Rich's lesson module checklist

- ❑ Slides, Project, Lab X1 and Lab X2 posted
- ❑ WB converted from PowerPoint
- ❑ Print out agenda slide and annotate page numbers
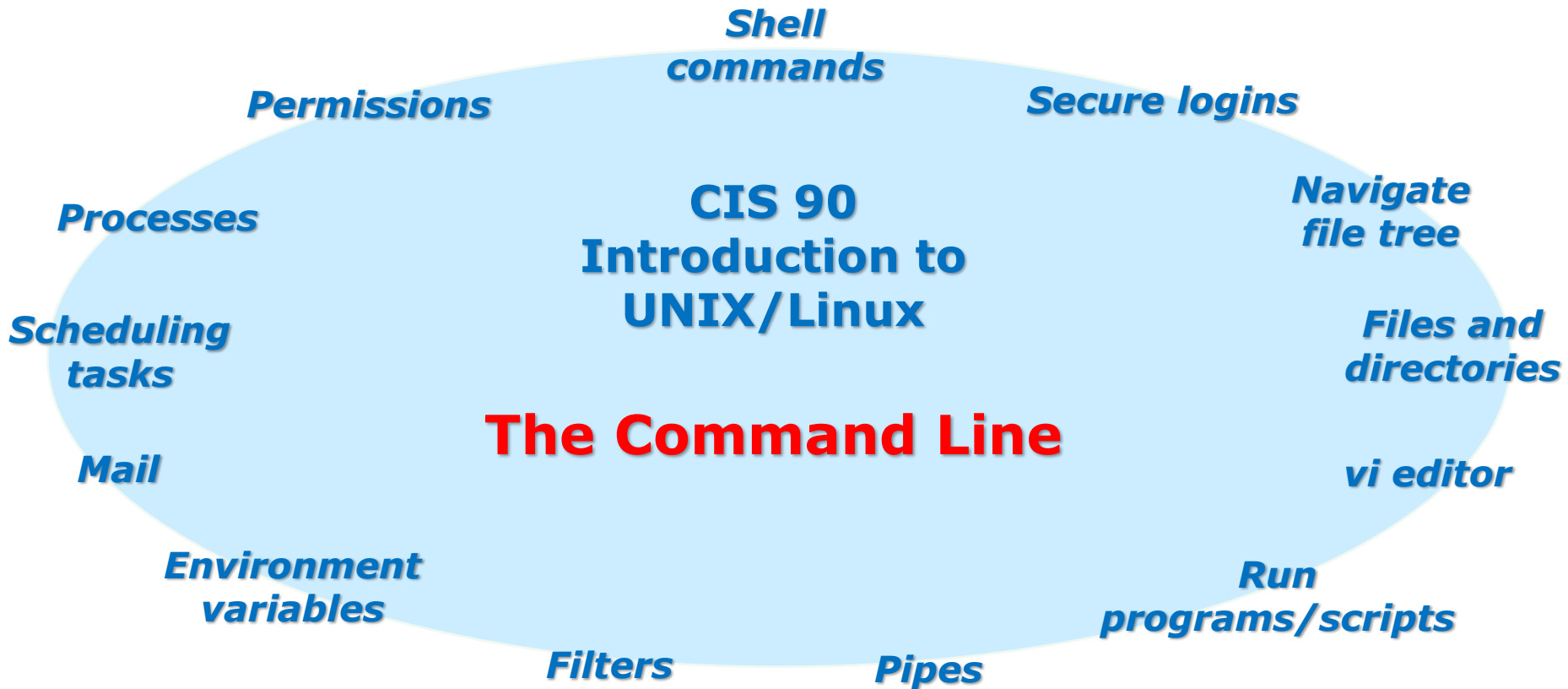
- ❑ Flash cards
- ❑ Page numbers
- ❑ 1st minute quiz
- ❑ Web Calendar summary
- ❑ Web book pages
- ❑ Commands

- ❑ CUPS & printer demo equipment (optional)
- ❑ Lab X1 and X2 posted
- ❑ Lock turnin directory at midnight (scripts/schedule-submit-locks)

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

- ❑ Check CCC Confer and 3C Media videos

Last updated 11/18/2017

1

Shell commands

Permissions

Secure logins

CIS 90 Introduction to UNIX/Linux

Navigate file tree

Processes

Scheduling tasks

Files and directories

The Command Line

Mail

vi editor

Environment variables

Run programs/scripts

Filters

Pipes

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.

2

Introductions and Credits

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

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

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

Student checklist for attending class

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

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

4

Student checklist for suggested screen layout

- ❑ *Google*
- ❑ *CCC Confer*
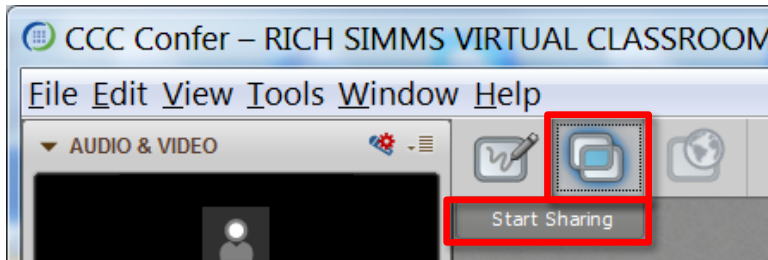- ❑ *Downloaded PDF of Lesson Slides*

- ❑ *CIS 90 website Calendar page*
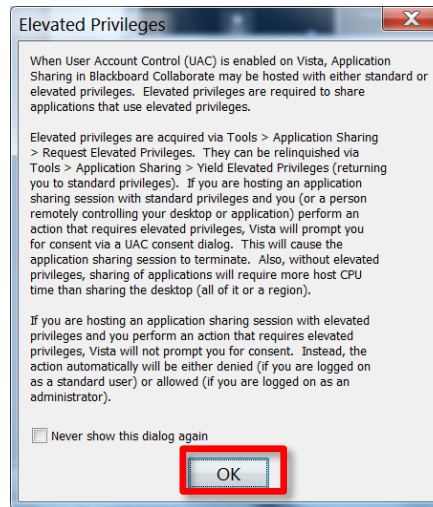- ❑ *One or more login sessions to Opus-II*

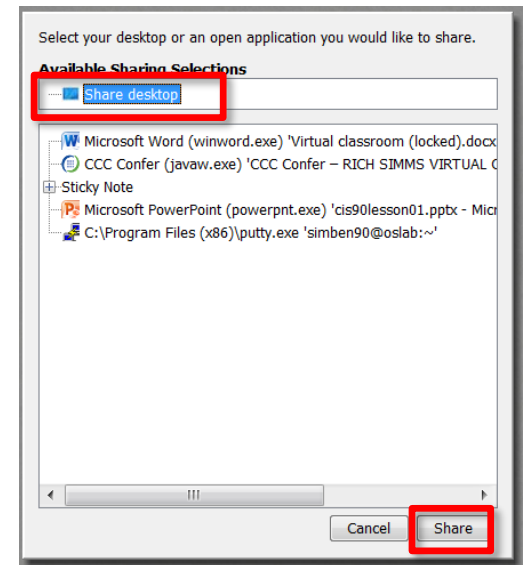# Student checklist for sharing desktop with classmates

1) Instructor gives you sharing privileges

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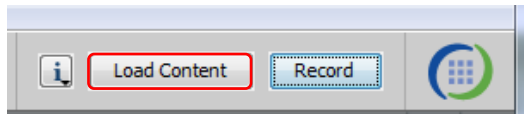
3) Click OK button.

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

Rich's CCC Confer checklist - setup

CCC Confer

[ ] Preload White Board

[ ] Connect session to Teleconference

Session now connected to teleconference

MAIN ROOM (2)

Rich Simms
Moderator (You)

Teleconference

Audio Setup Wizard...
Microphone Settings...
Speaker Settings...
Maximum Simultaneous Talkers...
Adjust Microphone Level Up
Adjust Microphone Level Down
Adjust Speaker Level Up
Adjust Speaker Level Down
Configure Telephone Conference...
Connect Session To Teleconference...
Camera Settings...
Maximum Simultaneous Cameras...
Make Video Follow Moderator Focus
Make Video Follow Speaker
Send Camera Snapshot To Whiteboard
Detach Panel

[ ] Is recording on?

Load Content   Recording ●

Red dot means recording

[ ] Use teleconferencing, not mic

Should be grayed out

AUDIO & VIDEO

Teleconference

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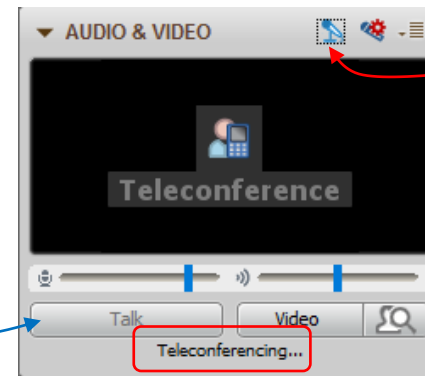
Should change from phone handset icon to little Microphone icon and the Teleconferencing … message displayed

7

Rich's CCC Confer checklist - screen layout

foxit for slides

chrome

putty

vSphere Client

[ ] layout and share apps

Rich's CCC Confer checklist - webcam setup
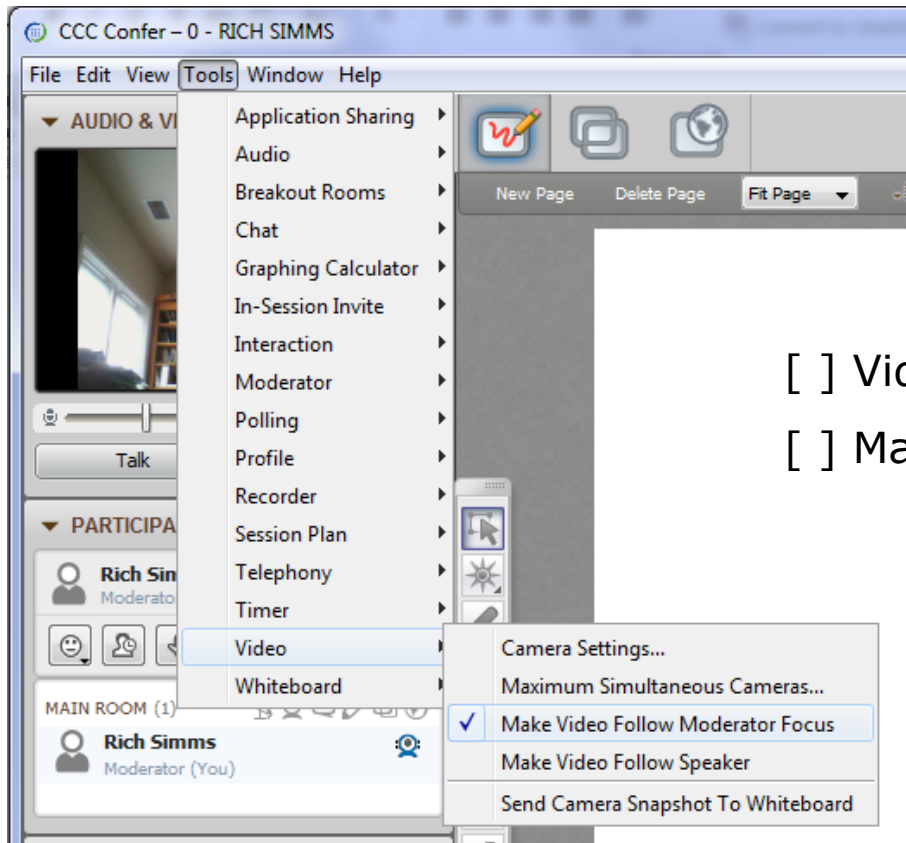
[ ] Video (webcam)

[ ] Make Video Follow Moderator Focus

Rich's CCC Confer checklist - Elmo
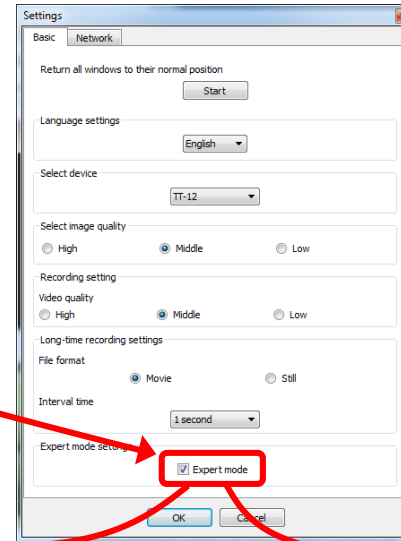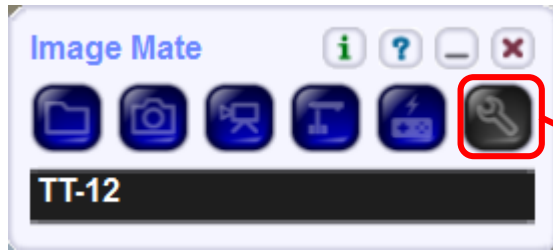
The "rotate image" button is necessary if you use both the side table and the white board.

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

Elmo rotated down to view side table

Rotate image button

Elmo rotated up to view white board

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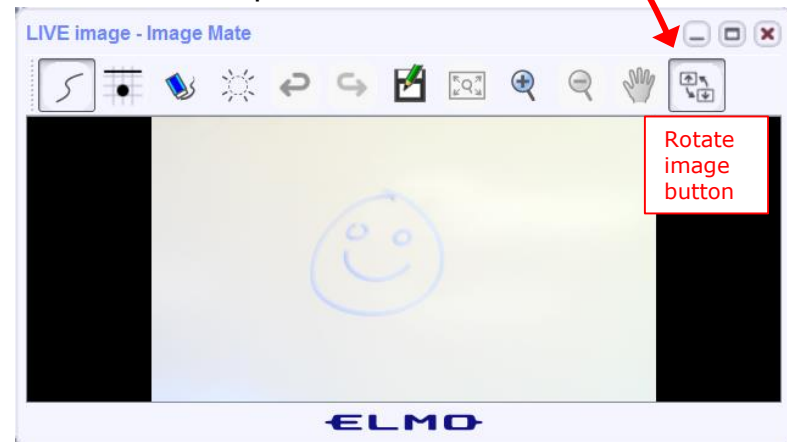
Run and share the Image Mate program just as you would any other app with CCC Confer

Rich's CCC Confer checklist - universal fixes

Universal Fix for CCC Confer:
1) Shrink (500 MB) and delete Java cache
2) Uninstall and reinstall latest Java runtime
3) http://www.cccconfer.org/support/technicalSupport.aspx

Control Panel (small icons)

General Tab > Settings…
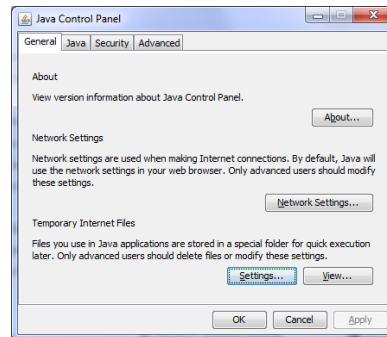
500MB cache size
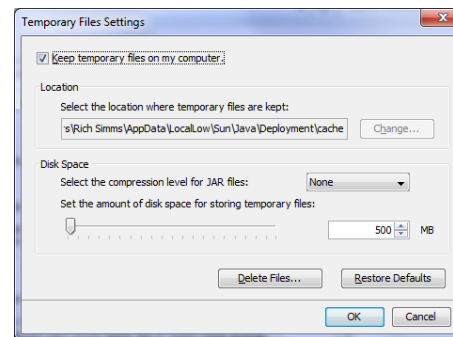
Delete these
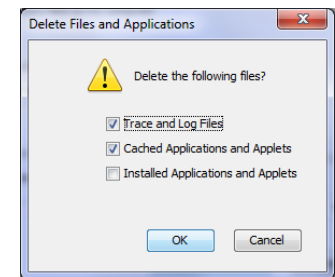
Google Java download
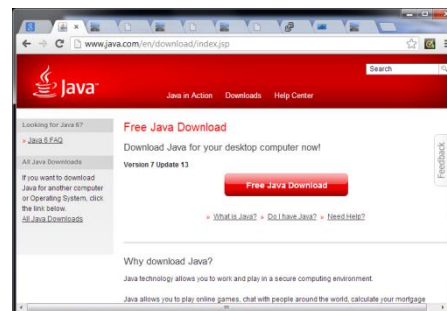
Rich's CCC Confer checklist - digital certificate work around
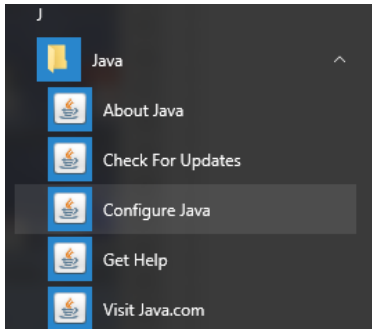
1. Open the   Java Control Panel
2. Select the **Security** tab
3. Select **Edit Site List...**
4. Select **Add**
5. Click into the white box next to the red exclamation mark and type
   https://na-downloads.elluminate.com
6. Press OK
7. Press **Continue** on the pop-up message
8. Press OK
9. Access your session or recording once more

