Lesson Module Checklist

- Slides –
- Properties –
- Flash cards –
- First minute quiz –
- Web calendar summary –
- Web book pages –
- Commands –
- Lab –

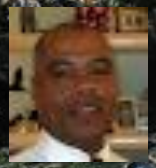
- Email Tech file for Lab 9
- CCC Confer wall paper –

- Materials uploaded –
- Backup headset charged –
- Backup slides, CCC info, handouts on flash drive -

- Check that room headset is charged –

CIS 90 - Lesson 11

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



Dieskau

Jonathan

Ana

David

Obie

Dave

Cole

Corey

Nancy

Ryan

Elia

Tasha

Darren

Scott

Devin

Everett

Juan

Raven

Rogan

Mike

Mook

Melissa

Cameron

Jose

Jeff

Matt

Kenneth

Ousmane

Ian

Solomon

Henry

Matthew

Mason

Chan

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

Quiz

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

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

[ ] Has the phone bridge been added?

[ ] Is recording on?

[ ] Share slides, putty x 3, and chrome

[ ] Disable spelling on PowerPoint

vi editor

Objectives	Agenda
• Create and modify text files	• Quiz • Questions from last week • grep • Review on processes • vi • Wrap up

Questions?

- Test 2?
- Lab 8?
- Previous course material?

Housekeeping

Previous material and assignment

1. Questions?

2. Lab 8 due tonight

```
at 11:59pm
at> cat files.out bigshell > lab08
at> cp lab08 /home/rsimms/turnin/lab08.$LOGNAME
at> Ctrl-D
```

Don't wait till midnight tonight to see if this worked! Test with an earlier time.

3. Note: Lab 9 and five posts due next week

More on grep

grep usage

What is my account information in /etc/passwd?

```
/home/cis90/simben $ grep $LOGNAME /etc/passwd
simben90:x:1000:90:Benji Simms:/home/cis90/simben:/bin/bash
```

or

```
/home/cis90/simben $ grep simben90 /etc/passwd
simben90:x:1000:90:Benji Simms:/home/cis90/simben:/bin/bash
```

or

```
/home/cis90/simben $ cat /etc/passwd | grep $LOGNAME
simben90:x:1000:90:Benji Simms:/home/cis90/simben:/bin/bash
```

My user account is simben90, my password is kept in /etc/shadow, my user ID is 1000, my primary group ID is 90, my full name is Benji Simms, my home directory is /home/cis90/simen, my shell is /bin/bash

grep usage

Is the CUPS daemon (print service) running right now?

```
/home/cis90/simben $ ps -ef | grep cups
root      2729     1  0 Apr13 ?        00:00:00 cupsd
simben90 15231 15180  0 05:44 pts/0    00:00:00 grep cups
```

Yes it is, with PID=2729

grep usage

Is the Apache web server (httpd) installed?

```
/home/cis90/simben $ rpm -qa | grep httpd
httpd-2.2.3-31.el5
```

Yes, it has been installed already

grep usage

How many CIS 90 user accounts are there?

```
/home/cis90/simben $ grep cis90 /etc/passwd | wc -l
37
```

There are 37

grep usage

When were that last 5 times I logged in?

```
/home/cis90/simben $ last | grep $LOGNAME | head -n5
simben90 pts/0        50-0-68-235.dsl. Mon Apr 23 05:39   still logged in
simben90 pts/6        10.64.25.2       Wed Apr 18 12:48 - 16:51  (04:02)
simben90 pts/5        10.64.25.2       Wed Apr 18 12:48 - 16:51  (04:02)
simben90 pts/4        10.64.25.2       Wed Apr 18 12:48 - 16:51  (04:03)
simben90 pts/1        50-0-68-235.dsl. Wed Apr 18 09:06 - 10:23  (01:17)
```

grep usage

```
/home/cis90/simmsben $ ls /bin/*sh
/bin/bash  /bin/csh  /bin/jsh  /bin/ksh  /bin/rbash  /bin/sh  /bin/tcsh

/home/cis90/simmsben $ csh
[simmsben@opus ~]$ bash
[simmsben@opus ~]$ sh
sh-3.2$ jsh
Enter Command: ksh

$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1200 20540 20539  0  75   0 -  1168 wait   pts/1    00:00:00 bash
0 S  1200 20618 20540  0  75   0 -  1330 rt_sig pts/1    00:00:00 csh
0 S  1200 20639 20618  0  75   0 -  1169 wait   pts/1    00:00:00 bash
0 S  1200 20663 20639  0  75   0 -  1167 wait   pts/1    00:00:00 sh
0 S  1200 20666 20663  0  75   0 -   380 wait   pts/1    00:00:00 jsh
0 S  1200 20669 20666  0  76   0 -  1236 wait   pts/1    00:00:00 ksh
0 R  1200 20673 20669  0  76   0 -  1054 -      pts/1    00:00:00 ps

$ ps -l | grep csh
0 S  1200 20618 20540  0  75   0 -  1330 rt_sig pts/1    00:00:00 csh
$ ps -l | grep csh > bigshell

$ cat bigshell
0 S  1200 20618 20540  0  75   0 -  1330 rt_sig pts/1    00:00:00 csh
```

Which shell is the biggest (Lab 8)?

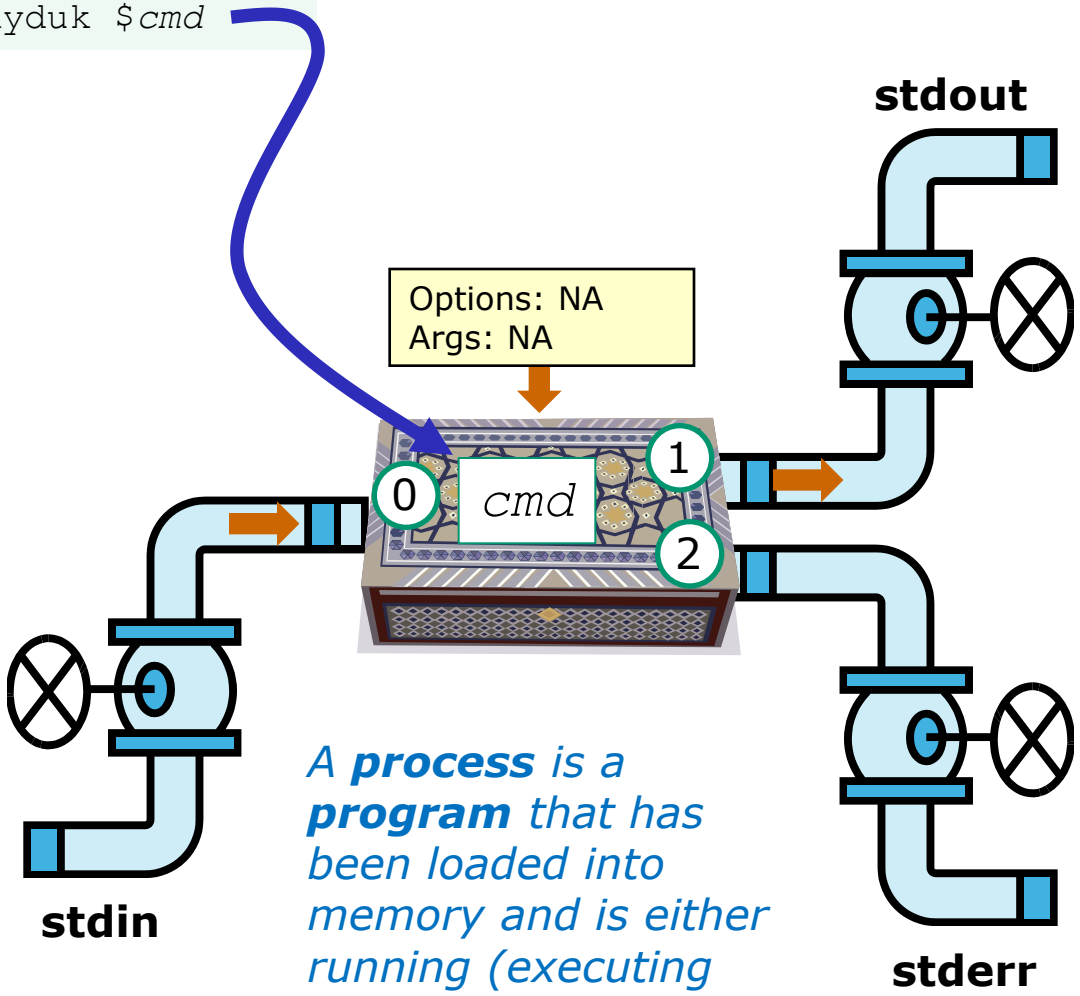
grep practice

- How many CIS 172 accounts are there on Opus?
- Is the cronjob daemon (crond) running right now?
- Has the mysql-server package been installed on Opus?

Review of Processes

Program to process

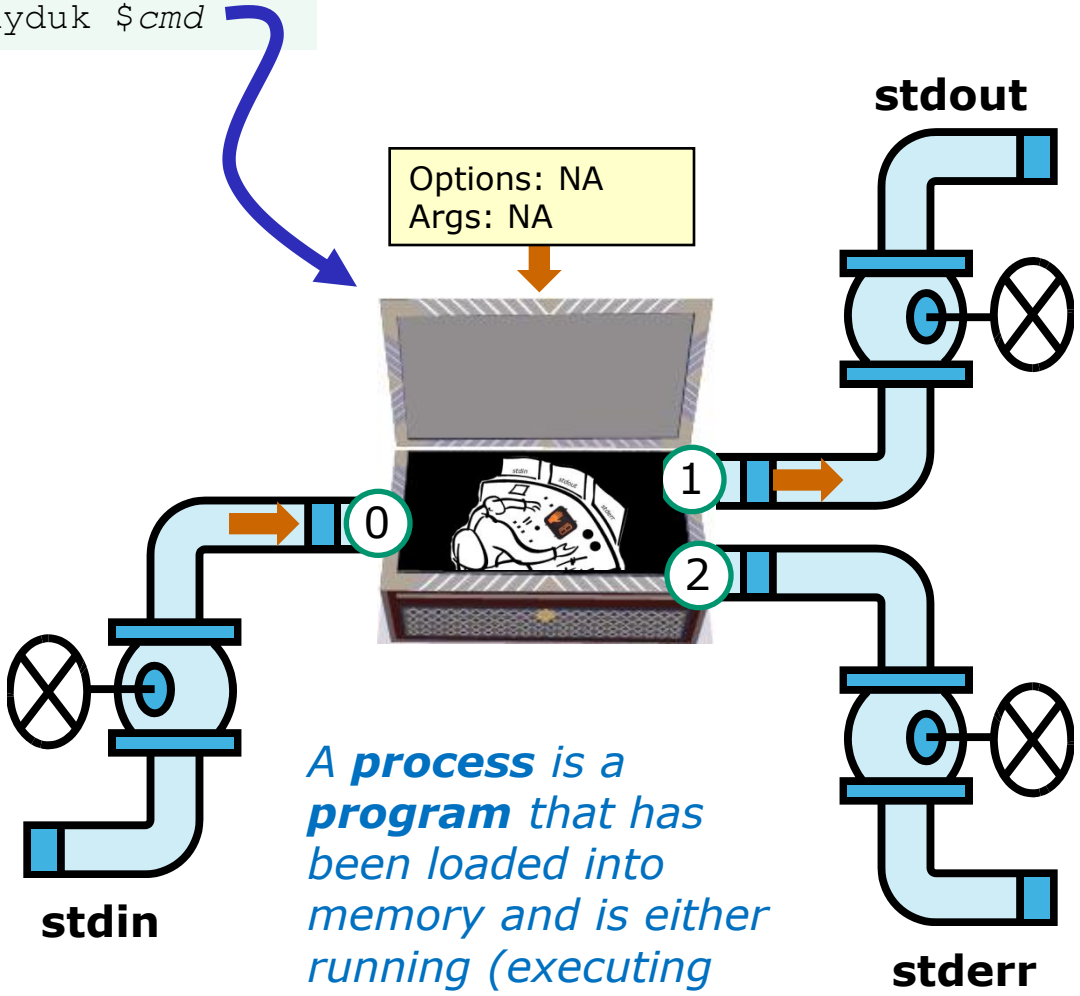
```
/home/cis90/roddyduk $cmd
```



*A **process** is a **program** that has been loaded into memory and is either running (executing instructions) or waiting to run*

Program to process

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



*A **process** is a **program** that has been loaded into memory and is either running (executing instructions) or waiting to run*

A Process at Work

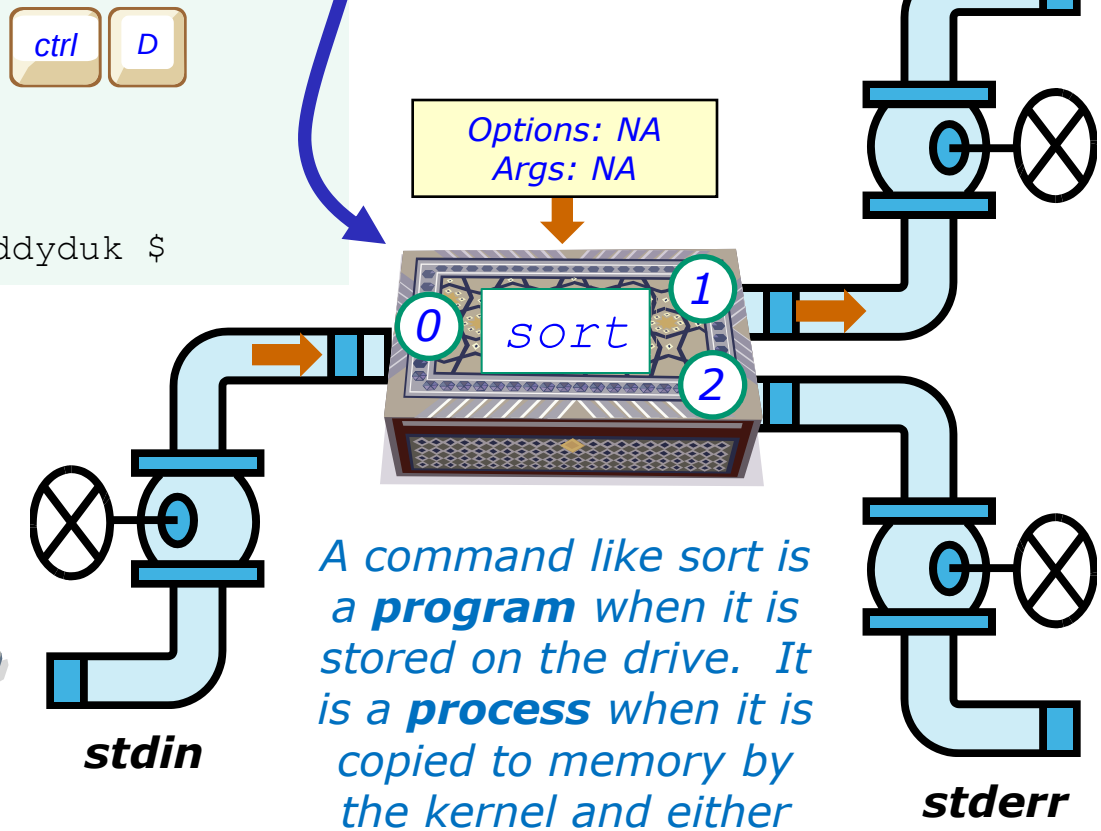
A **process**
- reads from **stdin**
- writes to **stdout**
- puts error messages in **stderr**

