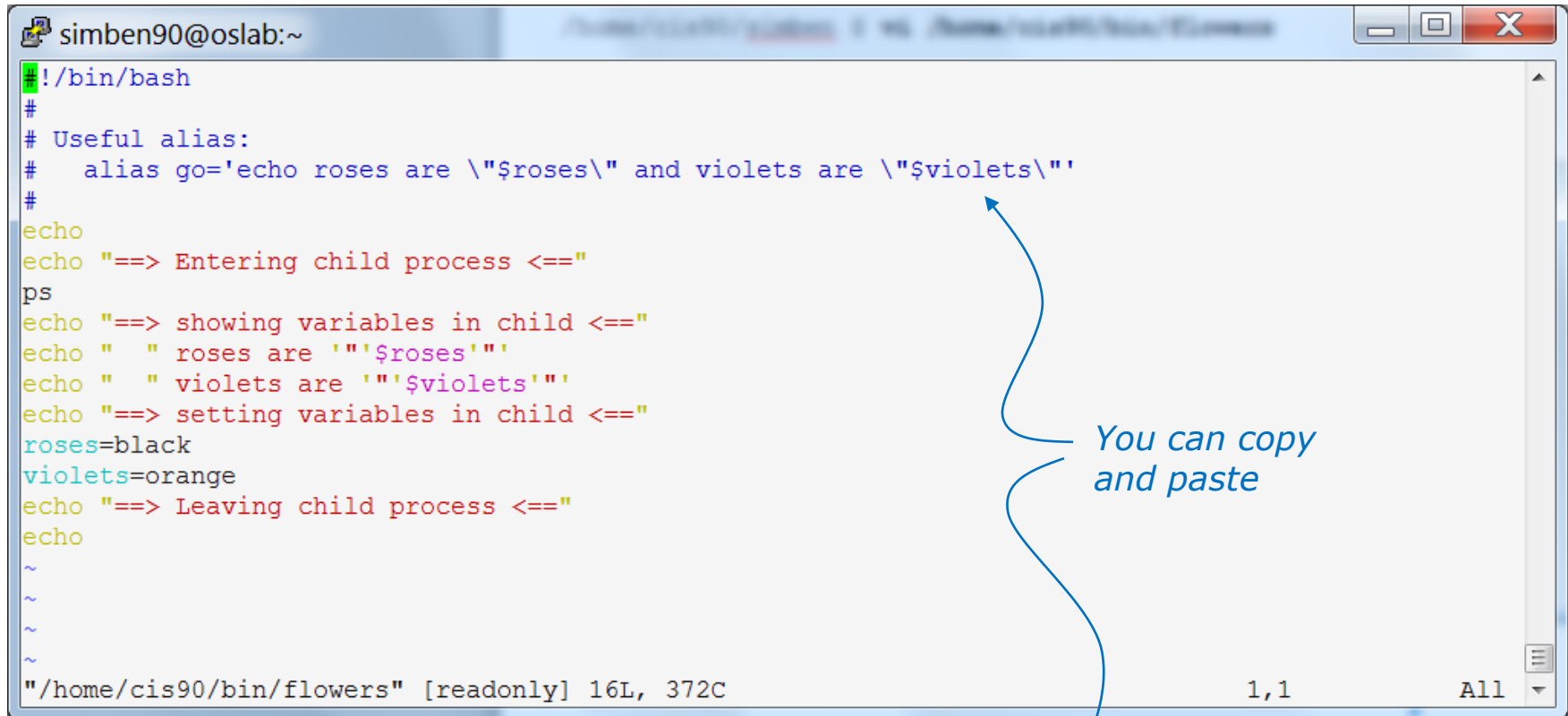
If a UNIX command is run using the **exec** command, the bash code in the process is overlaid by the command code, when finished the process will terminate