12
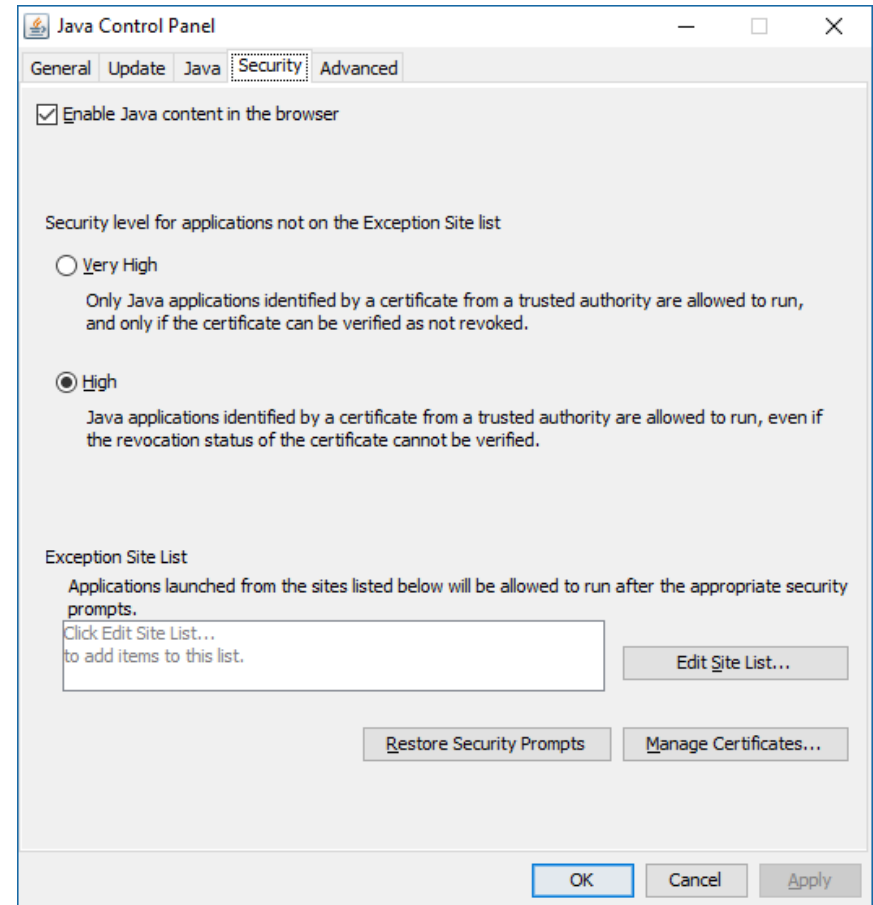
Rich's CCC Confer checklist - Putty Colors
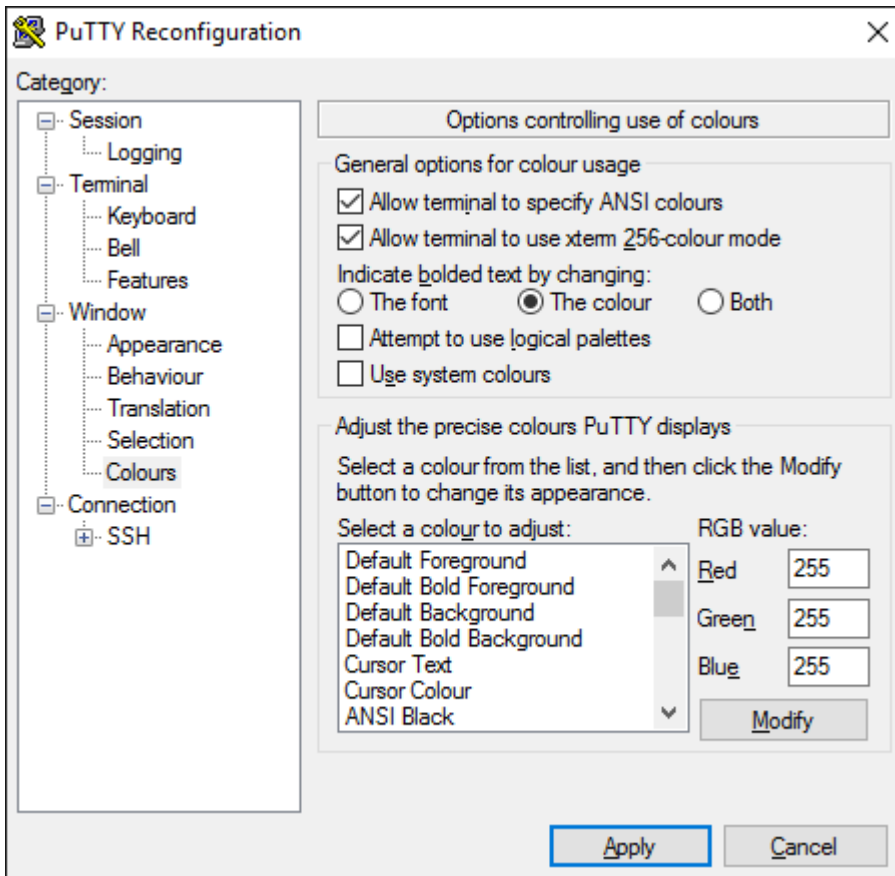
Putty Colors
Default Foreground 255 255 255
Default Bold Foreground 255 255 255
Default Background 51 51 51
Default Bold Background 255 2 85
Cursor Text 0 0 0
Cursor Color 0 255 0
ANSI Black 77 77 77
ANSI Black Bold 85 85 85
ANSI Red 187 0 0
ANSI Red Bold 255 85 85
ANSI Green 152 251 152
ANSI Green Bold 85 255 85
ANSI Yellow 240 230 140
ANSI Yellow Bold 255 255 85
ANSI Blue 205 133 63
ANSI Blue Bold 135 206 235
ANSI Magenta 255 222 173
ANSI Magenta Bold 255 85 255
ANSI Cyan 255 160 160
ANSI Cyan Bold 255 215 0
ANSI White 245 222 179
ANSI White Bold 255 255 255

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

13

Start

Sound Check

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

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

Cabrillo College
est. 1959

Instructor: **Rich Simms**
Dial-in: **888-886-3951**
Passcode: **136690**
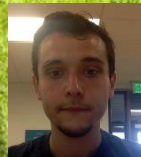
Marvin   William   Vinny   Hayden   Nick   Ramon   Nicholas   Manuel   Emmanuel
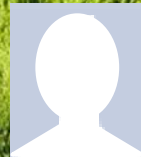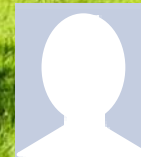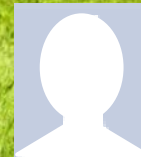
Oscar   Daniel P.   Jason   Brian   Vincent P.   Kyle   Sam X.   Jacobs   Tyler
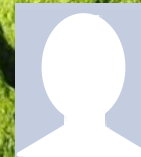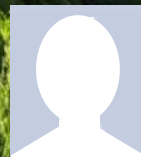
Sean   Alejandro   Moises   Joseph   David   Ben

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

First Minute Quiz

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

Use CCC Confer White Board

email answers to: risimms@cabrillo.edu

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

Shell Scripting and Printing

Objectives	Agenda
• Understand how to write a script and how they run. • Learn how to print and manage print jobs waiting to print.	• Quiz • Questions • Breaking things in Lab 10 • Extra Credit Answer • Lesson 12 review • Grok that? • Housekeeping • Shell scripting 101 • Final project myscript • Final project grading rubric • Final project permissions • Umask again! • Final project getting started • Final project forum tips • Scripting tips - echo • Tips on script names • Review how scripts are run • Printers • Printer configuration via CUPS • Printing in Linux • Managing print jobs • Assignment • Wrap up

18

Questions

Questions?

Lesson material?

Labs?   Tests?

How this course works?

· Graded work in home directories

· Answers in /home/cis90/answers

> *Who questions much, shall learn much, and retain much.*
> — Francis Bacon

> *If you don't ask, you don't get.*
> — Mahatma Gandhi

Chinese Proverb	他問一個問題，五分鐘是個傻子，他不問一個問題仍然是一個傻瓜永遠。
	He who asks a question is a fool for five minutes; he who does not ask a question remains a fool forever.

20

```
alias bill="cd /home/cis90/${LOGNAME%90}/poems/Shakespeare"
```

What the heck was this all about?

```
/home/cis90/milhom $ echo $LOGNAME
milhom90
/home/cis90/milhom $ echo ${#LOGNAME}          Length of the string
8
/home/cis90/milhom $ echo ${LOGNAME%90}         Extracts "90" from end of string
milhom
/home/cis90/milhom $ echo ${LOGNAME:3:3}        Substring extraction from
hom                                             position 3 length 3
/home/cis90/milhom $ echo ${LOGNAME#mil}
hom90                                           Extracts "mil" from front of string
```

For MANY MORE ways to manipulate strings Google "bash string manipulation"
or browse to **http://tldp.org/LDP/abs/html/string-manipulation.html**

21

Breaking things in Lab 10

The path (PATH) variable … a Review

- Lab 10 often results in clobbered paths and students may think some or all of the commands have disappeared!

- The path is a list of directories each containing commands, programs and scripts.

- The path is used by the shell to locate commands to run.

- The PATH variable defines the directories (separated by ":"s) and the search order.

- If your path gets clobbered it is still possible to run commands. However to do that you must specify the full absolute pathname. For example you can always run the **tty** command as follows:

```
/home/cis90/simben $ /usr/bin/tty
/dev/pts/0
```

The path (PATH) variable … a Review

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

1. Determine the 4th directory on the path above.

2. What is the name of the first command, in alphabetic order, found in this directory?

Put your answer in the chat window

25

Clobber your path on purpose

```
/home/cis90/simben $ oldpath=$PATH
/home/cis90/simben $ unset PATH
```
Backup your current path

```
/home/cis90/simben $ tty
-bash: tty: No such file or directory
```
The tty command can no longer be run by typing just it's name

```
/home/cis90/simben $ /usr/bin/tty
/dev/pts/0
```
Instead the full absolute pathname must be used

26

Class Activity

Backup and remove your path variable:

```
/home/cis90/simben $ oldpath=$PATH

/home/cis90/simben $ unset PATH
/home/cis90/simben $ echo $PATH

/home/cis90/simben $ tty
/home/cis90/simben $ /usr/bin/tty
```

What is your shell path now?

Put your answer in the chat window

27

Life without a path

```
/home/cis90/simben $ ls letter
-bash: ls: No such file or directory

/home/cis90/simben $ /bin/ls letter
letter
/home/cis90/simben $
```

On Opus the ls command is in the /bin directory. If we know that a temporary workaround is to specify the full path to the command

28

Life without a path

Some commands still work without a path … why?

```
/home/cis90/simben $ echo "I want my path back"
I want my path back

/home/cis90/simben $ type echo
echo is a shell builtin

/home/cis90/simben $ type type
type is a shell builtin
```

The shell has some commands built into it. The shell does not have to search the path to find these commands so they are always available.

Making a path from scratch

Fixing the path, one directory at a time …

```
/home/cis90/simben $ ls letter
-bash: ls: No such file or directory
```

```
/home/cis90/simben $ PATH=/bin     The ls command is in /bin
/home/cis90/simben $ ls letter     so lets put that on the path
letter
```

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

```
/home/cis90/simben $ PATH=$PATH:/usr/bin    The stat command is in /usr/bin so
/home/cis90/simben $ stat letter            lets append that directory to the
  File: `letter'                            current path
  Size: 1059           Blocks: 16         IO Block: 4096
regular file
Device: fd00h/64768d    Inode: 102594       Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/simben90)   Gid: (
90/   cis90)
Access: 2012-04-30 15:43:28.000000000 -0700
Modify: 2012-03-20 10:31:30.000000000 -0700
Change: 2012-04-30 07:34:30.000000000 -0700
```

30

You try it

```
ls letter
PATH=/bin
echo $PATH
ls letter

stat letter
PATH=$PATH:/usr/bin
echo $PATH
stat letter
```

What is your shell path now?

Put your answer in the chat window

Making a path from scratch

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

/home/cis90/simben $ PATH=$PATH:/home/cis90/bin
/home/cis90/simben $ allscripts
```

*The **allscripts** shell script is in /home/cis90/bin so let's add that directory to the path as well*

```
*************************************************************
*              Fall 2012 CIS 90 Online Projects             *
*************************************************************
1) Andrew
2) Ben
3) Benji
4) Bryn
5) Carlile
6) Carlos
   <snipped>
21) Ray
22) Rita
23) Sean C.
24) Sean F.
25) Shahram

99) Exit

Enter Your Choice:
```

32

You try it

```
allscripts
PATH=$PATH:/home/cis90/bin
echo $PATH
allscripts
```

What is your shell path now?

Put your answer in the chat window

33

Making a path from scratch

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

*The **tryme** shell script is in your own bin directory so lets add that to the path as well*

```
/home/cis90/simben $ PATH=$PATH:/home/cis90/simben/bin
/home/cis90/simben $ tryme
My name is "tryme"
I am pleased to make your acquaintance, Homer Miller
/tmp
/home/cis90/simben $
```

34

You try it

```
tryme
PATH=$PATH:/home/cis90/simben/bin
echo $PATH
tryme
```
Change this to your own home directory

or

```
tryme
PATH=$PATH:$HOME/bin
echo $PATH
tryme
```

What is your shell path now?

Put your answer in the chat window

35

Making a path from scratch

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

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

A temporary workaround is to put a ./ in front of the command

How can I run a script in the current directory without having to put a ./ in front of it?

36

Making a path from scratch

Easy … add the "." directory to the path

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

/home/cis90/simben $ PATH=$PATH:.
/home/cis90/simben $ dogbone
What is your name? Benji
What is your favorite bone? Chicken
Hi Benji, your favorite bone is Chicken
```

37

You try it

```
cd
cp /home/cis90/depot/scripts/dogbone .
```

Did you do this the hard way or use tab completes?

```
chmod +x dogbone

dogbone
./dogbone

PATH=$PATH:.
dogbone
```

What is your shell path now?

Put your answer in the chat window

38

Making a path from scratch

Rebuilding the path by appending directories one at a time

```
/home/cis90/simben $ unset PATH
/home/cis90/simben $ echo $PATH

/home/cis90/simben $ PATH=/bin          Start with /bin which has all the
/home/cis90/simben $ echo $PATH         essential UNIX/Linux commands
/bin

/home/cis90/simben $ PATH=$PATH:/usr/bin          Append /usr/bin which has hundreds of
/home/cis90/simben $ echo $PATH                   additional UNIX/Linux commands
/bin:/usr/bin

/home/cis90/simben $ PATH=$PATH:/home/cis90/bin          Append the CIS 90 class
/home/cis90/simben $ echo $PATH                          bin directory
/bin:/usr/bin:/home/cis90/bin

/home/cis90/simben $ PATH=$PATH:/home/cis90/simben/bin          Append your own student bin
/home/cis90/simben $ echo $PATH                                 directory
/bin:/usr/bin:/home/cis90/bin:/home/cis90/simben/bin

/home/cis90/simben $ PATH=$PATH:.
/home/cis90/simben $ echo $PATH
/bin:/usr/bin:/home/cis90/bin:/home/cis90/simben/bin:.          Append the current directory
```

CIS 90 class bin
directory

Student bin
directory

Current
directory

39

.bash_profile

Making the path permanent using .bash_profile

```
/home/cis90/simben $ cat .bash_profile
# .bash_profile

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

# User specific environment and startup programs

PATH=$PATH:/home/cis90/bin:$HOME/bin:.
BASH_ENV=$HOME/.bashrc
USERNAME=""
PS1='$PWD $ '
export USERNAME BASH_ENV PATH
umask 002
set -o ignoreeof
stty susp
eval `tset -s -m vt100:vt100 -m :\?${TERM:-ansi} -r -Q `

/home/cis90/simben $
```

This customizes the normal path by appending the class bin directory, the student's bin directory and the "current" directory

40

Extra Credit Special Answer

Extra Credit Special
(from Lesson 12)

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 Lesson 13 class starts

42

Lesson 12 Review

The rules of the road for variables

Process Rule #1: When a shell forks a child, only copies of exported variables are made available to the child.

