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 -

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

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

- Check that room headset is charged – done

CIS 90 - Lesson 11

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

Quiz

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

1. Name four states a process can be in.
2. What command shows the current running processes?
3. What is the difference between the fork and exec system calls?

email answers to: risimms@cabrillo.edu

[ ] 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

vi editor

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

✵ = hands on exercise for topic

Housekeeping

Previous material and assignment

1. Questions?

2. Lab 8 due at midnight

```
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

grep

grep usage

What is my account information in /etc/passwd?

```
/home/cis90/simmsben $ grep $LOGNAME /etc/passwd
simmsben:x:1200:90:Benji Simms:/home/cis90/simmsben:/bin/bash
```

or

```
/home/cis90/simmsben $ grep simmsben /etc/passwd
simmsben:x:1200:90:Benji Simms:/home/cis90/simmsben:/bin/bash
```

or

```
/home/cis90simmsben $ cat /etc/passwd | grep $LOGNAME
simmsben:x:1200:90:Benji Simms:/home/cis90/simmsben:/bin/bash
```

My user account is simmsben, my password is kept in /etc/shadow, my user ID is 1200, my primary group ID is 90, my full name is Benji Simms, my home directory is /home/cis90/simmben, my shell is /bin/bash

grep usage

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

```
/home/cis90/simmsben $ ps -ef | grep cups
root      3365     1  0 Sep28 ?        00:00:00 cupsd
simmsben 20598 20540  0 08:19 pts/1    00:00:00 grep cups
root     31822     1  0 Nov02 ?        00:00:00 eggcups --sm-client-id default4
```

Yes it is, with 3365

grep usage

Is Samba (File and Print services) installed?

```
/home/cis90/roddyduk $ rpm -qa | grep samba
system-config-samba-1.2.39-1.el5
samba-client-3.0.28-1.el5_2.1
samba-3.0.28-1.el5_2.1
samba-common-3.0.28-1.el5_2.1
/home/cis90/roddyduk $
```

Yes, the client, server and common packages have been installed already

grep usage

How many CIS 90 user accounts are there?

```
/home/cis90ol/simmsben $ grep cis90 /etc/passwd | wc -l
56
/home/cis90ol/simmsben $ grep "/cis90/" /etc/passwd | wc -l
31
/home/cis90ol/simmsben $ grep "/cis90ol/" /etc/passwd | wc -l
25
```

There are 56. 31 for the regular section and another 25 for the online section

grep usage

Which shell is the biggest (Lab 8)?

