Lesson Module Status

- Slides – draft
- Properties - done
- Flash cards –
- First minute quiz – done
- Web calendar summary – done
- Web book pages – done
- Commands – done
- Lab – done
- Supplies () - na
- Class PC's – na
- Chocolates - bringing

- Backup headset charged - done
- CCC Confer wall paper - done
- Slides & Lab uploaded - done
- Final project posted - done
- Extra credit lab posted - done

CIS 90 - Lesson 13


Dennis

Christopher


Francisco


Rich

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

Abd


Sarah


Astitow

Mike D.

Alex

Christine

Steven

Richie

Nathan


Tony


Sergio

Anthony

Fernando

Miguel

Lars


Jennifer

Rudy


Laura P.

Nick

Juan

Jacob


Andrew

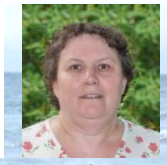

Luke

Saulius

Online Class Students

Edtson


James B.


Liz

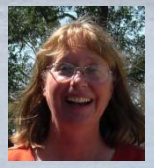

Casady

Jason

Aaron

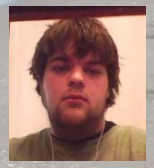

Steve

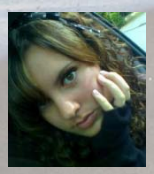

Songul

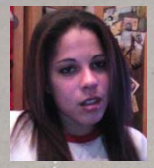

Stephanie


Victor

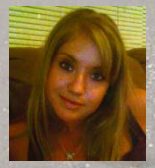
Tanya


Mike P.


Adriana


Laura S.


Olivia


Janelle

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

The LAST Quiz

Please close your books, notes, lesson materials, forum and answer these questions **in the order** shown:

email answers to: risimms@cabrillo.edu

(If you are in the classroom you can write your answers on a scrap piece of paper and hand it in)

[ ] Has the phone bridge been added?

[ ] Is recording on?

[ ] Does the phone bridge have the mike?

[ ] Share slides, putty (rsimms, simmsben, roddyduk), and Chrome

[ ] Disable spelling on PowerPoint

Printing

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

Questions

Previous material and assignment

1. Previous material
2. Lab 10

Housekeeping

Previous material and assignment

1. Lab 10 due midnight tonight
2. Grades Page – please check progress and grade choice
3. Extra Credit Labs X1 and X2 (30 points each)
4. Calendar endgame
5. Forum code tagging

Stuff

Silence is golden

Many UNIX commands that run successfully produce no output

```
[roddyduk@opus bin]$ alias details=file
[roddyduk@opus bin]$ cp quiet quiet.bak
[roddyduk@opus bin]$ umask 002
[roddyduk@opus bin]$ cat quiet > /dev/null
[roddyduk@opus bin]$
```

Silence is golden

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

```
[roddyduk@opus bin]$ cat quiet
alias details=file
cp quiet quiet.bak
umask 002
cat quiet > /dev/null

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

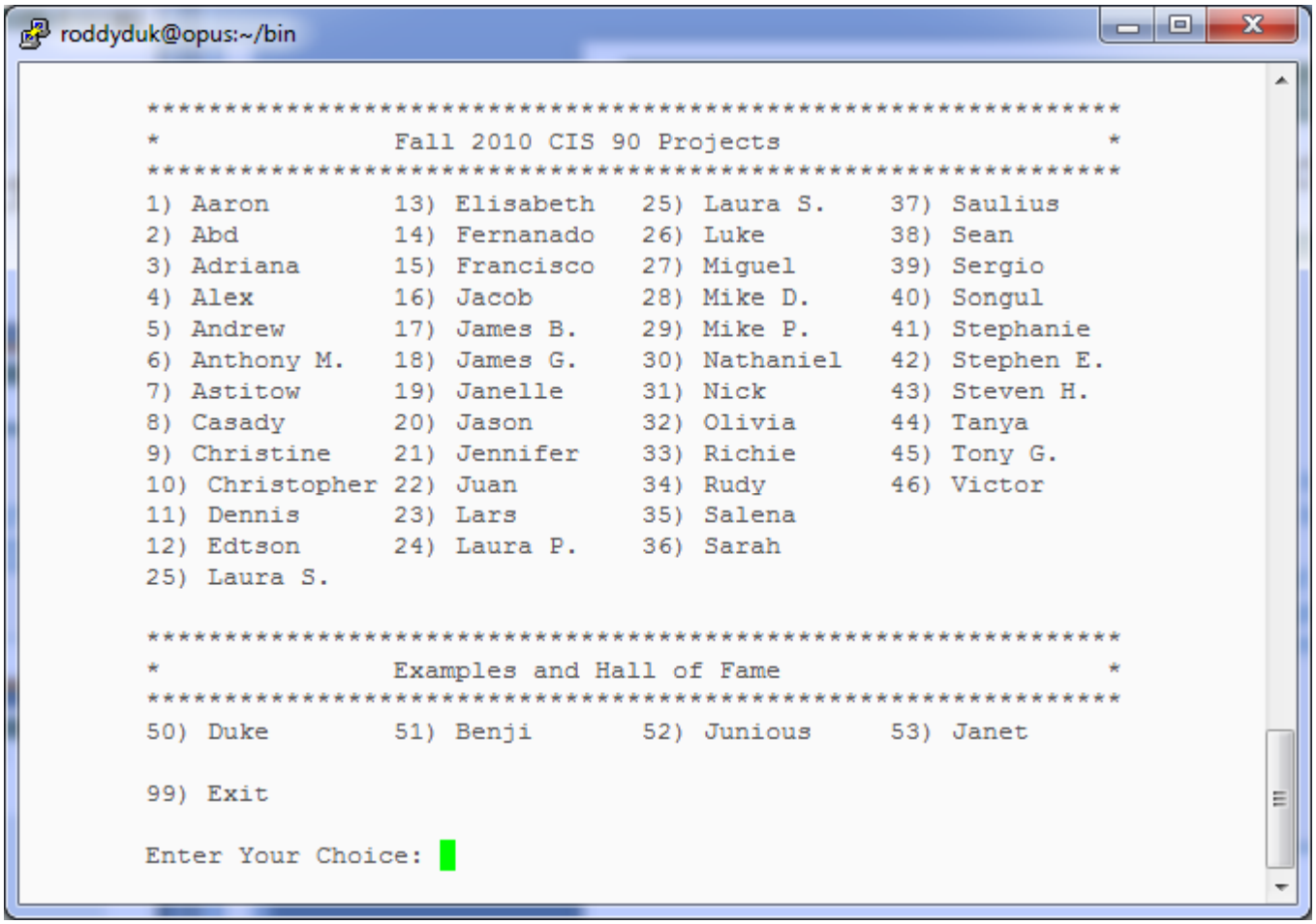
Silence is golden

Shell script developers will use the echo command to provide some indication of status or progress with the scripts they write.

```
[roddyduk@opus bin]$ cat quiet
alias details=file
cp quiet quiet.bak
umask 002
cat quiet > /dev/null
echo "Quiet script successfully completed"

[roddyduk@opus bin]$ quiet
Quiet script successfully completed
[roddyduk@opus bin]$ source quiet
Quiet script successfully completed
[roddyduk@opus bin]$
```

final project permissions

Final Project



There are several example scripts available.

Benji went a tad overboard with some of his scripts (he will do anything for some chicken)

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!
~~~~~~

Permissions

rsimms@opus:/home/cis90/roddyduk/bin

```
[rsimms@opus bin]$ ls -l /home/cis90/*/bin/myscript
-rwxr-xr-x 1 antiden  cis90   481 Nov 17 14:03 /home/cis90/antiden/bin/myscript
-rwxrwxr-x 1 birmijam cis90   680 Nov 17 14:29 /home/cis90/birmijam/bin/myscript
-rw-rw-r-- 1 botoschr cis90   481 Nov 17 14:37 /home/cis90/botoschr/bin/myscript
-rwxrwxr-x 1 brownliz cis90   690 Nov 17 16:28 /home/cis90/brownliz/bin/myscript
-rwxrw-r-- 1 dakkaabd cis90   738 Nov 17 14:47 /home/cis90/dakkaabd/bin/myscript
-rwxr-xr-x 1 daviesar cis90  1408 Nov 17 14:31 /home/cis90/daviesar/bin/myscript
-rwxrwxr-x 1 dawadast cis90   481 Nov 17 14:10 /home/cis90/dawadast/bin/myscript
-rwxr-xr-x 1 garciton cis90   761 Nov 17 14:23 /home/cis90/garciton/bin/myscript
-rwxr-xr-x 1 hrdinste cis90   662 Nov 17 14:25 /home/cis90/hrdinste/bin/myscript
-rwxrwxr-x 1 komicser cis90   658 Nov 19 12:49 /home/cis90/komicser/bin/myscript
-rwxrwxr-x 1 mottste  cis90   808 Nov 17 14:27 /home/cis90/mottste/bin/myscript
-rwxrwx--x 1 orozcmig cis90   685 Nov 22 15:45 /home/cis90/orozcmig/bin/myscript
-rwxrwxr-x 1 palmilar cis90   590 Nov 17 14:24 /home/cis90/palmilar/bin/myscript
-rwxrwxrwx 1 parrijen cis90   640 Nov 17 14:25 /home/cis90/parrijen/bin/myscript
-rwxr-xr-x 1 pennitan cis90   481 Nov 17 14:02 /home/cis90/pennitan/bin/myscript
-rwxr-xr-x 1 pirkllau cis90 10761 Nov 22 15:34 /home/cis90/pirkllau/bin/myscript
-rwxrw-r-- 1 rochajua cis90   493 Nov 17 14:30 /home/cis90/rochajua/bin/myscript
-rwxr-xr-x 1 roddyduk cis90  4297 Nov 23 14:04 /home/cis90/roddyduk/bin/myscript
-rwxrw-r-- 1 salinjac cis90   656 Nov 17 21:22 /home/cis90/salinjac/bin/myscript
-rwxr-xr-x 1 simmsben cis90 10511 Nov 13 13:27 /home/cis90/simmsben/bin/myscript
-rwxr-xr-x 1 srecklau cis90   626 Nov 21 09:50 /home/cis90/srecklau/bin/myscript
-rwxr-xr-x 1 valaddre cis90   627 Nov 17 14:37 /home/cis90/valaddre/bin/myscript
-rwxrwxr-x 1 velasliv cis90   633 Nov 17 15:10 /home/cis90/velasliv/bin/myscript
[rsimms@opus bin]$
```

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?

chmod 750 myscript

You should set your permissions so that other CIS 90 class members can run and view your script.

It's up to you it you want to give them write access as well!

Permissions

Why can other classmates write to my new files?

```
/home/cis90/roddyduk/bin $ touch newscript
/home/cis90/roddyduk/bin $ ls -l newscript
-rw-rw-r-- 1 roddyduk cis90 0 Nov 23 16:17 newscript

/home/cis90/roddyduk/bin $ chmod +x newscript
/home/cis90/roddyduk/bin $ ls -l newscript
-rwxrwxr-x 1 roddyduk cis90 0 Nov 23 16:17 newscript

/home/cis90/roddyduk/bin $ umask
0002
```

With a umask setting of 002 all new files you create allow write permission to group users:

```
 666
-002
----
 664 = rw-rw----
```

Before Lab 10

Permissions

Why can other classmates write to my new files?

```
/home/cis90/roddyduk/bin $ touch newscript
/home/cis90/roddyduk/bin $ ls -l newscript
-rw-rw---- 1 roddyduk cis90 0 Nov 23 16:17 newscript

/home/cis90/roddyduk/bin $ chmod +x newscript
/home/cis90/roddyduk/bin $ ls -l newscript
-rwxrwx--x 1 roddyduk cis90 0 Nov 23 16:17 newscript

/home/cis90/roddyduk/bin $ umask
0006
```

With a umask setting of 006 all new files you create allow write permission to group users:

```
 666
-006
----
 660 = rw-rw----
```

After Lab 10

Permissions

```
[roddyduk@opus bin]$ cat /home/cis90/roddyduk/.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 your change this setting to 006.

- Change your umask to 022
- Can group or other user modify your 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?

dates

Fun with Dates

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

/home/cis90/roddyduk $ date +'%m/%d/%y'
11/26/08

/home/cis90/roddyduk $ date +'%m/%d/%Y'
11/26/2008

/home/cis90/roddyduk $ date +'%m/%d/%Y and %N nanoseconds'
11/26/2008 and 334957229 nanoseconds

/home/cis90/roddyduk $ date +'Time: %H hours and %M minutes'
Time: 15 hours and 41 minutes

/home/cis90/roddyduk $ man date
```

See the man page for lots of other % sequences

- Write a short script, named mydate, that prints out the date as mm/dd/yyyy

tips on scripts

Don't name your scripts "script"

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


[roddyduk@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"

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

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

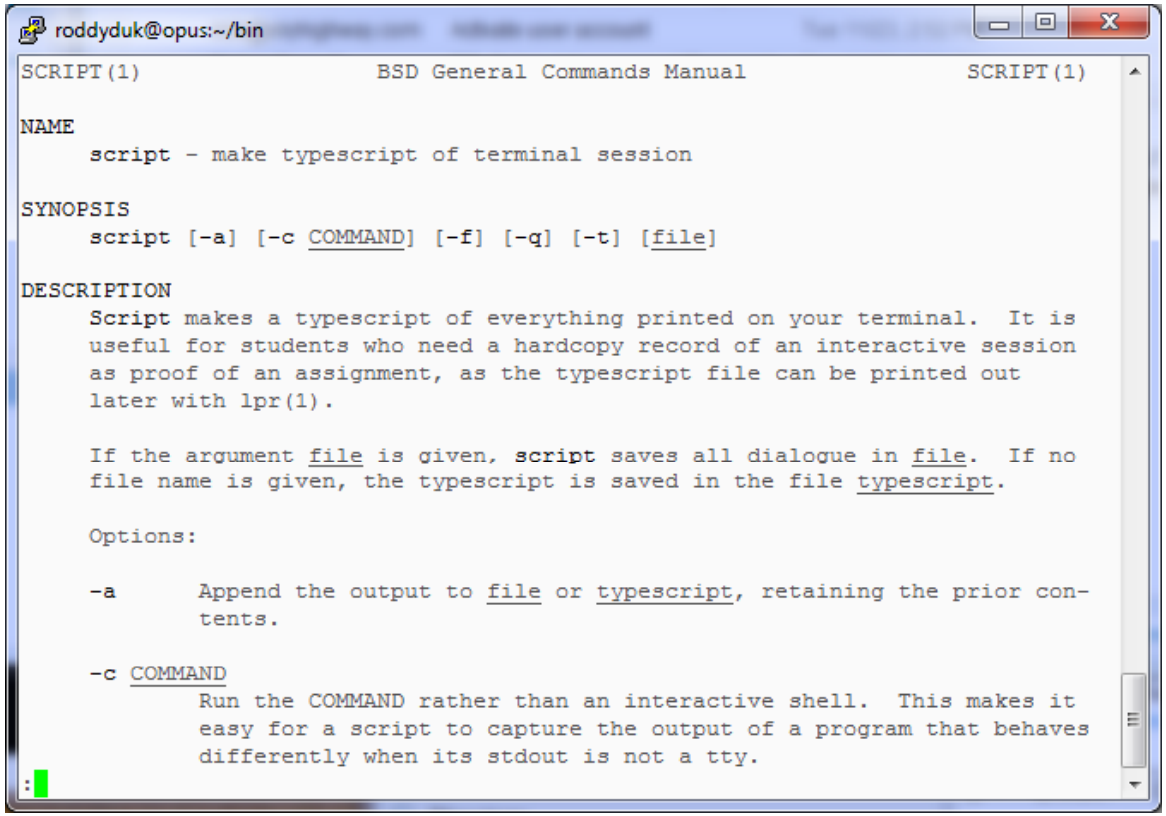
Why the heck doesn't my script do what it's supposed to do?

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

Script done on Wed 13 May 2009 08:00:47 AM PDT
[roddyduk@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!

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



Don't name your scripts "script"

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

```
[roddyduk@opus bin]$ type script
script is hashed (/usr/bin/script)
[roddyduk@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
```

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

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

Don't name your scripts "script"

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

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

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

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

[roddyduk@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/roddyduk/bin

Try the script command

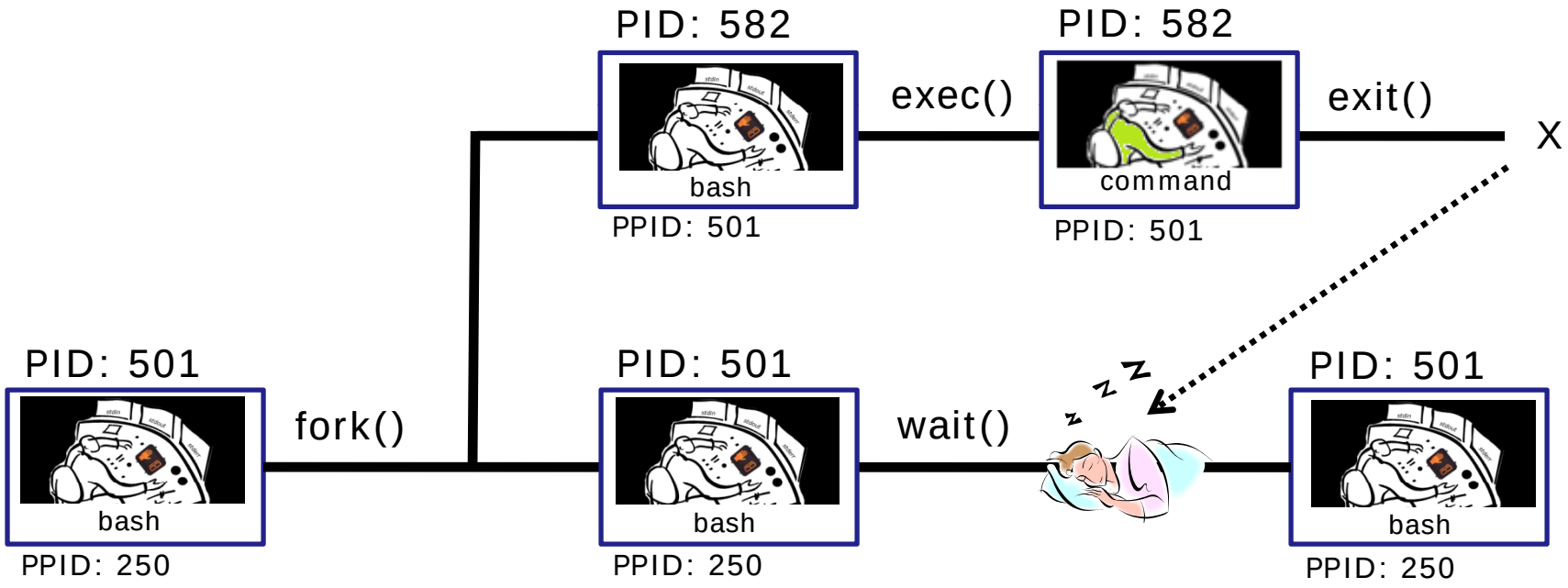
- Use the **script** command to start recording
- Type various commands of your choice
- Type **exit** or hit Ctrl-D to end recording
- Use **cat typescript** to see what you recorded

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

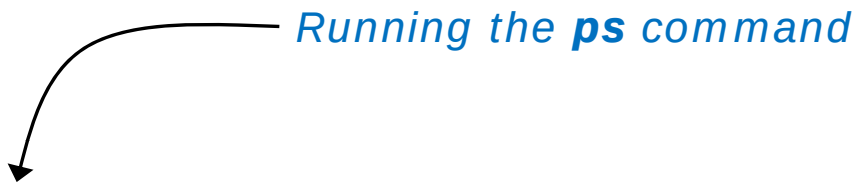
Refresh

Process Life Cycle

Process Lifecycle



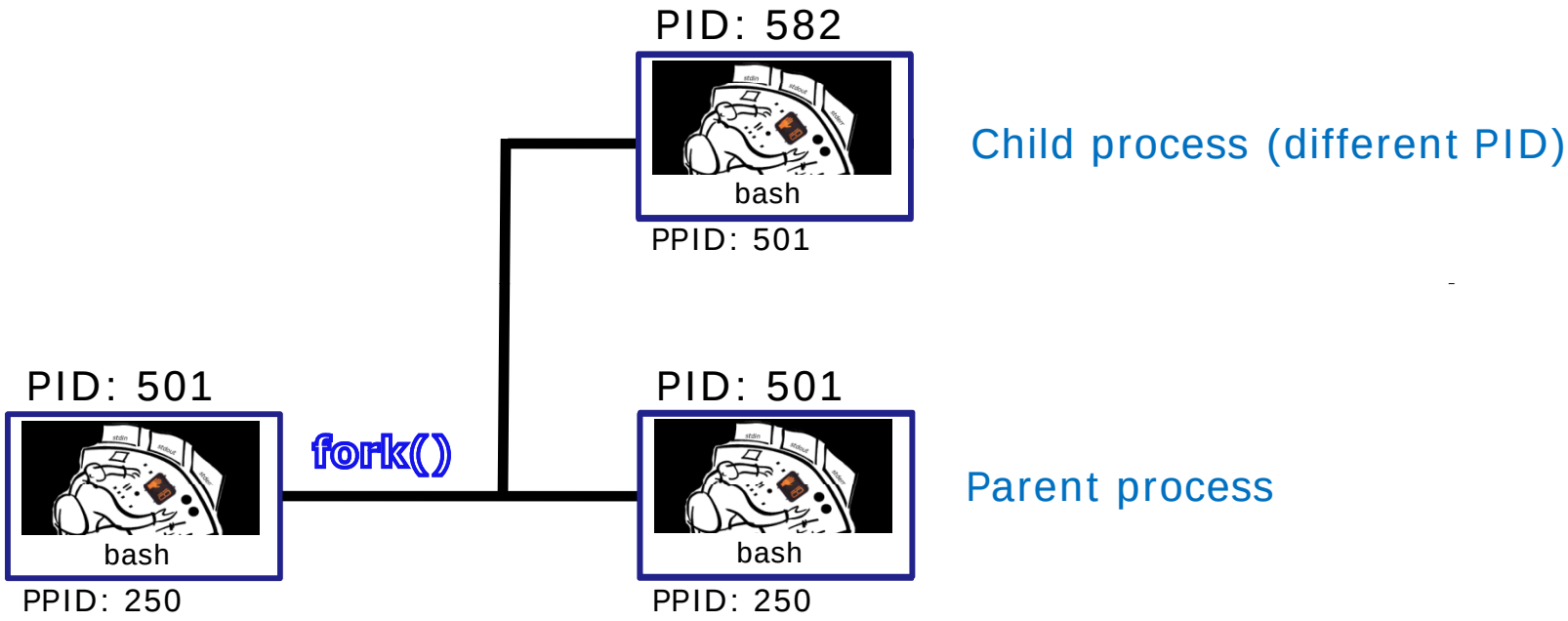
An example command




```
[rsimms@opus ~]$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S   201  6204  6203  0  75   0 -  1165 wait   pts/6    00:00:00 bash
0 R   201  6521  6204  0  77   0 -  1050 -      pts/6    00:00:00 ps
```

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

Process Lifecycle

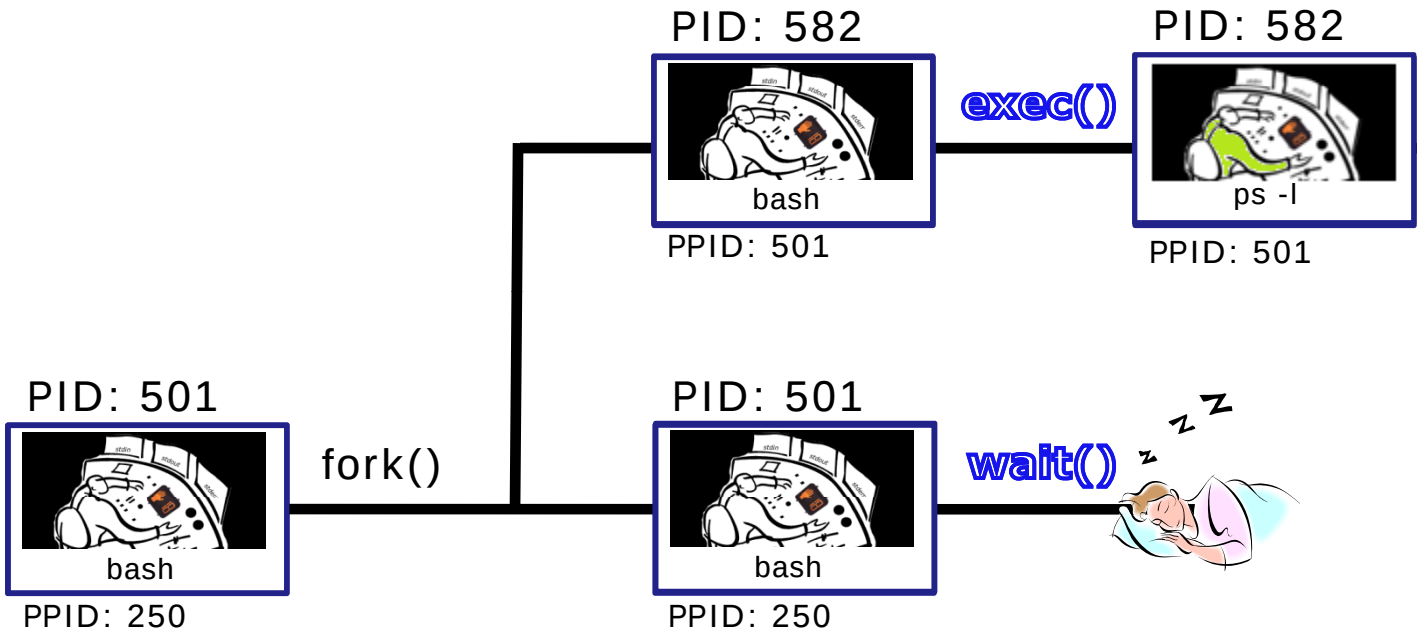


1) When a program is loaded into memory a new process must be created.

