Lesson Module Checklist
- Slides
- WB

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

- LabX1 and Project posted

- Materials uploaded
- Backup slides, CCC info, handouts on flash drive
- Check that backup room headset is charged
- Spare 9v battery for mic

Introductions and Credits

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

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

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

CIS 90 - Lesson 13

Instructor: **Rich Simms**
Dial-in: **888-450-4821**
Passcode: **761867**

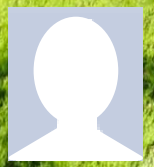

Buzz


Carlos


Emily


Jon M.


Jon W.


Jordan


Joseph


JJ


Kiernan


Maria


Mathew

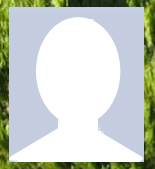

Mike C.


Michael F.


Mike M.


Nick L.

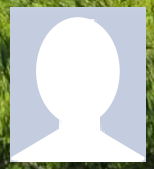

Rebecca


Ricardo


Robert


Steve


Tess


Tim


Troy

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

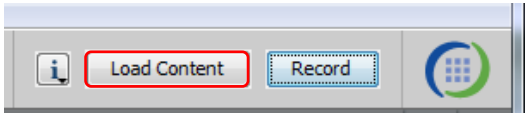
First Minute Quiz

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

THE LAST QUIZ!

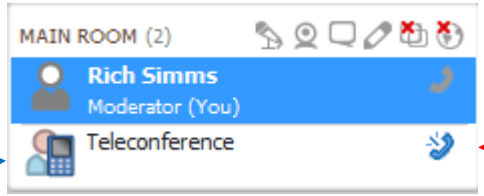
For credit email answers to:
risimms@cabrillo.edu
within the first few minutes of class

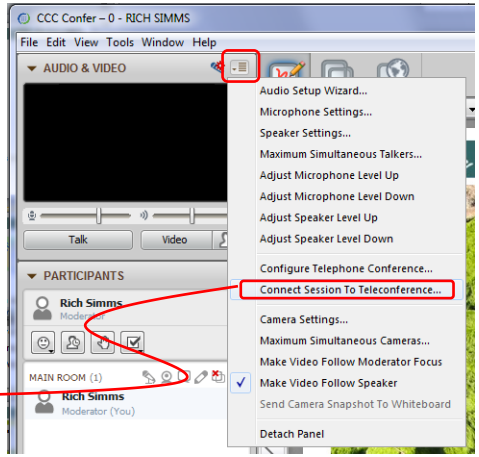
[ ] Preload White Board with cis*lesson??*-WB



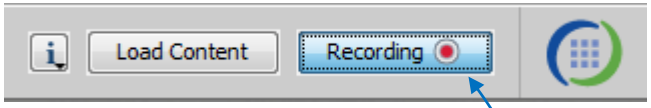
[ ] Connect session to Teleconference

Session now connected to teleconference





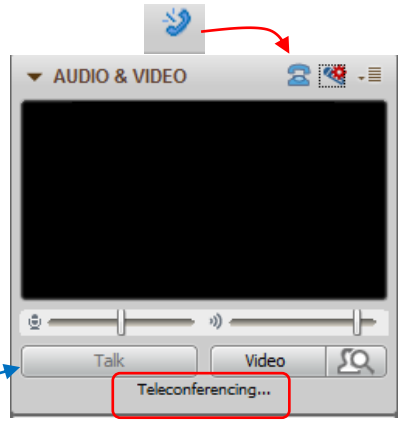
[ ] Is recording on?



Red dot means recording

[ ] Use teleconferencing, not mic

Should be greyed out

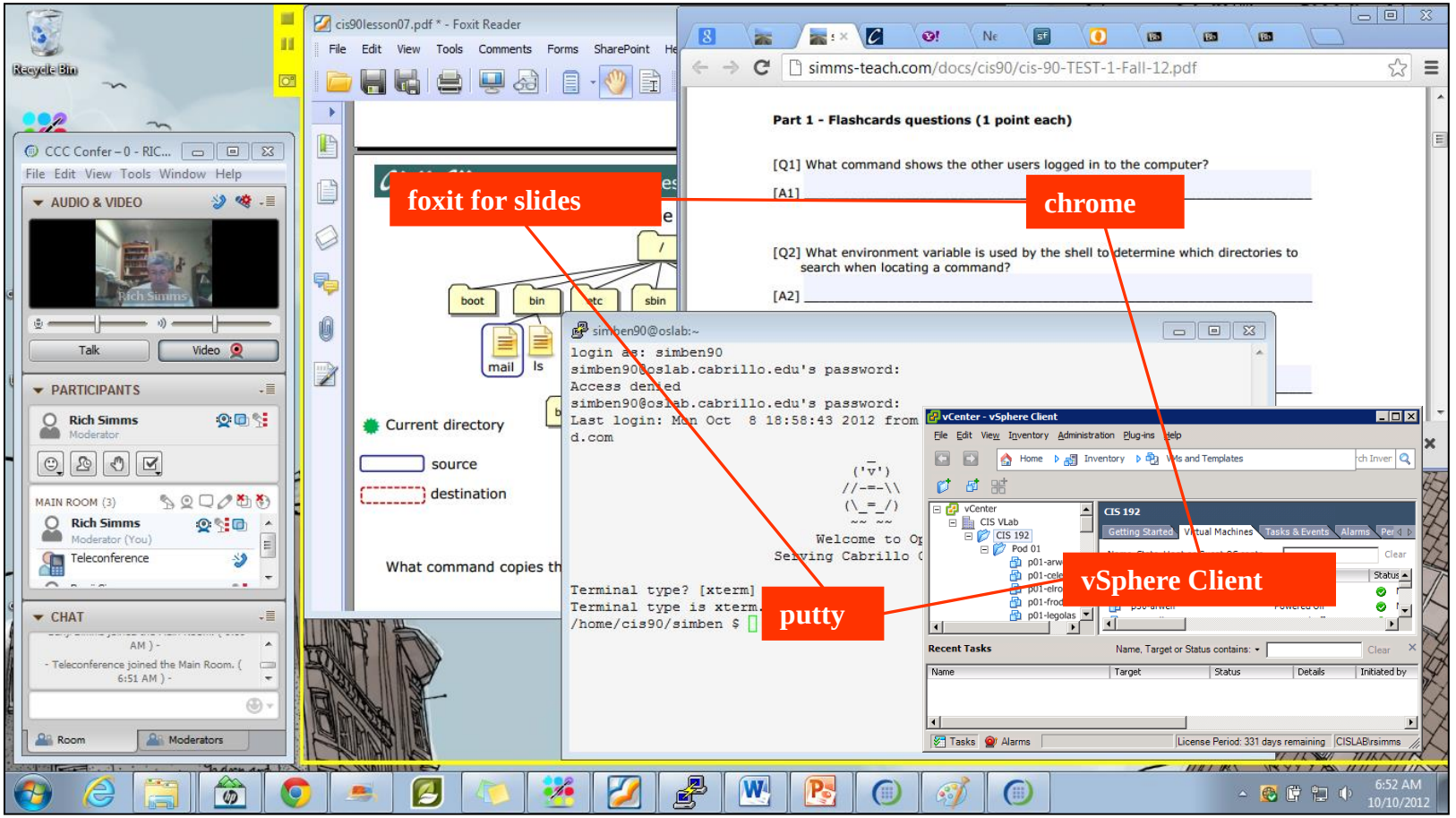


Keep wireless mic transmitter away from cell phone and podium if excess static occurs

CCC Confer

Don't Forget

[ ] layout and share apps

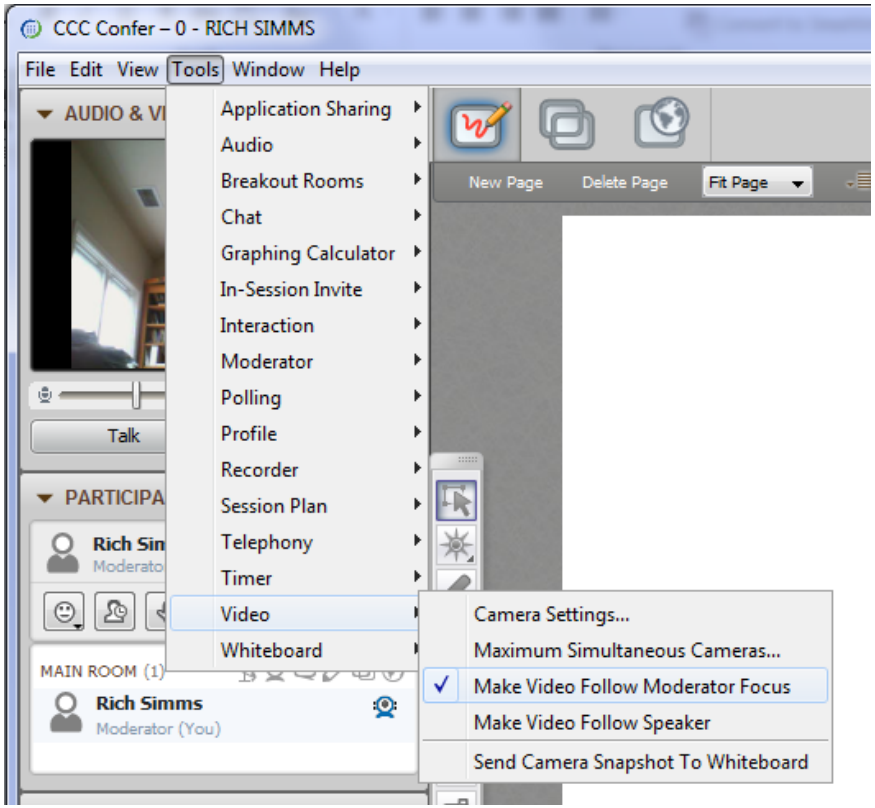


CCC Confer

[ ] Video (webcam) optional

[ ] Follow moderator

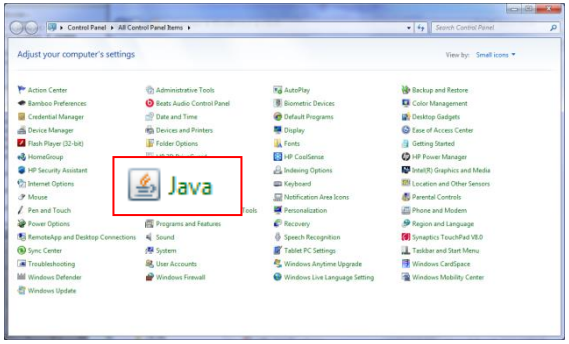
[ ] Double-click on postages stamps



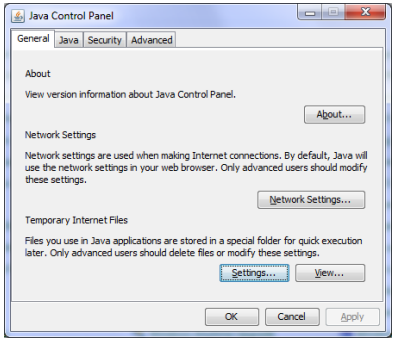
Universal Fix for CCC Confer:

1) Shrink (500 MB) and delete Java cache
2) Uninstall and reinstall latest Java runtime

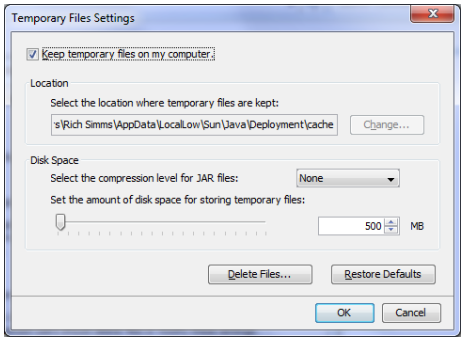
Control Panel (small icons)



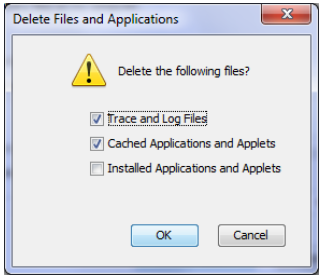
General Tab > Settings…



500MB cache size



Delete these



Google Java download

Shell Scripting and Printing

Objectives	Agenda
• Be able to print, view the print queue and cancel print jobs	• Quiz • Housekeeping • Refresh • Shell scripting • Printing

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.*

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

Breaking your path in Lab 10

The path (PATH) variable … a Review

- Lab 10 often results in clobbered paths and students may think all 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 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. What is the fourth directory on this path?
2. Can you name the first command found in this directory?

Put your answers in the chat window

The Path

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

The tty command is in the /usr/bin directory

```
/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

```
/home/cis90/simben $ PATH=$oldpath
/home/cis90/simben $ tty
/dev/pts/0
```

Restore your path to what it was

Class Activity: Your turn

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
```

The Path

If the path is not defined then each command to run must be specified using an absolute pathname

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

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

The 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 Path

Fixing the path, one directory at a time …

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

*The **ls** command is in /bin so lets put that on the path*

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

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

*The **stat** command is in /usr/bin so lets append that directory too*

```
/home/cis90/simben $ PATH=$PATH:/usr/bin
/home/cis90/simben $ stat letter
  File: `letter'
  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
```

The Path

```
/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:
```

The Path

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

*The **datecal** 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 $ datecal

Tue May  8 14:30:59 PDT 2012

     April 2012              May 2012               June 2012
Su Mo Tu We Th Fr Sa    Su Mo Tu We Th Fr Sa    Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7                1  2  3  4  5                1  2
 8  9 10 11 12 13 14     6  7  8  9 10 11 12     3  4  5  6  7  8  9
15 16 17 18 19 20 21    13 14 15 16 17 18 19    10 11 12 13 14 15 16
22 23 24 25 26 27 28    20 21 22 23 24 25 26    17 18 19 20 21 22 23
29 30                   27 28 29 30 31          24 25 26 27 28 29 30


/home/cis90/simben $
```

The Path

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

*The **dogbone** shell script is in the current directory but not on the path*

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

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

The Path

Easy … add "here" or "." 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
```

The Path

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

Start with /bin which has all the essential UNIX/Linux commands

```
/home/cis90/simben $ PATH=$PATH:/usr/bin
/home/cis90/simben $ echo $PATH
/bin:/usr/bin
```

Append /usr/bin which has hundreds of additional UNIX/Linux commands

```
/home/cis90/simben $ PATH=$PATH:/home/cis90/bin
/home/cis90/simben $ echo $PATH
/bin:/usr/bin:/home/cis90/bin
```

Append the CIS 90 class bin directory

```
/home/cis90/simben $ PATH=$PATH:/home/cis90/simben/bin
/home/cis90/simben $ echo $PATH
/bin:/usr/bin:/home/cis90/bin:/home/cis90/simben/bin
```

Append your own student bin directory

```
/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

The Path

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

. and exec

. and exec

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

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

. *myscript*
source *myscript* } *equivalent*

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

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

exec clear

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

grok that?

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.

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

rsimms@oslab:~

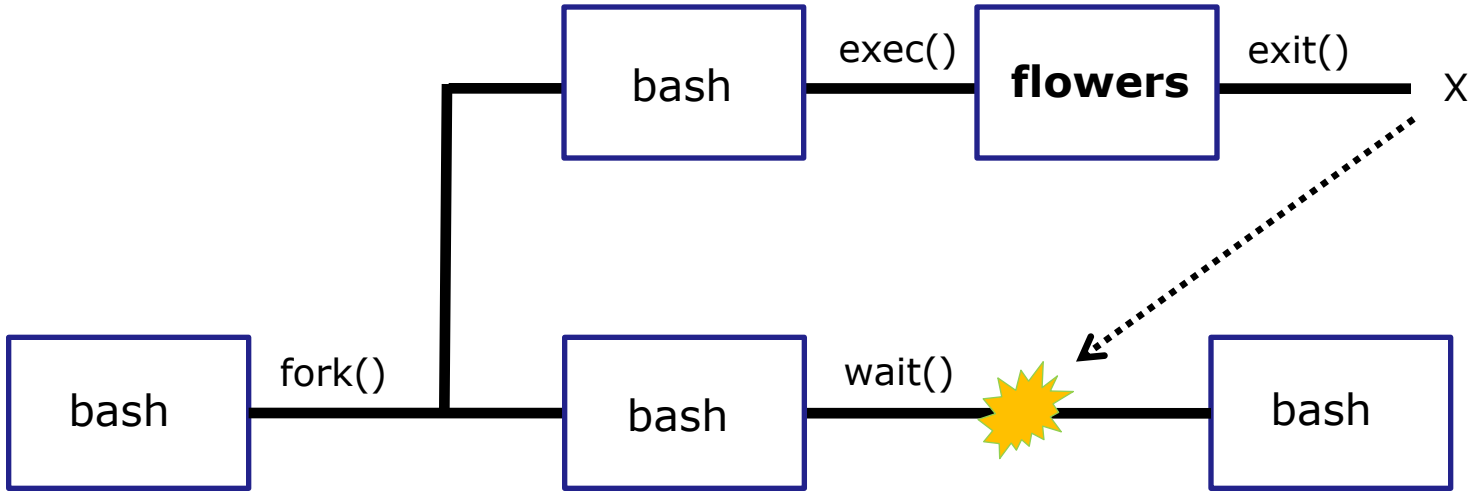
```
#!/bin/bash
#
# Useful alias:
#   alias go='echo roses are \"$roses\" and violets are \"$violets\"'
#
echo
echo "==> Entering child process <=="
ps -f
echo "==> showing variables in child <=="
echo "  " roses are '"'$roses'"'
echo "  " violets are '"'$violets'"'
echo "==> setting variables in child <=="
roses=black
violets=orange
echo "  " roses are '"'$roses'"'
echo "  " violets are '"'$violets'"'
echo "==> Leaving child process <=="
echo
~
                                                    1,1           All
```

You can copy and paste this alias to use today

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

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

running the flowers script



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

As a convenience create an alias to show variable values

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

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

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

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

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

The flowers script

```
#!/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

Create and initialize variables

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

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

Now the roses variable has been created and initialized

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

Now the violets variable has been created and initialized

Unset variables

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

Now the roses variable no longer exists

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

Now the violets variable no longer exists

Create and initialize variables again

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

Now both variables have been created and initialized again

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

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

/home/cis90/simben $ flowers

==> Entering child process <==
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 ""
   violets are ""
==> setting variables in child <==
   roses are "black"
   violets are "orange"
==> Leaving child process <==

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

The parent can view the values of the roses and violets variables

ps is a child of flowers, flowers is a child of bash

The child cannot view the values of the parent's non-exported variables (Rule #1)

The child sets the variables to black and orange

The parent's variables are unchanged (Rule #2)

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

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

/home/cis90/simben $ flowers

==> Entering child process <==
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 $ go
roses are "red" and violets are "blue"
```

The parent exports just the roses variable

ps is a child of flowers, flowers is a child of bash

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

The child sets the variables to black and orange

The variables are unchanged after running flowers script (Rule #2)

Run flowers script as a child process
(script sourced)

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

/home/cis90/simben $ source flowers

==> Entering child process <==
UID        PID  PPID  C STIME TTY          TIME CMD
simben90  4559 25106  0 17:35 pts/8    00:00:00 ps -f
simben90 25106 25059  0 17:16 pts/8    00:00:00 -bash
==> 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"
```

The parent sees roses and violets

flowers is NOT running as a child process

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

The script sets the variables to black and orange

The variables are CHANGED after sourcing the flowers script

```
/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 the usual way 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"
```

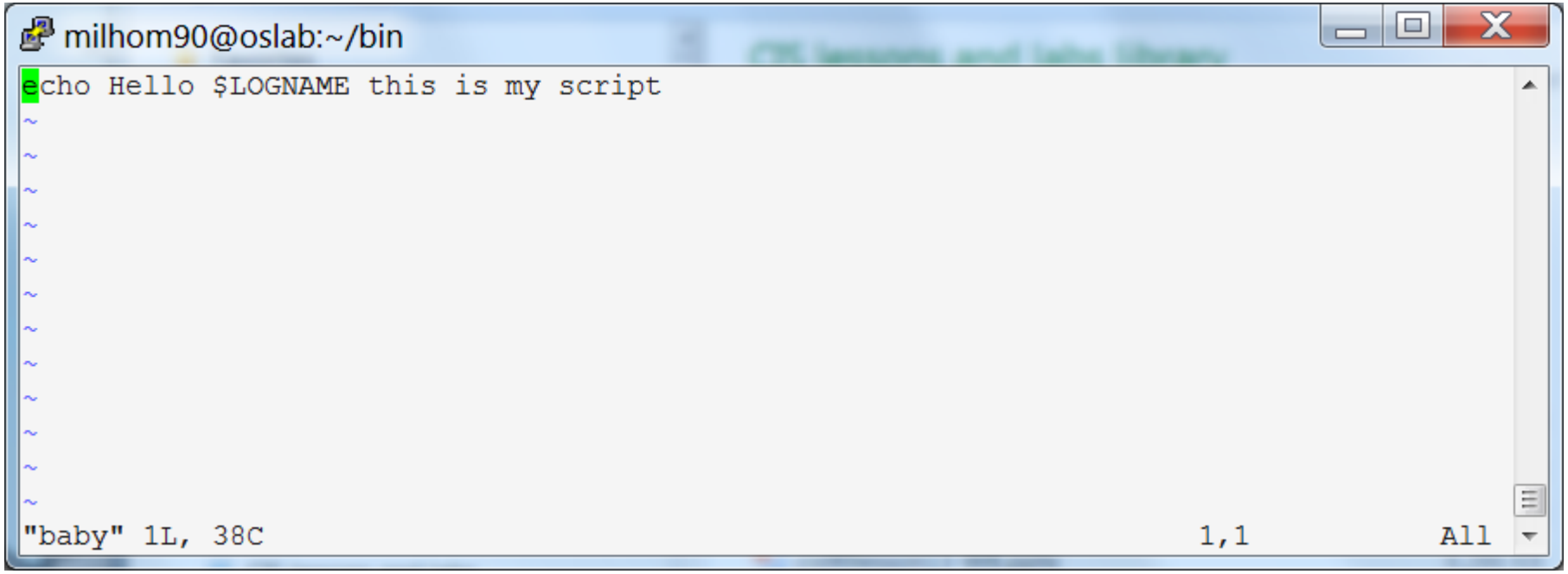
Scripting

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.

