



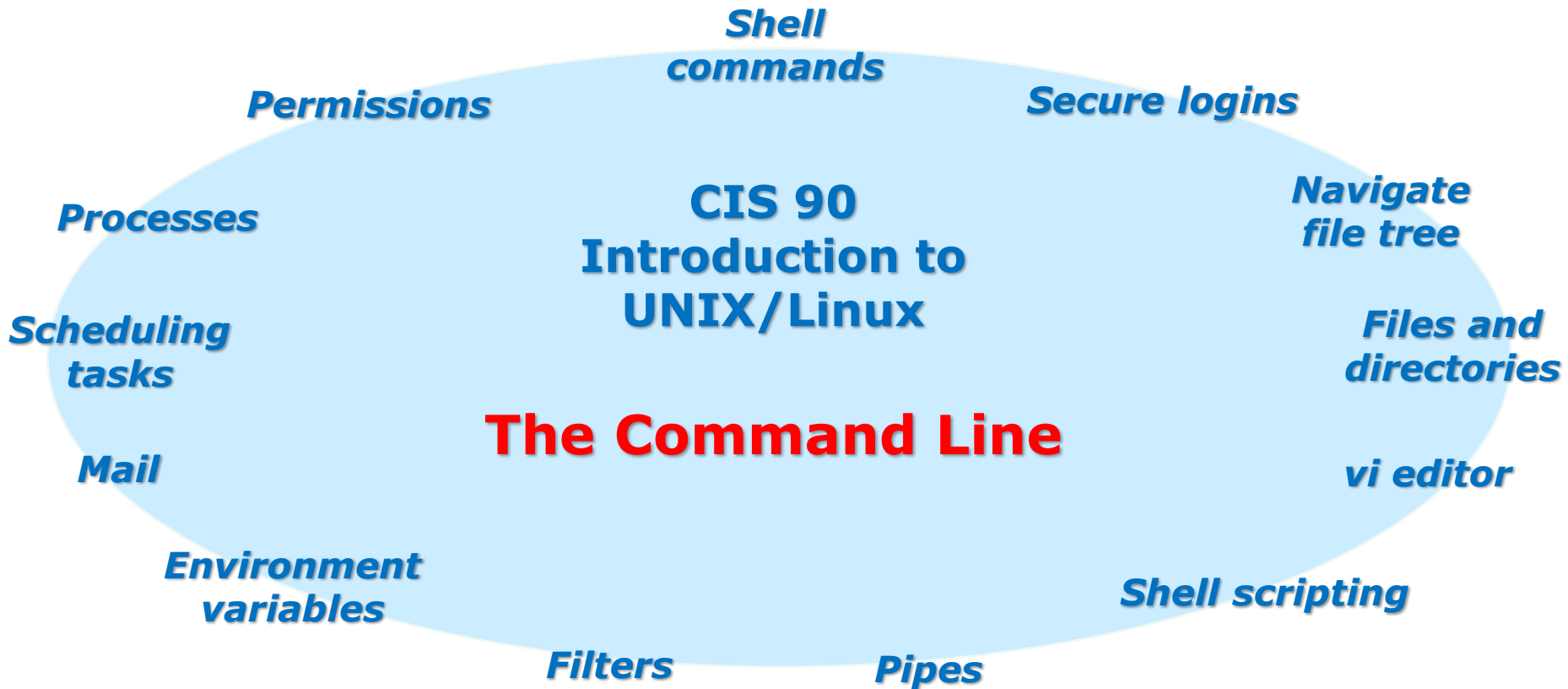
Rich's lesson module checklist

Last modified 04/24/2019

- Zoom recording named and published for previous lesson
- Slides, Lab 10 and Project posted
- Print out agenda slide and annotate page numbers
- Flash cards
- 1st minute quiz
- Web Calendar page updated
- Lock turnin directory at midnight (scripts/schedule-submit-locks)
- Project updated and published
 - allscripts updated
 - myscript in depot
 - sample myscripts for the doggies
- Lab 10 updated and published
 - riddle set to riddle1
- flowers script in bin
- Allscripts related slides updated with latest classmates
- Backup slides, CCC info, handouts on flash drive
- Spare 9v battery for mic
- Key card for classroom door

<https://zoom.us>

- Putty + Slides + Chrome
- Enable/Disable attendee sharing
 - ^ > Advanced Sharing Options > Only Host
- Enable/Disable attended annotations
 - Share > More > Disable Attendee Sharing



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: <https://web.archive.org/web/20140209023942/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. John's site: <http://teacherjohn.com/>
- Jaclyn Kostner for many webinar best practices: e.g. mug shot page.



Student checklist - Before class starts

Rich's Cabrillo College CIS Classes
CIS 90 Calendar

CIS 90 (Fall 2014) Calendar

Course Dates: [Genda](#) [Calendar](#)

[CIS 90](#)

Lesson	Date	Topics	Links
	9/2	<p>Class and Linux Overview</p> <ul style="list-style-type: none"> Understand how the course will work High-level overview of computers, operating systems, and virtual machines Overview of LINUX/Linux market and architecture Using SSH for remote network logs Using terminals and the command line <p>Methods</p> <p>Presentation slides (download)</p> <p>Supplemental</p> <ul style="list-style-type: none"> PowerPoint: Logging into Opus (download) <p>Assignments</p> <ul style="list-style-type: none"> Student Survey Lab 1 <p>ECE Center</p> <p>Enter virtual classroom</p> <p>Quiz 1</p> <p>Comments</p>	<p>(pdf)</p> <p>2.4.5 p163-172 p164-172 (pdf)</p>

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 ConferZoom.
7. Log into Opus-II with Putty or ssh command.



Student checklist - Before class starts

Google

ConferZoom

Downloaded PDF of Lesson Slides. I like Foxit Reader so I can take notes using annotations.

The screenshot shows a Zoom meeting interface. The main window displays a virtual car with the text "Get into the car" overlaid. The Zoom toolbar at the bottom includes buttons for Unmute, Start Video, Invite, Participants, Share Screen, Chat, Record, and Leave Meeting. Several browser windows are open in the background:

- A Google search page.
- A page titled "Rich's Cabrillo College CIS 90 Calendar" with a table of lessons.
- A PDF document titled "CIS 90 - Lesson 1" with the text "90 System Playground" and "Each student gets their own Arya VM for the term".
- A terminal window showing a login process for "Opus-II".

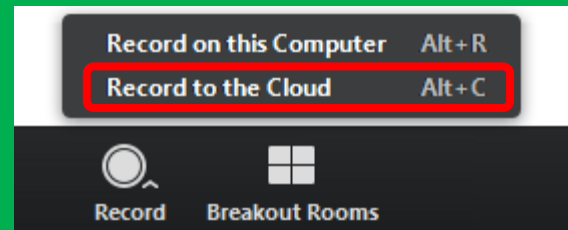
Lesson	Date	Topics
1	1/31	Class and Linux Overview • Understand how this course works • Overview of computers and networks • Learn how to login via ssh • Learn first UNIX/Linux commands

CIS 90 website Calendar page

One or more login sessions to Opus-II



Start



Start Recording

Audio Check



Start Recording

Audio & video Check



Instructor: **Rich Simms**
Dial-in: **669-900-6833 (toll)**
Meeting ID: **426 283 384**



Nick



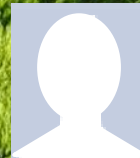
Ryan



Erik



Matt



David



Jon



Cheryl



Wais



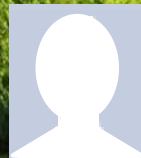
Tanisha



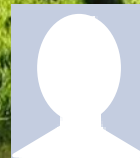
Daniel



Ohunayo



Sequoia



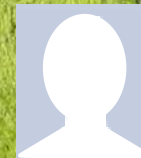
Scott



Lucky



Cole



Shane



Jim



Joseph



Mark



Adina



Evie



Cody

First Minute Quiz

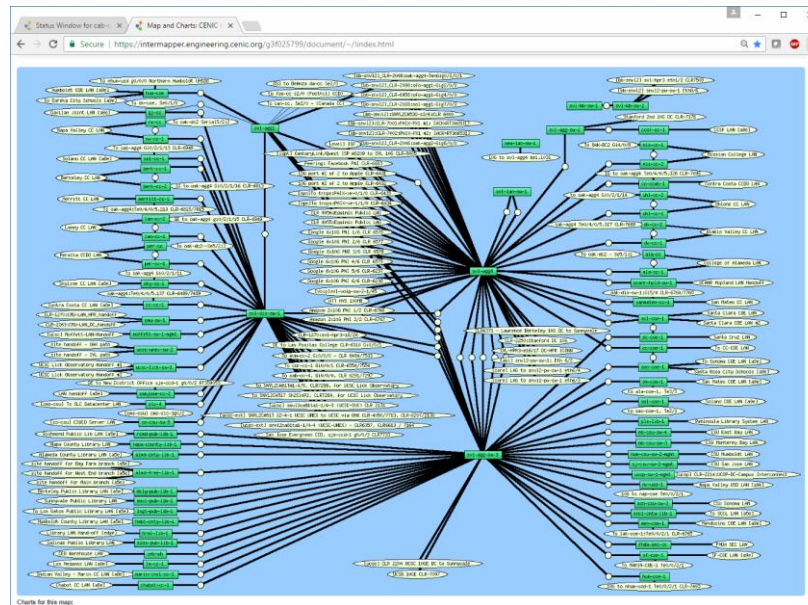
Please answer these questions **in the order** shown:

Use ConferZoom White Board

email answers to: risimms@cabrillo.edu

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

Network Check



[https://intermapper.engineering.cenic.org/g3f025799/
document/~!/index.html](https://intermapper.engineering.cenic.org/g3f025799/document/~!/index.html)

The Shell Environment

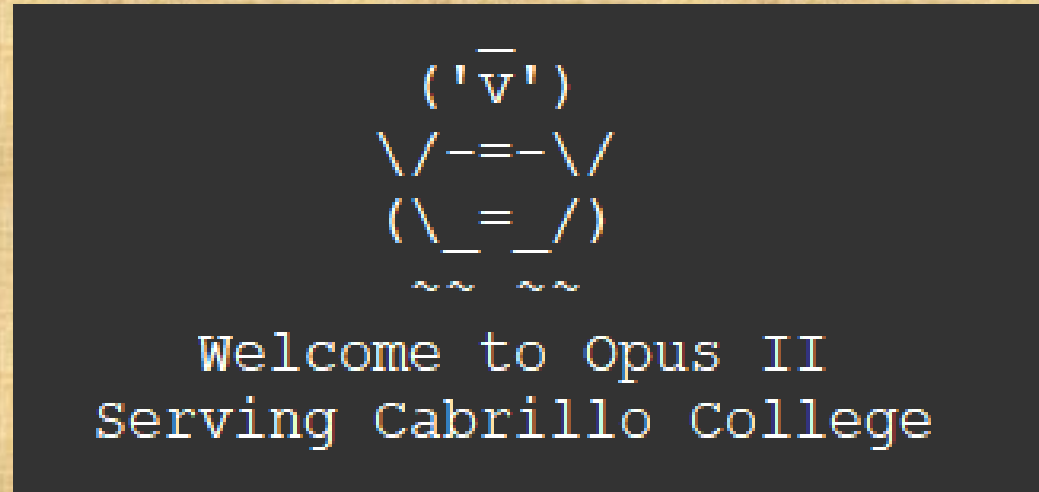
Objectives

- 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

Class Activity



If you haven't already,
log into Opus-II

Class Activity



Unit 3

Electronic Mail

- Guest speaker: Denise Moore on OTC (On-The-Job) training programs
- Learn how to use the LINC communication tools write and /bin/mail
- Overview on and-to and mail

Materials

- Presentation slides ([download](#))

Supplemental

- Howto #318: Accessing vlab ([download](#))

Assignment

- Read/skim Lesson 3 slides

<https://simms-teach.com/cis90calendar.php>

If you haven't already,
download the lesson slides

Class Activity

	<ul style="list-style-type: none">• <u>Read/skim Lesson 1 slides</u>• <u>Student Survey</u>• <u>Lab 1</u>		
	ConferZoom <ul style="list-style-type: none">• <u>Enter virtual classroom</u>• <u>Class archives</u>		
	Quiz 1		
	Commenda <ul style="list-style-type: none">• Understand how the UNIX login operation		

<https://simms-teach.com/cis90calendar.php>

If you haven't already, join
ConferZoom classroom



Questions



Questions?