This is done by the **parent** process (bash) making a copy of itself using the fork system call.

The new **child** process is a duplicate of the **parent** but it has a different PID.

Process Lifecycle



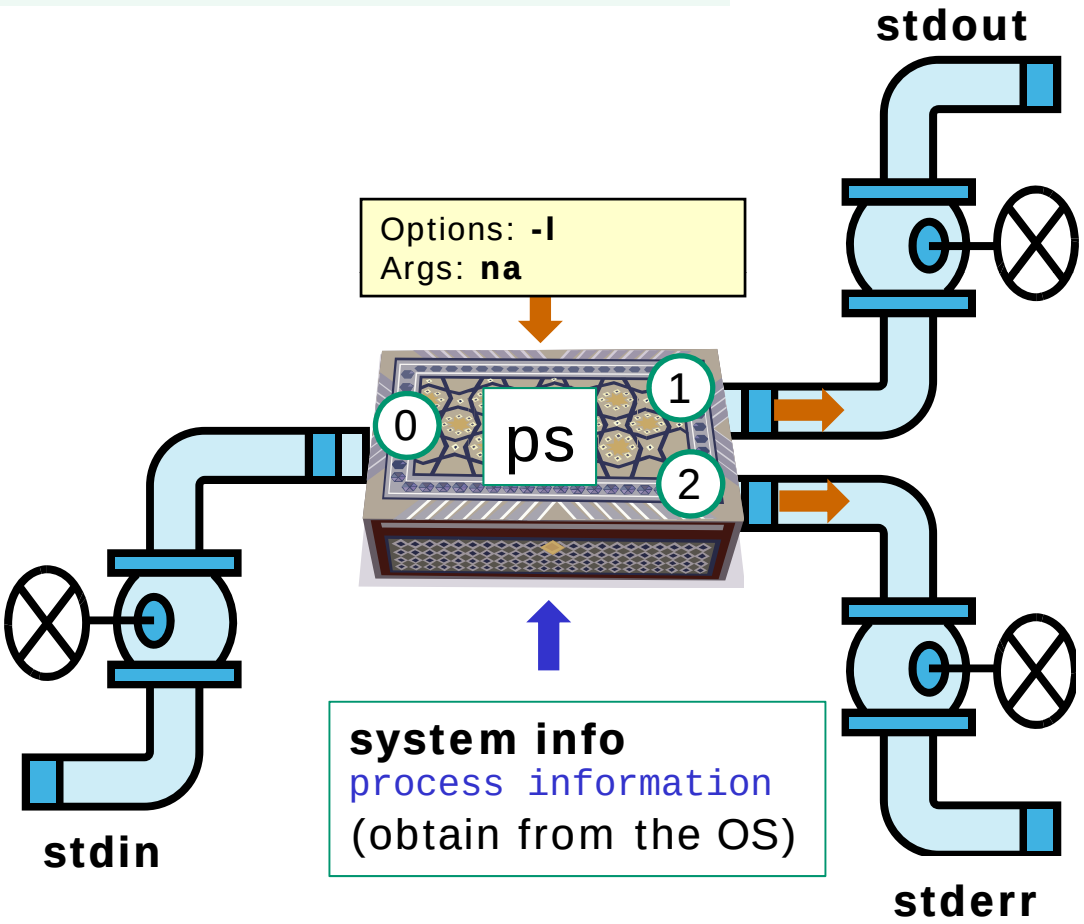
2) An exec system call is issued to overlay the **child** process with the instructions of the requested command. The new instructions then are executed.

The **parent** process issues the wait system call and goes to sleep.

When a program is run, its loaded into memory

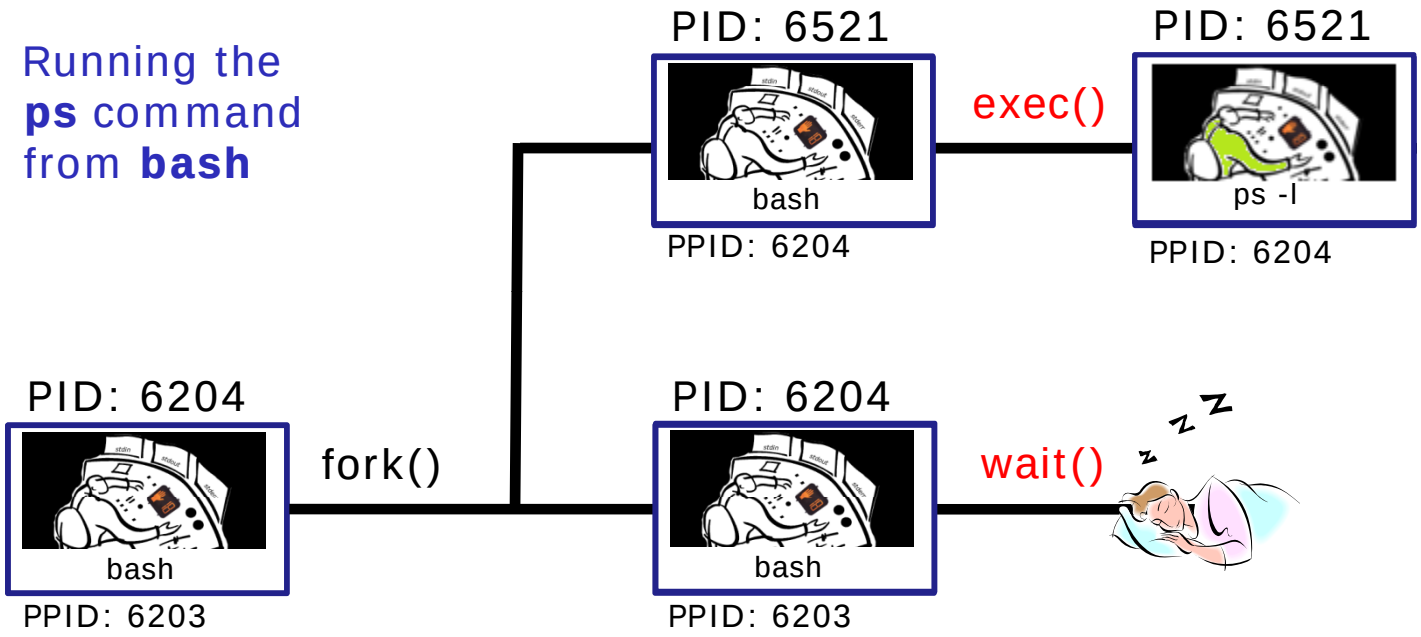
```
$ ps -l
```

```
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S   201  6204  6203  0  75   0 -  1165 wait   pts/6    00:00:00 bash
0 R   201  6521  6204  0  77   0 -  1050 -      pts/6    00:00:00 ps
```



Process Lifecycle

Running the **ps** command from **bash**

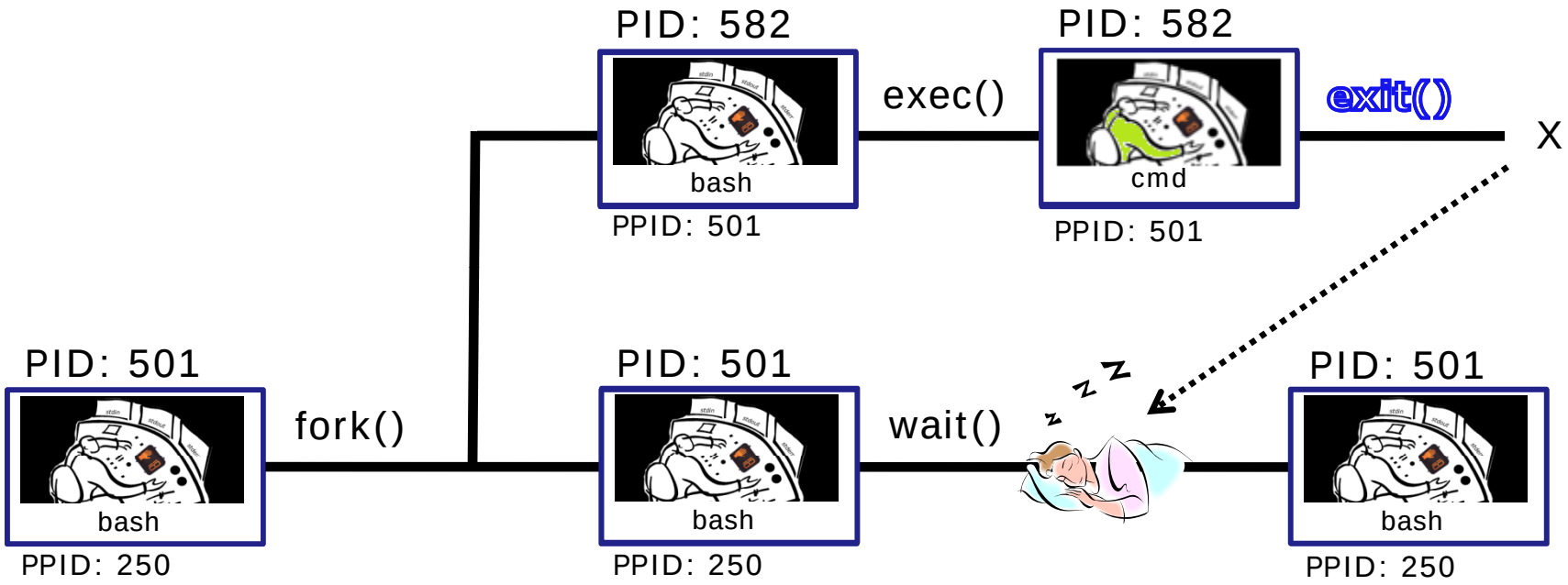



```
[rsimms@opus ~]$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S   201  6204  6203  0  75   0 -  1165 wait   pts/6    00:00:00 bash
0 R   201  6521  6204  0  77   0 -  1050 -      pts/6    00:00:00 ps
```

2) An **exec** system call is issued to overlay the **child** process with the instructions of the requested command. The new instructions then are executed.

The **parent** process issues the **wait** system call and goes to sleep.

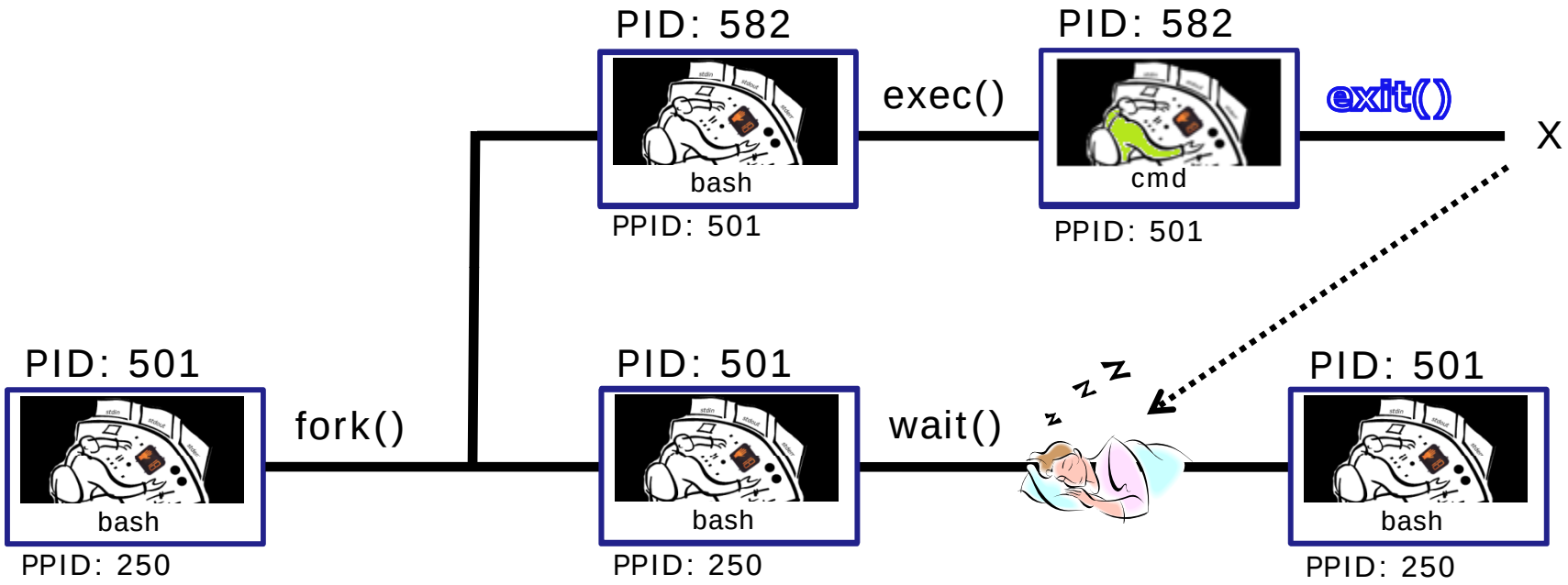
Process Lifecycle



3) When the **child** process finishes executing the instructions it issues the exit system call. At this point it gives up all its resources becomes a **zombie**.

The **parent** is woken up and once the **parent** has informed the kernel it has finished working with the **child**, the **child** process is killed and removed from the process table.

Process Lifecycle



3) If the **parent** process were to die before the **child**, the zombie will become an **orphan**. Fortunately the init process will adopt any orphaned **zombies**.

Signals

Signals

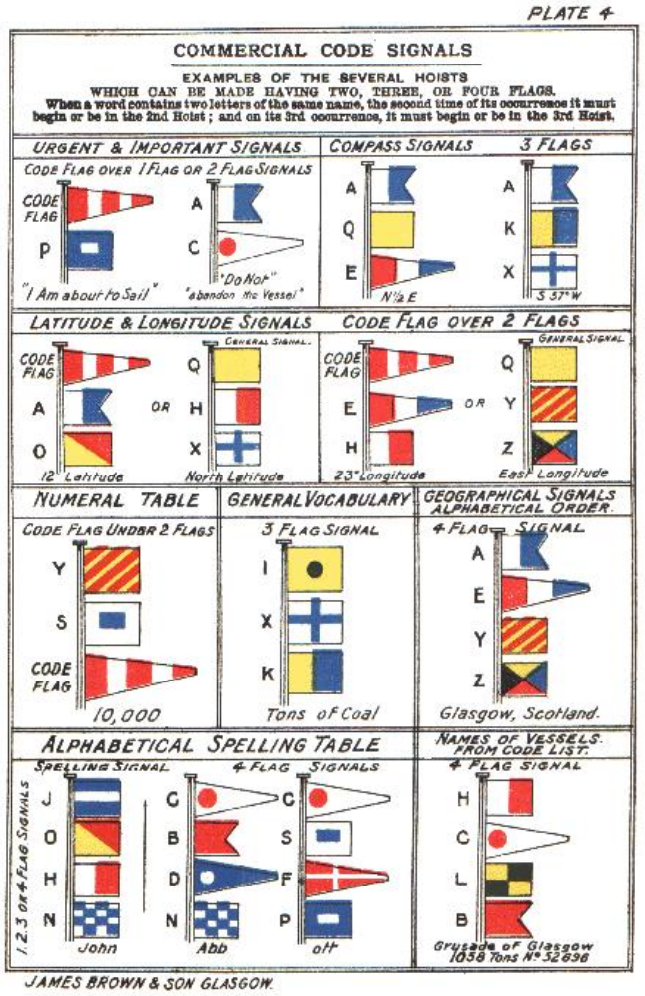


PLATE 4

COMMERCIAL CODE SIGNALS

EXAMPLES OF THE SEVERAL HOISTS WHICH CAN BE MADE HAVING TWO, THREE, OR FOUR FLAGS.

When a word contains two letters of the same name, the second time of its occurrence it must begin or be in the 2nd Hoist; and on its 3rd occurrence, it must begin or be in the 3rd Hoist.

URGENT & IMPORTANT SIGNALS — CODE FLAG OVER 1 FLAG OR 2 FLAG SIGNALS

CODE FLAG P — "I Am about to Sail"

A C — "Do Not" "abandon the Vessel"

COMPASS SIGNALS — 3 FLAGS

A Q E — N½E

A K X — S 37° W

LATITUDE & LONGITUDE SIGNALS — CODE FLAG OVER 2 FLAGS

CODE FLAG A O — OR — GENERAL SIGNAL Q H X — 12° Latitude — North Latitude

CODE FLAG E H — OR — GENERAL SIGNAL Q Y Z — 23° Longitude — East Longitude

NUMERAL TABLE — CODE FLAG UNDER 2 FLAGS

Y S CODE FLAG — 10,000

GENERAL VOCABULARY — 3 FLAG SIGNAL

I X K — Tons of Coal

GEOGRAPHICAL SIGNALS ALPHABETICAL ORDER. — 4 FLAG SIGNAL

A E Y Z — Glasgow, Scotland.

ALPHABETICAL SPELLING TABLE — SPELLING SIGNAL — 4 FLAG SIGNALS

1.2.3 or 4 FLAG SIGNALS

J O H N — John

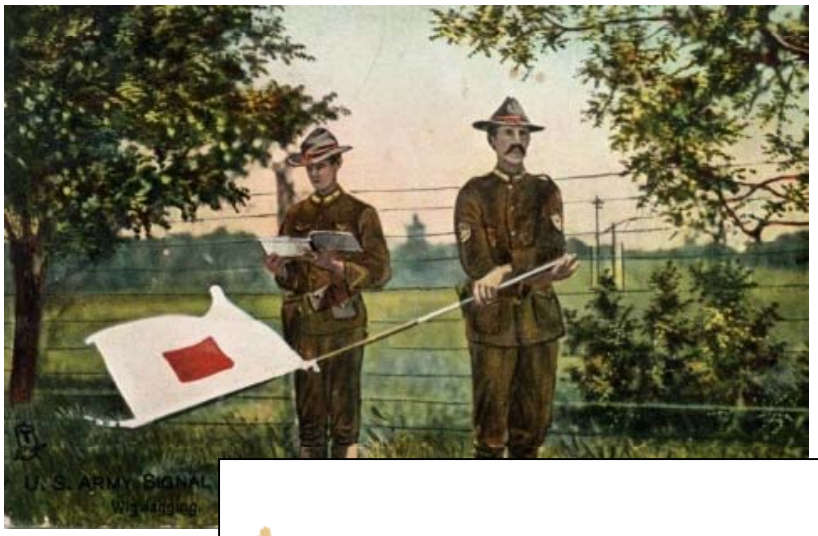
C B D N — Abb

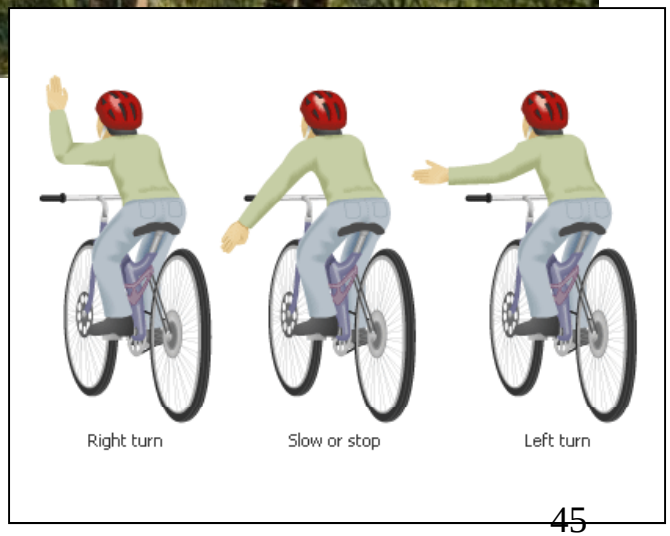
C S F P — ott

NAMES OF VESSELS FROM CODE LIST. — 4 FLAG SIGNAL

H C L B — Crusade of Glasgow 1858 Tons No 32696

JAMES BROWN & SON GLASGOW.





Signals

Signals are asynchronous messages sent to processes

They can result in one of three courses of action:

1. be ignored,
2. default action (die)
3. execute some predefined function.

Signals are sent:

- Using the kill command: **`$ kill -# PID`**
  - Where # is the signal number and PID is the process id.
  - if no number is specified, SIGTERM is sent.

- Using special keystrokes
  - limited to just a few signals

Use kill –l to see all signals

Signals

Signals are asynchronous messages sent to processes

Running process getting a signal

Asynchronous means it can happen at any time

Signals

SIGHUP	1	Hangup (POSIX)
SIGINT	2	Terminal interrupt (ANSI) ***Ctrl-C***
SIGQUIT	3	Terminal quit (POSIX) ***Ctrl-***
SIGILL	4	Illegal instruction (ANSI)
SIGTRAP	5	Trace trap (POSIX)
SIGIOT	6	IOT Trap (4.2 BSD)
SIGBUS	7	BUS error (4.2 BSD)
SIGFPE	8	Floating point exception (ANSI)
SIGKILL	9	Kill (POSIX) **can't be caught or ignored**
SIGUSR1	10	User defined signal 1 (POSIX)
SIGSEGV	11	Invalid memory segment access (ANSI)
SIGUSR2	12	User defined signal 2 (POSIX)
SIGPIPE	13	Write on a pipe with no reader, Broken pipe (POSIX)
SIGALRM	14	Alarm clock (POSIX)
SIGTERM	15	Termination (ANSI) **default kill signal when not specified**

Use kill –l to see all signals

Signals

SIGSTKFLT	16	Stack fault
SIGCHLD	17	Child process has stopped or exited, changed (POSIX)
SIGCONT	18	Continue executing, if stopped (POSIX)
SIGSTOP	19	Stop executing(can't be caught or ignored) (POSIX)
SIGTSTP	20	Terminal stop signal (POSIX) ***Ctrl-Z or Ctrl-F***
SIGTTIN	21	Background process trying to read, from TTY (POSIX)
SIGTTOU	22	Background process trying to write, to TTY (POSIX)
SIGURG	23	Urgent condition on socket (4.2 BSD)
SIGXCPU	24	CPU limit exceeded (4.2 BSD)
SIGXFSZ	25	File size limit exceeded (4.2 BSD)
SIGVTALRM	26	Virtual alarm clock (4.2 BSD)
SIGPROF	27	Profiling alarm clock (4.2 BSD)
SIGWINCH	28	Window size change (4.3 BSD, Sun)
SIGIO	29	I/O now possible (4.2 BSD)
SIGPWR	30	Power failure restart (System V)

Use kill –l to see all signals

Try and kill one of your login sessions

- Start up a second Putty session on Opus
- Use ps -u $LOGNAME
- Kill the second session from the first session
- Use kill *<-#>* *<PID>*
- Which process did you target? (bash, sshd, ...)
- Which signal did you send? (default, -9, ...)

Aliases

alias command
(a shell builtin)

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

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

alias command
showing all aliases

```
/home/cis90/roddyduk $ alias
alias bill='cd /home/cis90/roddyduk/poems/Shakespeare'
alias bye='clear;exit'
alias l.='ls -d .* --color=tty'
alias ll='ls -l --color=tty'
alias ls='ls --color=tty'
alias me='finger roddyduk'
alias print='echo -e'
alias rm='rm -i'
alias vi='vim'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'
/home/cis90/roddyduk $
```

*Typing **alias** by itself will show all your current aliases*

alias command

creating a new alias

```
/home/cis90/roddyduk $ alias s="clear; head -10 ~/edits/small_town"
/home/cis90/roddyduk $ s
HOW SMALL IS SMALL?

YOU KNOW WHEN YOU'RE IN A SMALL TOWN WHEN...
   The airport runaway is terraced.
   The polka is more popular than a mashpit on on Saturday night.
   Third Street is on the edge of town.
   Every sport is played on dirt.
   The editor and publisher of the newspaper carries a camera at all times.
   You don't use your turn signal because everyone knows where you are
      going knows where you are going.
```

*Make an alias, called **s**, that prints the first 10 lines of smalltown*

alias command

showing and deleting an alias

```
/home/cis90/roddyduk $ alias s="clear; head -10 ~/edits/small_town"


/home/cis90/roddyduk $ type s
s is aliased to `clear; head -10 ~/edits/small_town'
/home/cis90/roddyduk $ alias s
alias s='clear; head -10 ~/edits/small_town'
/home/cis90/roddyduk $ unalias s
/home/cis90/roddyduk $
```

Note the type command or the alias command will show an alias

alias command

strong (') or weak (") quote marks

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

double

```
$ alias p="echo $ac $fan $defrost"
$ alias p
alias p='echo on medium off'
$ p
on medium off
$ ac=off
$ p
on medium off
```

single

```
$ alias p='echo $ac $fan $defrost'
$ alias p
alias p='echo $ac $fan $defrost'
$ p
on medium off
$ ac=off
$ p
off medium off
```

Note: using strong quotes (') prevents bash from expanding the variables when setting up the alias

- Make this alias which we will use later

```
alias show='echo fan=$fan ac=$ac; type copy;  env | grep ac'
```

Shell Variables

Shell Variables

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

Environment Variables

Environment Variables

- A subset of the shell variables are environment variables.
- Environment variables are shell variables that have been exported.
- The **env** command will display the current environment variables and their values. Using the **export** command by itself will also show all the environment variables.
- The **export** command is used to make a shell variable into an environment variable. E.g. **dog=benji; export dog** creates a new environment variable named dog.
- The **export -n** command is used to make an environment variable back into a normal shell variable. E.g. **export -n dog** makes dog back into a regular shell variable.
- Child processes are provided copies of the parent's environment variables. Any changes made by the child will not effect the parent's copies.