Shell Script Examples

```
/home/cis90/milhom/bin $ vim baby
```

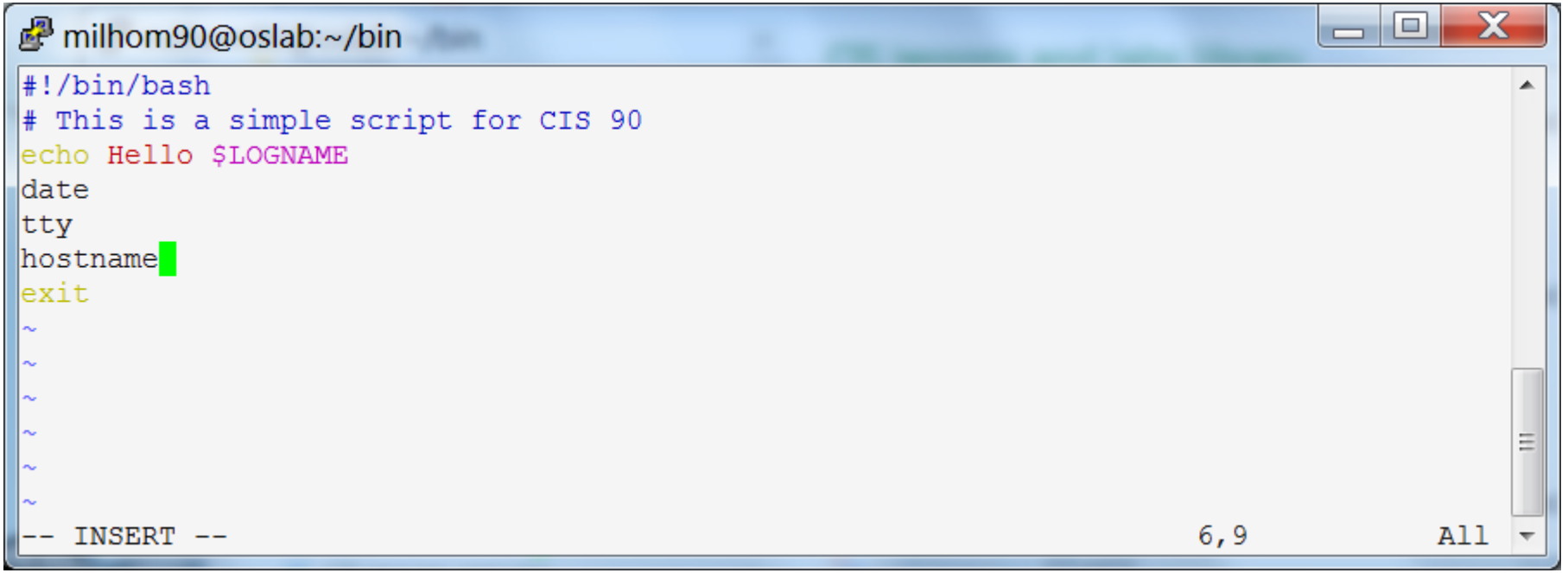


use Esc **:wq** *to save file and quit vi*

```
/home/cis90/milhom/bin $ chmod 750 baby
/home/cis90/milhom/bin $ baby
Hello milhom90 this is my script
```

Shell Script Examples

```
/home/cis90/milhom/bin $ vim toddler
```

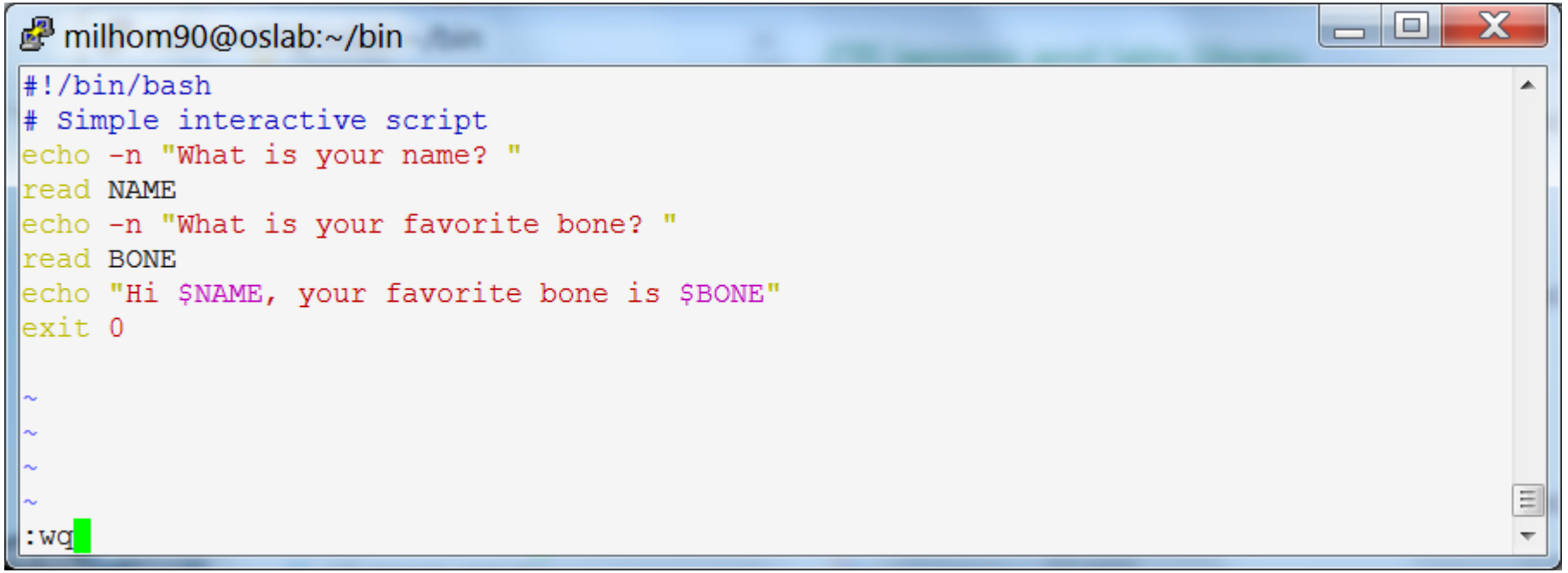


use Esc **:wq** *to save file and quit vi*

```
/home/cis90/milhom/bin $ chmod 750 toddler
/home/cis90/milhom/bin $ toddler
Hello milhom90
Mon Nov 25 17:57:15 PST 2013
/dev/pts/9
oslab.cishawks.net
```

Shell Script Examples

```
/home/cis90/milhom/bin $ vim dogbone
```



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

```
/home/cis90/milhom/bin $ vim dogbone
/home/cis90/milhom/bin $ chmod 750 dogbone
/home/cis90/milhom/bin $ dogbone
What is your name? Homer
What is your favorite bone? Turkey
Hi Homer, your favorite bone is Turkey
```

Final Project

Getting Started

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

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

3) Run the script:
   myscript

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

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
~
                                                              1,1           All
```

Scripting Tips
echo

Silence is golden

Many UNIX commands that run successfully produce no output

```
[simben90@opus bin]$ alias details=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
```

Silence is golden

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

```
[simben90@opus bin]$ cat quiet
alias details=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

You can use the echo command in your scripts to provide:

- *interaction*
- *feedback*
- *tracing (for debugging)*

```
[simben90@opus bin]$ cat quiet
alias details=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]$ cat not-so-quiet
alias details=file
cp quiet quiet.bak
value=002
umask $value
echo TRACE: value=$value
cat quiet > /dev/null
echo "Quiet script successfully completed"

[simben90@opus bin]$ not-so-quiet
TRACE: value=002
Quiet script successfully completed
```

Housekeeping

Previous material and assignment

1. Lab 10 due by 11:59PM tonight
2. The Extra Credit Labs X1 and X2 (30 points each) are available.
3. The Final Project is available.

Sage advice:

Get one "practice" task script working in your project before you leave class today.

Review the final project grading rubric to see how many points you have completed so far with your practice script.

Fall 2014 Linux Classes

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: CS 1L or CIS 72.

Transfer Credit: CSU.

Section	Days	Times	Units	Instructor	Room
84743	T	01:00PM-04:05PM	3.00	R.Simms	OL
Section 84743 is an ONLINE course. Meets weekly throughout the semester online during the scheduled times by remote technology. For details, see instructor's web page at go.cabrillo.edu/online.					
86576	T	01:00PM-04:05PM	3.00	R.Simms	828

CIS 191AB UNIX/Linux Installation, Configuration and Administration

Introduces skills required to administer UNIX/Linux systems. Prerequisite: CIS 90 or equivalent.

Section	Days	Times	Units	Instructor	Room
84737	TH	01:00PM-05:05PM	4.00	M.Matera	OL
&	Arr.	Arr.		M.Matera	OL
Section 84737 is an ONLINE course. Meets weekly throughout the semester online at the scheduled times by remote technology with an additional 4 hr 5 min online lab per week. For details, see instructor's web page at go.cabrillo.edu/online.					
86577	TH	01:00PM-05:05PM	4.00	M.Matera	828
&	Arr.	Arr.		M.Matera	OL
Section 86577 is a Hybrid ONLINE course. Meets weekly throughout the semester at the scheduled times with an additional 4 hr 5 min online lab per week. For details, see instructor's web page at go.cabrillo.edu/online.					

Final Exam

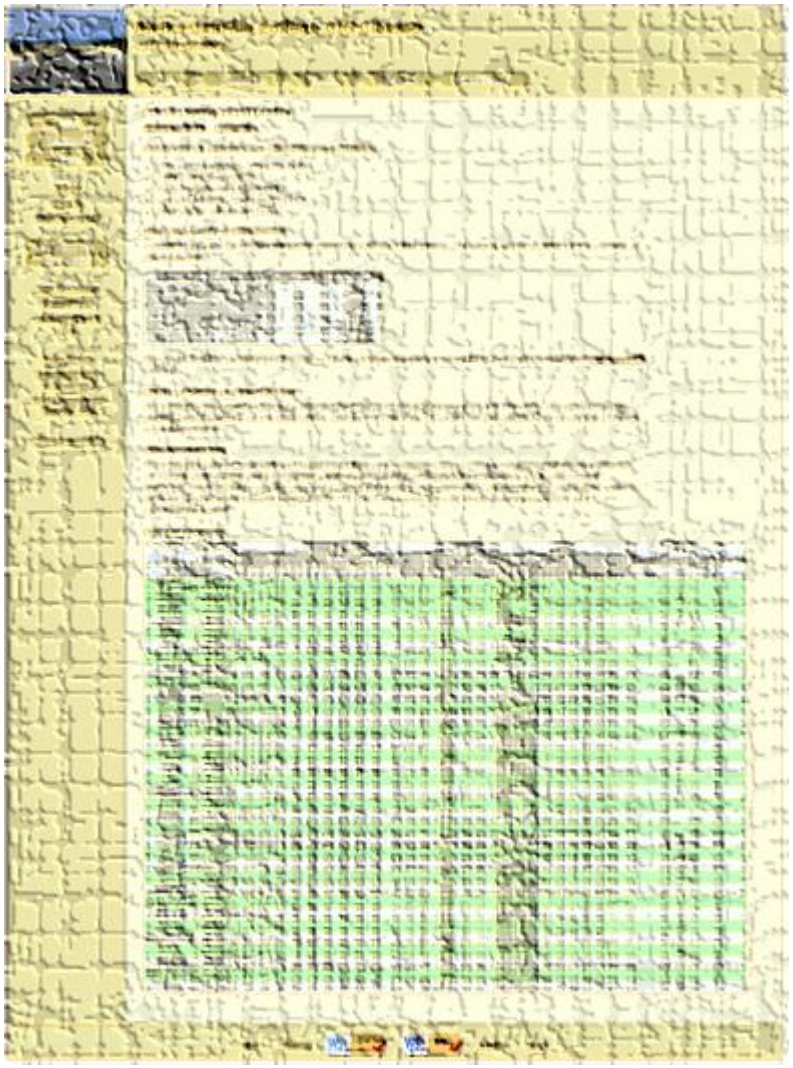
Test #3 (final exam)

- Must be face-to-face or proctored (not online using CCC Confer).
- Room 828 on campus.
- Timed test (no 11:59PM grace period)

	5/21	**Test #3 (the final exam)** **Time** • 7:00AM - 9:50AM in Room 828 **Materials** • Test (download)		5 posts Lab X1 Lab X2

- If you are a long distance student, contact the instructor for options.

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



GRADES

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

Current Point Tally

As of 4/28/2014

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**

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

alatar: 69% (288 of 417 points)
anborn: 81% (340 of 417 points)
aragorn: 77% (324 of 417 points)
arwen: 100% (420 of 417 points)
bilbo: 43% (181 of 417 points)
celebrian: 98% (410 of 417 points)
dwalin: 94% (392 of 417 points)
eomer: 92% (387 of 417 points)
faramir: 95% (399 of 417 points)
frodo: 78% (327 of 417 points)
gwaihir: 105% (440 of 417 points)
ioreth: 93% (391 of 417 points)
legolas: 89% (375 of 417 points)
marhari: 70% (296 of 417 points)
orome: 81% (340 of 417 points)
pallando: 0% (0 of 417 points)
pippen: 70% (295 of 417 points)
quickbeam: 92% (384 of 417 points)
rian: 0% (0 of 417 points)
samwise: 83% (348 of 417 points)
strider: 87% (364 of 417 points)
theoden: 44% (187 of 417 points)
treebeard: 105% (440 of 417 points)
tulkas: 87% (365 of 417 points)
ulmo: 84% (353 of 417 points)

Jesse's checkgrades python script

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

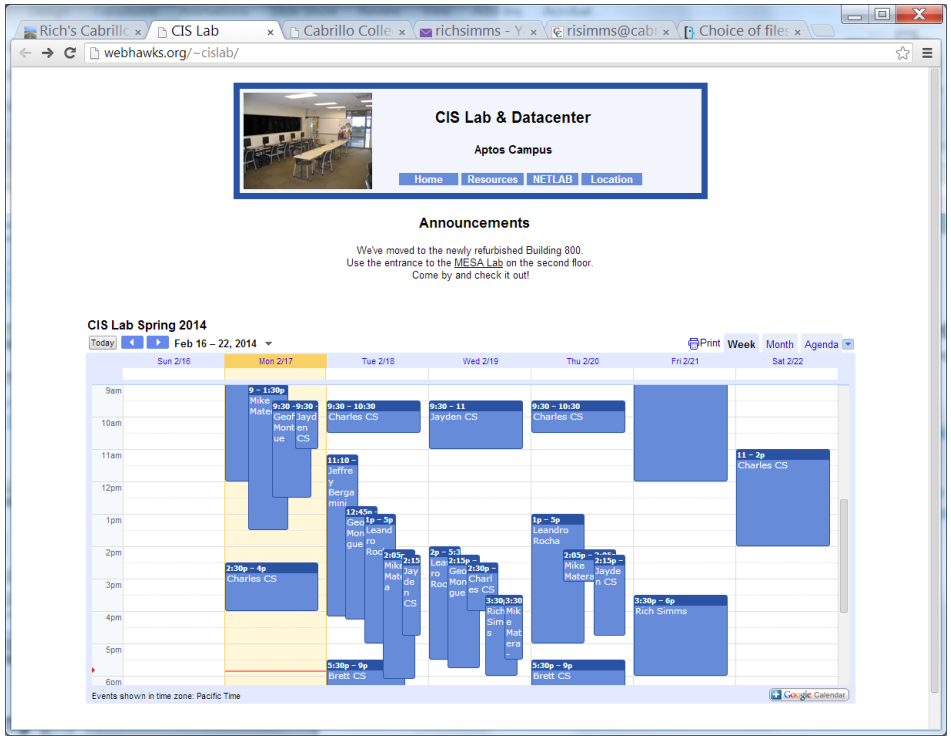
```
/home/cis90/simben $ checkgrades smeagol

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

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


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

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


You've earned 15 points of extra credit.

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

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

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

CIS Lab Schedule

http://webhawks.org/~cislab/



Not submitting tests or lab work?

Would like some additional help?

Come to the CIS Lab to work with classmates, lab assistants and instructors on Lab assignments.

Rich is in the lab Wednesdays and Fridays from 3:30 - 6:00 PM

Final Project Grading Rubric

Grading rubric (60 points maximum)

Possible Points	Requirements
30	Implementing all five tasks (6 points each): • Requirements for each task: – Minimum of 10 "original" script command lines – Has comments to explain what it does – 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 a GID or group (5 points) • Use of a UID or user (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 are 25 points.
5	Present your script in front of 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
Extra credit	
30	Up to three additional tasks (10 points each)

Final Project forum

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 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

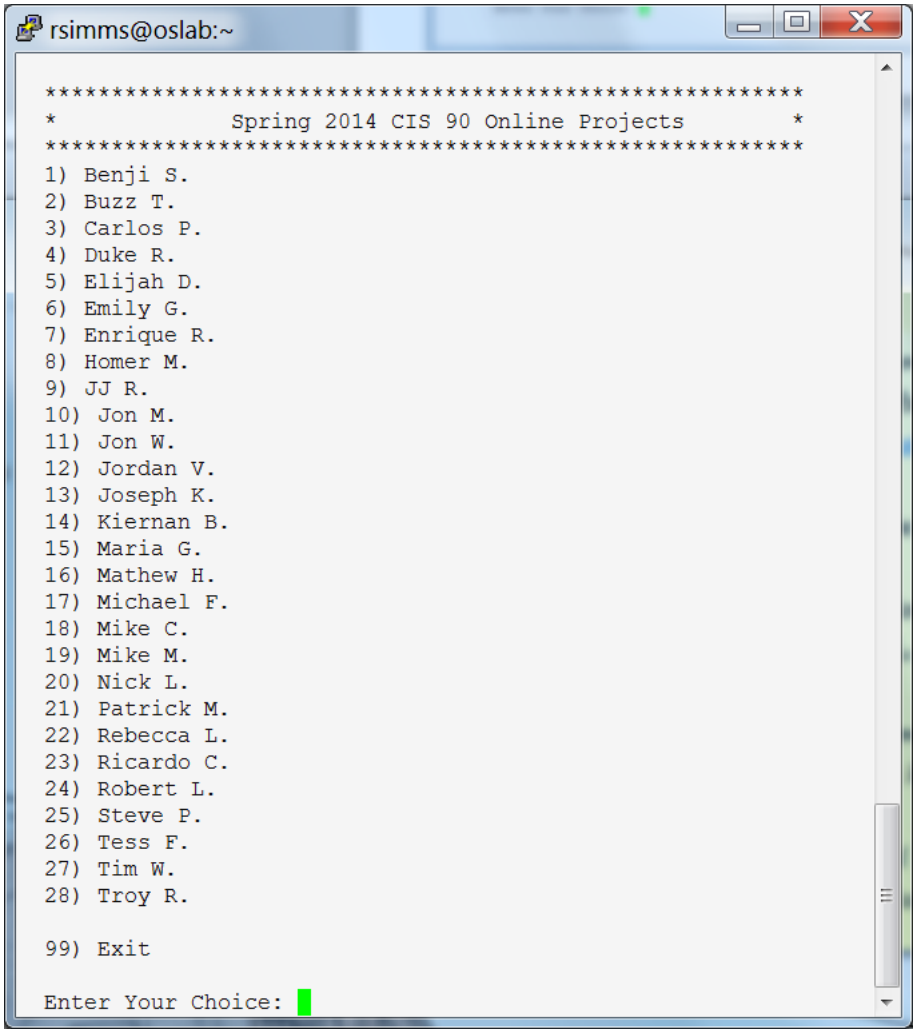
Best ...

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

The thread will also benefit future CIS 90 students

Final Project permissions

Final Project



Before leaving class today you want to make sure you can run your script from ***allscripts***

Permissions

A past forum post …

Ha Ha Class

by [redacted] on Tue May 12, 2009 12:22 pm

I'm sure this is some kind of payback for last week "Hacking" attempt

I will find out who did this