- and may get interrupted from time to time by a **signal**

*A **process** is a **program** that has been loaded into memory and is either running (executing instructions) or waiting to run*

Example program to process: sort command

```
/home/cis90/roddyduk $ sort
duke
benji
star
homer
benji
duke
homer
star
/home/cis90/roddyduk $
```

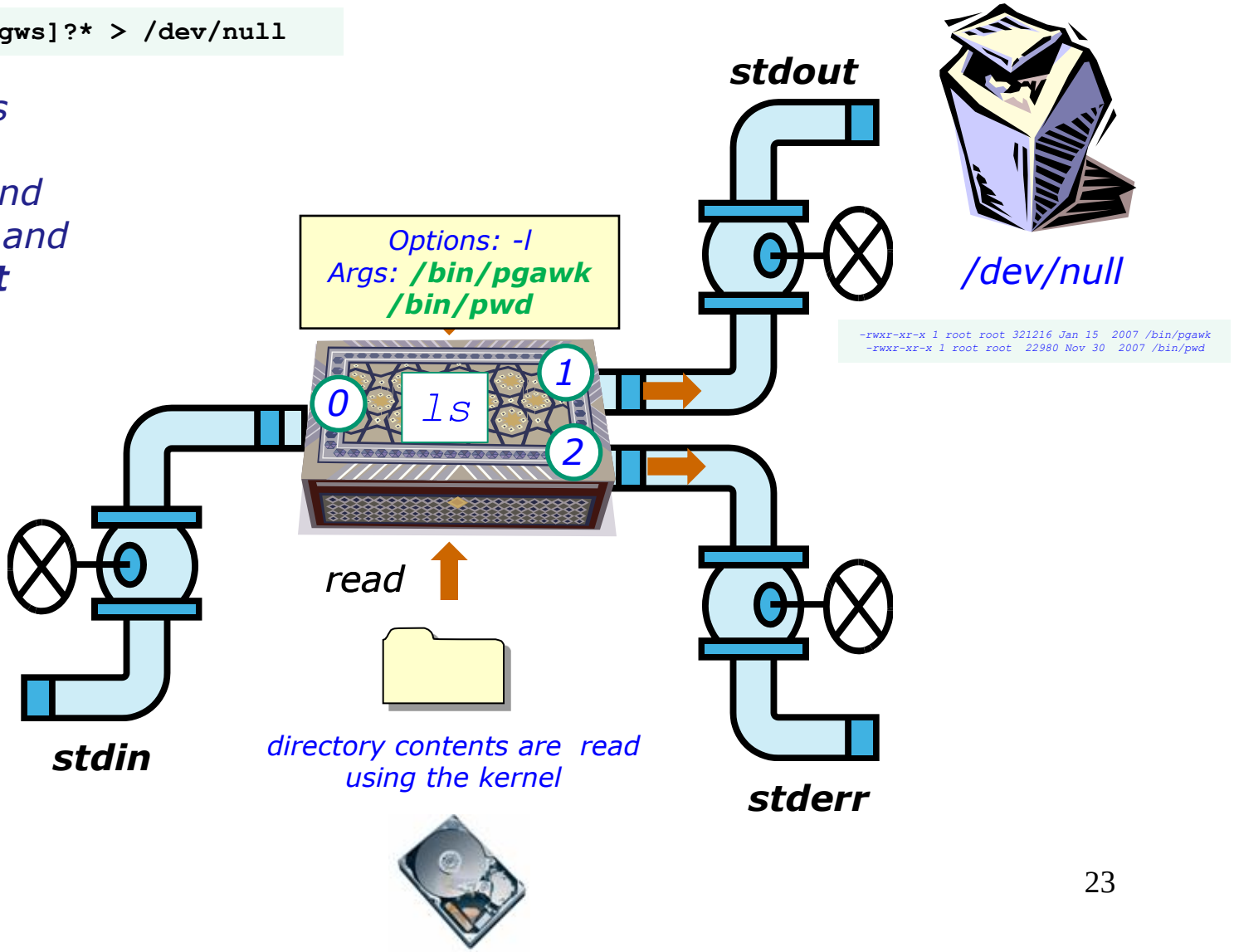


*A command like sort is a **program** when it is stored on the drive. It is a **process** when it is copied to memory by the kernel and either running or waiting to run.*

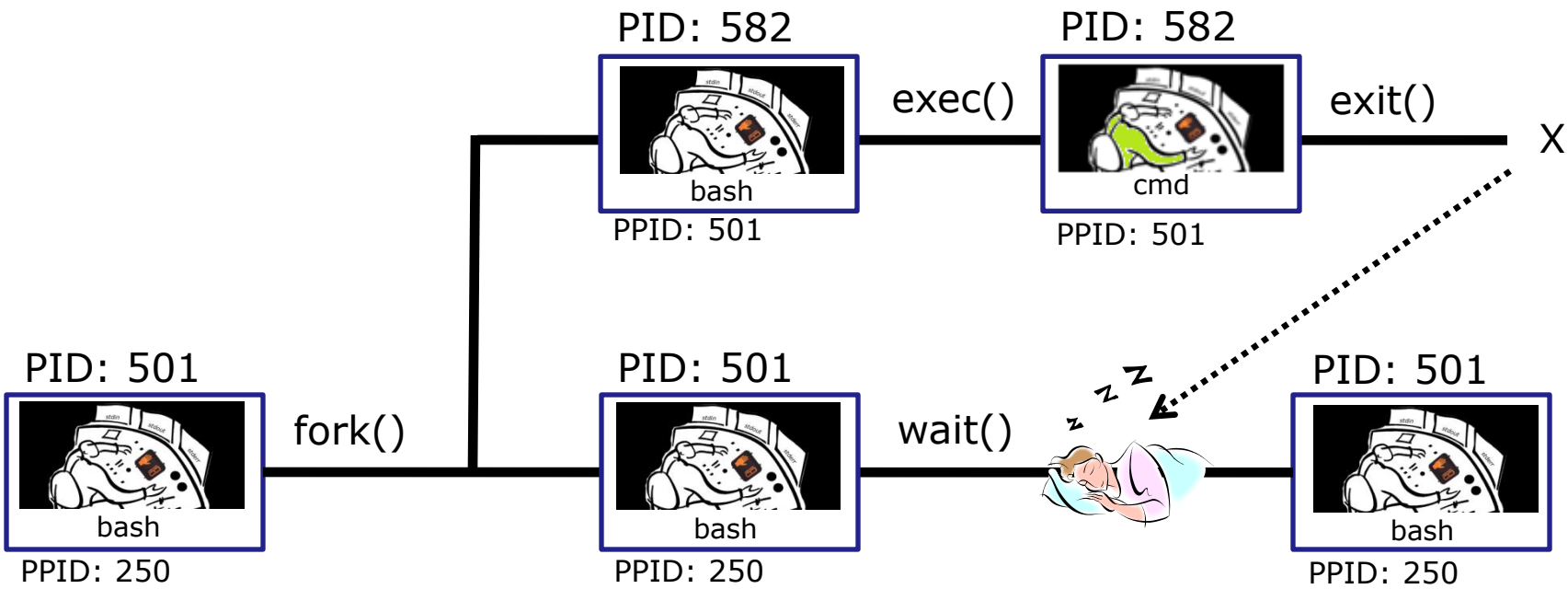
example program to process

```
$ ls -l /bin/p[gws]?* > /dev/null
```

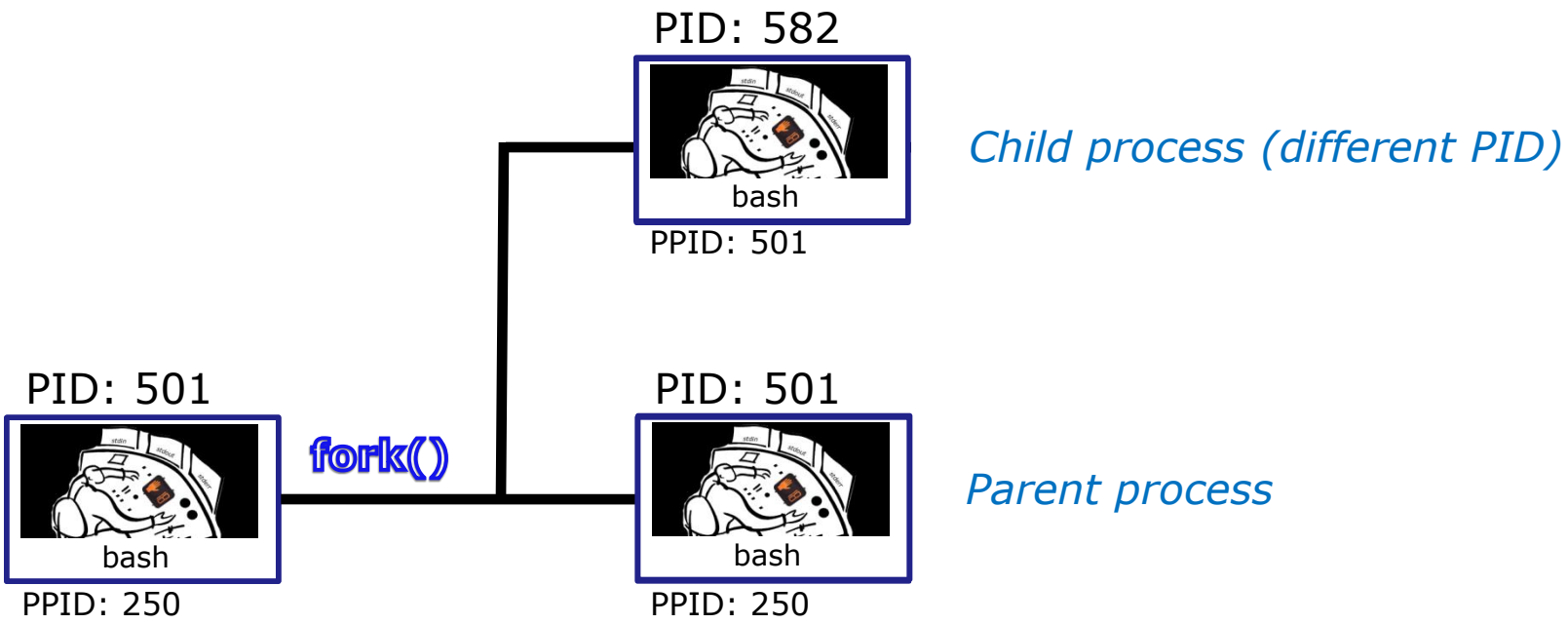
*Note: ls gets its input from the command line and the OS (kernel) and writes to **stdout** (redirected to /dev/null) and **stderr**.*



Process Lifecycle



Process Lifecycle

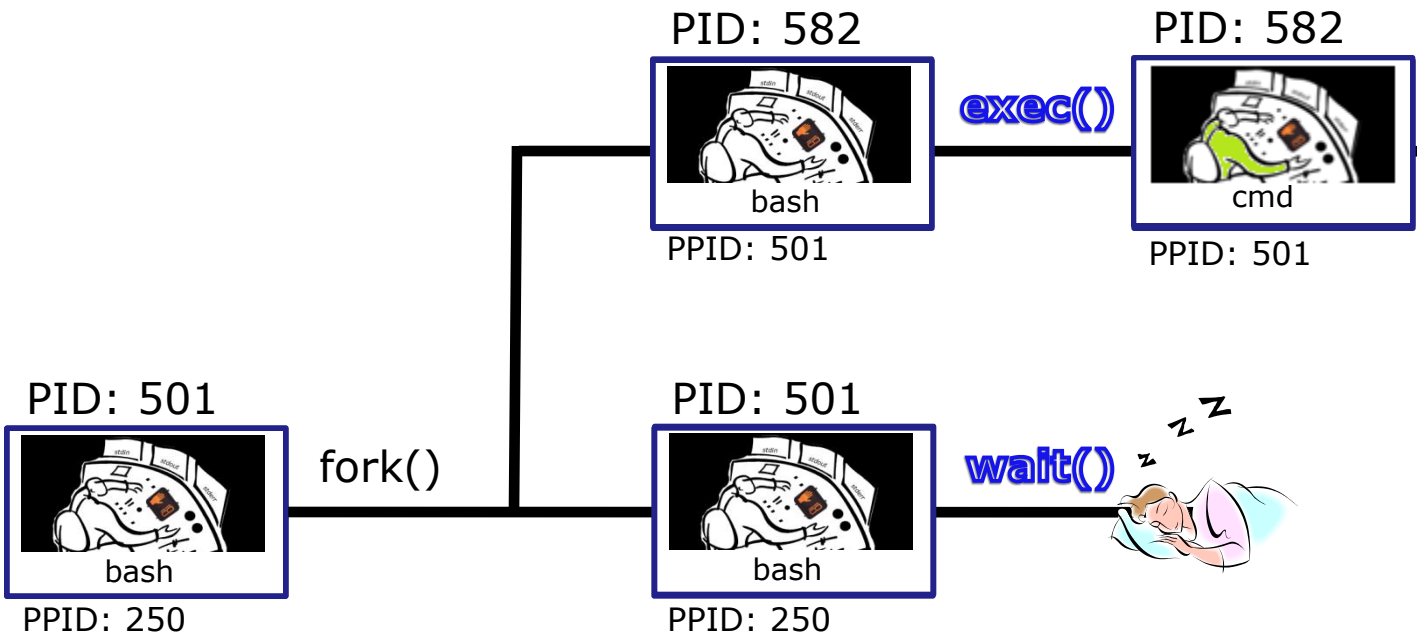


1) When a program is loaded into memory a new process must be created.