Common Environment Variables

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

On Opus, PS1 is set in /etc/bashrc and then redefined in .bash_profile

Environment Variables

env command – show all environment variables

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

These are all shell variables that have been exported and they are available to child processes

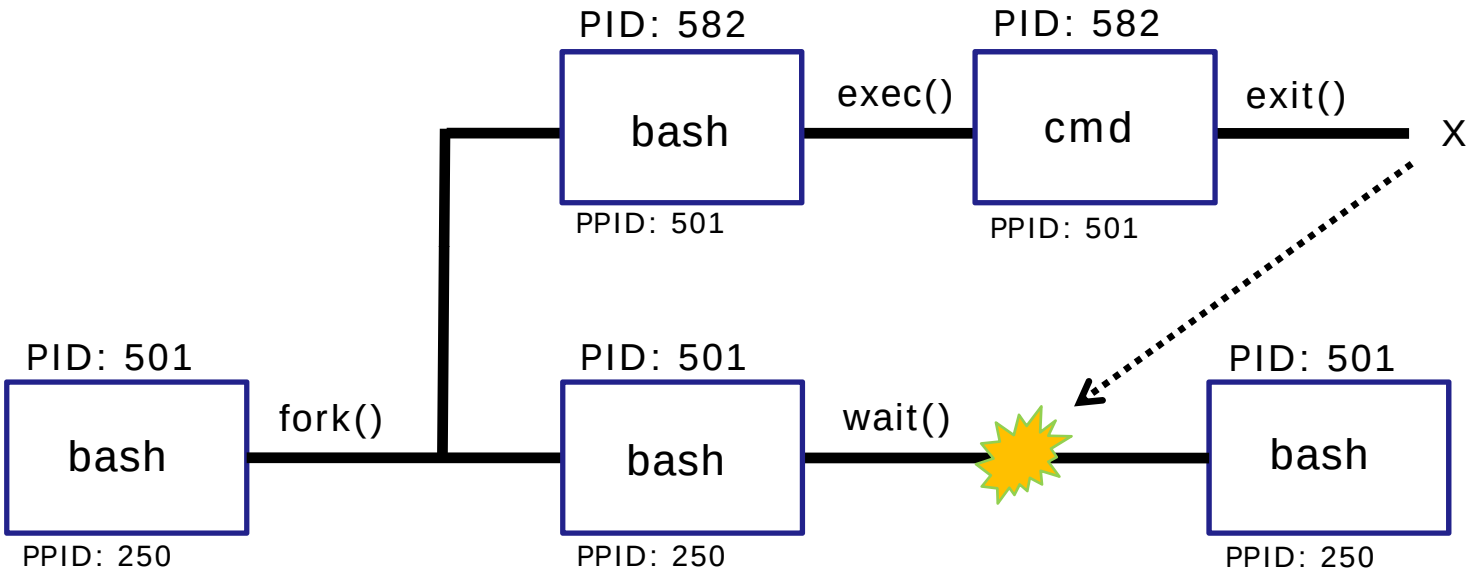
Environment Variables

export command – show all exported variables

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

These are all shell variables that have been exported and they are available to child processes

Children only see exported (environment) variables



When a shell forks a child, not all of the variables get passed on to the child. Only those the environment variables (which have been exported) are passed on to the child.

- Use **env** to see all the environment variables
- Use **export** to make a shell variable an environment variable and available to child processes e.g. **export BIRTHDAY**

Shell Environment

Customizing the shell environment

- It possible to customize your shell environment by editing the hidden **.bash_profile** and **.bashrc** files in your home directory.
- You can create and initialize shell variables.
- You can modify existing environment variables, e.g. PATH and PS1
- You can create new environment variables.
- You can modify or add new aliases
- You can specify the umask setting
- You can run commands or scripts

bash startup files

only executed when logging in

/etc/profile (all)

- adds root's special path

/etc/profile.d/*.sh (all)

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

.bash_profile (user specific)

- adds user's bin to path

.bashrc (user specific)

- add aliases here

To permanently customize your shell environment you modify these home directory files in Lab 10

/etc/bashrc (all)

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

.bash_profile

.bash_profile

- The .bash_profile is a shell script that sets up a user's shell environment.
- This script is run (sourced) each time the user logs in.
- The .bash_profile is used for initializing shell variables, running the user's .bashrc file, running basic commands like umask and set -o options.
- .bash_profile is not run for sub-shells

.bash_profile for CIS 90 accounts

simmsben@opus:~

```
# .bash_profile

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

# User specific environment and startup programs

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

~
~
~
~
~
~
~
".bash_profile" 19L, 354C                                    1,1           All
```

.bash_profile for CIS 90 accounts

roddyduk@opus:~

```
# .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 006
set -o ignoreeof
stty susp ^F
eval `tset -s -m vt100:vt100 -m :\?${TERM:-ansi} -r -Q `
mesg n
BIRTHDAY=05/05/93
export BIRTHDAY
riddle
~
~
~
~
                                                    16,1          All
```

After doing Lab 10

.bashrc

.bashrc

The .bashrc is a shell script that is executed during user login and whenever a new shell is invoked.

- This script is run (sourced) each time the user logs in.
- The .bashrc is typically used for defining aliases
- .bashrc is run for sub-shells (e.g. using the bash command to start a new sub-shell)

.bashrc for CIS 90 accounts

simmsben@opus:~

```
# .bashrc

# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
alias print="echo -e"
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
".bashrc" 9L, 146C                                    1,1           All
```

.bashrc for CIS 90 accounts

roddyduk@opus:~

```
# .bashrc

# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
alias print="echo -e"
alias bye="clear;exit"
alias rm="rm -i"
alias bill="cd /home/cis90/$LOGNAME/poems/Shakespeare"
alias me="finger $LOGNAME"
~
~
~
~
~
~
~
~
~
~
~
~
~
~
".bashrc" 13L, 268C                                    13,1          All
```

After doing Lab 10

. and exec

. and exec

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

Sometimes it is desireable 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.

. ***myscript*** or **source** ***myscript***

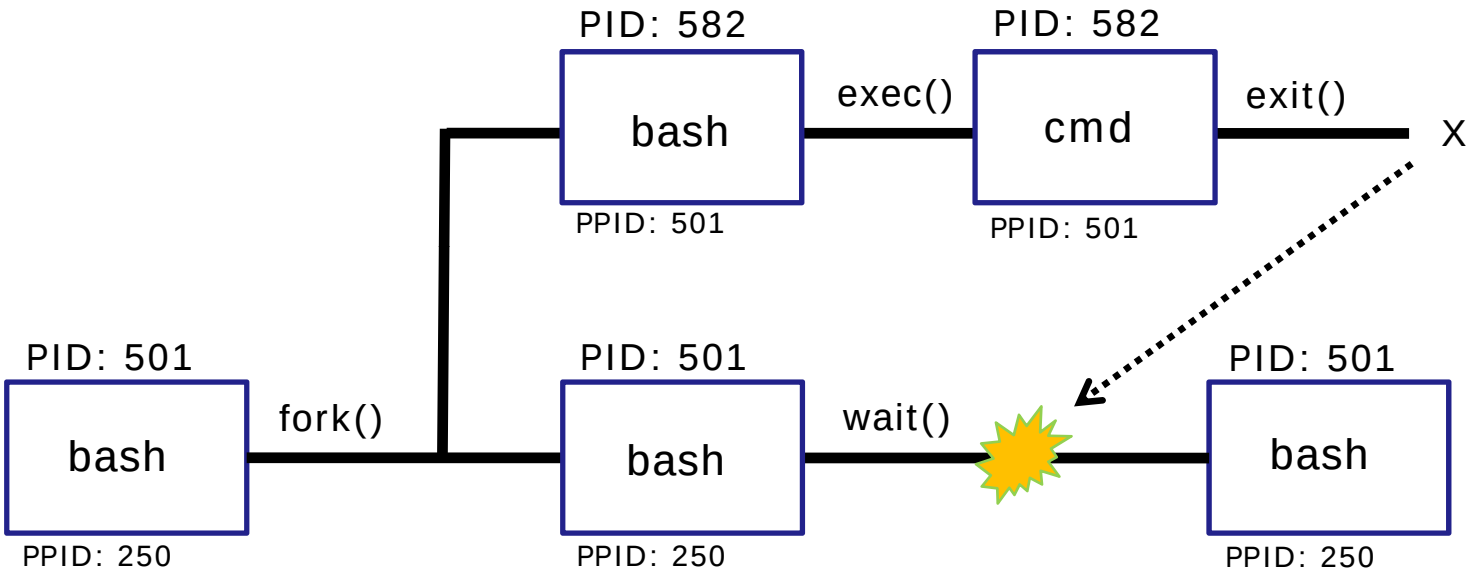
In this example, the commands in the file shscript are run by the parent shell, and therefore, any changes made to the environment will last for the duration of the login session.

If a UNIX command is run using the exec command, the shell will terminate upon the exiting of that command:

exec clear

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

Children can not change the parent's variables



When a shell forks a child, not all of the variables get passed on to the child. Only those the environment variables (which have been exported) are passed on to the child.

- The child gets a copy of the parents environment variables
- Changes made to the copies do not change the parent's variables

. and exec

```
/home/cis90/roddyduk $ cat program
echo "program is being run"
fan=high
ac=on
export ac
alias copy=cp
```

A sample script to create some variables and an alias. Note only one variable is exported.

```
/home/cis90/roddyduk $ echo fan=$fan ac=$ac; type copy;  env | grep ac
fan= ac=
-bash: type: copy: not found
```

Initial state

```
/home/cis90/roddyduk $ program
program is being run
/home/cis90/roddyduk $ echo fan=$fan ac=$ac; type copy; env | grep ac
fan= ac=
-bash: type: copy: not found
```

Not changed!

```
/home/cis90/roddyduk $ source program
program is being run
/home/cis90/roddyduk $ echo fan=$fan ac=$ac; type copy; env | grep ac
fan=high ac=on
copy is aliased to `cp'
ac=on
```

Changed when sourced!

A child cannot make changes to the parent, use source or . when you need a script to make changes.

. and exec

parent

```
/home/cis90/roddyduk $ bash
```

Start a sub-shell child precess

child

```
[roddyduk@opus ~]$ echo fan=$fan ac=$ac; type copy; env | grep ac
fan= ac=on
bash: type: copy: not found
ac=on

[roddyduk@opus ~]$ . program
program is being run
[roddyduk@opus ~]$ echo fan=$fan ac=$ac; type copy; env | grep ac
fan=high ac=on
copy is aliased to `cp'
ac=on

[roddyduk@opus ~]$ exec program
program is being run
```

parent

```
/home/cis90/roddyduk $
```

Only the exported variables exist for the child

. can be used for the source command

exec replaces bash code with program script. When finished the child is killed

We are back in the parent shell because we used exec. If we had not been a child process our session would have abruptly ended!

print command (alias)

What is going on ??????????????

Make a print alias for lp, then try it in a sub-shell (child process) and the behavior completely changes!

```
/home/cis90/roddyduk $ alias print=lp
/home/cis90/roddyduk $ print lab10
request id is hplaser-9 (1 file(s))
/home/cis90/roddyduk $ bash
```

child

```
[roddyduk@opus ~]$ ls lab10
lab10
[roddyduk@opus ~]$ print lab10
lab10
[roddyduk@opus ~]$ print A B C $LOGNAME
A B C roddyduk
```

*The **lp** command is used to print files on a printer*

*Huh? Why is print now behaving as if it were the **echo** command instead of the **lp** command*

What is going on ??????????????

child

```
[roddyduk@opus ~]$ type print
print is aliased to `echo -e'
[roddyduk@opus ~]$ alias print
alias print='echo -e'
[roddyduk@opus ~]$ cat .bashrc
# .bashrc

# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
alias print="echo -e"
alias bye="clear;exit"
alias rm="rm -i"
alias bill="cd /home/cis90/$LOGNAME/poems/Shakespeare"
[roddyduk@opus ~]$
```

*Our print alias was changed! It is no longer aliased to the **lp** command*

.bashrc is sourced when starting a new sub-shell and this reset the alias!

Moral of the story is ...

child

```
[roddyduk@opus ~]$ exit
exit
/home/cis90/roddyduk $ type print
print is aliased to `lp'
/home/cis90/roddyduk $ print lab10
request id is hplaser-10 (1 file(s))
/home/cis90/roddyduk $
```

When we exit the sub-shell our new print alias is back in action

Moral of the story is, aliases do not get exported like environment variables. If you want an alias to be available in a child process you must add it to .bashrc

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)*
- *PictBridge (USB based)*

Back then:
- *Serial cable*
- *Parallel printer cable*

Printer Configuration

Printing Commands

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

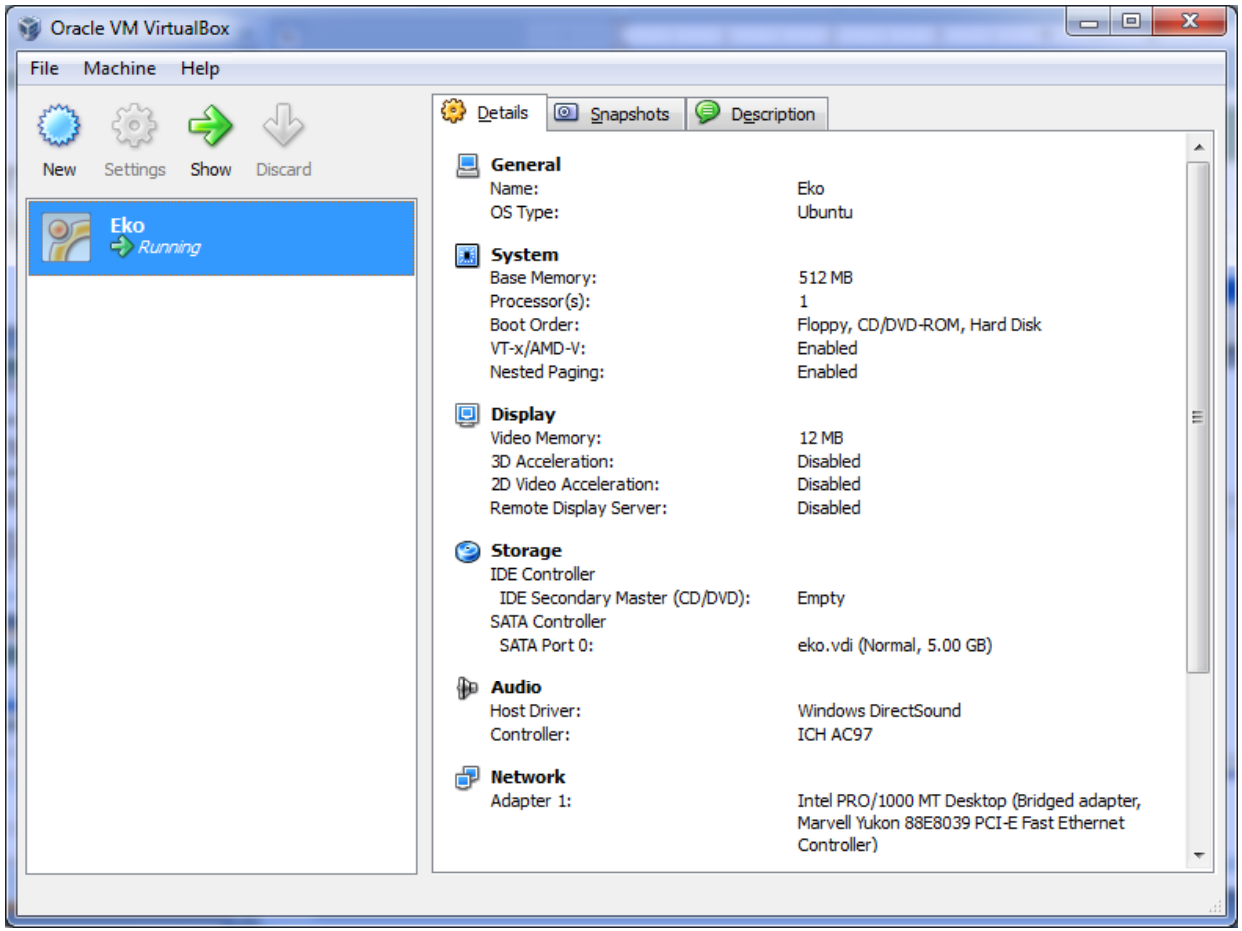
BSD based print subsystem
- lpr (to print)
- lpq (queue management)
- lprm (to remove jobs)

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

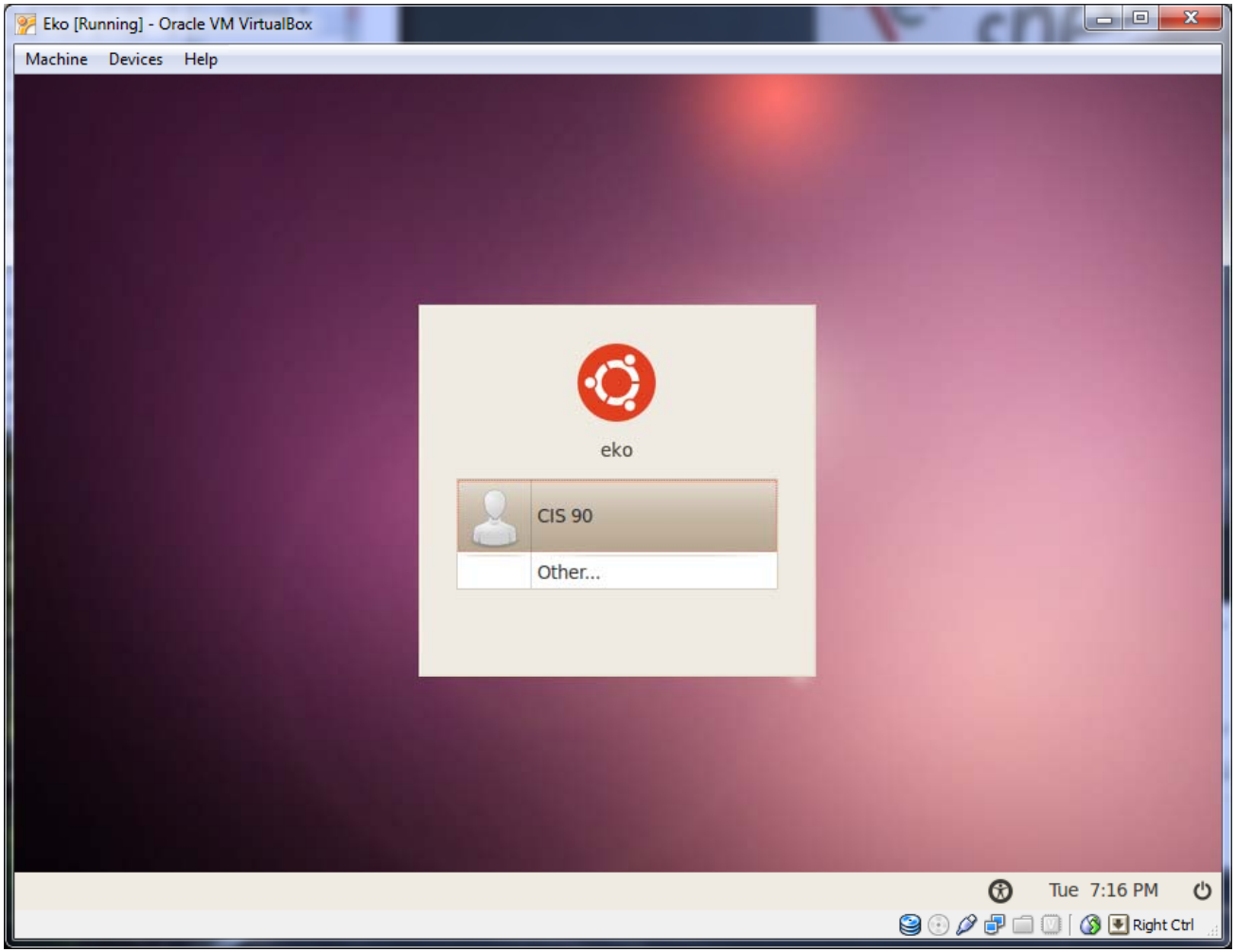
We will be just looking at CUPS

CUPS

For the lesson on printing we will be using the Eko virtual machine.

