Lesson Module Checklist
- Slides
- WB

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

- at jobs: lab 9 turnin dir locked
- Lab 10 and Final Project uploaded
- allscripts updated
- myscript in depot
- flowers and riddle in bin
- sample myscripts for Benji and Homer

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

# 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/)

# CIS 90 - Lesson 12

Instructor: **Rich Simms**
Dial-in: **888-886-3951**
Passcode: **136690**

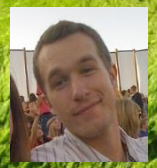

Francisco

Leila

Justin

Jesus


Shenghong


Paul

Roberto


Sam


Navin


Jimmy

Luis

Tommy


Adrian


Ann


Cameron

Cody

Alejandrino


Deane

Nadia


Richard Z.

Gabriel


Ryan


Takashi


Jeff


Nick


Jonathan

Shea

Dylan

Joshua

Richard I.


Aaron


Nicole

James

Matthew

Abraham

Chris

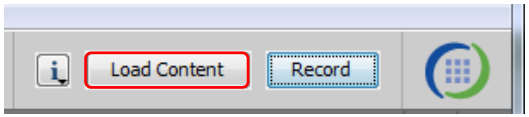
Ronald

Scott

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

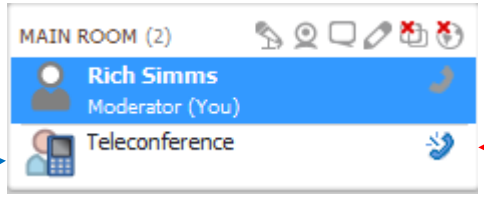
# Instructor CCC Confer checklist

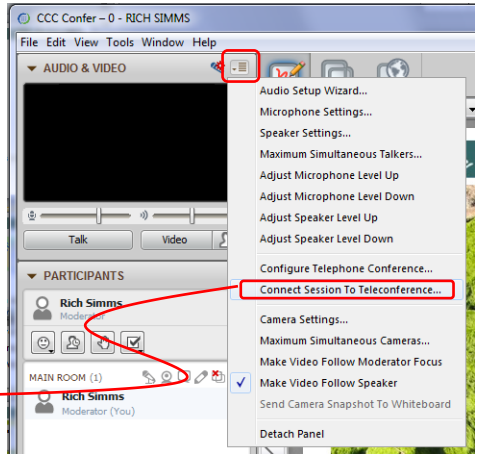
[ ] Preload White Board



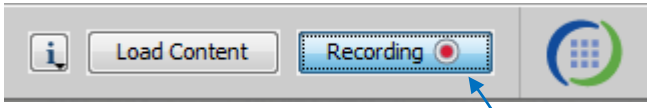

[ ] Connect session to Teleconference

*Session now connected to teleconference*



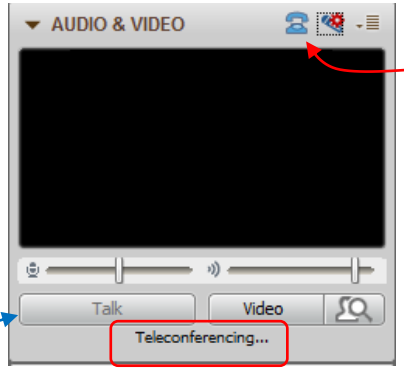

[ ] Is recording on?




*Red dot means recording*

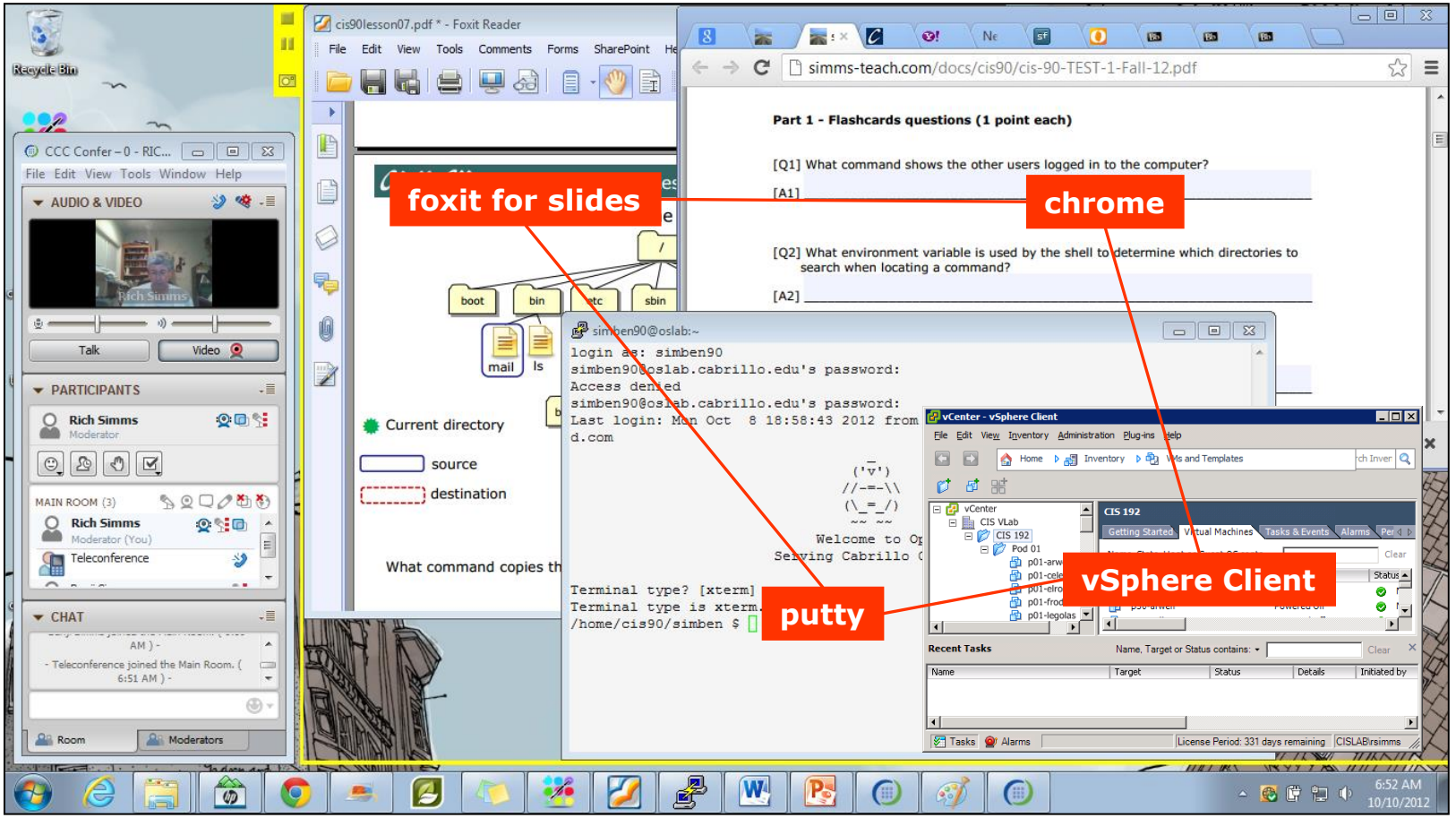
[ ] Use teleconferencing, not mic

*Should be greyed out*



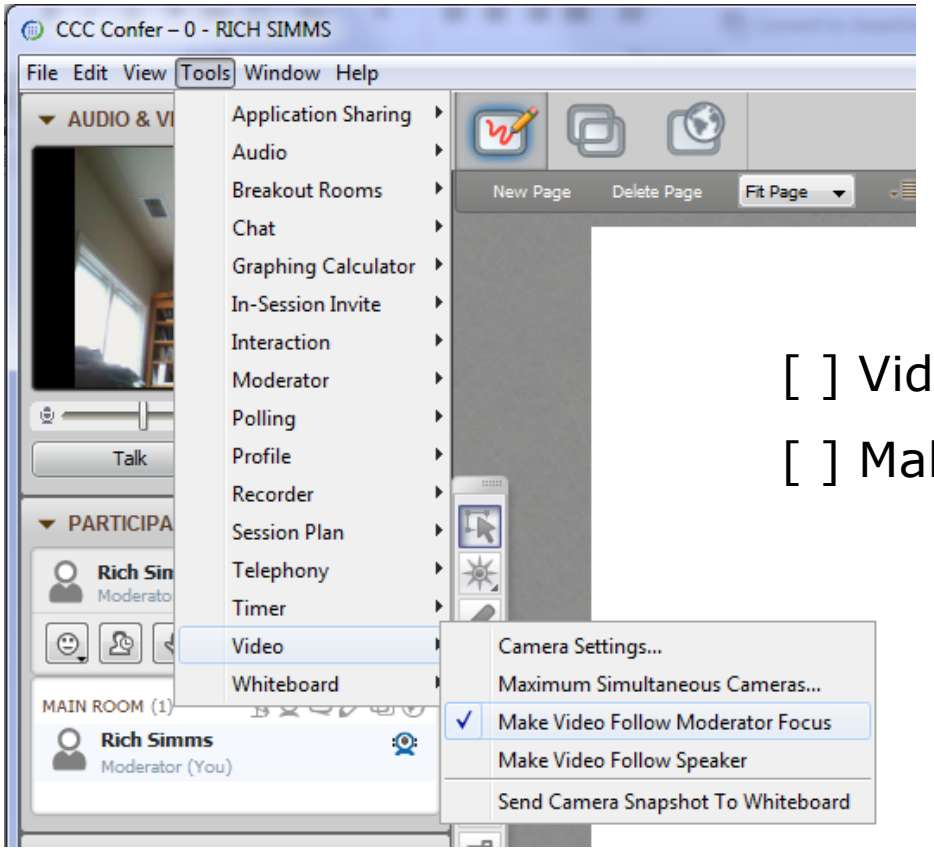

*Should show as this live "off hook" telephone handset icon and the Teleconferencing … message displayed*

# Instructor CCC Confer checklist




[ ] layout and share apps 

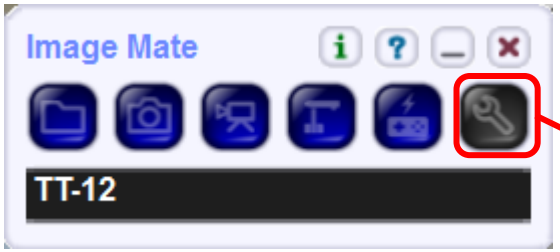
# Instructor CCC Confer checklist

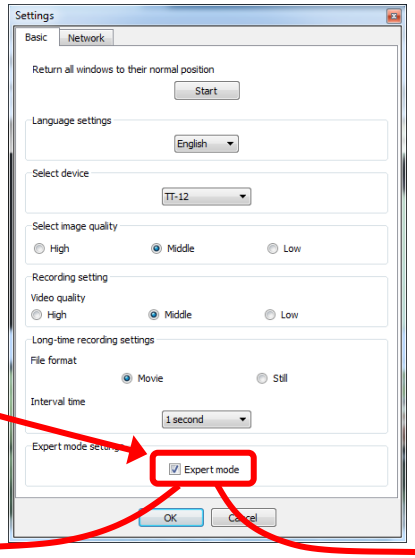



[ ] Video (webcam)

[ ] Make Video Follow Moderator Focus

# Using Elmo with CCC Confer



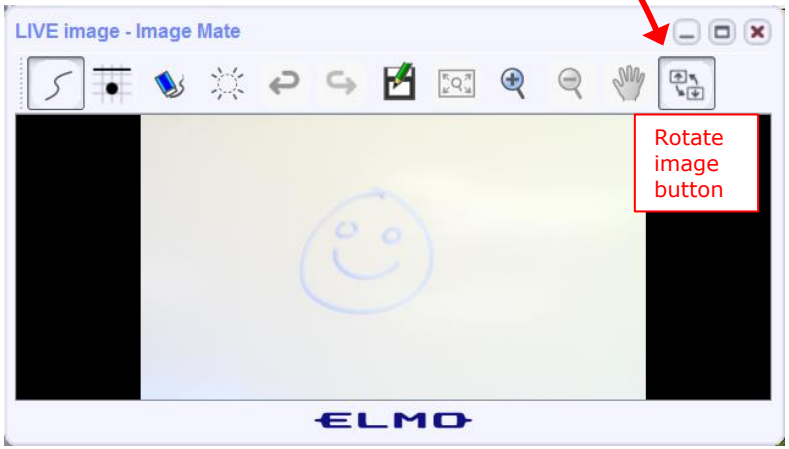




*The "rotate image" button is necessary if you use both the side table and the white board.*

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

Elmo rotated down to view side table


Elmo rotated up to view white board

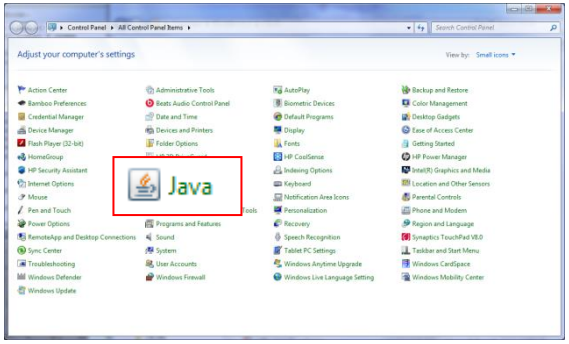



*Run and share the Image Mate program just as you would any other app with CCC Confer*

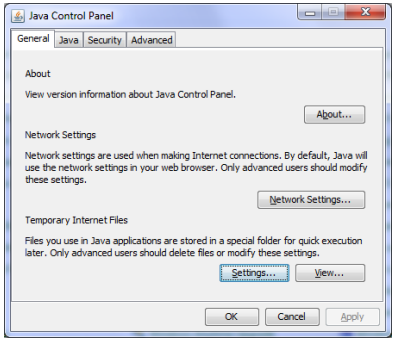
# Instructor CCC Confer checklist

Universal Fix for CCC Confer:
1) Shrink (500 MB) and delete Java cache
2) Uninstall and reinstall latest Java runtime

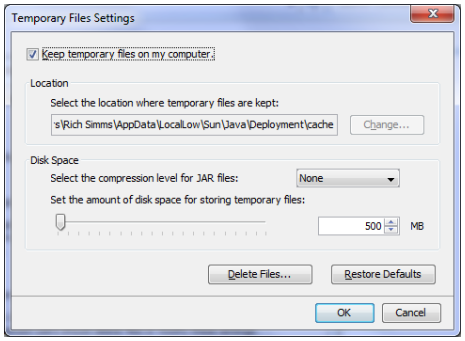
Control Panel (small icons)



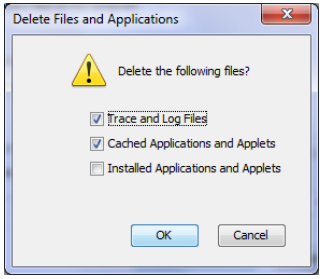

General Tab > Settings…




500MB cache size




Delete these




Google Java download

# Quiz

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

**See electronic white board**

**email answers to: risimms@cabrillo.edu**

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

# The Shell Environment

| Objectives | Agenda |
| --- | --- |
| • Be able to set, view and unset shell variables<br>• Describe the difference between the set and env commands<br>• Explain the importance of the export command.<br>• Describe three actions that are handled by the .bash_profile file<br>• Define user-defined aliases<br>• Explain the . (dot) command and the exec command. | • Quiz<br>• Housekeeping<br>• Spell checking<br>• Review pathnames<br>• Final project prep<br>• Variables<br>• The shell environment<br>• Aliases<br>• .bash_profile<br>• .bashrc |

# 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。<br>*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*

# Esc :!q vs Esc :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*

*Editing vocab in one login session*

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
@
```

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

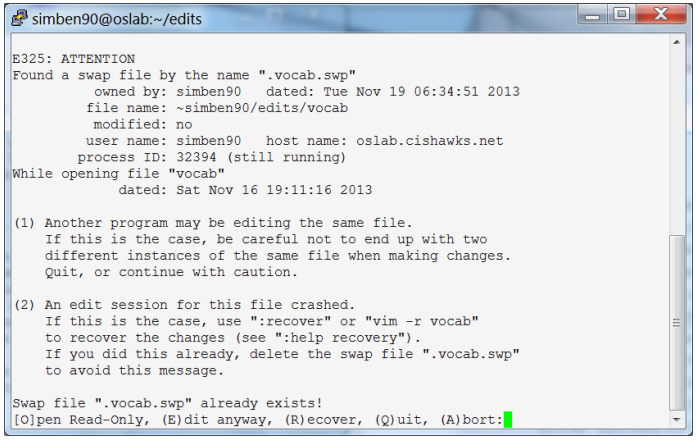
simben90@oslab:~/edits