Lesson material?

Labs? Tests?

How this course works?

• Graded work & tests
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

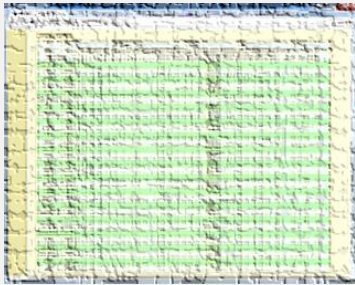
他問一個問題，五分鐘是個傻子，他不問一個問題仍然是一個
傻瓜永遠。

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

Review your progress in the course

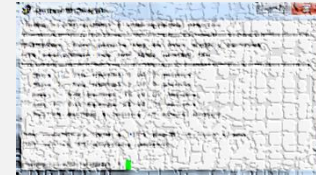
Check the website Grades page

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



Or check on Opus-II

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



Written by Jesse Warren a past CIS 90 Alumnus

- **Send me your survey to get your LOR codename.**
- **Graded labs and tests are in your home directories.**

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

Extra Credit

*In lesson slides
(search for extra credit)*

On the forum

Be sure to monitor the forum as I may post extra credit opportunities without any other notice!

On some labs

Extra credit (2 points)

For a small taste of what you would learn in CIS 191 let's add a new user to your Arya VM. Once added we will see how the new account is represented in `/etc/passwd` and `/etc/shadow`.

1. Log into your Arya VM as the cis90 user. Make sure it's your VM and not someone else's.
2. Install the latest updates:
`sudo apt-get update`
`sudo apt-get upgrade`
3. Add a new user account for yourself. You may make whatever username you wish. The example below shows how Benji would make the same username he uses on Opus:
`sudo useradd -G sudo -c "Benji Simms" -m -s /bin/bash simben90`



On the website

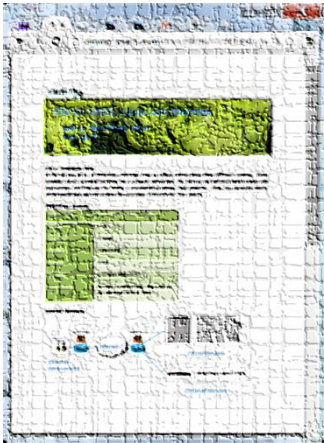
<http://simms-teach.com/cis90grades.php>

For some flexibility, personal preferences or family emergencies there is an additional 90 points available of **extra credit** activities.

<http://simms-teach.com/cis90extracredit.php>

• **Website content review** - The first person to email the instructor pointing out an error or typo on this website will get one point of extra credit for each unique error. The email must specify the specific document or web page, pinpoint the location of the error, and specify what the correction should be. Duplicate errors count as a single point. This does not apply to pre-published material that has been updated but not yet presented in class. (Up to 20 points total)

Lab Assignments -- Pearls of Wisdom



- Don't wait till the last minute to start.
- Plan for things to go wrong and give yourself time to ask questions and get answers.
- The *slower* you go the *sooner* you will be finished.
- A few minutes reading the forum can save you hour(s).
- Line up materials, references, equipment and software ahead of time.
- It's best if you fully understand each step as you do it. Use Google or refer back to lesson slides to understand the commands you are using.
- Keep a growing cheat sheet of commands and examples.
- Study groups are very productive and beneficial.
- Use the forum to collaborate, ask questions, get clarifications and share tips you learned while doing a lab.
- **Late work is not accepted** so submit what you have for partial credit.

Getting Help When Stuck on an Assignment

- Google the topic/error message.
- Search the Lesson Slides (they are PDFs) for a relevant example on how to do something.
- Check the forum. Someone else may have run into the same issue and found a way past it. If not start a new topic, explain what you are trying to do and what you have tried so far.
- Talk to a tutor/assistant at the CTC (room 1403) or CIS Lab (STEM Center).
- Come see me during my office or lab hours:

<https://www.cabrillo.edu/salsa/listing.php?staffId=1426>

I'm in the CTC (room 1403) every Tuesday from 3:30-6:00 pm.

- Make use of the Open Questions time at the start of every class.
- Make a cheat sheet of commands and examples so you never again get stuck on the same thing!

CIS Labs always involve some troubleshooting!

Help Available! In the CTC and CIS Lab

Rich's Cabrillo College CIS Classes CIS 90 Calendar

Home

Resources

Forums

Tutors

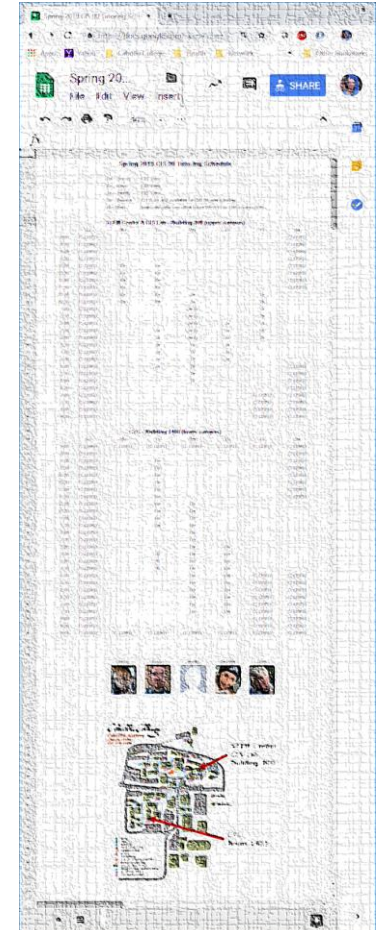
Canvas

Cabrillo College
Cabrillo Gallery
Library #1002
831-479-6308

CIS Lab
in STEM Center
Building 800

*To see tutor
schedule, click
the Tutors link
on the
website.*

*Instructors, tutors
and equipment are
available for CIS
students to work on
assignments.*



CTC
Room 1403

Help Available! In the CTC and CIS Lab

Rich's Cabrillo College CIS Classes
CIS 90 Calendar

Home Resources Forums **Tutors** Canvas

*To see tutor schedule, click the
Tutors link on the website.*



*The CIS Lab is in the STEM
center (Building 800)*



*Room 1403 is in the
CTC (Building 1400)*



The slippery slope



- 1) If you didn't submit the last lab ...
- 2) If you were in class and didn't submit the last quiz ...
- 3) If you didn't send me the student survey assigned in Lesson 1 ...
- 4) If you haven't made a forum post in the last quarter of the course ...
- 5) If you had trouble doing the last test ...

Please contact me by email, see me during my office hours or when I'm in the CTC

Email: risimms@cabrillo.edu



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@opus-ii:~
Worldwide Game of Thrones Vocabulary
Albanian: Dimri po vjen.
Chinese: 冬天来了。
Czech: Zima se blíží.
Danish: Vinteren er på vej.
Dutch: De winter komt eraan.
English: Winter is coming.
Finish: Talvi on tulossa.
French: L'hiver arrive.
German: Der Winter kommt.
Hindoi: सर्दी अ रही है।
Hungarian: Közeleg a tél.
Irish: Geimhridh ag teacht.
Italian: L'inverno sta arrivando.
Japanese: 冬が来ています。
Kazakh: Қысқы келе жатыр.
Latvian: Zieme nāk.
Lithuanian: Žiema ateina.
Polish: Zima się zbliża.
Portuguese: O inverno está chegando.
Russian: Скоро зима.
Spanish: Se acerca el invierno.
Swedish: Vintern är på väg.
Turkish: Kış geliyor.
Ukrainian: Скоро зима.
Welsh: Gaeaf yn dod.
~
"vocab" 26L, 772C
```

Editing vocab in one login session

```
simben90@opus-ii:~
E325: ATTENTION
Found a swap file by the name ".vocab.swp"
      owned by: simben90   dated: Mon Apr 23 16:40:33 2018
      file name: ~simben90/vocab
      modified: no
      user name: simben90   host name: opus-ii.cis.cabrillo.edu
      process ID: 21770 (still running)
While opening file "vocab"
      dated: Mon Apr 23 16:40:14 2018

(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@opus-iii:~
E325: ATTENTION
Found a swap file by the name ".vocab.swp"
      owned by: simben90   dated: Mon Apr 23 16:40:33 2018
      file name: ~simben90/vocab
      modified: no
      user name: simben90   host name: opus-ii.cis.cabrillo.edu
      process ID: 21770 (still running)
While opening file "vocab"
      dated: Mon Apr 23 16:40:14 2018

(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.



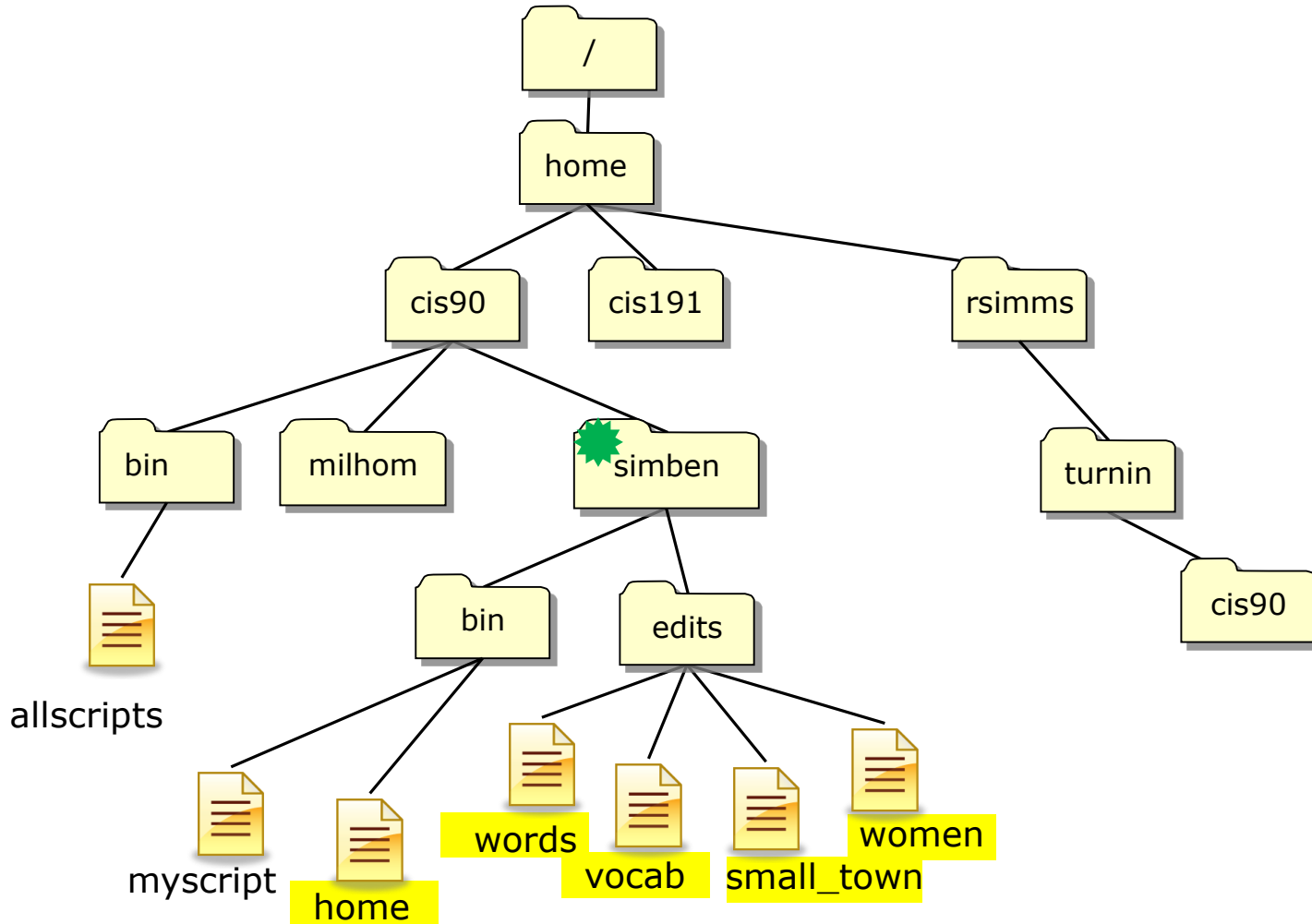
Submitting Lab 9 & Pathnames!