Process Rule #2: A child can modify the variables it receives but those modifications will not change the parent's variables.

Running a script

PID: 582

PID: 582

child

bash

exec()

<script>

exit()

X

PPID: 501

PPID: 501

PID: 501

PID: 501

PID: 501

bash

fork()

bash

wait()

bash

PPID: 250

parent

PPID: 250

PPID: 250
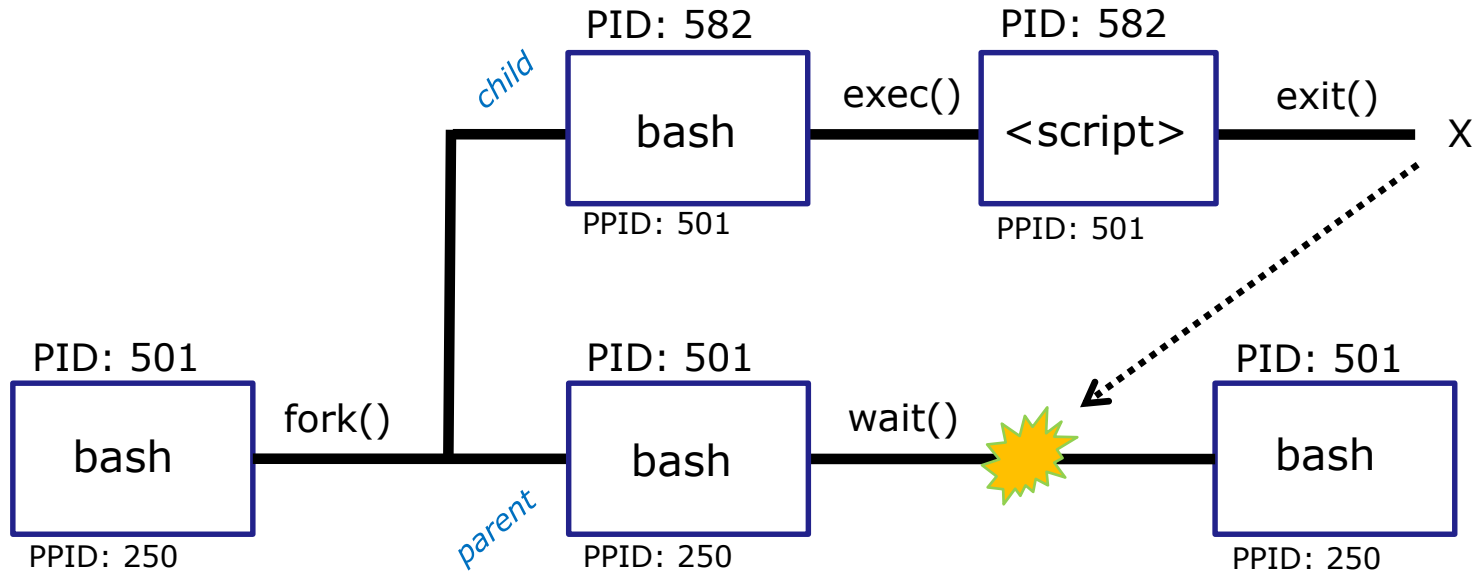
Scripts run as a child process and the rules apply:

- When a shell forks a child process, only copies of exported variables are made available to the child.

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

But what if we want a script to change the parent's variables?

45

. and source

Sometimes it is desirable to run a shell script (like .bash_profile or .bashrc) that will initialize or change shell variables in the parent environment.

. *<script>*
source *<script>*   *equivalent*

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.

In the generic example above, the commands in the file *<script-name>* are run by the parent process, and therefore, any changes made to the environment will last for the duration of the login session.

You try it

```
echo "smartphone=android" > google
echo 'echo smartphone is $smartphone' >> google
cat google
chmod +x google
```

Check that your google file contains:
`smartphone=android`
`smartphone is $smartphone`

```
echo $smartphone
```
Should be null

Method 1
```
google
echo $smartphone
```
Run google script as a child process

Method 2
```
. google
echo $smartphone
```
Source google script so it runs as part of the parent process

Which method of running a script above changed the parent's smartphone variable?

Put your answer in the chat window

47

The exec system call
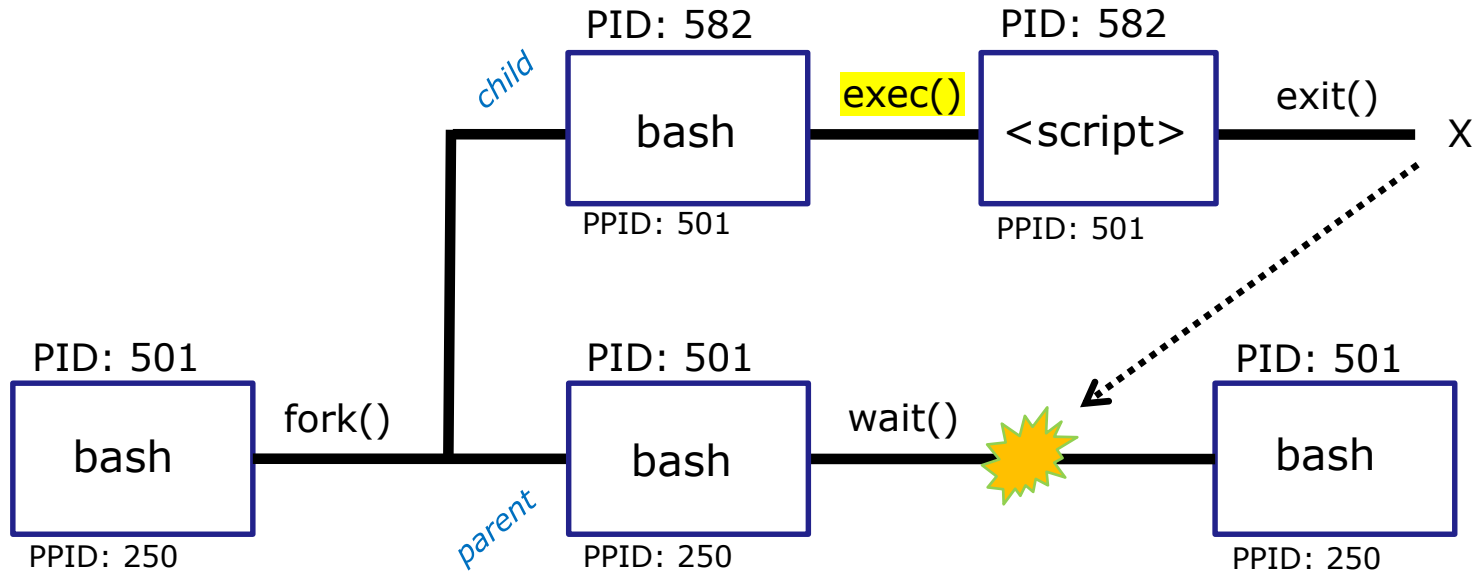
The exec() system call overlays the code in the child process with the script commands

exec command

exec *<command>*

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.

You try it

```
echo "smartphone=android" > google
echo 'echo smartphone is $smartphone' >> google
cat google
chmod +x google
```
Check that your google file contains:
`smartphone=android`
`smartphone is $smartphone`

```
echo $smartphone
```
Should be null

Method 1
```
google
echo $smartphone
```
Run google script as a child process

Method 2
```
exec google
```
Exec the script so it replaces the code in the parent bash process

When you exec a script what happens when the script is finished?

Put your answer in the chat window

50

grok that?
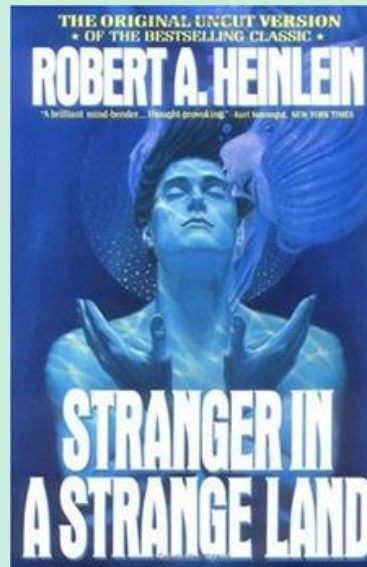
The flowers script
/home/cis90/bin/flowers

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

Show the parent, child and the ps processes

Show the values of the roses and violets variables

Set the values of the roses and violets variables to new values

The flowers script
/home/cis90/bin/flowers

```
/home/cis90/simben $ flowers

==> Entering child process <==
UID          PID   PPID  C STIME TTY          TIME CMD
simben90 17518  17512  0 08:32 pts/0    00:00:00 -bash
simben90 17568  17518  0 08:33 pts/0    00:00:00 /bin/bash /home/cis90/bin/flowers
simben90 17575  17568  8 08:33 pts/0    00:00:00 ps -f
==> showing variables in child <==
   roses are ""
   violets are ""
==> setting variables in child <==
   roses are "black"
   violets are "orange"
==> Leaving child process <==

/home/cis90/simben $
```

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

The flowers script
/home/cis90/bin/flowers
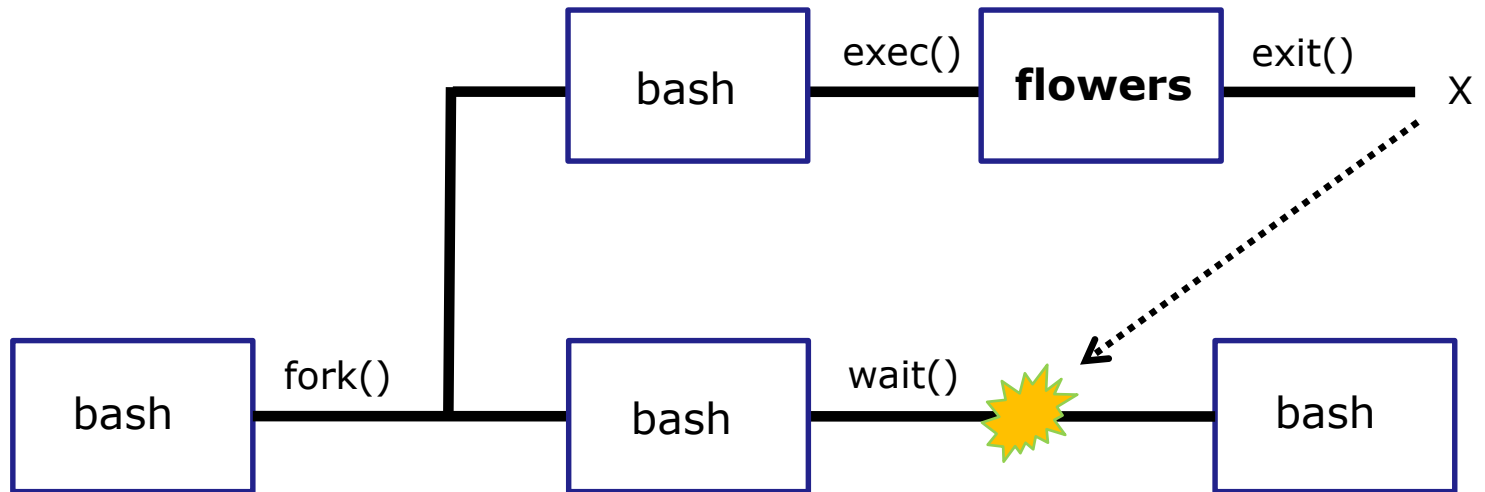
*Use the **flowers** script to test your understanding of how variables are handled with child processes*

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.

55

Activity

Setup this alias so you can use it in activities that follow:

`alias go='echo roses are \"$roses\" and violets are \"$violets\"'`

What happens now when you type the go command?

Type your answer in the chat window

Use the alias to show the values of the two 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

57

Use the alias to show the values of the two 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*

58

Activity

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

When the flowers script runs will it see the values of the roses and violets variables?

Write your answer in the chat window

59

NO, the roses and violets variables were not exported

```
/home/cis90/simben $ flowers

==> Entering child process <==
UID         PID  PPID  C STIME TTY          TIME CMD
simben90 25106 25059  0 17:16 pts/8     00:00:00 -bash
simben90 27052 25106  0 17:19 pts/8     00:00:00 /bin/bash /home/cis90/bin/flowers
simben90 27059 27052  0 17:19 pts/8     00:00:00 ps -f
==> showing variables in child <==
   roses are ""        The child cannot view the values of the parent's
   violets are ""      non-exported variables (Rule #1)
==> setting variables in child <==
   roses are "black"
   violets are "orange"
==> Leaving child process <==

/home/cis90/simben $
```

60

Activity

```
/home/cis90/simben $ roses=red; violets=blue
/home/cis90/simben $ export roses
/home/cis90/simben $ env | grep roses
roses=red
/home/cis90/simben $ env | grep violets
/home/cis90/simben $ go
roses are "red" and violets are "blue"
/home/cis90/simben $ flowers
```

When the flowers script runs will it see the value of the roses variable or the violets variable?

Write your answer in the chat window

Yes, the flowers script can see the roses variable now which was exported

```
/home/cis90/simben $ flowers

==> Entering child process <==
UID          PID  PPID  C STIME TTY          TIME CMD
simben90 25106 25059  0 17:16 pts/8    00:00:00 -bash
simben90 32147 25106  0 17:27 pts/8    00:00:00 /bin/bash /home/cis90/bin/flowers
simben90 32154 32147  0 17:27 pts/8    00:00:00 ps -f
==> showing variables in child <==
   roses are "red"
   violets are ""
==> setting variables in child <==
   roses are "black"
   violets are "orange"
==> Leaving child process <==

/home/cis90/simben $
```

The child now sees the value of roses but not violets (Rule #1)

62

Activity

```
/home/cis90/simben $ roses=red; violets=blue
/home/cis90/simben $ export roses violets
/home/cis90/simben $ env | grep roses
roses=red
/home/cis90/simben $ env | grep violets
violets=blue
/home/cis90/simben $ go
roses are "red" and violets are "blue"
/home/cis90/simben $ flowers
```

**Will the flowers process change the values of
the roses and violets variables?**

Write your answer in the chat window

63

No, the flowers script which runs as a child process cannot change the parent's variables

```
/home/cis90/simben $ flowers

==> Entering child process <==
UID          PID   PPID  C STIME TTY          TIME CMD
simben90 28732 28724  0 17:51 pts/0    00:00:00 -bash
simben90 29383 28732  0 18:11 pts/0    00:00:00 /bin/bash /home/cis90/bin/flowers
simben90 29390 29383  0 18:11 pts/0    00:00:00 ps -f
==> showing variables in child <==
   roses are "red"
   violets are "blue"
==> setting variables in child <==
   roses are "black"
   violets are "orange"
==> Leaving child process <==

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

The child can only change copies of the parents variables

The child cannot change the parent's variables (Rule #2)

Activity

```
/home/cis90/simben $ roses=red; violets=blue
/home/cis90/simben $ export roses violets
/home/cis90/simben $ env | grep roses
roses=red
/home/cis90/simben $ env | grep violets
violets=blue
/home/cis90/simben $ go
roses are "red" and violets are "blue"
/home/cis90/simben $ . flowers
```

Now will the flowers process change the values of the roses and violets variables?

Write your answer in the chat window

Yes, if sourced, flowers will not run as a child process and can change the parent's variables

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

==> Entering child process <==
UID          PID   PPID  C STIME TTY          TIME CMD
simben90 28732 28724  0 17:51 pts/0    00:00:00 -bash
simben90 29480 28732  0 18:15 pts/0    00:00:00 ps -f
==> showing variables in child <==
   roses are "red"
   violets are "blue"
==> setting variables in child <==
   roses are "black"
   violets are "orange"
==> Leaving child process <==

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

66

```
/home/cis90/rodduk $ 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
eval `tset -s -m vt100:vt100 -m

/home/cis90/rodduk $
```

And now you know why the bash login scripts are sourced rather than run as child processes.

*Note: the **.** (dot) and **source** commands are equivalent*

```
/home/cis90/rodduk $ cat .bashrc
# .bashrc

# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
alias print="echo -e"
```

67

Activity