```
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:
```

*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.

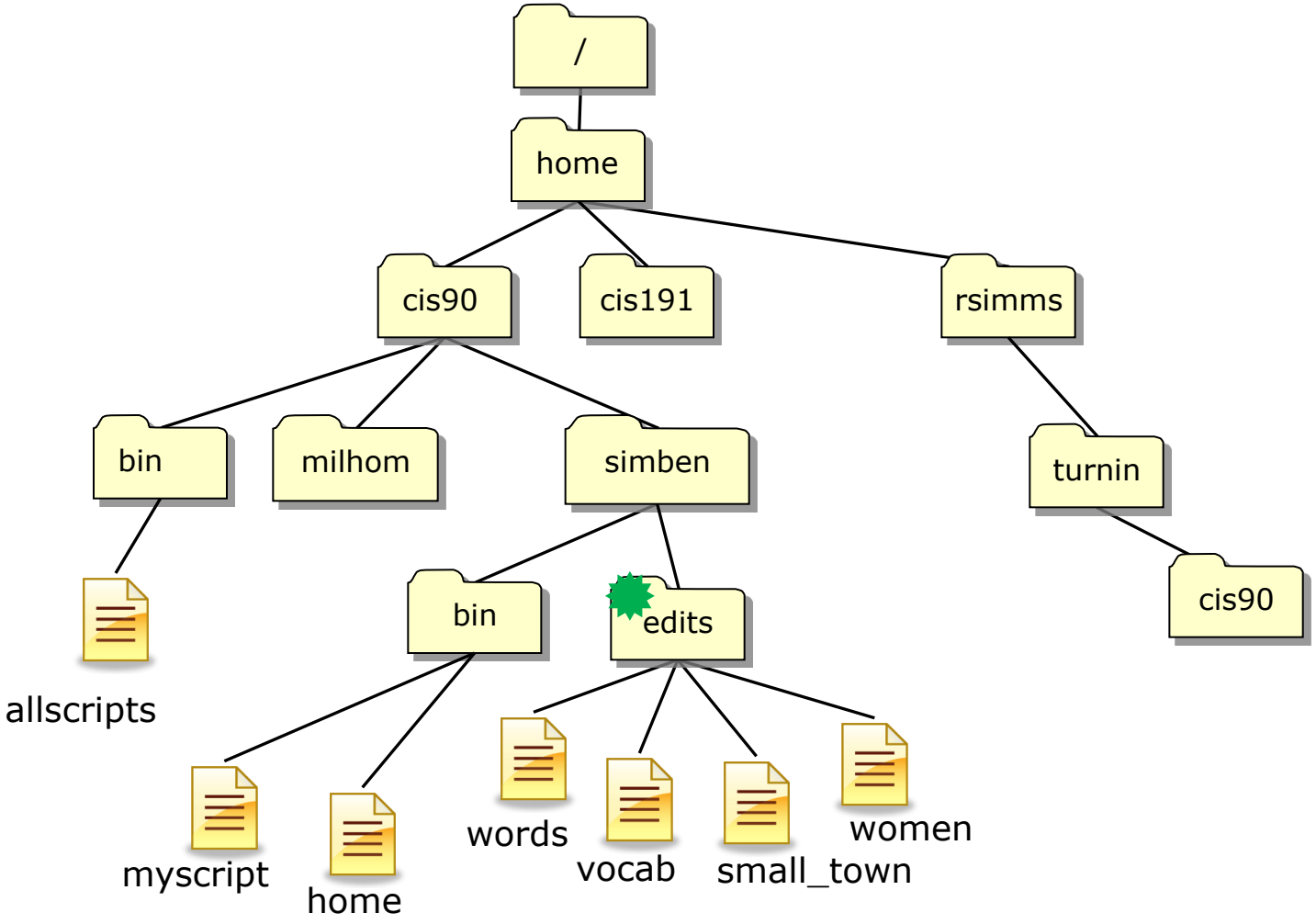



If you get this ATTENTION message it means the temporary .swp file still exists. You may be editing the same file in another session or your original editing session was disconnected before finishing. To get rid of this message you need to remove the .swp file.

# Submitting Lab 9 & PATHNAMES!

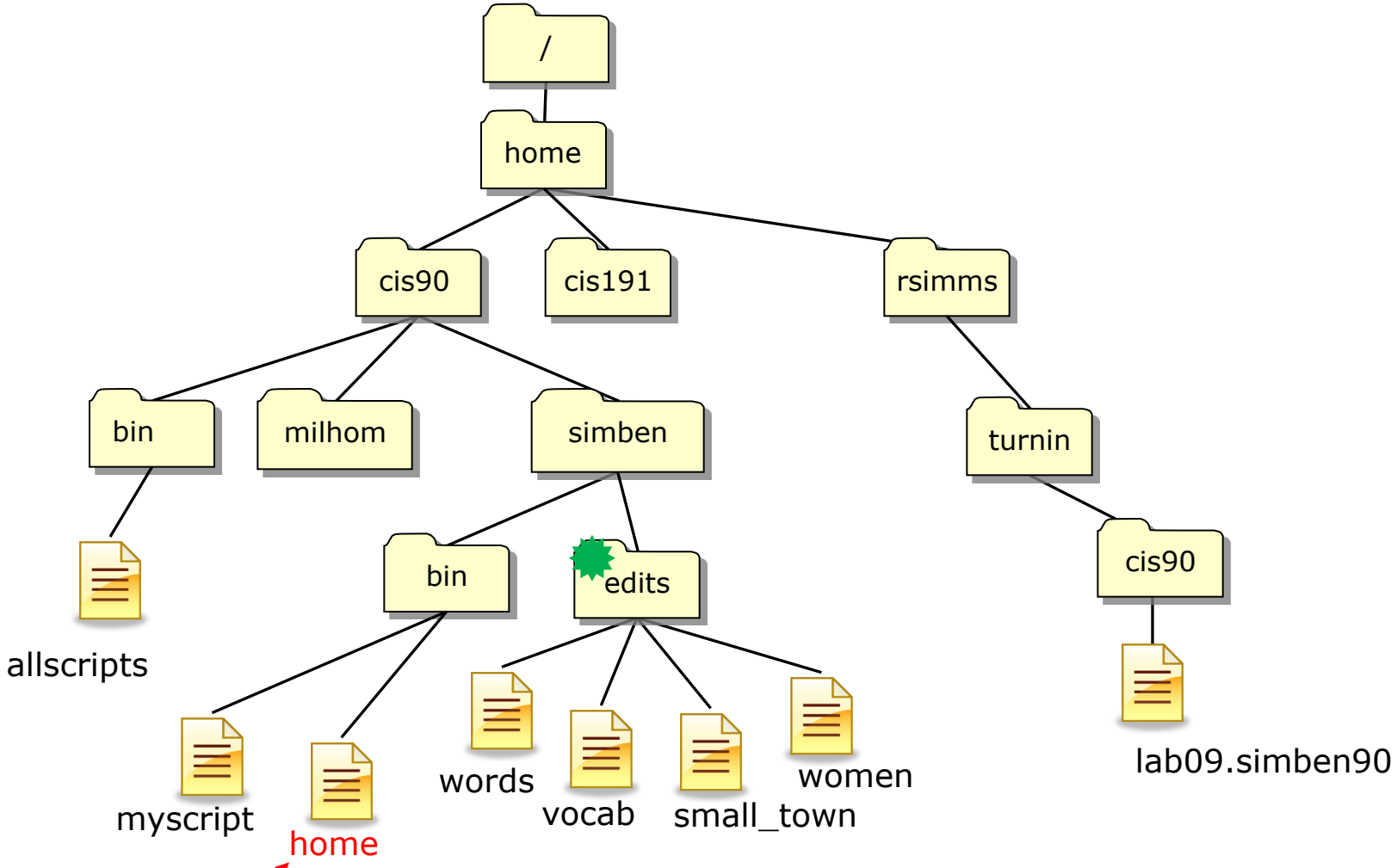
## REMINDER

- 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.



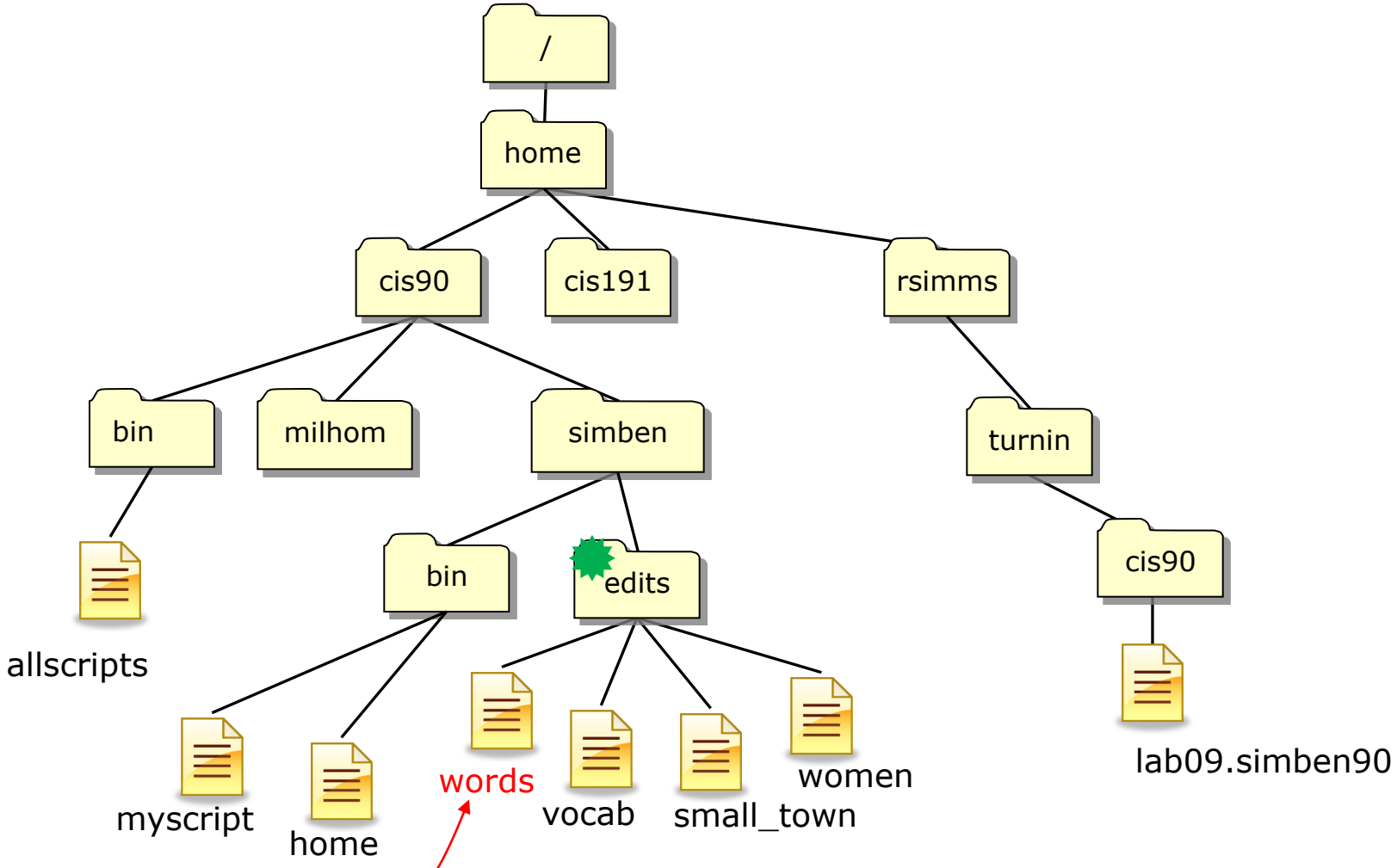

```
/home/cis90/simben/edits $ cat home words vocab small_town women > /home/rsimms/turnin/cis90/lab09.$LOGNAME
cat: home: No such file or directory
```

*Why does this command get an error message?*



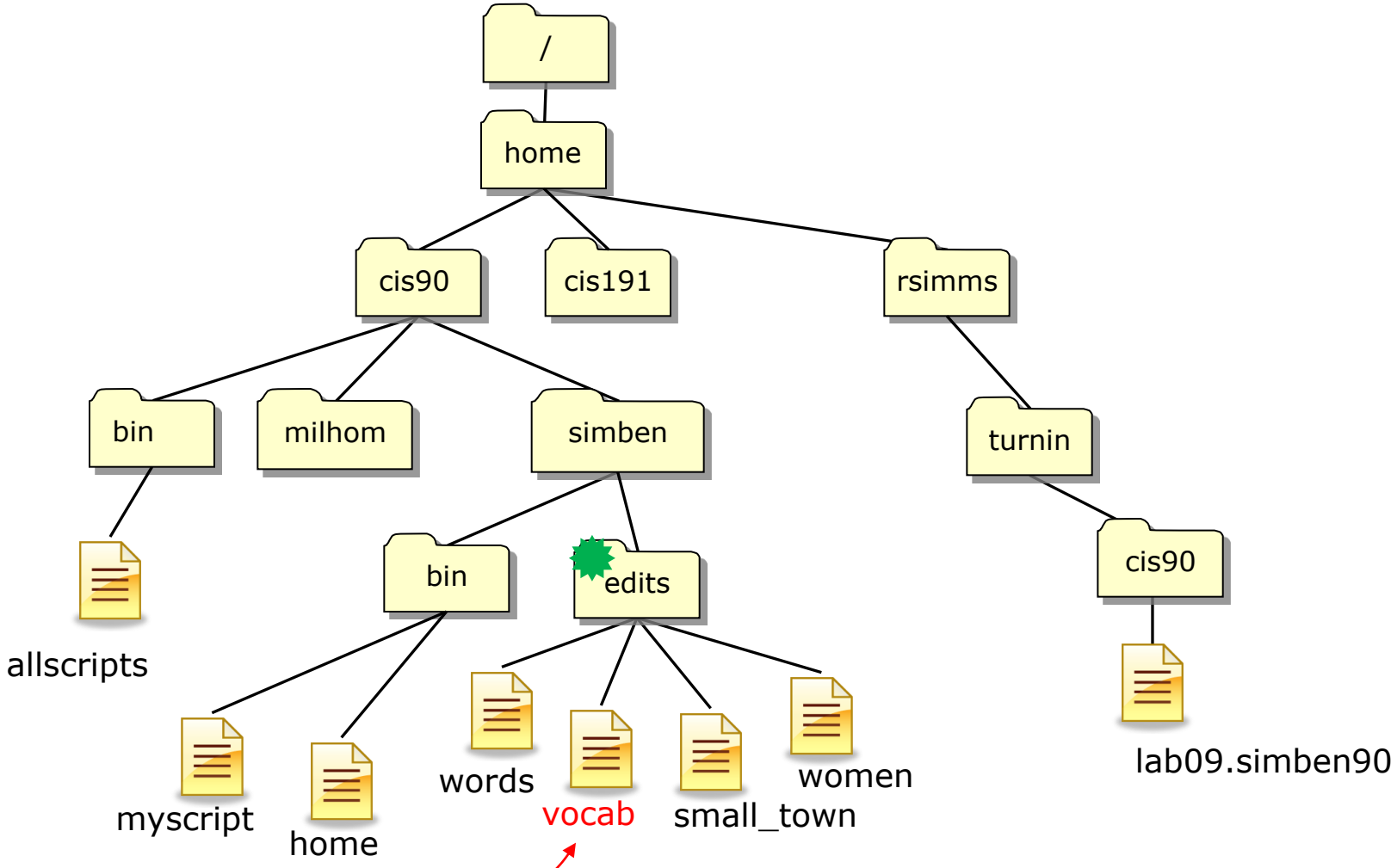