REMINDER

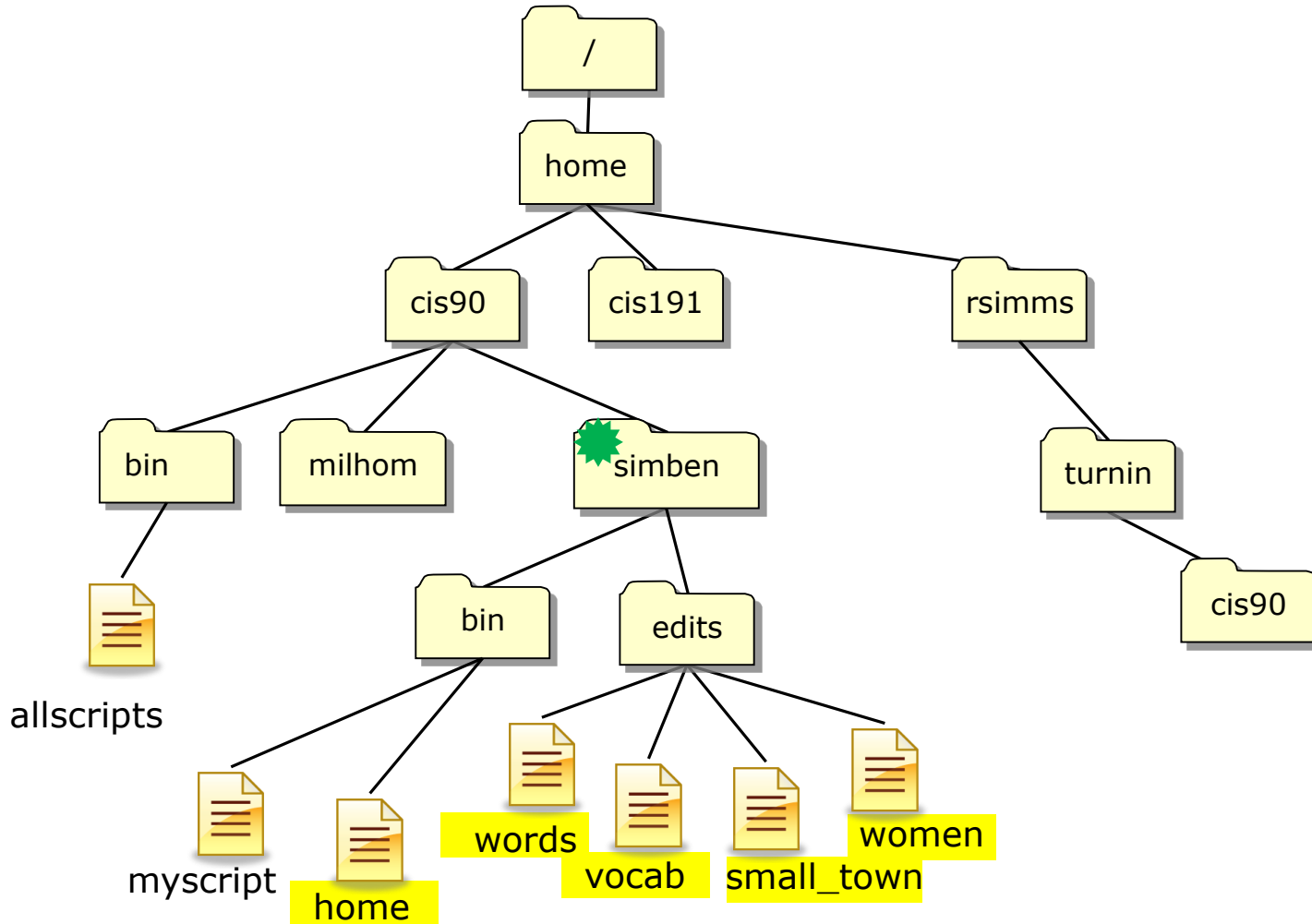
- You must **ALWAYS** use **VALID PATHNAMES** when specifying files as **ARGUMENTS** on a command.
- Pathnames can be relative or absolute.
- A common mistake in the past on Lab 9 is to ignore error messages and not submit all the files requested.




One way



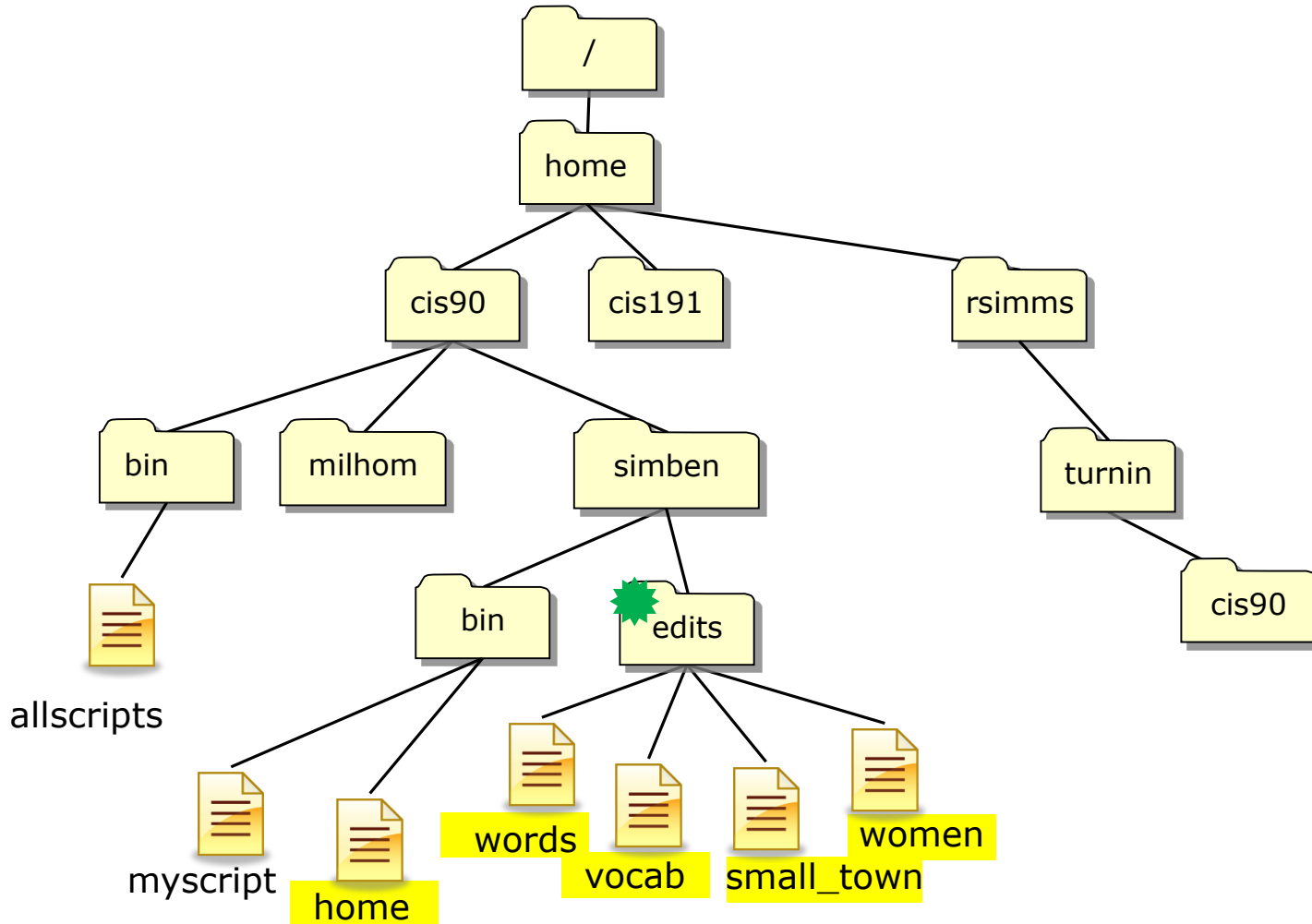
From  how could Benji concatenate the highlighted files into a file named lab09 in his home directory?




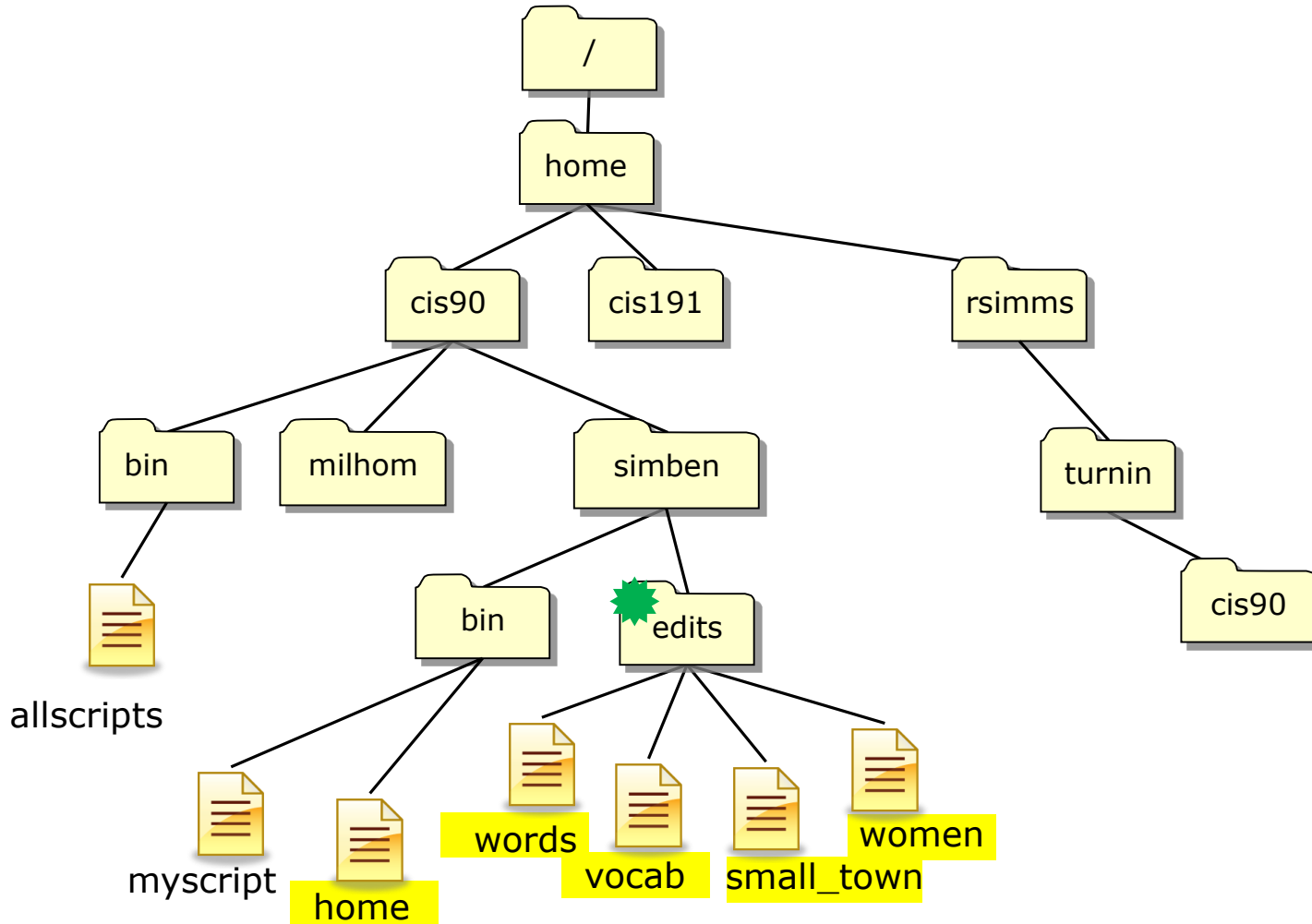
From  how could Benji concatenate the highlighted files into a file named lab09 in his home directory?