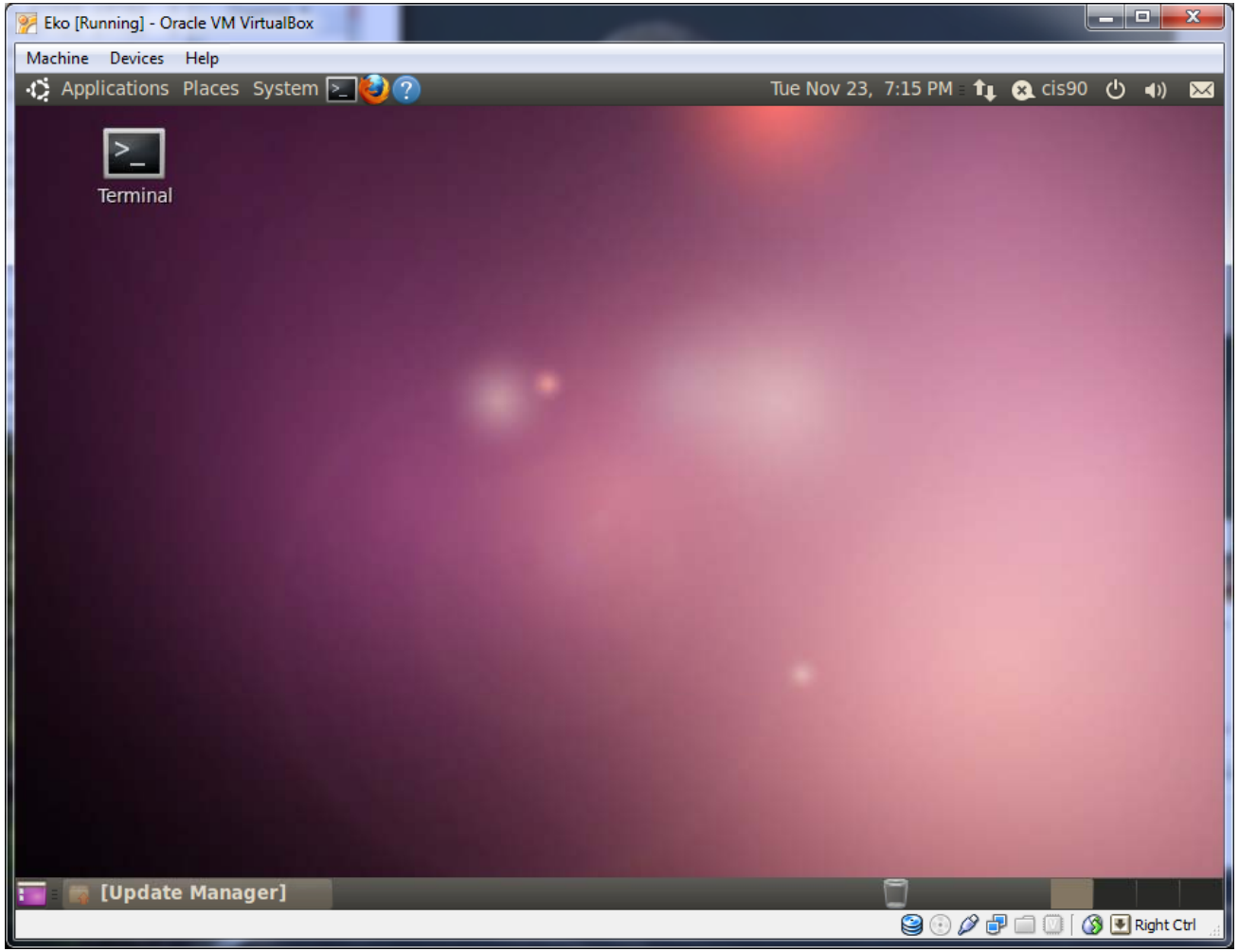


CUPS



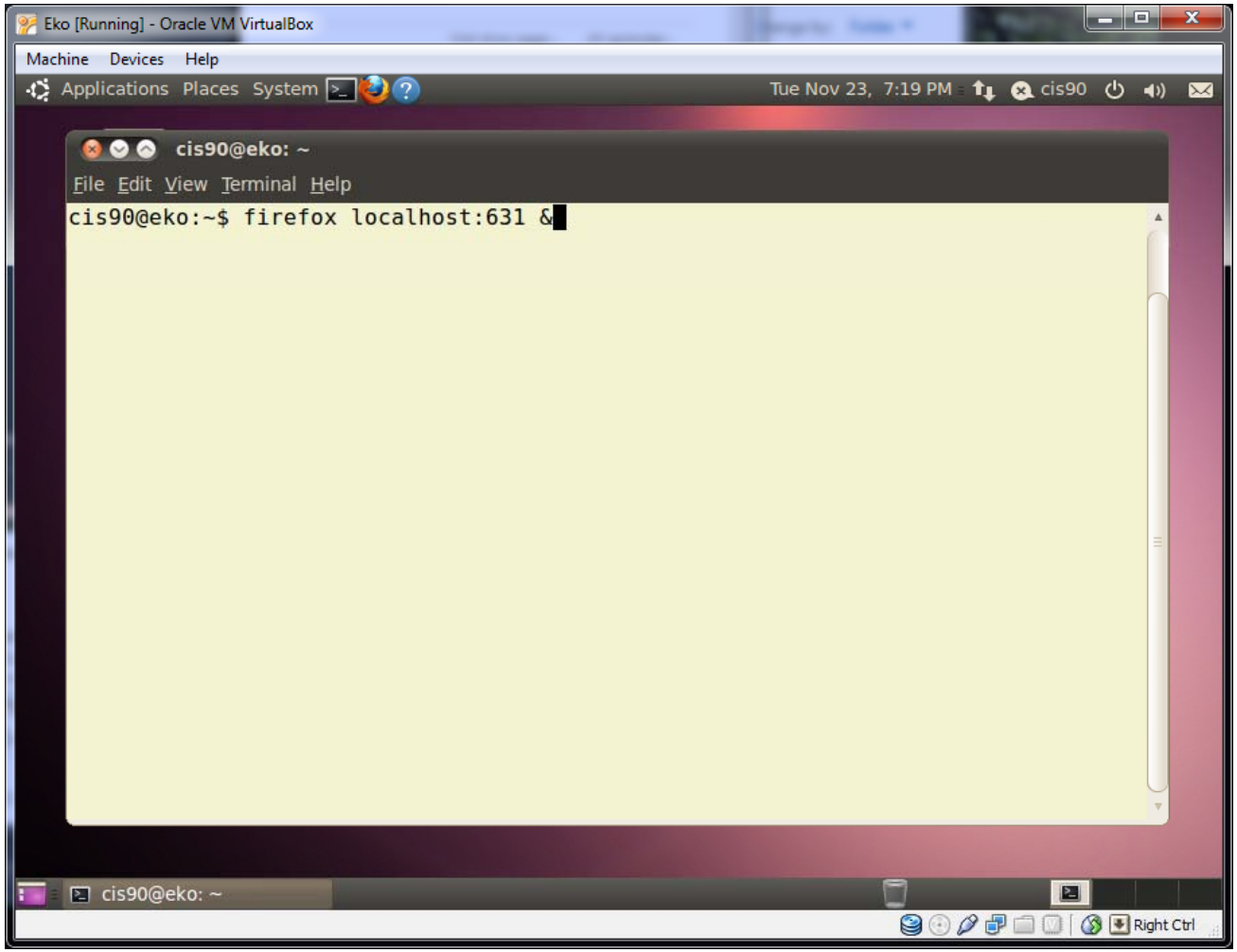
Login as cis90

CUPS



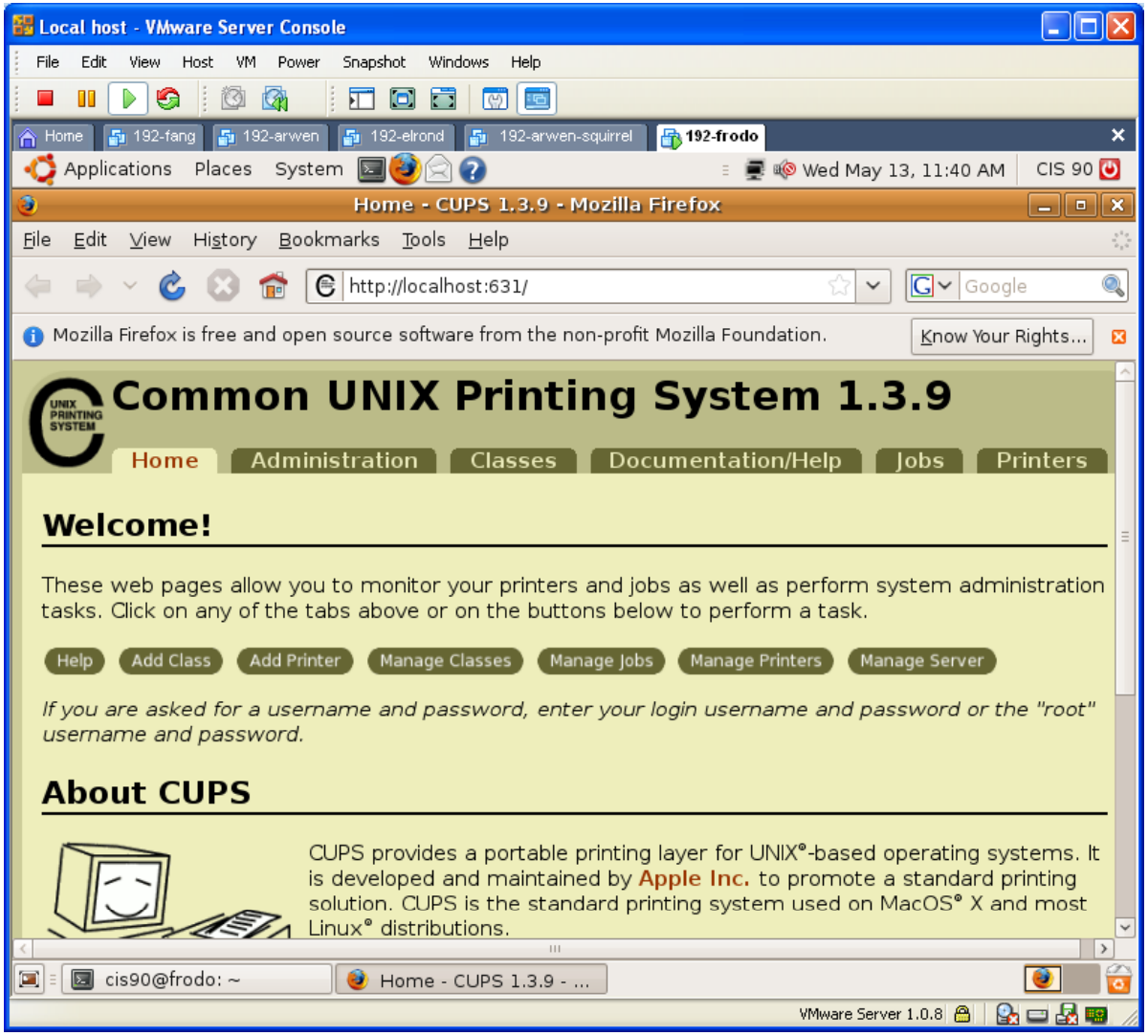
Open the Eko Terminal icon

CUPS



*Type the **firefox** command with **localhost:631** as the argument in the background with the **&***

CUPS



CUPS is managed by a web-based configuration utility on port 631

Local access only by default

CUPS

Next step is to add printers

Printer: HP LaserJet 1320n
Connection: LAN

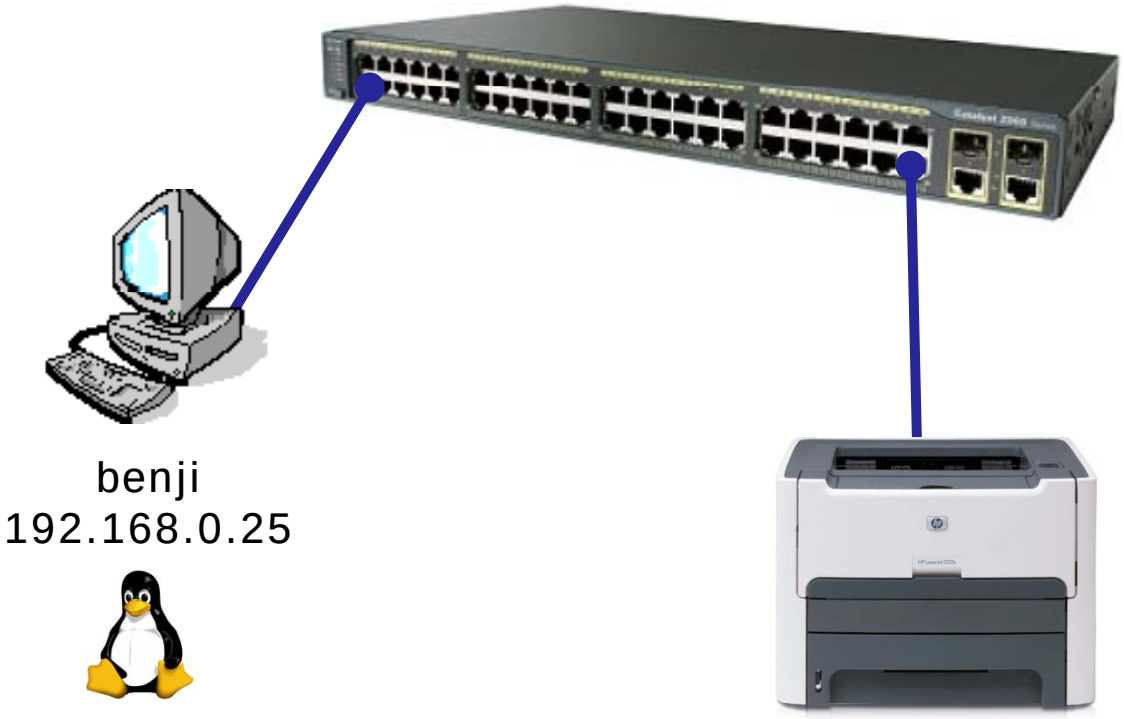
CUPS

The LaserJets also have a web-based management utility

IP Address for this 1320n is 192.168.0.12

CUPS

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



benji
192.168.0.25

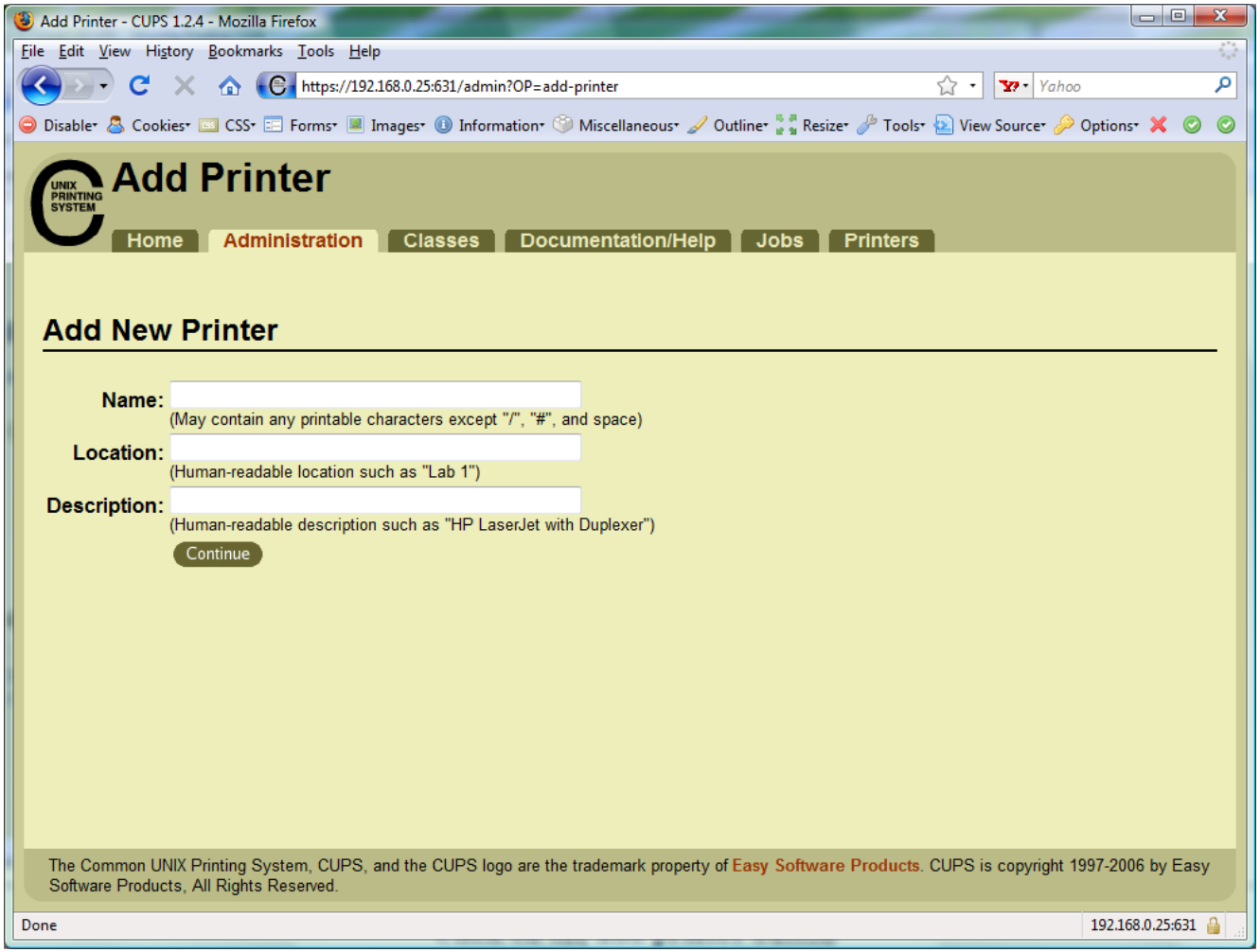
hp1320n
192.168.0.12

CUPS

To add in HP 1320N printer ...

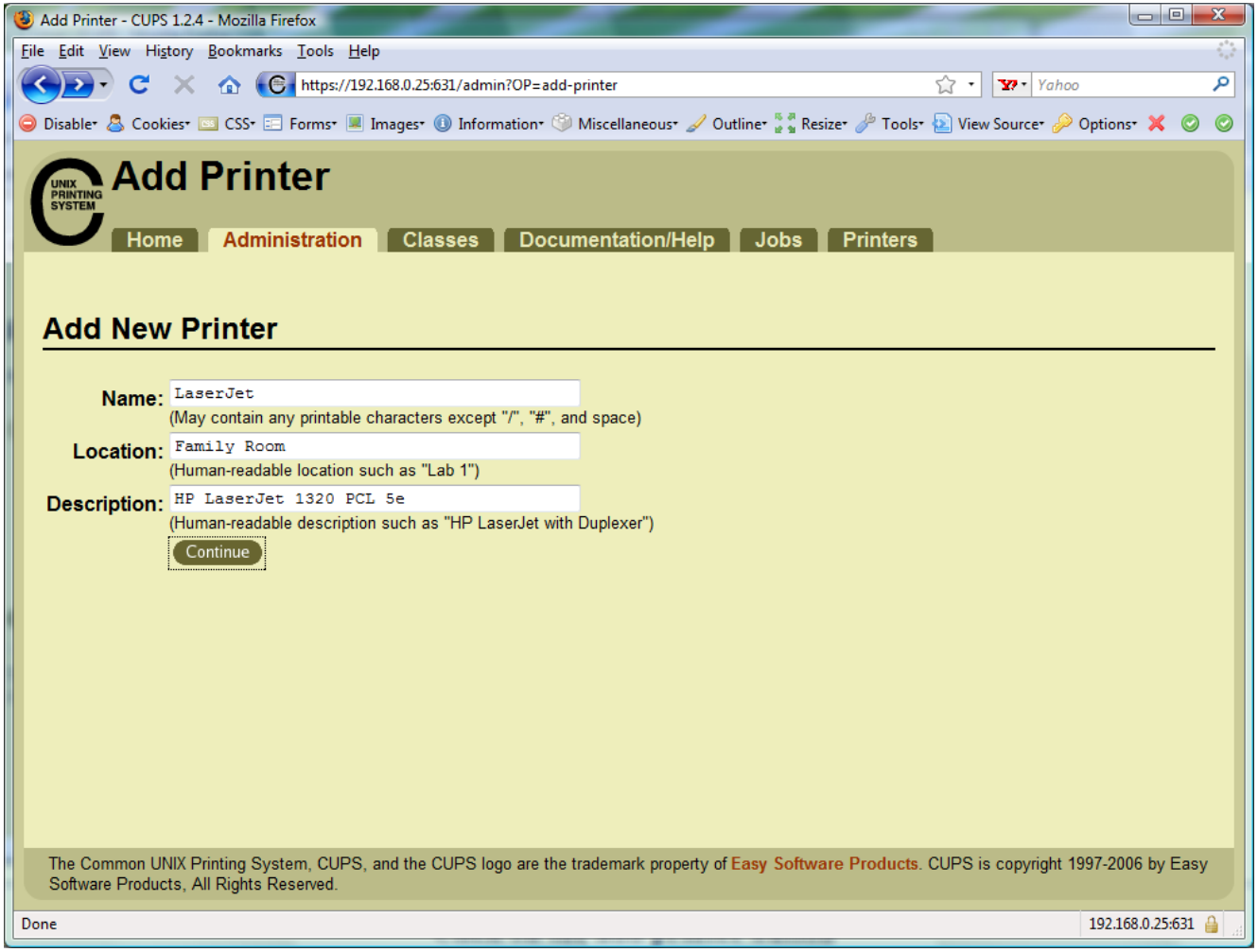
... the first step is to click the Add Printer button

CUPS



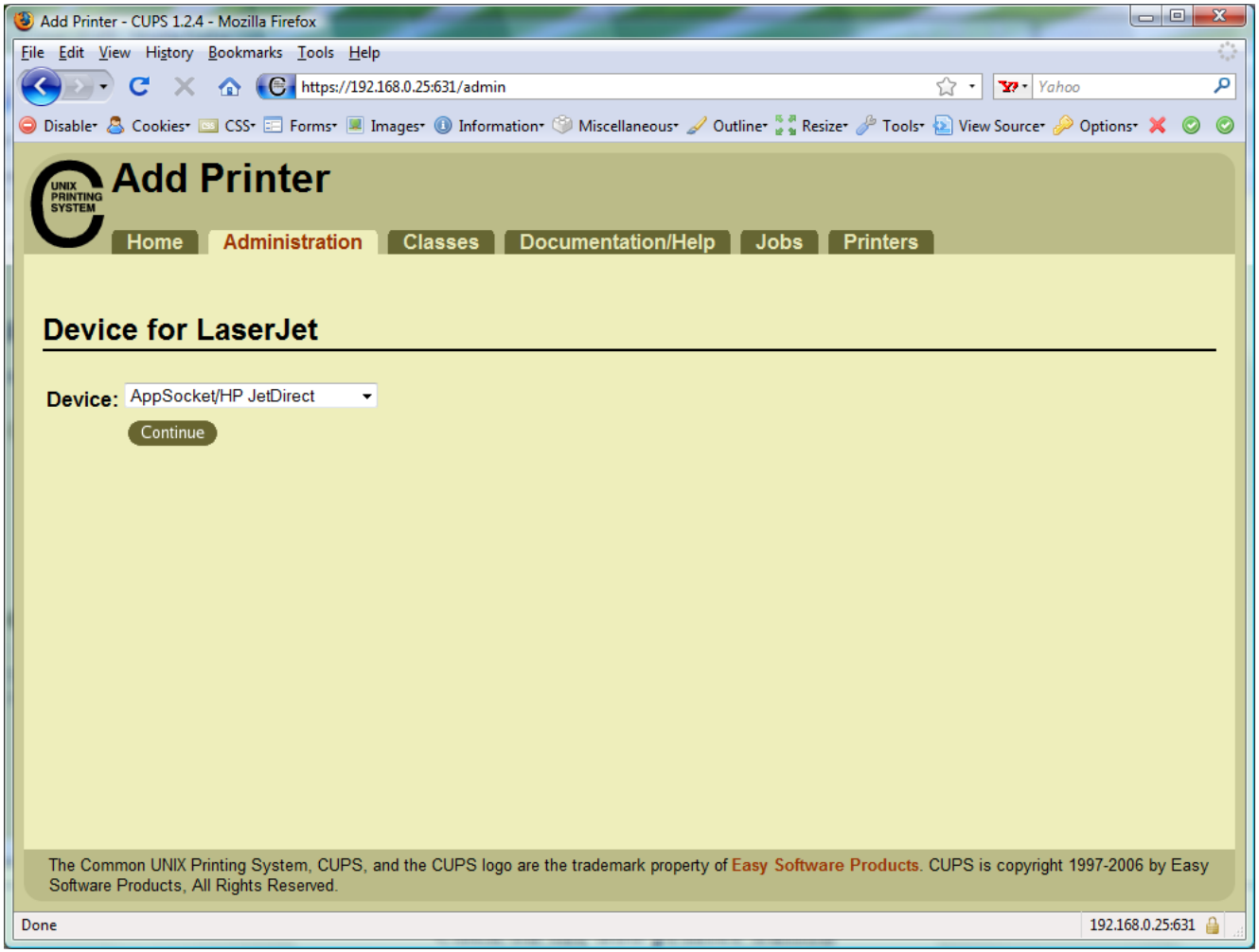
Now we can add the LaserJet

CUPS



Fill in basic information

CUPS

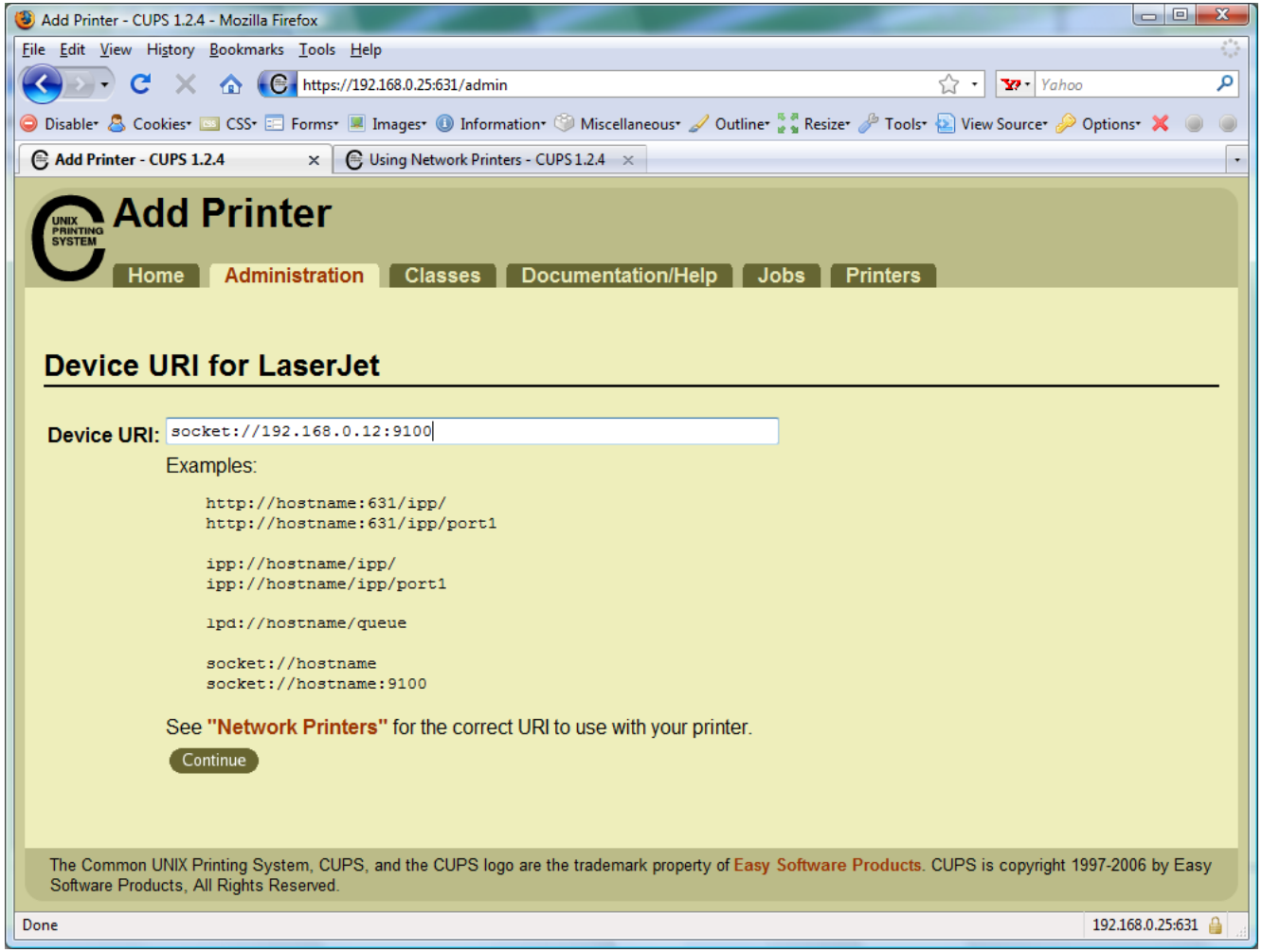


We will use JetDirect.

JetDirect is a small printer server built into some of HP's printers.