```
cat ../bin/home words vocab small_town women > /home/rsimms/turnin/cis90/lab09.$LOGNAME
```

*relative pathname to home in the bin directory*



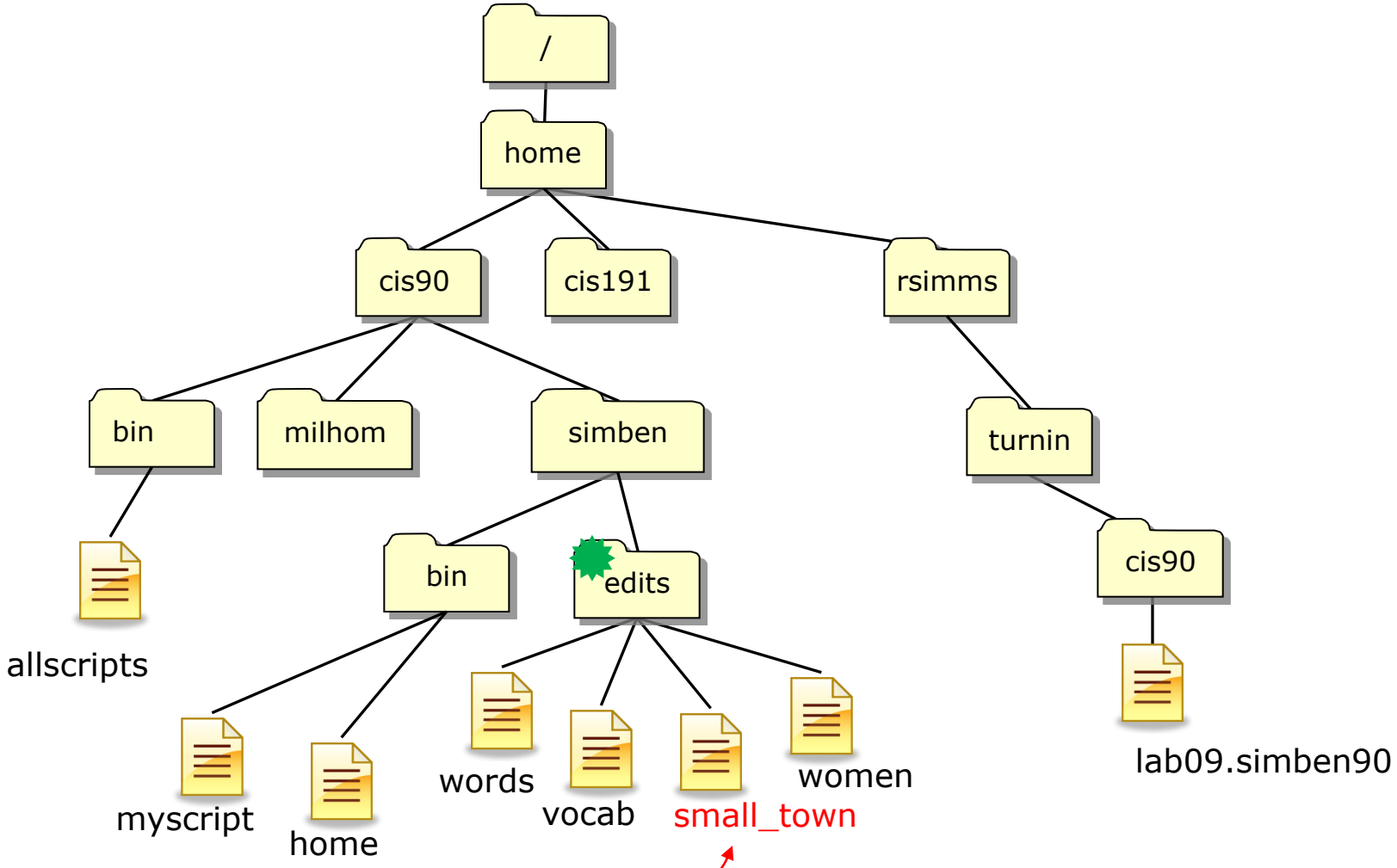

```
cat ../bin/home words vocab small_town women > /home/rsimms/turnin/cis90/lab09.$LOGNAME
```

*relative pathname*



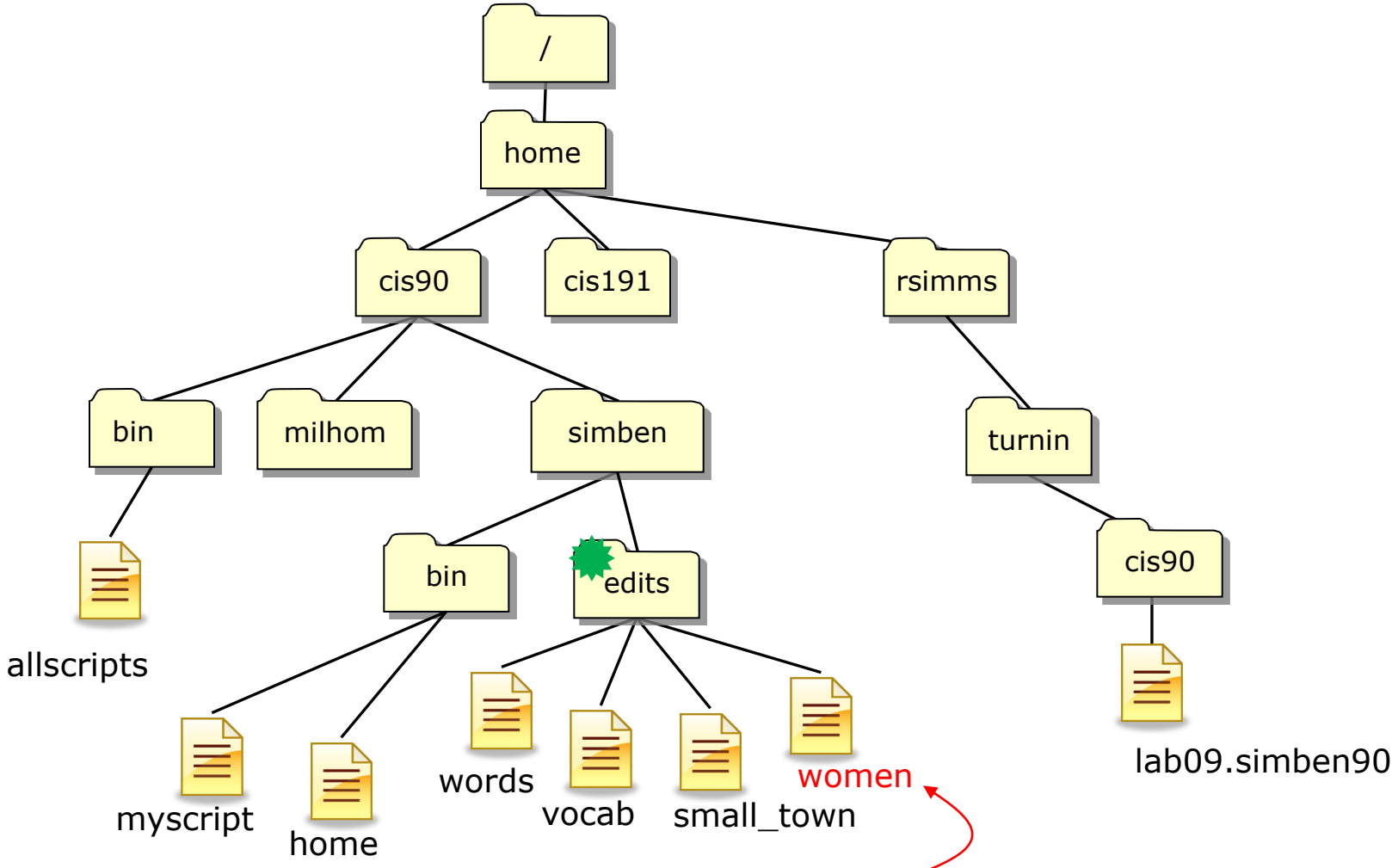

```
cat ../bin/home words vocab small_town women > /home/rsimms/turnin/cis90/lab09.$LOGNAME
```

*relative pathname*



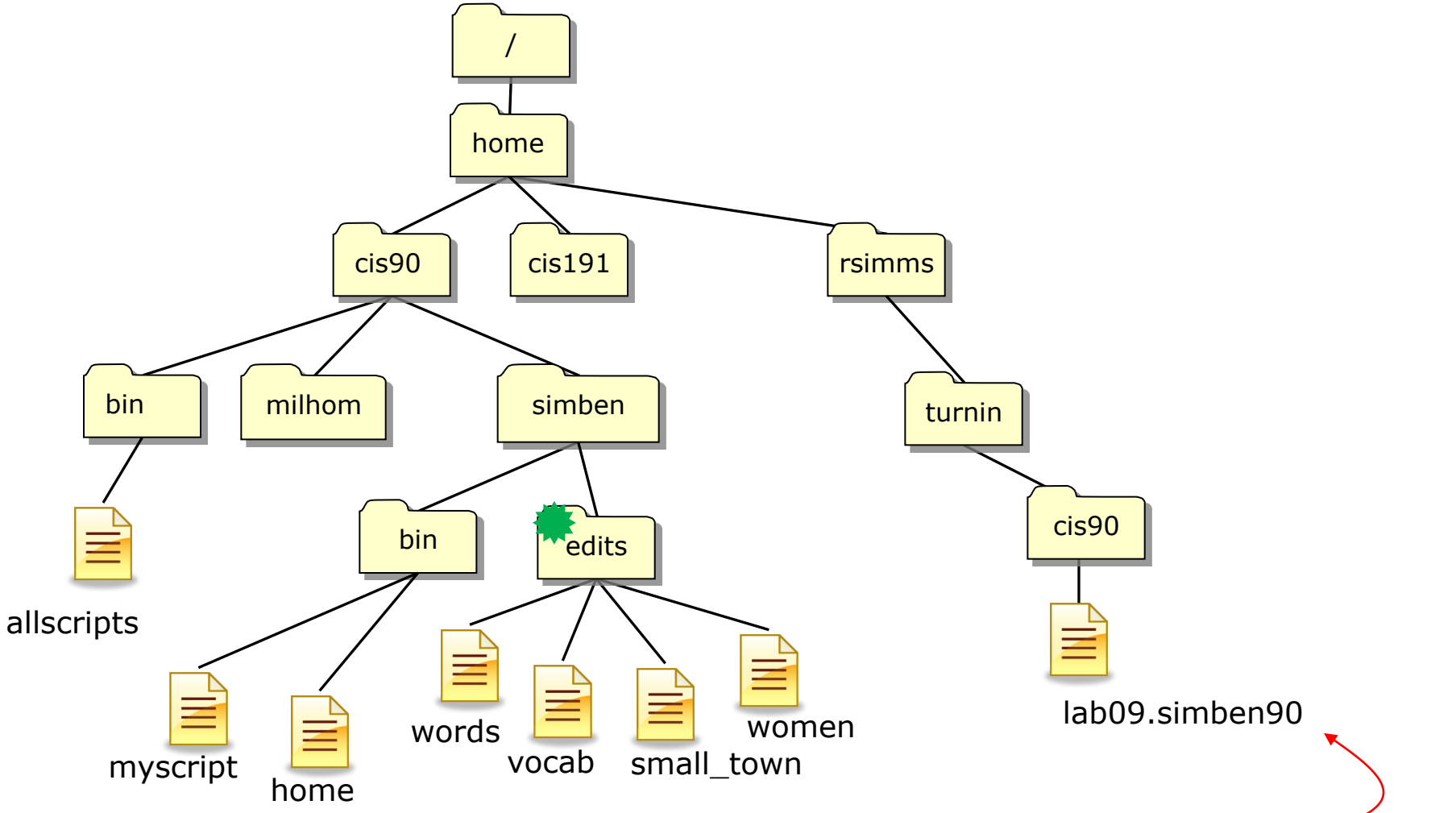

```
cat ../bin/home words vocab small_town women > /home/rsimms/turnin/cis90/lab09.$LOGNAME
```

*relative pathname*



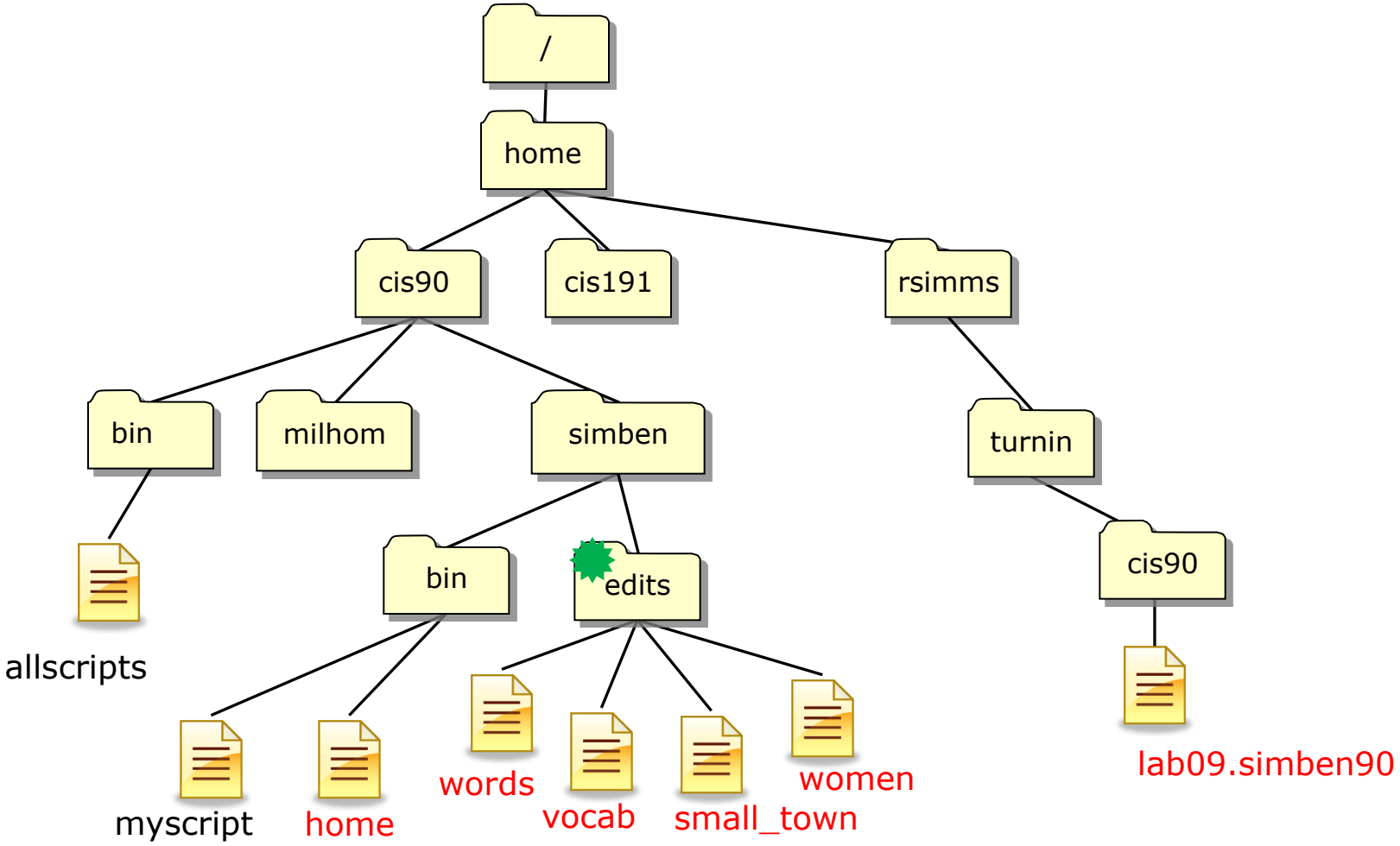

```
cat ../bin/home words vocab small_town women > /home/rsimms/turnin/cis90/lab09.$LOGNAME
```

*relative pathname*




```
cat ../bin/home words vocab small_town women > /home/rsimms/turnin/cis90/lab09.$LOGNAME
```

*absolute pathname*




*A much better way to do this:*

```
cat ../bin/home words vocab small_town women > lab09
less lab09
cp lab09 /home/rsimms/turnin/cis90/lab09.$LOGNAME
```

*Lets you review your work so you know what you are turning in*

# 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
No manual entry for spell
```