~~~~~~

ps. Im going to pass

Uh, oh … someone got hacked!
~~~~~~

Group Write Permissions

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

rsimms@oslab:~

```
[rsimms@oslab ~]$ ls -l /home/cis90/*/bin/myscript
-rwxrwxr-x. 1 beakie90 cis90  708 Apr 23 10:53 /home/cis90/beakie/bin/myscript
-rwxrwxr-x. 1 calmic90 cis90  801 Apr 23 11:20 /home/cis90/calmic/bin/myscript
-rwxrwxr-x. 1 fahmic90 cis90  728 Apr 23 10:43 /home/cis90/fahmic/bin/myscript
-rwxrwxr-x. 1 fitcon90 cis90  698 Apr 24 16:01 /home/cis90/fitcon/bin/myscript
-rwxrwxr-x. 1 keljos90 cis90  813 Apr 23 10:57 /home/cis90/keljos/bin/myscript
-rwxrwxr-x. 1 lefnic90 cis90  833 Apr 23 10:54 /home/cis90/lefnic/bin/myscript
-rwxrwxr-x. 1 lemrob90 cis90  720 Apr 23 11:08 /home/cis90/lemrob/bin/myscript
-rwxrwxr-x. 1 matjon90 cis90  708 Apr 23 10:46 /home/cis90/matjon/bin/myscript
-rwxrwxr-x. 1 milhom90 cis90  799 Apr 23 10:52 /home/cis90/milhom/bin/myscript
-rwxrwxr-x. 1 patcar90 cis90  798 Apr 23 13:04 /home/cis90/patcar/bin/myscript
-rwxrwxr-x. 1 rudtro90 cis90  562 Apr 29 18:04 /home/cis90/rudtro/bin/myscript
-rwxrwxr-x. 1 tilbuz90 cis90 6155 Apr 25 16:01 /home/cis90/tilbuz/bin/myscript
-rwxr-xr-x. 1 weljon90 cis90  546 Apr 29 10:16 /home/cis90/weljon/bin/myscript
-rwxrwxr-x. 1 weltim90 cis90  729 Apr 23 10:56 /home/cis90/weltim/bin/myscript
[rsimms@oslab ~]$
```

*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?

Group Read and Execute Permissions

rsimms@oslab:~

```
[rsimms@oslab ~]$ /home/cis90/bin/checkmyscripts
ls: cannot access /home/cis90/simben/bin/myscript: No such file or directory
-rwxrwxr-x. 1 milhom90 cis90 799 Apr 23 10:52 /home/cis90/milhom/bin/myscript
ls: cannot access /home/cis90/rodduk/bin/myscript: No such file or directory
-rwxrwxr-x. 1 beakie90 cis90 708 Apr 23 10:53 /home/cis90/beakie/bin/myscript
-rwxrwxr-x. 1 calmic90 cis90 801 Apr 23 11:20 /home/cis90/calmic/bin/myscript
ls: cannot access /home/cis90/casric/bin/myscript: No such file or directory
-rwxrwxr-x. 1 fahmic90 cis90 728 Apr 23 10:43 /home/cis90/fahmic/bin/myscript
-rwxrwxr-x. 1 fitcon90 cis90 698 Apr 24 16:01 /home/cis90/fitcon/bin/myscript
ls: cannot access /home/cis90/gutemi/bin/myscript: No such file or directory
ls: cannot access /home/cis90/hormat/bin/myscript: No such file or directory
-rwxrwxr-x. 1 keljos90 cis90 813 Apr 23 10:57 /home/cis90/keljos/bin/myscript
-rwxrwxr-x. 1 lefnic90 cis90 833 Apr 23 10:54 /home/cis90/lefnic/bin/myscript
ls: cannot access /home/cis90/lehreb/bin/myscript: No such file or directory
-rwxrwxr-x. 1 lemrob90 cis90 720 Apr 23 11:08 /home/cis90/lemrob/bin/myscript
-rwxrwxr-x. 1 patcar90 cis90 798 Apr 23 13:04 /home/cis90/patcar/bin/myscript
ls: cannot access /home/cis90/perste/bin/myscript: No such file or directory
ls: cannot access /home/cis90/ramjua/bin/myscript: No such file or directory
-rwxrwxr-x. 1 rudtro90 cis90 562 Apr 29 18:04 /home/cis90/rudtro/bin/myscript
-rwxrwxr-x. 1 tilbuz90 cis90 6155 Apr 25 16:01 /home/cis90/tilbuz/bin/myscript
ls: cannot access /home/cis90/vasjor/bin/myscript: No such file or directory
-rwxrwxr-x. 1 weltim90 cis90 729 Apr 23 10:56 /home/cis90/weltim/bin/myscript
ls: cannot access /home/cis90/mosmic/bin/myscript: Permission denied
-rwxr-xr-x. 1 weljon90 cis90 546 Apr 29 10:16 /home/cis90/weljon/bin/myscript
-rwxrwxr-x. 1 matjon90 cis90 708 Apr 23 10:46 /home/cis90/matjon/bin/myscript
ls: cannot access /home/cis90/genmar/bin/myscript: No such file or directory
[rsimms@oslab ~]$
```

Which myscript files cannot by run by classmates?

Class Activity

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

To meet this requirement use:

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

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!

Permissions

```
[rodduk90@opus 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.

Class Activity

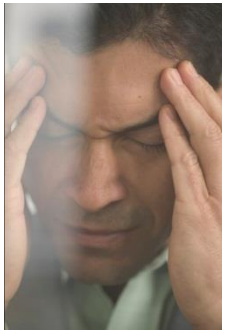
- Change your umask to 026
- Can group or other users modify future new files now?
- Try it, **touch** a new file and check the permissions with **ls -l**
- How would you make this a permanent umask setting?

Final Project

What takes longer?

Writing the script?

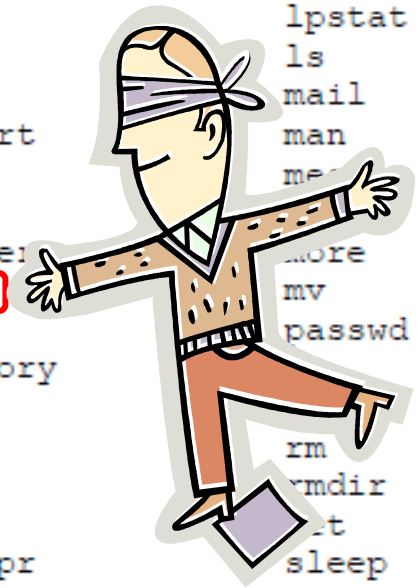
Or deciding what to script?



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

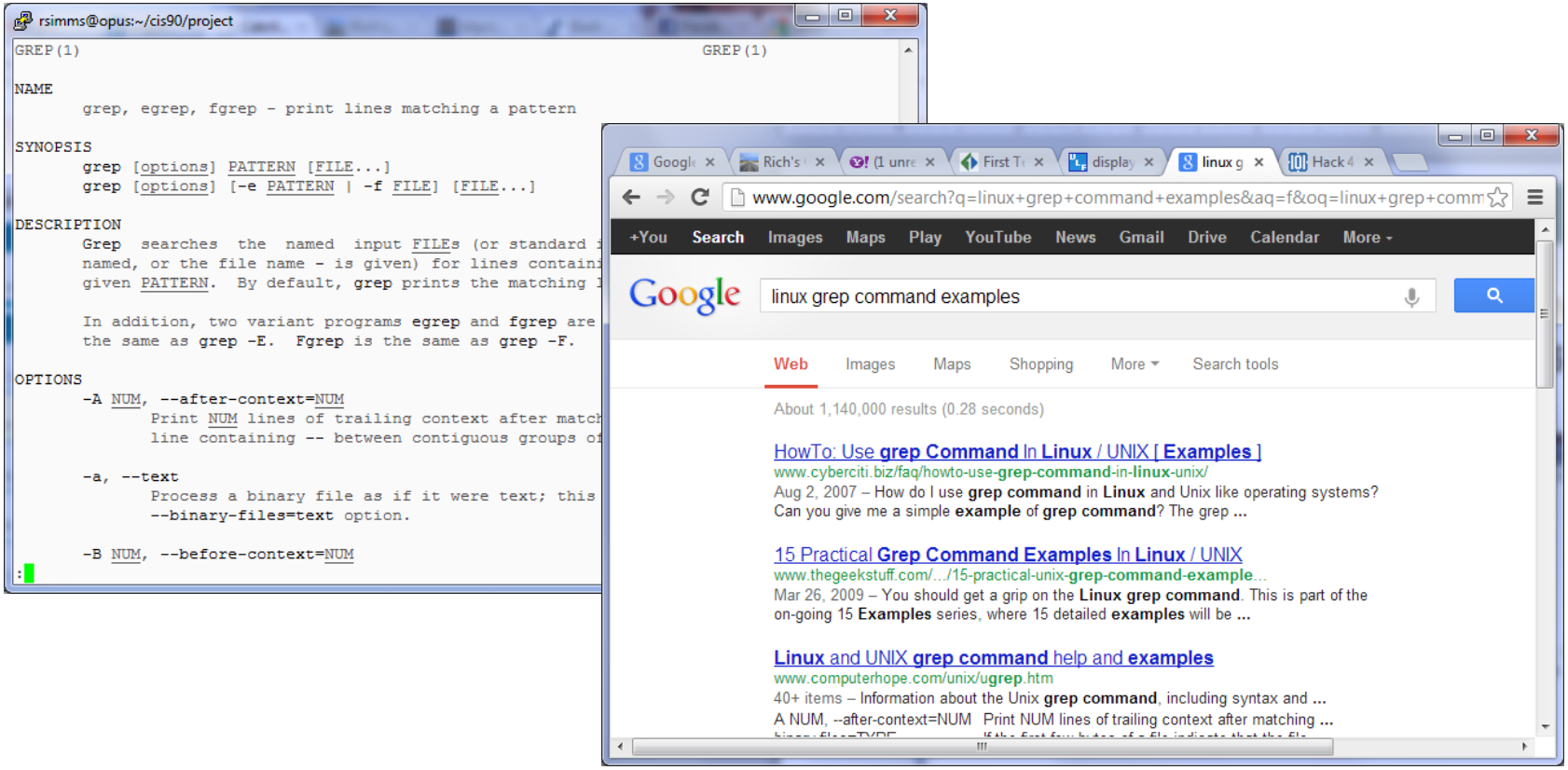
Commands

.	echo	lpstat	sort
at	env	ls	spell
banner	exit	mail	su
bash	export	man	tail
bc	file	me	tee
cal	find		touch
cancel	finger	more	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		write
date	lp/lpr	sleep	xxd



For this example we will pick the grep command

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

Before

rodduk90@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
                ;;
          2)    # Commands for Task
                ;;
"myscript" 37L, 546C
```

After

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
                ;;
-- INSERT --                                    10,5-12       Top
```

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*.

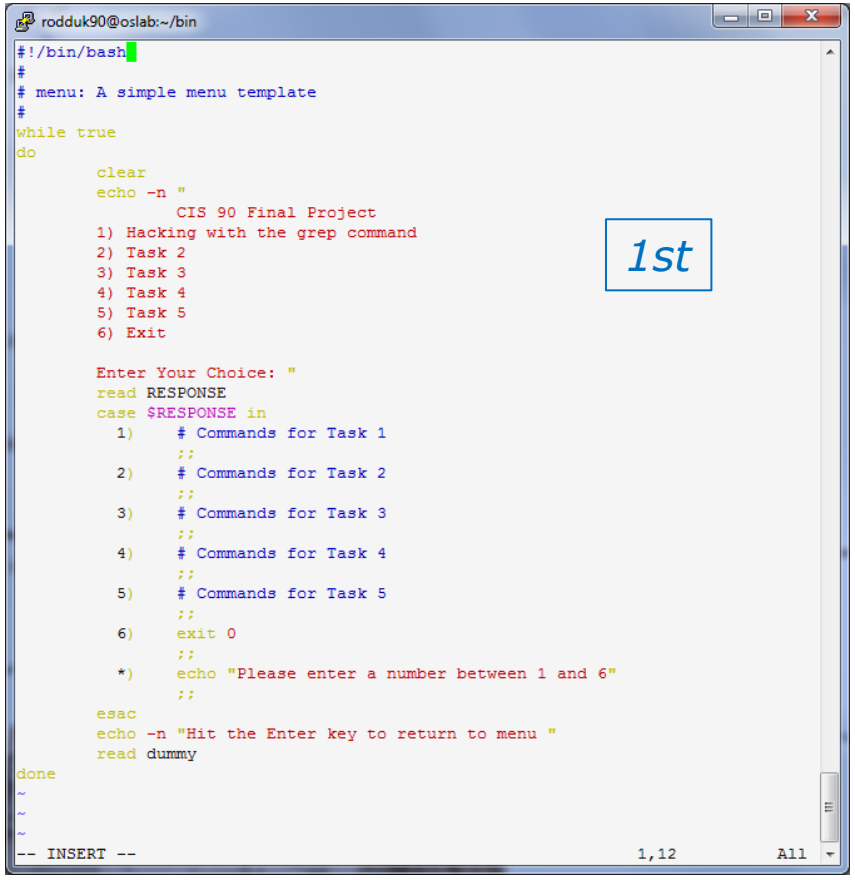
The term 'hacker' also tends to connote membership in the global community defined by the net (see *the network*. For discussion of some of the basics of this culture, see the How To Become A Hacker FAQ. It also implies that the person described is seen to subscribe to some version of the hacker ethic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

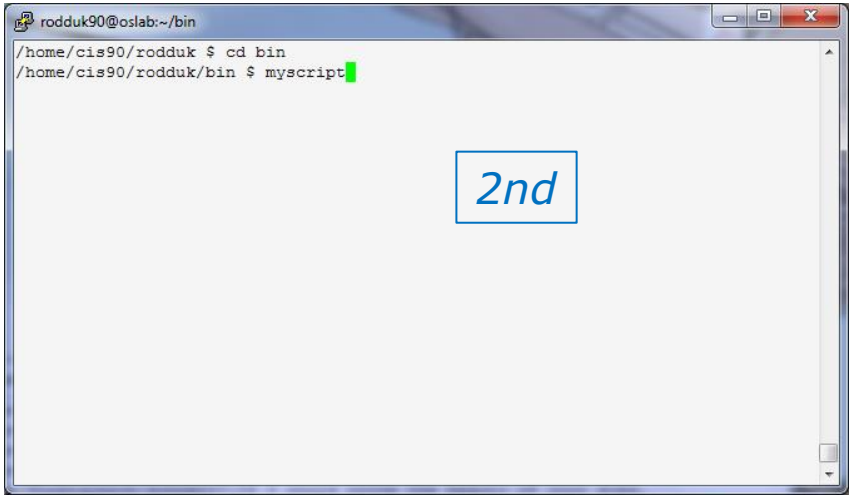
It is better to be described as a hacker by others than to describe oneself that way. Hackers consider themselves something of an elite (a meritocracy based on ability), though one to which new members are gladly welcome. There is thus a certain ego satisfaction to be had in identifying yourself as a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

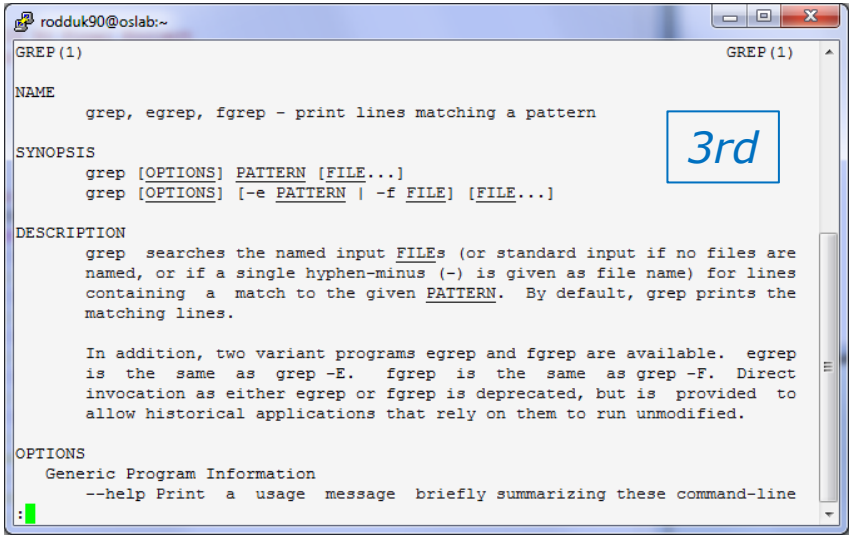
This term seems to have been first adopted as a badge in the 1960s by the hacker culture surrounding TMRC and the MIT AI Lab. We have a report that it was used in a sense close to this entry's by teenage radio hams and electronics tinkerers in the mid-1950s.