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

Another way



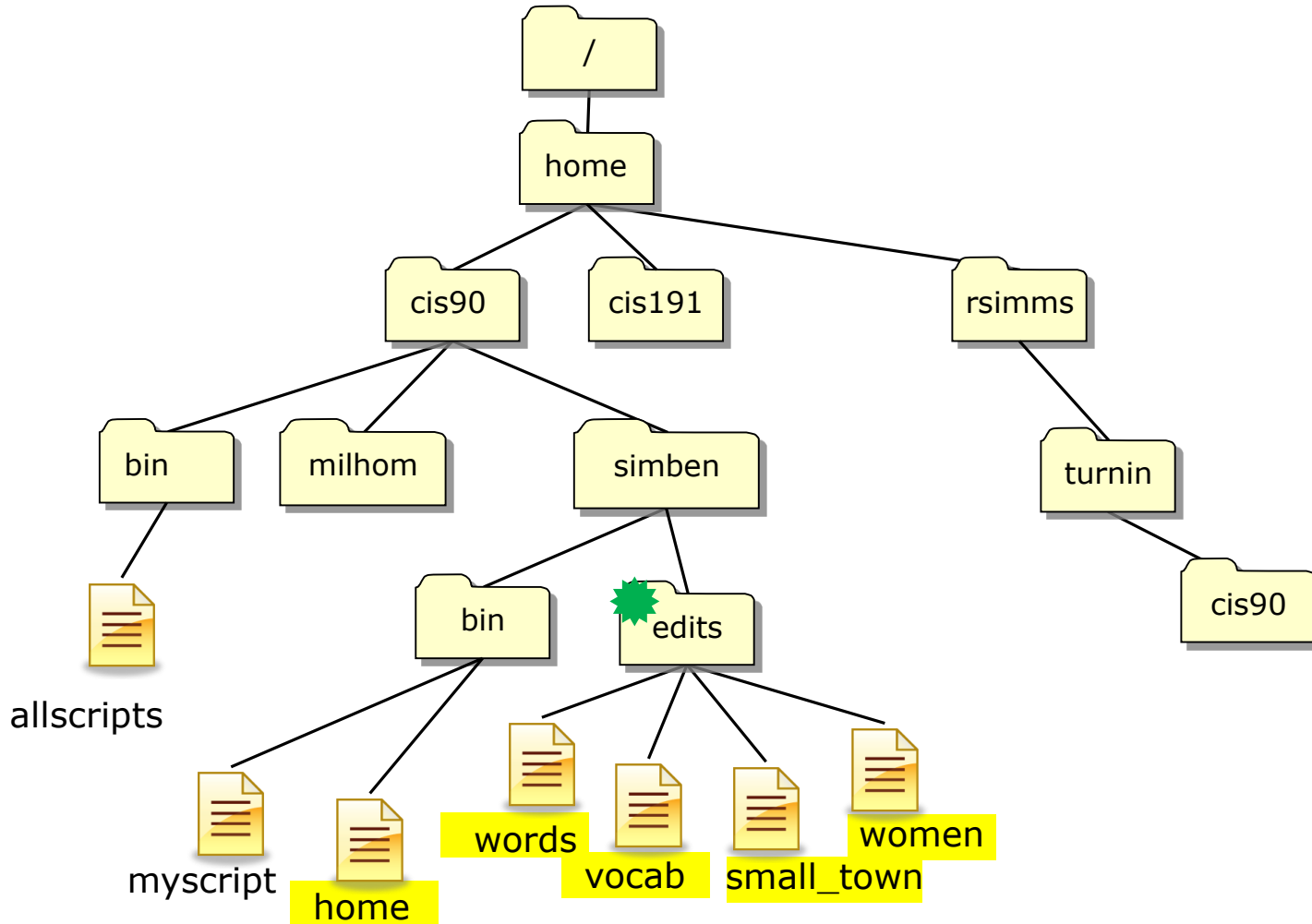
From  how could Benji concatenate the highlighted files into a file named lab09 in his home directory?




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
```

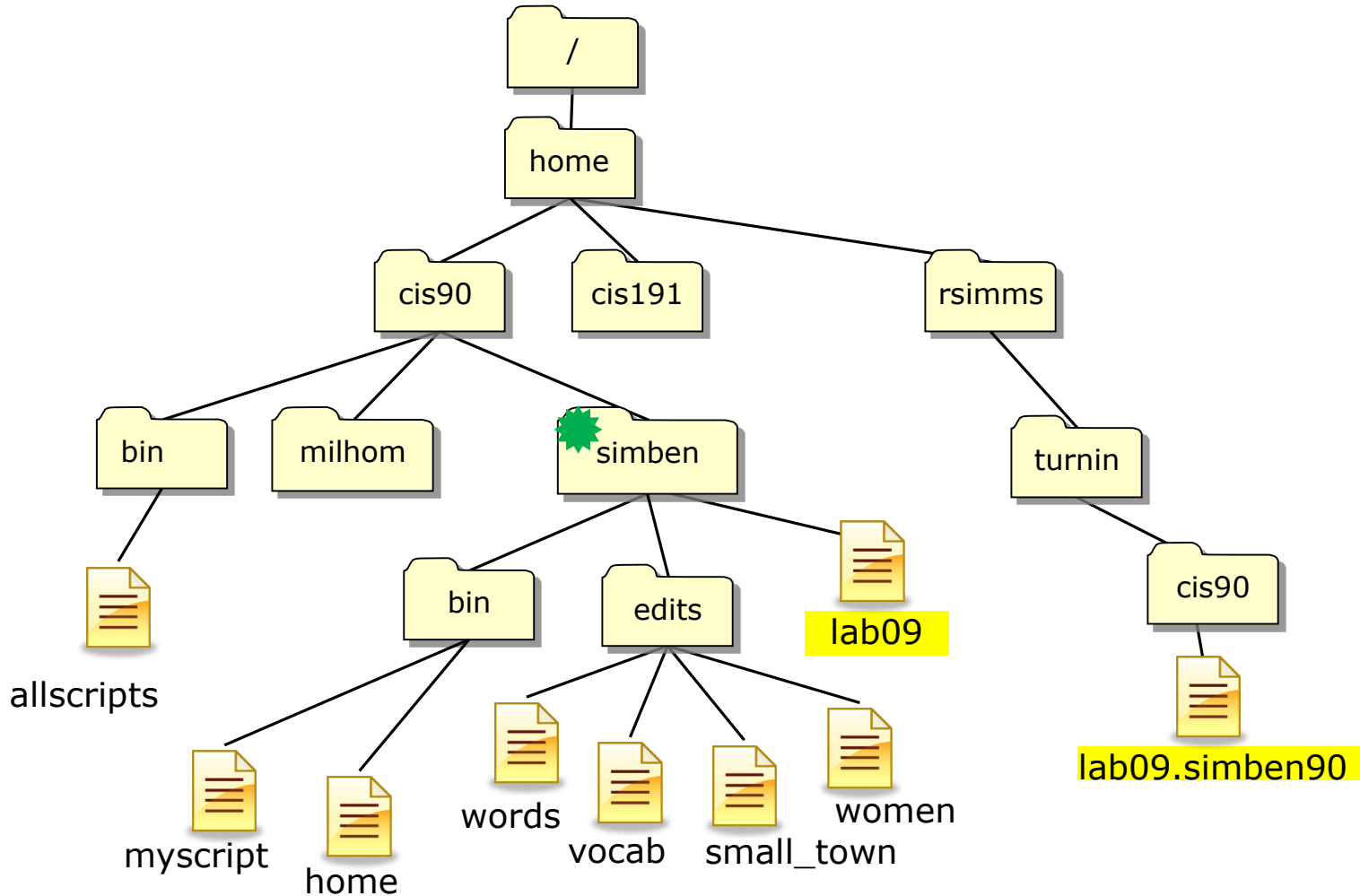

Or



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
```

Then



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  
Benji  
Soquel
```

Question: How can we add Benji and 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 /usr/bin/spell

Where is it on our path?

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

/usr/bin/spell: POSIX shell script, ASCII text executable

So what kind of file is it?

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

#!/bin/sh

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

aspell list mimicks the standard unix spell program, roughly.

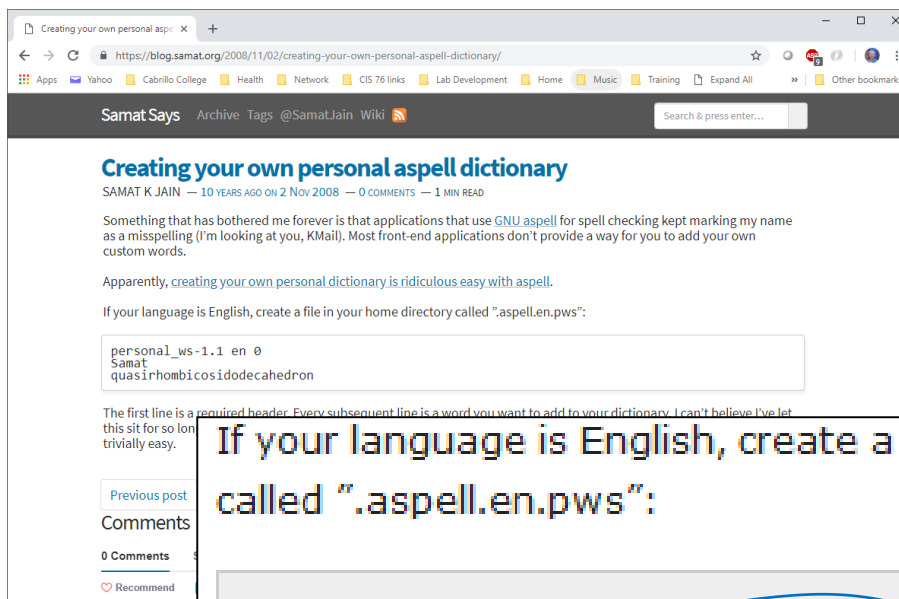
```
cat "$@" | aspell list --mode=none | sort -u
```

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

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
```

Add this line to the top

```
Samat
```

```
quasirhombicosidodecahedron
```

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 Benji >> .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 Benji and 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

Mashpit and Ayshire

Forum rules
Do not do such other and please don't post any username or password information!


POSTREPLY: Search this topic Search

Mashpit and Ayshire
[By Mare Ceudill • Sat Nov 16, 2013 11:04 am
Are we supposed to leave mashpitt and Ayshire misspelled?
Why? Does it add to the country charm of the prose?
This is a mashpitt:



Were people drooge their biscuits on Thanksgiving.

This is a moshpitt:



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

Here is some projects fun where you can re-live your youth and make your own moshpitt:
<http://mattblarbaum.github.io/moshpitts.js/>

Ayshire is a small town in Iowa.
www.ia.gov

Mare Ceudill
Posts: 49
Joined: Tue Nov 03, 2013 11:09 am

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

Only after you finish Lab 9

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

Note: Please leave just 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
```

The script named home that you created in Lab 9

1) Running the script fails when it's in your edits/ directory

```
/home/cis90/simben $ ls -l edits/home
-rwxrwxr-x. 1 simben90 cis90 104 Apr 23 16:49 edits/home
/home/cis90/simben $ home
-bash: home: command not found
```

2) Running the script succeeds when it's in your bin/ directory