*Hmmm. No man page for spell - weird!*

```
/home/cis90/simben $ type spell
spell is hashed (/usr/bin/spell)
```

*Where is it on our path?*

```
/home/cis90/simben $ file /usr/bin/spell
/usr/bin/spell: Bourne shell script text executable
```

*So what kind of file is it?*

```
/home/cis90/simben $ cat /usr/bin/spell
#!/bin/sh

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

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

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

*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?

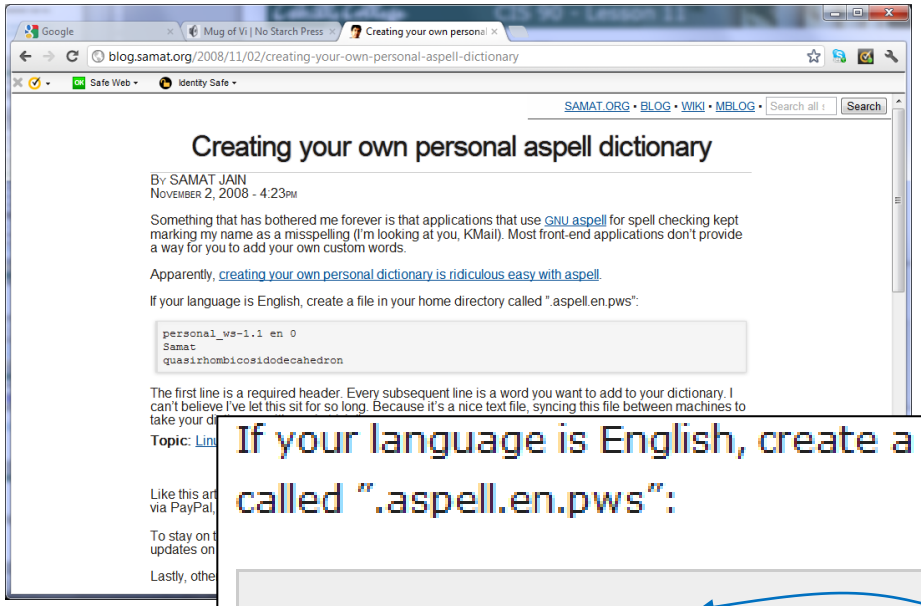
```
ASPELL(1)              Aspell Abbreviated User's Manual              ASPELL(1)

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*

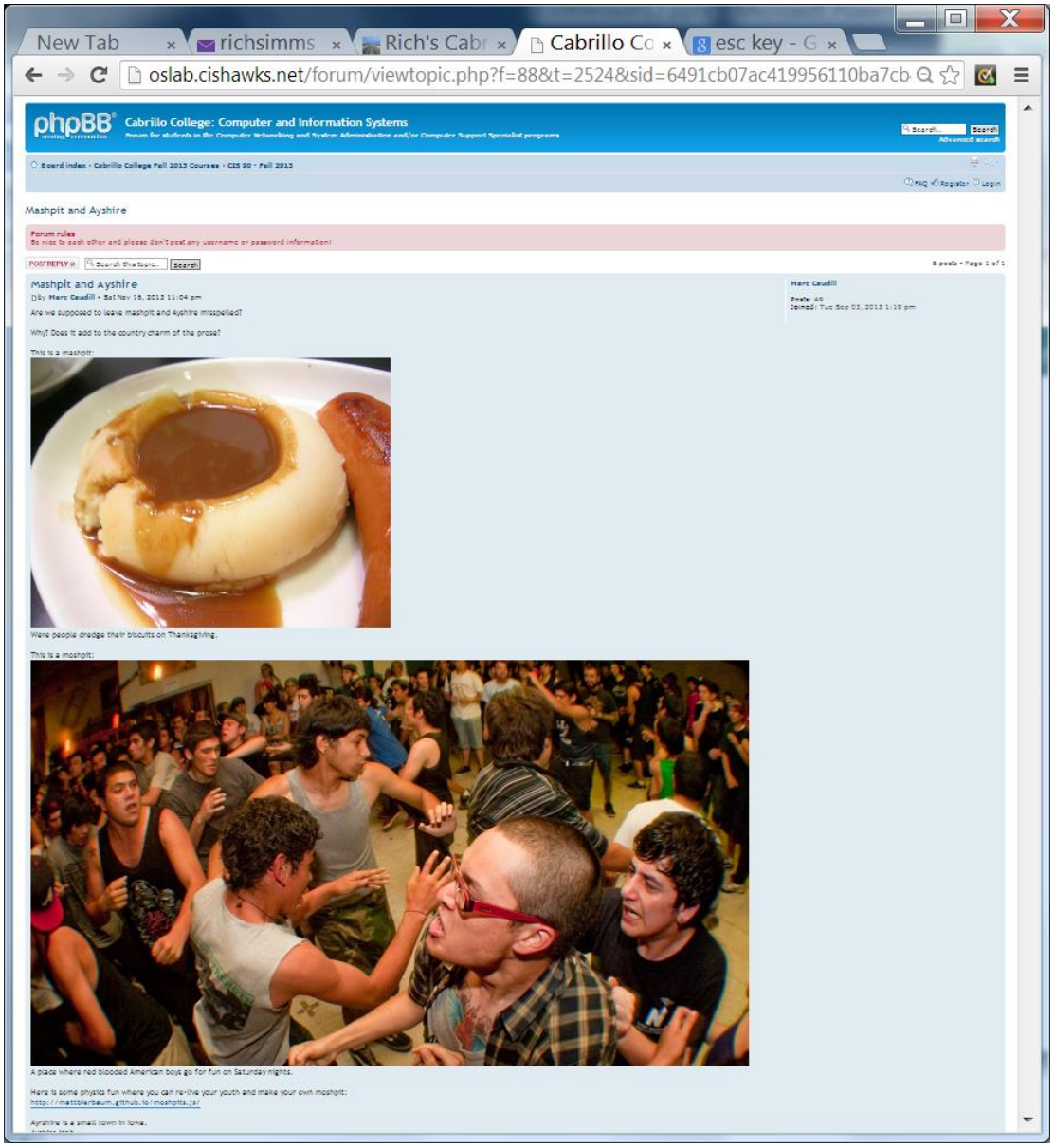
*This is FYI and not required for Lab 9*

```
/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
richsimms
Rich's Cabr
Cabrillo Co
esc key - G
oslab.cishawks.net/forum/viewtopic.php?f=88&t=2524&sid=6491cb07ac419956110ba7cb
phpBB
Cabrillo College: Computer and Information Systems
Mashpit and Ayshire
Mashpit and Ayshire
Are we supposed to leave mashpit and Ayshire misspelled?
Why! Does it add to the country charm of the prose?
This is a mashpit:
Were people dredge their biscuits on Thanksgiving.
This is a mashpit:
A place where red blooded American boys go for fun on Saturday nights.
Here is some physics fun where you can re-live your youth and make your own mashpit:
http://mattblenbaum.github.io/mashpits.js/
Ayshire is a small town in Iowa.

*moshpit?*

*Ayshire?*

urban DICTIONARY

**1. moshpit**

a place at a gig where you can dance with however the f*** 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.

bleeped

Merriam-Webster
m-w.com

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**

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, Samual 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]

# 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 the two words Ayshire and moshpit (or mashpit) in the file words 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
```

# WHY?

*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/*

**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

# Previous material and assignment

1. Lab 9 due 11:59PM tonight

2. Five posts due 11:59PM 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.*

# Final Exam

Test #3 (final exam)

| | | | | |
|---|---|---|---|---|
| | 12/18 | **Test #3 (the final exam)**<br>**Time**<br>• 1:00PM - 3:50PM in Room 828<br>**Materials**<br>• Test (blackboard)<br>**CCC Confer**<br>• Enter virtual classroom<br>• Class archives | | 5 posts<br>Lab X1<br>Lab X2 |

- All students will take the test at the same time.
- Working students will need to plan ahead to take time off from work for the test.

## resource How To

POSTREPLY | Search this topic... Search | 2 posts • Page 1 of 1

### resource How To

EDIT QUOTE

by **Ann Pike** » Mon Nov 17, 2014 10:29 am

I updated and Rich posted the How To on How to log into Opus From a Mac.
Here is the link:
http://simms-teach.com/howtos/students/ ... e-2014.pdf

I found the how to helpful, but not complete when I found it. If you have a Mac and have not been accessing Opus from it, you can try using this to do so. I documented the steps I needed to use and updated all the photos and text and hopefully now it is easy to use and know how to do the log in.

One advantage in using a Mac is, once you have set up the New Remote Connection, be it opus or any server, you have that to just click on. You will have to add your user name each time and also most important of all, add the port if it is different from the normal as is the case with our opus family. It is however an easy add!

**Ann Pike**

Posts: 59
Joined: Tue Sep 02, 2014 7:48 pm

PM

## Nov 21 Technology Career Meetup

POSTREPLY Search this topic... Search 1 post • Page 1 of 1

### Nov 21 Technology Career Meetup

EDIT QUOTE

by **Rich Simms** » Mon Nov 10, 2014 1:10 pm

This is a good way to meet some of the other CS/CIS instructors and discover various careers in technology.

- Rich

Computer Information Systems & Computer Science
Technology Career Meetup

When: November 21, 2014 @12:00 noon
Where: Cabrillo College Room 828

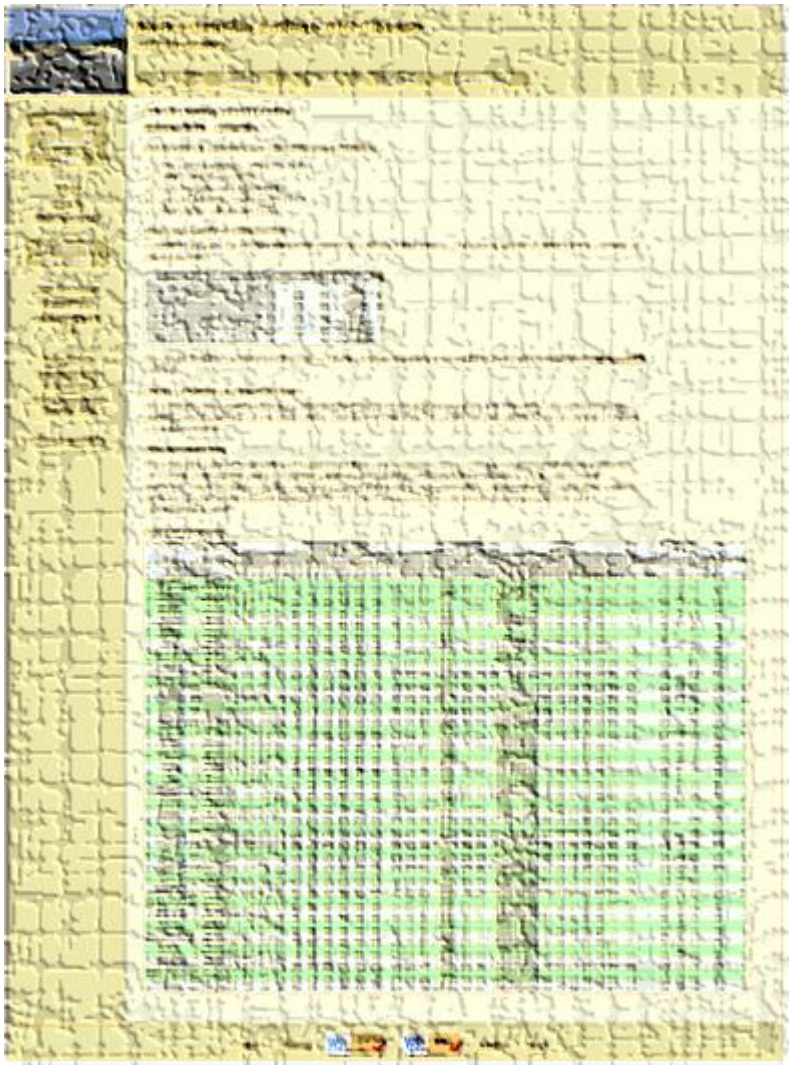
Learn more about careers in Computer Science (CS) and Computer and Information Systems (CIS) and about classes for Spring 2015. Meet CS and CIS Faculty


**Rich Simms**

Posts: 1549
Joined: Sat Jan 16, 2010 5:47 pm

PM

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



# GRADES

- Check your progress on the Grades page
- If you haven't already, send me a student survey to get your LOR secret code name
- Graded labs, tests and forum posts are placed in your home directories on Opus
- Answers to labs, tests and quizzes are in the */home/cis90/answers* directory on Opus

# Current Point Tally

As of 11/17/2014

| **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** |

| 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 |

**This Saturday 11/22/2014 - Last Withdraw Date**

# Jesse's checkgrades python script

http://oslab.cabrillo.edu/forum/viewtopic.php?f=31&t=773&p=2966

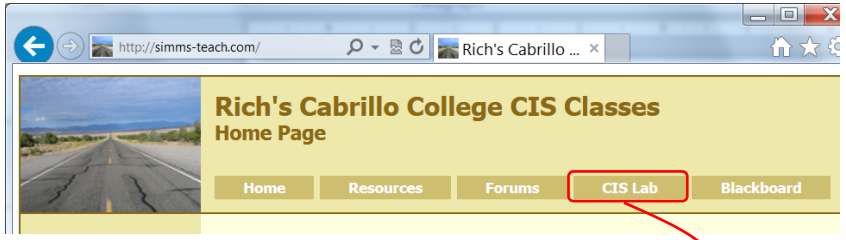
```
/home/cis90/simben $ checkgrades smeagol

Remember, your points may be zero simply because the
assignment has not been graded yet.

Quiz 1: You earned 3 points out of a possible 3.
Quiz 2: You earned 3 points out of a possible 3.
Quiz 3: You earned 3 points out of a possible 3.
Quiz 4: You earned 3 points out of a possible 3.


Forum Post 1: You earned 20 points out of a possible 20.

Lab 1: You earned 30 points out of a possible 30.
Lab 2: You earned 30 points out of a possible 30.
Lab 3: You earned 30 points out of a possible 30.
Lab 4: You earned 29 points out of a possible 30.


You've earned 15 points of extra credit.

You currently have a 109% grade in this class. (166 out of
152 possible points.)
```

*Use your LOR code name as an argument on the checkgrades command*

*Jesse is a CIS 90 Alumnus. He wrote this python script when taking the course. It mines data from the website to check how many of the available points have been earned so far.*

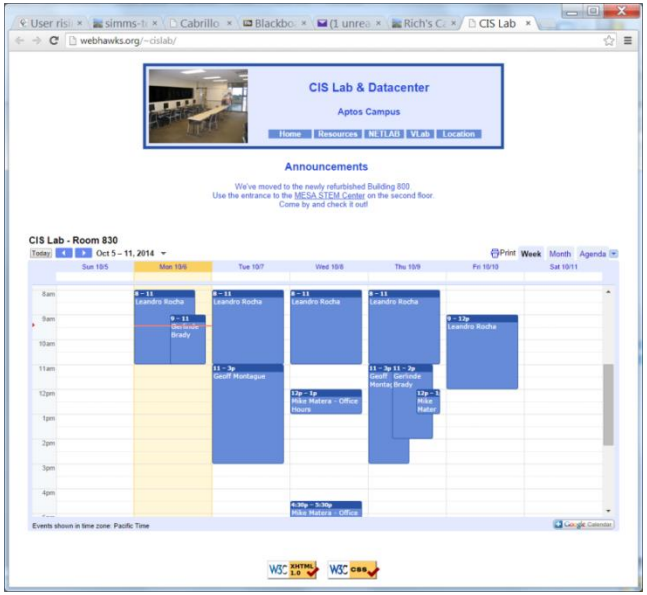
# CIS Lab Schedule

http://webhawks.org/~cislab/




*Not submitting tests or lab work?*

*If you would like some additional help come over to the CIS Lab.*




*Leandro, Geoff and Nick (starting soon) are all CIS 90 Alumni.*

*Michael is the other Linux instructor.*

Or hang around after class. Rich has his office hours right after each class in Room 828.