*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.*

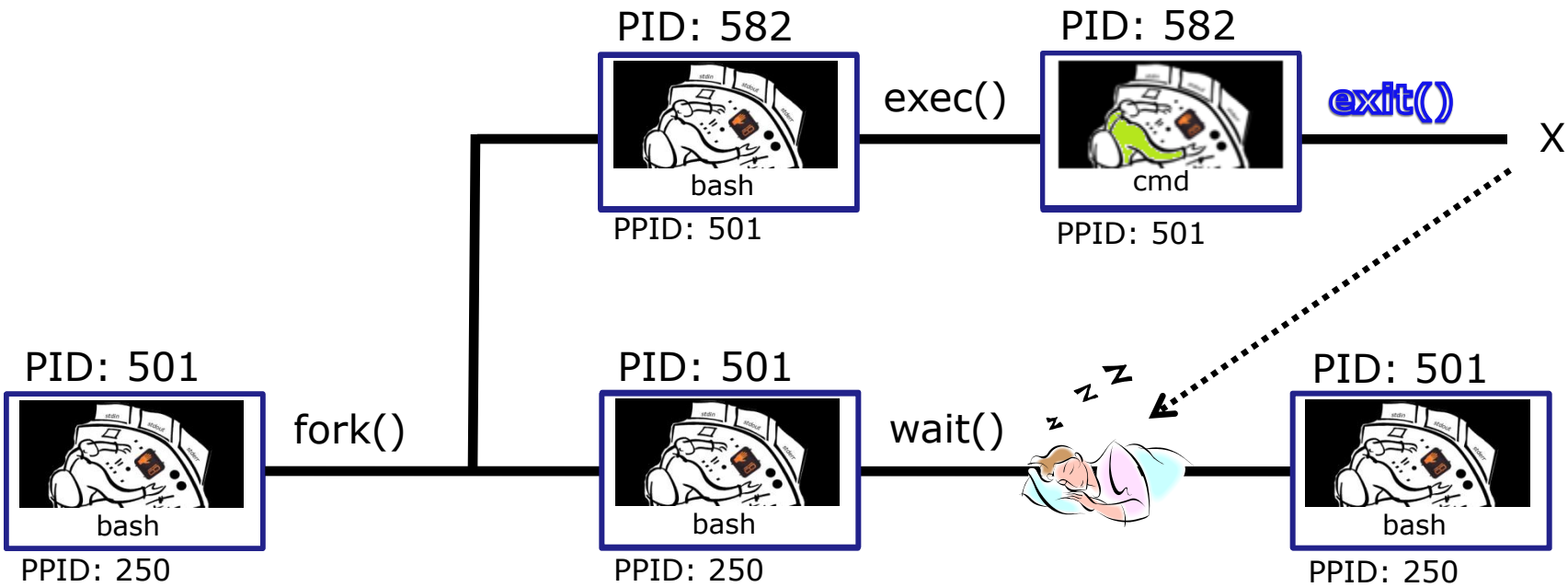
Process Lifecycle



*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.*

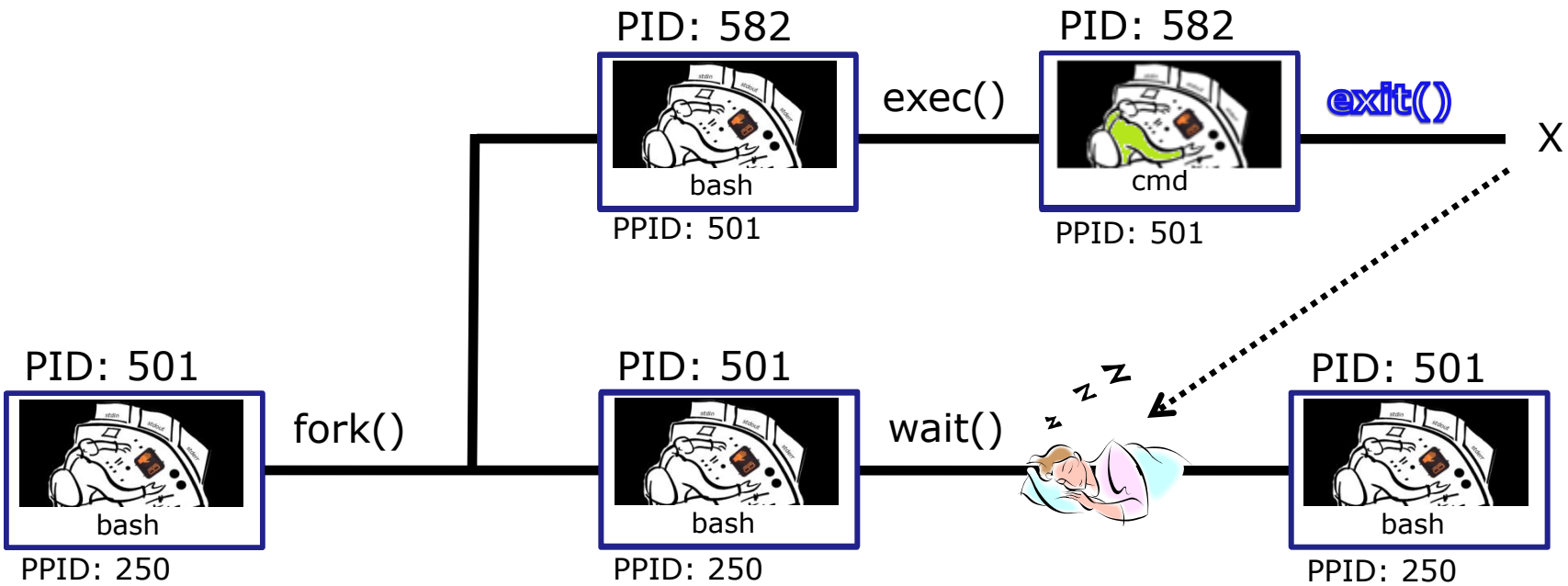
Process Lifecycle



*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**.*

Process Information

Use –l (long format) for additional information

```
[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

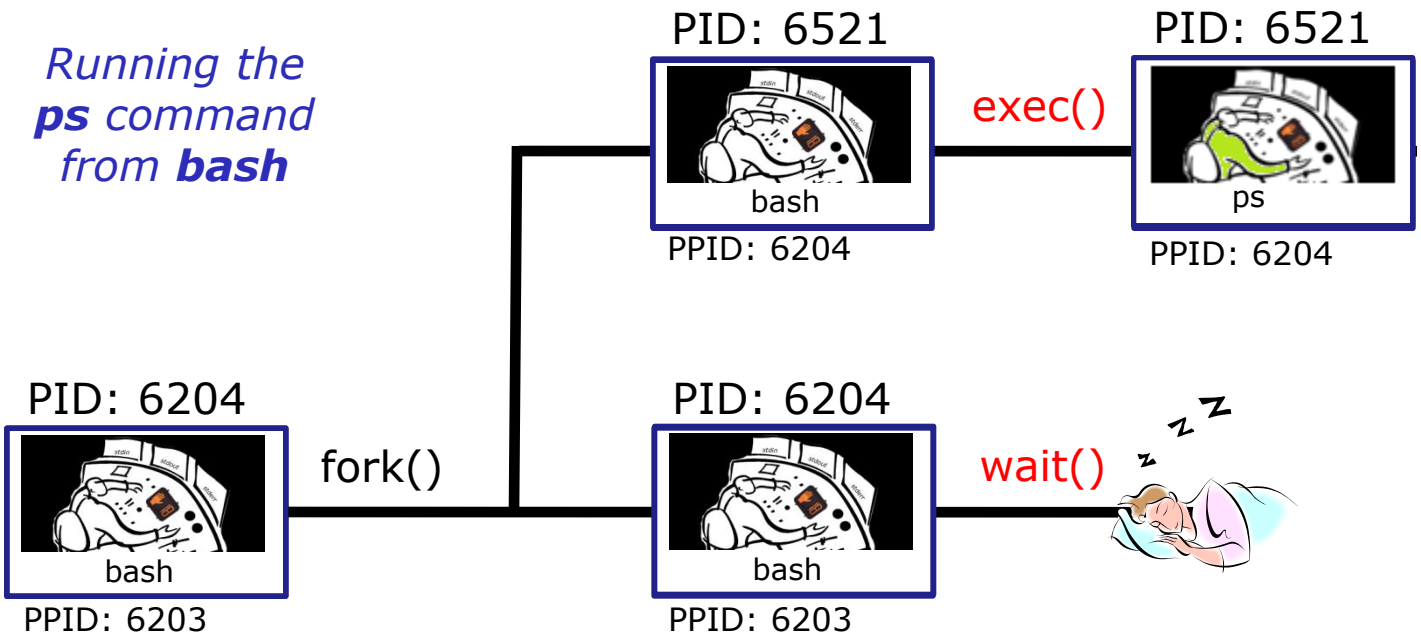
User ID

Running or sleeping

Size in 1K blocks

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.

Parent and child process practice

- Type **bash**
- Type **bash** again
- Type **bash** again
- Type **ps –l**
- Who is the parent of ps? Who is the parent of the parent of ps?
- Type **ps –ef**
- Track your family history as far back as you can go. Who is the most distant grandparent of ps?

Review of Signals

Signals

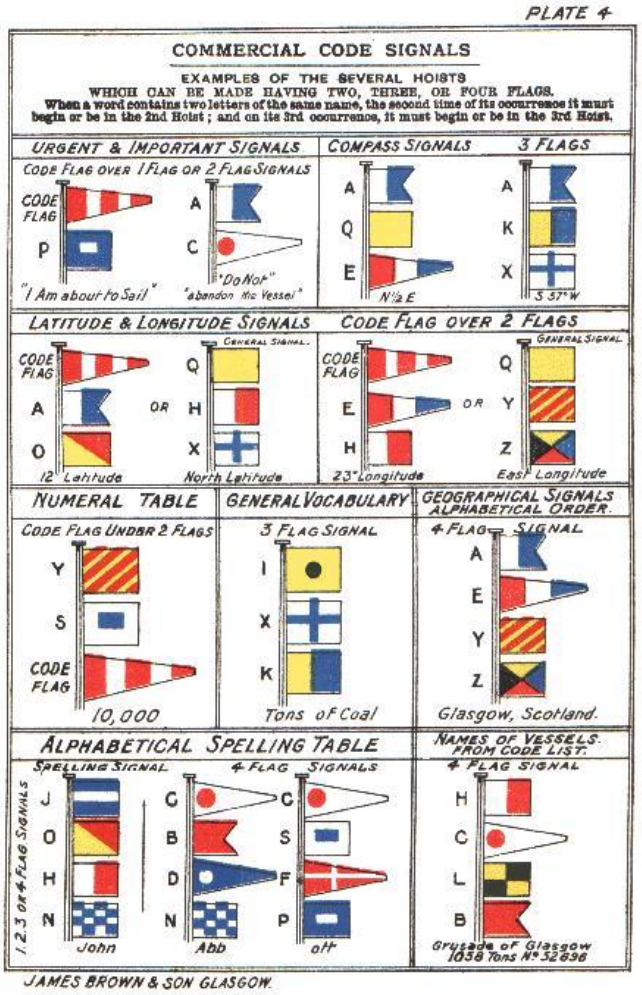
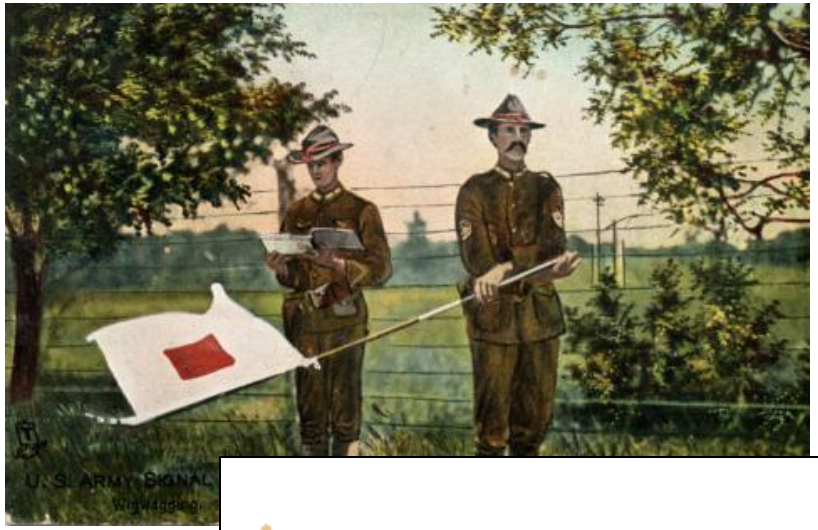
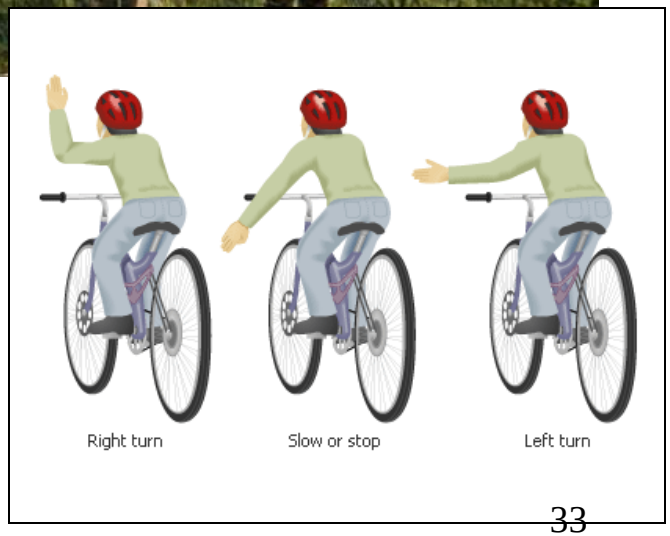


PLATE 4
COMMERCIAL CODE SIGNALS
EXAMPLES OF THE SEVERAL HOISTS
WHICH CAN BE MADE HAVING TWO, THREE, OR FOUR FLAGS.
When a word contains two letters of the same name, the second time of its occurrence it must begin or be in the 2nd Hoist; and on its 3rd occurrence, it must begin or be in the 3rd Hoist.
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
A
Q
E
N½E
3 FLAGS
A
K
X
S 57° W
LATITUDE & LONGITUDE SIGNALS
CODE FLAG
A
O
12° Latitude
OR
GENERAL SIGNAL.
Q
H
X
North Latitude
CODE FLAG OVER 2 FLAGS
CODE FLAG
E
H
23° Longitude
OR
GENERAL SIGNAL
Q
Y
Z
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
1, 2, 3 OR 4 FLAG SIGNALS
J
O
H
N
John
4 FLAG SIGNALS
C
B
D
N
Abb
C
S
F
P
off
NAMES OF VESSELS FROM CODE LIST.
4 FLAG SIGNAL
H
C
L
B
Gruzade of Glasgow
1058 Tons No 32896
JAMES BROWN & SON GLASGOW.


U. S. ARMY SIGNAL


18


Right turn
Slow or stop
Left turn

A Process at Work

A **process**
- reads from **stdin**
- writes to **stdout**
- puts error messages in **stderr**

- and may get interrupted from time to time by a **signal**

*A **process** is a **program** that has been loaded into memory and is either running (executing instructions) or waiting to run*

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.

How are signals sent?

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:

kill command

Using the kill command: **$ kill -# PID**

- Where # is the signal number and PID is the process id.
- if no number is specified, SIGTERM (-15) is sent.

Using special keystrokes

- limited to just a few signals
- limited to when you have control of the keyboard

Use kill –l to see all signals

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 (can't be caught or ignored) (POSIX)	
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)	

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

Signals

The result of sending a signal to a process:

- be ignored
- default action (die)
- execute some predefined function

Review of kill command usage

Jim's app script

Signal 2's (Ctrl-C) are ignored

rsimms@opus:/home/cis90/depot