```
/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
```

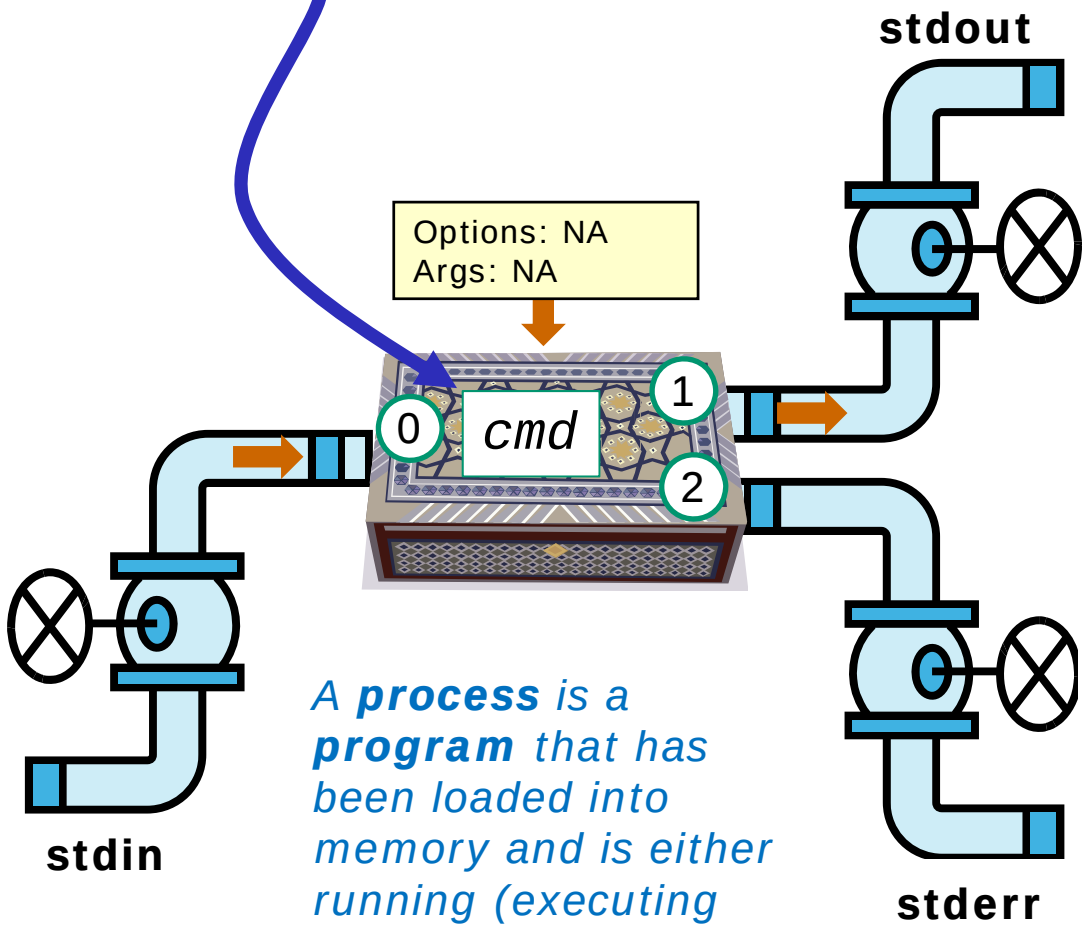
grep practice

- How many CIS130 accounts are there?
- Is the cronjob daemon (crond) running right now?