```
/home/cis90/simben $ mv edits/home bin
/home/cis90/simben $ ls -l bin/home
-rwxrwxr-x. 1 simben90 cis90 104 Apr 23 16:49 bin/home
/home/cis90/simben $ home
This is the home directory of simben90
=====
a                edits/          myletter
accounts@       errors         names
< snipped >
```

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

Put your answer in the chat window

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

- 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/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/cis90/sim  
ben/../../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 11:59PM tonight.
2. Read your mail on Opus-II to verify your submission was both complete and received for grading.
3. Use **check9** to check your work.
4. Five more 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.

Housekeeping

Last Withdraw Date
This Saturday

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

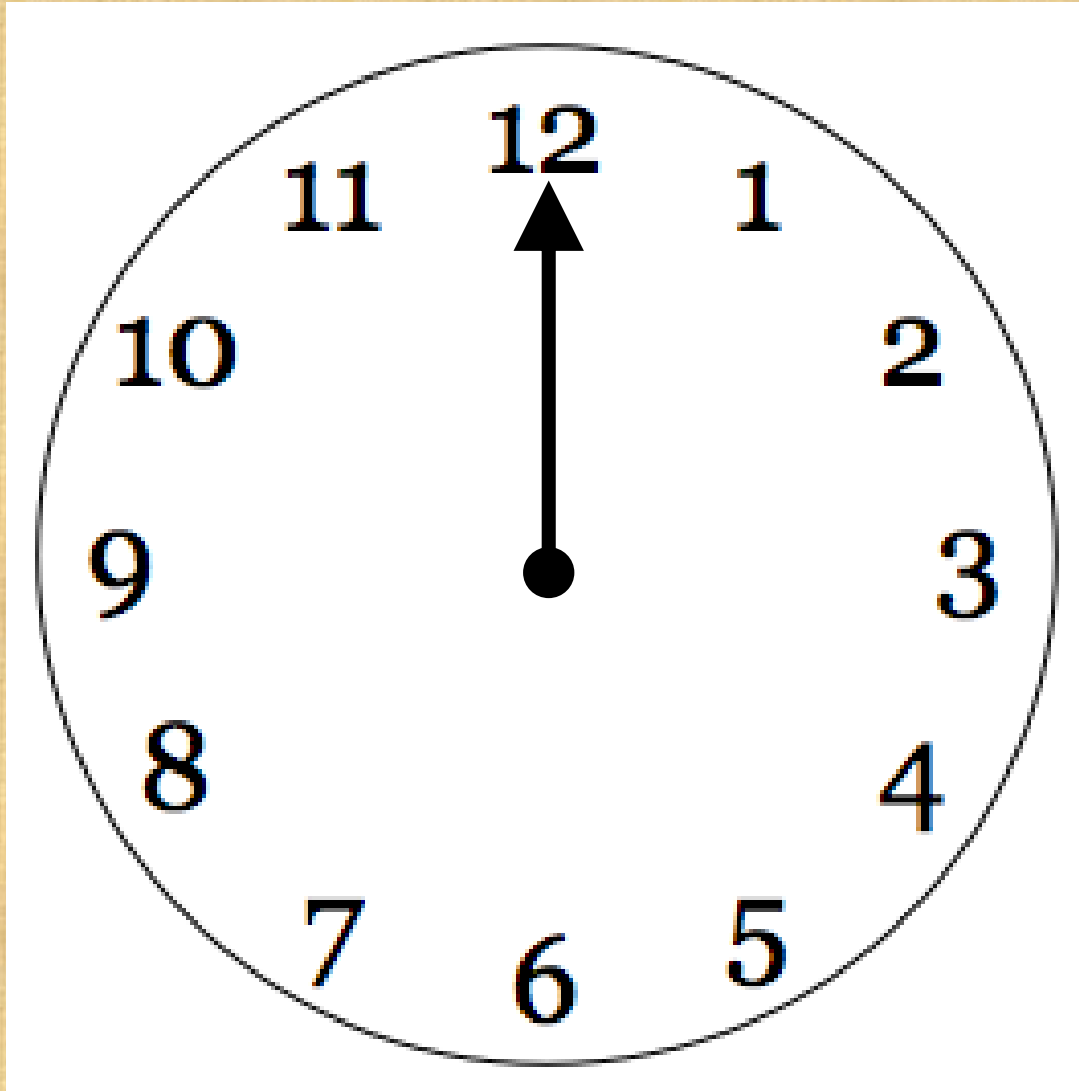
What day of the week is our final exam (Test #3)?

Su Mo Tu We Th Fr Sa

Use Zoom annotations to indicate the correct day



What time does our final exam (Test #3) start?



Use Zoom annotations to add the little hand of the clock

Heads up on Final Exam

Test #3 (final exam) is **Wednesday May 22, 7-9:50AM**

Wed	5/22	Test #3 (the final exam)		5 posts Lab X1 Lab X2
		Time <ul style="list-style-type: none"> WEDNESDAY 7:00AM - 9:50AM in Room 828 or online Materials <ul style="list-style-type: none"> Presentation slides (download) Test (canvas) ConferZoom <ul style="list-style-type: none"> Enter virtual classroom Class archives 		

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

- All students will take the test at the same time. The test starts at **7:00AM** must be completed by **9:50AM**.
- Working and long distance students can take the test online via ConferZoom 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)

**SPRING 2019 FINAL EXAMINATIONS SCHEDULE
MAY 20 TO MAY 25**

DAYTIME FINAL SCHEDULE

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 the Division Dean.**

STARTING CLASS TIME / DAY(S)	EXAM HOUR	EXAM DATE
<i>Classes starting between:</i>		
6:30 am and 8:55 am, MW/Daily	7:00 am-9:50 am	Monday, May 20
9:00 am and 10:15 am, MW/Daily	7:00 am-9:50 am	Wednesday, May 22

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
1	W	9:00AM-12:05PM	3.00	R.Simms	OL
Section 1 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 .					
2	W	9:00AM-12:05PM	3.00	R.Simms	828
&	Arr.	Arr.		R.Simms	OL
Section 2 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 .					

Optional "30-second" Test #2 Workshop

April 28 30-second T2 workshop

by **Rich Simms** » Fri Apr 19, 2019 6:14 am



Optional workshop for anyone that wants to do any of the the Test #2 questions in 30 seconds or less.

Date: Sunday, April 28

Time: 2:00 PM

Place: <https://cccconfer.zoom.us/j/426283384>

Mark your calendars if interested

- Rich



ONLINE

Rich Simms
Site Admin

Posts: 573

Joined: Sat Aug 05, 2017 12:07 pm

Contact:

See announcement post on the forum

Send ourselves some reminders for the workshop

```
cat ../depot/event
```

```
cat ../depot/reminder
```

```
cp -v ../depot/reminder .
```

```
vi reminder
```

(edit and replace "your email address here" with your actual email address)

```
cat reminder
```

```
at now + 1 minute < reminder
```

```
at 8pm sat < reminder
```

```
at 10am sun < reminder
```

```
at 1pm sun < reminder
```

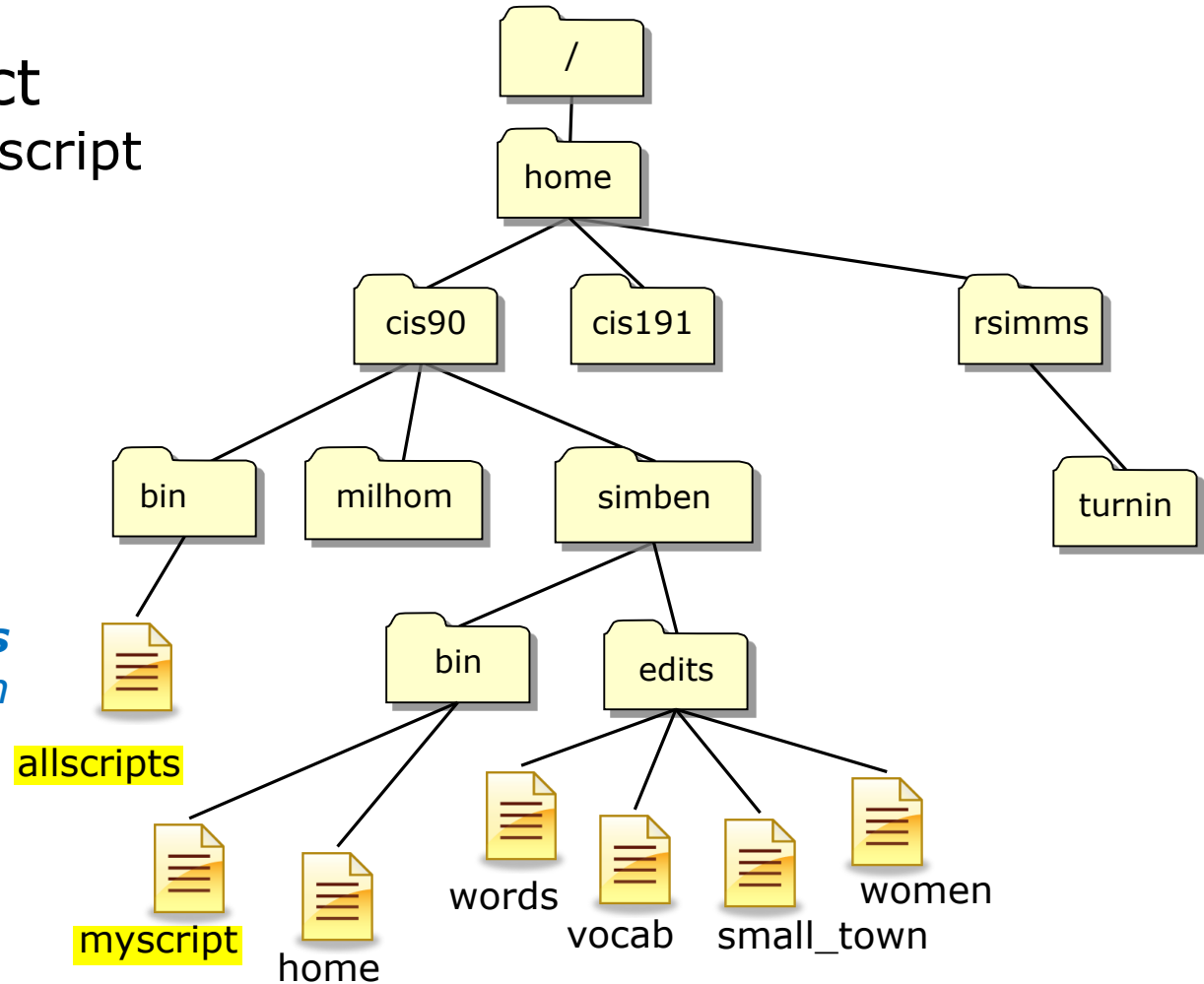
```
atq
```



final project *preview*

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
  
```

```
cat ../bin/allscripts
```

```
simben90@opus-ii:~  
#!/bin/bash  
#  
# menu: A simple menu template  
#  
while true  
do  
  clear  
  echo -n "  
*****  
*           Spring 2019 CIS 90 Online Projects           *  
*****  
1) Adina  
2) Benji  
3) Cheryl  
4) Cody  
5) Cole  
6) Daniel  
7) Danny  
8) David  
9) Duke  
10) Erik  
11) Evie  
12) Homer  
13) Jim  
14) Jon  
15) Joseph  
16) Lucky  
17) Mark  
18) Matt  
19) Nick  
20) Ohunayo  
21) Ryan  
22) Scott  
23) Sequoia  
24) Shane  
25) Tanisha  
26) Wais  
  
99) Exit  
  
Enter Your Choice: "  
read RESPONSE
```

The **allscripts** bash script

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


```
cat /home/cis90/bin/allscripts
```

```
read response
case $response in
```

```
1) # Benji
2) # Benji
3) # Benji
4) # Benji
5) # Benji
6) # Benji
7) # Benji
8) # Benji
9) # Benji
10) # Benji
11) # Benji
12) # Benji
13) # Benji
14) # Benji
15) # Benji
16) # Benji
17) # Benji
18) # Benji
19) # Benji
20) # Benji
21) # Benji
22) # Benji
23) # Benji
24) # Benji
25) # Benji
26) # Benji
27) # Benji
28) # Benji
29) # Benji
30) # Benji
31) # Benji
32) # Benji
33) # Benji
34) # Benji
35) # Benji
36) # Benji
37) # Benji
38) # Benji
39) # Benji
40) # Benji
41) # Benji
42) # Benji
43) # Benji
44) # Benji
45) # Benji
46) # Benji
47) # Benji
48) # Benji
49) # Benji
50) # Benji
51) # Benji
52) # Benji
53) # Benji
54) # Benji
55) # Benji
56) # Benji
57) # Benji
58) # Benji
59) # Benji
60) # Benji
61) # Benji
62) # Benji
63) # Benji
64) # Benji
65) # Benji
66) # Benji
67) # Benji
68) # Benji
69) # Benji
70) # Benji
71) # Benji
72) # Benji
73) # Benji
74) # Benji
75) # Benji
76) # Benji
77) # Benji
78) # Benji
79) # Benji
80) # Benji
81) # Benji
82) # Benji
83) # Benji
84) # Benji
85) # Benji
86) # Benji
87) # Benji
88) # Benji
89) # Benji
90) # Benji
91) # Benji
92) # Benji
93) # Benji
94) # Benji
95) # Benji
96) # Benji
97) # Benji
98) # Benji
99) # Benji
100) # Benji
```