```
#!/bin/sh
#
# app - script to demostrate use of signals
#
# Usage:  run app with no options or parameters
#
# Send signals to it with keystrokes or kill command
#
# Notes:
# stty -echo stop the display of characters typed
# stty echo makes typed characters visible again
# stty susp ^Z sets suspend keystroke to Ctlr-Z (to stop forground processes)
# stty susp @ sets suspend character to @ (to stop foreground processes)
#
trap '' 2  #Ignore SIGINT
trap 'echo -n quit it!' 3 #Handle SIGQUIT
trap 'stty echo susp ^Z;echo ee; echo cleanup;exit' 15 #Handle SIGTERM
clear
banner testing
stty -echo susp @
sleep 1
echo  one
sleep 1
echo two
sleep 1
echo -n thr
while :
do sleep 1
done
~
                                                    13,1          All
```

Jim's app script

Signal 3's (Cntrl-\) print quit it message

rsimms@opus:/home/cis90/depot

```
#!/bin/sh
#
# app - script to demostrate use of signals
#
# Usage:  run app with no options or parameters
#
# Send signals to it with keystrokes or kill command
#
# Notes:
# stty -echo stop the display of characters typed
# stty echo makes typed characters visible again
# stty susp ^Z sets suspend keystroke to Ctlr-Z (to stop forground processes)
# stty susp @ sets suspend character to @ (to stop foreground processes)
#
trap '' 2  #Ignore SIGINT
trap 'echo -n quit it!' 3 #Handle SIGQUIT
trap 'stty echo susp ^Z;echo ee; echo cleanup;exit' 15 #Handle SIGTERM
clear
banner testing
stty -echo susp @
sleep 1
echo  one
sleep 1
echo two
sleep 1
echo -n thr
while :
do sleep 1
done
~
                                                    13,1          42 All
```

Jim's app script

Signal 15's close gracefully

rsimms@opus:/home/cis90/depot

```
#!/bin/sh
#
# app - script to demostrate use of signals
#
# Usage:  run app with no options or parameters
#
# Send signals to it with keystrokes or kill command
#
# Notes:
# stty -echo stop the display of characters typed
# stty echo makes typed characters visible again
# stty susp ^Z sets suspend keystroke to Ctlr-Z (to stop forground processes)
# stty susp @ sets suspend character to @ (to stop foreground processes)
#
trap '' 2  #Ignore SIGINT
trap 'echo -n quit it!' 3 #Handle SIGQUIT
trap 'stty echo susp ^Z;echo ee; echo cleanup;exit' 15 #Handle SIGTERM
clear
banner testing
stty -echo susp @
sleep 1
echo  one
sleep 1
echo two
sleep 1
echo -n thr
while :
do sleep 1
done
~
                                                    13,1          All
```

Jim's app script

Redefines the keystroke to suspend a job and move it to the background

rsimms@opus:/home/cis90/depot

```
#!/bin/sh
#
# app - script to demostrate use of signals
#
# Usage:  run app with no options or parameters
#
# Send signals to it with keystrokes or kill command
#
# Notes:
# stty -echo stop the display of characters typed
# stty echo makes typed characters visible again
# stty susp ^Z sets suspend keystroke to Ctlr-Z (to stop forground processes)
# stty susp @ sets suspend character to @ (to stop foreground processes)
#
trap '' 2  #Ignore SIGINT
trap 'echo -n quit it!' 3 #Handle SIGQUIT
trap 'stty echo susp ^Z;echo ee; echo cleanup;exit' 15 #Handle SIGTERM
clear
banner testing
stty -echo susp @
sleep 1
echo  one
sleep 1
echo two
sleep 1
echo -n thr
while :
do sleep 1
done
~
                                                    13,1          44 All
```

Jim's app script

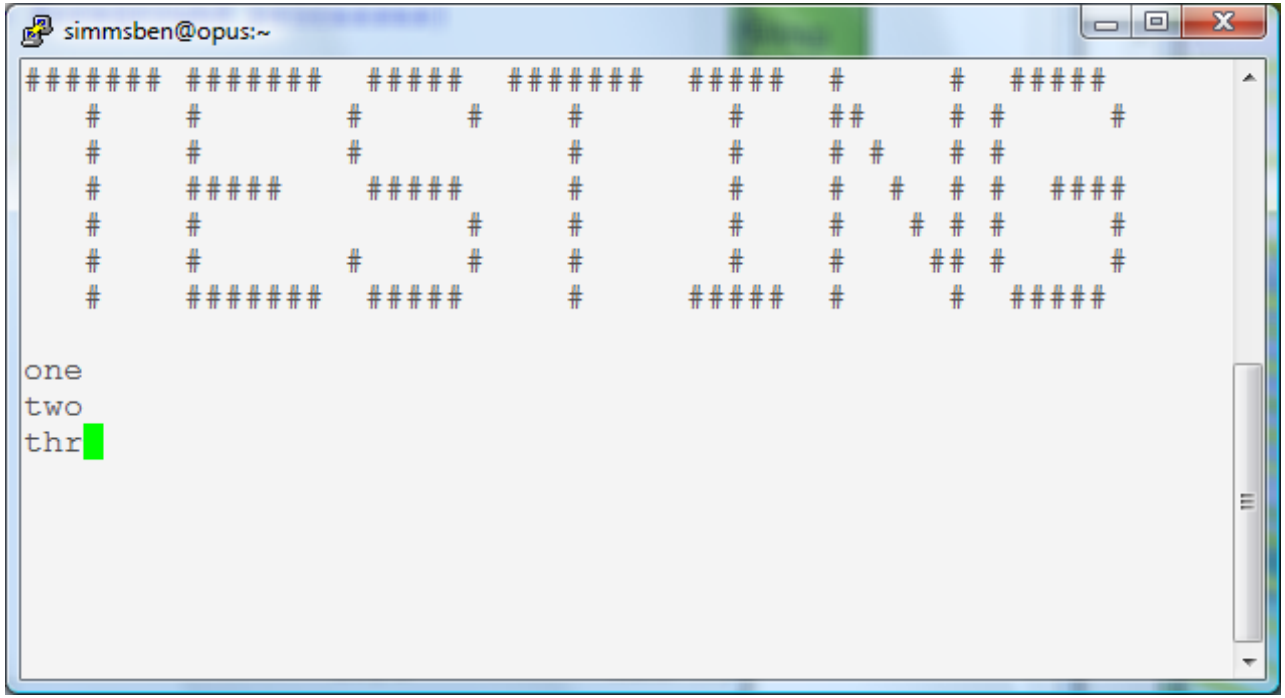
rsimms@opus:/home/cis90/depot

```
#!/bin/sh
#
# app - script to demostrate use of signals
#
# Usage:  run app with no options or parameters
#
# Send signals to it with keystrokes or kill command
#
# Notes:
# stty -echo stop the display of characters typed
# stty echo makes typed characters visible again
# stty susp ^Z sets suspend keystroke to Ctlr-Z (to stop forground processes)
# stty susp @ sets suspend character to @ (to stop foreground processes)
#
trap '' 2  #Ignore SIGINT
trap 'echo -n quit it!' 3 #Handle SIGQUIT
trap 'stty echo susp ^Z;echo ee; echo cleanup;exit' 15 #Handle SIGTERM
clear
banner testing
stty -echo susp @
sleep 1
echo  one
sleep 1
echo two
sleep 1
echo -n thr
while :
do sleep 1
done
~
                                                    13,1          45 All
```

Endless loop

Signals

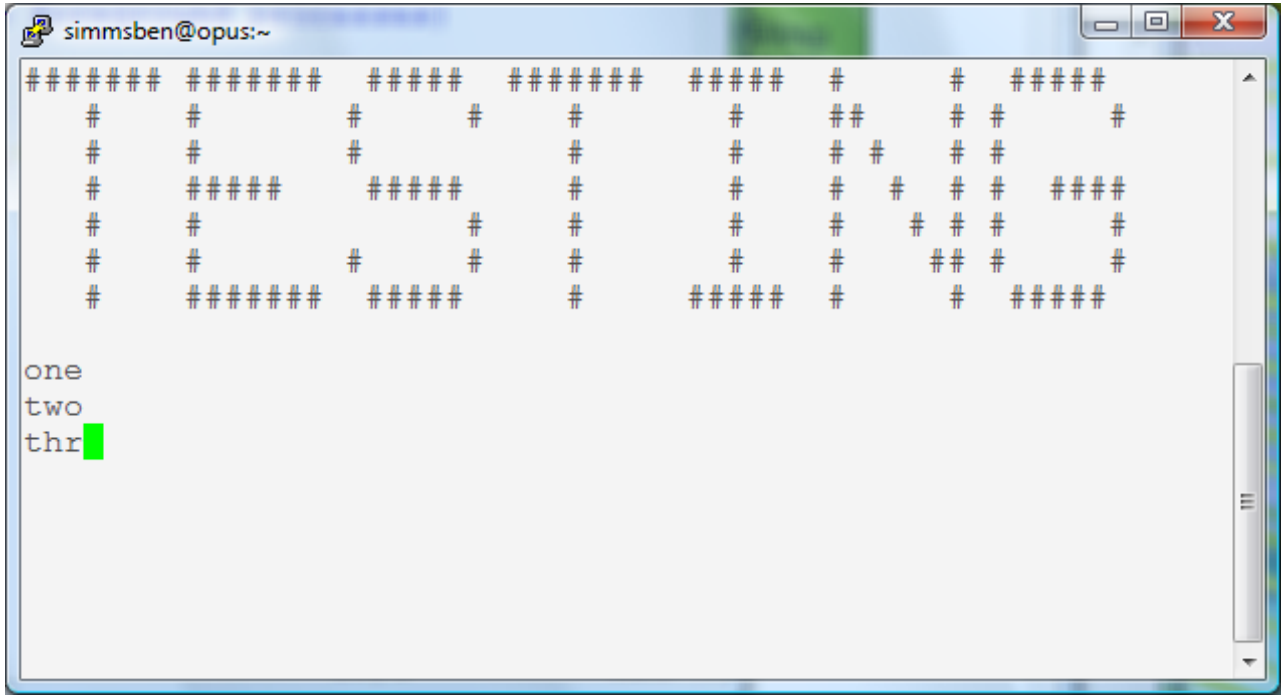
Benji runs app



Benji logs in and runs app … uh oh, its stuck !

Signals

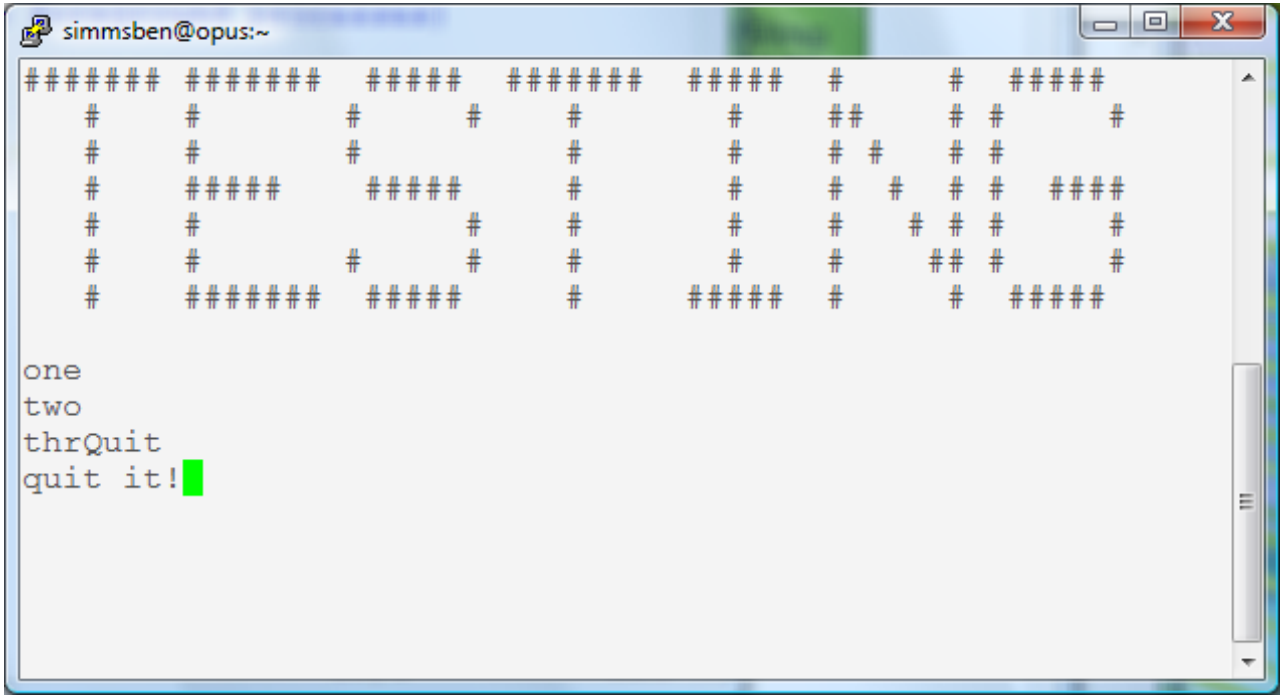
Benji runs app



Benji tries using the keyboard to send a SIGINT/2 using Ctrl-C but nothing happens (because app is ignoring SIGINT)

Signals

Benji runs app



Benji tries using the keyboard to send a SIGQUIT/3 using Ctrl-\ but app reacts by saying "quit it"

Signals

Benji runs app

```
roddyduk@opus:~
/home/cis90/roddyduk $ ps -u simmsben
  PID TTY          TIME CMD
 6657 ?        00:00:00 sshd
 6658 pts/1    00:00:00 bash
 7033 ?        00:00:00 sshd
 7034 pts/2    00:00:00 bash
 7065 pts/2    00:00:00 app
 7579 pts/2    00:00:00 sleep
/home/cis90/roddyduk $ kill 7065
-bash: kill: (7065) - Operation not permitted
/home/cis90/roddyduk $
```

Benji asks his friend Duke to kill off his stalled app process. Duke uses ps to look it up but does not have permission to kill it off

Signals

Benji runs app

simmsben@opus:~

```
one
two
thrQuit
quit it!
```

simmsben@opus:~

```
/home/cis90/simmsben $ ps -u simmsben
  PID TTY          TIME CMD
 6657 ?        00:00:00 sshd
 6658 pts/1    00:00:00 bash
 7033 ?        00:00:00 sshd
 7034 pts/2    00:00:00 bash
 7065 pts/2    00:00:00 app
 7843 pts/2    00:00:00 sleep
 7844 pts/1    00:00:00 ps
/home/cis90/simmsben $ kill -2 7065
/home/cis90/simmsben $
```

Benji logs into another Putty session and sends a SIGINT/2 using the kill command … but nothing happens

Signals

Benji runs app

simmsben@opus:~