# Free CIS 90 Tutoring Available

http://www.cabrillo.edu/services/tutorials/

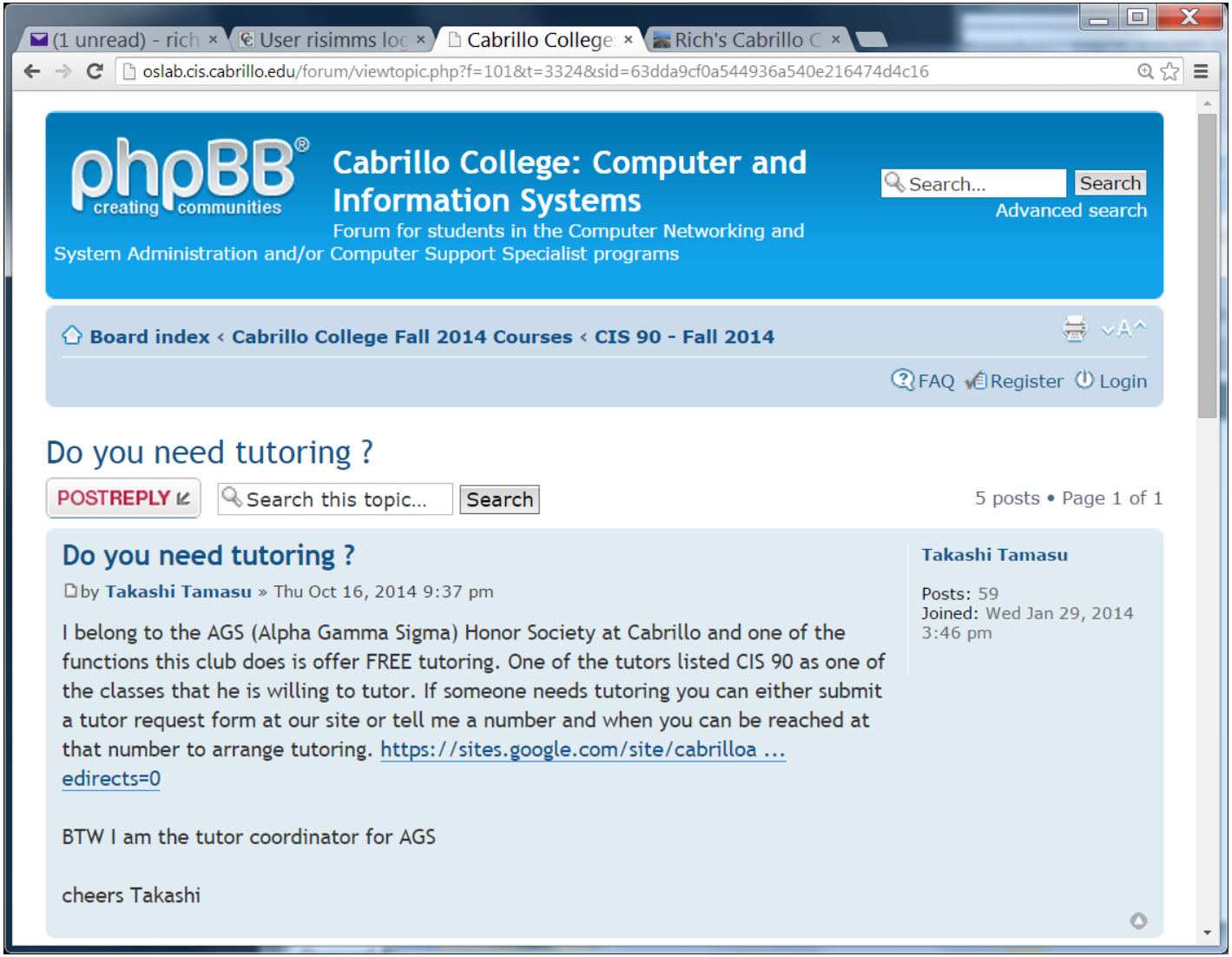

Matt Smithey

All students interested in tutoring in CIS 90, 172, and 81 classes need to come directly to the Tutorials Center to schedule, register and fill out some paperwork. This is just a one-time visit.

The tutoring will take place at the STEM center and they will log in and log out on a computer you have designated (I will figure out exactly what that means).

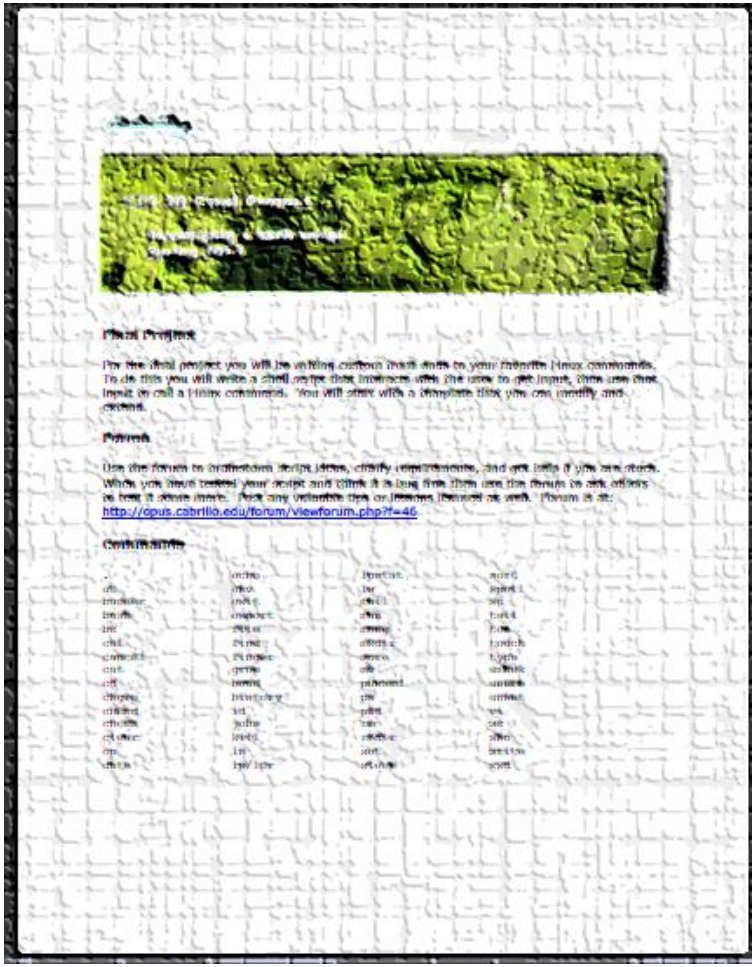
Matt is available M: 9:00-5:00, T: 9-11 and 2-5, Wed: 9-12 and Th: 9-11 and 3-5.

# More CIS 90 Tutoring Available



# final project *preview*

# Final Project

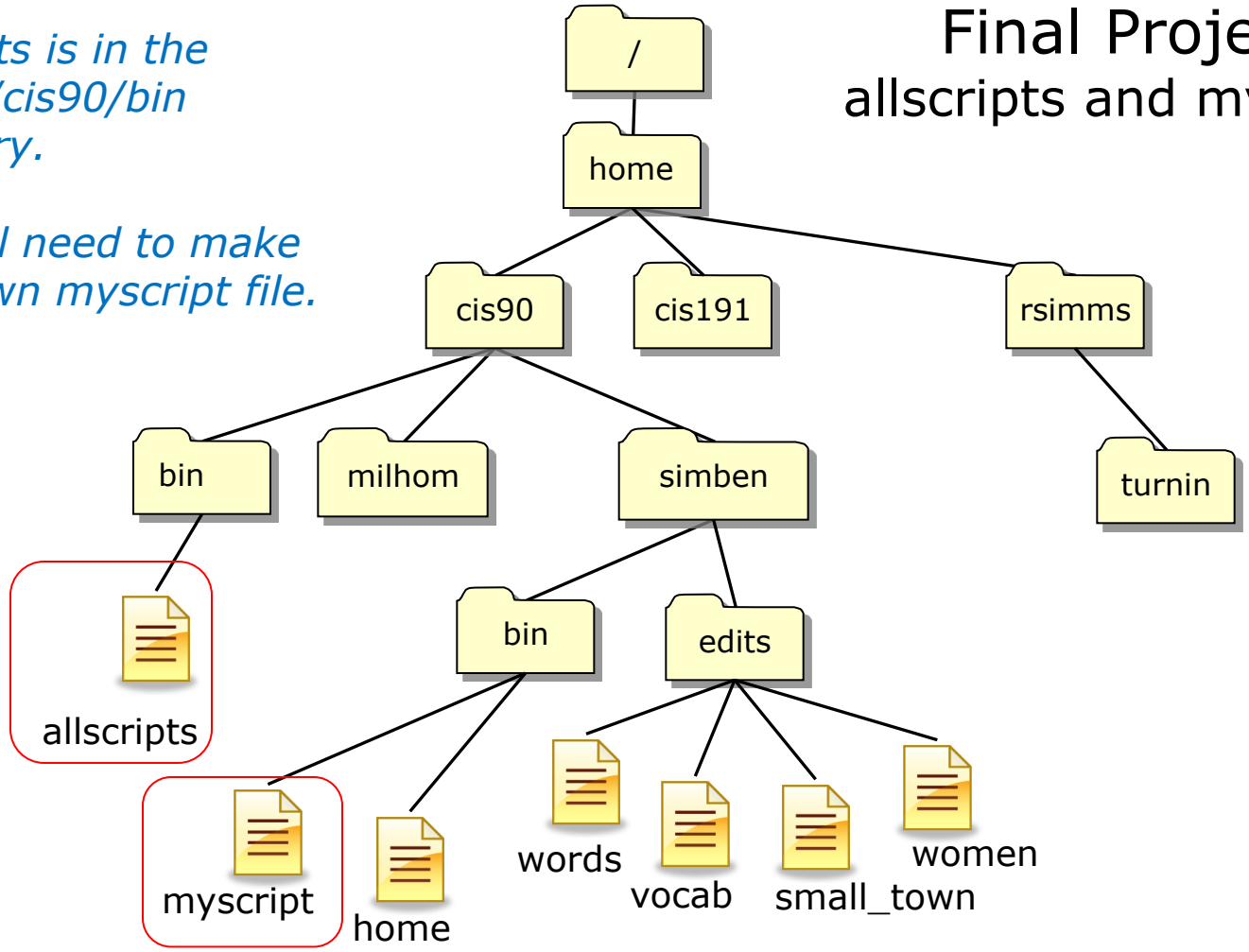


*You now have the necessary skills to begin the final project!*

# Final Project
## allscripts and myscript

*allscripts is in the /home/cis90/bin directory.*

*You will need to make your own myscript file.*



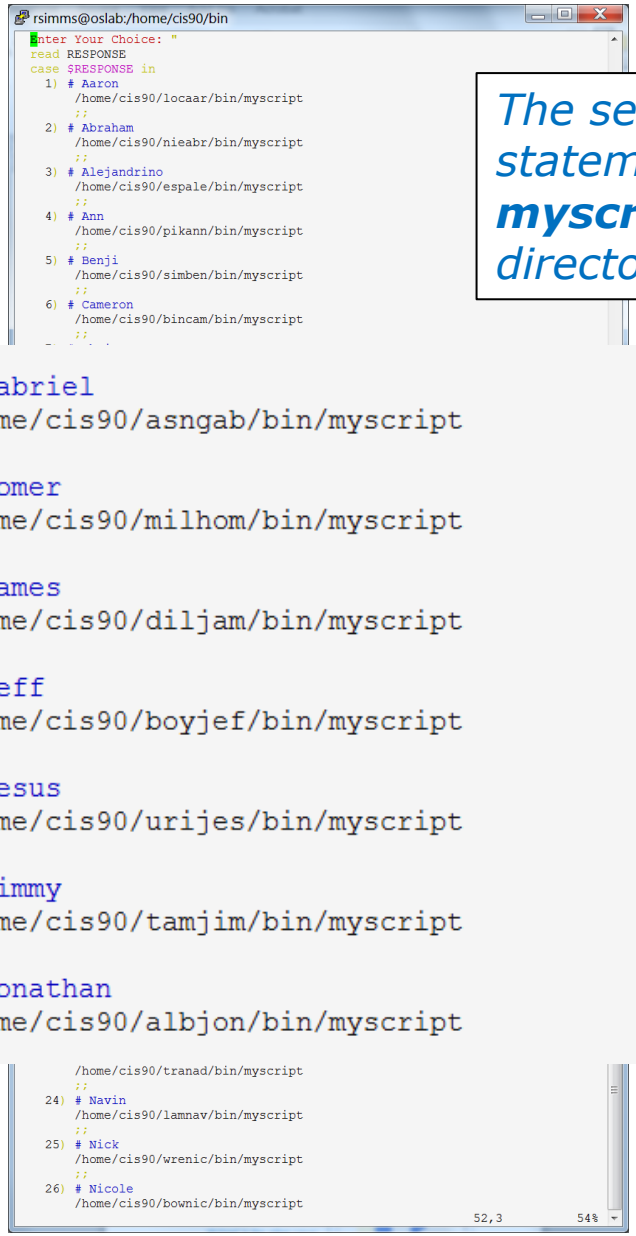

```
/home/cis90/simben $ ls -l /home/cis90/bin/allscripts  bin/myscript
-rwxr-xr-x 1 simben90 cis90 4296 Nov 13 13:07 bin/myscript
-rwxr-xr-x 1 rsimms   staff 4381 Nov 13 18:17 /home/cis90/bin/allscripts
```

rsimms@oslab:/home/cis90/bin

```
#!/bin/bash
#
# menu: A simple menu template
#
while true
do
  clear
  echo -n "
  ************************************************************
  *             Fall 2014 CIS 90 Online Projects             *
  ************************************************************
  1) Aaron
  2) Abraham
  3) Alejandrino
  4) Ann
  5) Benji
  6) Cameron
  7) Chris
  8) Cody
  9) Deane
  10) Duke
  11) Francisco
  12) Gabriel
  13) Homer
  14) James
  15) Jeff
  16) Jesus
  17) Jimmy
  18) Jonathan
  19) Joshua
  20) Justin
  21) Luis
  22) Matthew
  23) Nadia
  24) Navin
  25) Nick
  26) Nicole
  27) Paul
  28) Richard I.
  29) Richard Z.
  30) Roberto
  31) Ronald
  32) Ryan
  33) Sam
  34) Shea
  35) Shenghong
  36) Takashi
  37) Tommy

  99) Exit

  Enter Your Choice: "
  read RESPONSE
"allscripts" 173L, 3529C                    28,3          Top
```

***allscripts*** *is a bash script that will run your project script.*

*The first part of* ***allscripts*** *uses a long* ***echo*** *command to print a selection menu of the CIS 90 students. The user will enter the number corresponding to the student whose script they want to run.*