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

```
1) # Benji
2) # Benji
3) # Benji
4) # Benji
5) # Benji
6) # Benji
7) # Benji
8) # Benji
9) # Benji
10) # Benji
11) # Benji
12) # Benji
13) # Benji
14) # Benji
15) # Benji
16) # Benji
17) # Benji
18) # Benji
19) # Benji
20) # Benji
21) # Benji
22) # Benji
23) # Benji
24) # Benji
25) # Benji
26) # Benji
27) # Benji
28) # Benji
29) # Benji
30) # Benji
31) # Benji
32) # Benji
33) # Benji
34) # Benji
35) # Benji
36) # Benji
37) # Benji
38) # Benji
39) # Benji
40) # Benji
41) # Benji
42) # Benji
43) # Benji
44) # Benji
45) # Benji
46) # Benji
47) # Benji
48) # Benji
49) # Benji
50) # Benji
51) # Benji
52) # Benji
53) # Benji
54) # Benji
55) # Benji
56) # Benji
57) # Benji
58) # Benji
59) # Benji
60) # Benji
61) # Benji
62) # Benji
63) # Benji
64) # Benji
65) # Benji
66) # Benji
67) # Benji
68) # Benji
69) # Benji
70) # Benji
71) # Benji
72) # Benji
73) # Benji
74) # Benji
75) # Benji
76) # Benji
77) # Benji
78) # Benji
79) # Benji
80) # Benji
81) # Benji
82) # Benji
83) # Benji
84) # Benji
85) # Benji
86) # Benji
87) # Benji
88) # Benji
89) # Benji
90) # Benji
91) # Benji
92) # Benji
93) # Benji
94) # Benji
95) # Benji
96) # Benji
97) # Benji
98) # Benji
99) # Benji
100) # Benji
```

```
esac
```

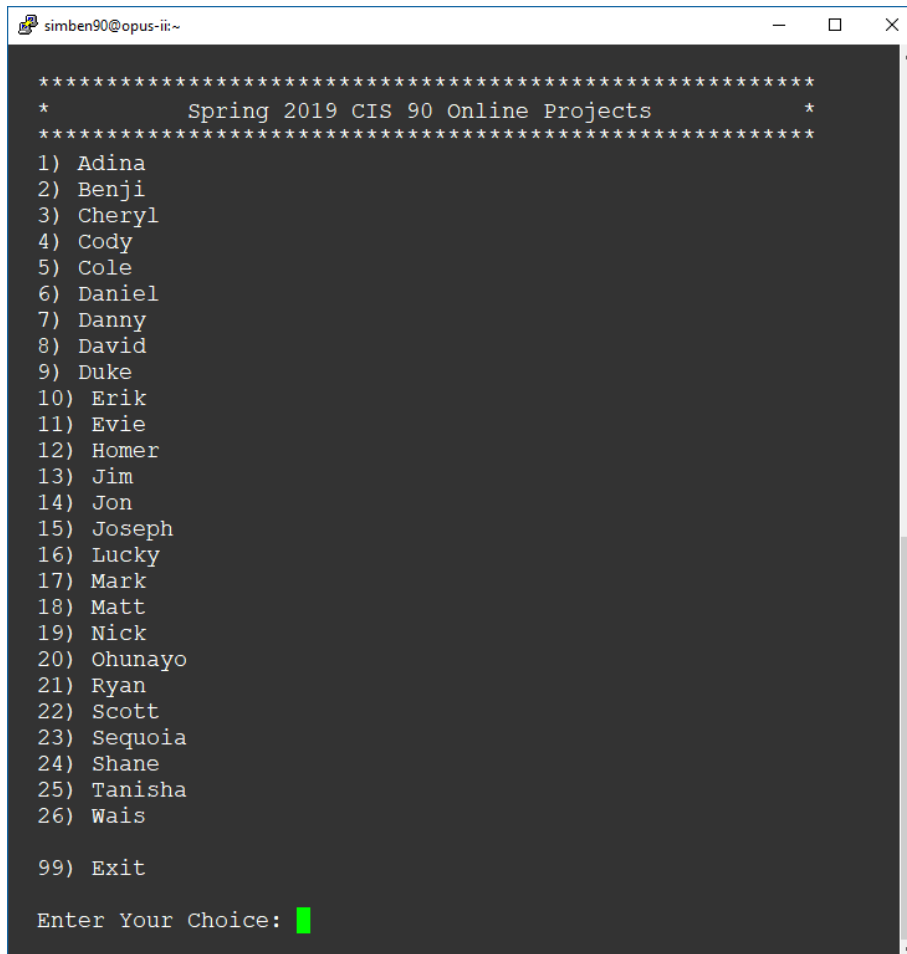
The **allscripts** bash script

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.

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

The **allscripts** bash script

Running **allscripts** looks like this



```
simben90@opus-ii:~  
*****  
*           Spring 2019 CIS 90 Online Projects           *  
*****  
1) Adina  
2) Benji  
3) Cheryl  
4) Cody  
5) Cole  
6) Daniel  
7) Danny  
8) David  
9) Duke  
10) Erik  
11) Evie  
12) Homer  
13) Jim  
14) Jon  
15) Joseph  
16) Lucky  
17) Mark  
18) Matt  
19) Nick  
20) Ohunayo  
21) Ryan  
22) Scott  
23) Sequoia  
24) Shane  
25) Tanisha  
26) Wais  
  
99) Exit  
  
Enter Your Choice: █
```

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



`vi ~/bin/myscript`

```

simben90@opus-ii:~
#!/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 response
done
1,1 All
    
```

The **myscript** bash script

*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

Make your own copy of the *myscript* file

Getting Started

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

```
cd ~/bin
```

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

- 2) Give your script execute permissions with:

```
chmod +x myscript
```

- 3) Run the script:

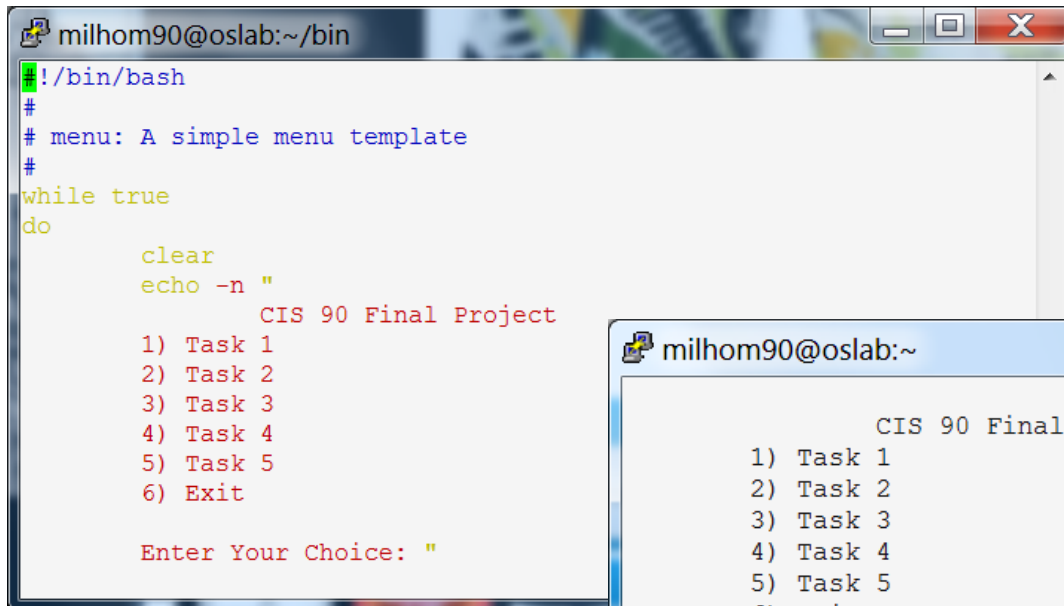
```
myscript
```

Indicate in the chat window if it works

Final Project

Testing you can run your myscript file

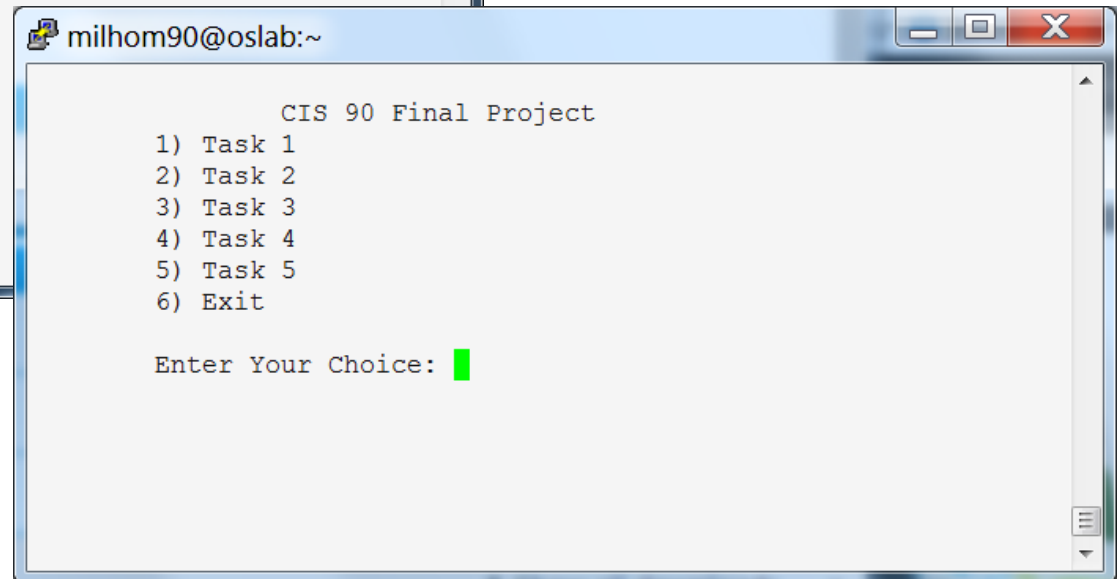
`vi myscript`



```
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: "
```

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: █
```

Viewing and then running your myscript file (after you add execute permissions)

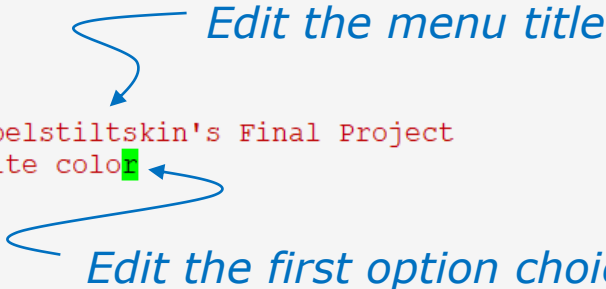
Final Project

Modifying your myscript file

`vi myscript`

```
milhom90@opus-ii:~/bin
#!/bin/bash
#
# menu: A simple menu template
#
while true
do
    clear
    echo -n "
        Rumpelstiltskin'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, 571C written
```



Modify the menu title and the name of the first task, save and exit

myscript

Run myscript and test your modifications

```
milhom90@opus-ii:~/bin
        Rumpelstiltskin's Final Project
    1) My favorite color
    2) Task 2
    3) Task 3
    4) Task 4
    5) Task 5
    6) Exit

    Enter Your Choice: █
```


Final Project

Testing a default task

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"
      ;;
esac
```

View the first task

myscript

```
milhom90@opus-ii:~/bin
Rumpelstiltskin'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
```

*Running Task 1 doesn't
do anything yet*

Final Project

Making a simple task

vi myscript

```

milhom90@opus-ii:~/bin
Enter Your Choice: "
read response
case $response in
  1)    # Make favorite color banner
        echo -n "What is your first name? "
        read name
        echo -n "What is your favorite color? "
        read color
        banner $name likes $color
        ;;
  2)    # Commands for Task 2
        ;;
  3)    # Commands for Task 3
        ;;
-- INSERT --
16,1
55%
```

```

echo -n "What is your first name? "
read name
echo -n "What is your favorite color? "
read color
banner $name likes $color
```

Indicate in the chat window when you have finished modifying your myscript file.