```
one
two
thrQuit
quit it!quit it!quit it!
```

```
/home/cis90/simmsben $ kill -3 7065
/home/cis90/simmsben $ kill -3 7065
/home/cis90/simmsben $
```

Benji ups the ante and sends several SIGQUIT/3 signals but the app process shrugs them off with "quit it!"

Signals

Benji runs app

```
simmsben@opus:~
/home/cis90/simmsben $ kill -3 7065
/home/cis90/simmsben $ kill -15 7065
/home/cis90/simmsben $
```

```
one
two
thrQuit
quit it!quit it!quit it!ee
cleanup
/home/cis90/simmsben $
```

Benji decides to send a SIGTERM this time and the app process finishes, cleans up and exits

Signals

Benji runs app

simmsben@opus:~

```
/home/cis90/simmsben $ ps -u simmsben
  PID TTY          TIME CMD
 6657 ?        00:00:00 sshd
 6658 pts/1    00:00:00 bash
 7033 ?        00:00:00 sshd
 7034 pts/2    00:00:00 bash
 8237 pts/2    00:00:00 app
 8279 pts/2    00:00:00 sleep
 8280 pts/1    00:00:00 ps
/home/cis90/simmsben $
```

simmsben@opus:~

```
one
two
thr
```

The same thing happens again another day. This time Benji does not care what happens with app …

Signals

Benji runs app

simmsben@opus:~

```
one
two
thrKilled
/home/cis90/simmsben $
```

simmsben@opus:~

```
/home/cis90/simmsben $ ps -u simmsben
  PID TTY          TIME CMD
 6657 ?        00:00:00 sshd
 6658 pts/1    00:00:00 bash
 7033 ?        00:00:00 sshd
 7034 pts/2    00:00:00 bash
 8237 pts/2    00:00:00 app
 8279 pts/2    00:00:00 sleep
 8280 pts/1    00:00:00 ps
/home/cis90/simmsben $ kill -9 8237
/home/cis90/simmsben $
```

So he sends a SIGKILL/9 this time … and app never even sees it coming … poof … app is gone

Using &

to run a command in the background

& Append to a command to run it in the background

Example 1

```
/home/cis90/simmsben $ find / -user 1200 2> duh | sort > huh
```

No prompt

For long running commands or scripts you must wait for the command to finish before you type more commands

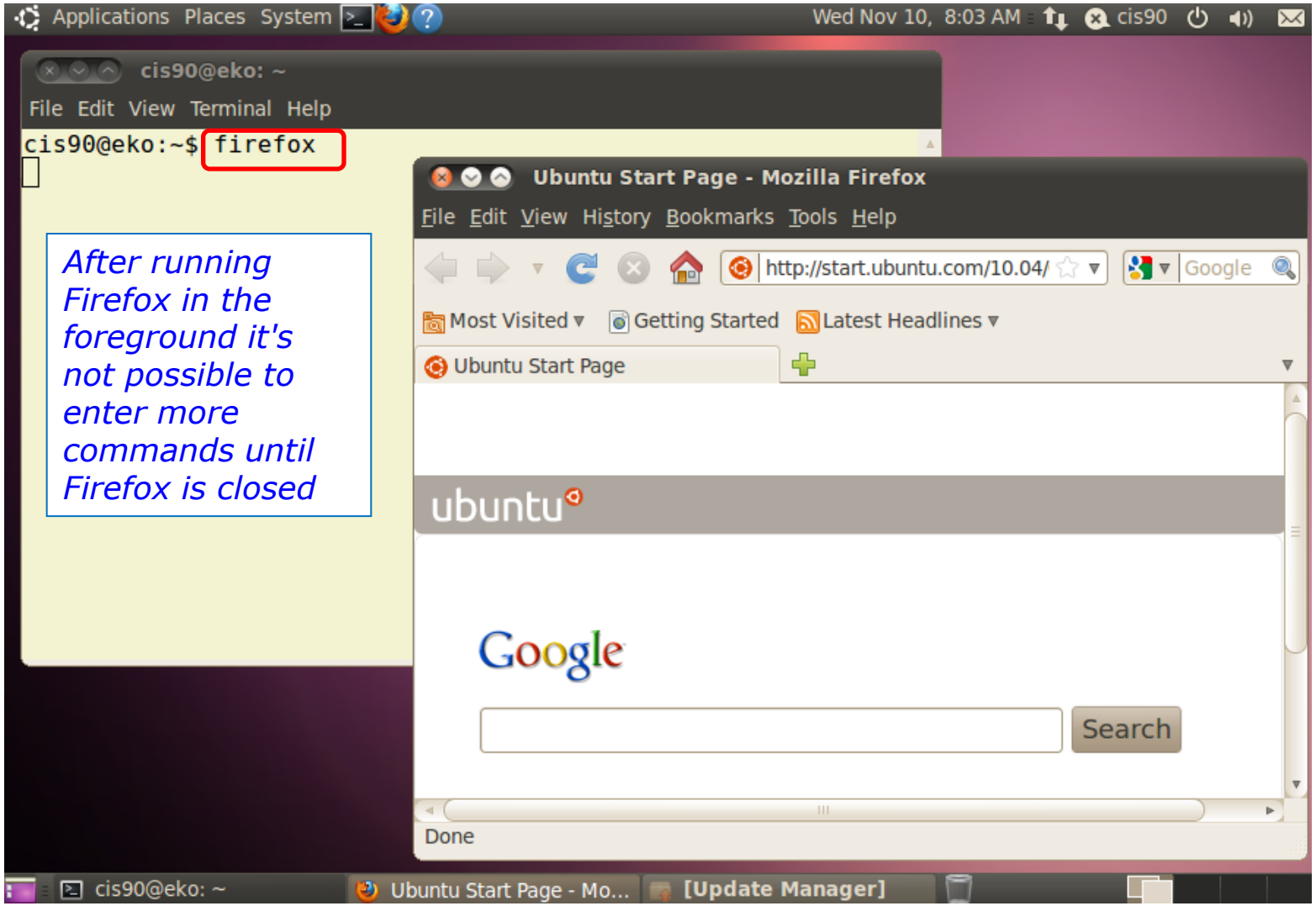
Example 2

```
/home/cis90/simmsben $ find / -user 1200 2> duh | sort > huh &
[1] 11601
/home/cis90/simmsben $ date
Tue Nov  9 14:38:35 PST 2010
```

Hit enter to get the prompt and continue working while the find command runs in the background

Job Control

Using **&** to run a command in the background



After running Firefox in the foreground it's not possible to enter more commands until Firefox is closed

Job Control

Using **&** to run a command in the background

```
cis90@eko:~$ firefox
cis90@eko:~$ firefox &
[1] 1465
cis90@eko:~$ ps
  PID TTY          TIME CMD
 1370 pts/0    00:00:00 bash
 1465 pts/0    00:00:00 firefox
 1470 pts/0    00:00:00 run-moz
 1474 pts/0    00:00:01 firefox
 1489 pts/0    00:00:00 ps
cis90@eko:~$
```

After running Firefox in the background, it is still possible to enter more commands.

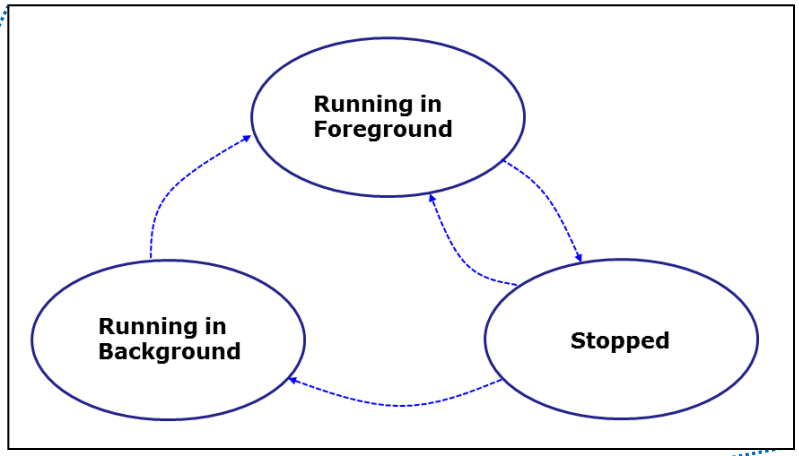
Job Control
A feature of the bash shell

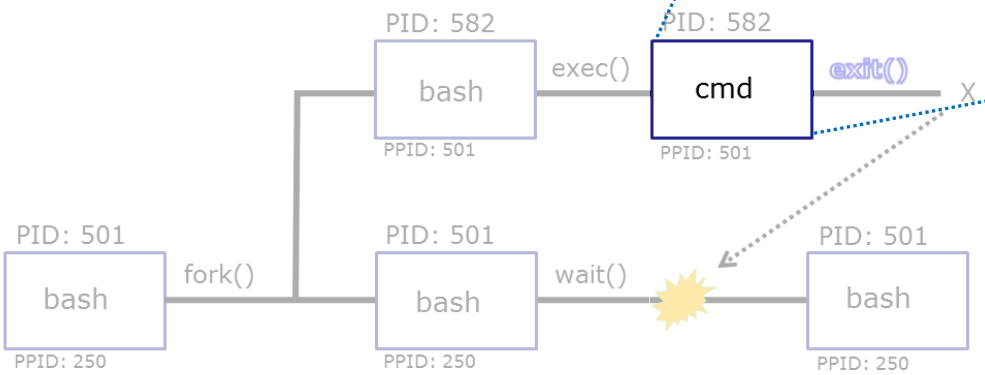
&	Append to a command to run it in the background
bg	Resumes a suspended job in the background
fg	Brings the most recent background process to the foreground
jobs	Lists all background jobs

*Use **jobs**, **bg**, **fg** to list and resume jobs in the foreground or background*

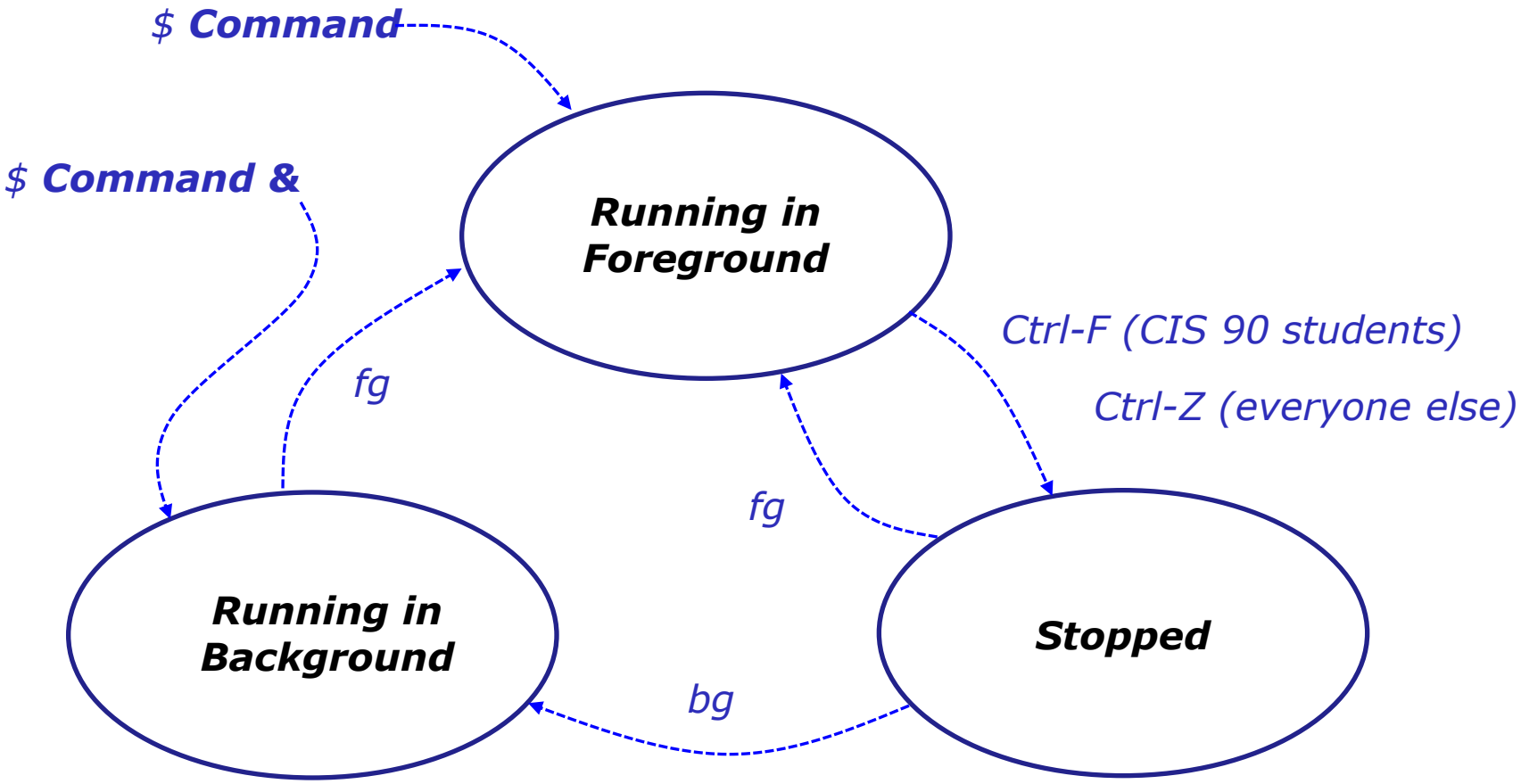
Job Control
A feature of the bash shell

*When a process is **running** (status=R) the user can **stop** it (status=T) and choose whether it runs in the **background** or **foreground***





Job Control
A feature of the bash shell



*Use the **jobs** command to view stopped and background jobs*

Job Control

Find out with keystroke combination is configured to suspend a process

```
/home/cis90ol/simmsben $ stty -a
speed 38400 baud; rows 24; columns 80; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^F; rprnt = ^R;
werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 -hupcl -cstopb cread -clocal -crtscts -cdtrdsr
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff
-iuclc -ixany -imaxbel -iutf8
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt
echoctl echoke
/home/cis90ol/simmsben $
```

In this case it is Ctrl-F that will be used to suspend a process

How is yours configured?

Job Control

Managing jobs

```
/home/cis90ol/simmsben $ sleep 120
```
Ctrl-Z or *Ctrl-F* *(to suspend process)*
```
[1]+  Stopped                 sleep 120
/home/cis90ol/simmsben $ sleep 110
```
Ctrl-Z or *Ctrl-F* *(to suspend process)*
```
[2]+  Stopped                 sleep 110
/home/cis90ol/simmsben $ sleep 100
```
Ctrl-Z or *Ctrl-F* *(to suspend process)*
```
[3]+  Stopped                 sleep 100