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

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

# Final Project
## allscripts (continued)

Running **/home/cis90/bin/allscripts** looks like this


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

# Final Project
## myscript

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

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
~
                                                        1,1           All
```

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

*vi understands shell scripts and will use color syntax styling.*

# Final Project

Getting Started

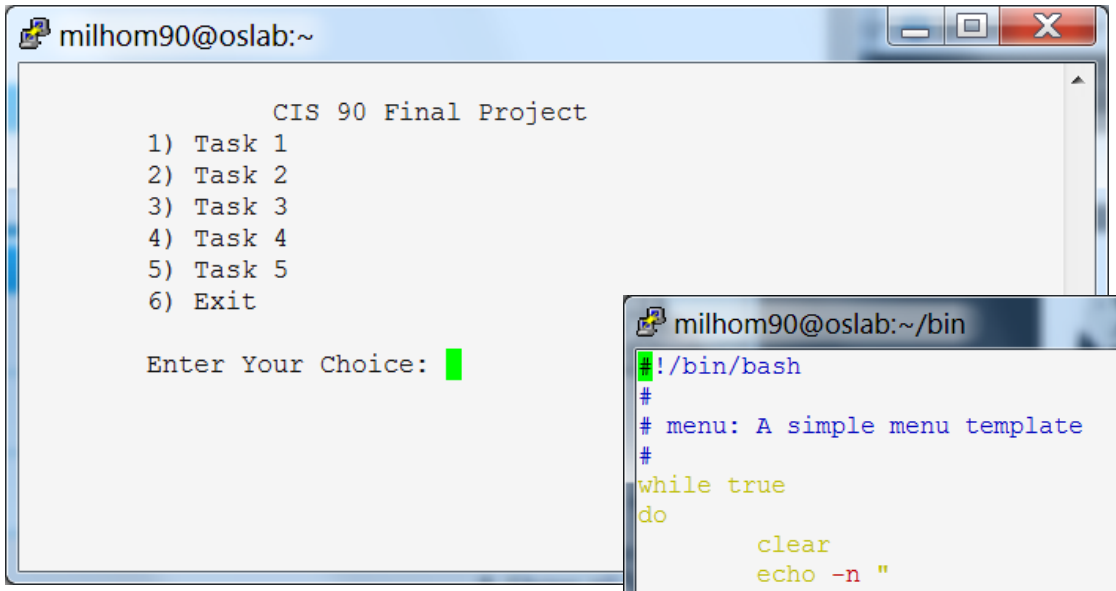
1) On Opus, cd to your home directory and enter:
   **cp ../depot/myscript bin/**

2) Give your script execute permissions with:
   **chmod +x bin/myscript**

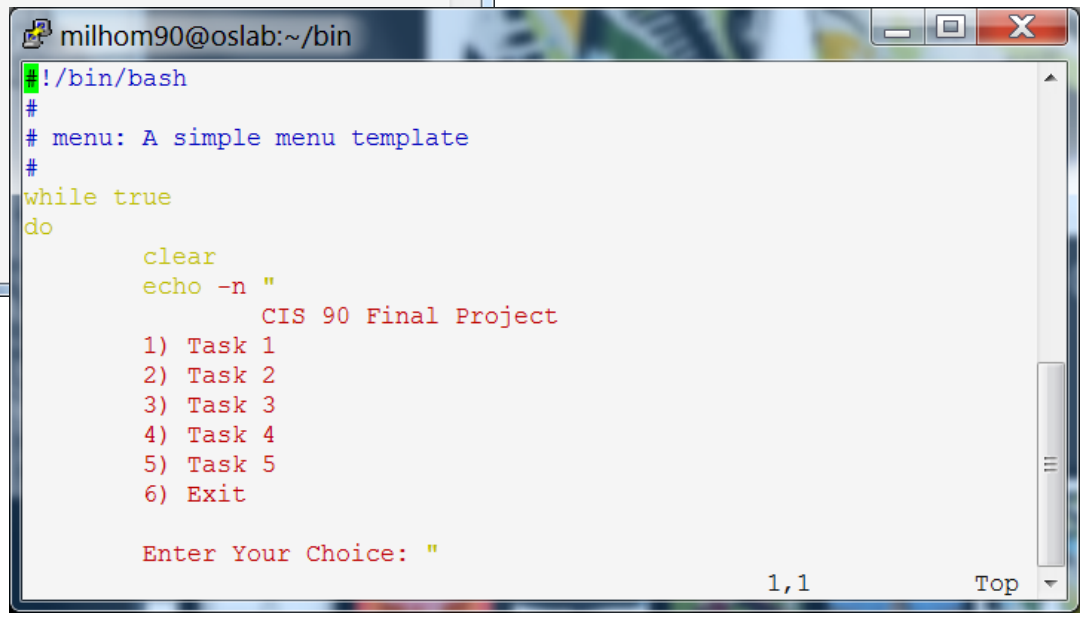
3) Run the script:
   **myscript**

# Final Project

**myscript**



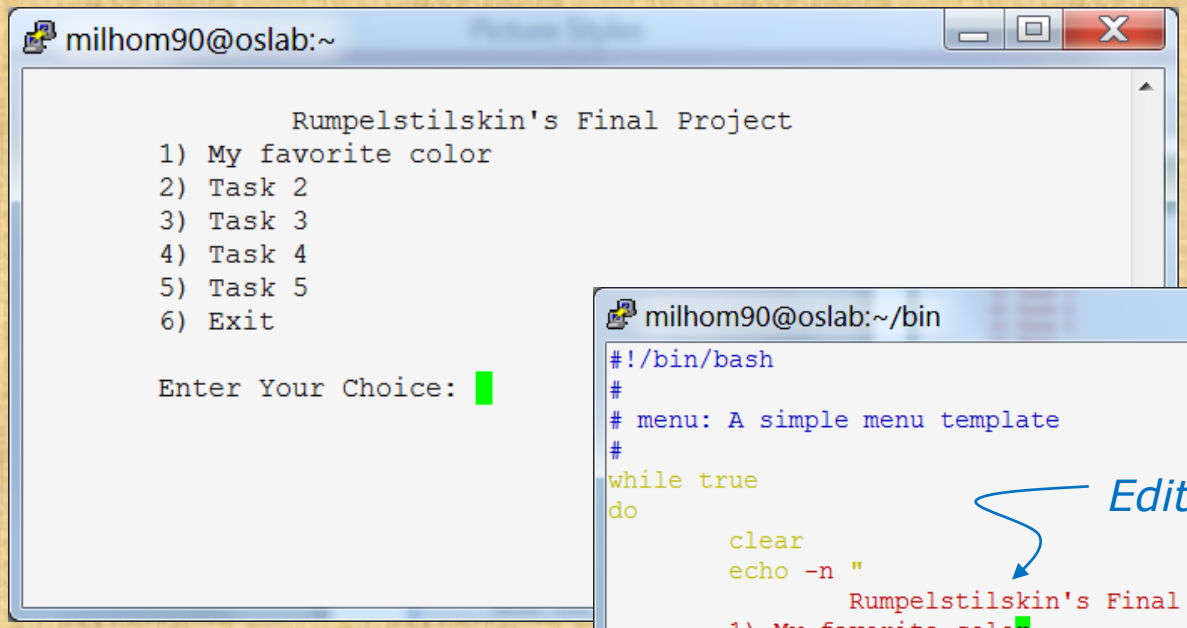

**vi myscript**



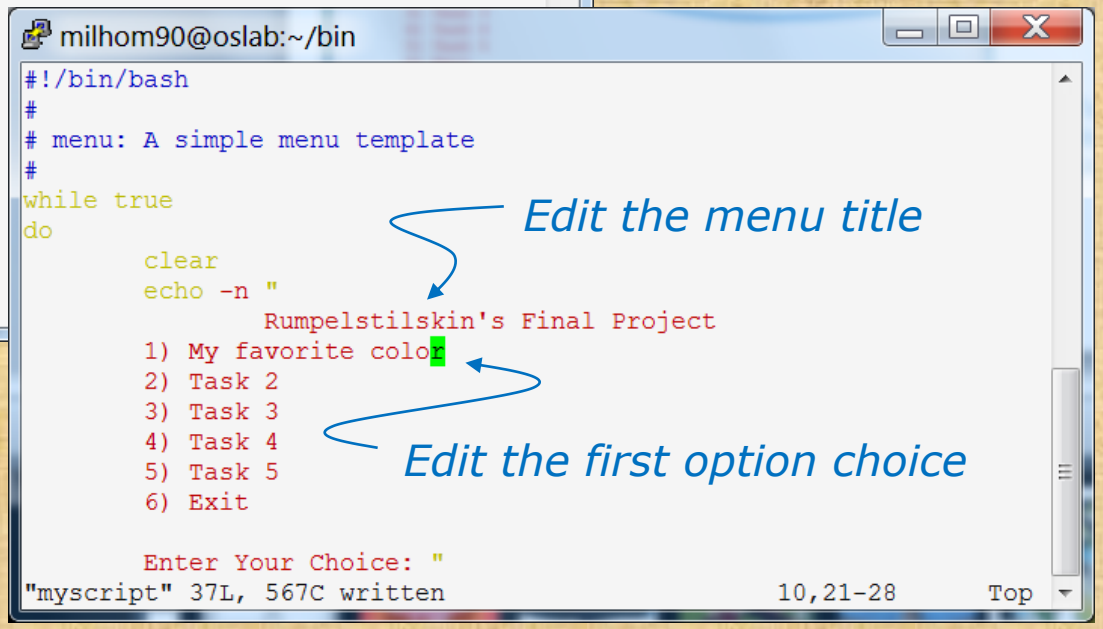

*Running and viewing the generic myscript file*

# Final Project Getting Started

**myscript**



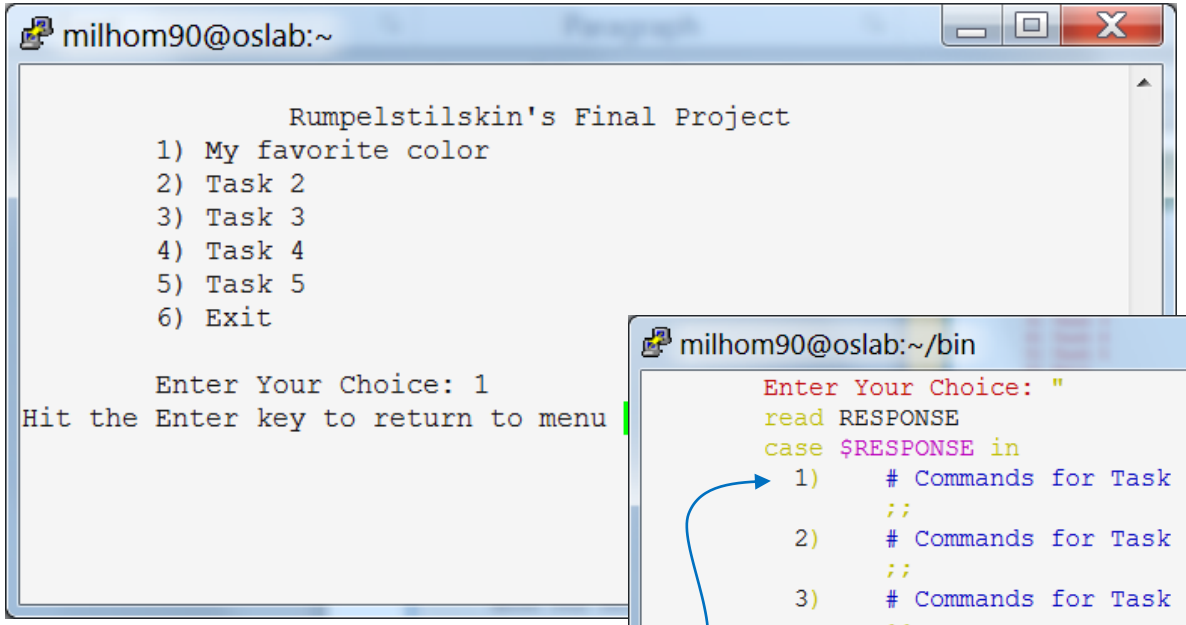

**vi myscript**



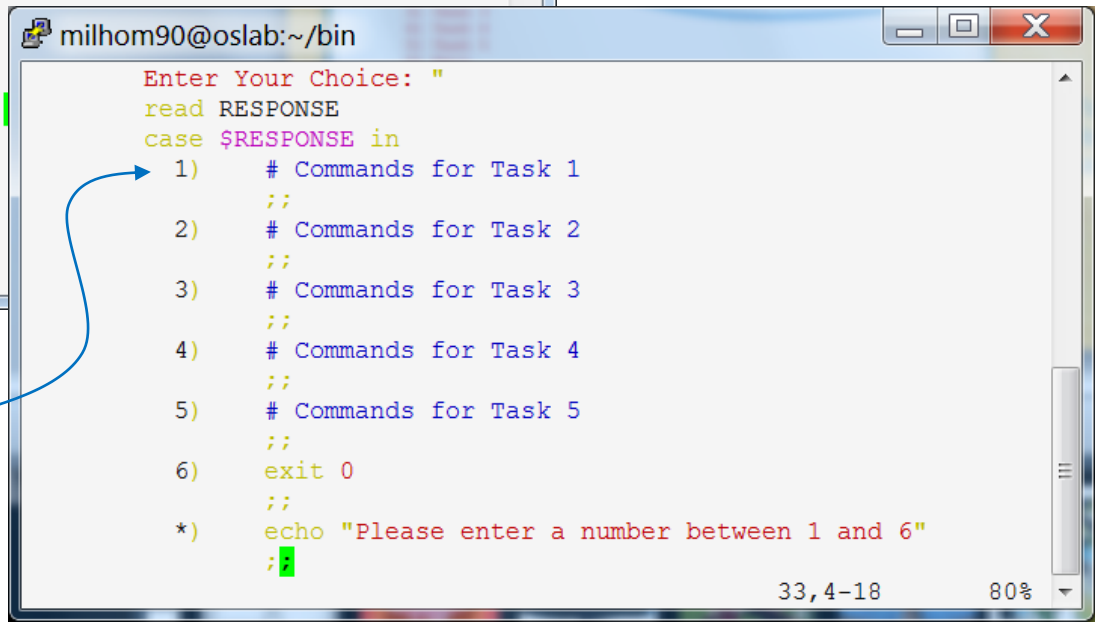

*Editing the menu title and option*

# Final Project Getting Started

**myscript**



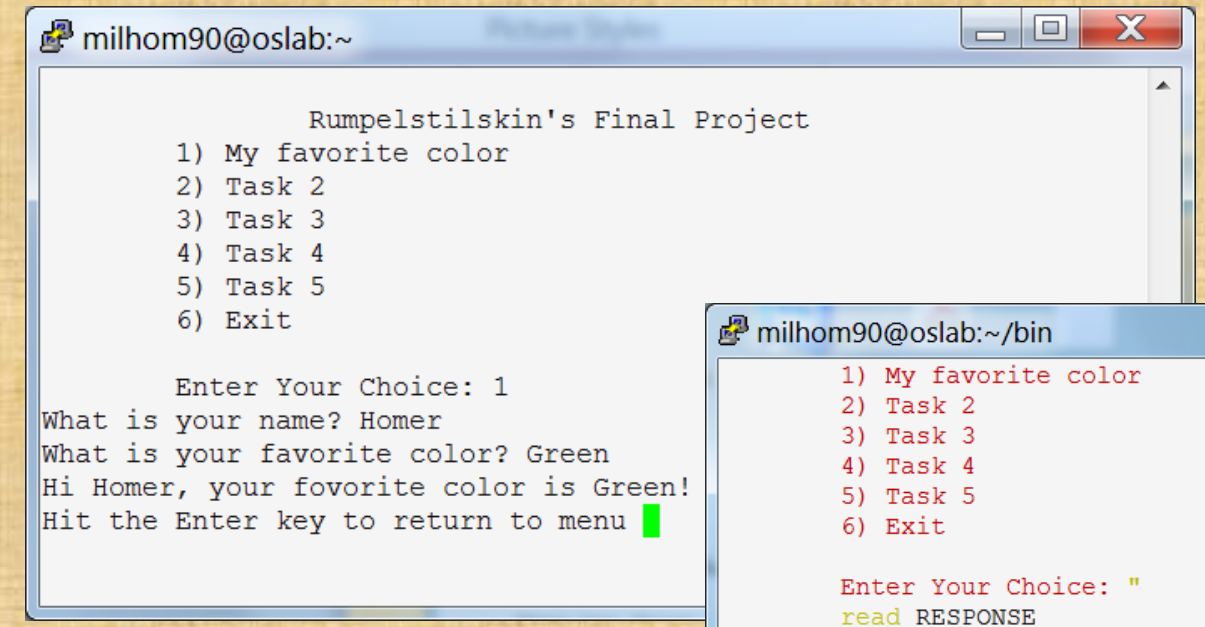

**vi myscript**



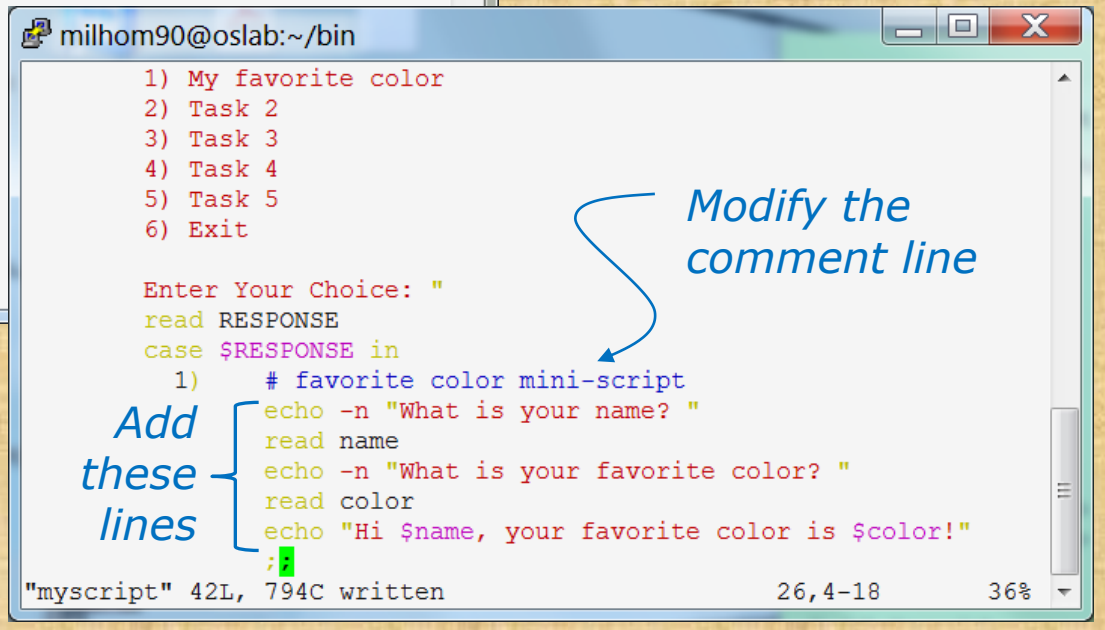

*Running Task 1 doesn't do anything yet*

# Final Project Getting Started

**myscript**




**vi myscript**




*Modify the comment line*

*Add these lines*

*We've implemented a simple task for option 1*

# 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

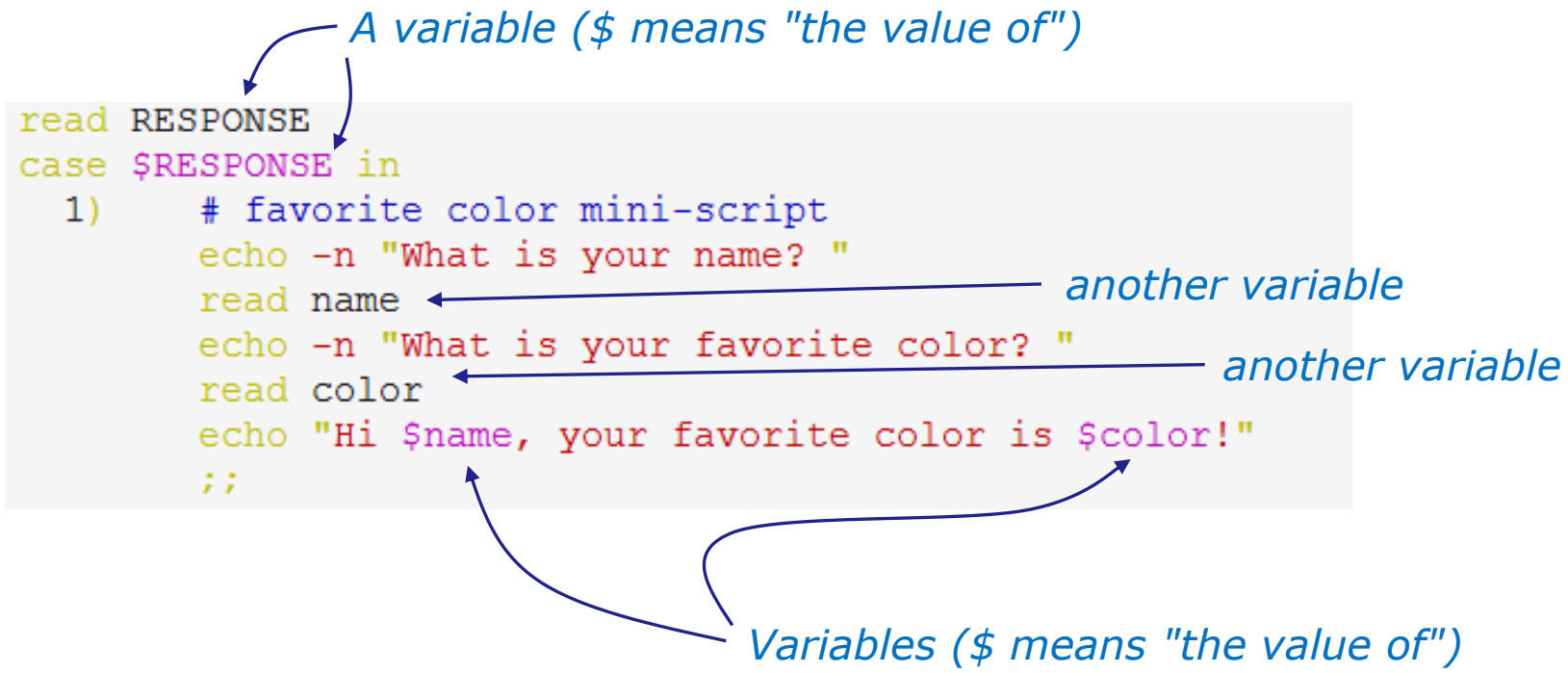
*case statement begins here*