- Has the mysql package been installed on Opus?

Review of Processes

Program to process

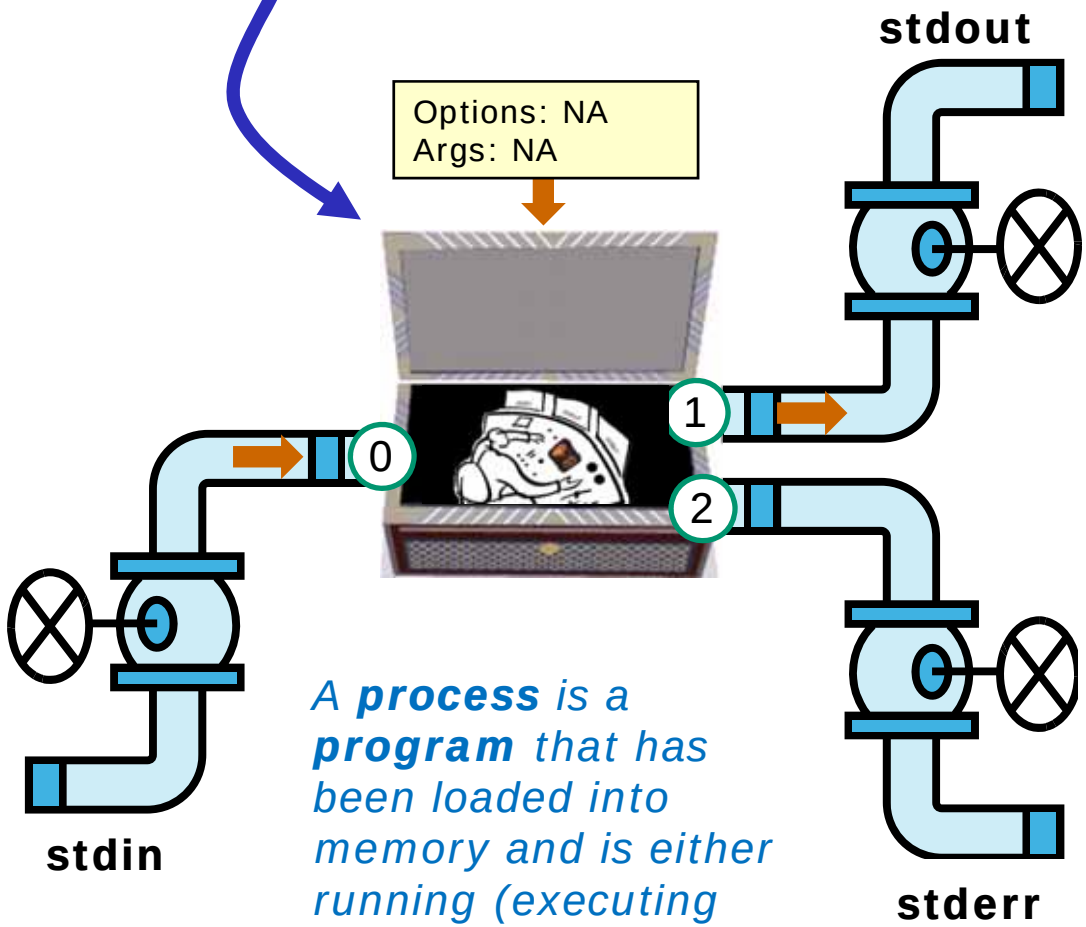
```
/home/cis90/roddyduk $cmd
```