/home/cis90ol/simmsben $ jobs
[1]   Stopped                 sleep 120
[2]-  Stopped                 sleep 110
[3]+  Stopped                 sleep 100
```

Lets start up 3 sleep commands and suspend each of them.

Note: The sleep command is a simple way to run a command that will take awhile to finish.

***sleep 120** will last 120 seconds before it is finished.*

Job Control

Managing jobs

```
/home/cis90ol/simmsben $ jobs
[1]   Stopped                 sleep 120
[2]-  Stopped                 sleep 110
[3]+  Stopped                 sleep 100


/home/cis90ol/simmsben $ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1082  5364  5363  0  75   0 -  1168 wait   pts/2    00:00:00 bash
0 T  1082  5452  5364  0  75   0 -   929 finish pts/2    00:00:00 sleep
0 T  1082  5453  5364  0  75   0 -   929 finish pts/2    00:00:00 sleep
0 T  1082  5454  5364  0  75   0 -   929 finish pts/2    00:00:00 sleep
0 R  1082  5459  5364  0  77   0 -  1054 -      pts/2    00:00:00 ps
```

Note, all three processes are sTopped

Job Control

Managing jobs

```
/home/cis90ol/simmsben $ bg 2
[2]- sleep 110 &
/home/cis90ol/simmsben $ jobs
[1]-  Stopped                 sleep 120
[2]   Running                 sleep 110 &
[3]+  Stopped                 sleep 100
```

*Jobs can be resumed in the background using **bg***

```
/home/cis90ol/simmsben $ bg 1
[1]- sleep 120 &
/home/cis90ol/simmsben $ jobs
[1]   Running                 sleep 120 &
[2]-  Running                 sleep 110 &
[3]+  Stopped                 sleep 100
```

*or in the foreground using **fg***

```
/home/cis90ol/simmsben $ fg 3
sleep 100
```

At this point we lose control of the keyboard again until sleep 100 is finished

Job Control
Managing jobs

```
/home/cis90ol/simmsben $ jobs
[1]-  Done
sleep 120
[2]+  Done
sleep 110
```

Background jobs are all done!

Review of Load Balancing

Load Balancing

The **at** command:

- reads from stdin for a list of commands to run
- runs those commands at the specified time
- Any output from those commands will be emailed
- Use **atq** and **atrm** to manage scheduled commands

*Use **at** to schedule commands to run in the future*

Load Balancing

Managing queued jobs

```
at now + 5 minutes
at now + 1 hour
at 7:58AM
at 7:47PM 5/5/2012
at teatime
```

Ways to specify future times

Load Balancing

Managing queued jobs

```
/home/cis90/simben $ atq
25      2011-11-12 14:09 a simben90
28      2011-12-12 03:00 a simben90
27      2011-11-19 12:10 a simben90
26      2011-11-12 16:00 a simben90
24      2011-11-12 12:14 a simben90
```

*The **atq** command lists jobs queued to run in the future*

```
/home/cis90/simben $ atrm24
/home/cis90/simben $ atq
25      2011-11-12 14:09 a simben90
28      2011-12-12 03:00 a simben90
27      2011-11-19 12:10 a simben90
26      2011-11-12 16:00 a simben90
```

*The **atrm** command is used to remove jobs from the queue*

```
/home/cis90/simben $ jobs
```

*Note: The **jobs** command lists processes running or suspended in the background and is NOT used for **at** commands.*

Load Balancing

What the heck will this do?

```
/home/cis90/simben $ at 7:18AM 4/23/2012
at> echo "Wake up Sarah" | mail -s "Reminder" simben90
at> <EOT>
job 373 at 2012-04-23 07:18
/home/cis90/simben $
```

Type Ctrl-D to end adding more commands

Load Balancing

What the heck will this do? Try it!

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

/home/cis90/simben $ echo 'echo " " > /dev/pts/0' > hola
/home/cis90/simben $ echo 'banner "hola" > /dev/pts/0' >> hola
/home/cis90/simben $ cat hola
echo " " > /dev/pts/0
banner "hola" > /dev/pts/0

/home/cis90/simben $ at now + 2 minutes < hola
job 375 at 2012-04-23 07:31
/home/cis90/simben $ atq
375     2012-04-23 07:31 a simben90
/home/cis90/simben $
```

text editors

There are lots of text editors …

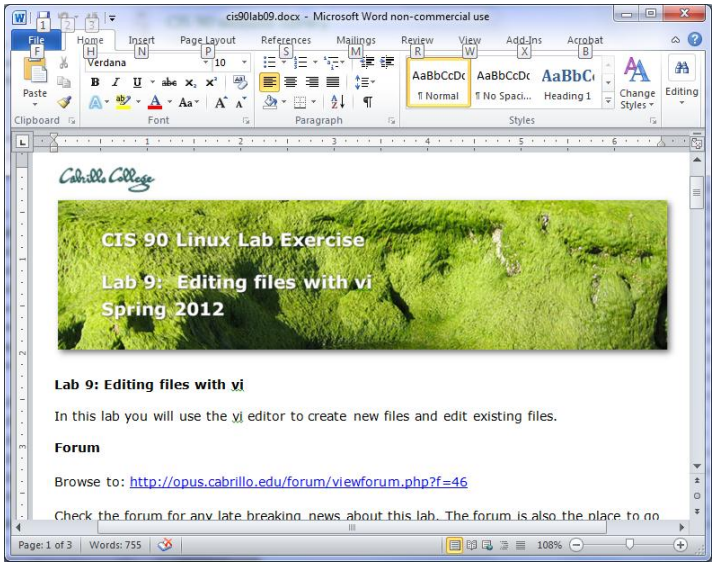
Windows
- notepad
- notepad++
- textpad

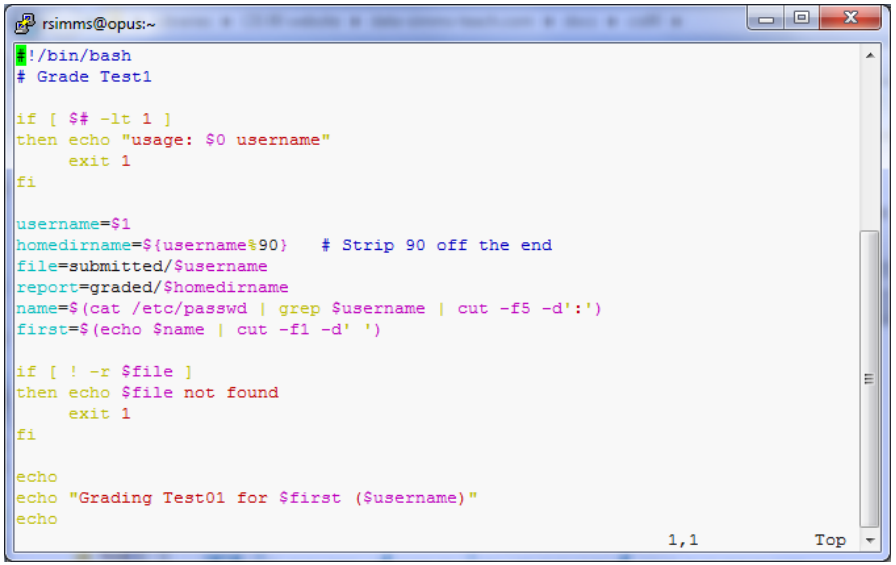
Mac
- TextWrangler

Linux
- gedit
- emacs
- nano
- vi

Text editors and word processors are different!

- *Word processors are used by many different people to create documents containing text and graphics.*
- *Text editors are used by programmers to develop software and web designers to create web sites.*





Word processors *allow a rich set of formatting (fonts, sizes, styles, color) and graphics to be added to documents.*

Text editors *use color to show the language syntax*

On Opus we are actually running VIM

```
/home/cis90/simben $ type -a vi
vi is aliased to `vim'
vi is /bin/vi
/home/cis90/simben $ type vim
vim is hashed (/usr/bin/vim)
```