```
/home/cis90/simben $ roses=red; violets=blue
/home/cis90/simben $ export roses violets
/home/cis90/simben $ env | grep roses
roses=red
/home/cis90/simben $ env | grep violets
violets=blue
/home/cis90/simben $ go
roses are "red" and violets are "blue"
/home/cis90/simben $ exec flowers
```

What will happen if flowers is exec'ed?

Write your answer in the chat window

The flowers script overlays and replaces the bash code in your current process. It runs to completion and your session ends!

Housekeeping

70

1. Lab 10 due by 11:59PM tonight

2. Use the **check10** script to check your work

3. Don't forget to submit your work!

4. After you submit your lab10 file you may comment out your riddle command in *.bash_profile*

5. The Extra Credit Labs X1 and X2 (30 points each) are available. The will be graded after the day of the final. Use **checkx2** to the second lab.

6. The Final Project is available and due in **two weeks**.

Heads up on Final Exam

Test #3 (final exam) is MONDAY December 11th 1-3:50PM

Test #3 (the final exam)

Time
- MONDAY 1:00PM - 3:50PM in Room 828

Mon 12/11

Materials
- Test (canvas)

CCC Confer
- Enter virtual classroom
- Class archives

5 posts
Lab X1
Lab X2

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

- All students will take the test at the <u>same</u> <u>time</u>. The test must be completed by 3:50PM.

- Working and long distance students can take the test online via CCC Confer and Canvas.

- Working students will need to plan ahead to arrange time off from work for the test.

- Test #3 is **mandatory** (even if you have all the points you want)

72

FALL 2017 FINAL EXAMINATIONS SCHEDULE
DECEMBER 11 TO DECEMBER 16

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
Classes starting between:		
6:30 am and 8:55 am, MW/Daily	7:00 am-9:50 am	Monday, December 11
9:00 am and 10:15 am, MW/Daily	7:00 am-9:50 am	Wednesday, December 13
10:20 am and 11:35 am, MW/Daily	10:00 am-12:50 pm	Monday, December 11
11:40 am and 12:55 pm, MW/Daily	10:00 am-12:50 pm	Wednesday, December 13
1:00 pm and 2:15 pm, MW/Daily	1:00 pm-3:50 pm	Monday, December 11
2:20 pm and 3:35 pm, MW/Daily	1:00 pm-3:50 pm	Wednesday, December 13
3:40 pm and 5:30 pm, MW/Daily	4:00 pm-6:50 pm	Monday, December 11

6:30 am and 8:55 am, TTh	7:00 am-9:50 am	
9:00 am and 10:15 am, TTh	7:00 am-9:50 am	
10:20 am and 11:35 am, TTh	10:00 am-12:50 pm	
11:40 am and 12:55 pm, TTH	10:00 am-12:50 pm	
1:00 pm and 2:15 pm, TTh	1:00 pm-3:50 pm	
2:20 pm and 3:35 pm, TTh	1:00 pm-3:50 pm	
3:40 pm and 5:30 pm, TTh	4:00 pm-6:50 pm	
Friday am	9:00 am-11:50 am	
Friday pm	1:00 pm-3:50 pm	
Saturday am	9:00 am-11:50 am	
Saturday pm	1:00 pm-3:50 pm	

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
98169	W	1:00PM-4:05PM	3.00	R.Simms	OL
&	Arr.	Arr.		R.Simms	OL

Section 98169 is an ONLINE course. Meets weekly throughout the semester online during the scheduled times by remote technology with an additional 50 min online lab per week. For details, see instructor's web page at go.cabrillo.edu/online.

Section	Days	Times	Units	Instructor	Room
98170	W	1:00PM-4:05PM	3.00	R.Simms	828
&	Arr.	Arr.		R.Simms	OL

Section 98170 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.

73

Where to find your grades
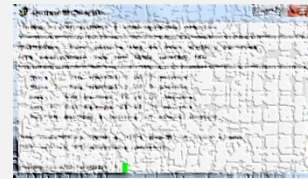
Send me your survey to get your LOR code name.

The CIS 90 website Grades page

http://simms-teach.com/cis90grades.php
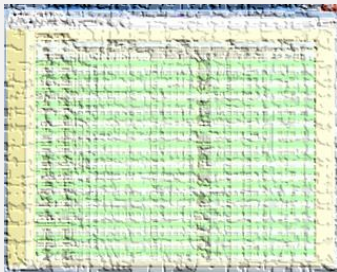
Or check on Opus-II

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

Written by Jesse Warren a past CIS 90 Alumnus

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

Points that could have been earned:
9 quizzes:	27 points
9 labs:	270 points
2 tests:	60 points
3 forum quarters:	60 points
Total:	**417 points**

At the end of the term I'll add up all your points and assign you a grade using this table
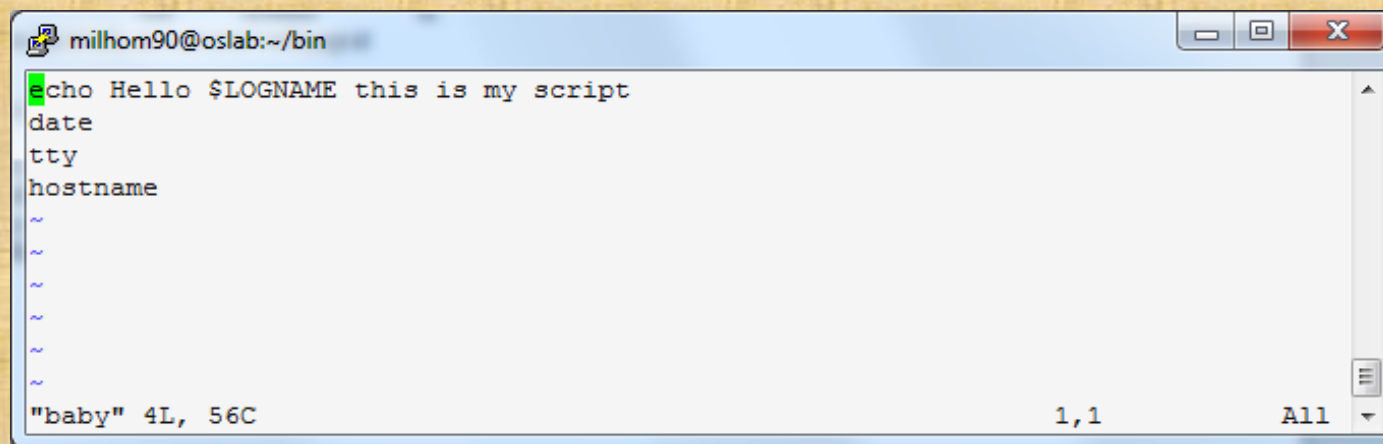
74

Shell Scripting 101

Shell Scripts

- In its simplest form a shell script can just be a list of commands in a file

- Execute "x" permissions must be enabled on the script file.

- The script must either be on your path or you must use an absolute pathname to run it.

- Putting #!/bin/bash on line 1 specifies which program should be used to execute the script. The default if not specified is /bin/bash. Note this enables vi to use color syntax.

- Putting the exit command at the end triggers a system call to the kernel to terminate the process and release all resources. Note a numerical status can be specified as an argument (e.g. exit 20) which will be communicated back to the parent process.

Class Activity

```
/home/cis90/milhom $ cd bin
/home/cis90/milhom/bin $ cp ~/../depot/scripts/baby .
/home/cis90/milhom/bin $ vi baby
```

```
milhom90@oslab:~/bin

echo Hello $LOGNAME this is my script
date
tty
hostname
~
~
~
~
~
~
"baby" 4L, 56C                                    1,1            All
```

use `Esc` `:wq` *to save file and quit vi*

```
/home/cis90/milhom/bin $ chmod +x baby
/home/cis90/milhom/bin $ baby
Hello milhom90 this is my script
Tue Nov 24 14:10:42 PST 2015
/dev/pts/3
oslab.cis.cabrillo.edu
```

78

$(*some-command*)

Utilizing $(*some-command*)

The **$** metacharacter provides the "value" of both variables, e.g. $PS1 or commands, e.g. $(*some-command*):

```
/home/cis90/simben $ echo $PS1
$PWD $

/home/cis90/simben $ echo $(grep love poems/Shakespeare/* | wc -l)
11

/home/cis90/simben $ myname=$(grep $LOGNAME /etc/passwd | cut -f5 -d":")
/home/cis90/simben $ echo My name is $myname
My name is Benji Simms
```

This is useful when you want to insert the output of a command into a sentence being echoed

80

More on the date command

Utilizing the date command

```
/home/cis90/milhom/bin $ date
Tue Nov 24 14:33:41 PST 2015

/home/cis90/milhom/bin $ date +'%r'
02:33:53 PM

/home/cis90/milhom/bin $ date +'%A'
Tuesday

/home/cis90/milhom/bin $ date +'%m/%d/%Y'
11/24/2015
```
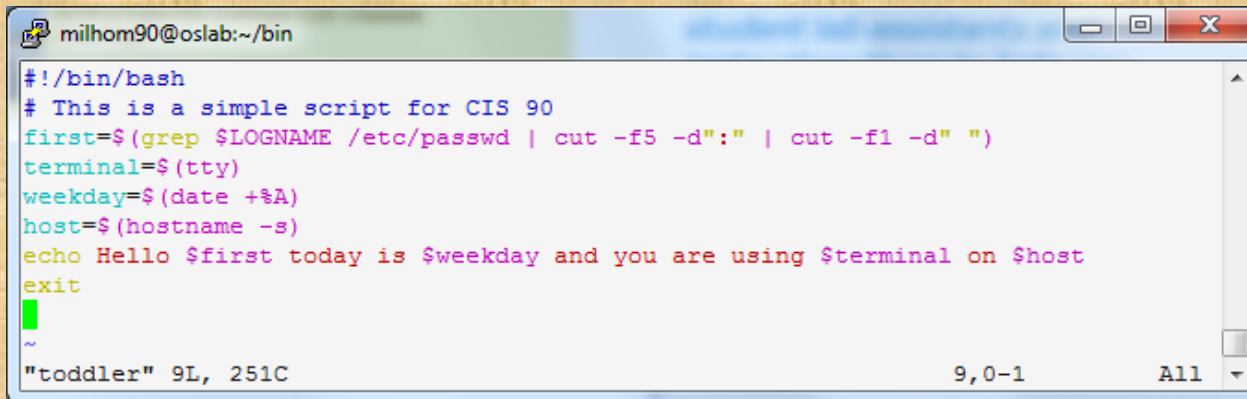
See the man page on date for lots of other % codes

Class Activity

```
/home/cis90/milhom/bin $ cd ~/bin
/home/cis90/milhom/bin $ cp ~/../depot/scripts/toddler .
/home/cis90/milhom/bin $ vi toddler
```



```
milhom90@oslab:~/bin

#!/bin/bash
# This is a simple script for CIS 90
first=$(grep $LOGNAME /etc/passwd | cut -f5 -d":" | cut -f1 -d" ")
terminal=$(tty)
weekday=$(date +%A)
host=$(hostname -s)
echo Hello $first today is $weekday and you are using $terminal on $host
exit

~
"toddler" 9L, 251C                                    9,0-1        All
```

use **Esc** **:wq** *to save file and quit vi*

```
/home/cis90/milhom/bin $ chmod +x toddler
/home/cis90/milhom/bin $ toddler
Hello Homer today is Tuesday and you are using /dev/pts/3 on oslab
```

83

Interactive scripts using the read command

Class Activity

```
/home/cis90/milhom/bin $ cd ~/bin
/home/cis90/milhom/bin $ cp ~/../depot/scripts/interactive .
/home/cis90/milhom/bin $ vi interactive
```


milhom90@oslab:~/bin

```
#!/bin/bash
echo Pick a number between 1 and 5
read a

echo -n "Pick a number between 1 and 5: "
read b

read -p "Pick a number between 1 and 5: " c

echo "You picked $a, $b, and $c."
exit
```

Use echo and read to prompt then read response

Use -n option on echo to suppress the newline (carriage return)

Use -p option on read to specify a prompt string without a preceding echo command

"interactive" 12L, 190C   1,1   All
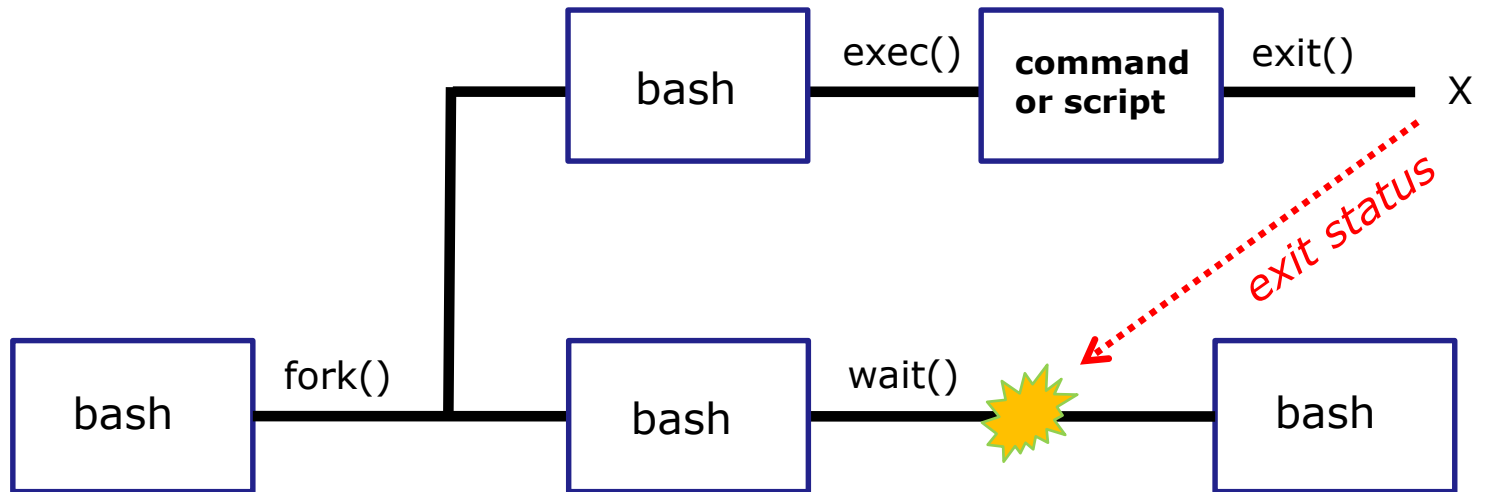
use **Esc** **:wq** *to save file and quit vi*

```
/home/cis90/milhom/bin $ chmod +x interactive
/home/cis90/milhom/bin $ interactive
Pick a number between 1 and 5
2
Pick a number between 1 and 5: 4
Pick a number between 1 and 5: 5
You picked 2, 4, and 5.
```

85

Communicating status back to parent

The child can communicate status back to the parent

The child process makes a exit() system call to release all resources.
The child remains a zombie until the exit status is communicated to
the parent.

Utilizing the status

Yes, there is a variable named ?

This variable will be set to the exit status of the command or script that just ran.

```
/home/cis90/milhom/bin $ grep bogus /etc/passwd > /dev/null
/home/cis90/milhom/bin $ echo $?
```
1   *status=1 (grep found no matches)*

```
/home/cis90/milhom/bin $ grep $LOGNAME /etc/passwd > /dev/null
/home/cis90/milhom/bin $ echo $?
```
0   *status=0 (grep found one or more matches)*

A status=0 typically indicates success and non-zero values are error codes

Utilizing the status

```
/home/cis90/milhom/bin $ ping -c1 son-of-opus.simms-teach.com
PING son-of-opus.simms-teach.com (52.8.145.169) 56(84) bytes of data.