*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**

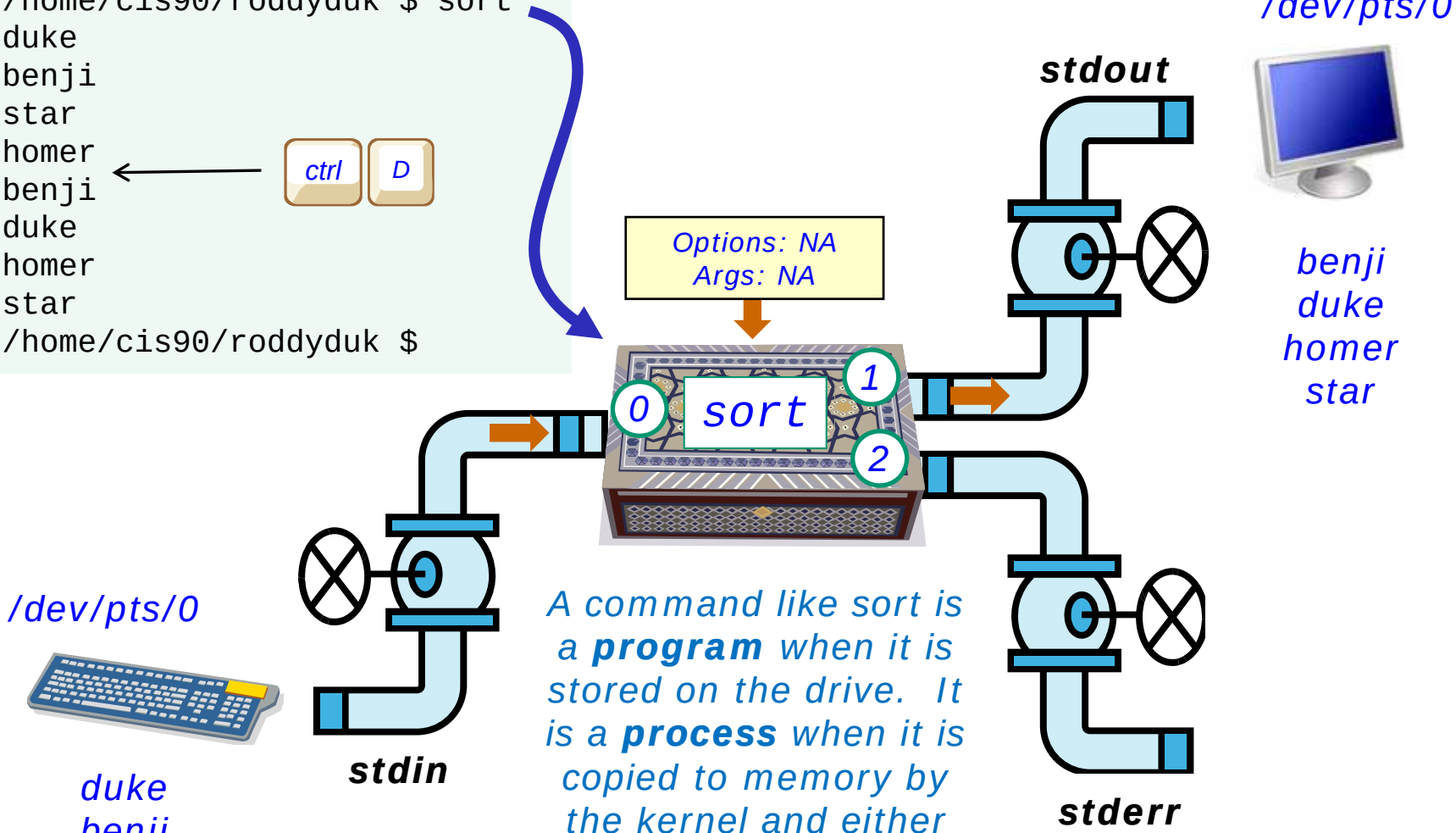
- and may get interrupted from time to time by a **signal**

*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 $
```