exec clear

This will have the effect of clearing the screen and logging off the computer

Grok this lesson?

```
/home/cis90/simben $ vi /home/cis90/bin/flowers
```



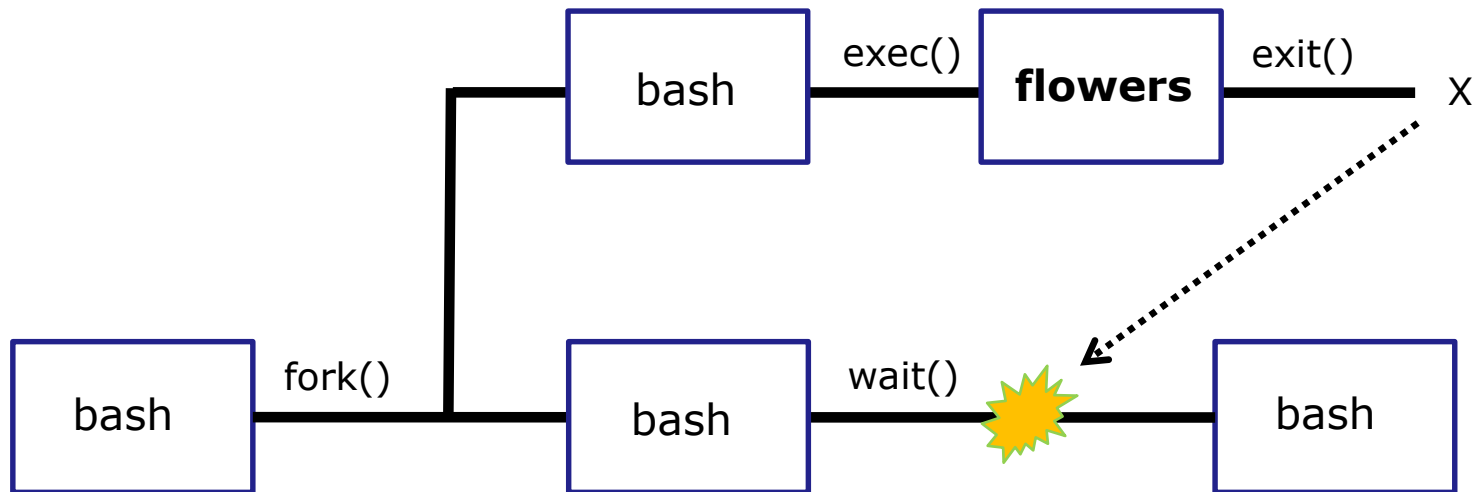
```
simben90@oslab:~
#!/bin/bash
#
# Useful alias:
#   alias go='echo roses are \"$roses\" and violets are \"$violets\"'
#
echo
echo "==> Entering child process <=="
ps
echo "==> showing variables in child <=="
echo "  roses are '$roses'"
echo "  violets are '$violets'"
echo "==> setting variables in child <=="
roses=black
violets=orange
echo "==> Leaving child process <=="
echo
~
~
~
~
"/home/cis90/bin/flowers" [readonly] 16L, 372C 1,1 All
```

You can copy and paste

```
/home/cis90/simben $ alias go='echo roses are \"$roses\" and violets are \"$violets\"'
/home/cis90/simben $ go
roses are "" and violets are ""
```

*The **go** alias is used to show the current values of the roses and violets variables*

running the flowers script



*Use the **flowers** script to test your understanding of how variables are handled with child processes*

As a convenience create an alias to show variable values

Note, the double quotes are escaped. We don't want bash to treat them as special metacharacters. We just want the double quotes preserved so they can be seen in the output of the echo command.

```
/home/cis90/simben $ alias go='echo roses are \"$roses\" and violets are \"$violets\"'
```

```
/home/cis90/simben $ alias go  
alias go='echo roses are \"$roses\" and violets are \"$violets\"'
```

```
/home/cis90/simben $ go  
roses are "" and violets are ""
```

Since there are no shell variables named roses or violets the echo command prints nothing for them.

Create and initialize variables

```
/home/cis90/simben $ go  
roses are "" and violets are ""
```

```
/home/cis90/simben $ roses=red  
/home/cis90/simben $ go  
roses are "red" and violets are ""
```

Now the roses variable has been created and initialized

```
/home/cis90/simben $ violets=blue  
/home/cis90/simben $ go  
roses are "red" and violets are "blue"
```

Now the violets variable has been created and initialized

Unset variables

```
/home/cis90/simben $ unset roses  
/home/cis90/simben $ go  
roses are "" and violets are "blue"
```

Now the roses variable no longer exists

```
/home/cis90/simben $ unset violets  
/home/cis90/simben $ go  
roses are "" and violets are ""
```

Now the violets variable no longer exists

Create and initialize variables again

```
/home/cis90/simben $ roses=red; violets=blue  
/home/cis90/simben $ go  
roses are "red" and violets are "blue"
```

Now both variables have been created and initialized again

Run flowers script as a child process (variables not exported)

```
/home/cis90/simben $ go  
roses are "red" and violets are "blue"
```

*The parent sees roses
and violets*

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

```
==> Entering child process <==
```

PID	TTY	TIME	CMD
28834	pts/0	00:00:00	bash
29447	pts/0	00:00:00	flowers
29454	pts/0	00:00:00	ps

```
==> showing variables in child <==
```

```
roses are ""  
violets are ""
```

*The child does not see
roses or violets*

```
==> setting variables in child <==
```

```
==> Leaving child process <==
```

```
/home/cis90/simben $ go  
roses are "red" and violets are "blue"
```

*The variables are
unchanged after
running flowers script*

Run flowers script as a child process (roses variable exported)

```
/home/cis90/simben $ export roses  
/home/cis90/simben $ go  
roses are "red" and violets are "blue"
```