--- son-of-opus.simms-teach.com ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 10000ms
/home/cis90/milhom/bin $ echo $?
1
```

status=1 (Son-of-Opus system at AWS is down right now to save $)

```
/home/cis90/milhom/bin $ ping -c1 simms-teach.com
PING simms-teach.com (208.113.154.64) 56(84) bytes of data.
64 bytes from apache2-dap.giles.dreamhost.com (208.113.154.64): icmp_seq=1 ttl=43 time=78.9 ms

--- simms-teach.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 164ms
rtt min/avg/max/mdev = 78.957/78.957/78.957/0.000 ms
/home/cis90/milhom/bin $ echo $?
0
```

status=0 (simms-teach.com website is up right now)

89

Class Activity

```
/home/cis90/milhom/bin $ cd ~/bin
/home/cis90/milhom/bin $ cp ~/../depot/scripts/kid .
/home/cis90/milhom/bin $ vi kid
```

```
#!/bin/bash
echo "This is the $0 script running as a child process"
read -p "Enter a status number (0-255) to return to the parent process: " status
echo "You entered $status, use: echo \$? to view from the parent"
exit $status
```

use Esc *:wq to save file and quit vi*

```
/home/cis90/milhom/bin $ ./kid
This is the ./kid script running as a child process
Enter a status number (0-255) to return to the parent process: 25
You entered 25, use: echo $? to view from the parent
/home/cis90/milhom/bin $ echo $?
25
```

Final Project
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: "
        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
~
```

You will modify and extend this script for your final project

1,1   All

92

Final Project

If you did not do this last week, please do so now

Getting Started

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

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

3) Run the script:
   myscript

Final Project


```
simben90@oslab:~                                                    —    □    ×

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

99) Exit

Enter Your Choice: 
```

*Before leaving class today, make sure you can run your **myscript** from **allscripts***

94

Final Project Grading Rubric

Possible Points	Requirements

30	Implementing all five tasks (6 points each):
	• Requirements for each task:
	  – Minimum of 10 "original" script command lines
	  – Has one or more non-generic comments to explain what it is doing
	  – Has user interaction

25	You don't have to do all of these but do at least five: • Redirecting stdin (5 points) • Redirecting stdout (5 points) • Redirecting stderr (5 points) • Use of permissions (5 points) • Use of filename expansion characters (5 points) • Use of absolute path (5 points) • Use of relative path (5 points) • Use of a PID (5 points) • Use of inodes (5 points) • Use of links (5 points) • Use of color (5 points) • Use of scheduling (5 points) • Use of a GID or group (5 points) • Use of a UID or user (5 points) • Use of a /dev/tty device (5 points) • Use of a signal (5 points) • Use of piping (5 points) • Use of an environment variable (5 points) • Use of /bin/mail (5 points) • Use of a conditional (5 points) The maximum for this section is 25 points.
5	Present your script to the class
Points lost	
-15	Fails to run from **allscripts**
-15	Other students in the class are unable to read and execute your script.
-15	Error messages are displayed when running one or more tasks
-up to 90	No credit for any task which contains unoriginal script code that: • Doesn't give full credit to the original author. • Doesn't indicate where the code was obtained from. • Doesn't include licensing terms. • Violates copyright or licensing terms.
-up to 90	For any "malware" scripts that steal credentials, exfiltrate confidential information, remove or encrypt a user's files or creates a denial of service condition on Opus-II.
Extra credit	
30	Up to three additional tasks (10 points each)

Plagiarizing another author's code is a NO-NO! All points lost!

Scripts that result in unauthorized hacking" is a NO-NO! All points lost!

96

Final Project

permissions

Permissions

A past forum post …

Uh, oh … someone got hacked!

Group Write Permissions

ls -l /home/cis90/*/bin/myscript

```
[rsimms@opus-ii ~]$ ls -l /home/cis90/*/bin/myscript
-rwxrwxr-x. 1 alvnic90 cis90 551 Nov 15 13:55 /home/cis90/alvnic/bin/myscript
-rwxrwxr-x. 1 bermoi90 cis90 549 Nov 15 13:41 /home/cis90/bermoi/bin/myscript
-rwxrwxr-x. 1 boysam90 cis90 875 Nov 15 13:51 /home/cis90/boysam/bin/myscript
-rwxrwxr-x. 1 bresea90 cis90 694 Nov 15 14:25 /home/cis90/bresea/bin/myscript
-rwxrwxr-x. 1 brokyl90 cis90 716 Nov 15 14:11 /home/cis90/brokyl/bin/myscript
-rwxrwxr-x. 1 jarale90 cis90 787 Nov 15 14:25 /home/cis90/jarale/bin/myscript
-rwxrwxr-x. 1 macdav90 cis90 549 Nov 15 13:41 /home/cis90/macdav/bin/myscript
-rwxrwxr-x. 1 mccben90 cis90 722 Nov 15 14:03 /home/cis90/mccben/bin/myscript
-rwxrwxr-x. 1 ottjac90 cis90 835 Nov 15 14:14 /home/cis90/ottjac/bin/myscript
-rwxrwxr-x. 1 panosc90 cis90 718 Nov 15 14:14 /home/cis90/panosc/bin/myscript
-rwxrwxr-x. 1 privin90 cis90 717 Nov 15 15:13 /home/cis90/privin/bin/myscript
-rwxr-xr-x. 1 samwil90 cis90 716 Nov 15 14:03 /home/cis90/samwil/bin/myscript
-rwxrwxr-x. 1 siljas90 cis90 726 Nov 15 14:15 /home/cis90/siljas/bin/myscript
-rwxrwxr-x. 1 simben90 cis90 549 Nov 12 17:59 /home/cis90/simben/bin/myscript
[rsimms@opus-ii ~]$
```

*Which **myscript** files can only be edited by their owner?*
Which ones could be edited by anyone in the CIS 90 class?
Which ones could be edited by anyone on Opus-II?

99

Group Read and Execute Permissions

```
rsimms@opus-ii:~                                                    —    □    ×
[rsimms@opus-ii ~]$ /home/cis90/bin/checkmyscripts
-rwxrwxr-x. 1 simben90 cis90 549 Nov 12 17:59 /home/cis90/simben/bin/myscript
ls: cannot access /home/cis90/milhom/bin/myscript: No such file or directory
ls: cannot access /home/cis90/rodduk/bin/myscript: No such file or directory
ls: cannot access /home/cis90/agunic/bin/myscript: No such file or directory
-rwxrwxr-x. 1 alvnic90 cis90 551 Nov 15 13:55 /home/cis90/alvnic/bin/myscript
-rwxrwxr-x. 1 bermoi90 cis90 549 Nov 15 13:41 /home/cis90/bermoi/bin/myscript
-rwxrwxr-x. 1 bresea90 cis90 694 Nov 15 14:25 /home/cis90/bresea/bin/myscript
ls: cannot access /home/cis90/howdan/bin/myscript: No such file or directory
ls: cannot access /home/cis90/lucram/bin/myscript: No such file or directory
ls: cannot access /home/cis90/malman/bin/myscript: No such file or directory
-rwxrwxr-x. 1 mccben90 cis90 722 Nov 15 14:03 /home/cis90/mccben/bin/myscript
ls: cannot access /home/cis90/mouvin/bin/myscript: No such file or directory
-rwxr-xr-x. 1 samwil90 cis90 716 Nov 15 14:03 /home/cis90/samwil/bin/myscript
ls: cannot access /home/cis90/wonmar/bin/myscript: No such file or directory
ls: cannot access /home/cis90/johemm/bin/myscript: No such file or directory
-rwxrwxr-x. 1 boysam90 cis90 875 Nov 15 13:51 /home/cis90/boysam/bin/myscript
-rwxrwxr-x. 1 brokyl90 cis90 716 Nov 15 14:11 /home/cis90/brokyl/bin/myscript
-rwxrwxr-x. 1 jarale90 cis90 787 Nov 15 14:25 /home/cis90/jarale/bin/myscript
-rwxrwxr-x. 1 macdav90 cis90 549 Nov 15 13:41 /home/cis90/macdav/bin/myscript
ls: cannot access /home/cis90/ngujos/bin/myscript: No such file or directory
-rwxrwxr-x. 1 ottjac90 cis90 835 Nov 15 14:14 /home/cis90/ottjac/bin/myscript
-rwxrwxr-x. 1 panosc90 cis90 718 Nov 15 14:14 /home/cis90/panosc/bin/myscript
ls: cannot access /home/cis90/prihay/bin/myscript: No such file or directory
-rwxrwxr-x. 1 privin90 cis90 717 Nov 15 15:13 /home/cis90/privin/bin/myscript
ls: cannot access /home/cis90/schmic/bin/myscript: No such file or directory
-rwxrwxr-x. 1 siljas90 cis90 726 Nov 15 14:15 /home/cis90/siljas/bin/myscript
ls: cannot access /home/cis90/temtyl/bin/myscript: No such file or directory
ls: cannot access /home/cis90/tbd02/bin/myscript: No such file or directory
ls: cannot access /home/cis90/tbd03/bin/myscript: No such file or directory
[rsimms@opus-ii ~]$ 
```

Which myscript files cannot by run by classmates?

100

Class Activity

Note: One of the requirements for the final project is setting permissions on your script so that all cis90 members can read and run it.

To meet this requirement use:

```
cd
chmod 750 bin bin/myscript
ls -ld bin bin/myscript
```

When finished check that your script can be run by other CIS 90 students:

```
su - cis90
   (use the "funny Cabrillo" password)
allscripts
exit
```

Run you script and write "success" or "not working" into the chat window

umask again!

Permissions
Why can other classmates write to my scripts?

Before Lab 10
```
/home/cis90/simben/bin $ umask
0002
/home/cis90/simben $ rm newscript; touch newscript
/home/cis90/simben $ ls -l newscript
-rw-rw-r-- 1 simben cis90 0 Nov 23 16:17 newscript
/home/cis90/simben $ chmod +x newscript
/home/cis90/simben $ ls -l newscript
-rwxrwxr-x 1 simben cis90 0 Nov 23 16:17 newscript
```

After Lab 10
```
/home/cis90/simben $ umask
0006
/home/cis90/simben $ rm newscript; touch newscript
/home/cis90/simben $ ls -l newscript
-rw-rw---- 1 simben cis90 0 May 12 08:44 newscript
/home/cis90/simben $ chmod +x newscript
/home/cis90/simben $ ls -l newscript
-rwxrwx--x 1 simben cis90 0 May 12 08:44 newscript
```

Because your umask setting allows group members to have write permission on any new files you create!

103

Permissions

```
[rodduk90@opus-ii bin]$ cat /home/cis90/rodduk/.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
eval `tset -s -m vt100:vt100 -m :\?${TERM:-ansi} -r -Q `
```

Note your umask is defined in .bash_profile which runs every time you login. In lab 10 you change this setting to 006.

104

Class Activity

- Change your umask to 026

- Can group or other users modify future new files now?

- Try it, **touch** a new file and check the permissions with **ls -l**

How would you make this a permanent umask setting?

Write your answer in the chat window

Final Project Getting Started

What takes longer?

Writing the script?

Or deciding what to script?

One way to get started … select a random command to build a script around

Commands

.	echo	lpstat	sort
at	env	ls	spell
banner	exit	mail	su
bash	export	man	tail
bc	file	me...	tee
cal	find		touch
cancel	finger	...ore	type
cat	grep	mv	umask
cd	head	passwd	uname
chgrp	history		unset
chmod	id		vi
chown	jobs	rm	wc
clear	kill	rmdir	who
cp	ln	...t	write
date	lp/lpr	sleep	xxd

For this example we will pick the grep command

108

Research your command by reading the man page and googling examples

Review the various options and arguments for the command

Next, decide what you want to do with the command you selected. For this example we will:

1. Start a new task in **myscript**

2. Customize the menu for the new task

3. Start with a simple **grep** command

4. Add some simple interaction

5. Add successive grep commands that experiment with different options

6. Iterate till happy with it.

Start hacking
the menu!

*Customize the menu
options for Task 1*

After

```
#!/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
                  ;;
          2)      # Commands for Task
                  ;;
"myscript" 37L, 546C
```

Before

```
#!/bin/bash
#
# menu: A simple menu template
#
while true
do
        clear
        echo -n "
                CIS 90 Final Project
        1) Hacking with the grep command
        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
                  ;;
-- INSERT --                                              10,5-12        Top
```

111

www.catb.org/jargon/html/H/hacker.html

hacker: n.

[originally, someone who makes furniture with an axe]

1. A person who enjoys exploring the details of programmable systems and how to stretch their capabilities, as opposed to most users, who prefer to learn only the minimum necessary. RFC1392, the *Internet Users' Glossary*, usefully amplifies this as: A person who delights in having an intimate understanding of the internal workings of a system, computers and computer networks in particular.

2. One who programs enthusiastically (even obsessively) or who enjoys programming rather than just theorizing about programming.

3. A person capable of appreciating *hack value*.

4. A person who is good at programming quickly.

5. An expert at a particular program, or one who frequently does work using it or on it; as in 'a Unix hacker'. (Definitions 1 through 5 are correlated, and people who fit them congregate.)

6. An expert or enthusiast of any kind. One might be an astronomy hacker, for example.

7. One who enjoys the intellectual challenge of creatively overcoming or circumventing limitations.

8. [deprecated] A malicious meddler who tries to discover sensitive information by poking around. Hence password hacker, network hacker. The correct term for this sense is *cracker*.

The term 'hacker' also tends to connote membership in the global community defined by the net (see *the network*. For discussion of some of the basics of this culture, see the How To Become A Hacker FAQ. It also implies that the person described is seen to subscribe to some version of the hacker ethic (see *hacker ethic*).

It is better to be described as a hacker by others than to describe oneself that way. Hackers consider themselves something of an elite (a meritocracy based on ability), though one to which new members are gladly welcome. There is thus a certain ego satisfaction to be had in identifying yourself as a hacker (but if you claim to be one and are not, you'll quickly be labeled *bogus*). See also *geek*, *wannabee*.

This term seems to have been first adopted as a badge in the 1960s by the hacker culture surrounding TMRC and the MIT AI Lab. We have a report that it was used in a sense close to this entry's by teenage radio hams and electronics tinkerers in the mid-1950s.

112

Hacking (building, exploring) is not cracking (malicious)

Layout your work area on the screen


```
rodduk90@oslab:~/bin
#!/bin/bash
#
# menu: A simple menu template
#
while true
do
        clear
        echo -n "
                CIS 90 Final Project
        1) Hacking with the grep command
        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
~
~
~
-- INSERT --                                    1,12        All
```

1st

```
rodduk90@oslab:~/bin
/home/cis90/rodduk $ cd bin
/home/cis90/rodduk/bin $ myscript
```

2nd

```
rodduk90@oslab:~
GREP(1)                                                     GREP(1)

NAME
        grep, egrep, fgrep - print lines matching a pattern

SYNOPSIS
        grep [OPTIONS] PATTERN [FILE...]
        grep [OPTIONS] [-e PATTERN | -f FILE] [FILE...]

DESCRIPTION
        grep  searches the named input FILEs (or standard input if no files are
        named, or if a single hyphen-minus (-) is given as file name) for lines
        containing  a  match to the given PATTERN.  By default, grep prints the
        matching lines.

        In addition, two variant programs egrep and fgrep are available.  egrep
        is  the  same  as  grep -E.   fgrep  is  the  same as grep -F.  Direct
        invocation as either egrep or fgrep is deprecated, but is  provided  to
        allow historical applications that rely on them to run unmodified.

OPTIONS
   Generic Program Information
        --help Print  a  usage  message  briefly summarizing these command-line
:
```

3rd

Utilize screen real estate with multiple windows:

- *the 1st for vi,*
- *the 2nd for testing **myscript**,*
- *and a 3rd for experimenting or showing man pages*

113

Test your menu change


```
#!/bin/bash
#
# menu: A simple menu template
#
while true
do
        clear
        echo -n "
                CIS 90 Final Project
        1) Hacking with the grep command
        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
~
~
~
"myscript" 37L, 569C written                    1,11        All
```

```
                CIS 90 Final Project
        1) Hacking with the grep command
        2) Task 2
        3) Task 3
        4) Task 4
        5) Task 5
        6) Exit

        Enter Your Choice:
```

Changes work!

```
GREP(1)                                                   GREP(1)

NAME
        grep, egrep, fgrep - print lines matching a pattern

SYNOPSIS
        grep [OPTIONS] PATTERN [FILE...]
        grep [OPTIONS] [-e PATTERN | -f FILE] [FILE...]

DESCRIPTION
        grep  searches the named input FILEs (or standard input if no files are
        named, or if a single hyphen-minus (-) is given as file name) for lines
        containing  a  match to the given PATTERN.  By default, grep prints the
        matching lines.

        In addition, two variant programs egrep and fgrep are available.  egrep
        is  the  same  as  grep -E.   fgrep  is  the  same as grep -F.  Direct
        invocation as either egrep or fgrep is deprecated, but is  provided  to
        allow historical applications that rely on them to run unmodified.

OPTIONS
   Generic Program Information
        --help Print  a  usage  message  briefly summarizing these command-line
:
```

*Run **myscript** in the 2nd window and verify your changes work*

114

Find the location to insert your new task commands


```
rodduk90@oslab:~/bin

     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
-- INSERT --                                         12,5-12        78%
```

Insert your new script commands here

Now its time to add some commands to the task.