```
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



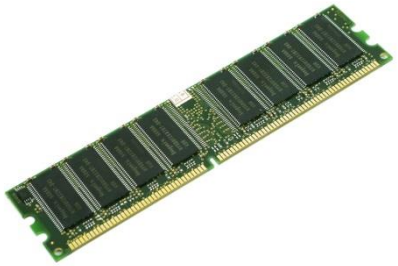
# Final Project Getting Started

```
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 #*

# Files & Variables

# Variables vs Files

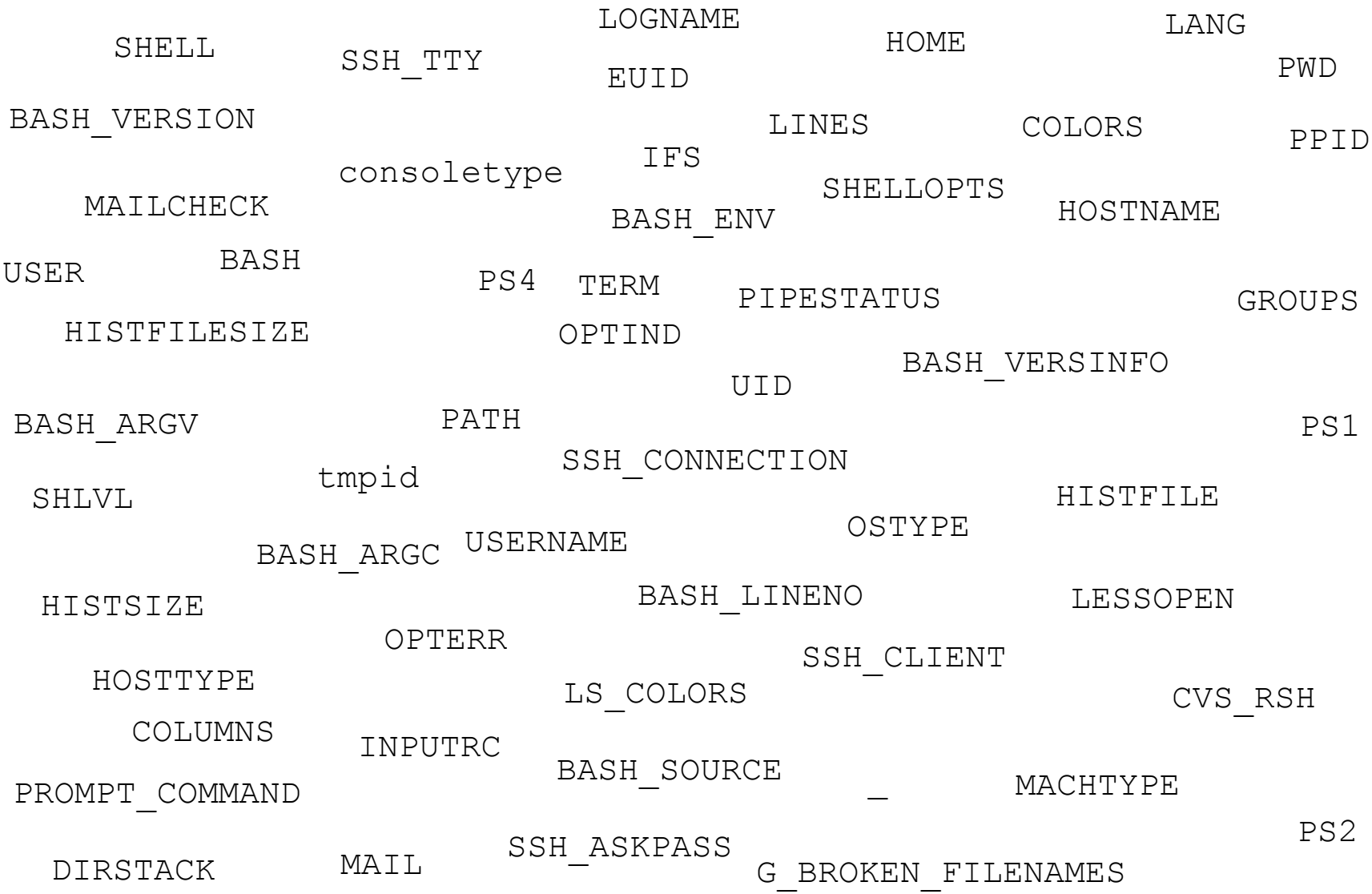


We use **variables** to reference data in memory. For example: PS1, PATH, LOGNAME, color, name


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

# Shell Variables

# Shell Variables






*Note the convention of using upper case*

# View all shell variables

```
/home/cis90/simben/Poems $ set
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

*Think of $ as "the value of"*

Use: **echo $***varname*

*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]\$
```

*Using the echo command and the $ to show the values of variables*

# Setting the values of variables

Use: *varname=value*
(no spaces please around the =)

*Do NOT use the $ when setting a variable*

*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

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

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

*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
```


```
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=84
CVS_RSH=ssh
DIRSTACK=()
EUID=1156
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=$' \t\n'
IGNOREEOF=10
INPUTRC=/etc/inputrc
LANG=en_US.UTF-8
LESSOPEN='|/usr/bin/lesspipe.sh %s'
LINES=39
LOGNAME=simben
LS_COLORS='no=00:fi=00:di=00;34:ln=00;36:pi=40;33:so=00;35:bd=40;33;01:cd=40;33;01:or=01;05;37;41:mi=01;05;37;41:ex=00;32:*.cmd=00;32:*.exe=00;32:*.com=00;32:*.btm=00;32:*.bat=00;32:*.sh=00;32:*.csh=00;32:*.tar=00;31:*.tgz=00;31:*.arj=00;31:*.taz=00;31:*
.lzh=00;31:*.zip=00;31:*.z=00;31:*.Z=00;31:*.gz=00;31:*.bz2=00;31:*.bz=00;31:*.tz=00;31:*.rpm=00;31:*.cpio=00;31:*.jpg=00;35:*.gif=00;35:*.bmp=00;35:*.xbm=00;35:*.xpm=00;35:*.png=00;35:*.tif=00;35:'
MACHTYPE=i686-redhat-linux-gnu
MAIL=/var/spool/mail/simben
MAILCHECK=60
OLDPWD=/home/cis90/simben/edits
OPTERR=1
OPTIND=1
OSTYPE=linux-gnu
PATH=/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/home/cis90/simben/../bin:/home/cis90/simben/bin:.
PIPESTATUS=([0]="0")
PPID=7254
PROMPT_COMMAND='echo -ne "\033]0;${USER}@${HOSTNAME%%.*}:${PWD/#$HOME/~}"; echo -ne "\007"'
PS1='$PWD $ '
PS2='> '
PS4='+ '
PWD=/home/cis90/simben
SHELL=/bin/bash
SHELLOPTS=braceexpand:emacs:hashall:histexpand:ignoreeof:interactive-comments:monitor
SHLVL=1
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
SSH_CLIENT='63.249.103.107 19509 22'
SSH_CONNECTION='63.249.103.107 19509 207.62.186.9 22'
SSH_TTY=/dev/pts/1
TERM=xterm
UID=1156
USER=simben
USERNAME=
_=
ac=off
consoletype=pty
defrost=on
fan=medium
```


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

*font reduced for the other variables to fit on slide*

# 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
```

# Removing Shell Variables

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

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

*show values*

```
/home/cis90/simben $ unset defrost
/home/cis90/simben $ echo  $defrost  $ac  $fan
off medium
```

*remove one of the variables*

```
/home/cis90/simben $ unset ac fan
/home/cis90/simben $ echo  $defrost  $ac  $fan

/home/cis90/simben $
```

*remove remaining variables*

# 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**

# Shell Variables

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

*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.*

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
```

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**

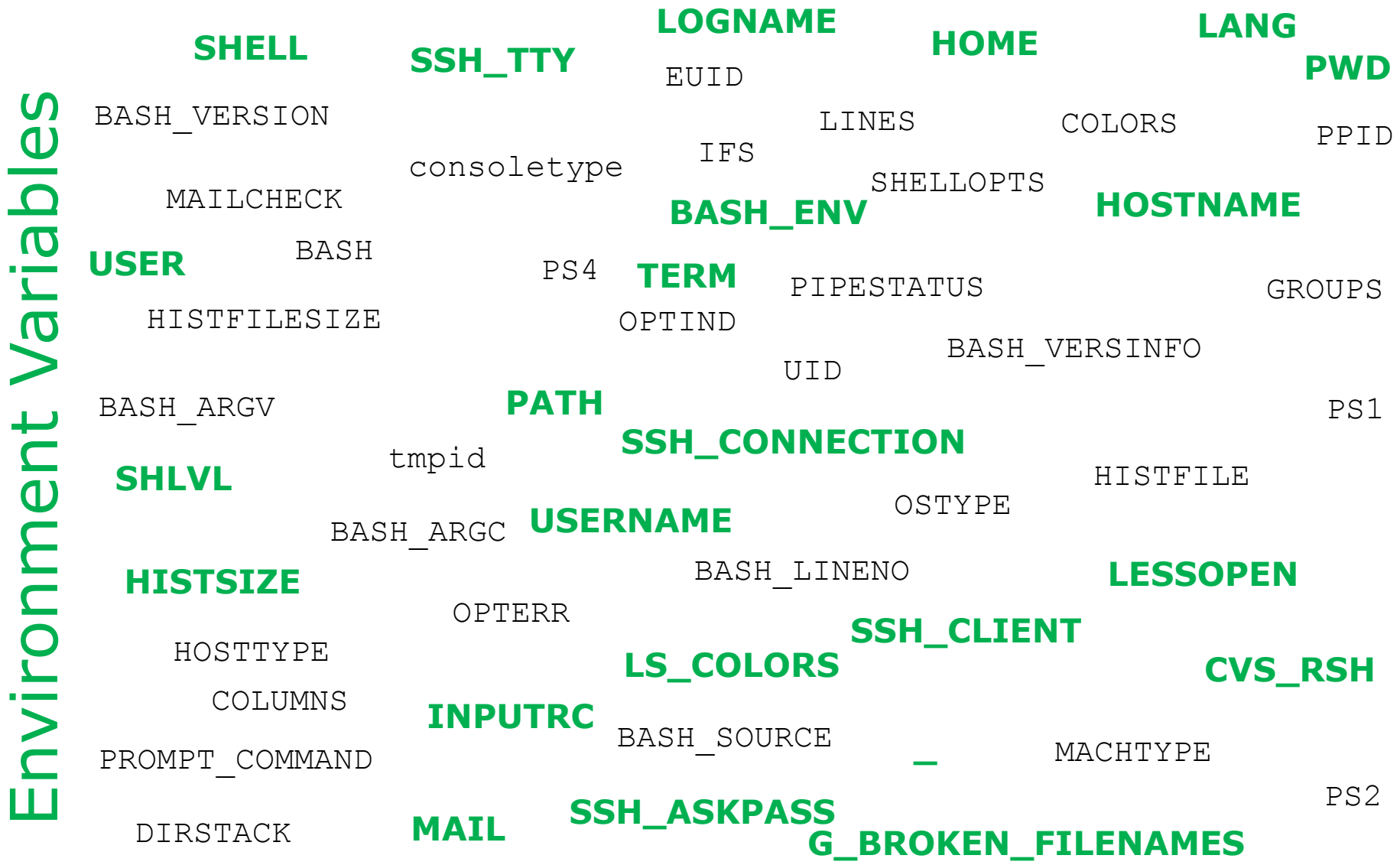
*insert the following lines then save*

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

**chmod +x funscript**

**./funscript**

# Environment Variables




*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*

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

# Customizing the shell prompt with PS1

| PS1 settings | Result |
|---|---|
| PS1='$PWD $' | /home/cis90/simben/Poems $ |
| PS1="\w $" | ~/Poems $ |
| PS1="\W $" | Poems $ |
| PS1="\u@\h $" | simben90@opus $ |
| PS1='\u@\h $PWD $' | simben90@opus /home/cis90/simben/Poems $ |
| PS1='\u@\$HOSTNAME $PWD $' | simben90@opus.cabrillo.edu /home/cis90/simben/Poems $ |
| PS1='\u \! $PWD $' | simben90 825 /home/cis90/simben/Poems $ |
| PS1="[\u@\h \W/\$" | [simben90@opus Poems/$ |
| PS1="Enter command: " | Enter command: |

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!