ctrl D

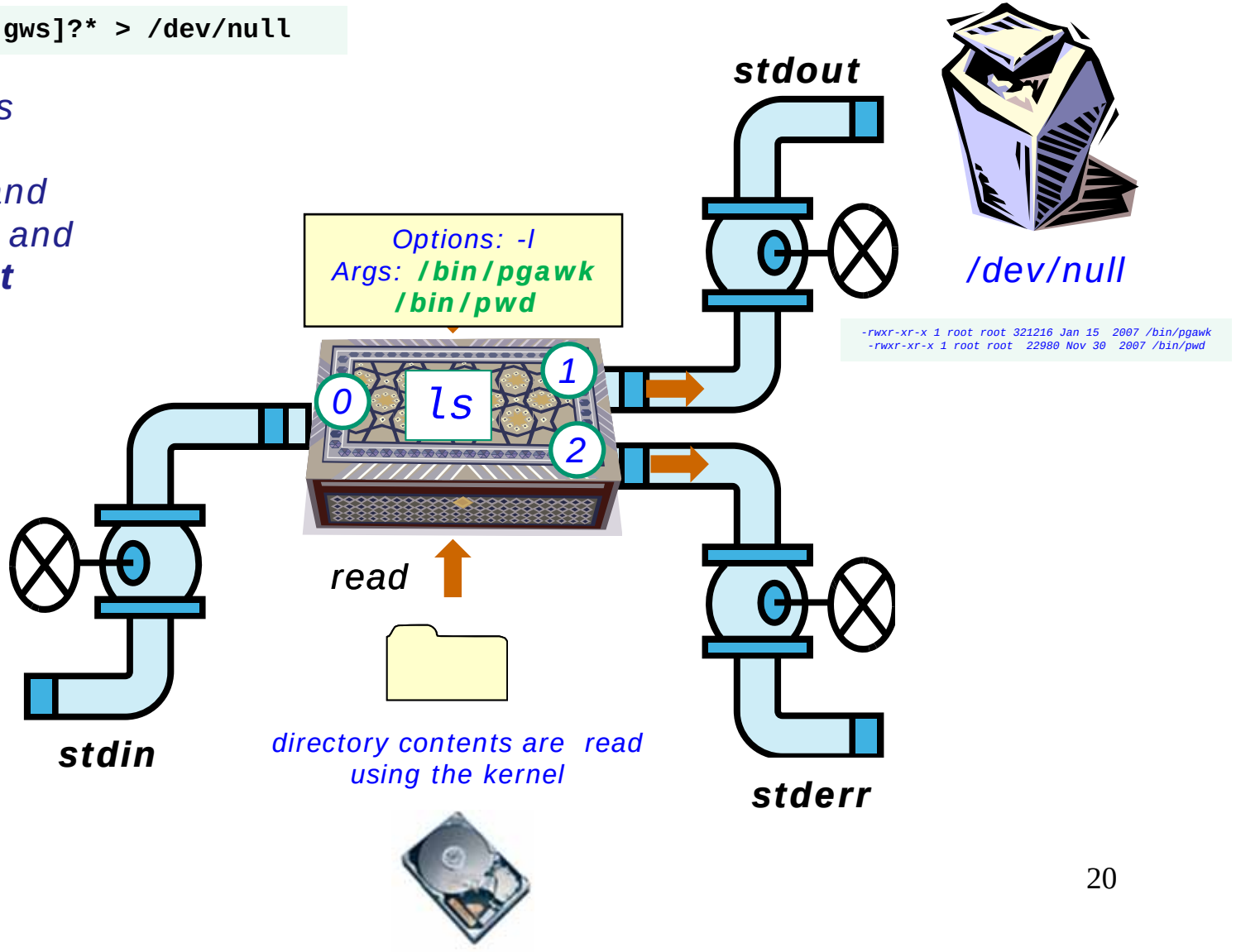


A command like sort is a **program** when it is stored on the drive. It is a **process** when it is copied to memory by the kernel and either running or waiting to run.