*Be sure to insert commands **after** the generic comment and **before** the ;;*

115

Add a simple command first and test it

rodduk90@oslab:~/bin

```
#!/bin/bash
#
# menu: A simple menu template
#
while true
do
        clear
        echo -n "
                CIS 90 Final Project
        1) Hacking with the grep command
        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
                grep beauty poems/*/*
                ;;
          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
~
~
"myscript" 38L, 593C written                     21,15-29      All
```

rodduk90@oslab:~/bin

```
                CIS 90 Final Project
        1) Hacking with the grep command
        2) Task 2
        3) Task 3
        4) Task 4
        5) Task 5
        6) Exit

        Enter Your Choice: 1
grep: poems/*/*: No such file or directory
Hit the Enter key to return to menu
```

Oops, the change broke the script! Why?
Because the relative path (beauty poems/*/*)
does not work from the bin directory

rodduk90@oslab:~

```
/home/cis90/rodduk $ grep beauty poems/*/*
poems/Shakespeare/sonnet1:That thereby beauty's rose might never die,
poems/Shakespeare/sonnet10:     That beauty still may live in thine or thee.
poems/Shakespeare/sonnet11:Herein lives wisdom, beauty, and increase;
poems/Shakespeare/sonnet17:If I could write the beauty of your eyes,
poems/Shakespeare/sonnet2:And dig deep trenches in thy beauty's field,
poems/Shakespeare/sonnet2:Then being ask'd, where all thy beauty lies,
poems/Shakespeare/sonnet2:How much more praise deserv'd thy beauty's use,
poems/Shakespeare/sonnet2:Proving his beauty by succession thine.
poems/Shakespeare/sonnet4:Upon thyself thy beauty's legacy?
poems/Shakespeare/sonnet4:        Thy unus'd beauty must be tomb'd with thee,
poems/Shakespeare/sonnet5:Beauty's effect with beauty were bereft,
poems/Shakespeare/sonnet7:Yet mortal looks adore his beauty still,
poems/Shakespeare/sonnet9:But beauty's waste hath in the world an end,
poems/Yeats/old:And loved your beauty with love false or true,
/home/cis90/rodduk $
```

*Experiment with a **grep** command in 3ʳᵈ window*

*In the 1ˢᵗ window add the new grep command then save with **<esc>:w** (don't quit vi)*

*Run **myscript** in the 2ⁿᵈ second window to test change.*

116

Fix it and test again

```
#!/bin/bash
#
# menu: A simple menu template
#
while true
do
        clear
        echo -n "
                CIS 90 Final Project
        1) Hacking with the grep command
        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
                  grep beauty /home/cis90/rodduk/poems/*/*
                  ;;
            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
~
~
"myscript" 38L, 612C written                    21,33-47    All
```

*Fix task in 1st window by using an absolute pathname then save with **<esc>:w***

*Re-run **myscript** in the 2nd second window and test your change. To do this quickly hit **Ctrl-C** then <**up arrow**> key.*

```
                CIS 90 Final Project
        1) Hacking with the grep command
        2) Task 2
        3) Task 3
        4) Task 4
        5) Task 5
        6) Exit

        Enter Your Choice: 1
/home/cis90/rodduk/poems/Shakespeare/sonnet1:That thereby beauty's rose might ne
ver die,
/home/cis90/rodduk/poems/Shakespeare/sonnet10:  That beauty still may live in th
ine or thee.
/home/cis90/rodduk/poems/Shakespeare/sonnet11:Herein lives wisdom, beauty, and i
ncrease;
/home/cis90/rodduk/poems/Shakespeare/sonnet17:If I could write the beauty of you
r eyes,
/home/cis90/rodduk/poems/Shakespeare/sonnet2:And dig deep trenches in thy beauty
's field,
/home/cis90/rodduk/poems/Shakespeare/sonnet2:Then being ask'd, where all thy bea
uty lies,
/home/cis90/rodduk/poems/Shakespeare/sonnet2:How much more praise deserv'd thy b
eauty's use,
/home/cis90/rodduk/poems/Shakespeare/sonnet2:Proving his beauty by succession th
ine.
/home/cis90/rodduk/poems/Shakespeare/sonnet4:Upon thyself thy beauty's legacy?
/home/cis90/rodduk/poems/Shakespeare/sonnet4:  Thy unus'd beauty must be tomb'd
 with thee,
/home/cis90/rodduk/poems/Shakespeare/sonnet5:Beauty's effect with beauty were be
reft,
/home/cis90/rodduk/poems/Shakespeare/sonnet7:Yet mortal looks adore his beauty s
till,
/home/cis90/rodduk/poems/Shakespeare/sonnet9:But beauty's waste hath in the worl
d an end,
/home/cis90/rodduk/poems/Yeats/old:And loved your beauty with love false or true
,
Hit the Enter key to return to menu
```

Fix worked!

```
poems/Shakespeare/sonnet5:Beauty's effect with beauty were bereft,
poems/Shakespeare/sonnet7:Yet mortal looks adore his beauty still,
poems/Shakespeare/sonnet9:But beauty's waste hath in the world an end,
poems/Yeats/old:And loved your beauty with love false or true,
/home/cis90/rodduk $
```

117

Add some interaction

Let's add some interaction

```
1)  # Commands for Task 1
    echo "Are you ready to search for beauty in the poems?"
    read dummy
    grep beauty /home/cis90/rodduk/poems/*/*
    ;;
```

And it works!

118

Try a new option on the command

Let's try the -h option and not print the leading file names

```
1) # Commands for Task 1
   echo "Are you ready to search for beauty in the poems?"
   read dummy
   grep -h beauty /home/cis90/rodduk/poems/*/*
   ;;
```

And it works!

119

Add a new feature

Let's add a count of the strings found now

```
1)  # Commands for Task 1
    echo "Are you ready to search for beauty in the poems?"
    read dummy
    grep -h beauty /home/cis90/rodduk/poems/*/*
    echo "Ready to count them?"
    read dummy
    grep -h beauty /home/cis90/rodduk/poems/*/* | wc -l
    ;;
```

```
#!/bin/bash
#
# menu: A s
#
while true
do
        cle
        ech

    1)
    2)
    3)
    4)
    5)
    6)

        Ent
        rea
    case $RESPONSE in
    1)      # Commands for Task 1
            echo "Are you ready to search for beauty in the poems?"
            read dummy
            grep -h beauty /home/cis90/rodduk/poems/*/*
            echo "Ready to count them?"
            read dummy
            grep -h beauty /home/cis90/rodduk/poems/*/* | wc -l
            ;;
    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
"myscript" 43L, 839C written                          26,53-67
```

```
rodduk90@oslab:~/bin

                    CIS 90 Final Project
            1) Hacking with the grep command
            2) Task 2
            3) Task 3
            4) Task 4
            5) Task 5
            6) Exit

            Enter Your Choice: 1
Are you ready to search for beauty in the poems?

That thereby beauty's rose might never die,
        That beauty still may live in thine or thee.
Herein lives wisdom, beauty, and increase;
If I could write the beauty of your eyes,
And dig deep trenches in thy beauty's field,
Then being ask'd, where all thy beauty lies,
How much more praise deserv'd thy beauty's use,
Proving his beauty by succession thine.
Upon thyself thy beauty's legacy?
        Thy unus'd beauty must be tomb'd with thee,
Beauty's effect with beauty were bereft,
Yet mortal looks adore his beauty still,
But beauty's waste hath in the world an end,
And loved your beauty with love false or true,
Ready to count them?

14
Hit the Enter key to return to menu
```

120

How many points so far?

Let's score our mini-script so far

```
1)  # Commands for Task 1
    echo "Are you ready to search for beauty in the poems?"
    read dummy
    grep -h beauty /home/cis90/rodduk/poems/*/*
    echo "Ready to count them?"
    read dummy
    grep -h beauty /home/cis90/rodduk/po
    ;;
```

Implementing all five tasks (6 points each):
- Requirements for each task:

NO –Minimum of 10 "original" script command lines
NO –Has one or more non-generic comments to explain what it is doing
✔ –Has user interaction

You don't have to do all of these but do at least five:
- Redirecting stdin (5 points)
- Redirecting stdout (5 points)
- Redirecting stderr (5 points)
- Use of permissions (5 points)
✔ • Use of filename expansion characters (5 points)
✔ • Use of absolute path (5 points)
- Use of relative path (5 points)
- Use of a PID (5 points)
- Use of inodes (5 points)
- Use of links (5 points)
- Use of scheduling (5 points)
- Use of a GID or group (5 points)
- Use of a UID or user (5 points)
- Use of a /dev/tty device (5 points)
- Use of a signal (5 points)
✔ • Use of piping (5 points)
- Use of an environment variable (5 points)
- Use of /bin/mail (5 points)
- Use of a conditional (5 points)

The maximum for this section is 25 points.

121

Make another enhancement

Enhance script to let user specify search string and use color

```
1) # Commands for Task 1
   echo "Are you ready to search for beauty in the poems?"
   read dummy
   grep -h beauty /home/cis90/rodduk/poems/*/*
   echo "Ready to count them?"
   read dummy
   grep -h beauty /home/cis90/rodduk/poems/*/* | wc -l
   echo "Enter a new string to search for"
   read string
   echo searching for '"'$string'"'
   grep -h --color $string /home/cis90/rodduk/poems/*/*
   ;;
```

```
read dummy
grep -h beauty /home/cis90/rodduk/poems/*/*
echo "Ready to count them?"
read dummy
grep -h beauty /home/cis90/rodduk/poems/*/* | wc -l
echo "Enter a new string to search for"
read string
echo searching for '"'$string'"'
grep -h --color $string /home/cis90/rodduk/poems/*/*
;;
:
```

```
        5) Task 5
        6) Exit

        Enter Your Choice: 1
Are you ready to search for beauty in the poems?

That thereby beauty's rose might never die,
        That beauty still may live in thine or thee.
Herein lives wisdom, beauty, and increase;
If I could write the beauty of your eyes,
And dig deep trenches in thy beauty's field,
Then being ask'd, where all thy beauty lies,
How much more praise deserv'd thy beauty's use,
Proving his beauty by succession thine.
Upon thyself thy beauty's legacy?
        Thy unus'd beauty must be tomb'd with thee,
Beauty's effect with beauty were bereft,
Yet mortal looks adore his beauty still,
But beauty's waste hath in the world an end,
And loved your beauty with love false
Ready to count them?

14
Enter a new string to search for
sweet
searching for "sweet"
Thyself thy foe, to thy sweet self too cruel.
To show me worthy of thy sweet respect:
To thy sweet will making addition thus.
Thou of thyself thy sweet self dost deceive,
        Leese but their show, their substance still lives sweet.
Hit the Enter key to return to menu
```

And it works!

122

Check the score again

Let's re-score modified script

```
1)  # Commands for Task 1
    echo "Are you ready to search for beauty in the poems?"
    read dummy
    grep -h beauty /home/cis90/rodduk/poems/*/*
    echo "Ready to count them?"
    read dummy
    grep -h beauty /home/cis90/rodduk/p
    echo "Enter a new string to search f
    read string
    echo searching for '"'$string'"'
    grep -h --color $string /home/cis90/
    ;;
```

Implementing all five tasks (6 points each):
- Requirements for each task:
  - ✔ –Minimum of 10 "original" script command lines
  - **NO** –Has one or more non-generic comments to explain what it is doing
  - ✔ –Has user interaction

You don't have to do all of these but do at least five:
- Redirecting stdin (5 points)
- Redirecting stdout (5 points)
- Redirecting stderr (5 points)
- Use of permissions (5 points)
- ✔ Use of filename expansion characters (5 points)
- ✔ Use of absolute path (5 points)
- Use of relative path (5 points)
- Use of a PID (5 points)
- Use of inodes (5 points)
- Use of links (5 points)
- Use of scheduling (5 points)
- Use of a GID or group (5 points)
- Use of a UID or user (5 points)
- Use of a /dev/tty device (5 points)
- Use of a signal (5 points)
- ✔ Use of piping (5 points)
- Use of an environment variable (5 points)
- Use of /bin/mail (5 points)
- Use of a conditional (5 points)

The maximum for this section is 25 points.

123

Bing - one task done that meets minimum requirements!

Add some comments to help others understand what you are doing

```
1)      # Task 1 - grep command explored

        # Simple grep for "beauty"
        echo "Are you ready to search for beauty in the poems?"
        read dummy
        grep -h beauty /home/cis90/rodduk/poem

        # Same as before but counts matches to
        echo "Ready to count them?"
        read dummy
        grep -h beauty /home/cis90/rodduk/poem

        # Prompt user to supply search string
        echo "Enter a new string to search for
        read string
        echo searching for '"'$string'"'
        grep -h $string /home/cis90/rodduk/poe
        ;;
```

Implementing all five tasks (6 points each):
- Requirements for each task:
  - ✔ –Minimum of 10 "original" script command lines
  - ✔ –Has one or more non-generic comments to explain what it is doing
  - ✔ –Has user interaction

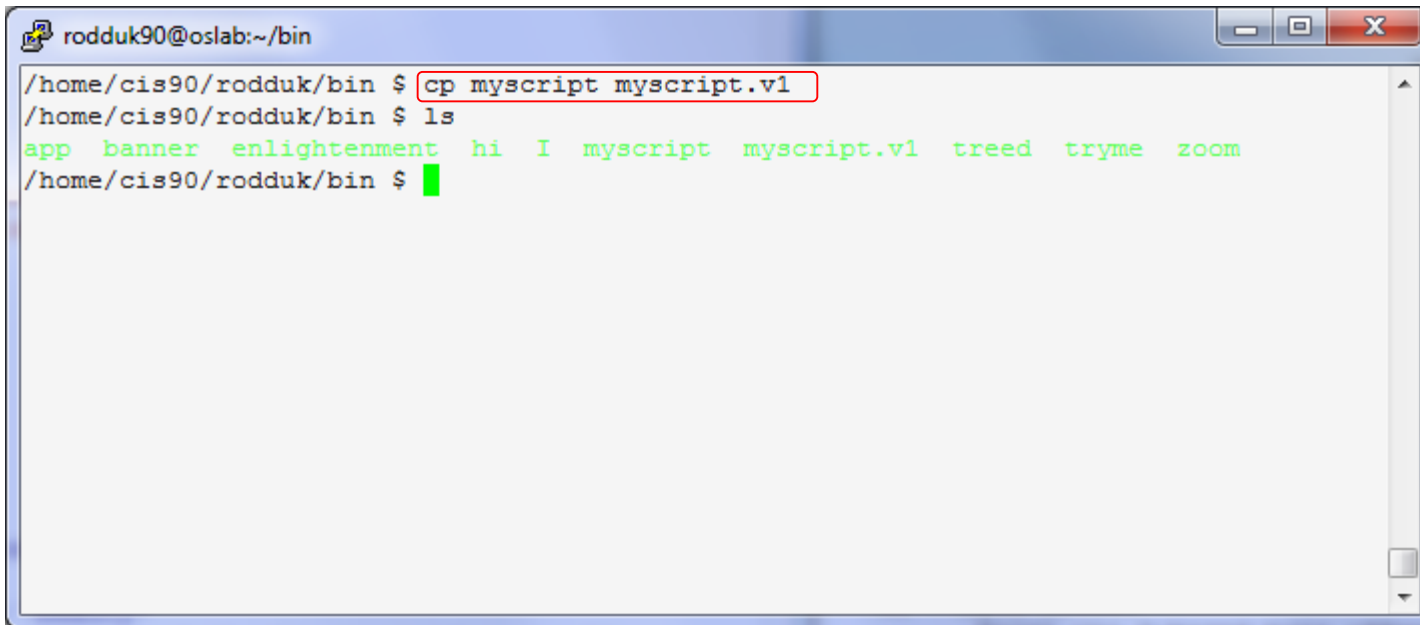You don't have to do all of these but do at least five:
- Redirecting stdin (5 points)
- Redirecting stdout (5 points)
- Redirecting stderr (5 points)
- Use of permissions (5 points)
- ✔ Use of filename expansion characters (5 points)
- ✔ Use of absolute path (5 points)
- Use of relative path (5 points)
- Use of a PID (5 points)
- Use of inodes (5 points)
- Use of links (5 points)
- Use of scheduling (5 points)
- Use of a GID or group (5 points)
- Use of a UID or user (5 points)
- Use of a /dev/tty device (5 points)
- Use of a signal (5 points)
- ✔ Use of piping (5 points)
- Use of an environment variable (5 points)
- Use of /bin/mail (5 points)
- Use of a conditional (5 points)

The maximum for this section is 25 points.

And has fulfilled three of the five requirements for the overall project!