*The parent sees roses
and violets*

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

```
==> Entering child process <==  
  PID TTY          TIME CMD  
28834 pts/0    00:00:00 bash  
29457 pts/0    00:00:00 flowers  
29464 pts/0    00:00:00 ps  
==> showing variables in child <==  
  roses are "red"  
  violets are ""  
==> setting variables in child <==  
==> Leaving child process <==
```

*The child now sees roses
since it was exported*

```
/home/cis90/simben $ go  
roses are "red" and violets are "blue"
```

*The variables are
unchanged after
running flowers script*

Run flowers script as a child process (script sourced)

```
/home/cis90/simben $ go  
roses are "red" and violets are "blue"
```

*The parent sees roses
and violets*

```
/home/cis90/simben $ source flowers
```

```
==> Entering child process <==  
  PID TTY          TIME CMD  
28834 pts/0        00:00:00 bash  
29469 pts/0        00:00:00 ps  
==> showing variables in child <==  
  roses are "red"  
  violets are "blue"  
==> setting variables in child <==  
==> Leaving child process <==
```

*script is not
running as child*

*The script now sees roses and
violets because it is running in
the parent process*

```
/home/cis90/simben $ go  
roses are "black" and violets are "orange"
```

*The variables are
changed after running
flowers script*

Assignment



Lab 10 - the last one!



You may end up locking yourself out of Opus or seeing other strange things when doing this lab.

I'll be monitoring the forum as usual if anyone needs help.

A full-page background image showing a sunset over a beach. The sky is filled with vibrant orange, pink, and purple clouds. The sun is low on the horizon, casting a warm glow. To the right, a dark, silhouetted cliff rises from the beach. The foreground shows the wet sand of the beach reflecting the colors of the sky, with some dark rocks scattered about.

Wrap up

Extra Credit Special

1) Why did the prompt change?



```
/home/cis90/simben $ bash  
[simben@opus ~]$ exit  
exit  
/home/cis90/simben $
```

2) What command could be issued prior to the bash command above that would prevent the prompt from changing?

*For 2 points extra credit, email risimms@cabrillo.edu answers to **both** questions before the next class starts*

New commands:

.
alias
unalias
set
env
export
exec
source

- source the commands
- create or show an alias
- remove an alias
- show all variables
- show environment variables
- export variable so child can use
- replace with new code
- same as .

New Files and Directories:

.bash_profile
.bashrc

- executed at login
- executed at login and new shells

Next Class

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

Lab 10

Quiz questions for next class:

- How do you make an alias setting permanent?
- What must you do to a variable so a child can use it?
- How would you use an alias to make a command named copy ... that would do what the cp command does?

Backup

vi and /bin/mail (review)

Best Practice - /bin/mail and vi

```
/home/cis90/simben $ mail rodduk90
```

```
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/simben $ mail rodduk90
```

```
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

CIS 90 - Lesson 12

/bin/mail and vi



The screenshot shows a terminal window titled "simmsben@opus:~". Inside the terminal, the vi editor is open, displaying an email message. The message text is as follows:

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

Below the message, there are several tilde (~) characters representing empty lines. At the bottom of the editor, the status line shows the file path and cursor position: `"/tmp/RecVQYE2" 7L, 141C`. The terminal window has standard Linux window controls (minimize, maximize, close) in the top right corner.

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

/bin/mail and vi

```
/home/cis90/simben $ mail rodduk90
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/simben $
```

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

/bin/mail and vi

```
/home/cis90/rodduk $ mail
```

```
Mail version 8.1 6/6/93.  Type ? for help.
```

```
"/var/spool/mail/rodduk90": 1 message 1 unread
```

```
>U  1 simben90@opus.cabrill  Mon Nov 10 20:25  22/782  "Good bones"  
& 1
```

```
Message 1:
```

```
From simben90@opus.cabrillo.edu  Mon Nov 10 20:25:32 2008
```

```
Date: Mon, 10 Nov 2008 20:25:32 -0800
```

```
From: Benji Simms <simben90@opus.cabrillo.edu>
```

```
To: rodduk90@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!*

```
&
```

Activity

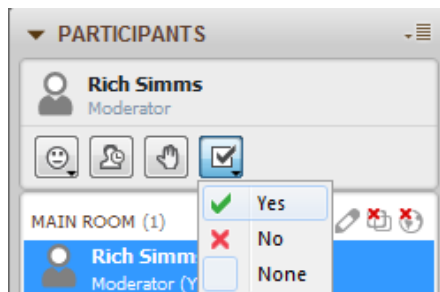
Try it!

Use /bin/mail and send yourself a message:

mail \$LOGNAME

Type a few lines into the message then use the **~v** command to correct or change them.

Read the email you sent yourself to see if your changes worked.



Did it work?

Start this activity by putting a **red x** in CCC Confer.

If you get it to work correctly change your **red x** to a **green checkmark**