Final Project

Making a sample task

myscript

```

milhom90@opus-ii-~/bin
Rumpelstiltskin'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 first name? Benji
What is your favorite color? Blue

#####          ##### # # # # #
# # # # # # # # # # # # #
# # # # # # # # # # #
#####          ##### # # # # #
# # # # # # # # # # # # #
# # # # # # # # # # #
#####          ##### # # # # #
# # # # # # # # # # # # #
# # # # # # # # # # #
# # # # # # # # # # #
#####          ##### # # # # #

##### # # # # #
# # # # # # #
# # # # # # #
##### # # # # #
# # # # # # #
# # # # # # #
##### # # # # #

Hit the Enter key to return to menu █

```

Indicate in the chat window if your sample script works or not.

If it doesn't we will debug it.

Final Project Getting Started

A new command

read, inputs text from stdin and stores it in the variable specified as an argument.

```
read response
case $response in
  1) # Make favorite color banner
     echo -n "What is your first name? "
     read name
     echo -n "What is your favorite color? "
     read color
     banner $name likes $color
     ;;
```

Another new command

case, allows different branches of code to be executed based on the value of the variable specified as an argument.

Final Project Getting Started

The case statement begins here

```

read response
case $response in
  1) [ # Make favorite color banner
      echo -n "What is your first name? "
      read name
      echo -n "What is your favorite color? "
      read color
      banner $name likes $color
      ;;

```

First case ends here

If the user enters a "1" then these lines of script will be executed

First case of case statement starts here

Final Project Getting Started

```

read response
case $response in
  1) # Make favorite color banner
     echo -n "What is your first name? "
     read name
     echo -n "What is your favorite color? "
     read color
     banner $name likes $color
  ;;

```

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

another variable

another variable

Variables (\$ means "the value of")

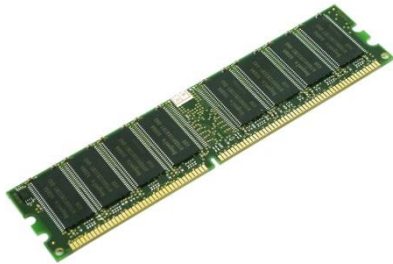
Final Project Getting Started

```
read response
case $response in
  1) # Make favorite color banner
     echo -n "What is your first name? "
     read name
     echo -n "What is your favorite color? "
     read color
     banner $name likes $color
     ;;
```

Comments begin with a # and are used to document script code.

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 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 INPUTRC LS_COLORS CVS_RSH
 PROMPT_COMMAND BASH_SOURCE _ MACHTYPE
 DIRSTACK MAIL SSH_ASKPASS PS2
 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
HISTSIZ=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:*.bat=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
```

```
BASH=/bin/bash
BASH_ARGC=()
BASH_ARGV=()
BASH_ENV=/home/cis90/simben/.bashrc
BASH_LINENO=()
BASH_SOURCE=()
BASH_VERSION=3.2.25(1)-release
BASH_VERBOSE[0]=3 [1]=2 [2]=25 [3]=1 [4]=release [5]=1686-redhat-linux-gnu
COLORS=/etc/DIR_COLORS.xterm
COLORMG=84
CPS=80
CURL=ash
DIRSTACK=()
EUID=116
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:ow=01;05;37;41:mi=00;05;37;41:ex=00;32*.cmd=00;32*.exe=00;32*.com=00;32*.bat=00;32*.sh=00;32*.csh=00;32*.tar=00;31*.tgz=00;31*.arj=00;31*.taz=00;31*.zip=00;31*.zipp=00;31*.z=00;31*.gz=00;31*.bz2=00;31*.bz=00;31*.tar.gz=00;31*.cpio=00;31*.pgp=00;35*.gif=00;35*.mp=00;35*.xbm=00;35*.xpm=00;35*.png=00;35*.tif=00;35*'
MAIL=/var/spool/mail/simben
MAILCHECK=60
OLDPWD=/home/cis90/simben/edits
OPTERR=1
OPTIND=1
OPTPR=linux-gnu
PATH=/usr/kernels/bin:/usr/local/bin:/bin:/usr/bin:/home/cis90/simben/./bin:/home/cis90/simben/bin:
PFISTERSTATUS={0}="0"
PPID=754
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
SHLV=1
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
SSH_CLIENT="63.249.103.107 1909 22"
SSH_CONNECTION="63.249.103.107 1909 207.62.186.9 22"
SSH_TTY=/dev/pts/1
TERM=xterm
UID=116
USER=simben
USERMSG=
_="
```

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

```
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 $
```

Activity

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

Activity

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
```

*Run: **stat -c %a 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 PS4 **BASH_ENV** SHELLOPTS **HOSTNAME**
 HISTFILESIZE **TERM** PIPESTATUS GROUPS
 BASH_ARGV **PATH** UID BASH_VERSINFO
SHLVL tmpid **SSH_CONNECTION** PS1
 BASH_ARGC **USERNAME** OSTYPE HISTFILE
HISTSIZ BASH_LINENO **LESSOPEN**
 HOSTTYPE OPTERR **SSH_CLIENT**
 COLUMNS **LS_COLORS** **CVS_RSH**
 PROMPT_COMMAND **INPUTRC** BASH_SOURCE _ MACHTYPE
 DIRSTACK **MAIL** **SSH_ASKPASS** PS2
 G_BROKEN_FILENAMES

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:*.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. Also using the **export** command with no arguments will show all the environment (exported) 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 changes an environment (exported) variable back to a normal shell variable. E.g. **export -n dog** makes dog back into a regular shell variable.

Child processes get copies of the parent's exported variables.

Any changes made by the child to these variables will not affect the parent's variables.

Environment (exported) Variables

Create, export and view

*Using **env** to show environment (exported) variables*

1

```
/home/cis90/simben $ env | wc -l  
26
```

*There are currently 26
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  
27
```

*Now there are 27
environment variables*

4

```
[simben@opus ~]$ env | grep fan  
fan=medium
```

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

```
[simben@opus ~]$ set | grep fan  
fan=medium
```

*use grep to show fan is a
shell variable*

Environment (exported) Variables

Create, export and view

Using export to create and show exported (environment) variables

1

```
/home/cis90/simben $ export | wc -l  
26
```

*There are currently 26
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 $ export | wc -l  
27
```

*Now there are 27
environment variables*

4

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

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

```
[simben@opus ~]$ set | grep fan  
fan=medium
```

*use grep to show fan is a
shell variable*

Activity

Recreate the variable named fan:

fan=high

Show that fan is now one of your shell variables:

set | grep fan

Show that fan has not been exported:

env | grep fan

Now export fan and show it:

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 ':'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.

Activity

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!