Backup your work!

cp myscript myscript.v1 *after first day of work*
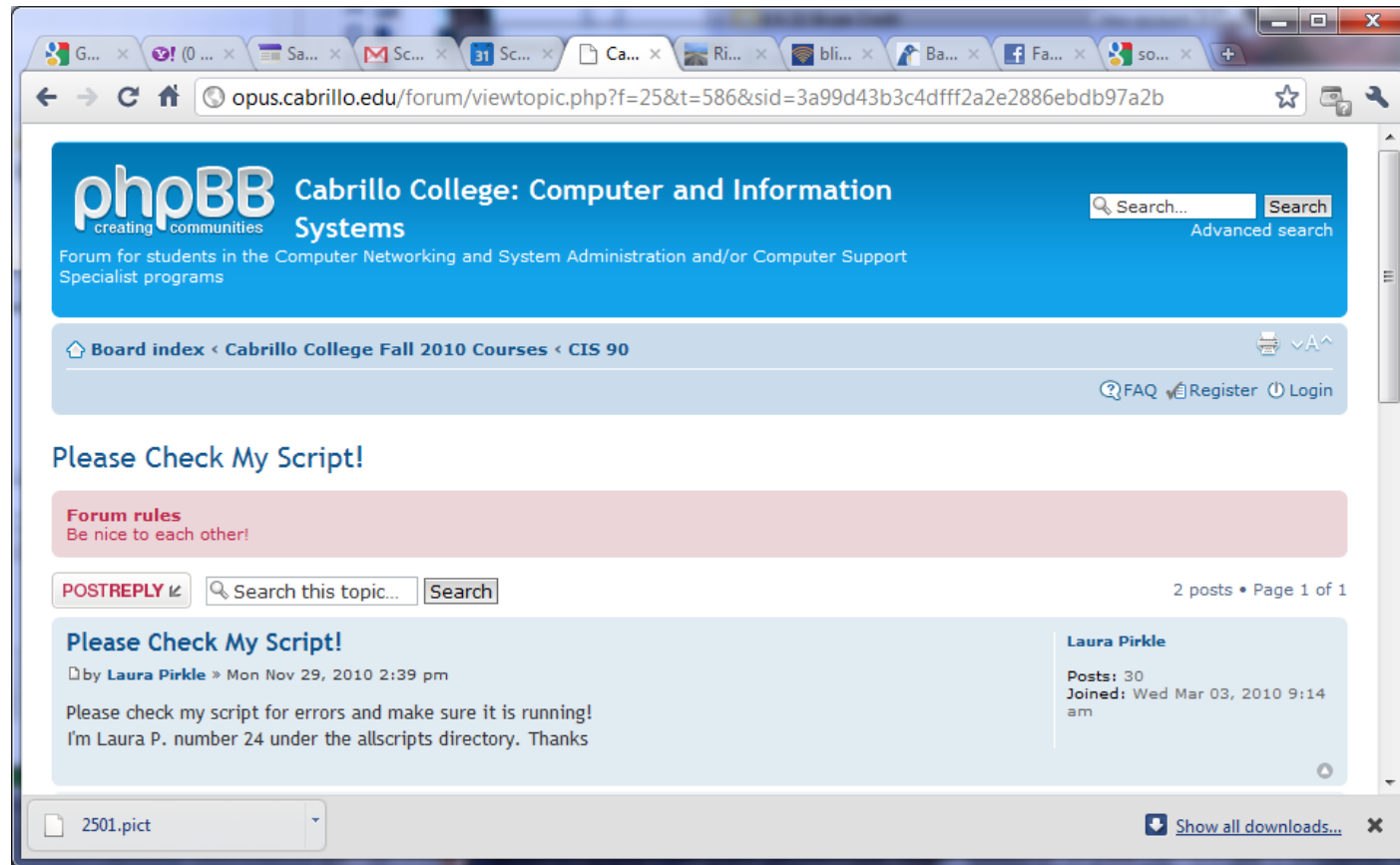

cp myscript myscript.v2 *after second day of work*
cp myscript myscript.v3 *and so on …*
cp myscript myscript.v4

*Always be able to revert back to an earlier version in case you
clobber the current one!*

125

Testing your script

The ask others on the forum to check your script and give you feedback

Plan extra time for:

- Figuring our how to do what you really want to do!

- Removing syntax errors

- Removing logic errors

- Posting script code on the forum and asking others to view it and suggest how to fix it

- Sleeping on it

Don't wait till the last minute to start your project!

Final Project forum tips

Use the forum effectively to get scripting help

Not so good …

> **Preview:**
> Help!
>
> My script is getting weird error
>
> - Homer

Not enough information has been provided on this post for others to help

Use the forum effectively to get scripting help

Better … but requires viewer to log into Opus-II and you may have modified the script since posting

Preview:
Help!

My script is getting weird error

My script is here:
/home/cis90/milhom/bin/myscript

And this is the error:

CODE: SELECT ALL

```
/home/cis90/simben/bin $ ./script99
simben90
-rwxr-x--- 1 simben90 cis90 10489 Apr 30 07:33 /home/cis90/simben/bin/myscript
./script99: line 8: unexpected EOF while looking for matching `"'
./script99: line 16: syntax error: unexpected end of file
/home/cis90/simben/bin $
```

- Homer

This post provides the location of the script and the error message which enables others to help you find and fix the problem

Use the forum effectively to get scripting help

B | *i* | u | Quote | Code | List | List= | [*] | Img | URL | Flash | Normal ▾ | Font colour
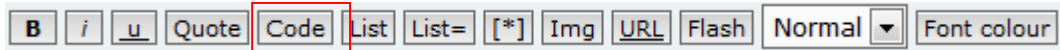
Preview:

Help!

My script is getting weird error

This is the script:

CODE: SELECT ALL

```
#!/bin/bash
# Test script
#
echo $LOGNAME
dir=/home/cis90/simben
ls -l $dir/bin/myscript
if [ -f "$dir/bin/myscript ]; then
  echo you have a myscript file in the bin directory
else
  echo there is no myscript file in your bin directory!]
fi
exit
```

And this is the error:

CODE: SELECT ALL

```
/home/cis90/simben/bin $ ./script99
simben90
-rwxr-x--- 1 simben90 cis90 10489 Apr 30 07:33 /home/cis90/simben/bin/myscript
./script99: line 8: unexpected EOF while looking for matching `"'
./script99: line 16: syntax error: unexpected end of file
/home/cis90/simben/bin $
```

- Homer

Best …

This post shows both the script and the error using code tags which enables others to help you find and fix the problem.

131

Scripting Tips echo

Silence is golden … but not always

Many UNIX commands that run successfully produce no output

```
[simben90@opus bin]$ alias probe=file
[simben90@opus bin]$ cp quiet quiet.bak
[simben90@opus bin]$ value=002
[simben90@opus bin]$ umask $value
[simben90@opus bin]$ cat quiet > /dev/null
[simben90@opus bin]$ > important_file$$
```

Note there is a variable named $ which gets set to the PID of your process.

133

Silence is golden … but not always

Running or sourcing a script full of UNIX commands that produce no output …. still produces no output!

```
[simben90@opus bin]$ cat quiet
alias probe=file
cp quiet quiet.bak
value=002
umask $value
cat quiet > /dev/null
> important_file$$

[simben90@opus bin]$ quiet
[simben90@opus bin]$


[simben90@opus bin]$ source quiet
[simben90@opus bin]$
```

Silence is golden ... but not always

```
[simben90@opus bin]$ cat not-so-quiet
echo TRACE: Entering not-so-quiet script
echo Press Enter to create probe alias
read dummy
alias probe=file
echo probe alias created, try: probe letter
cp quiet quiet.bak
value=002
umask $value
echo TRACE value=$value
cat quiet > /dev/null
> important_file$$
echo Warning: the file named important_file$$ was just created or emptied
echo TRACE: Exiting not-so-quiet script
```

You can use the echo command in your scripts to provide:
- *Interaction*
- *User feedback*
- *Tracing for debugging*

```
[simben90@opus bin]$ not-so-quiet
TRACE: Entering not-so-quiet script   Debugging
Press Enter to create probe alias   Interaction

probe alias created, try: probe letter   User feedback
TRACE value=002   Debugging
Warning: the file named important_file29538 was just created or emptied   User feedback
TRACE: Exiting not-so-quiet script   Debugging
```

135

tips
on
script names

Don't name your scripts "script"

```
[simben90@opus bin]$ ls -l script
-rwxr-x--- 1 simben90 cis90 47 Nov 23 16:44 script


[simben90@opus bin]$ cat script
echo "Hello from the script file named script"
```

What would happen if your ran the script above?

137

Don't name your scripts "script"

```
[simben90@opus bin]$ cat script
echo "Hello from the script file named script"

[simben90@opus bin]$ script
Script started, file is typescript
```

Why the heck doesn't my script do what it's supposed to do?

Don't name your scripts "script"

```
[simben90@opus bin]$ cat script
echo "Hello from the script file named script"

[simben90@opus bin]$ script
Script started, file is typescript
```

Why the heck doesn't my script do what it's supposed to do?

```
[simben90@opus bin]$ Where is my script?
bash: Where: command not found
[simben90@opus bin]$ exit
Script done, file is typescript
[simben90@opus bin]$ cat typescript
Script started on Wed 13 May 2009 08:00:02 AM PDT
[simben90@opus bin]$ Where is my script?
bash: Where: command not found
[simben90@opus bin]$ exit

Script done on Wed 13 May 2009 08:00:47 AM PDT
[simben90@opus bin]$
```

139

Don't name your scripts "script"

Why doesn't script do what it is supposed to do? … because script is the name of an existing UNIX command!

```
[simben90@opus bin]$ man script
[simben90@opus bin]$
```


140

Don't name your scripts "script"

There are (at least) two files named script on Opus

```
[simben90@opus bin]$ type script
script is hashed (/usr/bin/script)
[simben90@opus bin]$ file /usr/bin/script
/usr/bin/script: ELF 32-bit LSB executable, Intel 80386, version 1
(SYSV), for GNU/Linux 2.6.9, dynamically linked (uses shared libs),
for GNU/Linux 2.6.9, stripped


[simben90@opus bin]$ type /home/cis90/simben/bin/script
/home/cis90/simben/bin/script is /home/cis90/simben/bin/script
[simben90@opus bin]$ file /home/cis90/simben/bin/script
/home/cis90/simben/bin/script: ASCII text
[simben90@opus bin]$
```

*Question: Why did bash run the script in /usr/bin instead of the
script in /home/cis90/simben/bin?*

Don't name your scripts "script"

Question: Why did bash run the script in /usr/bin instead of the script in /home/cis90/simben/bin?

*The Linux **script** command is in this directory*

```
[simben90@opus bin]$ echo $PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/home/cis90/bin:
/home/cis90/simben/bin:.
```

*Our script, named **script**, is in this directory*

Answer: bash searches the path in the order the directories are listed. It finds the script command in /user/bin first.

142

Don't name your scripts "script"

To override the PATH you can always specify an absolute pathname to the file you want to run:

```
[simben90@opus bin]$ /home/cis90/simben/bin/script
Hello from the script file named script

[simben90@opus bin]$ ./script
Hello from the script file named script
```

Note the shell treats the . above as "here" which in this case is /home/cis90/simben/bin

143

Try the script command

- Use the **script** command to start recording

- Type various commands of your choice

- Type **exit** or hit **Ctrl-D** to end recording

- Use **cat typescript** to see what you recorded

This would be a good way to record a session such as working one of the lab assignments for future reference.

When finished type "done" in the chat window

Scripting Tips
color

Using Color

Black 0;30	Green 0;32	Red 0;31	Brown 0;33
Dark Gray 1;30	Light Green 1;32	Light Red 1;31	Yellow 1;33
Blue 0;34	Cyan 0;36	Purple 0;35	Light Gray 0;37
Light Blue 1;34	Light Cyan 1;36	Light Purple 1;35	White 1;37

```
/home/cis90/simben/bin $ echo -e "\e[00;31mMy favorite color is RED\e[00m"
My favorite color is RED
/home/cis90/simben/bin $ echo -e "\e[00;34mMy favorite color is BLUE\e[00m"
My favorite color is BLUE
/home/cis90/simben/bin $ echo -e "\e[00;32mMy favorite color is GREEN\e[00m"
My favorite color is GREEN
/home/cis90/simben/bin $
```

*Use **echo -e "\e[0**n;nn**m"** to turn on color and **\e[00m** to turn it off.*

(the -e option enables interpretation of backslash escapes)

Source: http://hacktux.com/bash/colors

146

Using Color

```
/home/cis90/simben/bin $ echo -e "\e[00;32m"          Change to
                                                      color green
/home/cis90/simben/bin $ head -4 ~/letter
Hello Mother!  Hello Father!

Here I am at Camp Granada.  Things are very entertaining,
and they say we'll have some fun when it stops raining.
/home/cis90/simben/bin $ echo -e '\e[00m'          Revert color
                                                   back to normal
/home/cis90/simben/bin $
```

Source: http://hacktux.com/bash/colors

147

Using Color


```
off="\e[00m"
red="\e[00;31m"
white="\e[01;37m"
blue="\e[00;34m"
echo -e $red RED $white WHITE $blue BLUE $off
 RED   WHITE   BLUE
echo -e ${red}RED ${white}WHITE ${blue}BLUE $off
RED WHITE BLUE
```

Demonstrating the use of variables and curly braces to make color easier to use.

Curly braces are used to clearly separate the variable name from adjacent text strings:
• $redRED is null
• ${red}RED is "\e[00;31mRED"

148

Class Exercise

Make a new script in your bin directory
```
cd bin
vi example4271
```

In vi add these lines to your script then save:
```
off="\e[00m"
green="\e[00;32m"
echo -e Hi there, you look a little ${green}GREEN${off} today!
```

Prepare and run your script
```
chmod +x example4271
example4271
```

Review

function runningScript ()
{

The rules of the road for variables

- Rule 1: A child process can only see variables the parent has exported.

- Rule 2: A child process cannot change the parent's variables.

Running a Script

```
/home/cis90/simben $ cat mydate
#!/bin/bash
echo "Hola $LOGNAME"
date +'%m/%d/%Y'
echo $myvar1 $myvar2 $myvar3
```
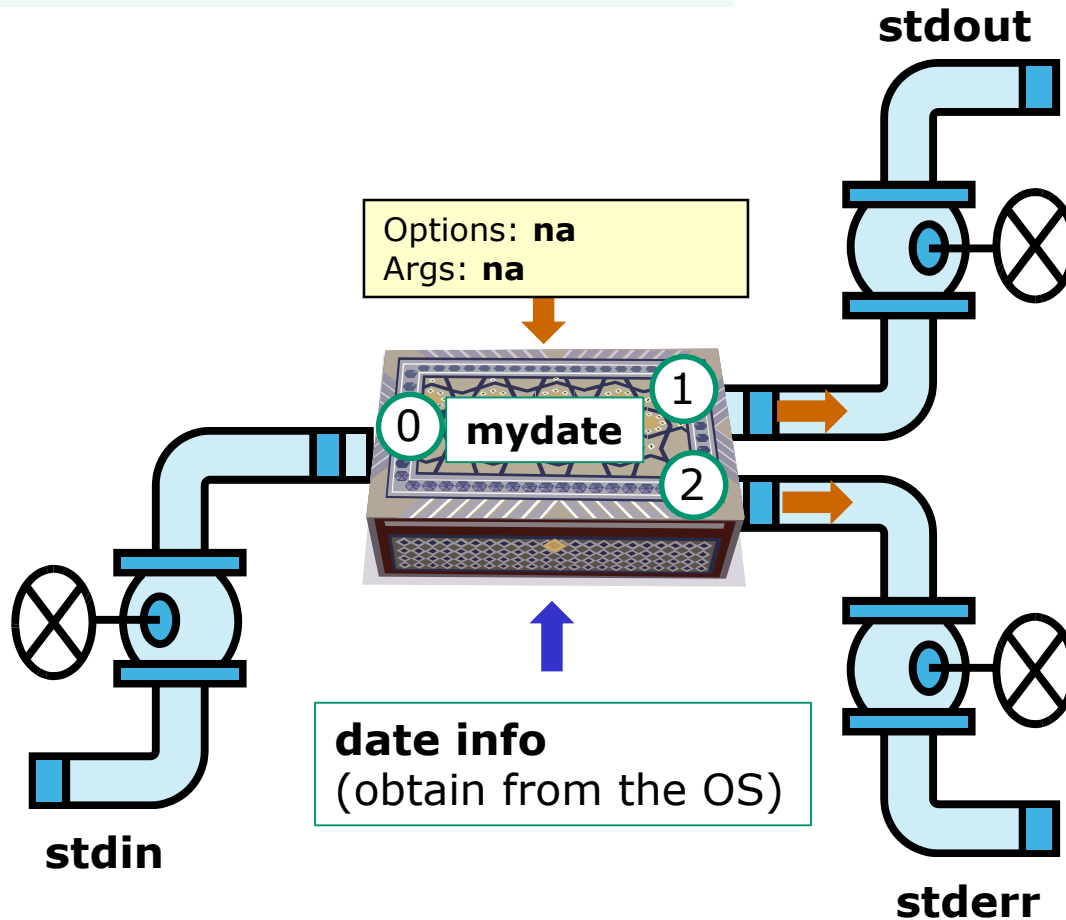
Add this line to the last script we made