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

Layout your work area on the screen







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*

Test your menu change

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
~
~
~
"myscript" 37L, 569C written                                1,11          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:
```

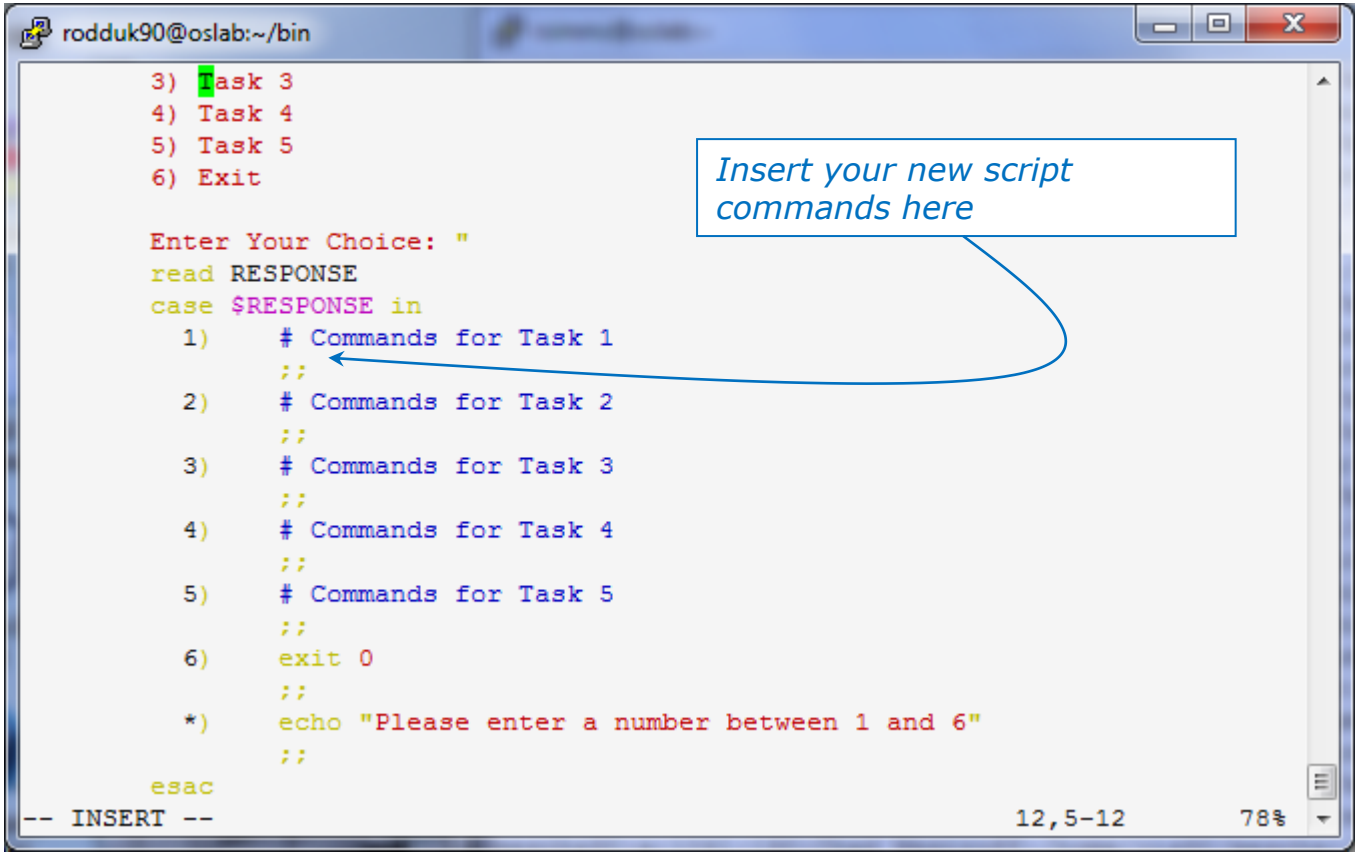
Changes work!

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
:
```

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

Find the location to insert your new task commands



Now its time to add some commands to the task.

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

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
```

*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.*

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 $
```

Fix it and test again

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 /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.*

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
/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 $
```

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/*/*
    ;;
```

rodduk90@oslab:~/bin

```
#!/bin/bash
#
# menu: A simple menu template
#
while true
do
        clear
        echo -n "
                CIS 90 Fin
        1) Hacking with th
        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
                echo "Are you ready to search for beauty in
                read dummy
                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
                ;;
        esac
        echo -n "Hit the Enter key to return to menu "
        read dummy
done
"myscript" 40L, 711C written
```

rodduk90@oslab:~/bin

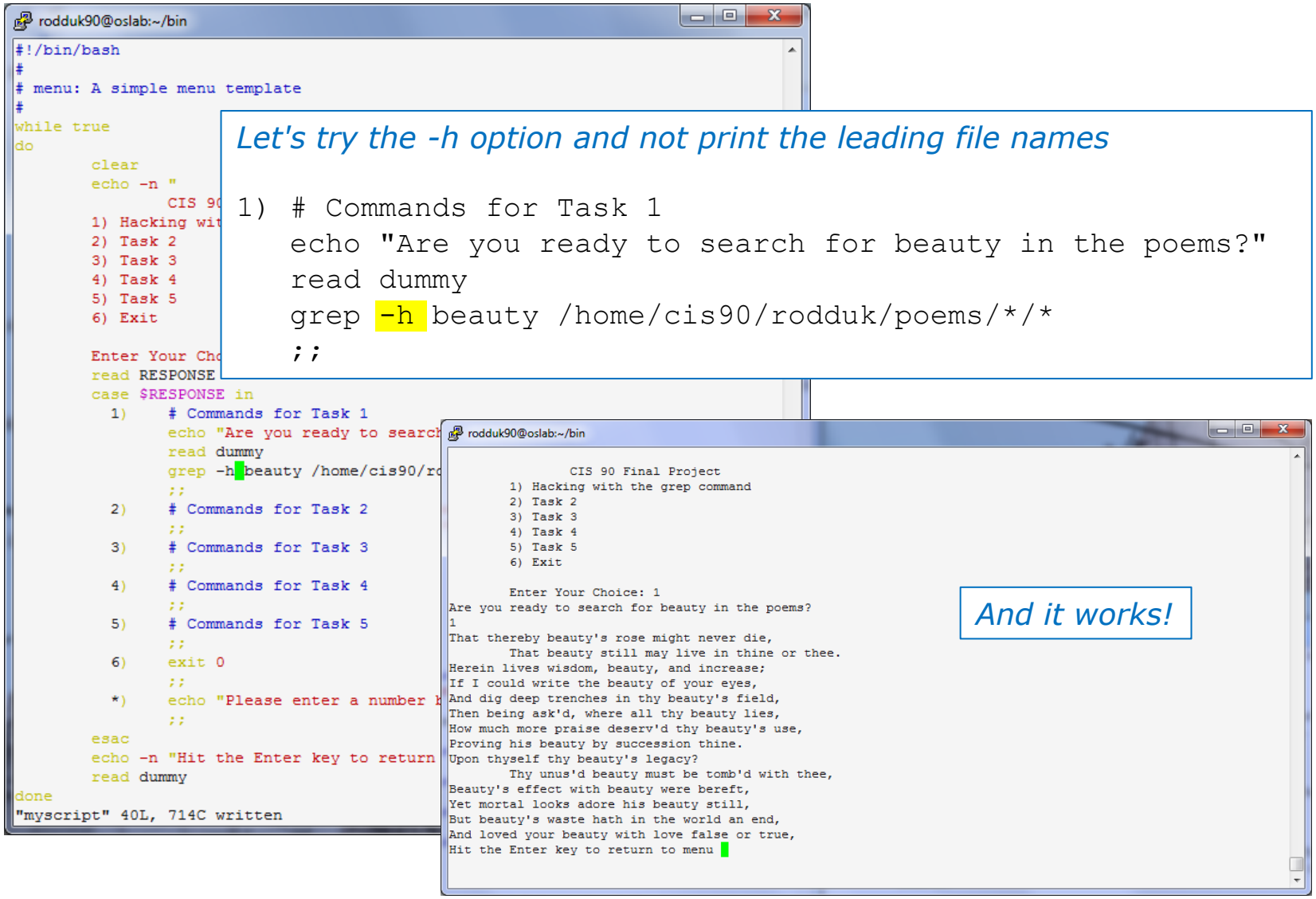
```
                CIS 90 Final Project
        1) Hacking with the grep command
        2) Task 2
        3) Task 3
        4) Task 4
        5) Task 5
        6) Exit

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

/home/cis90/rodduk/poems/Shakespeare/sonnet1:That thereby beauty's rose might never die,
/home/cis90/rodduk/poems/Shakespeare/sonnet10:  That beauty still may live in thine or thee.
/home/cis90/rodduk/poems/Shakespeare/sonnet11:Herein lives wisdom, beauty, and increase;
/home/cis90/rodduk/poems/Shakespeare/sonnet17:If I could write the beauty of your 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 beauty lies,
/home/cis90/rodduk/poems/Shakespeare/sonnet2:How much more praise deserv'd thy beauty's use,
/home/cis90/rodduk/poems/Shakespeare/sonnet2:Proving his beauty by succession thine.
/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 bereft,
/home/cis90/rodduk/poems/Shakespeare/sonnet7:Yet mortal looks adore his beauty still,
/home/cis90/rodduk/poems/Shakespeare/sonnet9:But beauty's waste hath in the world 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
```

And it works!

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/*/*
    ;;
```

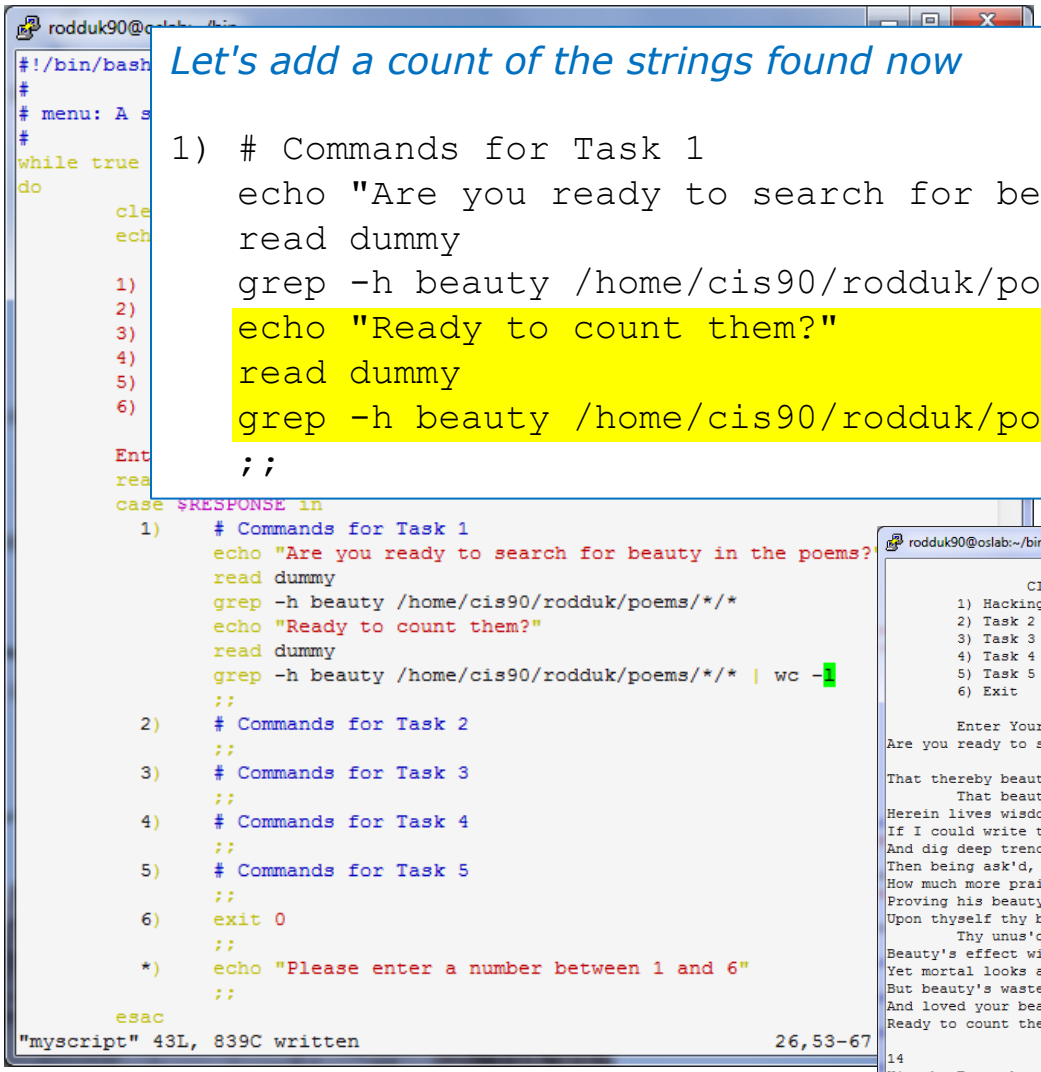


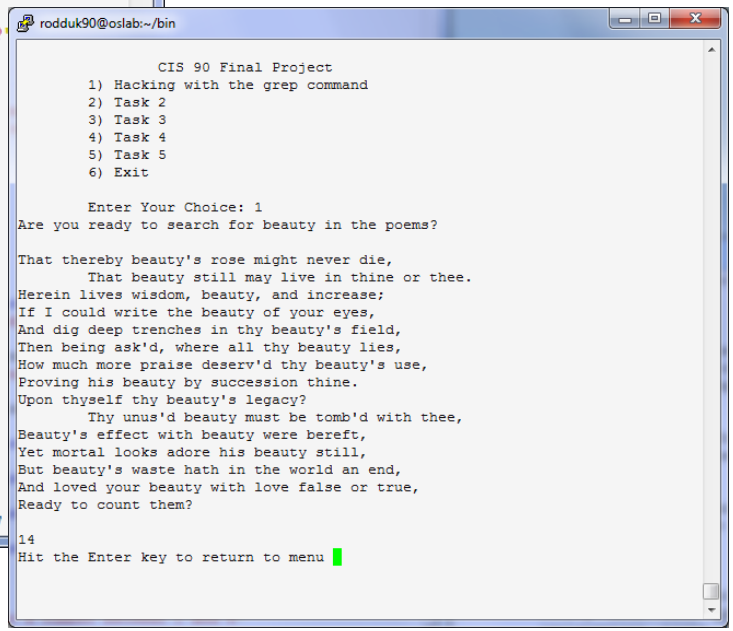
And it works!

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
    ;;
```





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:

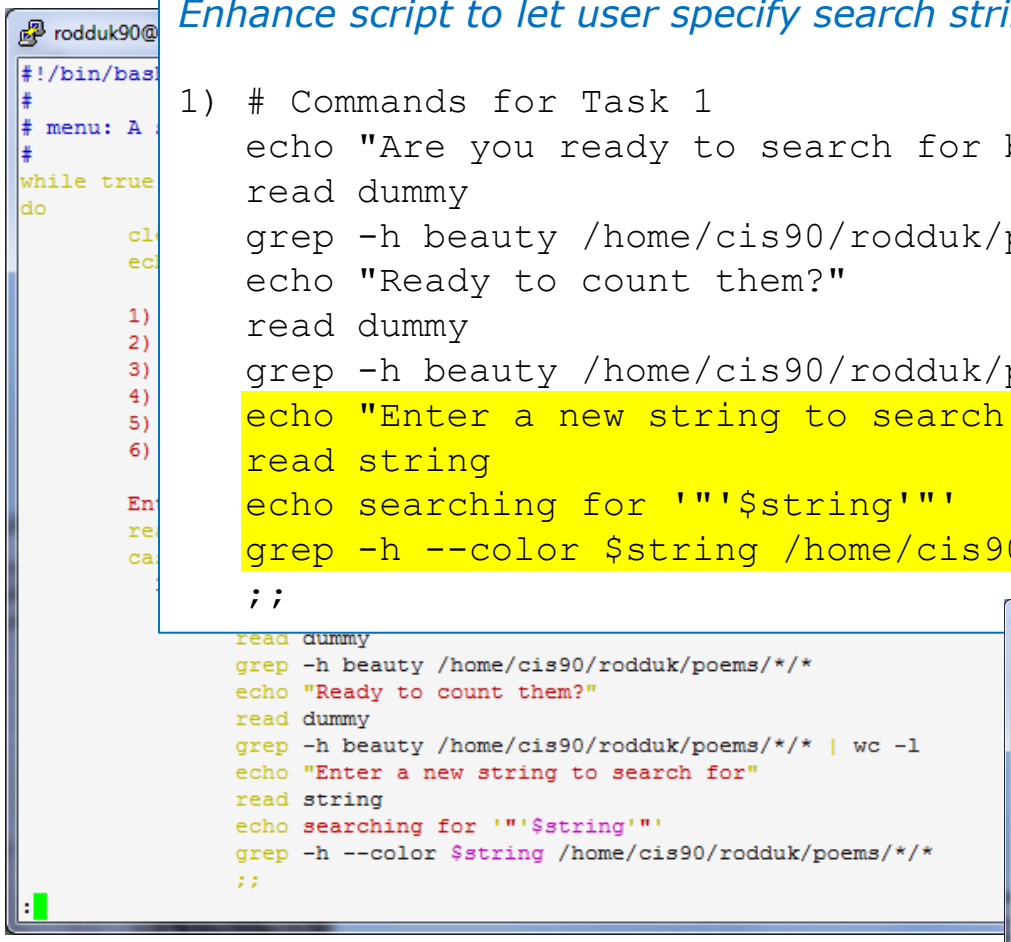
- Redirecting stdin (5 points)
- Redirecting stdout (5 points)
- Redirecting stderr (5 points)
- Use of permissions (5 points)
- ✔ Use of filename expansion characters (5 points)
- ✔ Use of absolute path (5 points)
- Use of relative path (5 points)
- Use of a PID (5 points)
- Use of inodes (5 points)
- 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)

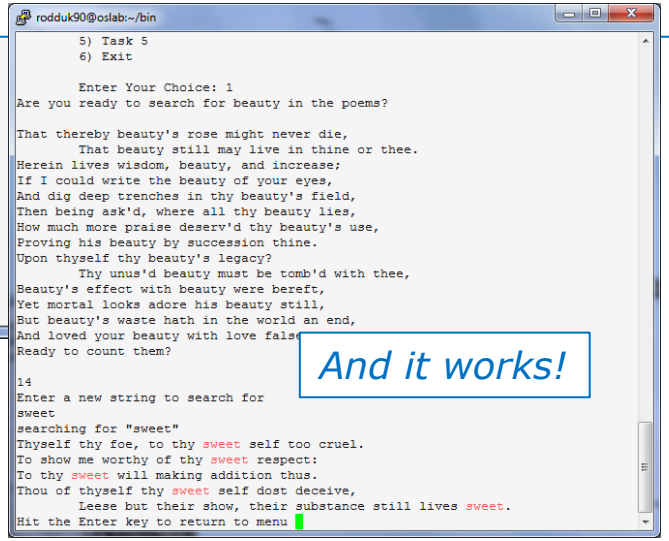
The maximum for this section is 25 points.