# 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) |

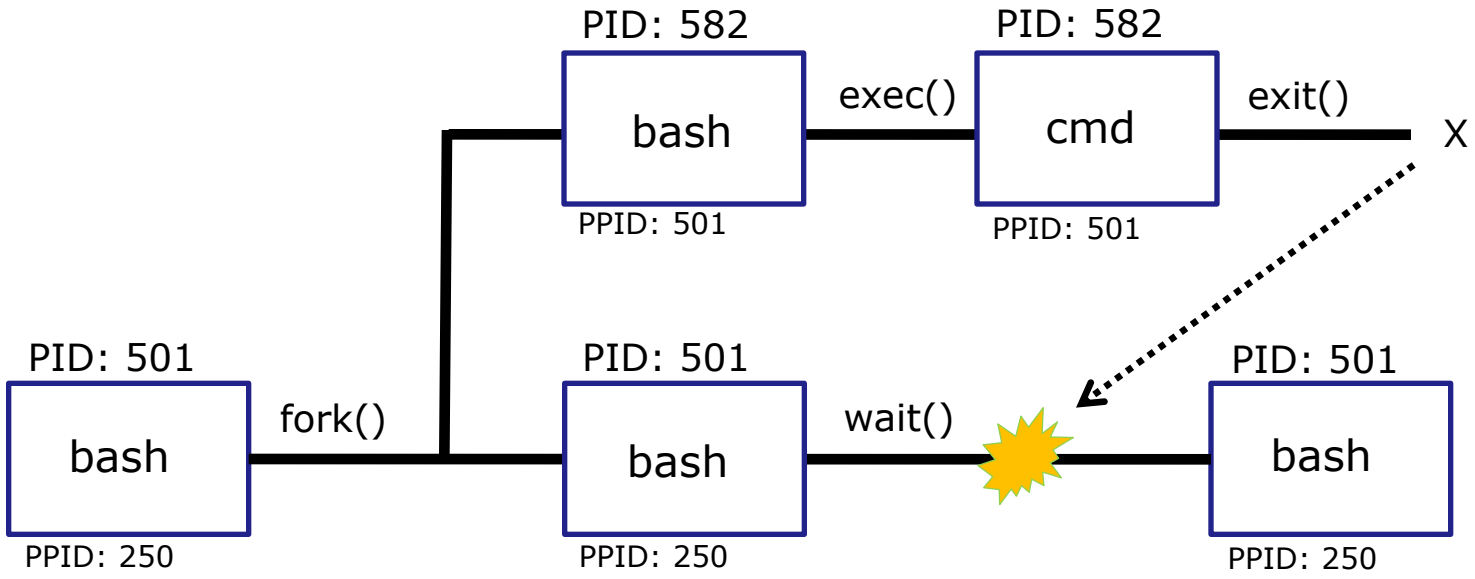
The prompt string can have any combination of text, variables and these codes.

# variables and child processes

# The rules of the road for 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.

# 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

- 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.

# 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

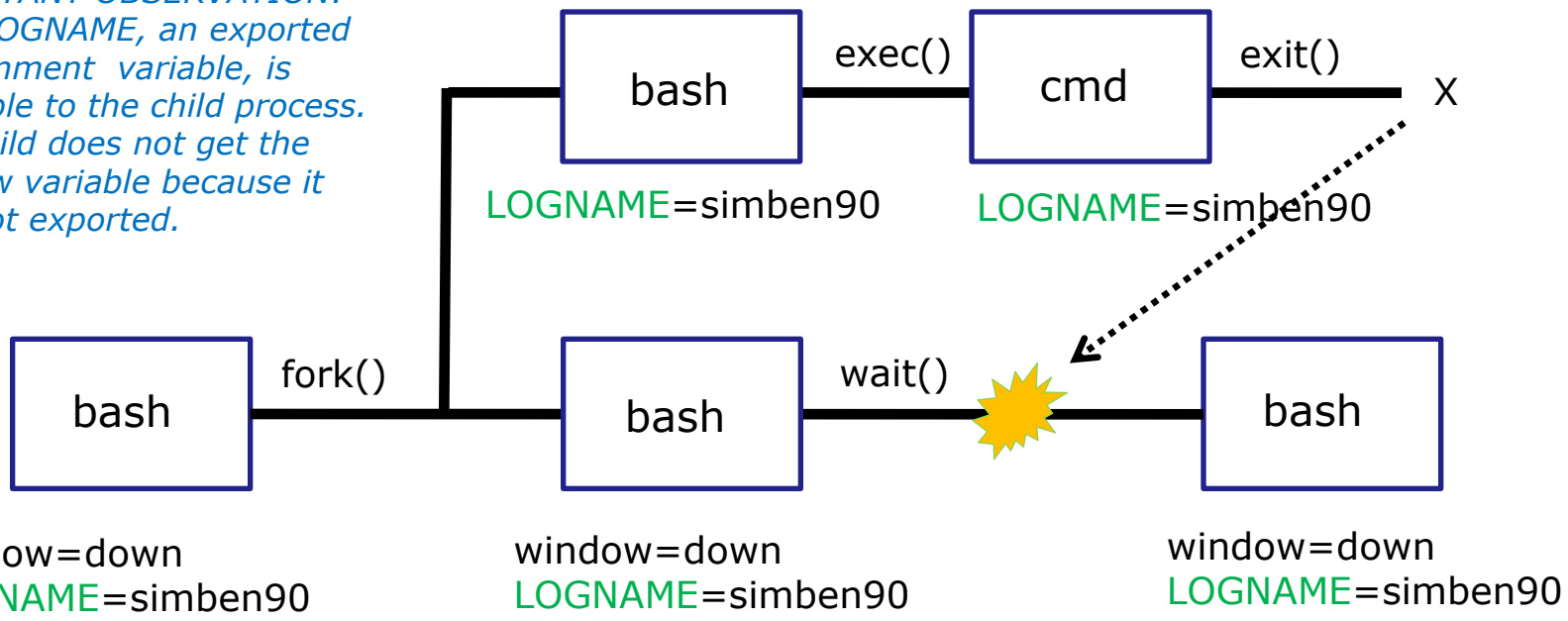
```
/home/cis90/simben $ bash
[simben@opus ~]$ echo $window $LOGNAME
simben90
[simben@opus ~]$ exit
exit
```

*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

- 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.

# Changes made by the child do not affect the parent

1 parent

```
/home/cis90/simben $ echo $window
down
/home/cis90/simben $ export window
```

*export window so it is available to children*

2 child

```
/home/cis90/simben $ bash
[simben@opus ~]$ echo $window
down
```

*a copy of window is now available to the child process*

3 child

```
[simben@opus ~]$ window=up
[simben@opus ~]$ echo $window
up
[simben@opus ~]$ exit
exit
```

*the child modifies the window variable*

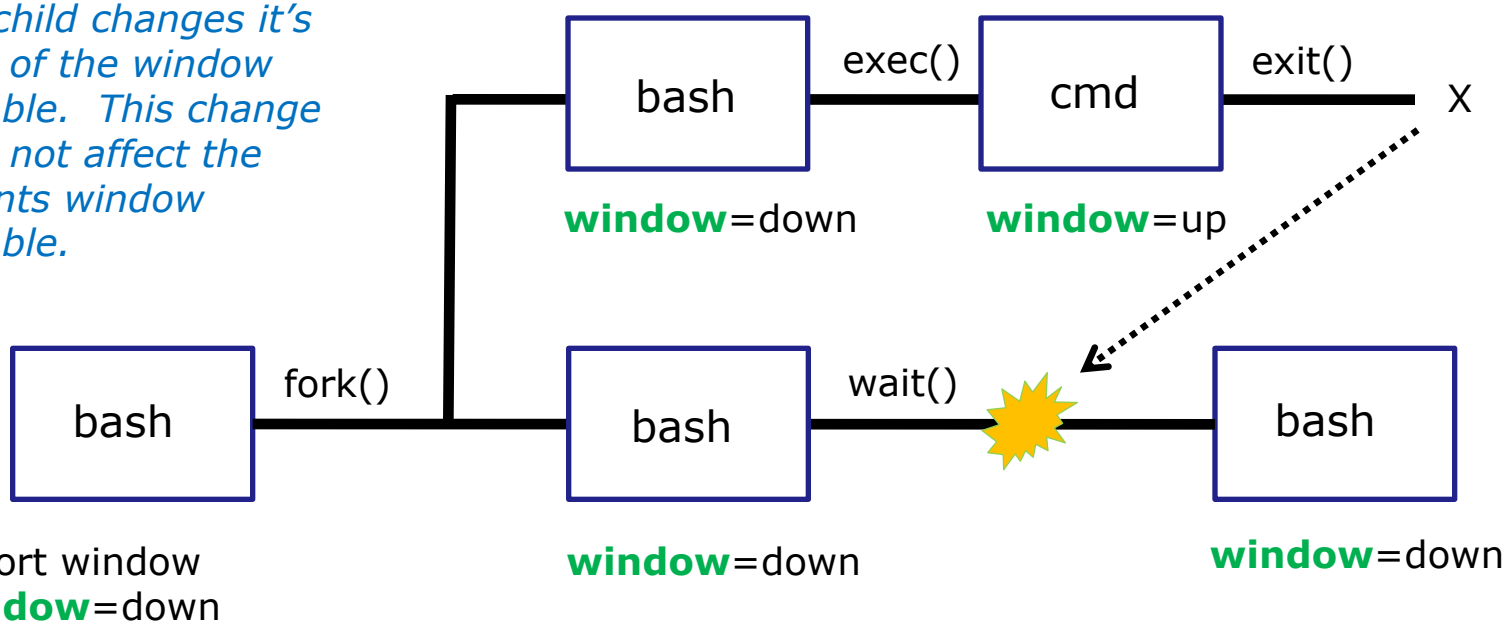
4 parent

```
/home/cis90/simben $ echo $window
down
```

*The modifications made by the child do not affect the parent's variable*

# Changes made by the child do not affect the parent

*The child changes it's 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.

# 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

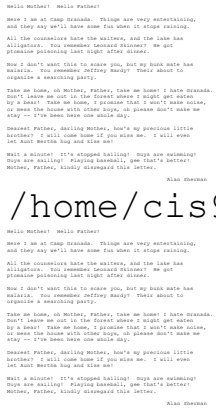
*Yes, an alias can be made using another alias*

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
```



*reduced size to fit on page*

2

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

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



*reduced size to fit on page*

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

For example:

- **alias mypath="echo $PATH"**
- **mypath**

- **alias probe=file**
- **probe /usr/bin/spell**

# 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** (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

# .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

# .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
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/../bin:$HOME/bin:.
BASH_ENV=$HOME/.bashrc
USERNAME=""
PS1='$PWD $ '
export USERNAME BASH_ENV PATH
umask 002
set -o ignoreeof
stty susp ^F
eval `tset -s -m vt100:vt100 -m :\?${TERM:-ansi} -r -Q `

[simben@opus ~]$
```

*sources the .bashrc file*

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

*The special prompt used for CIS 90 students is specified*

*variables are exported*

*umask value is set*

*EOF's are ignored*

*Suspend character redefined from Z to F*

*Terminal type is requested and set*

# .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
fi
alias print="echo -e"
[simben@opus ~]$
```

*sources the /etc/bashrc file*

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

# 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:~

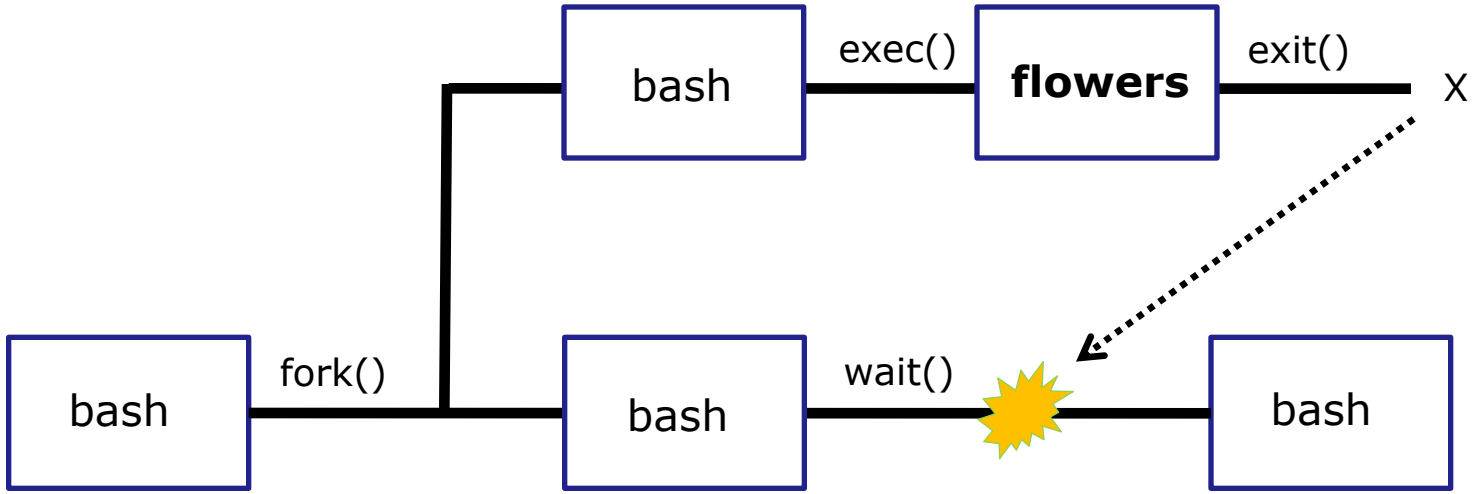
```
#!/bin/bash
#
# Useful alias:
#   alias go='echo roses are \"$roses\" and violets are \"$violets\"'
#
echo
echo "==> Entering child process <=="
ps
echo "==> showing variables in child <=="
echo "  " roses are '"'$roses'"'
echo "  " violets are '"'$violets'"'
echo "==> setting variables in child <=="
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"

/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 ""
==> setting variables in child <==
==> Leaving child process <==

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

*The parent sees roses and violets*

*The child does not see roses or violets*

*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"

/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 <==

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

*The parent sees roses and violets*

*The child now sees roses since it was exported*

*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"

/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 <==

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

*The parent sees roses and violets*

*script is not running as child*

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

*The variables are changed after running flowers script*

# Wrap up

# 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.*

# 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:

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

New Files and Directories:

| | |
|---|---|
| .bash_profile | - executed at login |
| .bashrc | - 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*

# /bin/mail and vi


*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.cabril  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

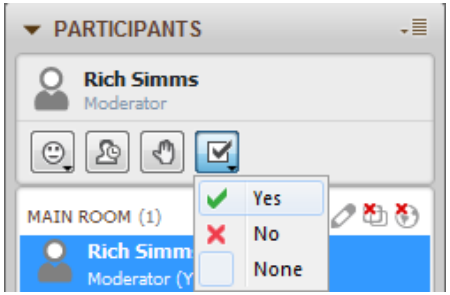
Try it!

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