Don't initialize them yet

```
/home/cis90/simben $ mydate
Hola simben90
05/16/2013

/home/cis90/simben $
```

Because the variables don't exist yet the last echo statement prints a blank line
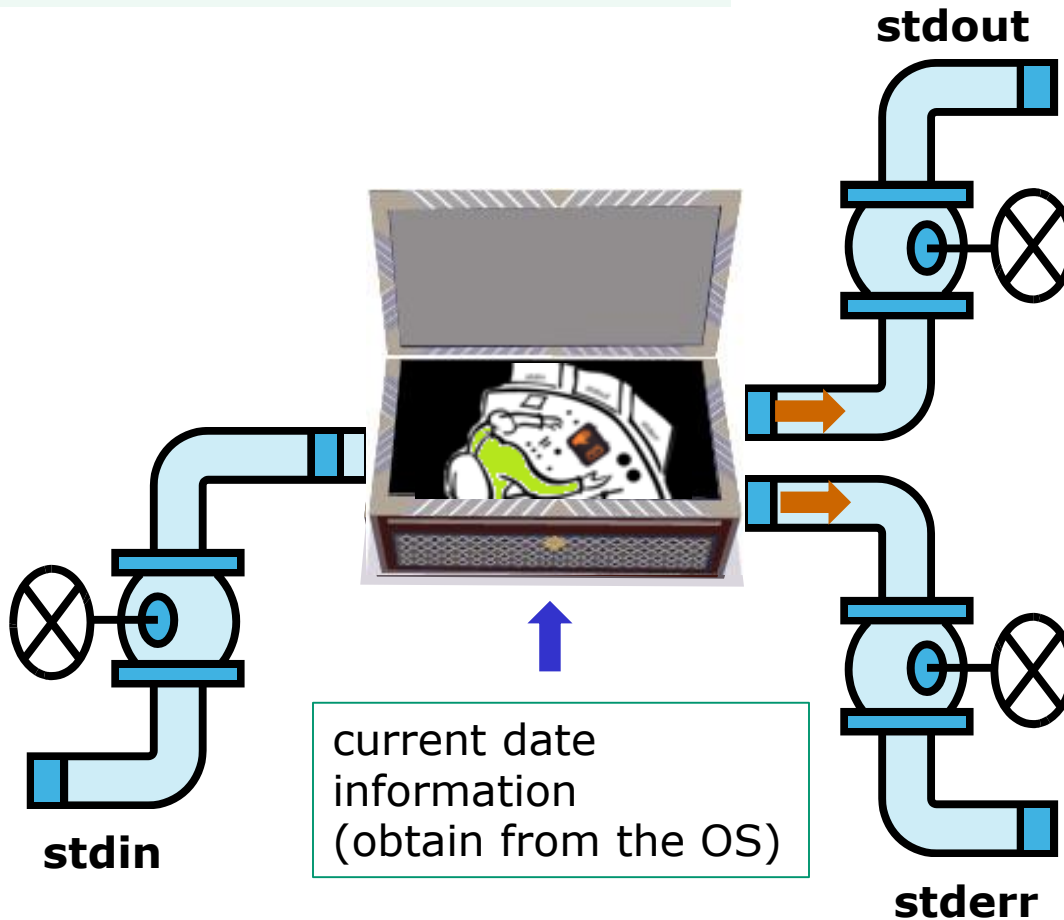
152

Running a Script

`$ mydate`

stdout

Hola simben90
05/09/2013

Options: **na**
Args: **na**

1

0   **mydate**

2

date info
(obtain from the OS)

stdin

stderr

*In this example, ouptput from **myscript** goes to stdout.*

stdout has not been redirected so it goes to the default terminal device (your screen).
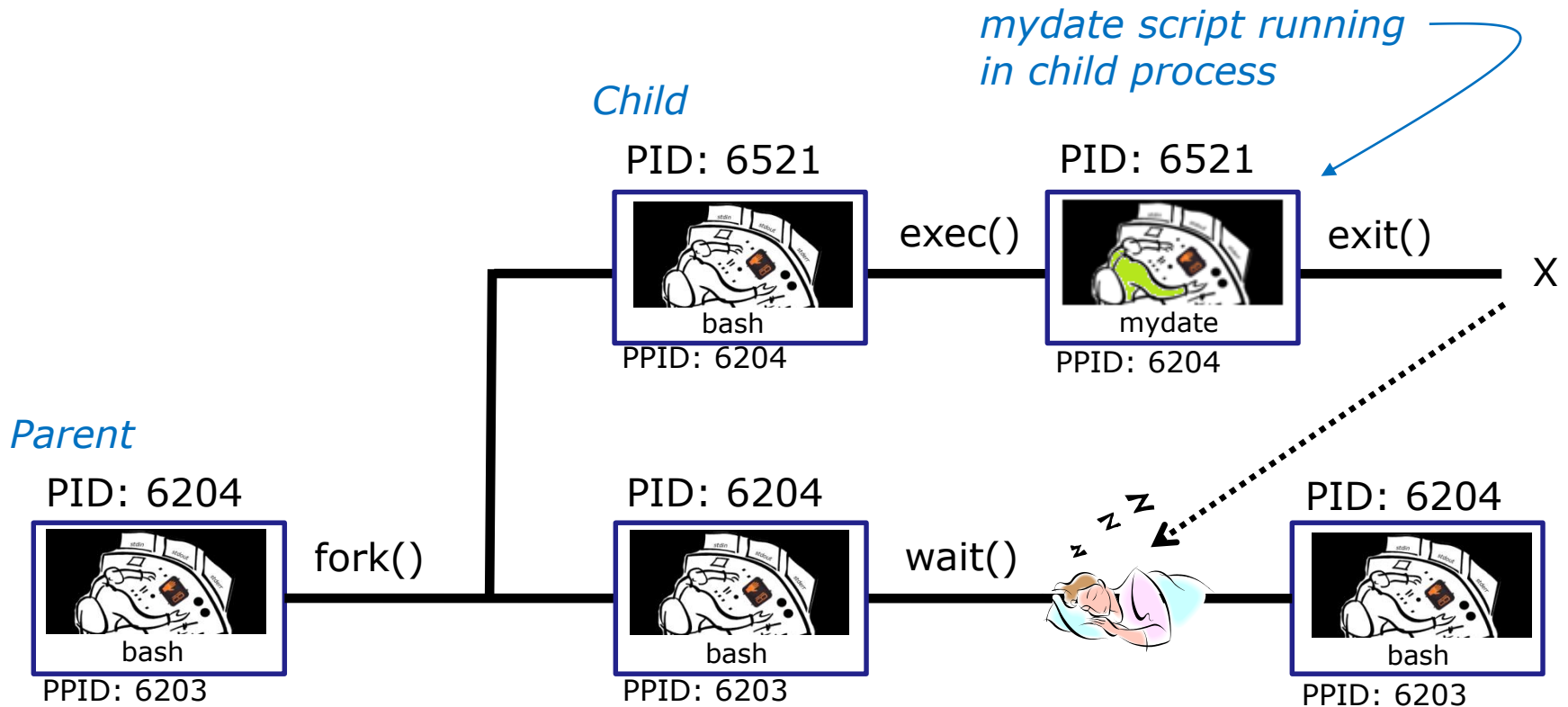
153

Running a Script

$ mydate

stdout

Hola simben90
05/16/2012

A sneak peek into memory to see what our process looks like!

current date
information
(obtain from the OS)

stdin

stderr

Running a Script

mydate script running in child process

Child

PID: 6521

bash
PPID: 6204

exec()

PID: 6521

mydate
PPID: 6204

exit()

X

Parent

PID: 6204

bash
PPID: 6203

fork()

PID: 6204

bash
PPID: 6203

wait()

z z z

PID: 6204

bash
PPID: 6203

*Whenever you run any command, program, or script it runs as a **child process***

155

Running a Script

```
/home/cis90/simben $ cat mydate
#!/bin/bash
echo "Hola $LOGNAME"
date +'%m/%d/%Y'
echo $myvar1 $myvar2 $myvar3
```

In the parent process, initialize the three variables

```
/home/cis90/simben $ myvar1=Tic; myvar2=Tac; myvar3=Toe
/home/cis90/simben $ echo $myvar1 $myvar2 $myvar3
Tic Tac Toe
```

*What happens if we run **mydate** now?*

Running a Script

```
/home/cis90/simben $ cat mydate
#!/bin/bash
echo "Hola $LOGNAME"
date +'%m/%d/%Y'
echo $myvar1 $myvar2 $myvar3

/home/cis90/simben $ myvar1=Tic; myvar2=Tac; myvar3=Toe
/home/cis90/simben $ echo $myvar1 $myvar2 $myvar3
Tic Tac Toe

/home/cis90/simben $ mydate
Hola simben90
05/09/2012

/home/cis90/simben $
```

*Running **mydate**
(as a child process)*

Why no Tic Tac Toe output?

157

Running a Script

```
/home/cis90/simben $ export myvar1
/home/cis90/simben $ mydate
Hola simben90
05/09/2012
Tic
```

Rule 1: A child process can only see variables the parent has exported

```
/home/cis90/simben $ export myvar2
/home/cis90/simben $ mydate
Hola simben90
05/09/2012
Tic Tac
```

```
/home/cis90/simben $ export myvar3
/home/cis90/simben $ mydate
Hola simben90
05/09/2012
Tic Tac Toe
```

158

Running a Script

```
/home/cis90/simben $ echo $myvar1 $myvar2 $myvar3
Tic Tac Toe

/home/cis90/simben $ cat mydate
#!/bin/bash
echo "Hola $LOGNAME"
date +'%m/%d/%Y'
echo $myvar1 $myvar2 $myvar3
myvar1=red myvar2=white myvar3=blue
echo $myvar1 $myvar2 $myvar3

/home/cis90/simben $ mydate
Hola simben90
05/09/2012
Tic Tac Toe
red white blue

/home/cis90/simben $ echo $myvar1 $myvar2 $myvar3
Tic Tac Toe
```

*Add these
new lines*

*Rule 2: A child process
cannot change the
parent's variables.*

Running a Script

Unless we want them to

```
/home/cis90/simben $ echo $myvar1 $myvar2 $myvar3
Tic Tac Toe

/home/cis90/simben $ source mydate
Hola simben90
05/09/2012
Tic Tac Toe
red white blue

/home/cis90/simben $ echo $myvar1 $myvar2 $myvar3
red white blue
```

Sourcing a script causes the instructions to be run in the parent process. A child process is not created

}
while не розумію
do
  runningScript
done

Printers

Two predominate types of printers
• Thermal inkjet technology
• Laser, drum, toner technology

So many ways to hook them up …

Now:
• Network
• USB
• Wireless (Bluetooth, IR)

Back then:
• Serial cable
• Parallel printer cable

Printing in Linux

Printing Commands

The ATT System V way
• lp (to print)
• lpstat (queue management)
• cancel (to remove jobs)

The BSD (Berkeley Software Distribution) way
• lpr (to print)
• lpq (queue management)   *BSD is a branch of UNIX that was*
• lprm (to remove jobs)   *developed at the University of California, Berkeley*

And now CUPS …
• Provides both System V and Berkeley based command-line interfaces
• Supports new Internet Printing Protocol
• Works with Samba

CUPS
lpstat command

Syntax: **lpstat** [*options*]

```
rsimms@hugo:~$ lpstat -p
printer HP_LaserJet_1320_series is idle.  enabled since Tue 08 May
2012 08:46:45 PM PDT
```

The –p option will show the available printers

```
rsimms@hugo:~$ lpstat -p -d
printer HP_LaserJet_1320_series is idle.  enabled since Tue 08 May
2012 08:46:45 PM PDT
system default destination: HP_LaserJet_1320_series
```

The –d option will identify the default printer

CUPS
lpstat command

On Opus

What printers are available on Opus?

Which is the default printer?

Write your answers in the chat window

CUPS
lp and lpr commands

*Use **lp** (or **lpr**) to print files*

```
/home/cis90/simben $ lp lab10
request id is hplaser-5 (1 file(s))

/home/cis90/simben $ lp -d hplaser lab10
request id is hplaser-6 (1 file(s))
```

*With **lp**, use the –d option to manually select the printer*

```
/home/cis90/simben $ lpr lab10

/home/cis90/simben $ lpr -P hplaser lab10
```

*With **lpr**, use the –P option to manually select a printer*

171

CUPS
lp and lpr commands

```
/home/cis90/simben $ echo "Print Me Quietly" | lpr -P hplaser
/home/cis90/simben $
```

Note that both lp and lpr will read from stdin.

This allows output from another command to be piped in

172

CUPS
Practice Printing

On Opus, print your lab10 and letter files

```
lp lab10
lpstat


lpr letter
lpstat


echo "Print Me Quietly" | lpr -P hplaser
lpstat
```

When finished type "done" in the chat window

Managing Print Jobs

CUPS
Showing jobs waiting to print

```
[root@benji ~]# lpq
hp7550 is not ready
Rank     Owner    Job      File(s)
Total Size
1st      root     22       myfile
1024 bytes
2nd      root     23       myfile
1024 bytes
3rd      root     24       myfile
1024 bytes
4th      root     25       myfile
1024 bytes
```

*Use **lpq** or **lpstat** with no options to show spooled print jobs*

```
[root@benji ~]# lpstat
hp7550-22                   root              1024    Sat
15 Nov 2008 12:20:23 PM PST
hp7550-23                   root              1024    Sat
15 Nov 2008 12:20:28 PM PST
hp7550-24                   root              1024    Sat
15 Nov 2008 12:20:31 PM PST
hp7550-25                   root              1024    Sat
15 Nov 2008 12:20:34 PM PST
```

177

CUPS
Removing/canceling pending print jobs

```
[root@benji ~]# lpq
hp7550 is not ready
Rank    Owner    Job      File(s)
Total Size
1st     root     22       myfile
1024 bytes
2nd     root     23       myfile
1024 bytes
3rd     root     24       myfile
1024 bytes
4th     root     25       myfile
1024 bytes

[root@benji ~]# cancel 22
[root@benji ~]# cancel 23
[root@benji ~]# lprm 24
[root@benji ~]# lprm 25

[root@benji ~]# lpq
hp7550 is not ready
no entries

[root@benji ~]# lpstat
[root@benji ~]#
```

*Use **cancel** or **lprm**
to remove print jobs*

178

CUPS
Practice Printing

On Opus

```
lpq
lpstat

cancel <print job number>
lpq

lprm <print job number>
lpq
```

When finished type "done" in the chat window

Assignment

Start your project!

Start early and finish on time!

187

Wrap up

Commands:

    lp, lpr   - Linux print command
    cancel, lprm   - cancel print job
    lpq, lpstat   - Show print queue

Web:
    http://hostname:631   - CUPS web based management utility
    http://hostname:9100   - HP JetDirect printer

Next Class

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

No Quiz
No Lab due

Work on final project - due in two weeks!

Optional extra credit labs

Project Workshop

• See if you can get one "starter" task scripted and working before leaving class today.

• Grade your starter script using the Final Project rubric

Implementing all five tasks (6 points each):
• Requirements for each task:
 −Minimum of 10 "original" script command lines
 −Has one or more non-generic comments to explain what it is doing
 −Has user interaction

You don't have to do all of these but do at least five:
• Redirecting stdin (5 points)
• Redirecting stdout (5 points)
• Redirecting stderr (5 points)
• Use of permissions (5 points)
• Use of filename expansion characters (5 points)
• Use of absolute path (5 points)
• Use of relative path (5 points)
• Use of a PID (5 points)
• Use of inodes (5 points)
• Use of links (5 points)
• Use of scheduling (5 points)
• Use of a GID or group (5 points)
• Use of a UID or user (5 points)
• Use of a /dev/tty device (5 points)
• Use of a signal (5 points)
• Use of piping (5 points)
• Use of an environment variable (5 points)
• Use of /bin/mail (5 points)
• Use of a conditional (5 points)
The maximum for this section is 25 points.

Backup