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/*/*
    ;;
```





And it works!

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/po
    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.

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

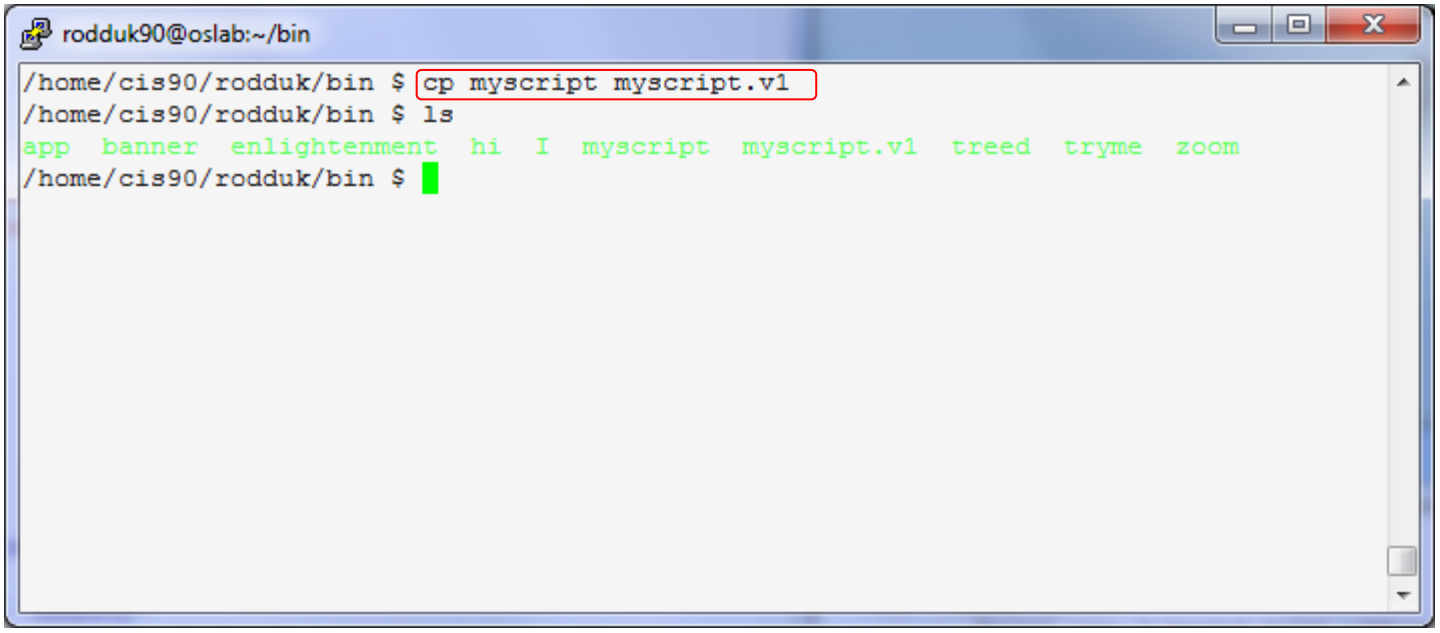
You don't have to do all of these but do at least five:
- Redirecting stdin (5 points)
- Redirecting stdout (5 points)
- Redirecting stderr (5 points)
- Use of permissions (5 points)
- ✔ Use of filename expansion characters (5 points)
- ✔ Use of absolute path (5 points)
- Use of relative path (5 points)
- 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!

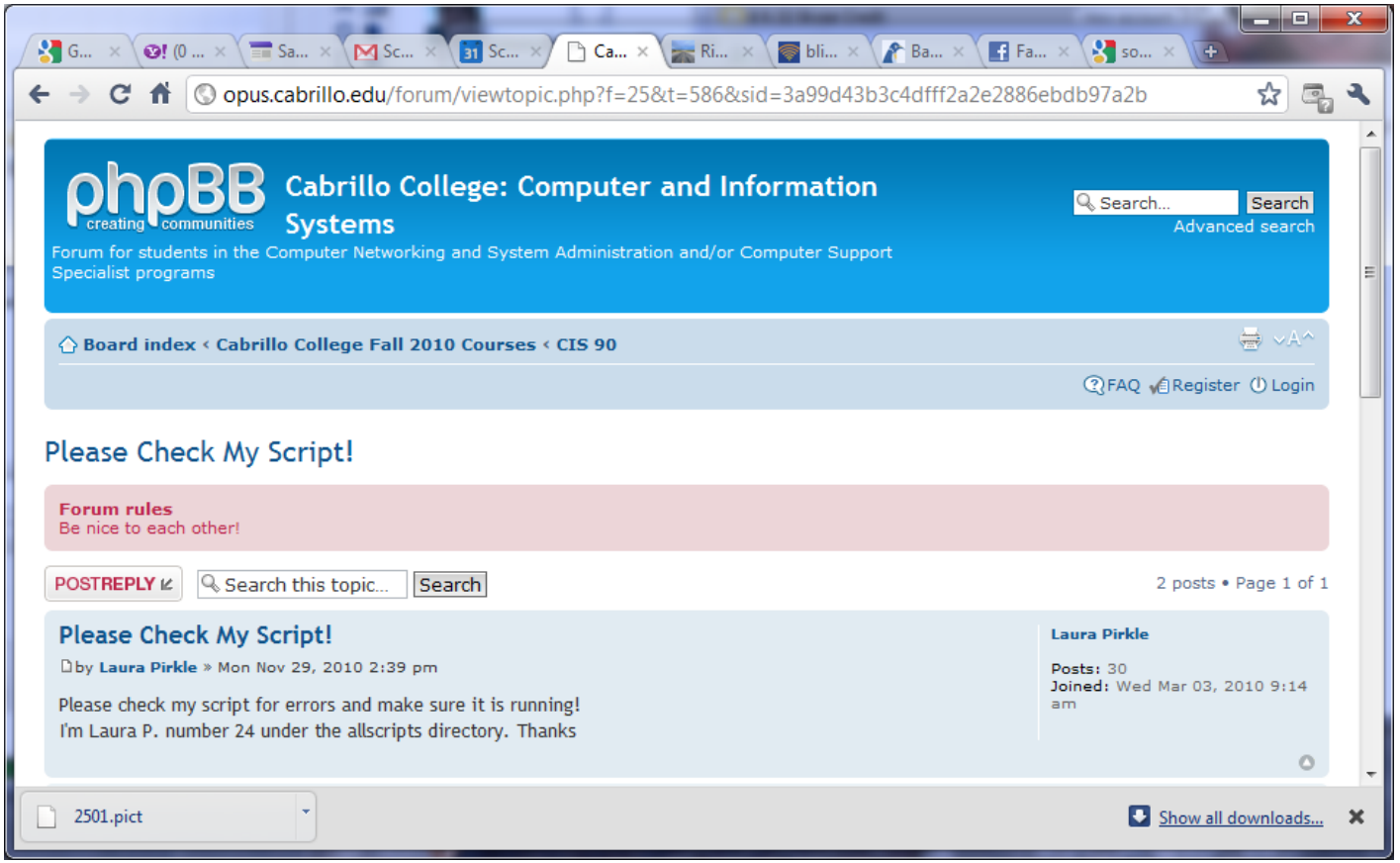
cp myscript myscript.v1 *after first day of work*



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!

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!

Scripting Tips

$(*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

Scripting Tips

date

Utilizing the date command

```
/home/cis90/simben $ date
Wed Nov 26 15:35:53 PST 2008


/home/cis90/simben $ date +'%r'
04:14:26 PM
/home/cis90/simben $ time=$(date +'%r')
/home/cis90/simben $ echo "At the tone the time will be $time"
At the tone the time will be 04:15:02 PM


/home/cis90/simben $ date +'%A'
Tuesday
/home/cis90/simben $ day=$(date +'%A')
/home/cis90/simben $ echo "Today is $day"
Today is Tuesday
```

See the man page on date for lots of other % codes

Class Activity

Your turn, make a script by adding the following two lines to a file named *mydate* using the vi editor:

```
echo "Hola $LOGNAME"
echo Today is $(date +'%m/%d/%Y')
```

Give the script execute permissions and run it:

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

Copy and paste the output of your script into the chat window

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?

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
```

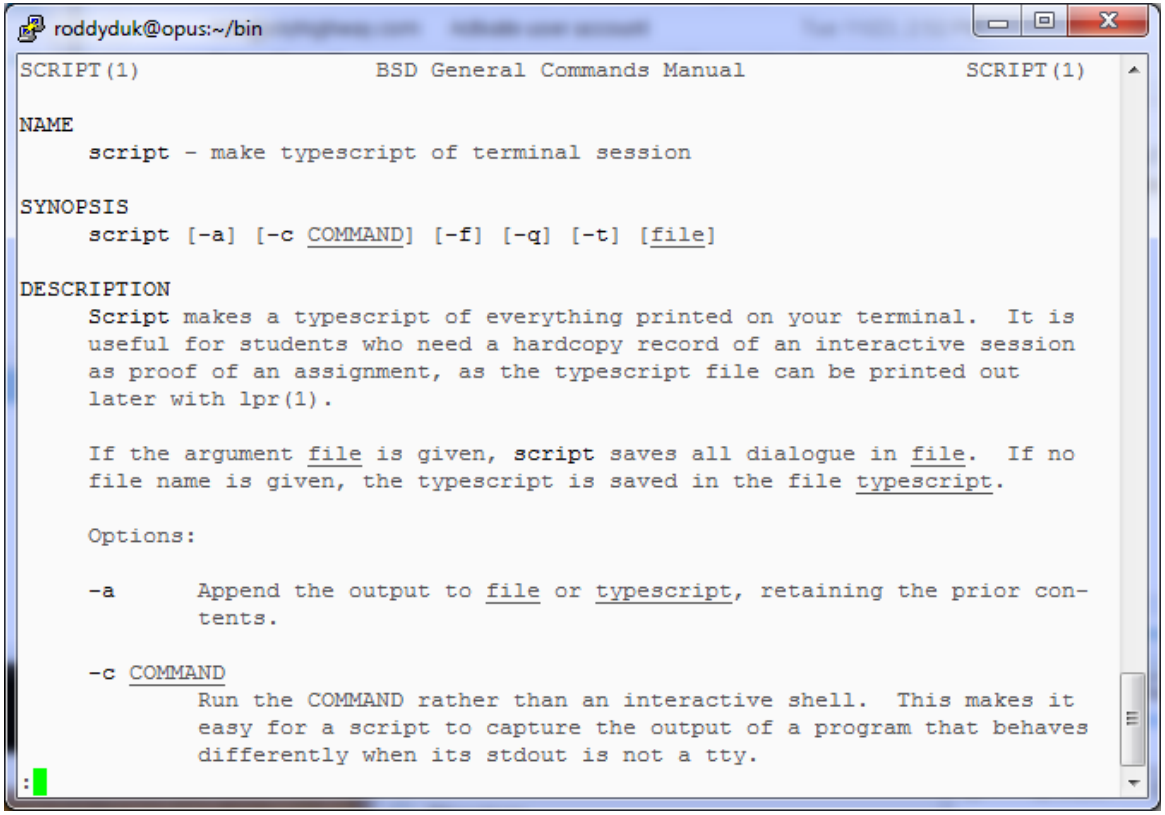
Why the heck doesn't my script do what it's supposed to do?

```
[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]$
```

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]$
```



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.*

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

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.

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

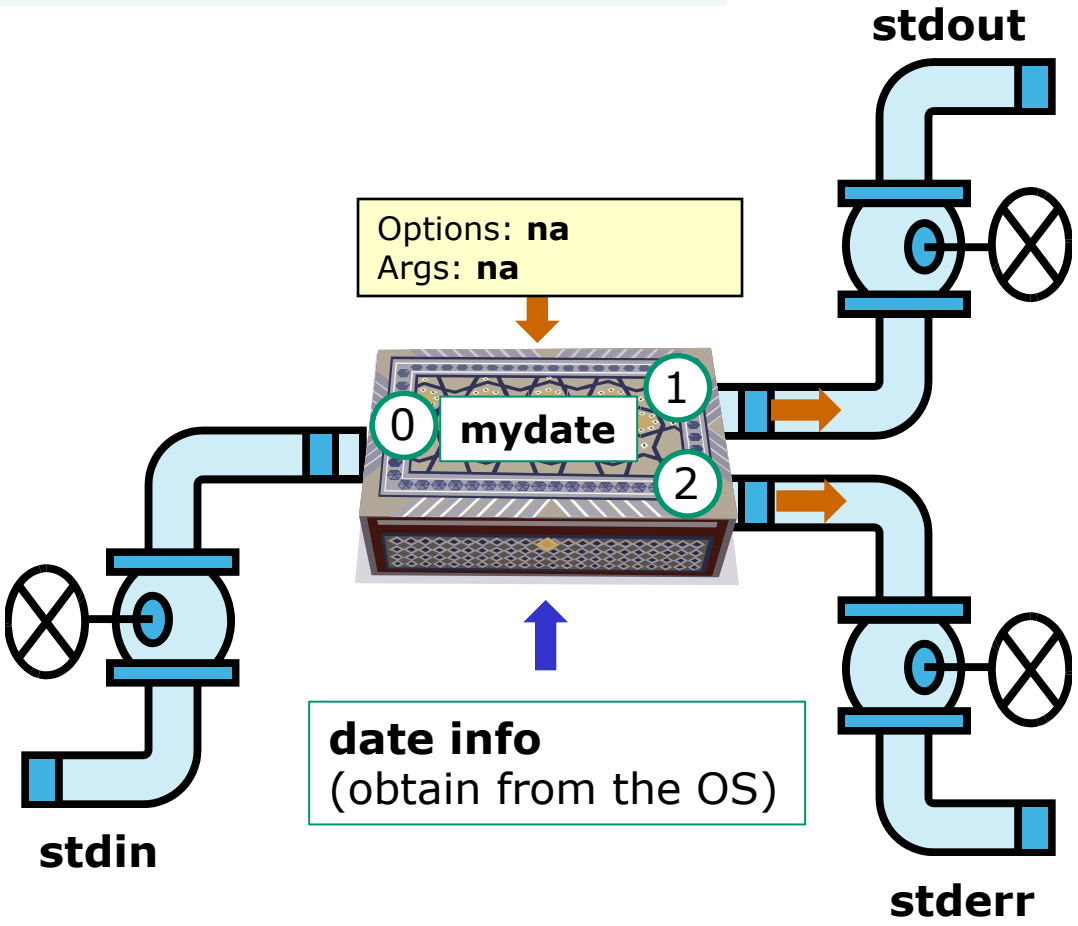
Don't initialize them yet

```
/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

Running a Script

$ mydate



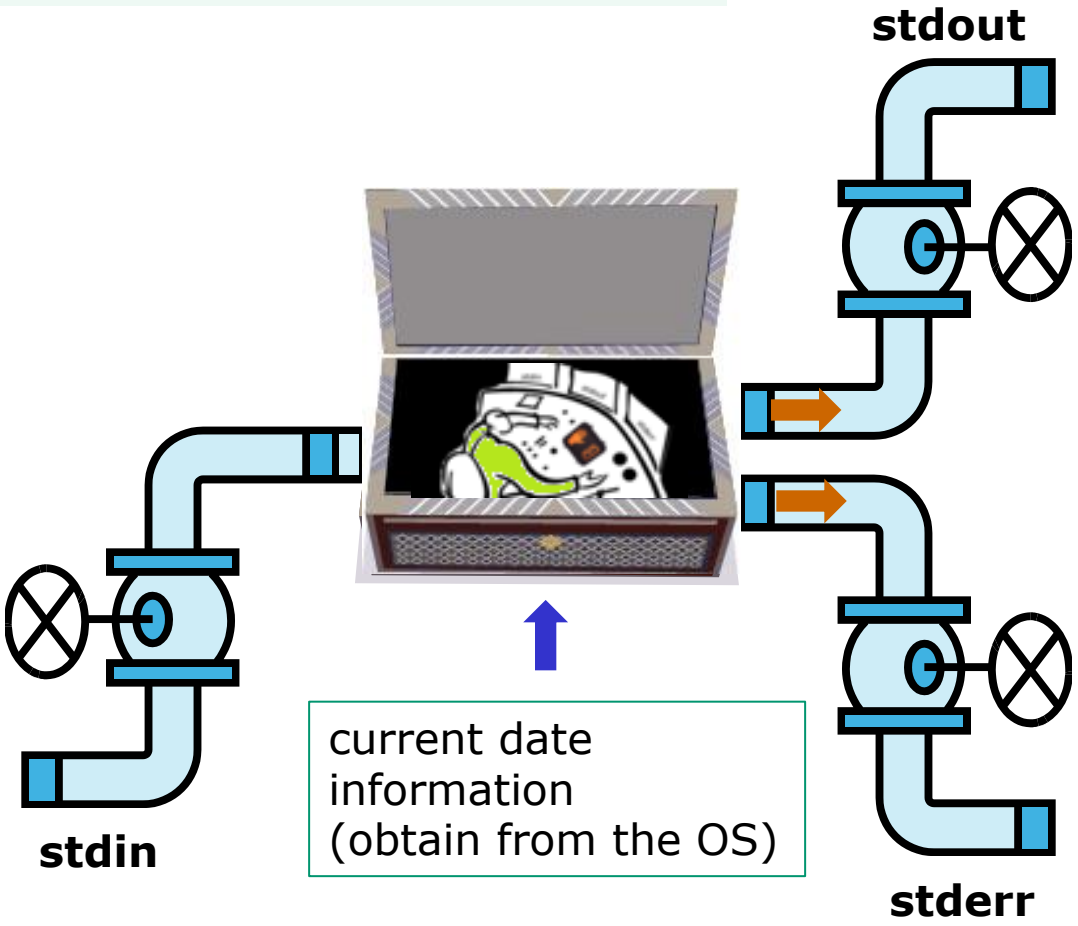

```
Hola simben90
05/09/2013
```

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

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

Running a Script

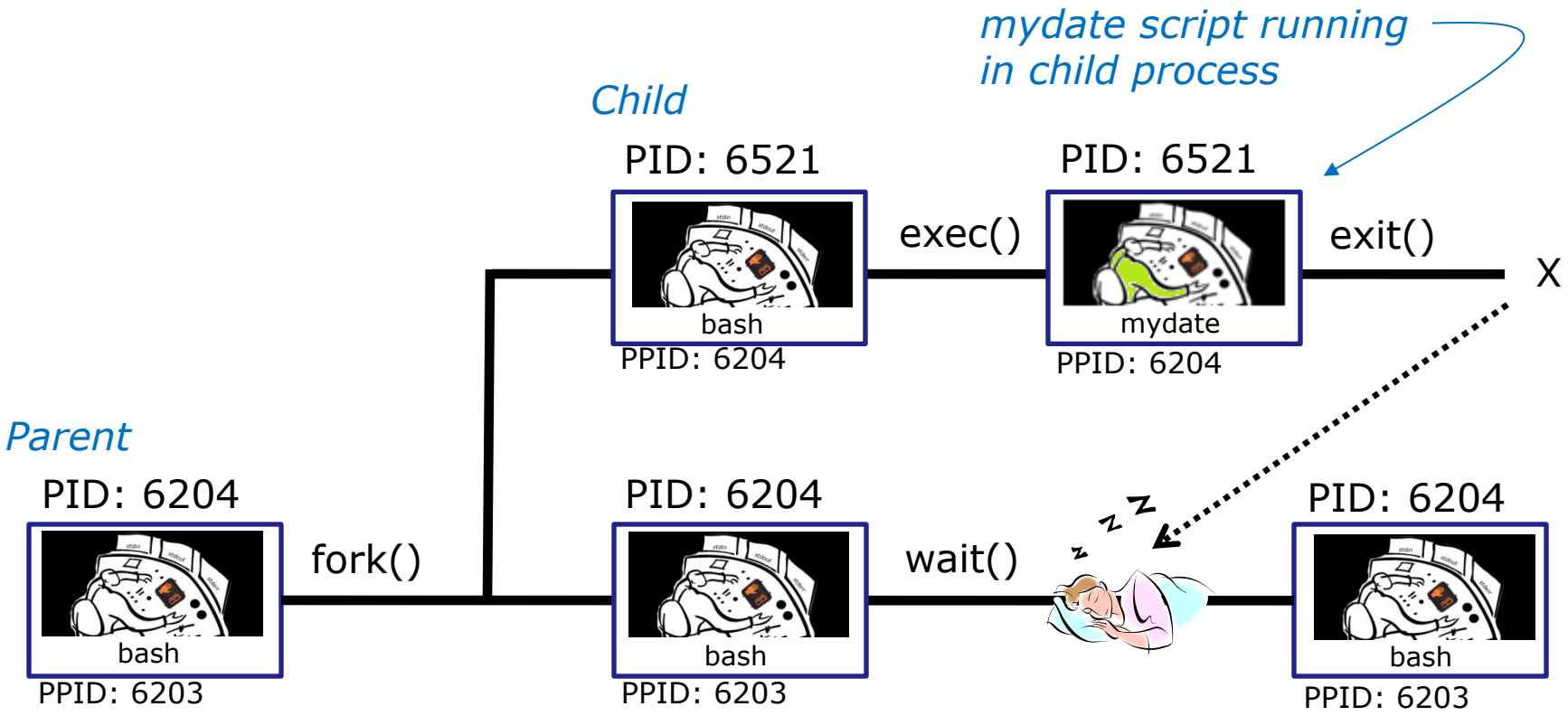
```
$ mydate
```




```
Hola simben90
05/16/2012
```

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

Running a Script



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

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?