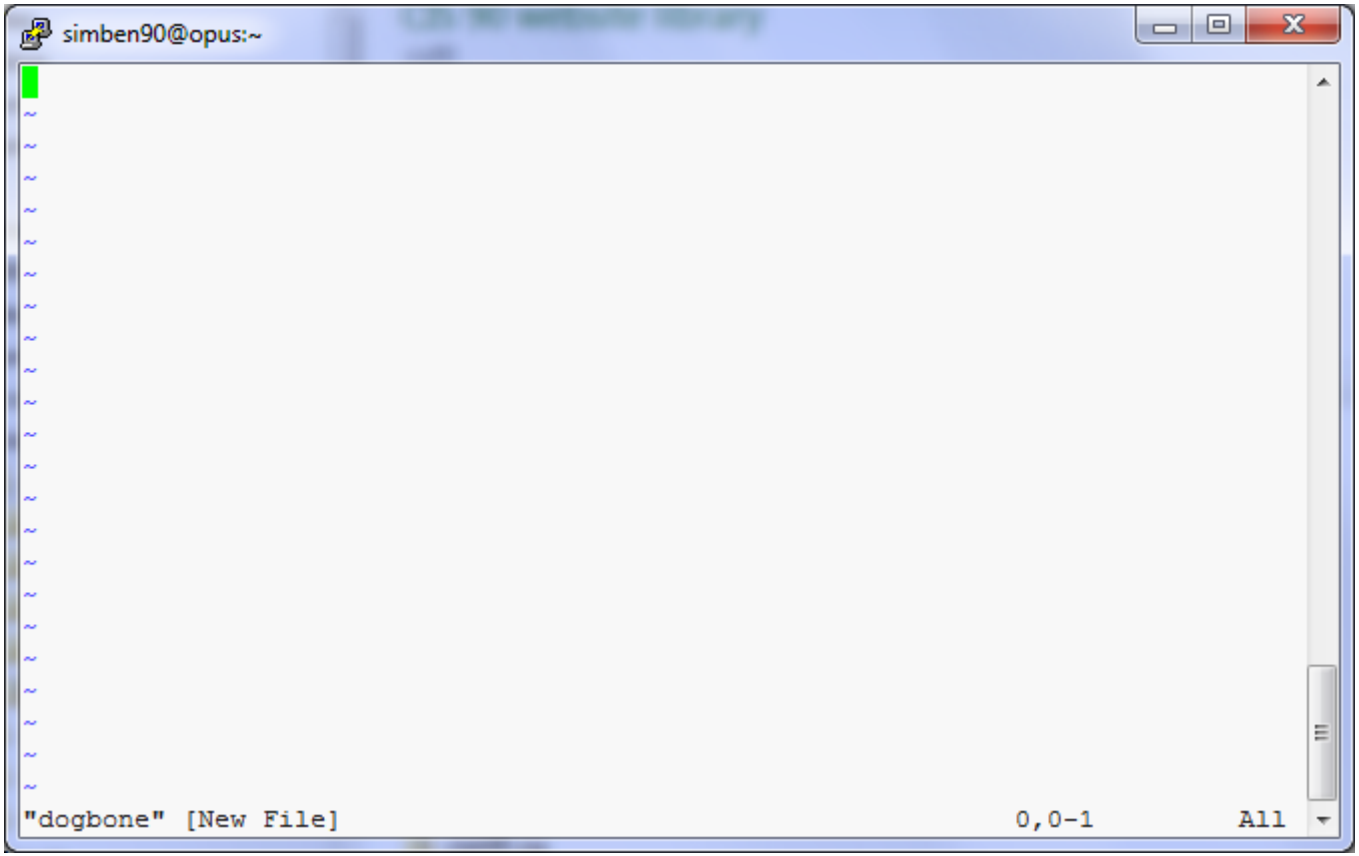
vim is an enhanced version of vi

vi 101

```
/home/cis90/simben $
/home/cis90/simben $ vi dogbone
```

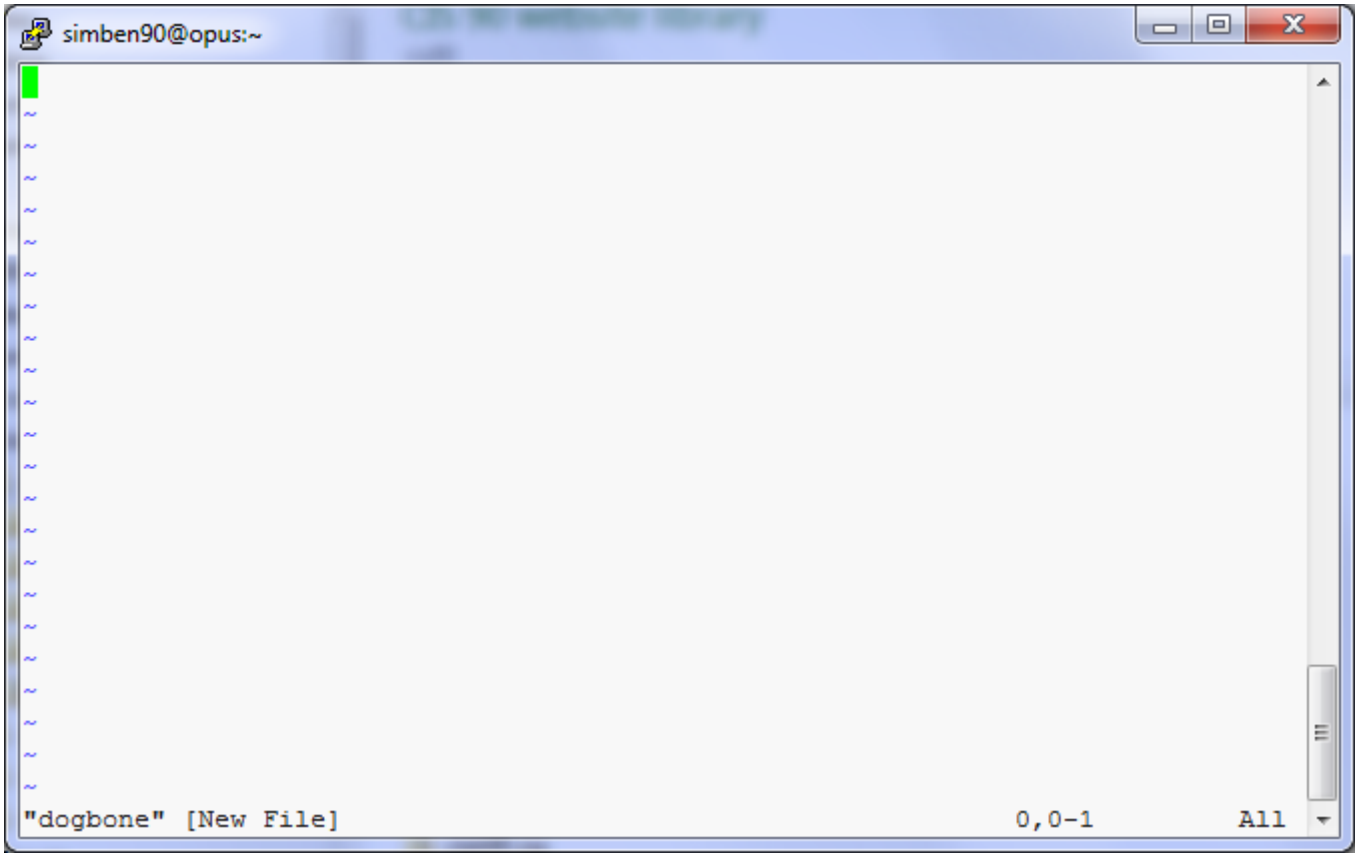
Type this

See this …



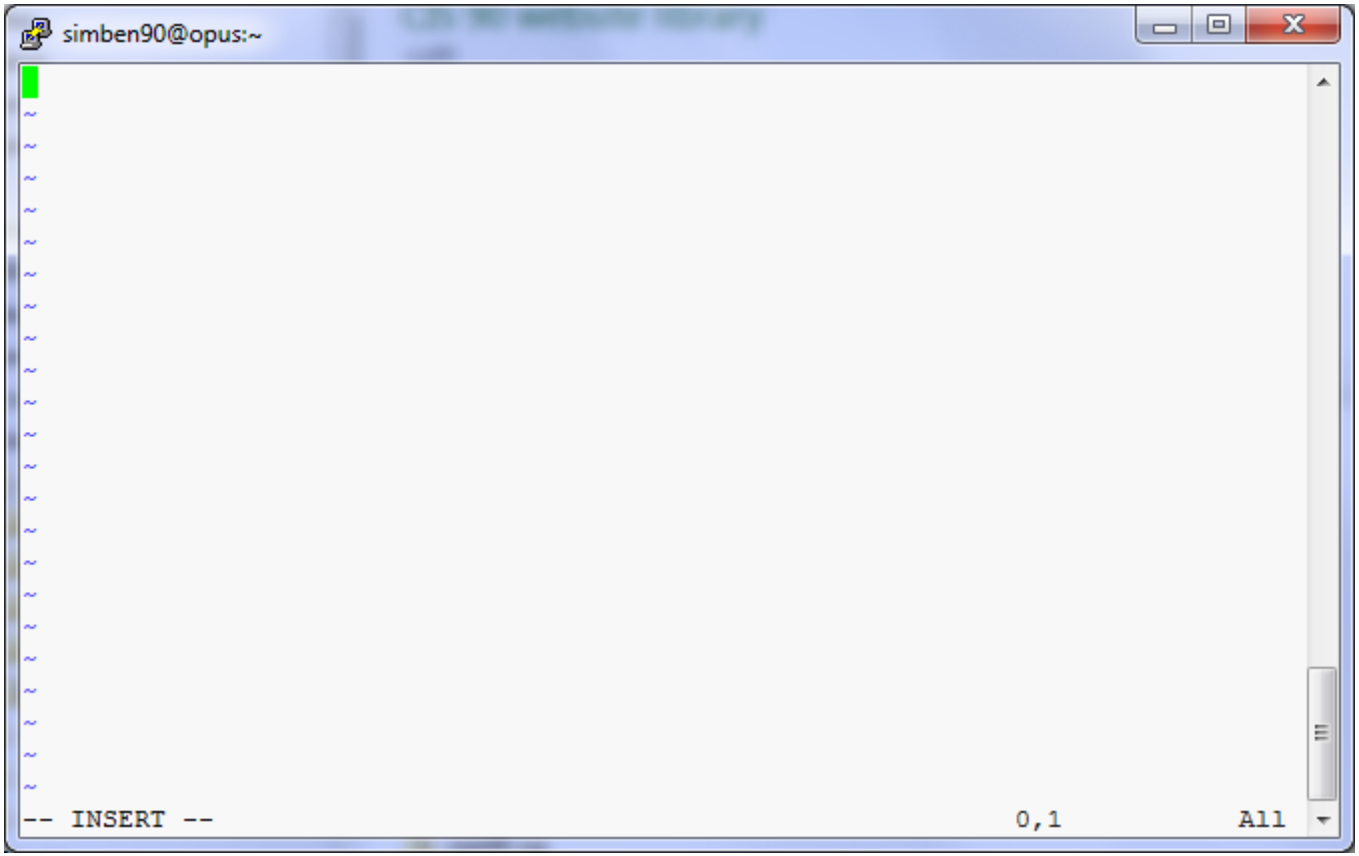
Take your hands OFF THE MOUSE – don't use it in vi!

Tap the letter i key (for insert)



Keep your hands OFF THE MOUSE – don't use it in vi!

See this …



Keep your hands OFF THE MOUSE – don't use it in vi!

Very carefully type these five lines

```
simben90@opus:~
echo -n "What is your name? "
read NAME
echo -n "What is your favorite bone? "
read BONE
echo "Hi $NAME, your favorite bone is $BONE"
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                        6,1           All
```

Keep your hands OFF THE MOUSE – don't use it in vi!

Have your neighbor check that your five lines are PERFECT

simben90@opus:~

```
echo -n "What is your name? "
read NAME
echo -n "What is your favorite bone? "
read BONE
echo "Hi $NAME, your favorite bone is $BONE"
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                    6,1           All
```

Keep your hands OFF THE MOUSE – don't use it in vi!

*Tap the **esc** key*

```
simben90@opus:~
echo -n "What is your name? "
read NAME
echo -n "What is your favorite bone? "
read BONE
echo "Hi $NAME, your favorite bone is $BONE"
~
~
~
~
~
~
~
~
~
~
~
~
~
~
                                        6,0-1          All
```

Keep your hands OFF THE MOUSE – don't use it in vi!

Type a **:**

simben90@opus:~

```
echo -n "What is your name? "
read NAME
echo -n "What is your favorite bone? "
read BONE
echo "Hi $NAME, your favorite bone is $BONE"
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:
```

Keep your hands OFF THE MOUSE – don't use it in vi!

*Type **wq***

```
simben90@opus:~
echo -n "What is your name? "
read NAME
echo -n "What is your favorite bone? "
read BONE
echo "Hi $NAME, your favorite bone is $BONE"
~
~
~
~
~
~
~
~
~
~
~
~
~
:wq
```

Keep your hands OFF THE MOUSE – don't use it in vi!

Tap the enter key

```
/home/cis90/simben $ vi dogbone
/home/cis90/simben $
```

Add execute permissions and try your new script

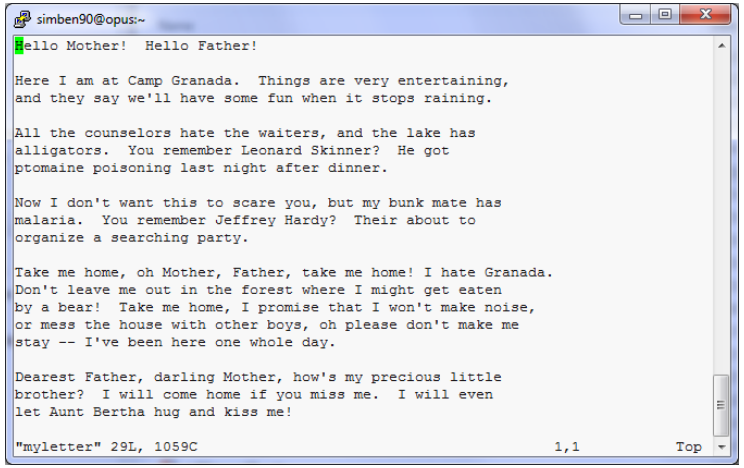
```
/home/cis90/simben $ chmod +x dogbone

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

vi

COMMAND mode
INSERT mode
command LINE mode

```
/home/cis90/simben $ cp letter myletter
/home/cis90/simben $ vi myletter
```

COMMAND mode



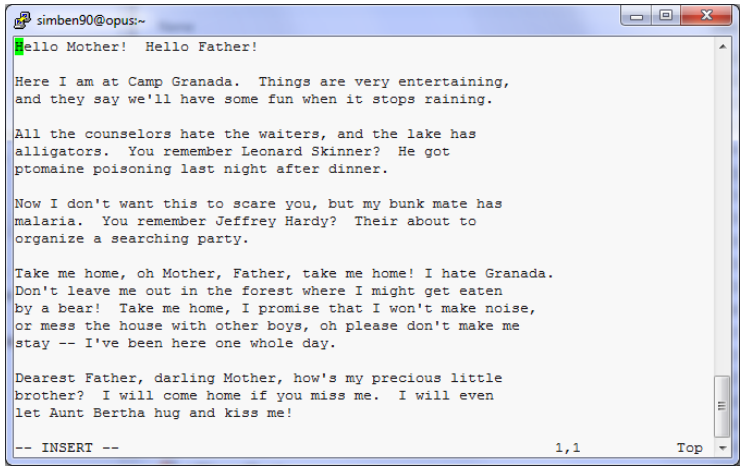
i

esc

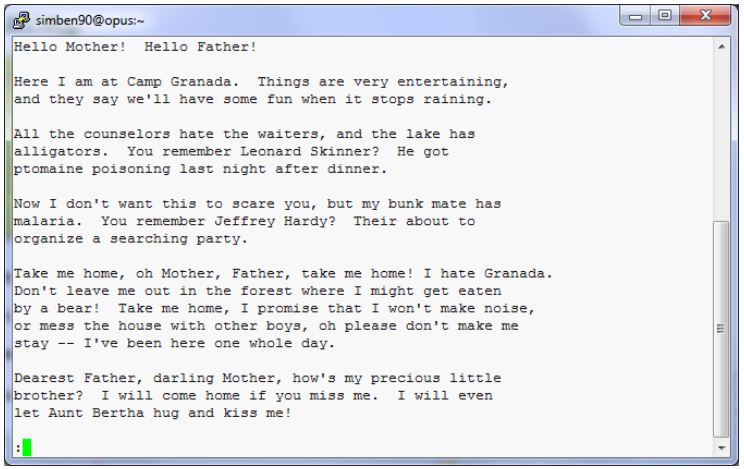
:

esc

INSERT mode



Command LINE mode



vi

Moving around in a file

Use in COMMAND mode

h moves the cursor one character to the left
j moves the cursor down one line
k moves the cursor up one line
l moves the cursor one character to the right

^d scrolls down 10 lines
^u scrolls up 10 lines
^f page forward one page
^b page back one page

Try typing a number in front of these commands and notice what happens

With vim (not vi) you can use arrow and page keys instead of these letter commands

vi

Moving around in a file

Use in COMMAND mode

Try typing a number in front of these commands and notice what happens

w moves the cursor one "word" forward
b moves the cursor one "word" back

0 (zero) moves the cursor to the beginning of the line
$ moves the cursor to the end of the line

G moves the cursor to the last line in the file
1G moves the cursor to the first line in the file
105G moves the cursor to line 105

vi
Saving and Quiting

Use in command LINE mode

:w writes any changes to the file you are editing (like Save)

:q quits vi if you have saved your changes
:q! quits vi even if you haven't saved changes

:wq writes and quits
:wq! writes and quits vi even if you haven't saved changes

vi

Reading in and Writing out files

Use in command LINE mode

:w *filename* saves your file to a new name (like Save As)
:w! *filename* saves your file to a new name overwriting any previous data

:r *filename* reads in the contents of *filename* starting from the cursor position

:e *filename* replaces the current content with the content from *filename*

vi
Entering INSERT mode

From command mode.

i Ready to insert characters immediately before the current cursor position
I Ready to insert characters at the start of the current line

a Ready to append characters immediately after the current cursor position
A Ready to append characters at the end of the current line

o Ready to input characters in a new line that opens up below the cursor
O Ready to input characters in a new line that opens up above the cursor

vi

Cut, Copy, Pasting Commands

Use in command mode

x Deletes the current character
r Replace the current character with the character you type next

dw Deletes the current word
dd Deletes the current line

D Deletes to the end of the line

yy Copies a line to the clipboard buffer
p Pastes whatever is in the clipboard buffer below the current cursor
P Pastes whatever is in the clipboard buffer above the current cursor

vi
Miscellaneous Useful Commands

Use in command mode.

^g Tells you the filename you are editing and what line your cursor is on

u Undoes the last command you executed
^r Undo the undo (redo)

. Repeats the last command you executed

/string Searches for the string of characters in the file
n Finds the next occurrence of the current search string looking down the file
N Finds the next occurrence of the current search string looking up the file

~ Changes the case of the current character

:%s /*string1*/*string2*/g replaces all string1 with string2 in the file

http://vim.wikia.com/wiki/Main_Page

Tips and tricks for VIM users

The Mug of vi

The Mug of Vi - Mozilla Firefox

http://nostarch.com/mug.htm

NO STARCH PRESS
"the finest in geek entertainment"™

Home | Catalog | Where to buy | About | Jobs | Media | Blog | Cart

The Mug of Vi
12 ounces
heavy-duty

$12.95

See mug text

Big Mug Label - Mozilla Firefox

http://nostarch.com/mug_big.htm

Click on the image to return to **Mug of Vi** main page.

THE MUG OF VI

FILE COMMANDS		DELETING /INSERTING TEXT			
vi *filename(s)*	edit a file or files	dw, dd, x	delete word, line, character	0	go to beginning of line (zero)
vi -r *filename*	retrieve saved file after crash	*n*dd, *n*x	delete *n* lines, *n* characters	), (	move to next, previous sentence
ZZ, :wq, :x	save and exit	x, X	delete character forward, backward	}, {	move to next, previous paragraph
:q, :q!	quit; quit without saving	D, d$	delete to end of line	w, b	move forward, back one word
:w, :w *fn*	save file, save file as *fn*	d*motion*	delete from cursor to *motion* ($, 0, etc.)	e	go to end of current or next word
:e *filename*	edit *filename*	:>, :<	indent, outdent line	CUT / COPY / PASTE	
:r *filename*	insert *filename*	S	replace text with blank line	*n*yy, *n*Y	copy *n* lines
:sh	drop to shell	o, O	insert new line below, above current line	y*w*, yy	copy word, line
:!*cmd*	run command *cmd*	u	undo last change	p, P	paste text after, before cursor
:r !*cmd*	execute *cmd* and insert output	.	repeat last change	a, i	insert text after, before cursor
SEARCH AND REPLACE		MOVING AROUND		A, I	insert text end, beginning of line
/*txt*, ?*txt*	find *txt* forward or backward	*n*G	move to line *n*	WICKED COOL STUFF	
/^*txt*	find next line that starts with *txt*	h, l, k, j	left, right, up, down one character	~	change case
n, N	repeat last search backward, forward	*n*b, *n*w	left or right *n* words	xp	transpose characters
R	replace text from current character	CTRL-B, F	back, forward one screen	J	combine current line with next
		CTRL-U, D	up, down one screen	m*p*	create a mark called *p*
		$, G	go to end of line, end of file	`*p*	return to *p*
				d`x, y`x	delete, copy text from mark to cursor
				:> *n*	indent *n* lines

NO STARCH PRESS

http://nostarch.com/mug.htm

/bin/mail and vi