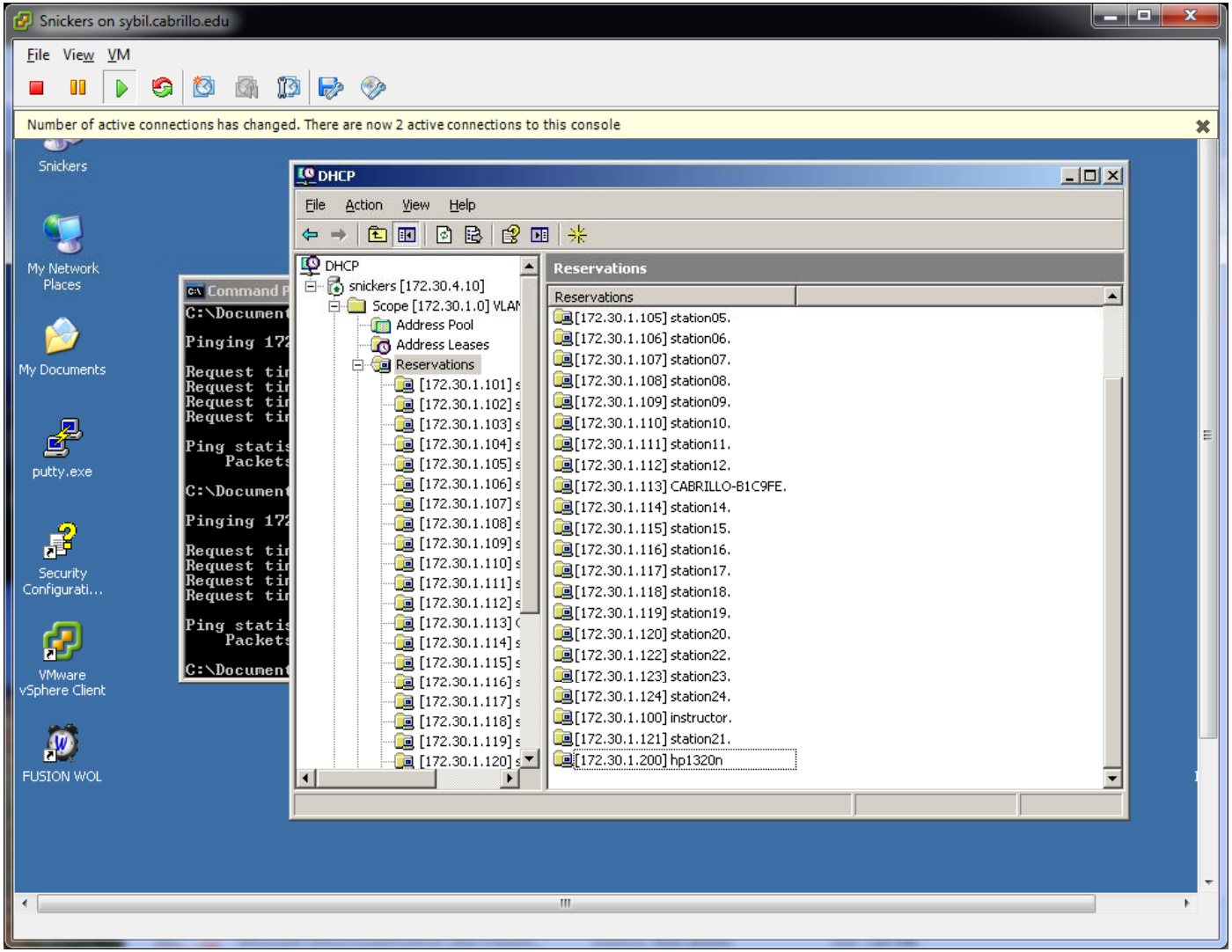
CUPS

socket://192.168.0.12:9100



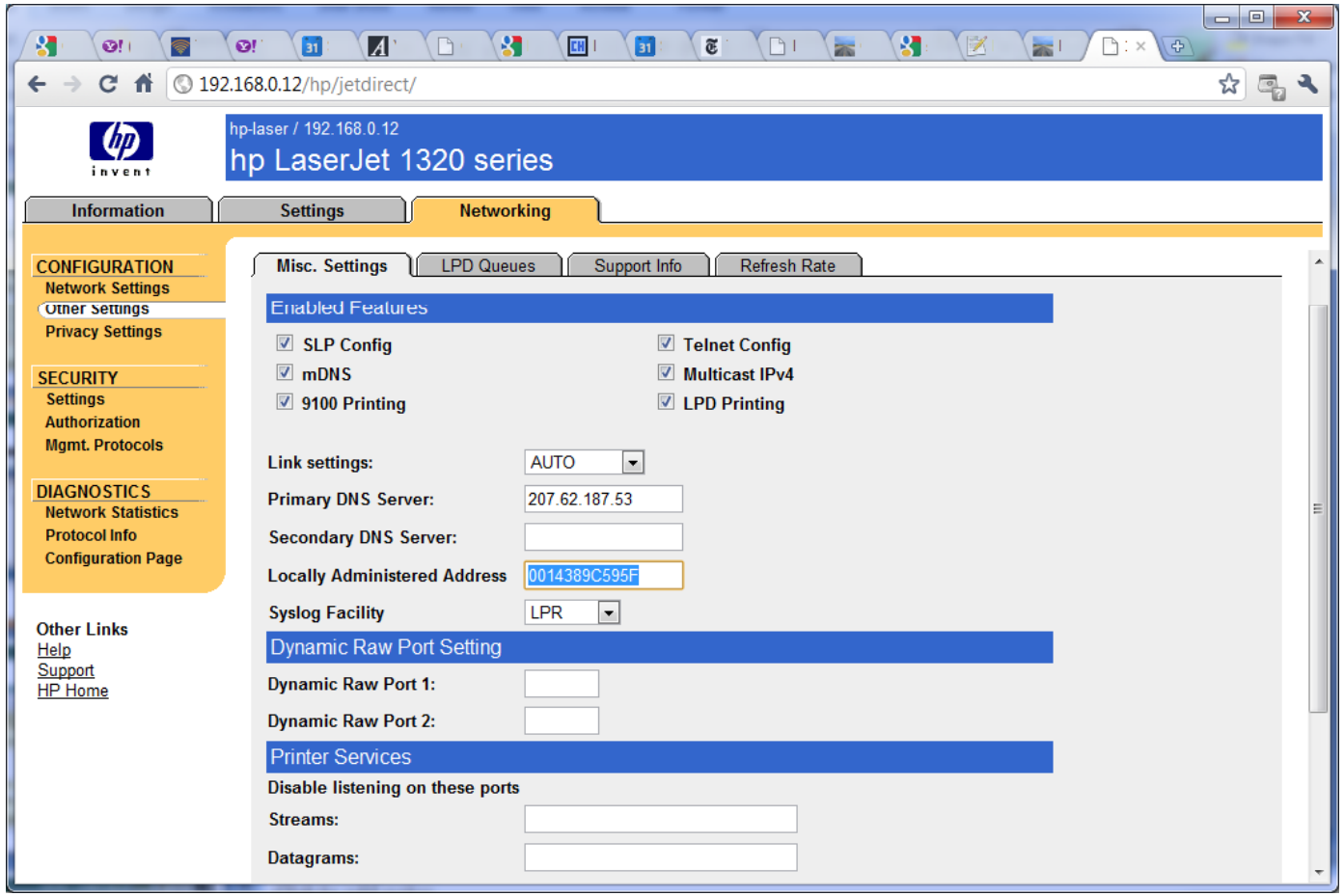
This defines how to communicate with the printer

CUPS



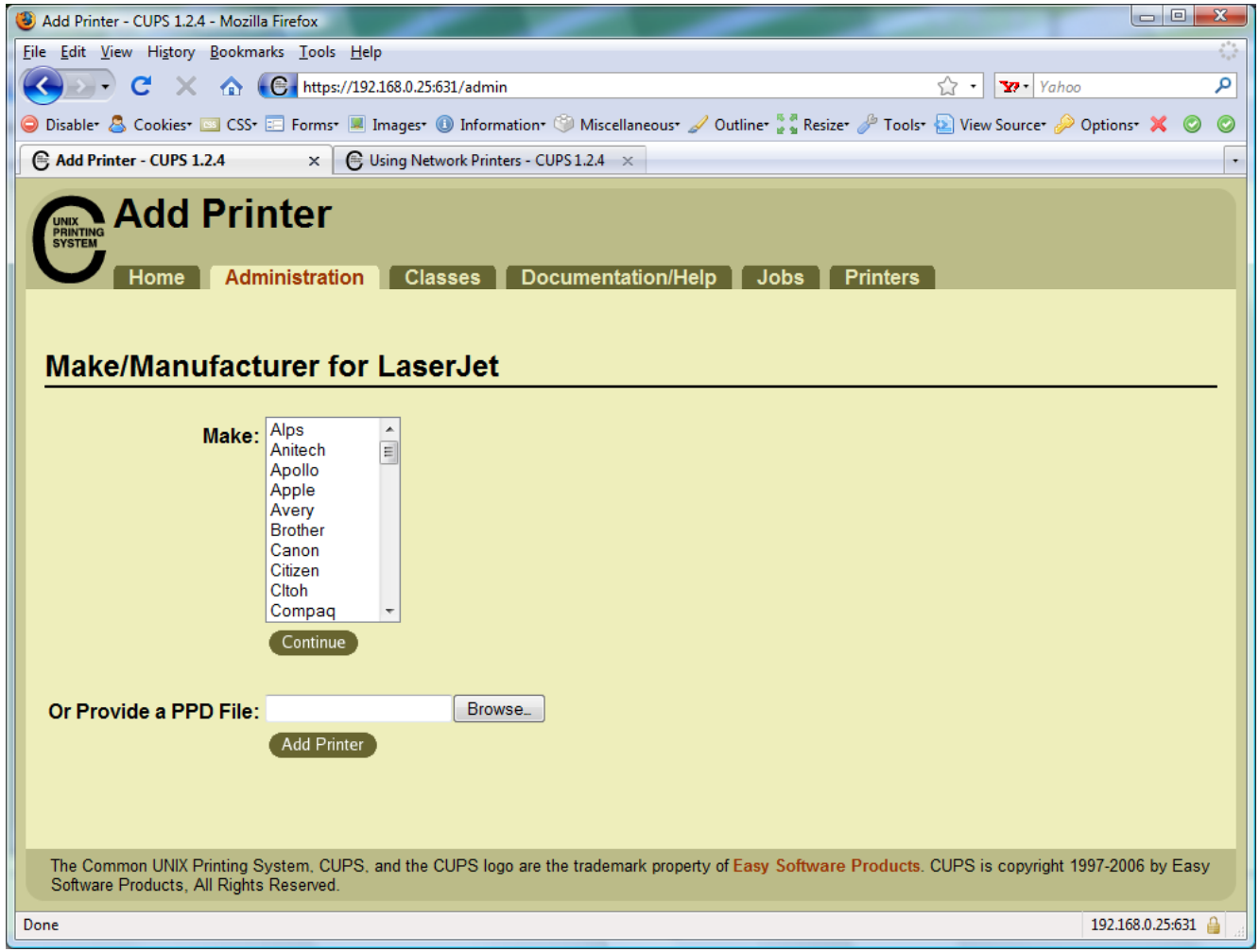
Room 2501: 172.30.1.200 for 0014389C595F

CUPS



Room 2501: 172.30.1.200 for 0014389C595F

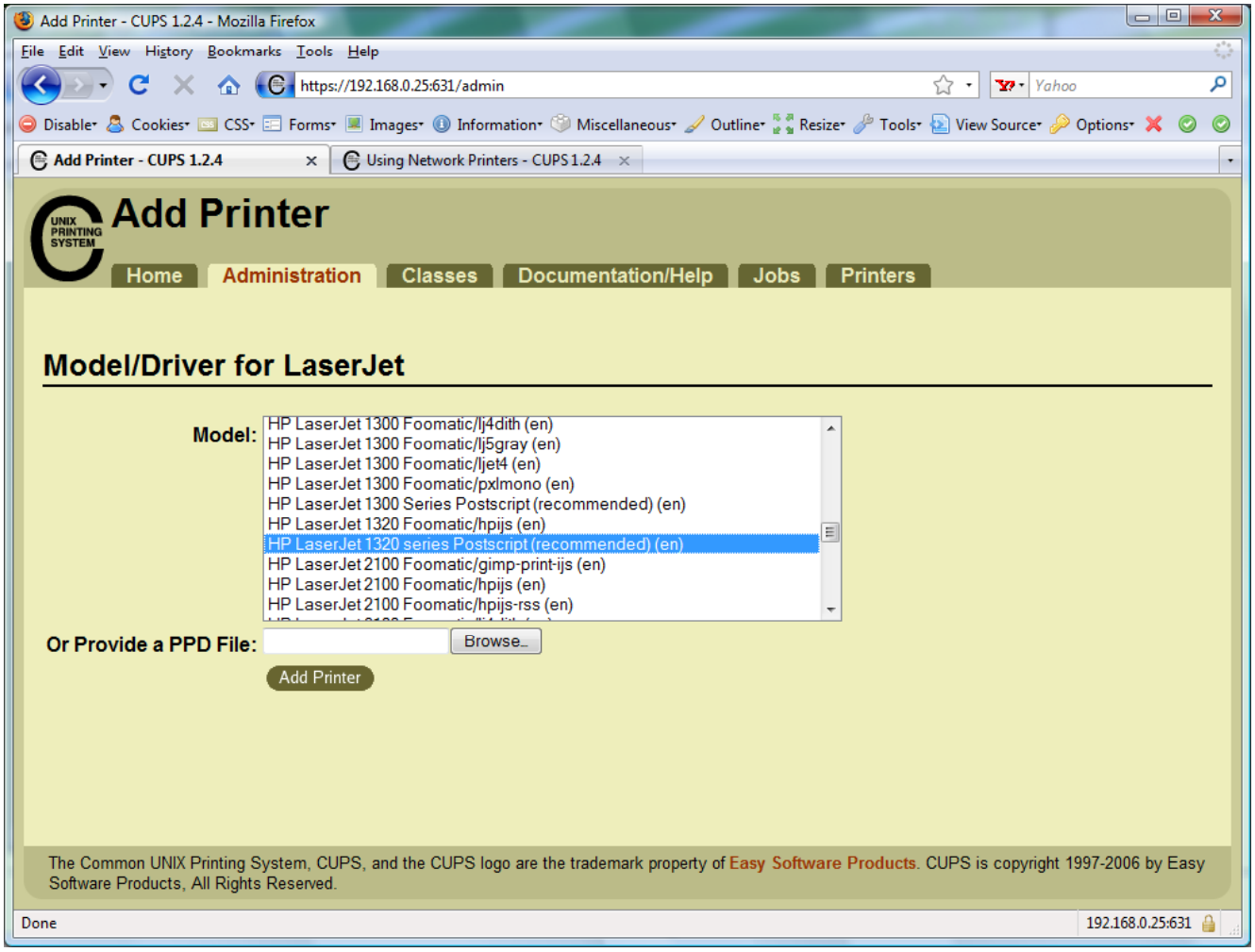
CUPS



(you will need to enter root's password)

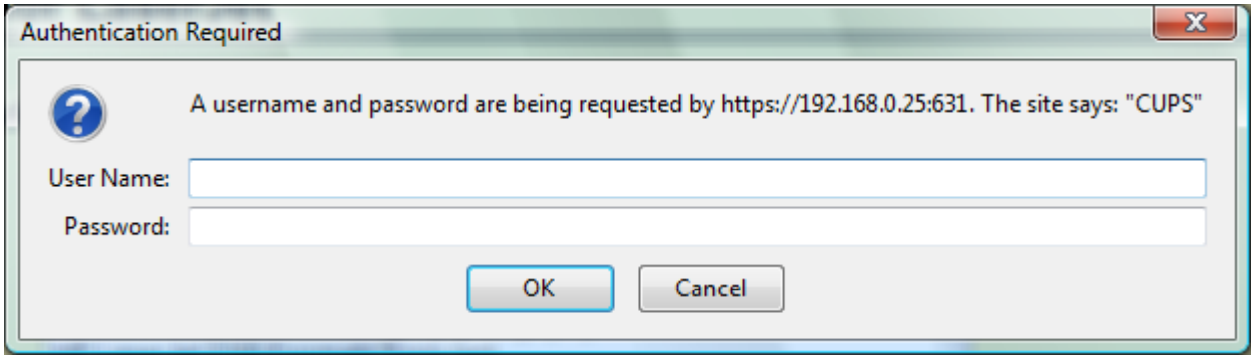
Service will restart

CUPS



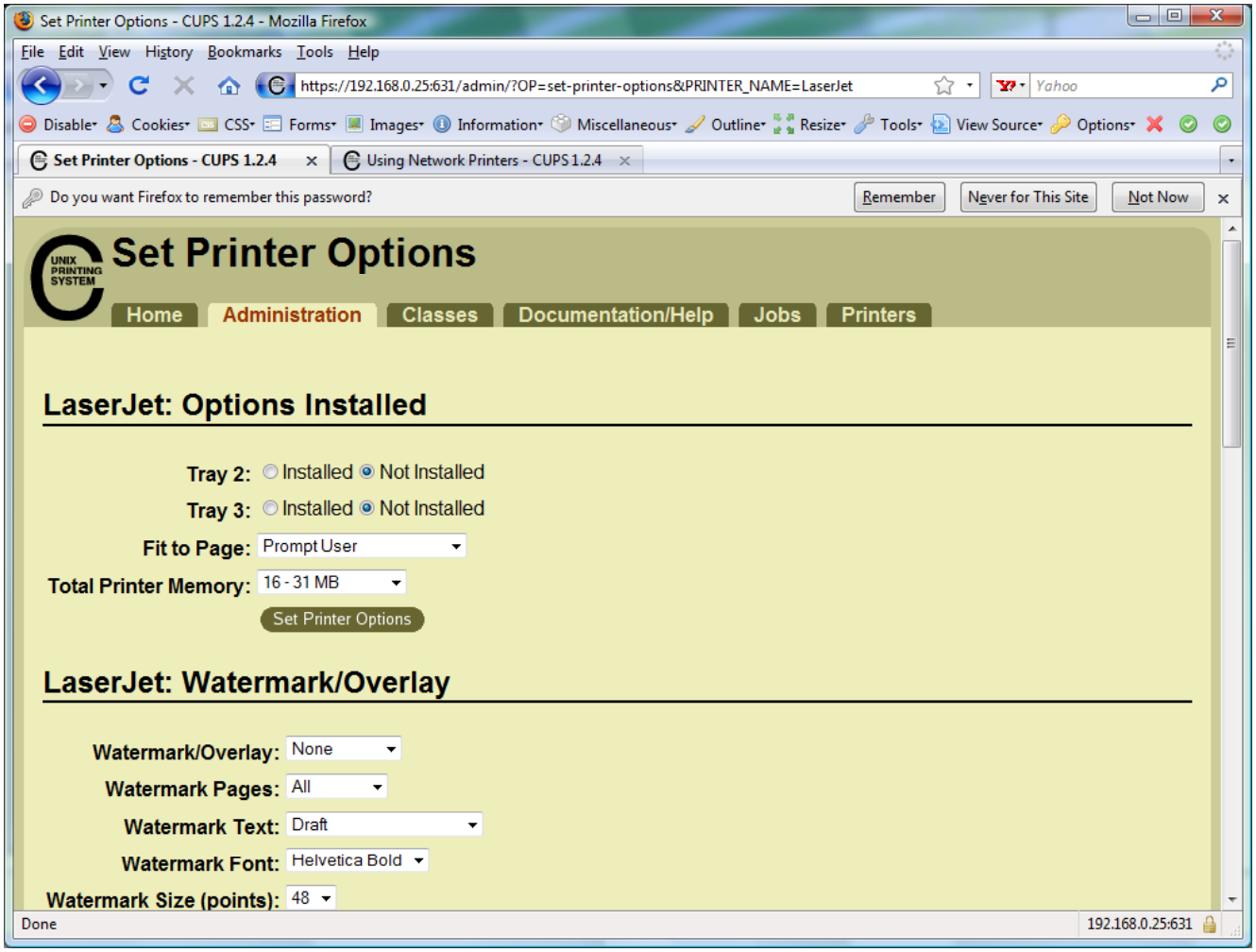
We will choose hp LaserJet 1320 series Postscript (recommended) (en)

CUPS



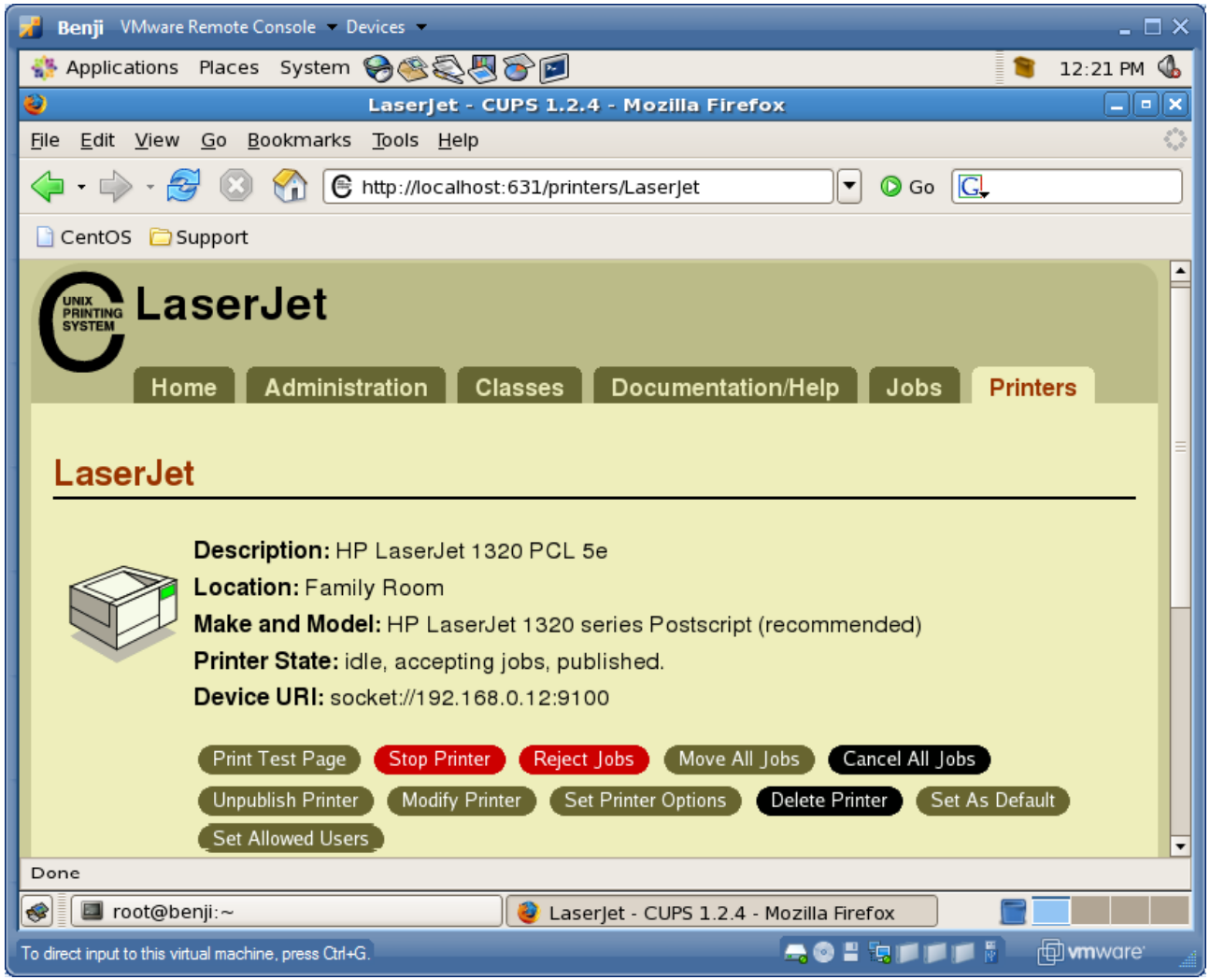
To finally add the printer it will be necessary authenticate as root

CUPS



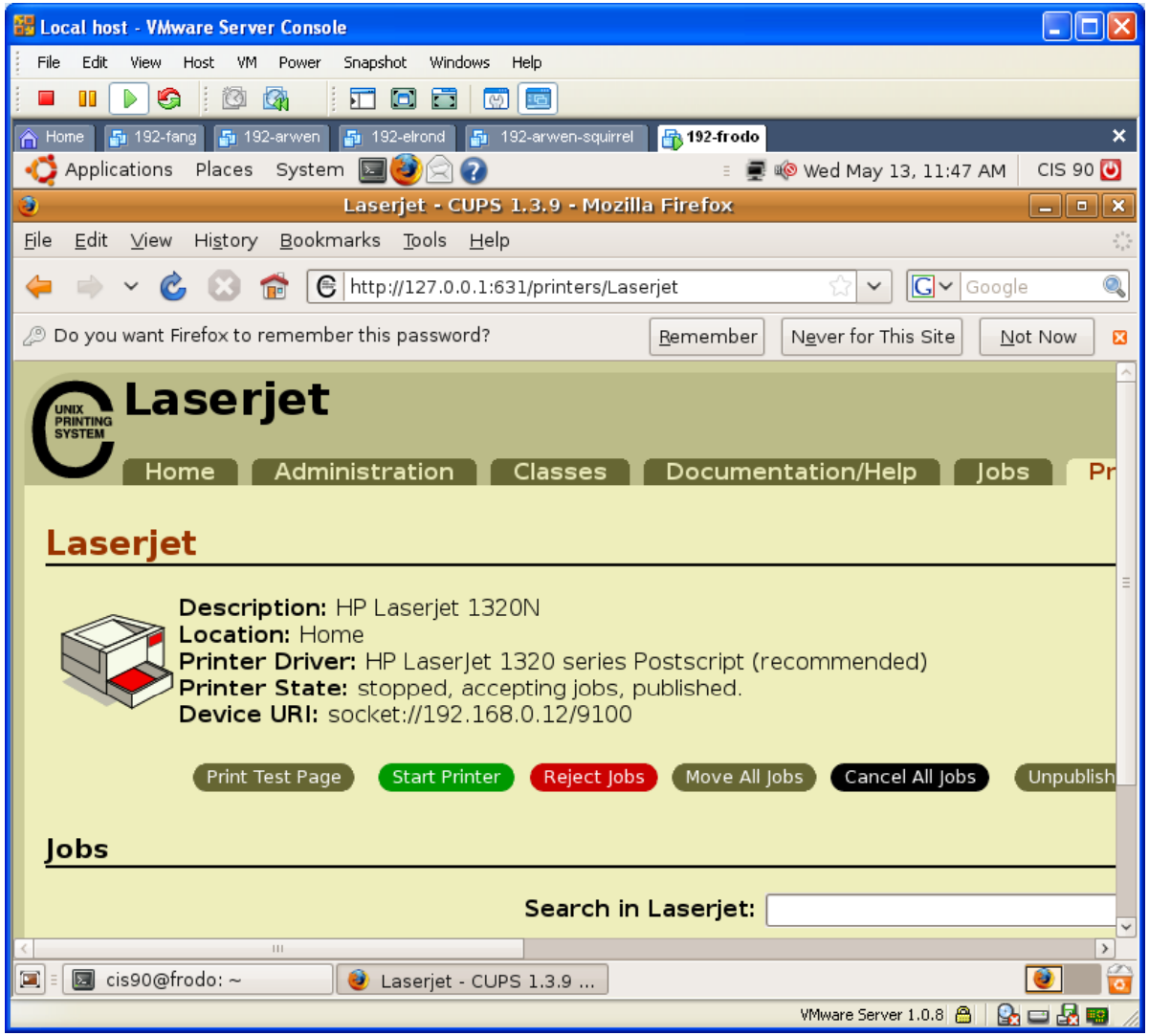
Printer has been added

CUPS



View of newly added printer from Printer tab

CUPS



Configure the printer so it is stopped but still accepts print jobs

CUPS

Lets add second printer

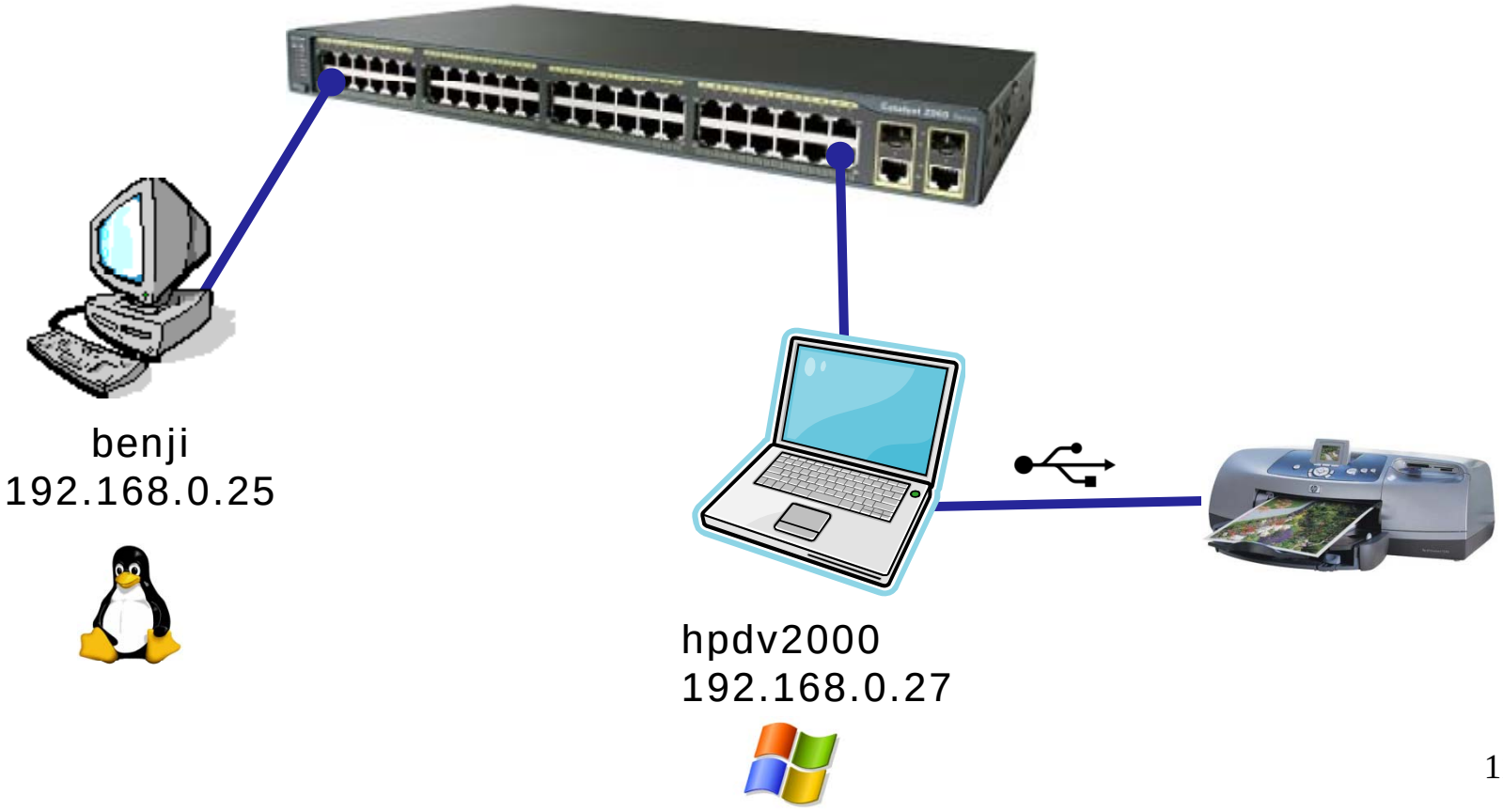
Printer: hp photosmart 7550 (color inkjet technology)
Connection: USB