Running a Script

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

/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
```

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

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 no-comprende
do
  runningScript
done
```

Printers

Sneak Peak for CIS 90 Students

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

Printer Configuration

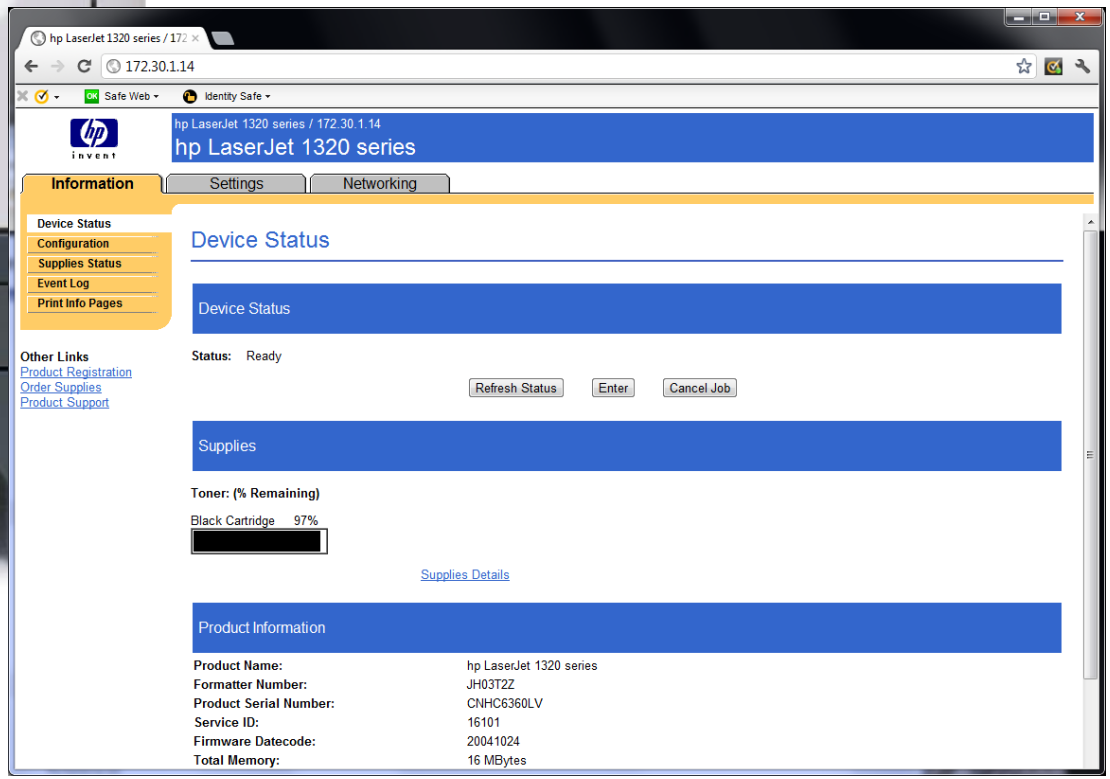
CUPS

Example printer configuration

Printer: HP LaserJet 1320n
Connection: LAN

CUPS

The LaserJets have a web-based management utility



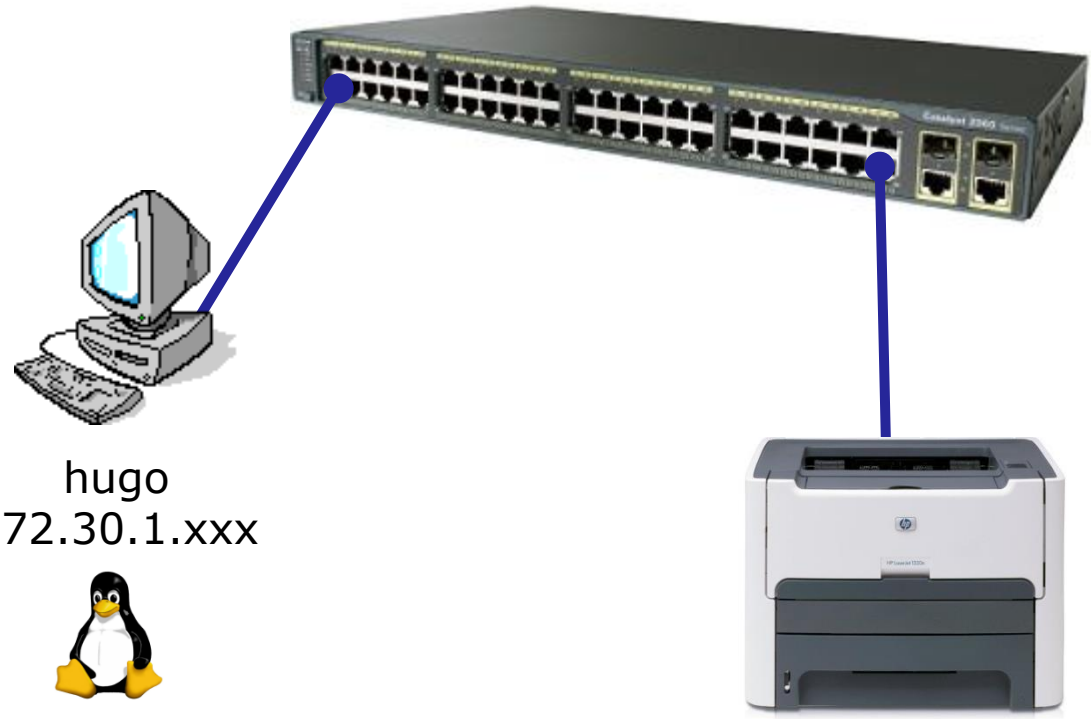
IP Address for this 1320n is 172.30.1.14

CUPS

CUPS

FYI
CIS 192

This example will show how to add the HP 1320n as a networked printer.

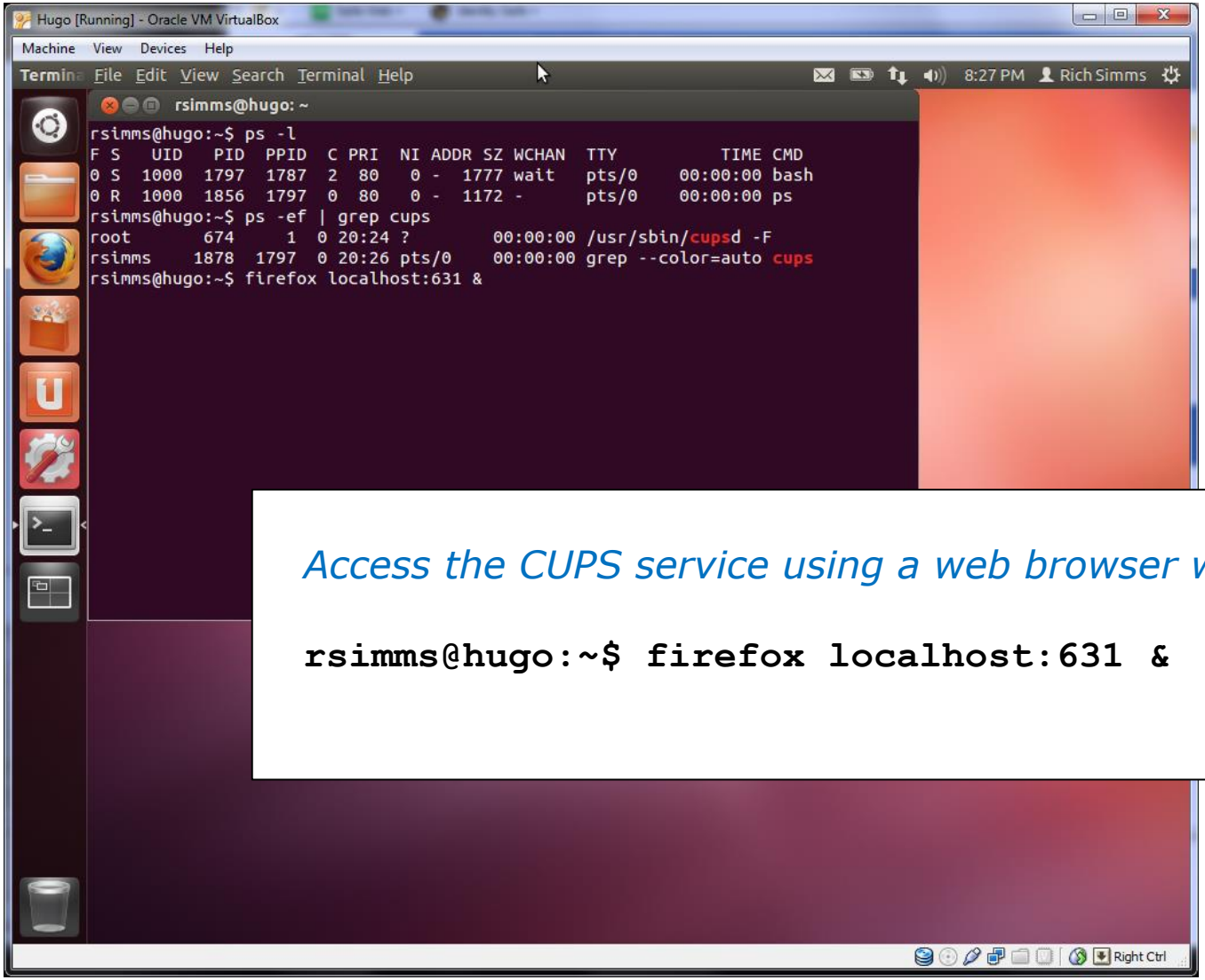


hugo
172.30.1.xxx

hp1320n
172.30.1.14

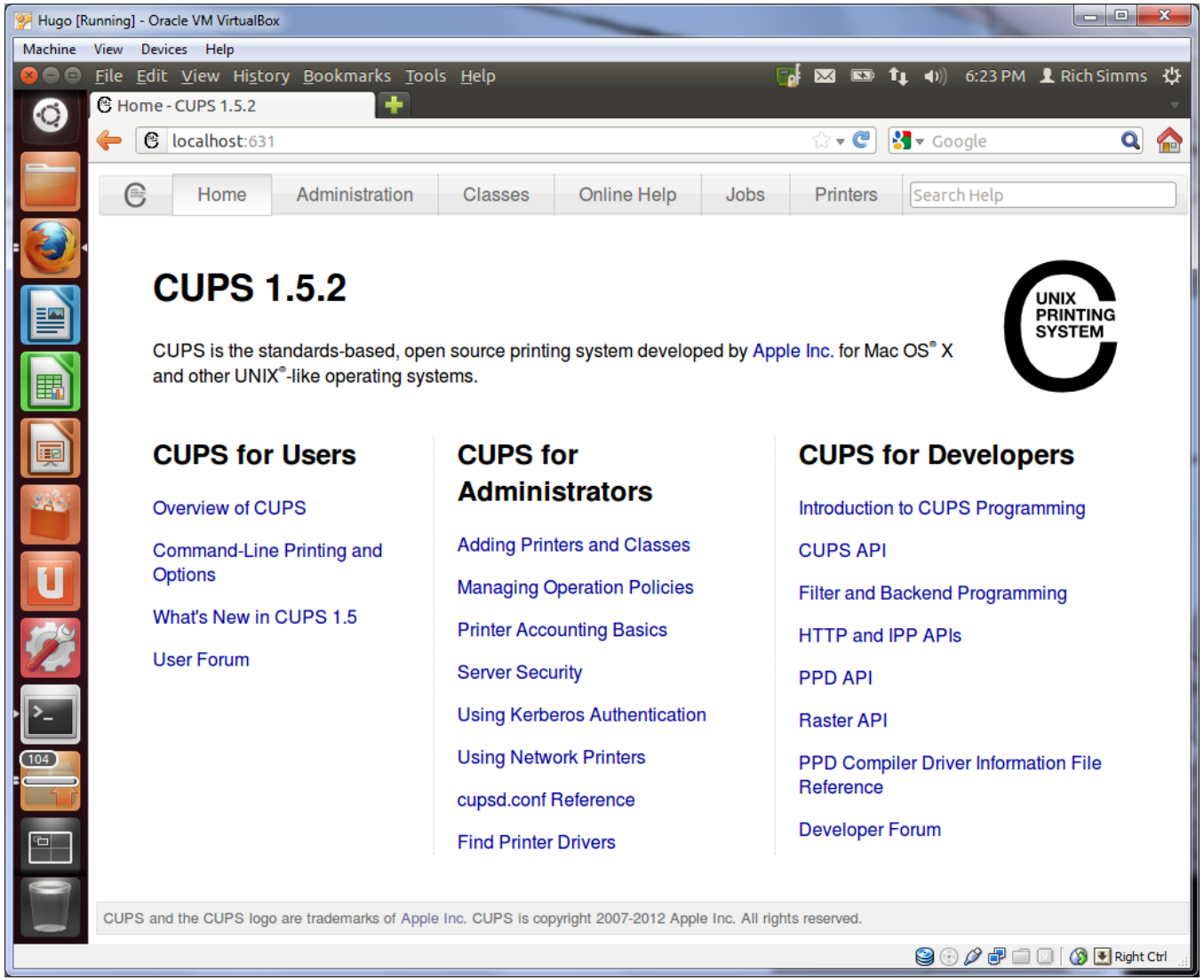
CUPS

FYI
CIS 192



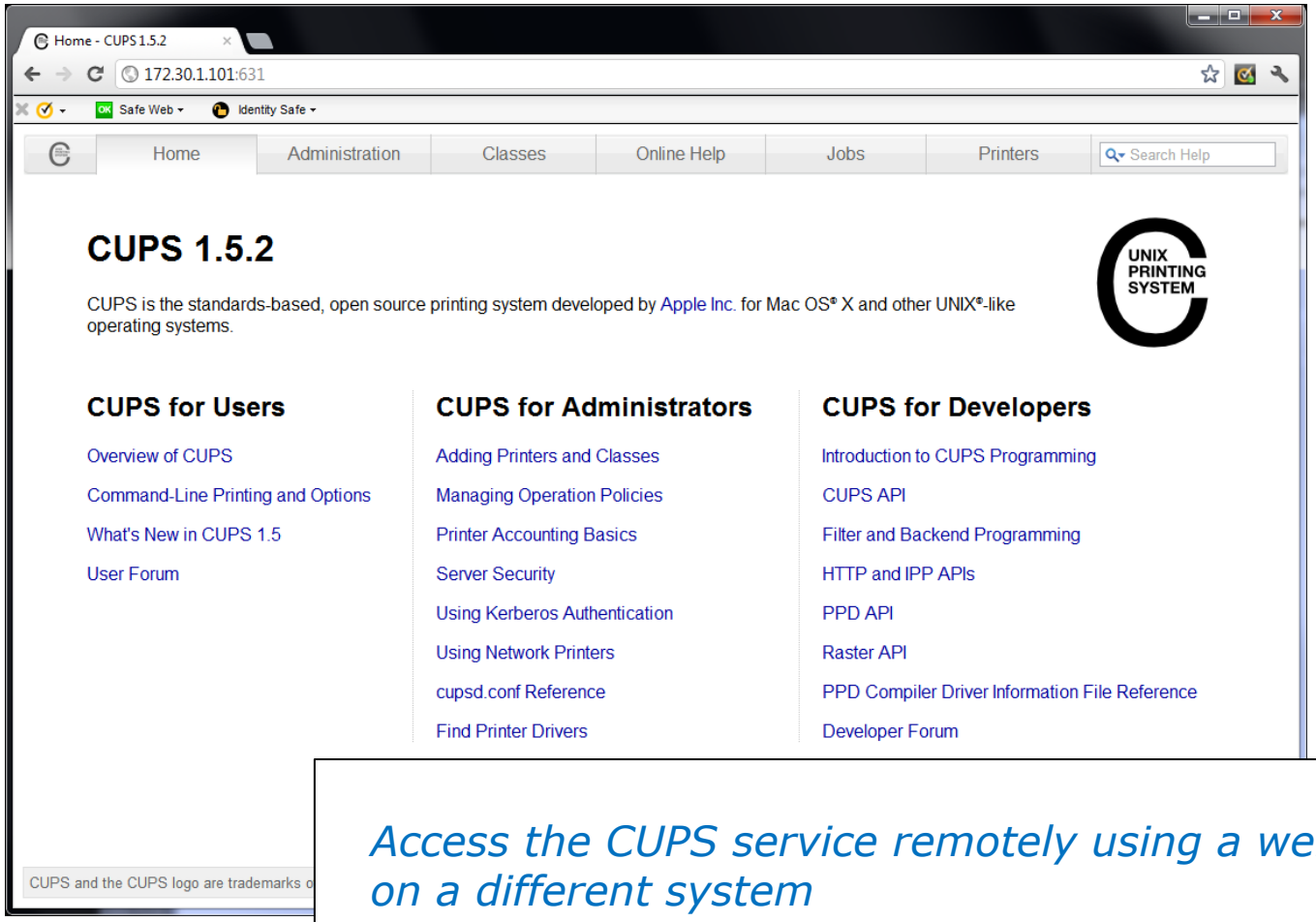
Access the CUPS service using a web browser with

```
rsimms@hugo:~$ firefox localhost:631 &
```