```
/home/cis90/simmsben $ mail roddyduk
Subject: Good bones
Hey Duke,
I really appreciate thatbone you sent me last week.
Let me knwo if you want to go mark some fench posts
this weekend.
Later,
Ben
```

You are composing a message and you spot some typos …
CRUD … what can you do?

/bin/mail and vi

```
/home/cis90/simmsben $ mail roddyduk
Subject: Good bones
Hey Duke,
I really appreciate thatbone you sent me last week.
Let me knwo if you want to go mark some fench posts
this weekend.
Later,
Ben
~v
```

Well … you could try the ~v command

/bin/mail and vi

```
simmsben@opus:~
Hey Duke,
I really appreciate thatbone you sent me last week.
Let me knwo if you want to go mark some fench posts
this weekend.
Later,
Ben
~
~
~
~
~
~
~
~
~
~
~
~
"/tmp/ReqY2d2b" 6L, 141C
```

The message is loaded into vi where changes or additions can be made. :wq is used to save and quit vi

/bin/mail and vi

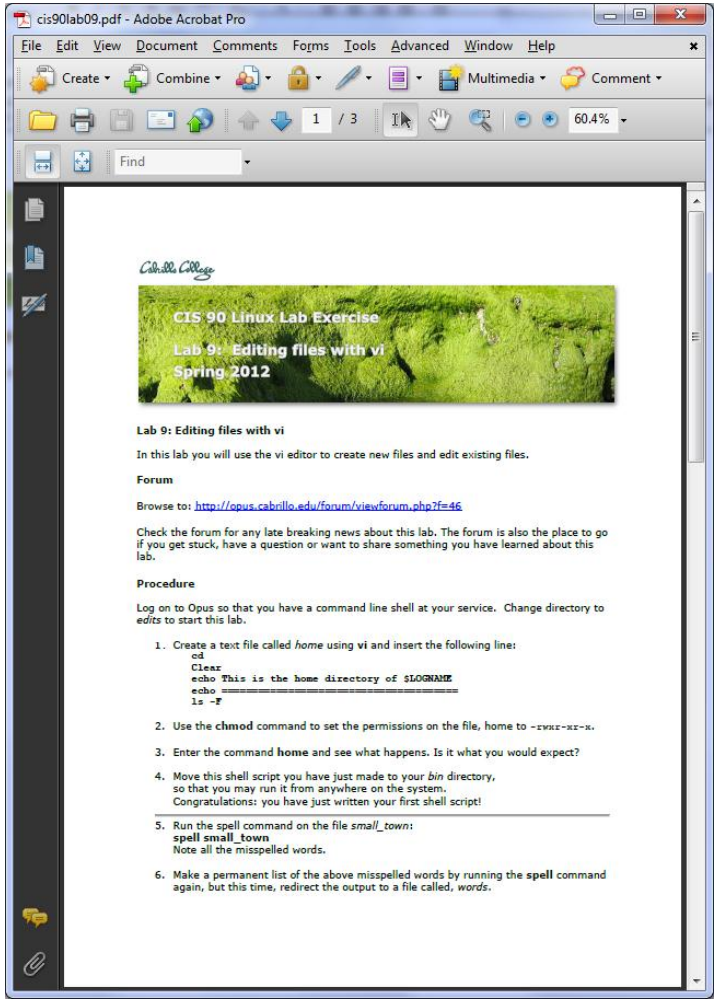
```
/home/cis90/simmsben $ mail roddyduk
Subject: Good bones
Hey Duke,
I really appreciate thatbone you sent me last week.
Let me knwo if you want to go mark some fench posts
this weekend.
Later,
Ben
~v
(continue)
.
Cc:
/home/cis90/simmsben $
```

The earlier text with typos is still showing, however the corrected version is what is actually sent.

/bin/mail and vi

```
/home/cis90/roddyduk $ mail
Mail version 8.1 6/6/93.  Type ? for help.
"/var/spool/mail/roddyduk": 1 message 1 unread
>U  1 simmsben@opus.cabril  Mon Nov 10 20:25  22/782   "Good bones"
& 1
Message 1:
From simmsben@opus.cabrillo.edu  Mon Nov 10 20:25:32 2008
Date: Mon, 10 Nov 2008 20:25:32 -0800
From: Benji Simms <simmsben@opus.cabrillo.edu>
To: roddyduk@opus.cabrillo.edu
Subject: Good bones

Hey Duke,
I really appreciate that bone you sent me last week.
Let me know if you want to go mark some fence posts
this weekend.
Later,
Ben

&
```

The message Duke reads has all the typos fixed.



Lab 9 will help you start building your vi skills!

Instructor: remember to mail students the tech file!

A Tangent on Spell

spell command

```
/home/cis90/roddyduk/edits $ cat text
Welcome to the CIS 90 class !!

/home/cis90/roddyduk/edits $ spell text
CIS
```

spell *command flags CIS as misspelled word.*

How can we add CIS to the dictionary?

spell command

```
/home/cis90/roddyduk/edits $ cat text
Welcome to the CIS 90 class !!
/home/cis90/roddyduk/edits $ spell text
CIS
```

How can we add CIS to the dictionary?

```
/home/cis90/roddyduk/edits $ man spell
No manual entry for spell
/home/cis90/roddyduk/edits $ type spell
spell is hashed (/usr/bin/spell)
/home/cis90/roddyduk/edits $ file usr/bin/spell
/usr/bin/spell: Bourne shell script text executable
/home/cis90/roddyduk/edits $ cat /usr/bin/spell
#!/bin/sh

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

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

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

Hmmm. No man page for spell ????????????

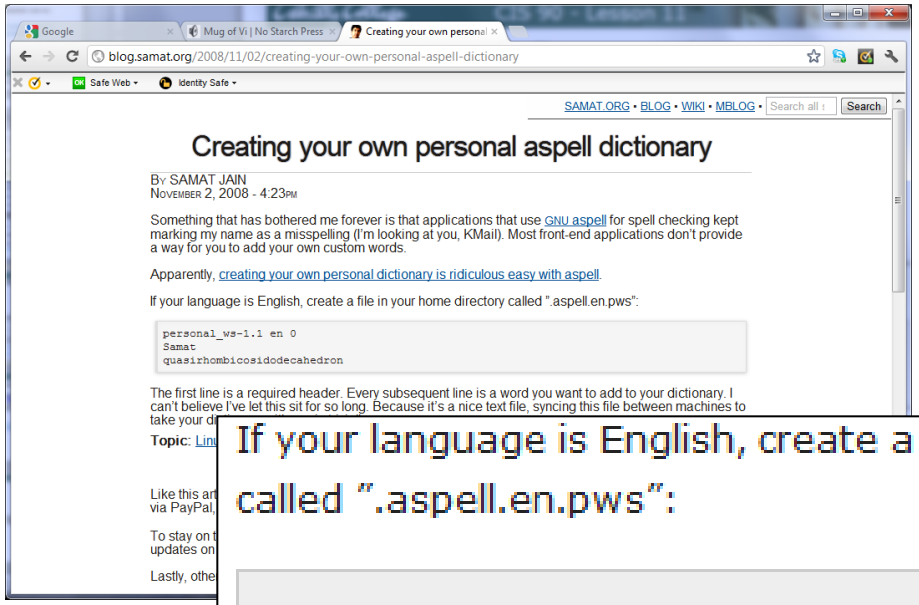
*OK, the actual command is **aspell***

spell command

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

NAME
       aspell - interactive spell checker

SYNOPSIS
       aspell [options] <command>

DESCRIPTION
       aspell  is  a utility that can function as an ispell -a replacement,
       as an independent spell checker, as  a  test  utility  to  test  out
       Aspell features, and as a utility for managing dictionaries.

COMMANDS
       <command> is one of:

       -?,help
              display the help message

       -c,check file
              to spell-check a file
```

There must be a way to add CIS …. but … lets try google

spell command



How to add words to your dictionary

If your language is English, create a file in your home directory called ".aspell.en.pws":

```
personal_ws-1.1 en 0
Samat
quasirhombicosidodecahedron
```

Googling "linux aspell personal dictionary" yields this page

Bingo! Thank you Samat Jain

spell command

```
/home/cis90/roddyduk/edits $ cd
/home/cis90/roddyduk $ echo "personal_ws-1.1 en 0" >  .aspell.en.pws
/home/cis90/roddyduk $ echo "CIS" >> .aspell.en.pws
/home/cis90/roddyduk $ cd edits/
/home/cis90/roddyduk/edits $ spell text
```

This is how you would add your own custom dictionary to be used with spell checks

Wrap up

New commands:

vi	Run vi editor

New Files and Directories:

na	na

Next Class

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

Lab 9
Five Posts

Quiz questions for next class:

- How do you send a SIGKILL to one of your own processes?
- What vi command is used to exit vi without saving any of the changes you made?
- What vi commands are used for copy and paste?

Backup

The **mystery** of Ctrl-Z vs Ctrl-F

Signals
Special keystrokes

```
/home/cis90/roddyduk $ stty -a
speed 38400 baud; rows 26; columns 78; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^F; rprnt = ^R;
werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
```

```
[rsimms@opus ~]$ stty -a
speed 38400 baud; rows 39; columns 84; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>; eol2 = <undef>;
swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W;
lnext = ^V; flush = ^O; min = 1; time = 0;
```

Why does the keystroke to send a Suspend (SIGTSTP or 20) signal differ between roddyduk (^F or Ctrl-F) and rsimms (^Z or Ctrl-Z)?

Job Control
A feature of the bash shell

Ctrl-Z or Ctrl-F (sends SIGTSTP 20 signal)

- Stops (suspends) a foreground process

```
[rsimms@opus ~]$ sleep 5

[1]+  Stopped                 sleep 5
```

Ctrl-Z is tapped which stops the sleep command

PID 7728 is stopped

```
[rsimms@opus ~]$ ps -l  -u rsimms
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
5 S   201  5368  5365  0  75   0 -  2460 -      ?        00:00:00 sshd
0 S   201  5369  5368  0  76   0 -  1165 wait   pts/0    00:00:00 bash
5 S   201  6203  6200  0  75   0 -  2491 -      ?        00:00:00 sshd
0 S   201  6204  6203  0  75   0 -  1165 -      pts/6    00:00:00 bash
0 T   201  7728  6204  0  75   0 -   926 finish pts/6    00:00:00 sleep
0 R   201  7730  5369  0  78   0 -  1062 -      pts/0    00:00:00 ps
[rsimms@opus ~]$
```

Job Control

A feature of the bash shell

bg command

- Resumes a suspended job in the background

```
[rsimms@opus ~]$ sleep 5

[1]+  Stopped                 sleep 5
[rsimms@opus ~]$ bg
[1]+ sleep 5 &
[rsimms@opus ~]$
```

bg resumes the sleep command

PID 7728 is gone

```
[rsimms@opus ~]$ ps -l  -u rsimms
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
5 S   201  5368  5365  0  75   0 -  2460 -      ?        00:00:00 sshd
0 S   201  5369  5368  0  76   0 -  1165 wait   pts/0    00:00:00 bash
5 S   201  6203  6200  0  75   0 -  2491 -      ?        00:00:00 sshd
0 S   201  6204  6203  0  75   0 -  1165 -      pts/6    00:00:00 bash
0 R   201  7742  5369  0  78   0 -  1061 -      pts/0    00:00:00 ps
[rsimms@opus ~]$
```

Signals

Jim's app script

rsimms@opus:/home/cis90/depot

```
#!/bin/sh
#
# app - script to demostrate use of signals
#
# Usage:  run app with no options or parameters
#
# Send signals to it with keystrokes or kill command
#
# Notes:
# stty -echo stop the display of characters typed
# stty echo makes typed characters visible again
# stty susp ^Z sets suspend keystroke to Ctlr-Z (to stop forground processes)
# stty susp @ sets suspend character to @ (to stop foreground processes)
#
trap '' 2  #Ignore SIGINT
trap 'echo -n quit it!' 3 #Handle SIGQUIT
trap 'stty echo susp ^Z;echo ee; echo cleanup;exit' 15 #Handle SIGTERM
clear
banner testing
stty -echo susp @
sleep 1
echo  one
sleep 1
echo two
sleep 1
echo -n thr
while :
do sleep 1
done
~
                                                        13,1          All
```

This is why Cntl-F (suspend) stopped working and we had to use Ctrl-Z

Tangent on bg and SIGCONT

Signals

What is signal 18?

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)

Signal 18 continues a stopped process ... isn't that what bg does?

The bg command is used to resume a stopped process

```
/home/cis90/roddyduk $ sleep 60
Ctrl-F (or Ctrl-Z) typed here
[1]+  Stopped                 sleep 60
/home/cis90/roddyduk $ bg
[1]+ sleep 60 &
/home/cis90/roddyduk $ jobs
[1]+  Running                 sleep 60 &
/home/cis90/roddyduk $ jobs
[1]+  Running                 sleep 60 &
/home/cis90/roddyduk $ jobs
[1]+  Done                    sleep 60
/home/cis90/roddyduk $
```

bg resumed the stopped process which runs till it is finished

*Instead of using **bg** to resume a stopped process in the background, lets try a SIGCONT (signal 18) instead*

```
/home/cis90/roddyduk $ sleep 60
Ctrl-F (or Ctrl-Z) typed here
[1]+  Stopped                 sleep 60
/home/cis90/roddyduk $ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000 10705 10704  0  76   0 -  1165 wait   pts/0    00:00:00 bash
0 T  1000 10743 10705  0  75   0 -   926 finish pts/0    00:00:00 sleep
0 R  1000 10744 10705  0  78   0 -  1051 -      pts/0    00:00:00 ps
/home/cis90/roddyduk $ jobs
[1]+  Stopped                 sleep 60
/home/cis90/roddyduk $ kill -18 10743
/home/cis90/roddyduk $ jobs
[1]+  Running                 sleep 60 &
/home/cis90/roddyduk $ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000 10705 10704  0  75   0 -  1165 wait   pts/0    00:00:00 bash
0 S  1000 10743 10705  0  85   0 -   926 322800 pts/0    00:00:00 sleep
0 R  1000 10746 10705  0  77   0 -  1050 -      pts/0    00:00:00 ps
/home/cis90/roddyduk $ jobs
[1]+  Running                 sleep 60 &
/home/cis90/roddyduk $ jobs
[1]+  Running                 sleep 60 &
/home/cis90/roddyduk $ jobs
[1]+  Done                    sleep 60
```

Note sending a 18 signal or using the bg command will resume a stopped process