Sidetrack – The previous 7550 "Hot Lips"

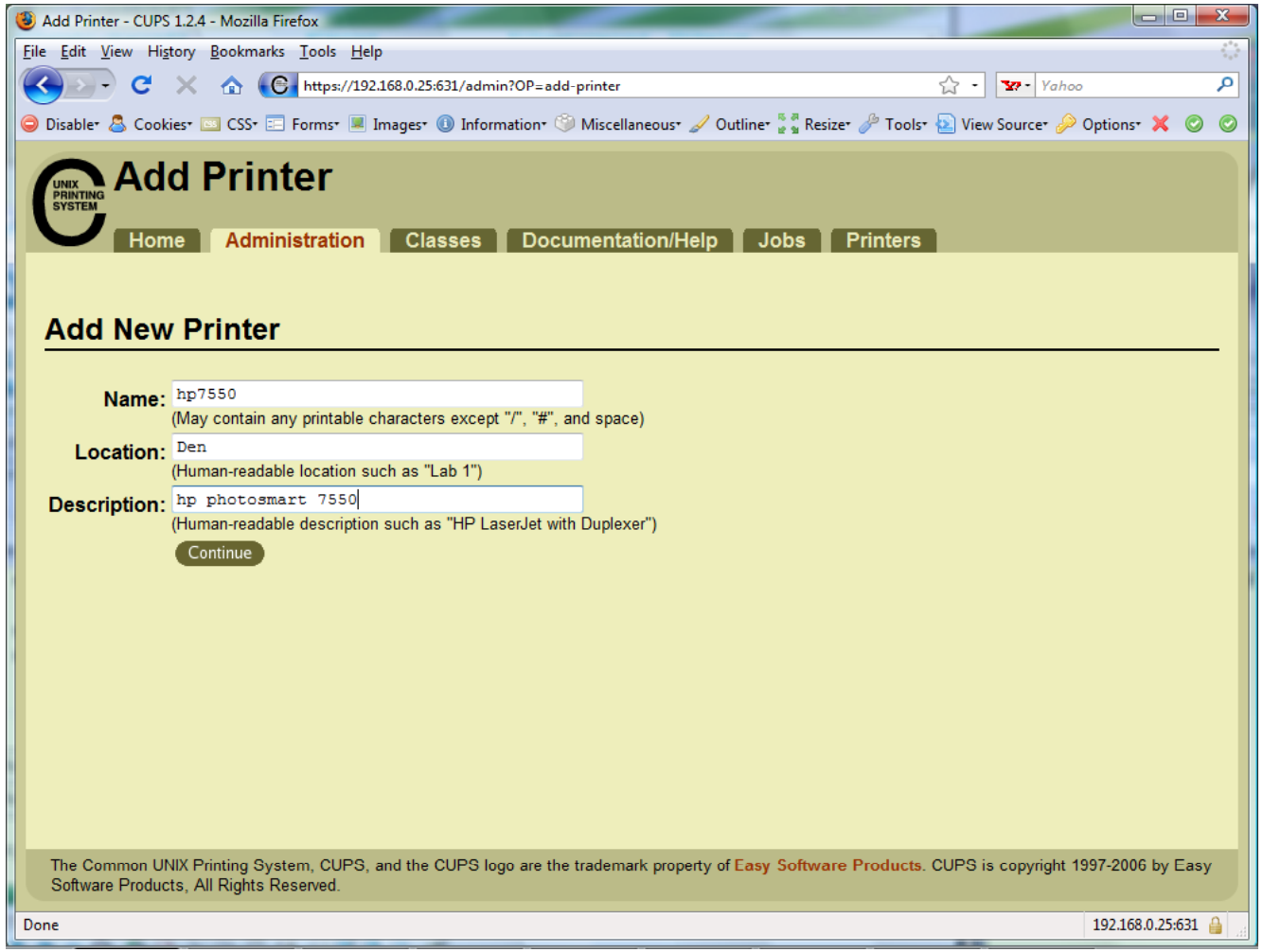
6 G's of acceleration
8-pen turret
Grit wheel technology from HP Labs

CUPS

The second printer is connected by USB to a Windows notebook computer

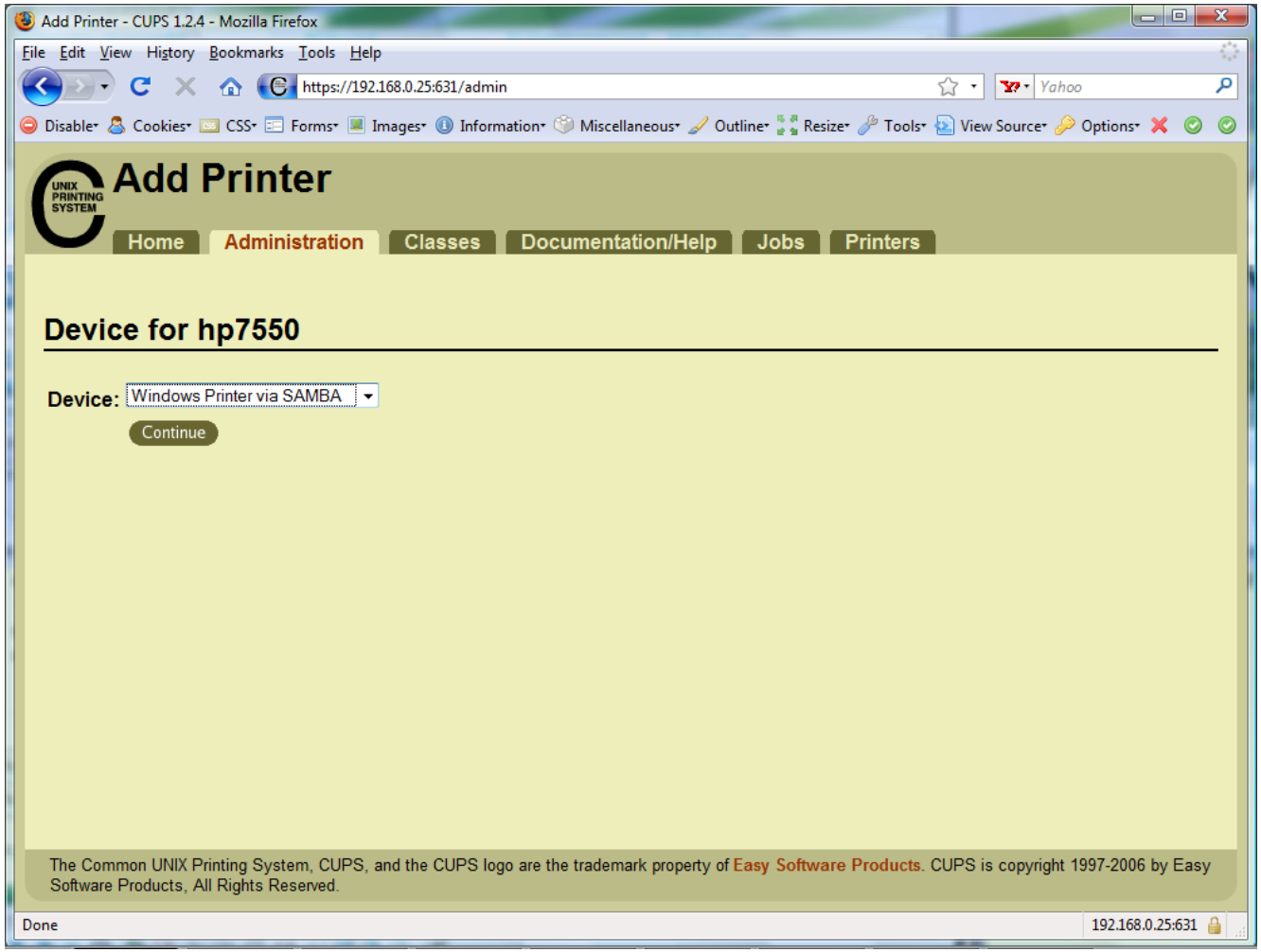


CUPS



First step is the same which is to fill out basic information on printer

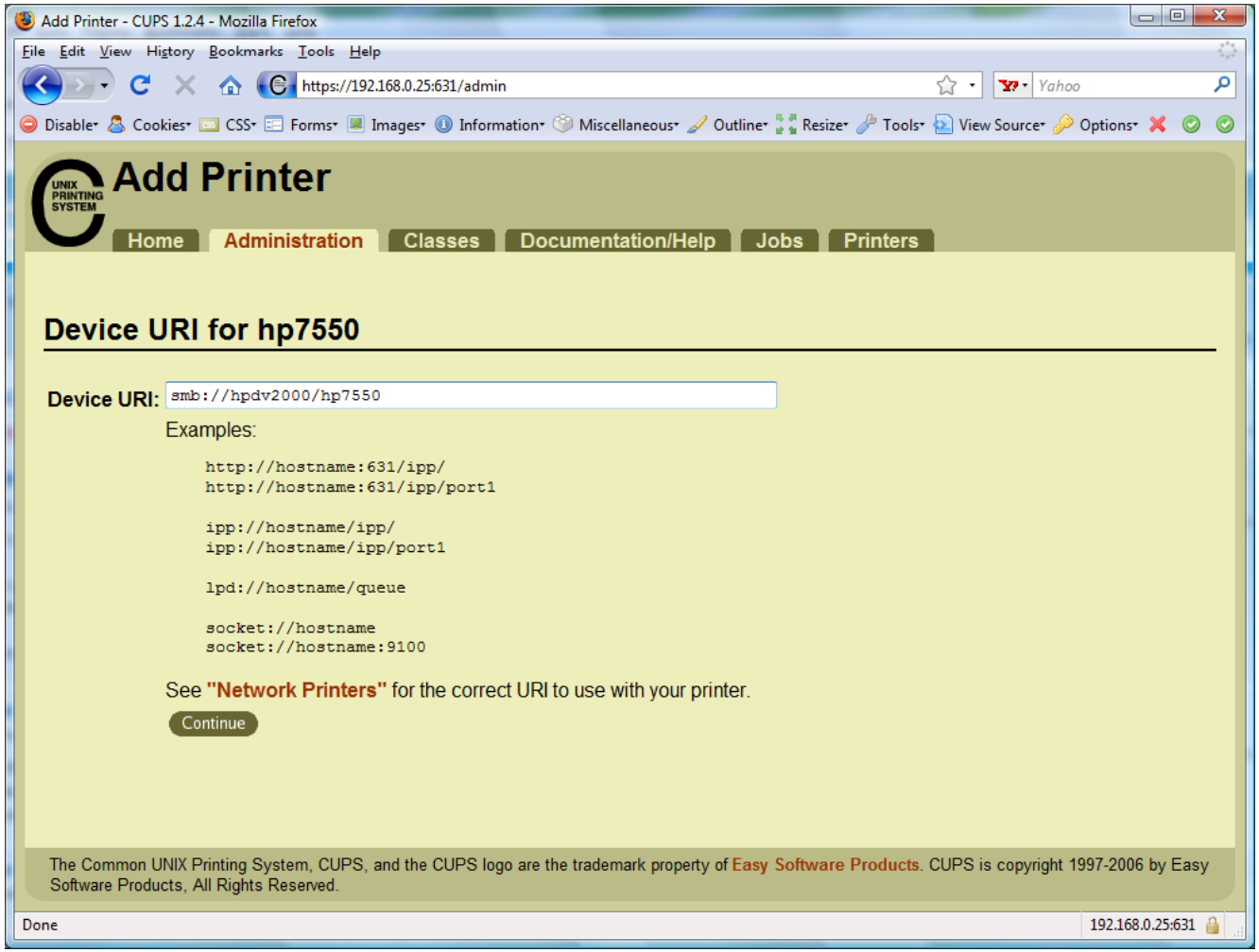
CUPS



For this connection we will use Samba. Samba implements Windows file and print services sharing on Linux.

Note Windows uses SMB (Server Message Block) protocol to implement these services

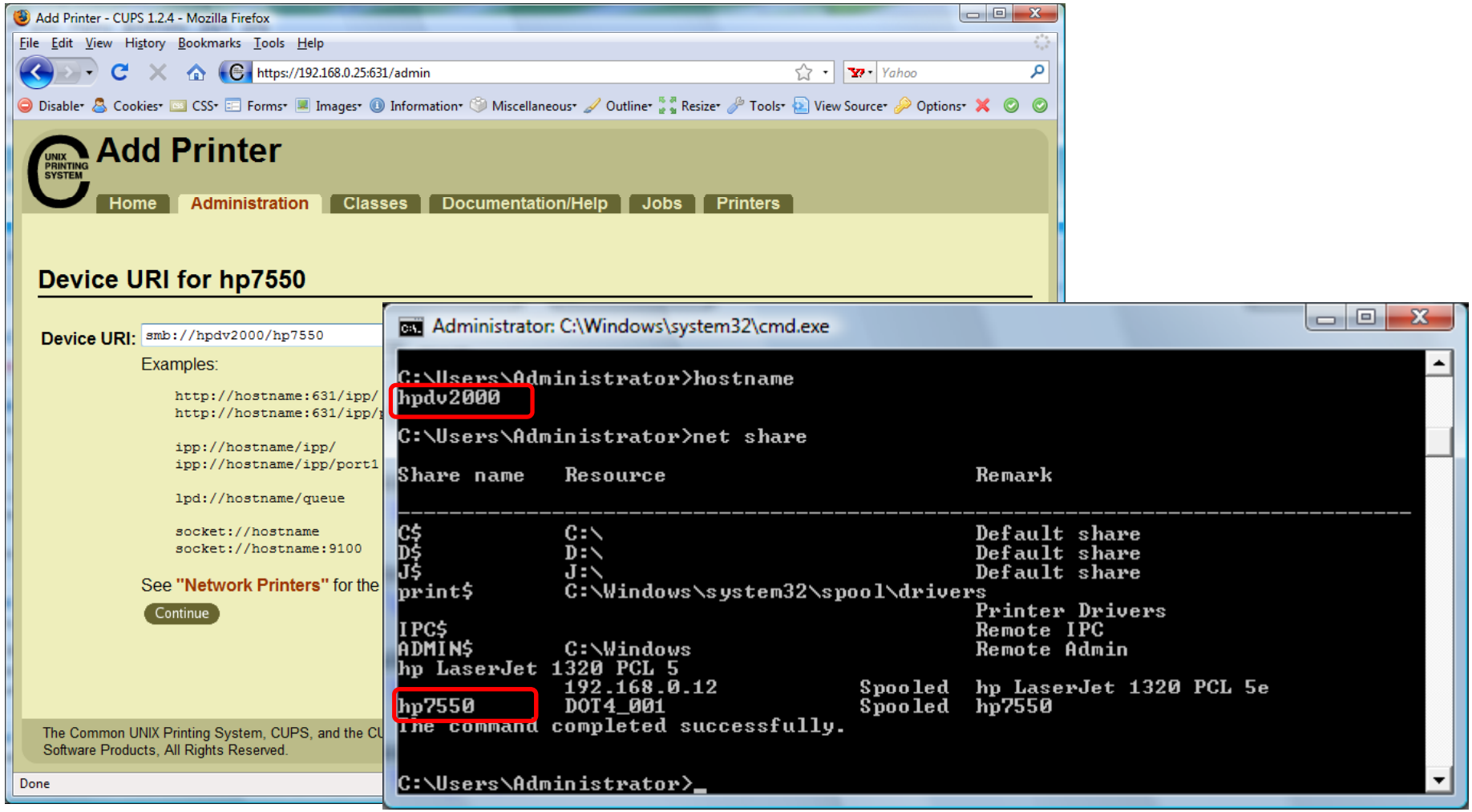
CUPS



Will need to specify the Windows print share

CUPS

Will need to specify the Windows print share as //hostname/printsharename



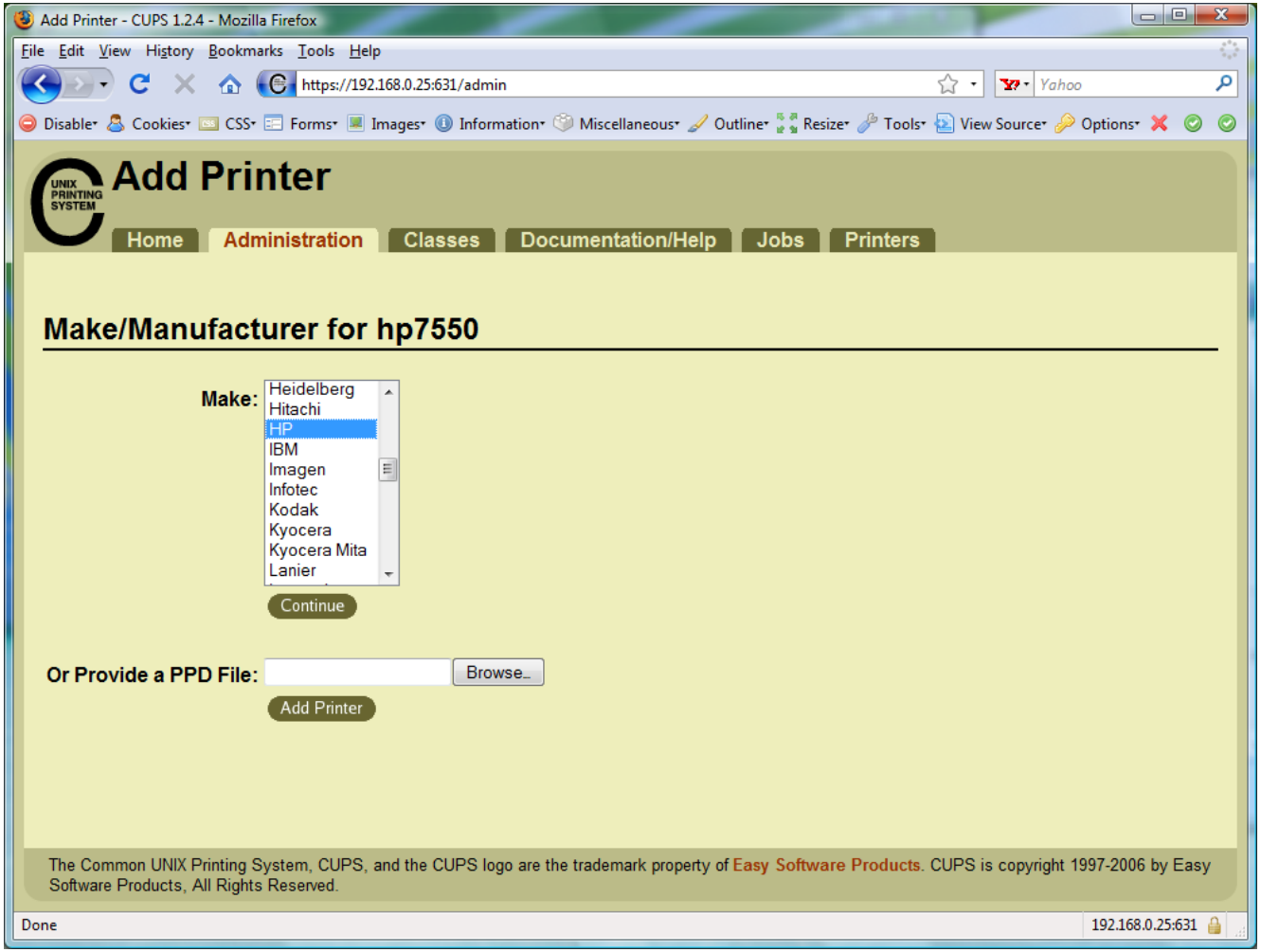
CUPS

Ways to specify a Windows share

	Username and password Not required
This machine is in the same workgroup	`smb://server/sharename`
This machine is in a different workgroup	`smb://workgroup/server/sharename`

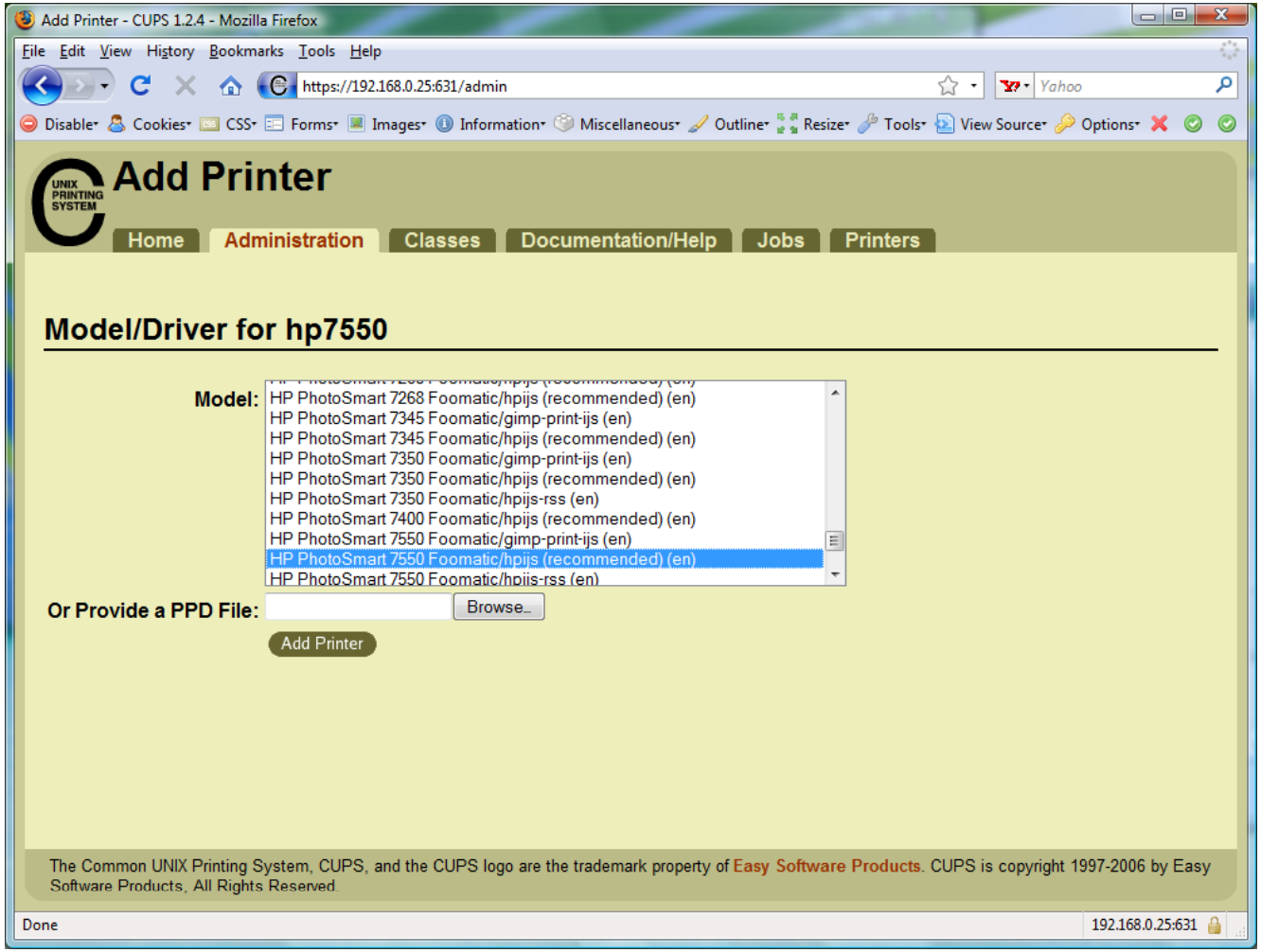
	Username and password required
This machine is in the same workgroup	`smb://username:password@server/sharename`
This machine is in a different workgroup	`smb://username:password@workgroup/server/sharename`