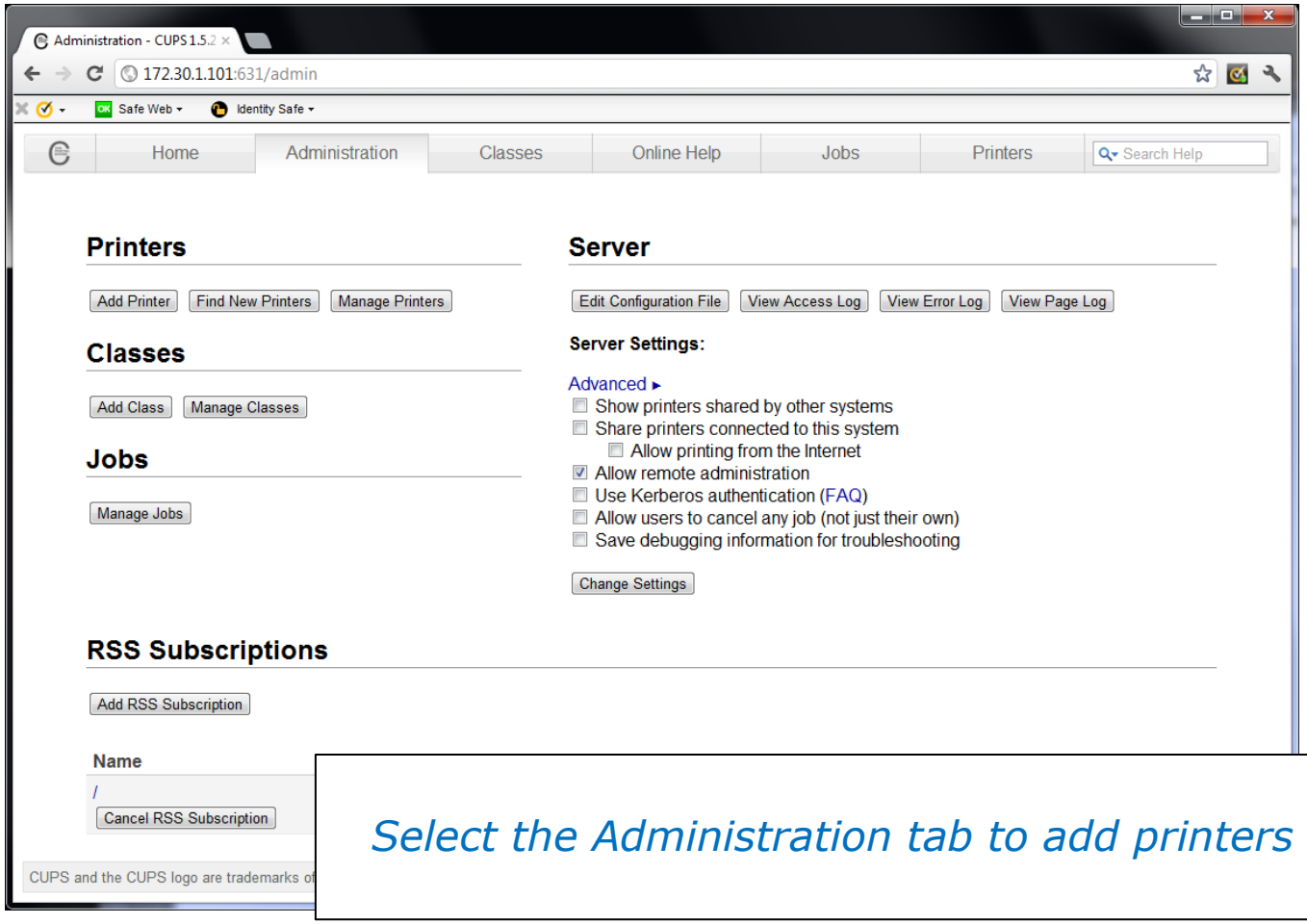

Hugo [Running] - Oracle VM VirtualBox
Machine View Devices Help
File Edit View History Bookmarks Tools Help
6:23 PM Rich Simms
Home - CUPS 1.5.2
localhost:631
Google
Home
Administration
Classes
Online Help
Jobs
Printers
Search Help
CUPS 1.5.2
CUPS is the standards-based, open source printing system developed by Apple Inc. for Mac OS® X and other UNIX®-like operating systems.
UNIX PRINTING SYSTEM
CUPS for Users
Overview of CUPS
Command-Line Printing and Options
What's New in CUPS 1.5
User Forum
CUPS for Administrators
Adding Printers and Classes
Managing Operation Policies
Printer Accounting Basics
Server Security
Using Kerberos Authentication
Using Network Printers
cupsd.conf Reference
Find Printer Drivers
CUPS for Developers
Introduction to CUPS Programming
CUPS API
Filter and Backend Programming
HTTP and IPP APIs
PPD API
Raster API
PPD Compiler Driver Information File Reference
Developer Forum
CUPS and the CUPS logo are trademarks of Apple Inc. CUPS is copyright 2007-2012 Apple Inc. All rights reserved.
Right Ctrl