Activity

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

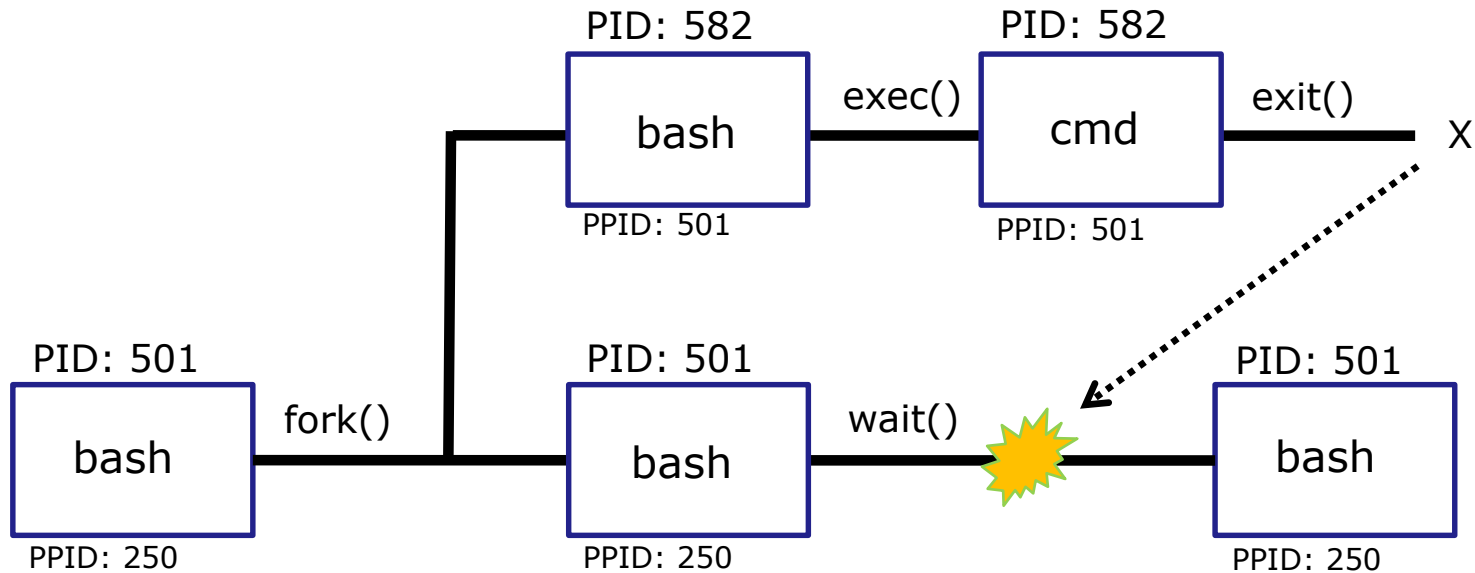


Variables and child processes

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

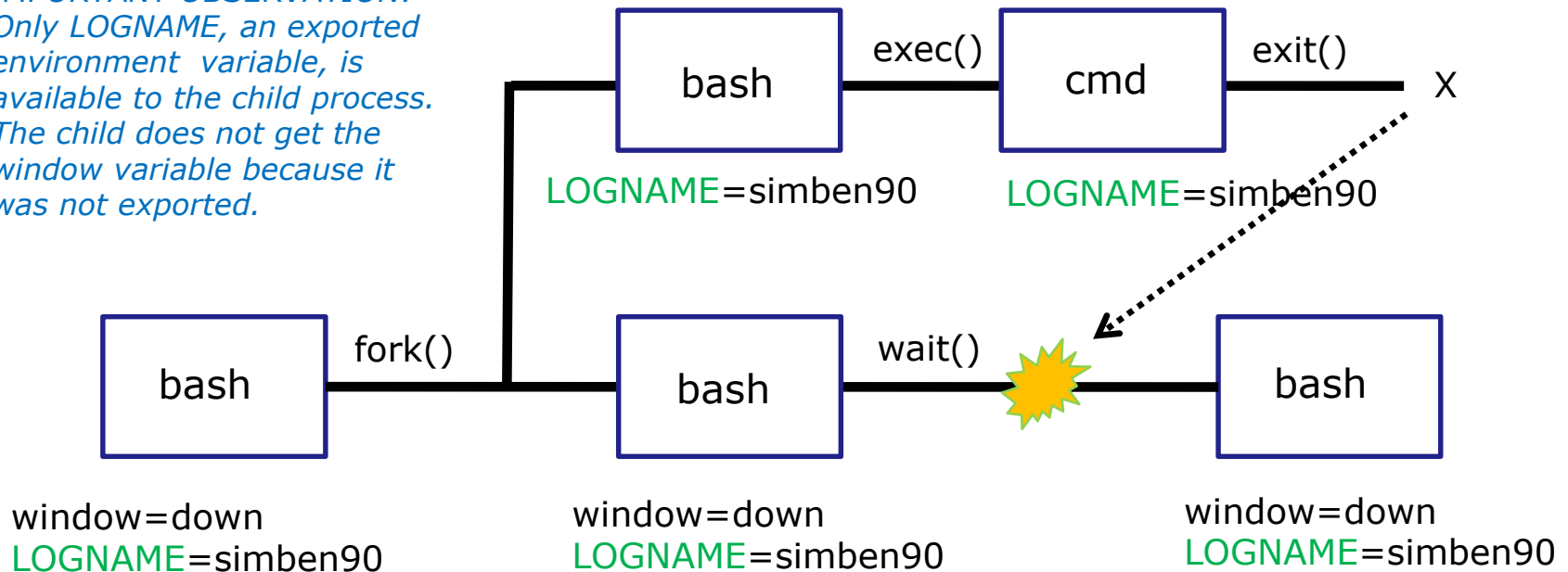
- 1 *parent*
- ```
/home/cis90/simben $ window=down
/home/cis90/simben $ echo $window $LOGNAME
down simben90
```
- Create a new variable named window*
- 
- 2 *parent*
- ```
/home/cis90/simben $ env | grep window
/home/cis90/simben $ set | grep window
window=down

/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.

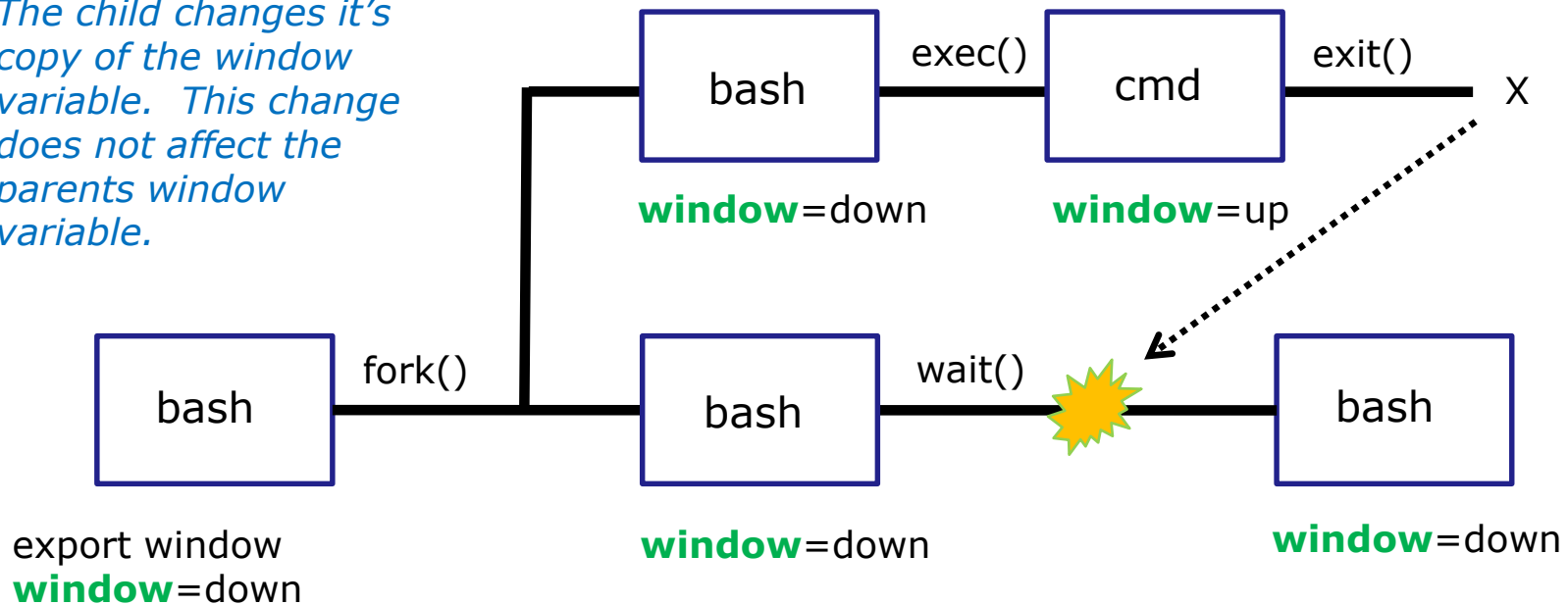


## Changes made by the child do not affect the parent

- |       |        |                                                                                                                              |                                                                                       |
|-------|--------|------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| 1     | parent | <pre> /home/cis90/simben \$ <b>echo \$window</b> down /home/cis90/simben \$ <b>export window</b> </pre>                      | <p><i>export window so it is available to children</i></p>                            |
| <hr/> |        |                                                                                                                              |                                                                                       |
| 2     | child  | <pre> /home/cis90/simben \$ <b>bash</b> [simben@opus ~]\$ <b>echo \$window</b> down </pre>                                   | <p><i>a copy of window is now available to the child process</i></p>                  |
| <hr/> |        |                                                                                                                              |                                                                                       |
| 3     | child  | <pre> [simben@opus ~]\$ <b>window=up</b> [simben@opus ~]\$ <b>echo \$window</b> up [simben@opus ~]\$ <b>exit</b> exit </pre> | <p><i>the child modifies the window variable</i></p>                                  |
| <hr/> |        |                                                                                                                              |                                                                                       |
| 4     | parent | <pre> /home/cis90/simben \$ <b>echo \$window</b> 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.

# Activity

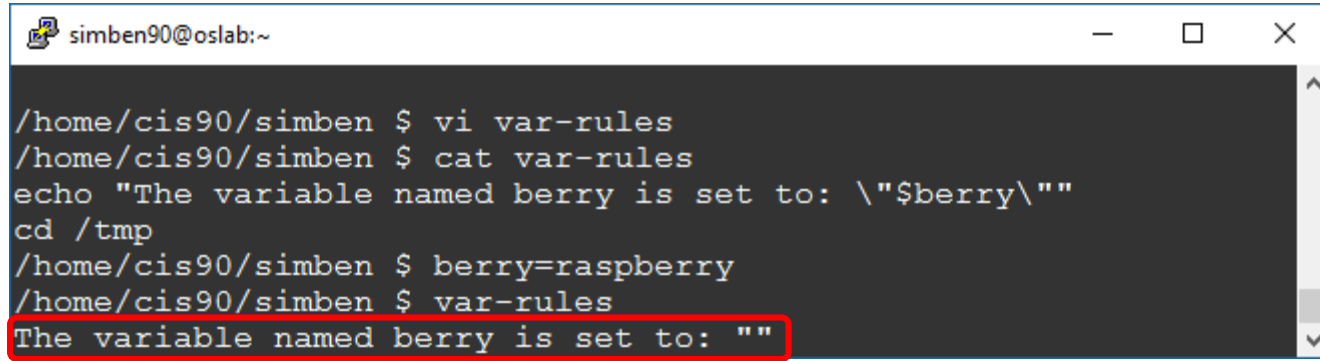
Look at the commands in this executable script:

```
/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: ""
```

*A child can only see variables the parent exports and berry was NOT exported!*

# Activity

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 $

```

*A child can only see variables the parent exported and berry was exported.*

# Activity

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 $

```

*A 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
```

*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 $ac $fan $defrost'`  
`$ alias p`

`alias p='echo on medium off'`

`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*



# Activity

## 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 prompt string



# .bash\_profile

(Red Hat family)

# .profile

(Debian family)

# .bash\_profile

User level environment and startup programs

- The *.bash\_profile* file is a shell script that sets up a specific user's shell environment.
- This script is executed each time the user logs in.
- The *.bash\_profile* script is used for customizing a user's environment and running custom startup programs.
- 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
```

```
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 002
```

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

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

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

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

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

*umask value  
is set*

*Terminal type is  
requested and  
set*

# .bashrc



# .bashrc

## User level functions and aliases

- 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.

*Note: Put user level environment stuff in .bash\_profile*

# .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"
```

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

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

# Activity

## 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
```

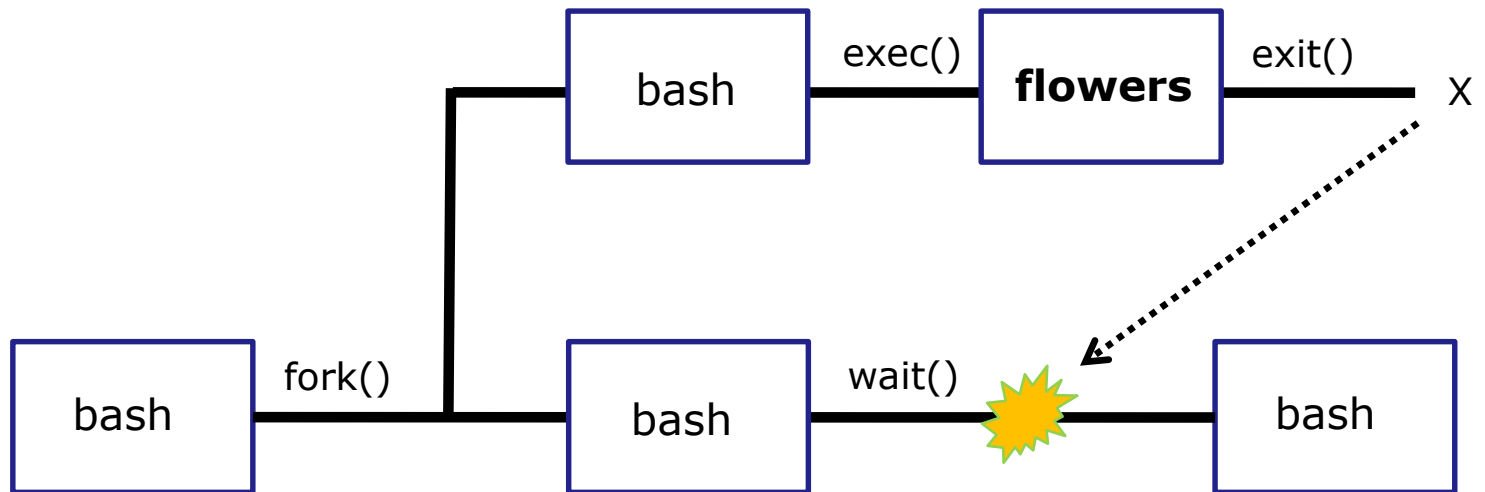
*You can copy and paste*

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

*The child does not see  
roses or violets*

```
/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*

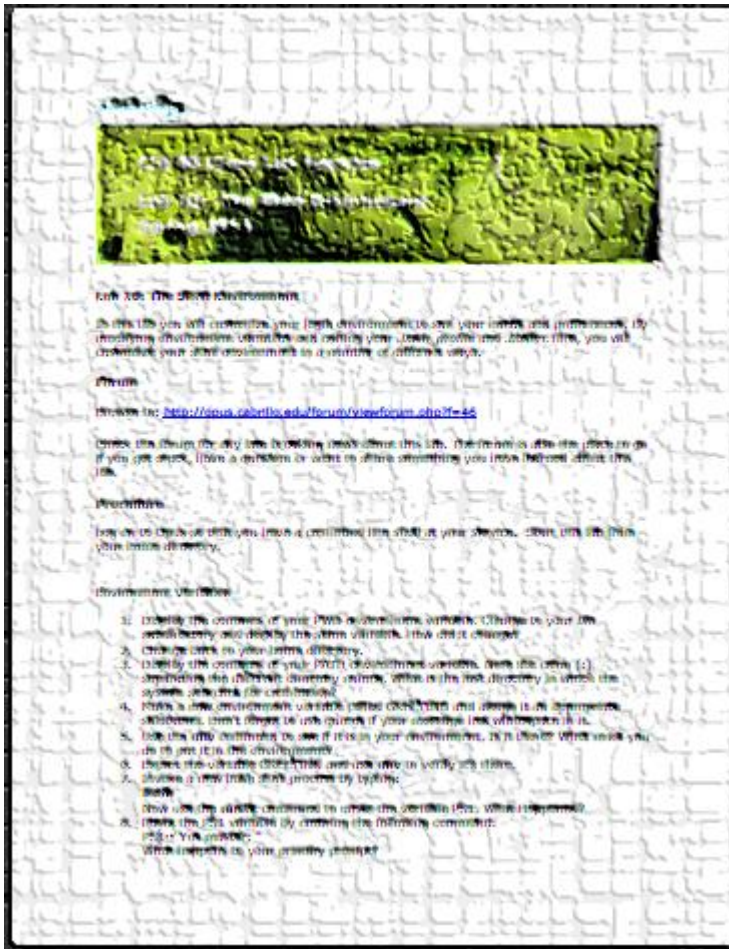


# Assignment





# 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.*




# 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](mailto: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?

End Meeting

End  
Meeting





# 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 image shows a terminal window titled 'simmsben@opus:~'. Inside the window, the vi editor is open to a file named '/tmp/RecVQYE2'. The editor's status line at the bottom shows '7L, 141C'. The content of the file is a text message:

```
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
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

*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

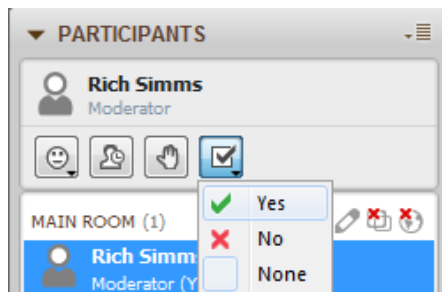
Try it!

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**