CUPS



Select make of printer

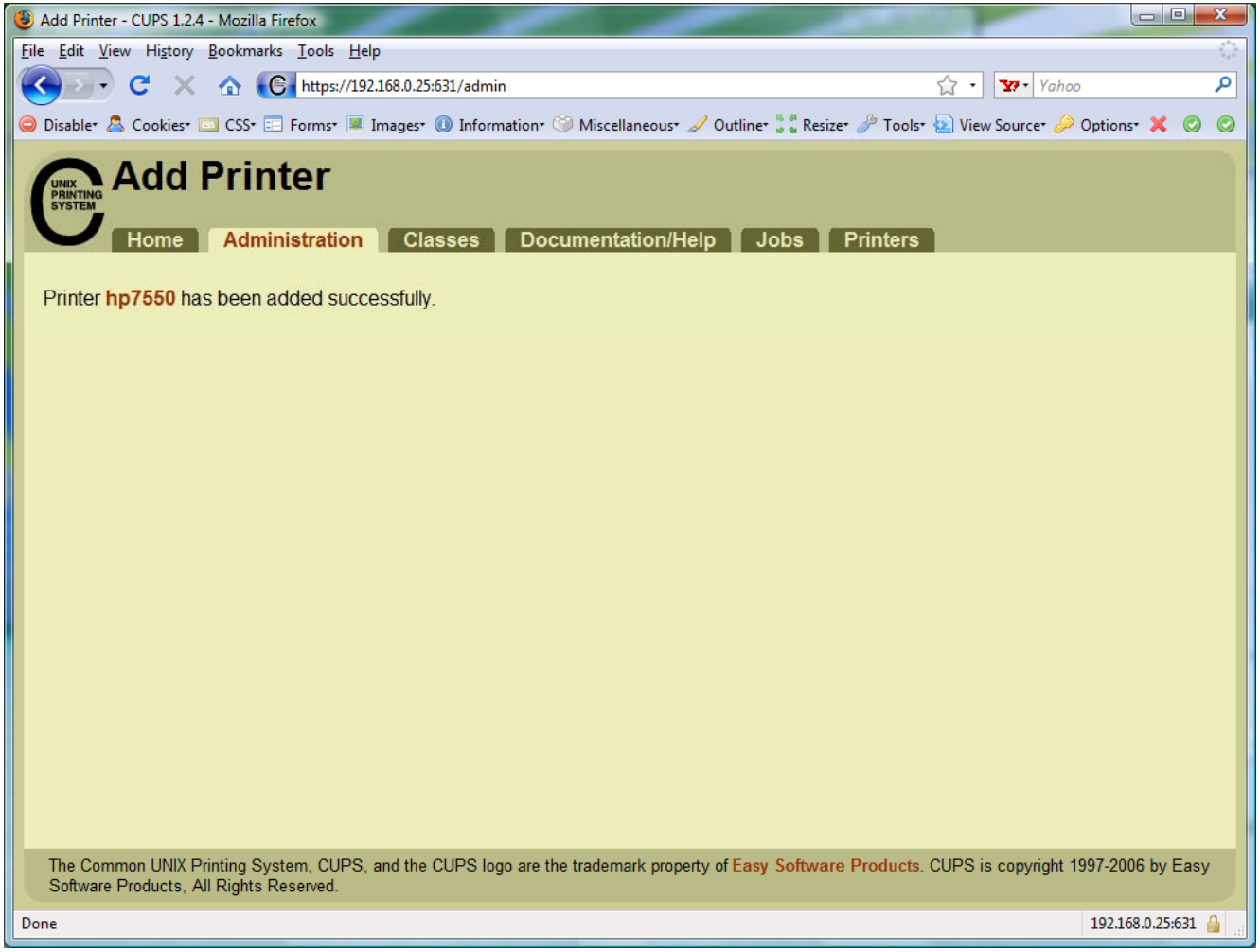
CUPS



Select model of printer

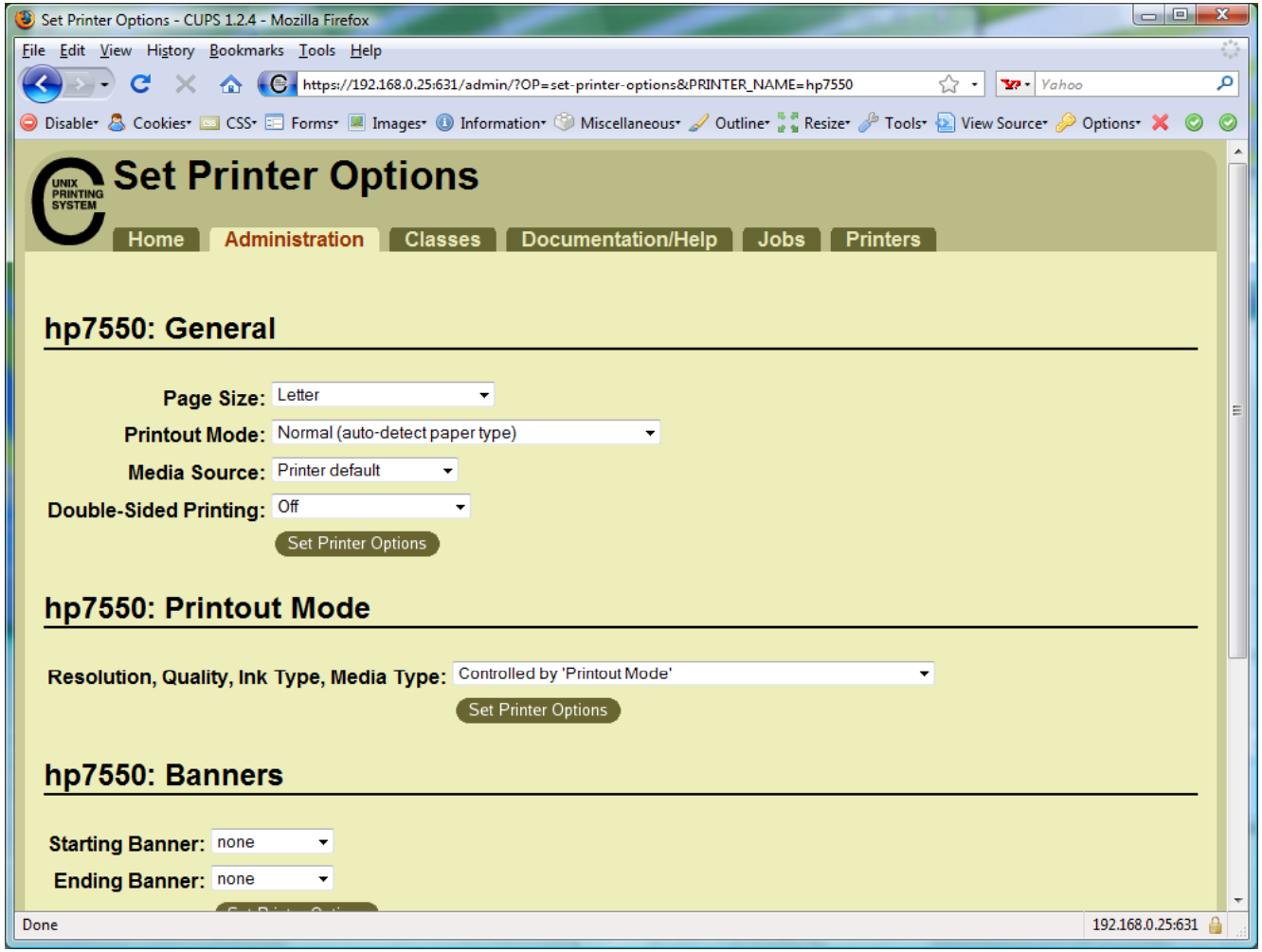
HP PhotoSmart 7550 Foomatic/hpijs (recommended) (en)

CUPS



Printer has been added

CUPS



View and set options as needed

Before using the printer we need to check that SAMBA is installed

Printing in Linux

Printing

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

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

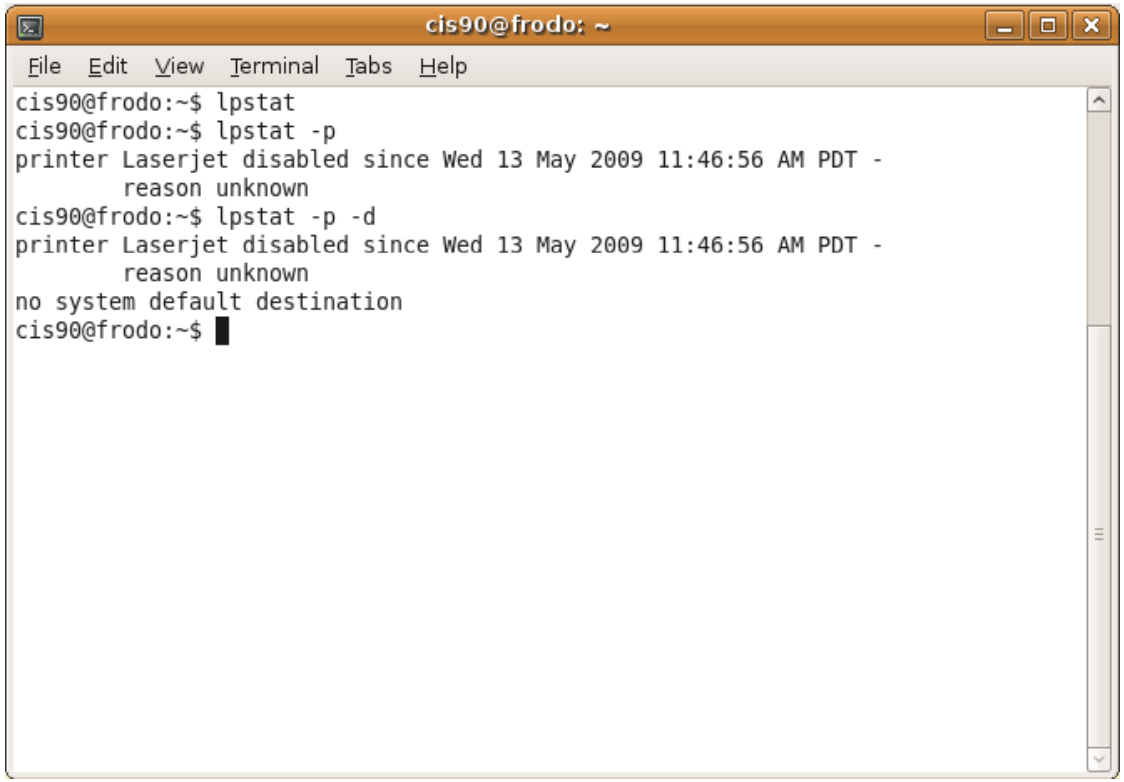
CUPS
- Provides both System V and Berkeley based command-line interfaces
- Supports new Internet Printing Protocol
- Works with Samba

We will be just looking at CUPS

CUPS

lpstat command

On the Frodo VM



The –p option will show the available printers
The –d option will identify the default printer

CUPS

lpstat command

On Opus

```
/home/cis90/roddyduk $ lpstat -p -d
printer epson disabled since Tue 11 Nov 2008 01:36:13 PM PST -
        reason unknown
printer hplaser disabled since Tue 11 Nov 2008 01:36:13 PM PST -
        reason unknown
system default destination: hplaser
/home/cis90/roddyduk $
```

The –p option will show the available printers
The –d option will identify the default printer

CUPS

lp and lpr commands


```
/home/cis90/roddyduk $ lp -d hplaser lab10
request id is hplaser-3 (1 file(s))

/home/cis90/roddyduk $ lpr -P hplaser lab10

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

/home/cis90/roddyduk $ lpr lab10
```

Either command will print lab10 to the selected printer

Or to the default printer

CUPS

lp and lpr commands


```
/home/cis90/roddyduk $ echo "Print Me Loudly" | lp -d epson
request id is epson-7 (1 file(s))

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

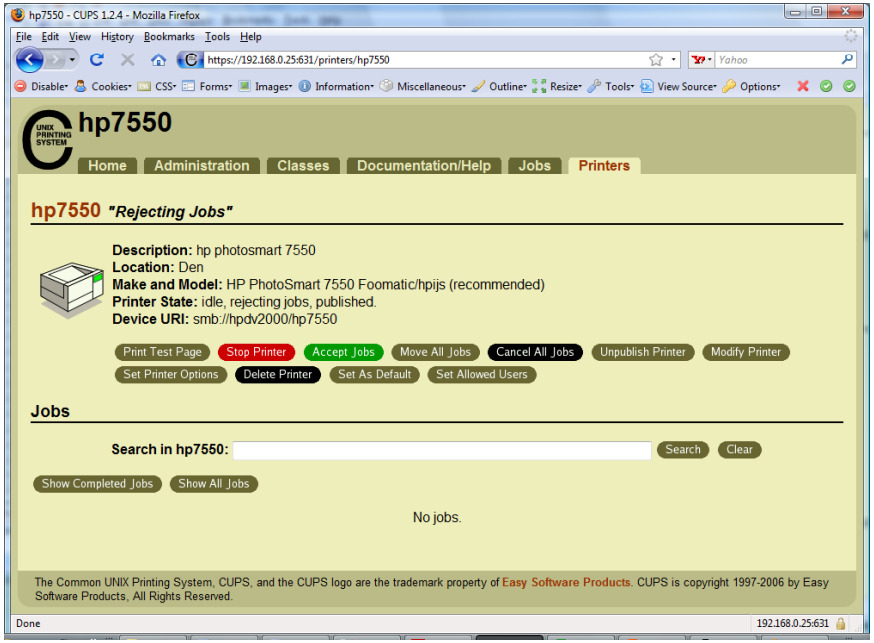
Note that both lp and lpr will read from stdin.

This allows output from another command to be piped in

Managing Print Jobs

CUPS

Rejecting Jobs



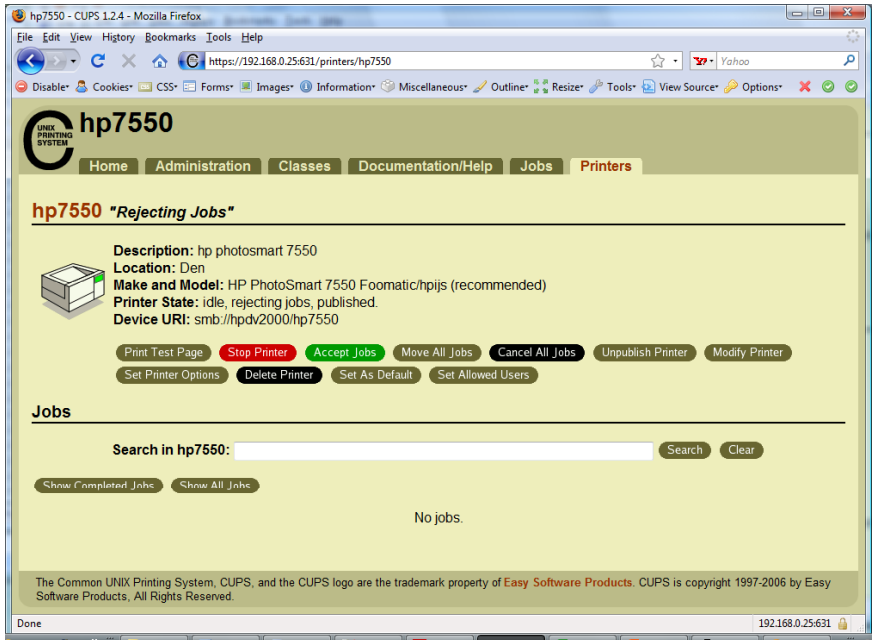
Clicking the Reject Jobs *button on the web based utility will reject further jobs*

```
[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

Stopping the Printer

```
[root@benji ~]# lp myfile
request id is hp7550-22 (1 file(s))
[root@benji ~]# lpr myfile
[root@benji ~]# lp myfile
request id is hp7550-24 (1 file(s))
[root@benji ~]# lpr myfile

[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 ~]# 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
```



Clicking the Stop Printer *button on the web based utility will still allow jobs to be spooled*

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 lpq or lpstat to show the spooled print jobs

Use cancel or lprm to remove print jobs

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

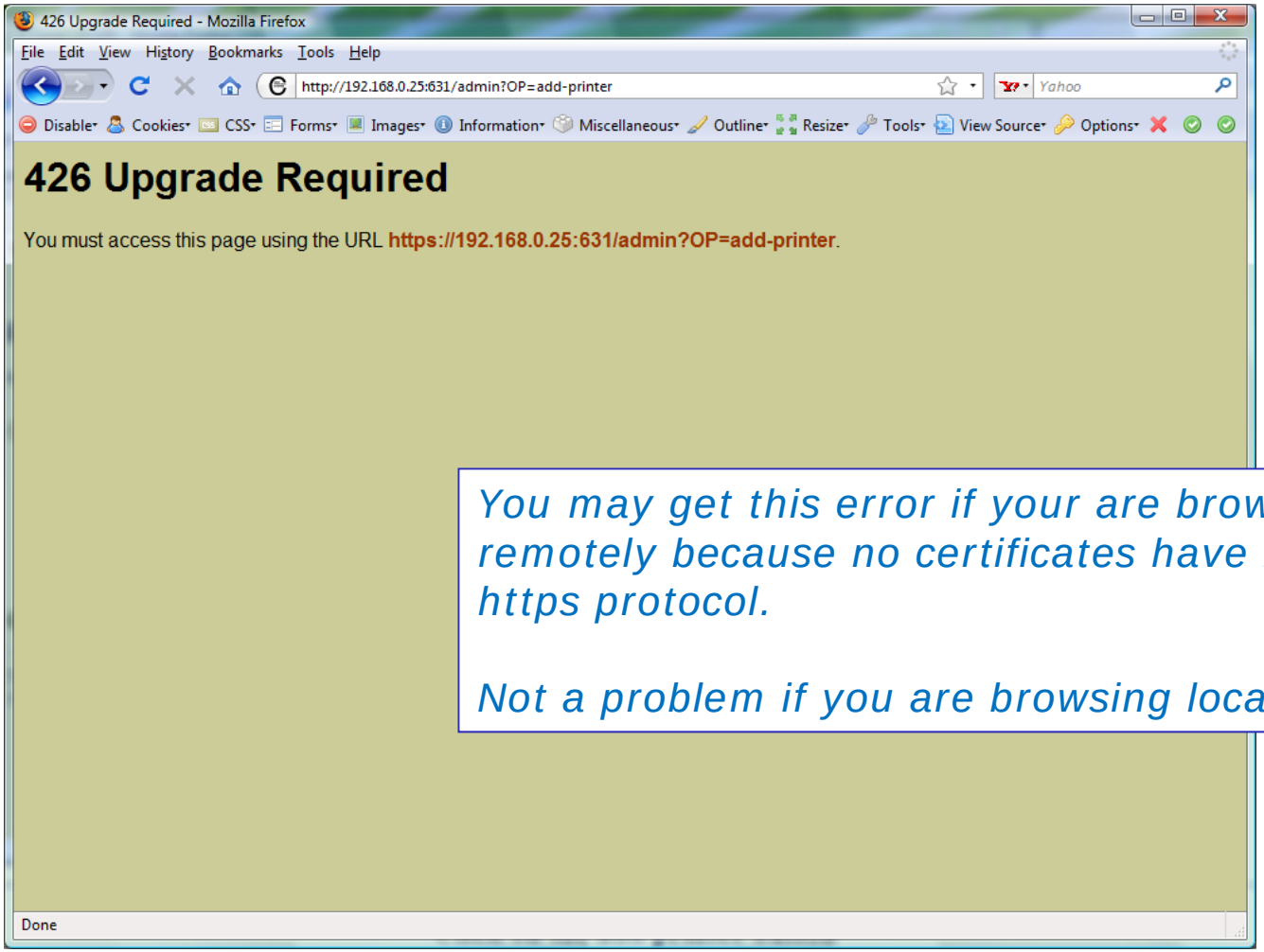
No Lab due

Work on final projects

Optional extra credit lab

Backup

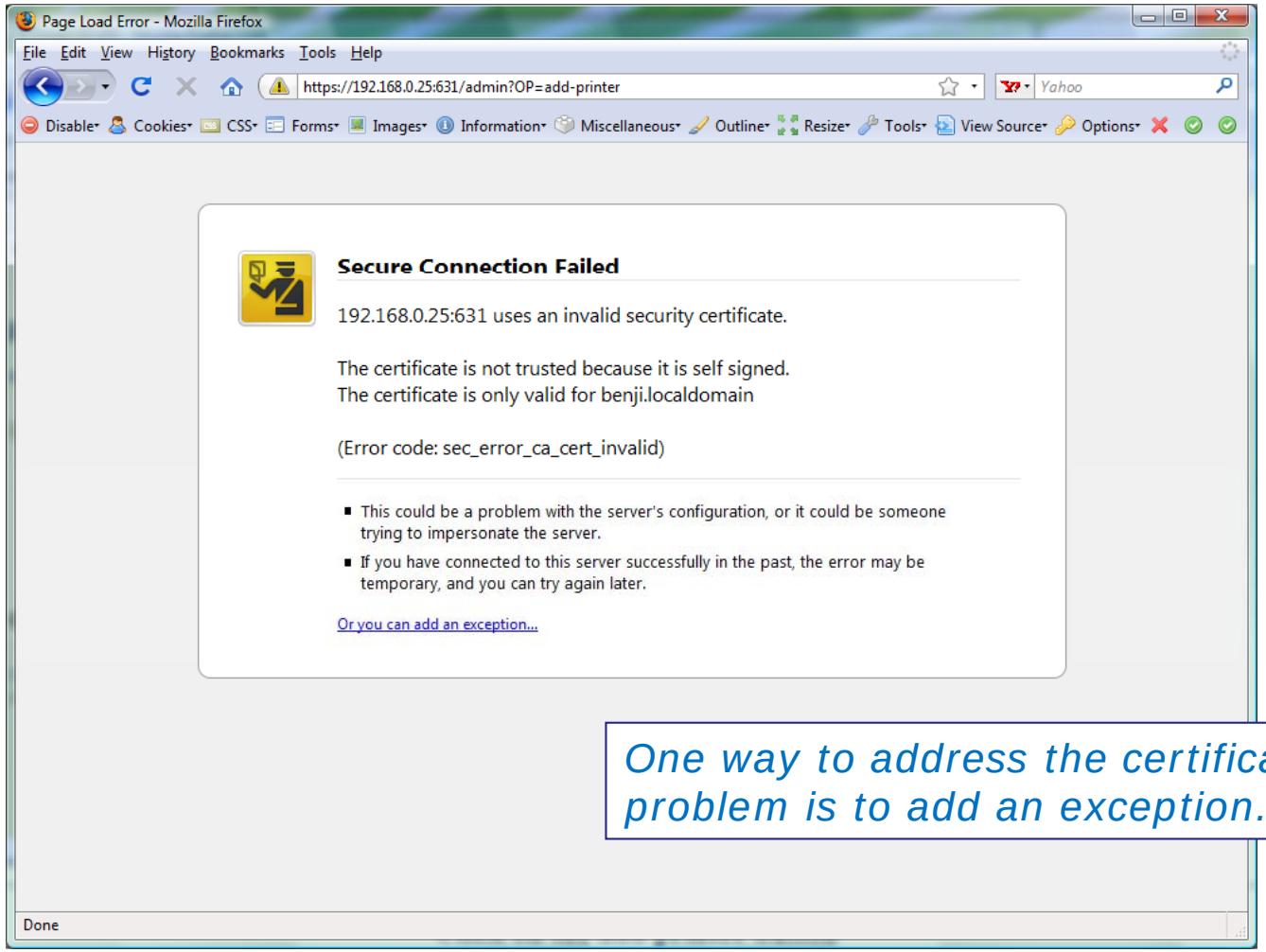
CUPS



You may get this error if your are browsing in remotely because no certificates have been set up for https protocol.