FYI CIS 192

FYI CIS 192

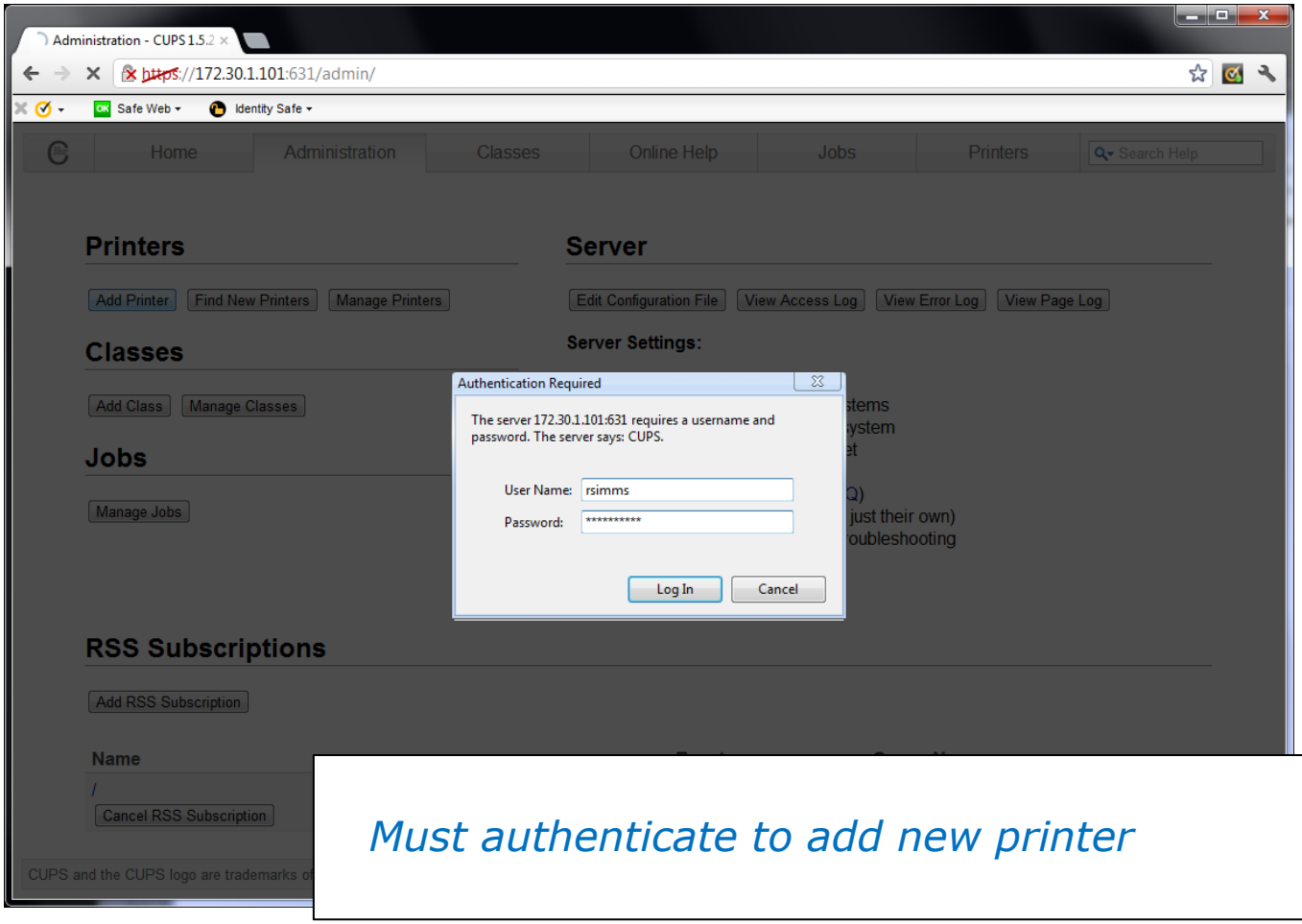


Access the CUPS service remotely using a web browser on a different system



Select the Administration tab to add printers

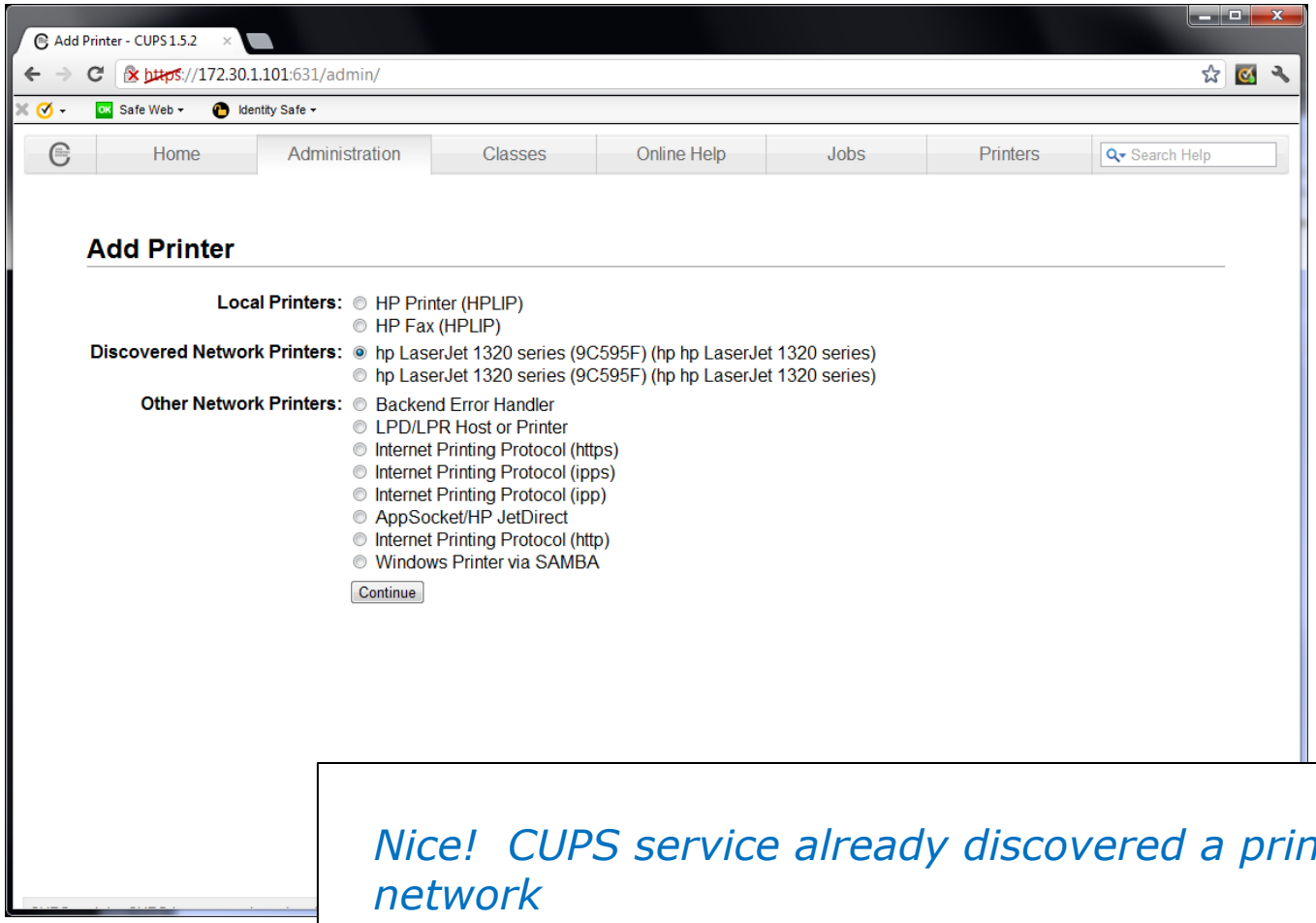
FYI
CIS 192



FYI
CIS 192

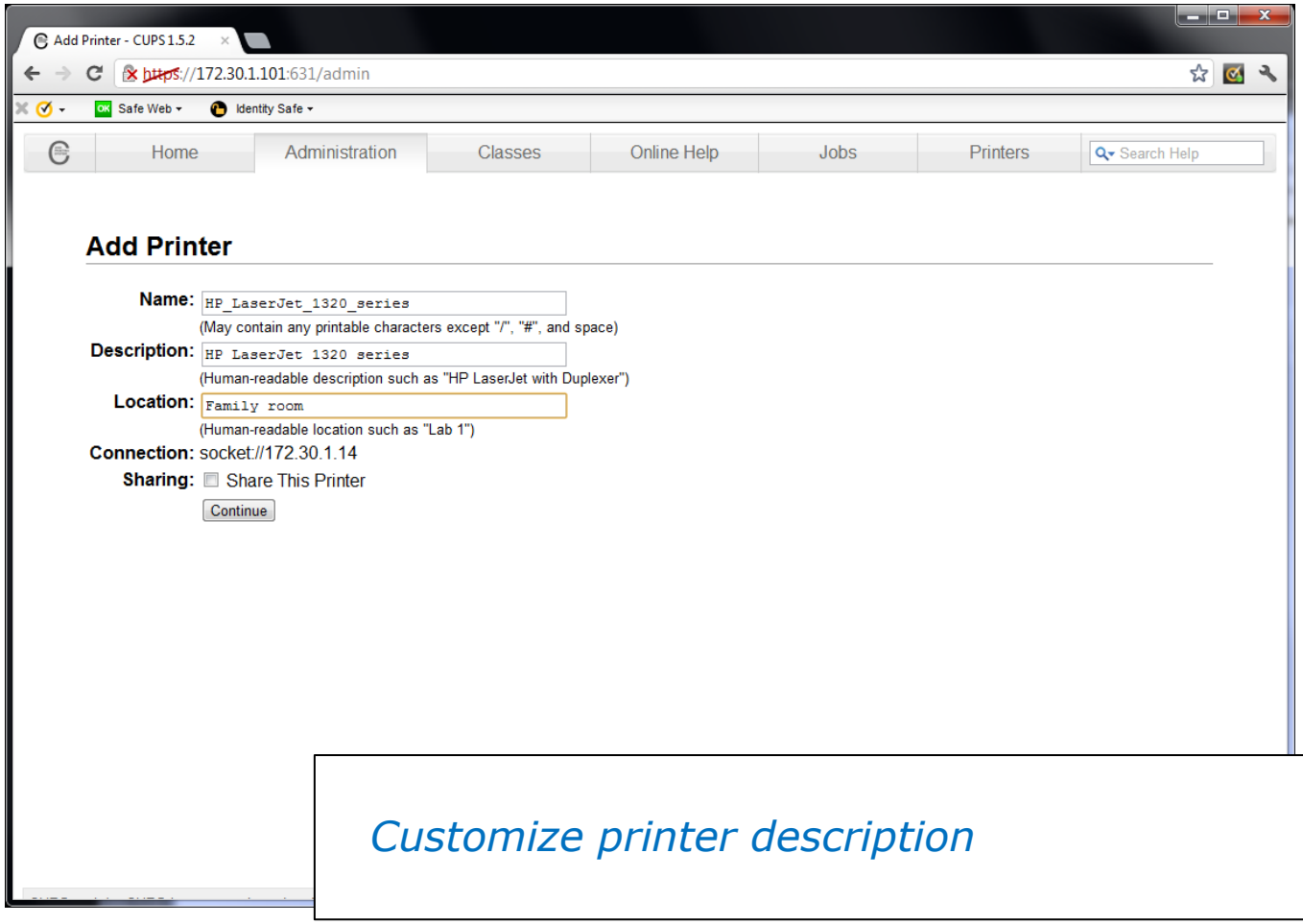
Must authenticate to add new printer

FYI
CIS 192

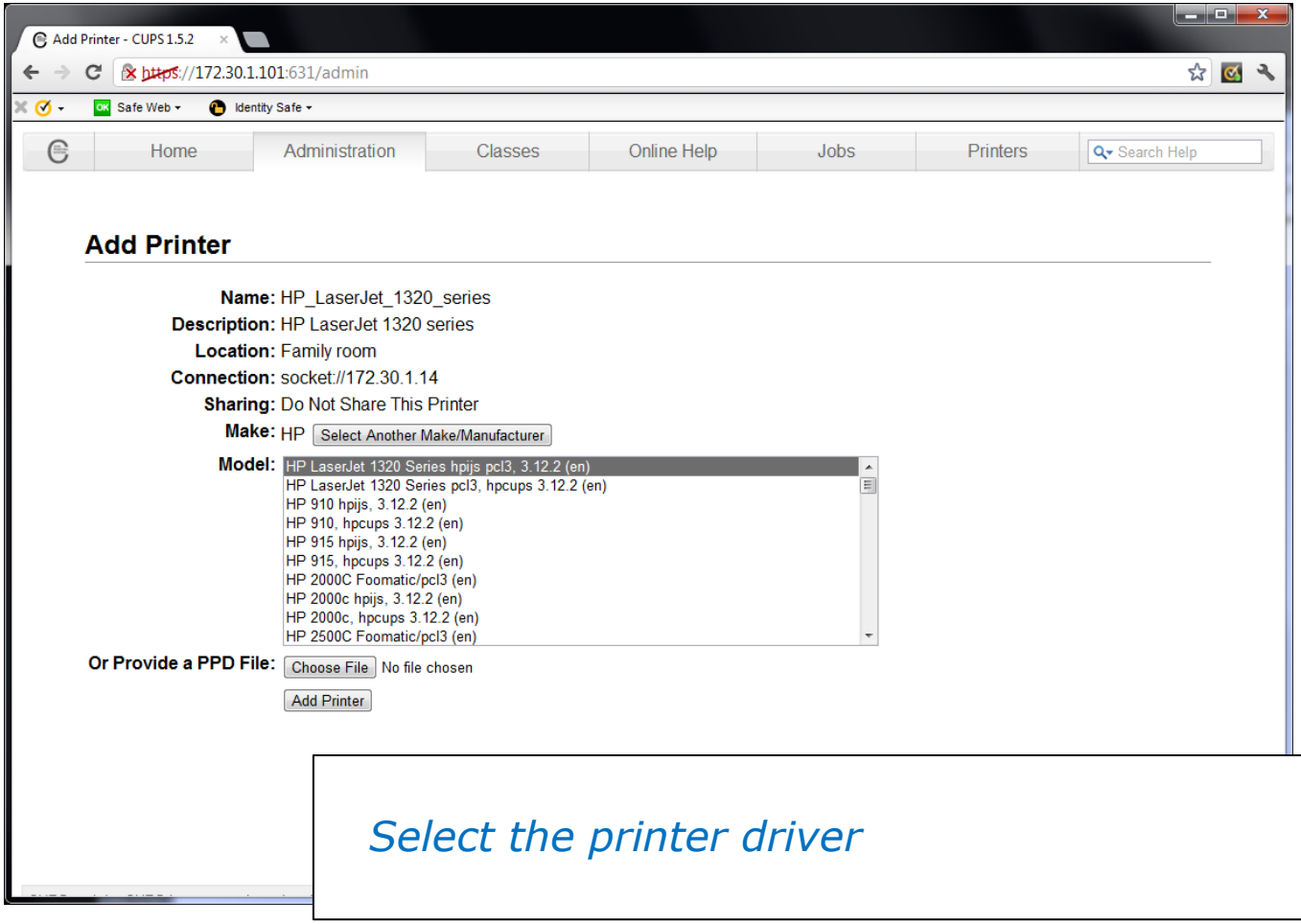


Nice! CUPS service already discovered a printer on the network

FYI CIS 192



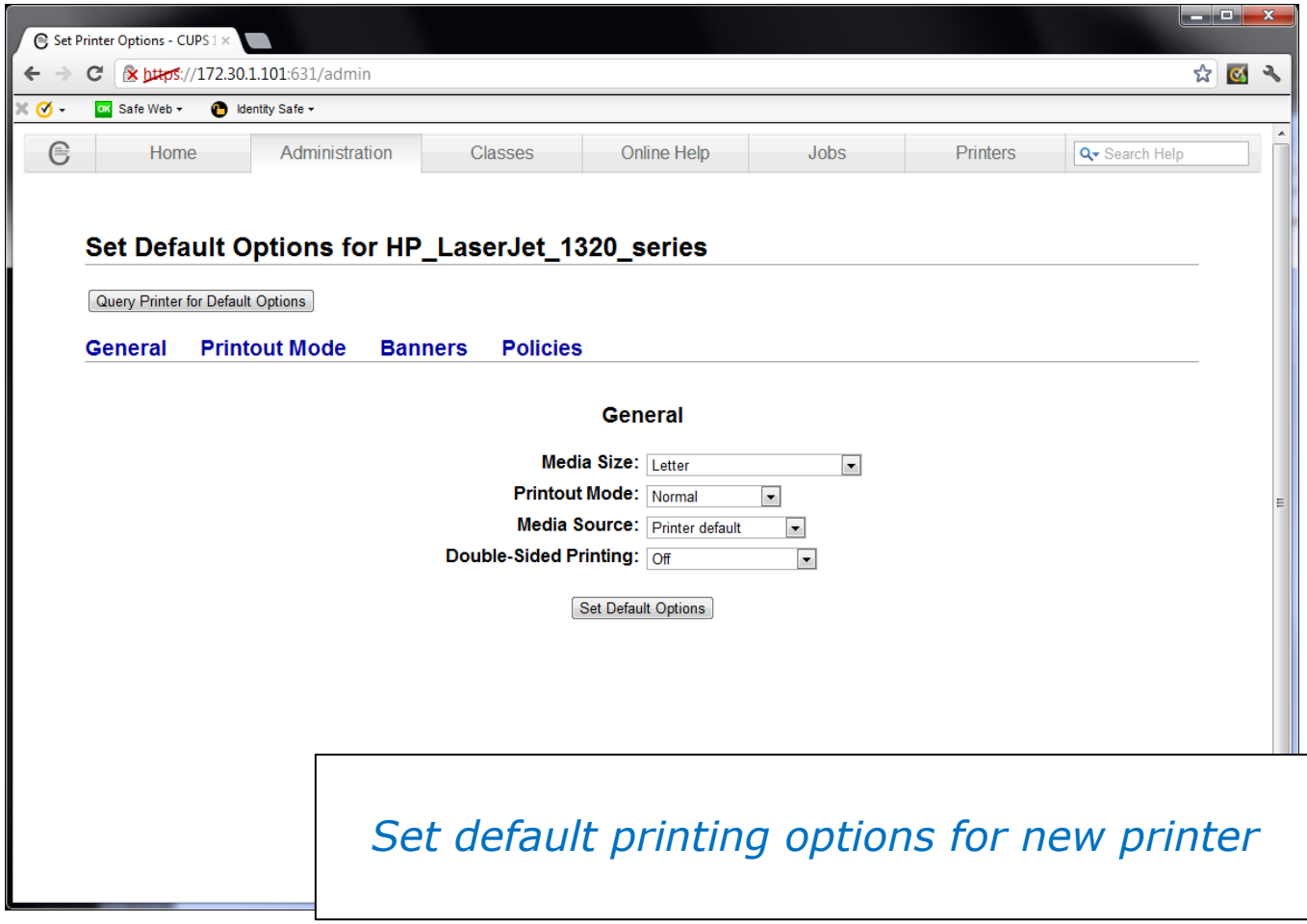
Customize printer description



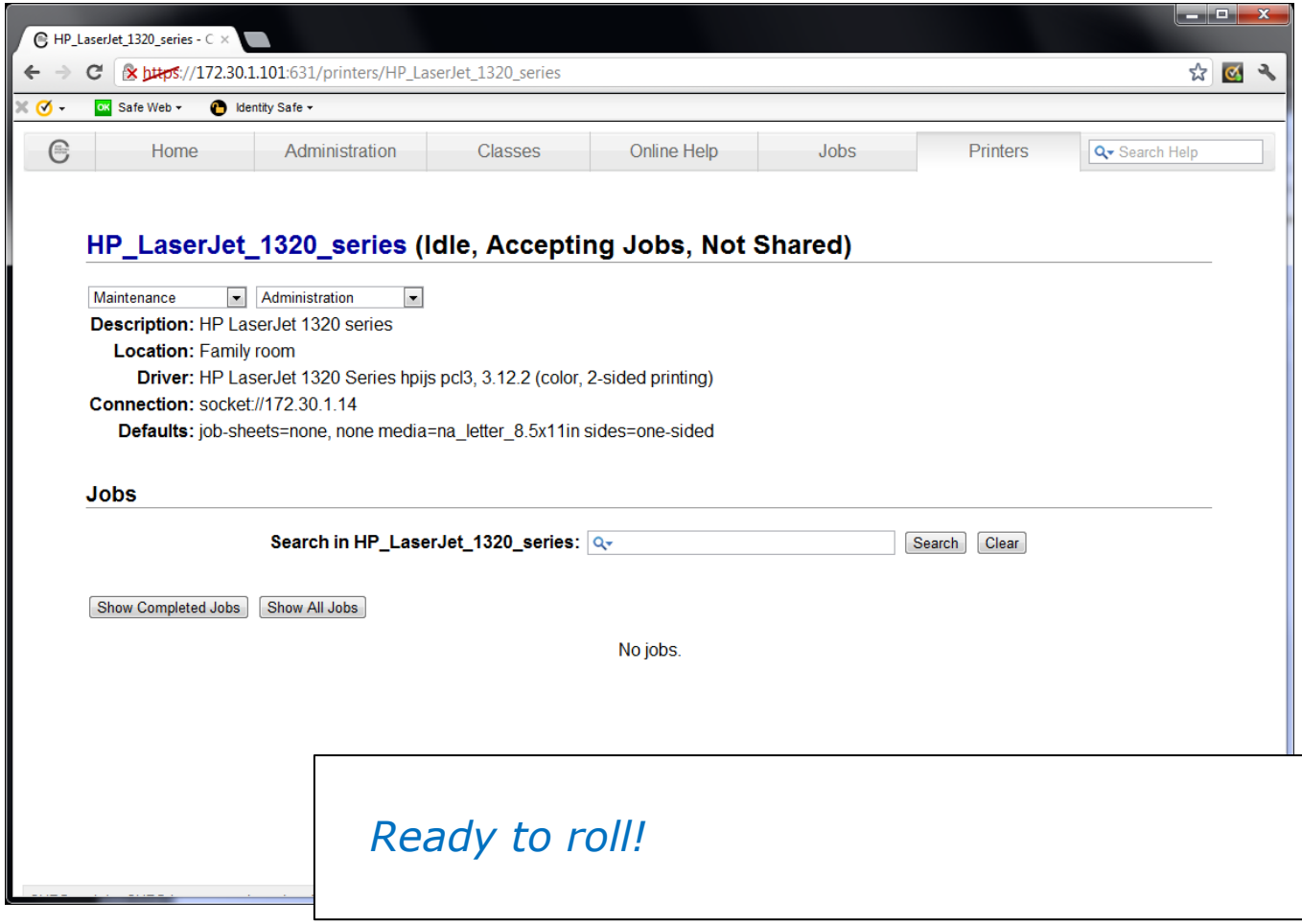
FYI
CIS 192

Select the printer driver

FYI CIS 192

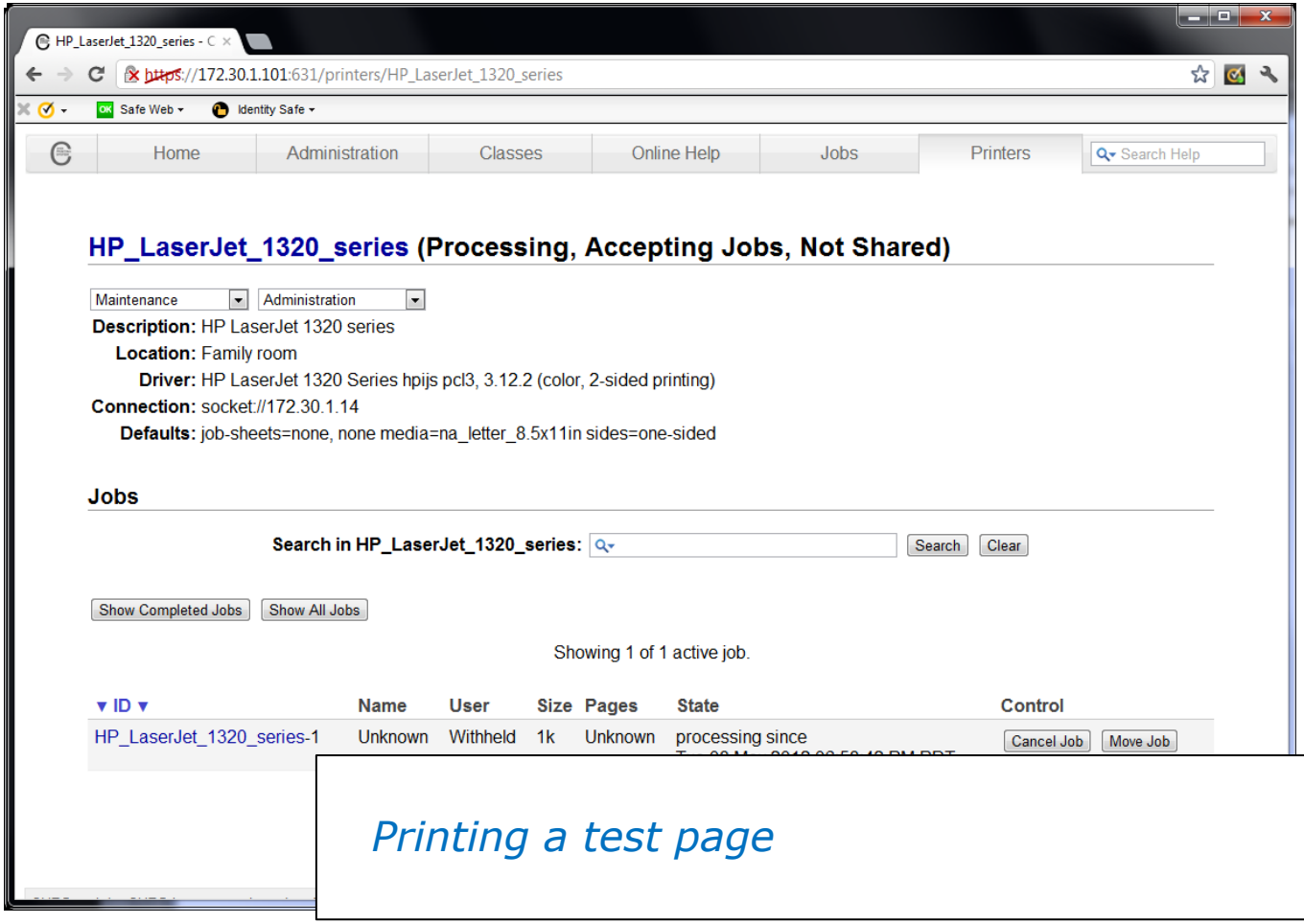


Set default printing options for new printer



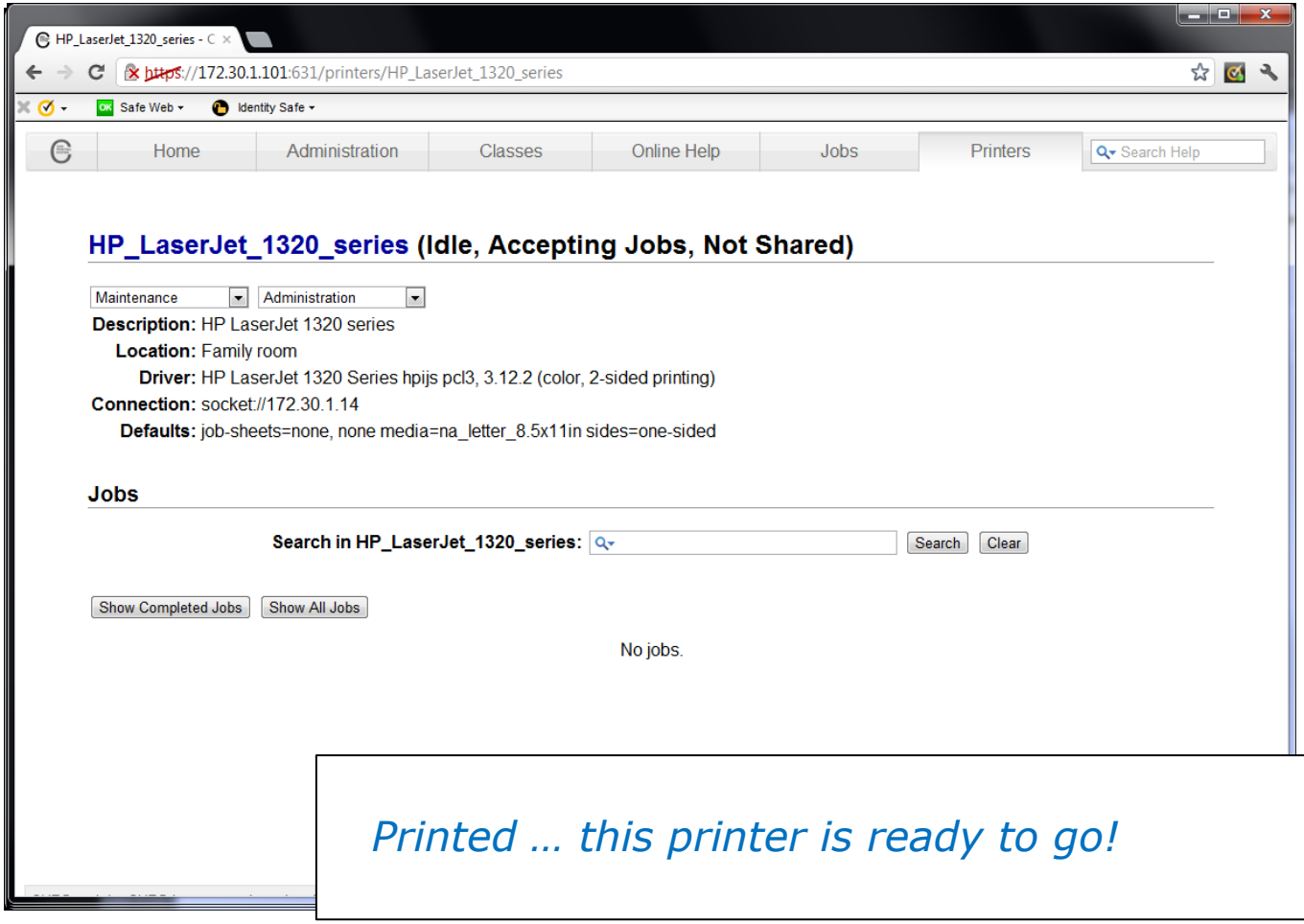
FYI
CIS 192

Ready to roll!



Printing a test page

FYI CIS 192



FYI
CIS 192

Printed … this printer is ready to go!

Printing in Linux

Printing Commands

ATT System V based print subsystem
- lp (to print)
- lpstat (queue management)
- cancel (to remove jobs)

BSD (Berkeley Software Distribution) based print subsystem
- lpr (to print)
- lpq (queue management)
- lprm (to remove jobs)

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

BSD is a branch of UNIX developed at the University of California, Berkeley

CUPS

lpstat command

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

On a system named Hugo

```
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*

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

CUPS

Practice Printing

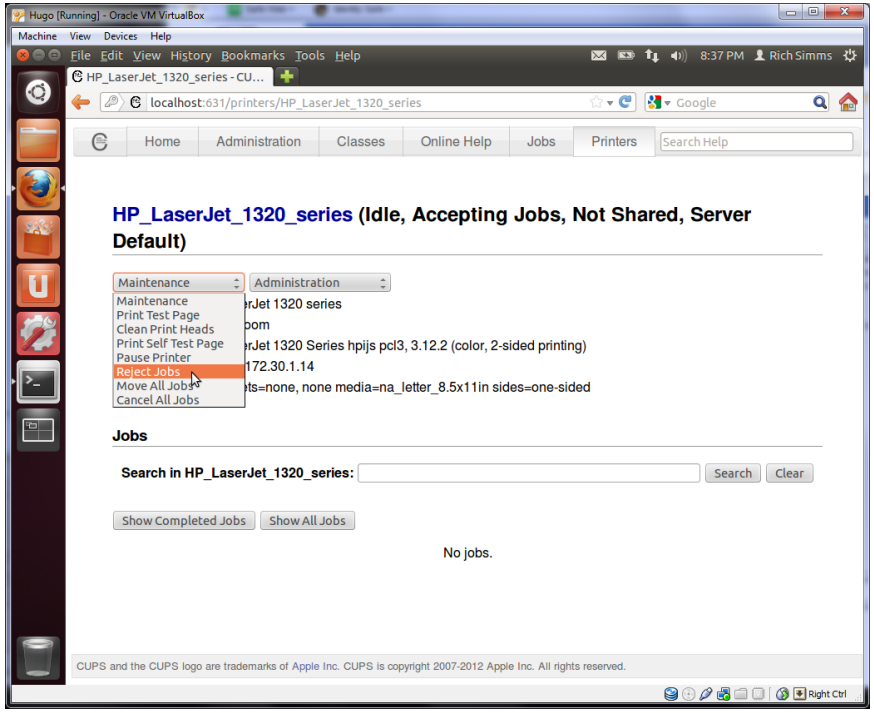
On Opus

```
lp lab10
lpstat


lpr letter
lpstat


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

Managing Print Jobs

CUPS

Rejecting Jobs



FYI
CIS 192

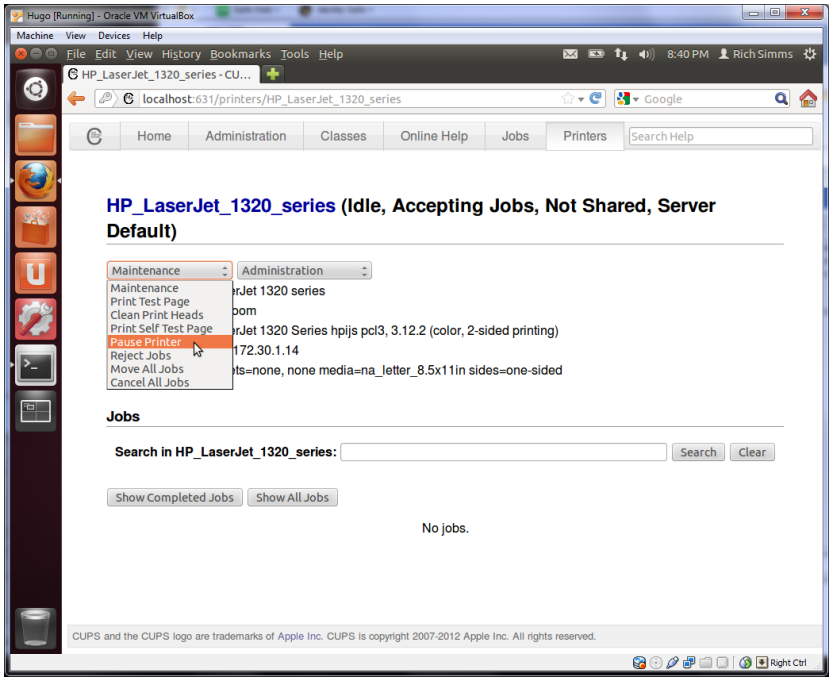
Clicking the Reject Jobs *selection on the web based utility will reject further jobs*

```
[root@benji ~]# lp myfile
lp: Destination "hp7550" is not accepting jobs.
[root@benji ~]#

[root@benji ~]# lpr myfile
lpr: Destination "hp7550" is not accepting jobs.
[root@benji ~]#
```

CUPS

Pausing the Printer

```
[root@benji ~]# lp myfile
request id is hp7550-22 (1 file(s))


[root@benji ~]# lpq
hp7550 is not ready
Rank    Owner   Job     File(s)
Total Size
1st     root    22      myfile
1024 bytes


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



Clicking the **Pause Printer** *selection on the web based utility will still allow jobs to be spooled*

FYI CIS 192

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** 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
```

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*

CUPS

Practice Printing

On Opus

```
lpq
lpstat
```

```
cancel <print job number>
lpq

lprm <print job number>
lpq
```

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 projects

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