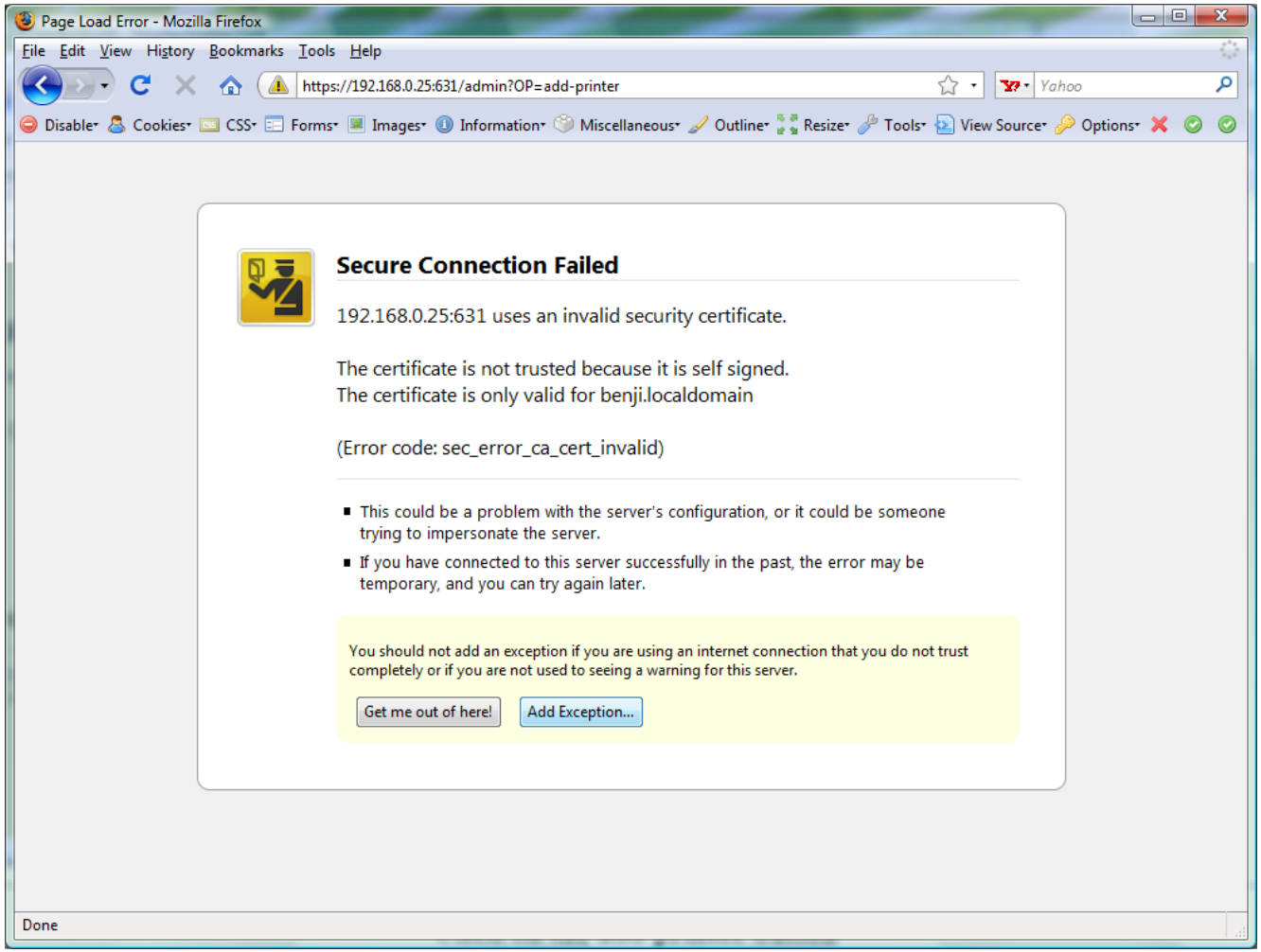
Not a problem if you are browsing locally

CUPS



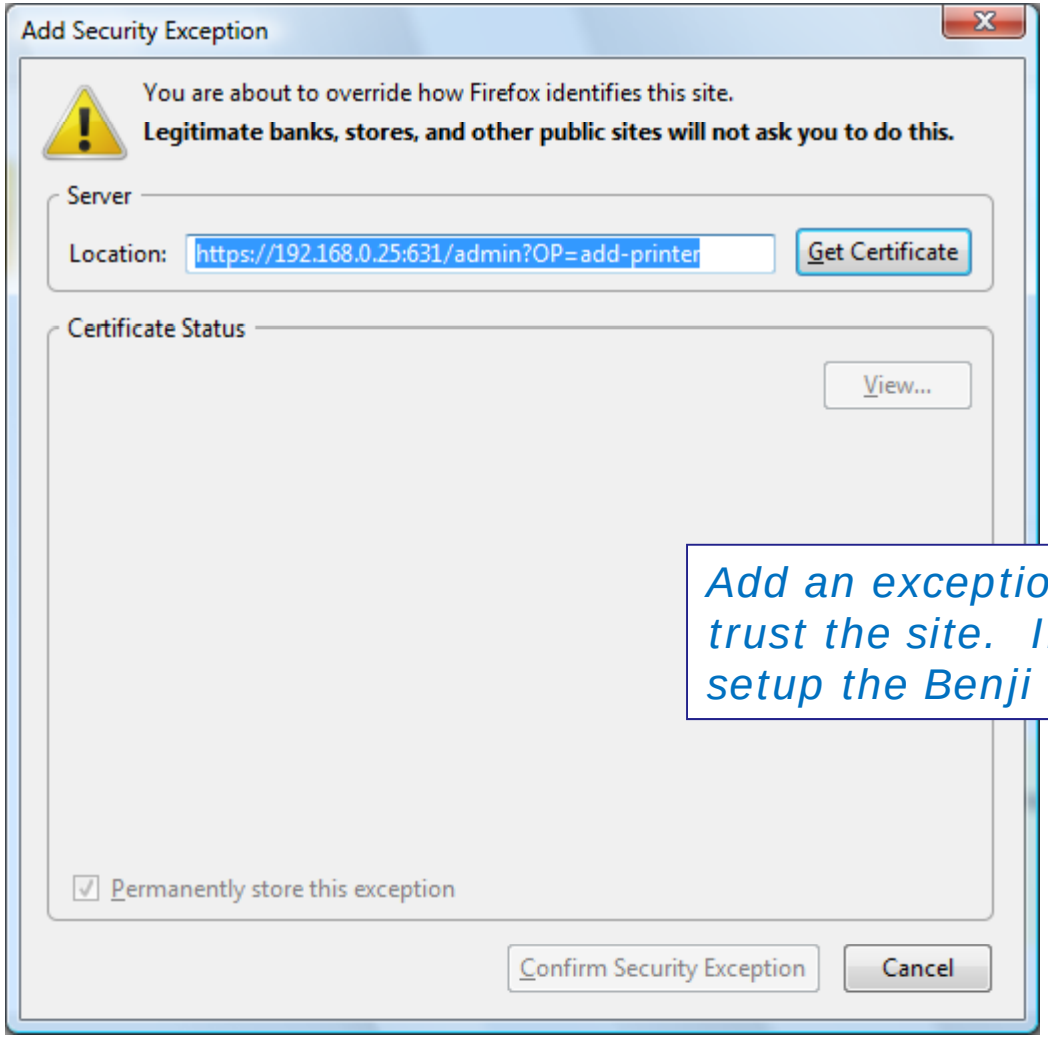
One way to address the certificate problem is to add an exception.

CUPS



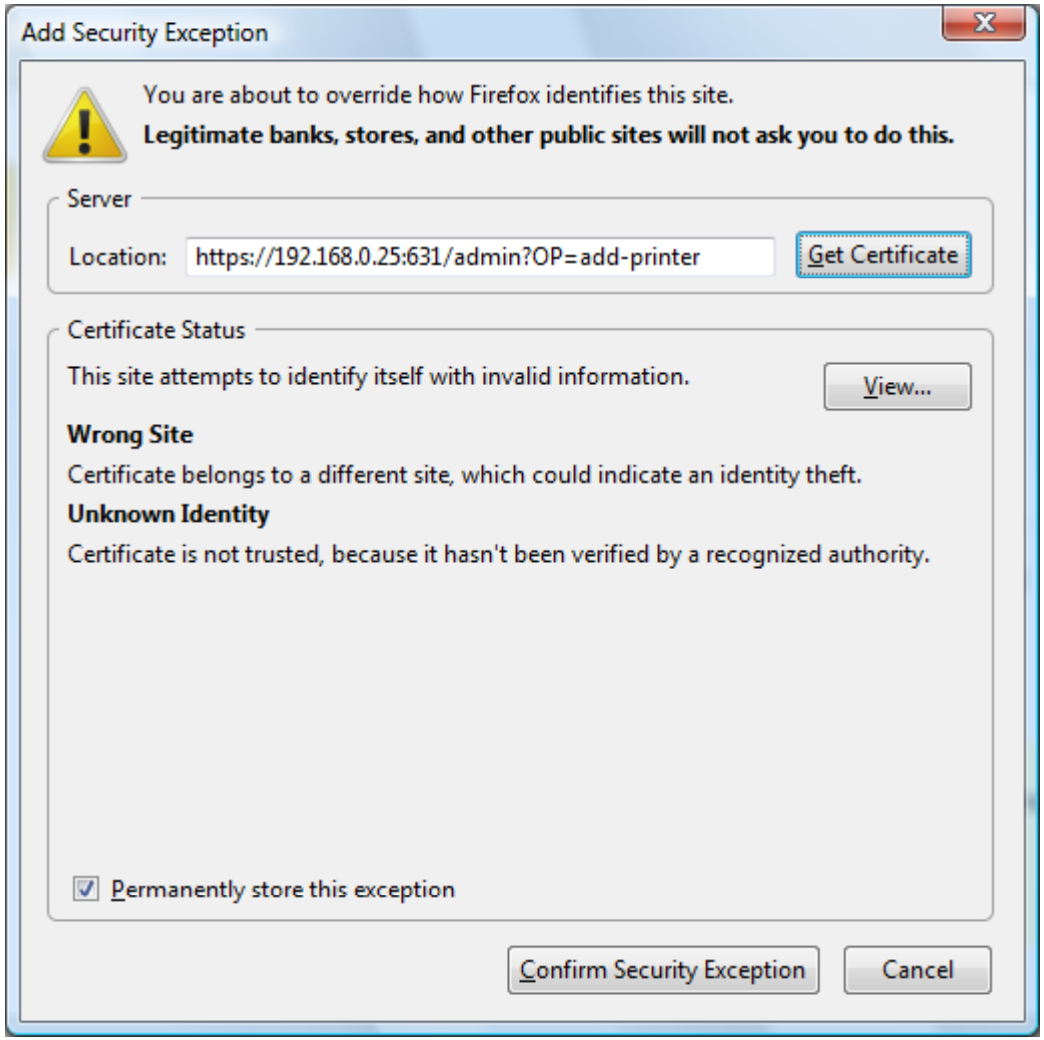
This adds the exception

CUPS



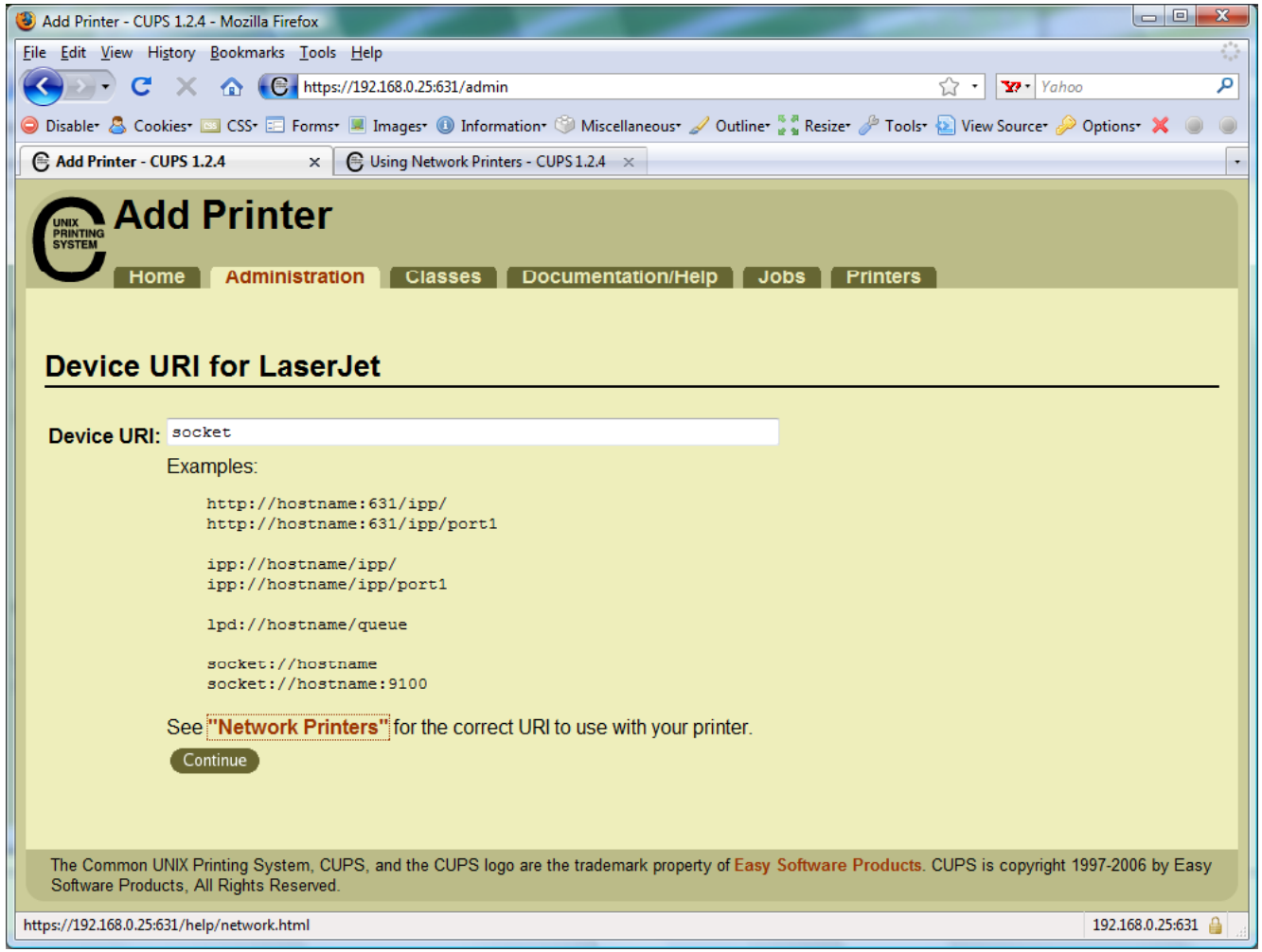
Add an exception only when you do trust the site. In this case we built and setup the Benji VM so we trust it.

CUPS



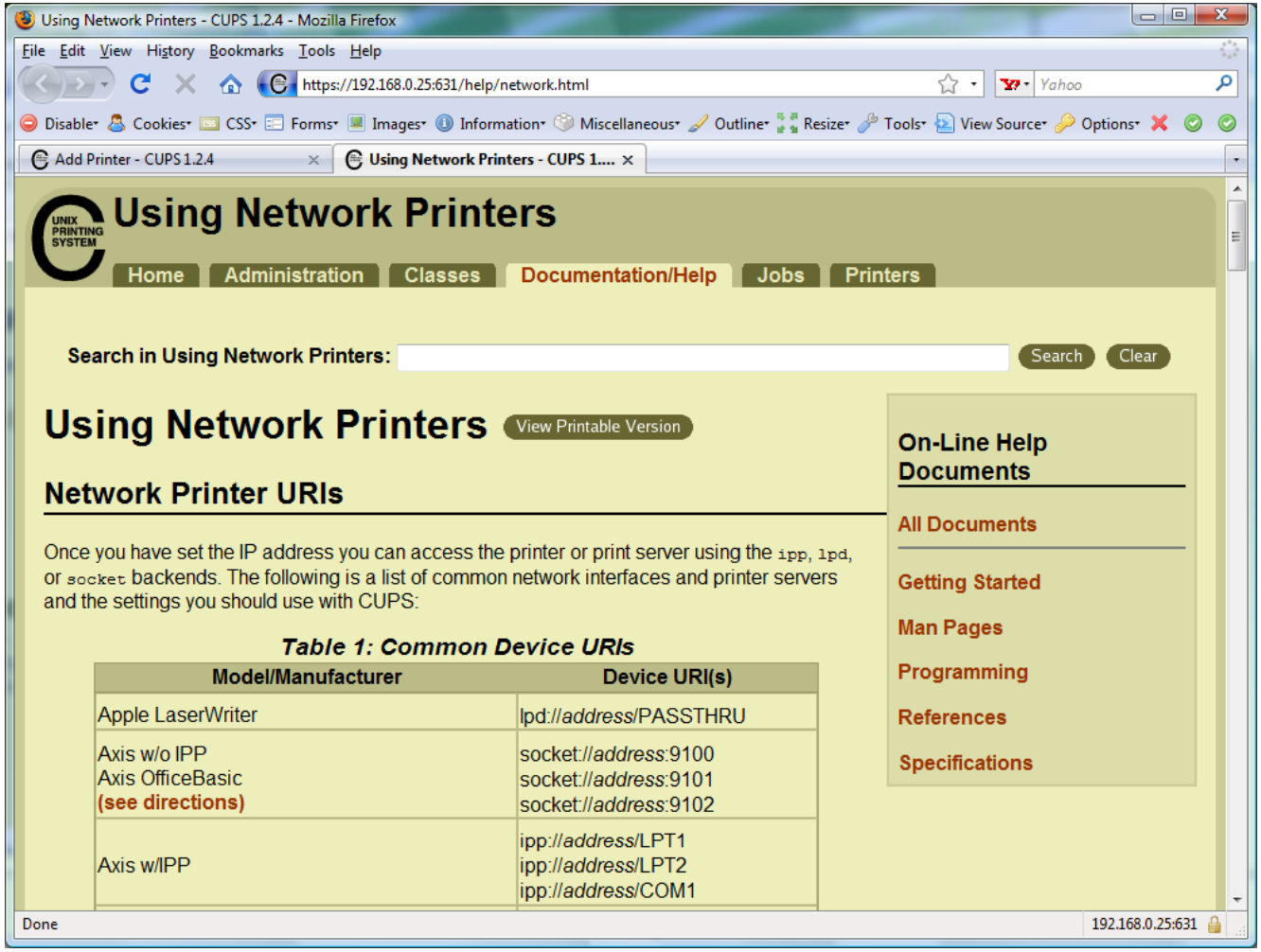
Click Confirm Security Exception button

CUPS



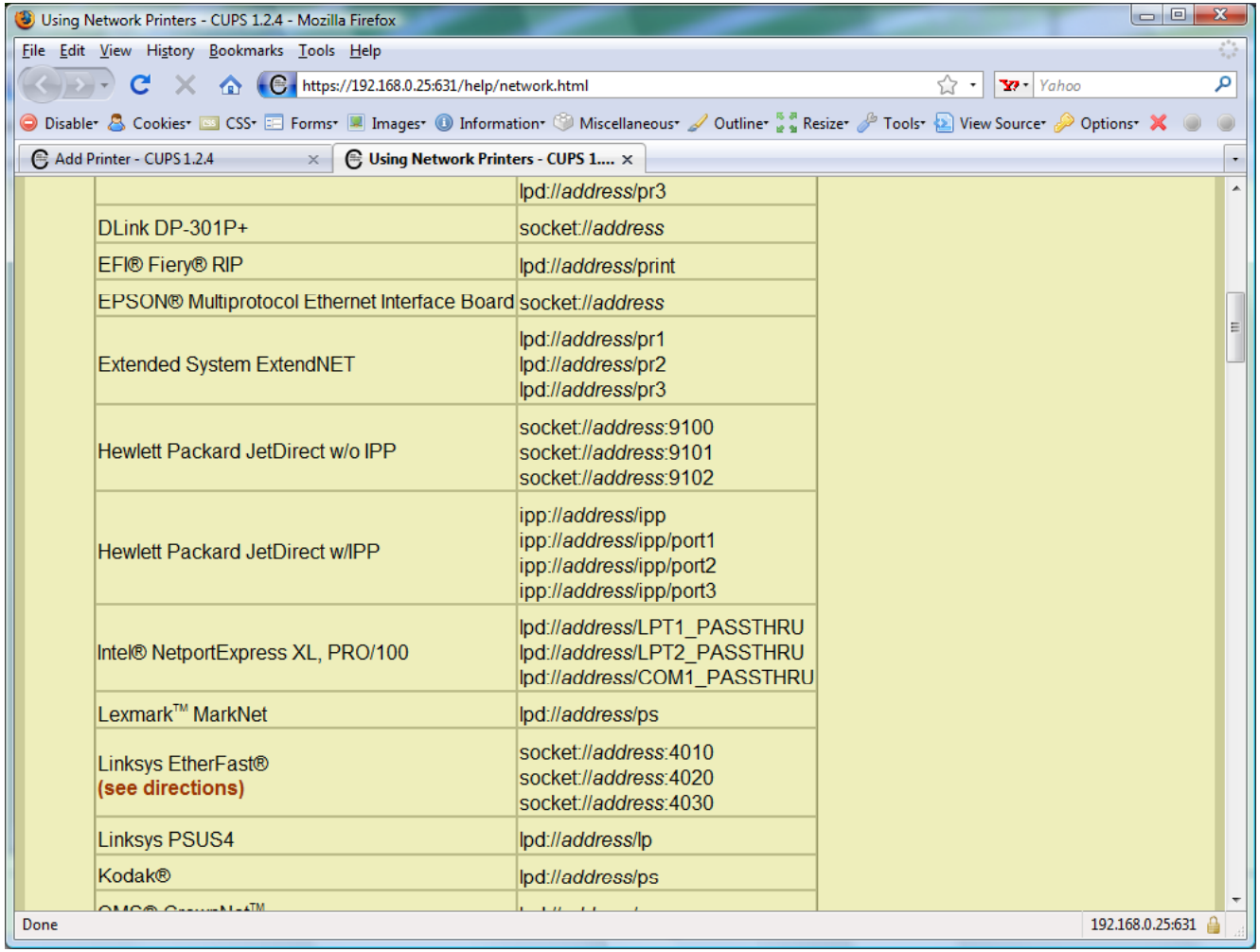
Hmmm …. lets click on Network Printers link to figure this one out.

CUPS



Scroll down to HP printers

CUPS



	lpd://*address*/pr3
DLink DP-301P+	socket://*address*
EFI® Fiery® RIP	lpd://*address*/print
EPSON® Multiprotocol Ethernet Interface Board	socket://*address*
Extended System ExtendNET	lpd://*address*/pr1 lpd://*address*/pr2 lpd://*address*/pr3
Hewlett Packard JetDirect w/o IPP	socket://*address*:9100 socket://*address*:9101 socket://*address*:9102
Hewlett Packard JetDirect w/IPP	ipp://*address*/ipp ipp://*address*/ipp/port1 ipp://*address*/ipp/port2 ipp://*address*/ipp/port3
Intel® NetportExpress XL, PRO/100	lpd://*address*/LPT1_PASSTHRU lpd://*address*/LPT2_PASSTHRU lpd://*address*/COM1_PASSTHRU
Lexmark™ MarkNet	lpd://*address*/ps
Linksys EtherFast® **(see directions)**	socket://*address*:4010 socket://*address*:4020 socket://*address*:4030
Linksys PSUS4	lpd://*address*/lp
Kodak®	lpd://*address*/ps

We will use the JetDirect w/o IPP for the HP 1320n

IPP is Internet Printing Protocol for send print jobs over the Internet via the http protocol

HP JetDirect cards use port 9100

A socket is the combination of an IP address and a port number.

CUPS

convert command

JPEG files need to be converted to postscript before printing with lp or lpr commands

```
[root@benji Desktop]# convert benji-500x420.jpg benji-500x420.ps
[root@benji Desktop]# lp benji-500x420.ps
request id is hp7550-29 (1 file(s))
[root@benji Desktop]# lpq
hp7550 is not ready
Rank    Owner   Job     File(s)                         Total Size
1st     root    28      benji-500x420.ps                1284096 bytes
2nd     root    29      benji-500x420.ps                1284096 bytes
[root@benji Desktop]# cancel 29
[root@benji Desktop]# cd /var/spool/cups/
[root@benji cups]# ls
0000001b  c00009  c00012  c00015  c00018  c00021  c00024  c00027  d00028-001
c00001    c00010  c00013  c00016  c00019  c00022  c00025  c00028  tmp
c00008    c00011  c00014  c00017  c00020  c00023  c00026  c00029
[root@benji cups]# ls tmp
```

Print job #28

Process Information

Use –l for additional options

```
[rsimms@opus ~]$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S   201  6204  6203  0  75   0 -  1165 wait   pts/6    00:00:00 bash
0 R   201  6521  6204  0  77   0 -  1050 -      pts/6    00:00:00 ps
```

Parent Process ID

Process ID

User ID

Running or sleeping

Size in 1K blocks

Common Environment Variables

```
/home/cis90/roddyduk $ 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 :\?${TERM:-ansi} -r -Q `

/home/cis90/roddyduk $
```

On Opus, PS1 is set in /etc/bashrc and then redefined in .bash_profile

Common Environment Variables

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

# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
alias print="echo -e"
/home/cis90/roddyduk $
```

```
/home/cis90/roddyduk $ cat /etc/bashrc | grep PS1
if [ "$PS1" ]; then
    [ "$PS1" = "\\s-\\v\\\$ " ] && PS1="[\u@\h \W]\\$ "
/home/cis90/roddyduk $
```

On Opus, PS1 is set in /etc/bashrc and then redefined in .bash_profile

```
/home/cis90/roddyduk $ cat program
echo "program is being run"
fan=high
ac=on
export ac
alias copy=cp
```

*Use **vi** to make this file, then use **chmod** to give it execute permissions*

```
/home/cis90/roddyduk $ show
fan= ac=
-bash: type: copy: not found
```

Initial state

```
/home/cis90/roddyduk $ program
program is being run
/home/cis90/roddyduk $ show
fan= ac=
-bash: type: copy: not found
```

Not changed

```
/home/cis90/roddyduk $ source program
program is being run
/home/cis90/roddyduk $ show
fan=high ac=on
copy is aliased to `cp'
ac=on
```

Changed

Do you get the same results?

*Note: using **alias show='echo fan=$fan ac=$ac; type copy; env | grep ac'***