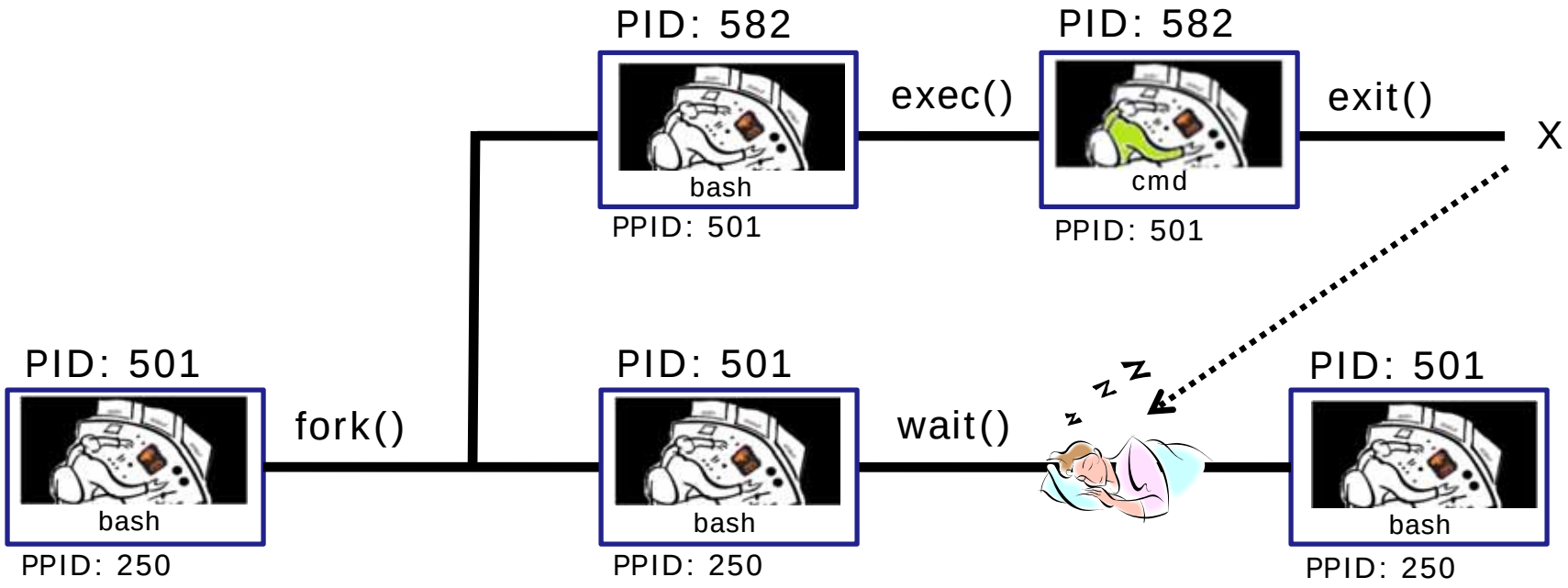
example program to process

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

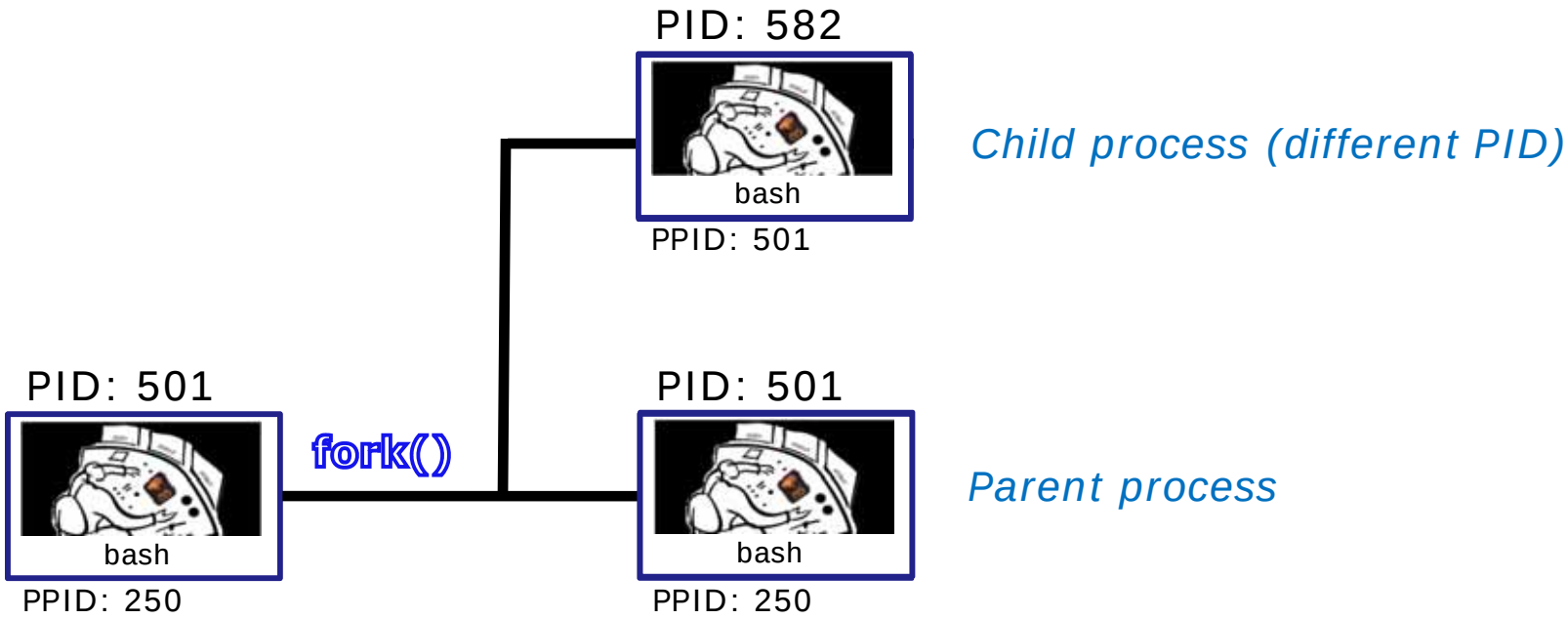
*Note: ls gets its input from the command line and the OS (kernel) and writes to **stdout** (redirected to /dev/null) and **stderr**.*



Process Lifecycle



Process Lifecycle

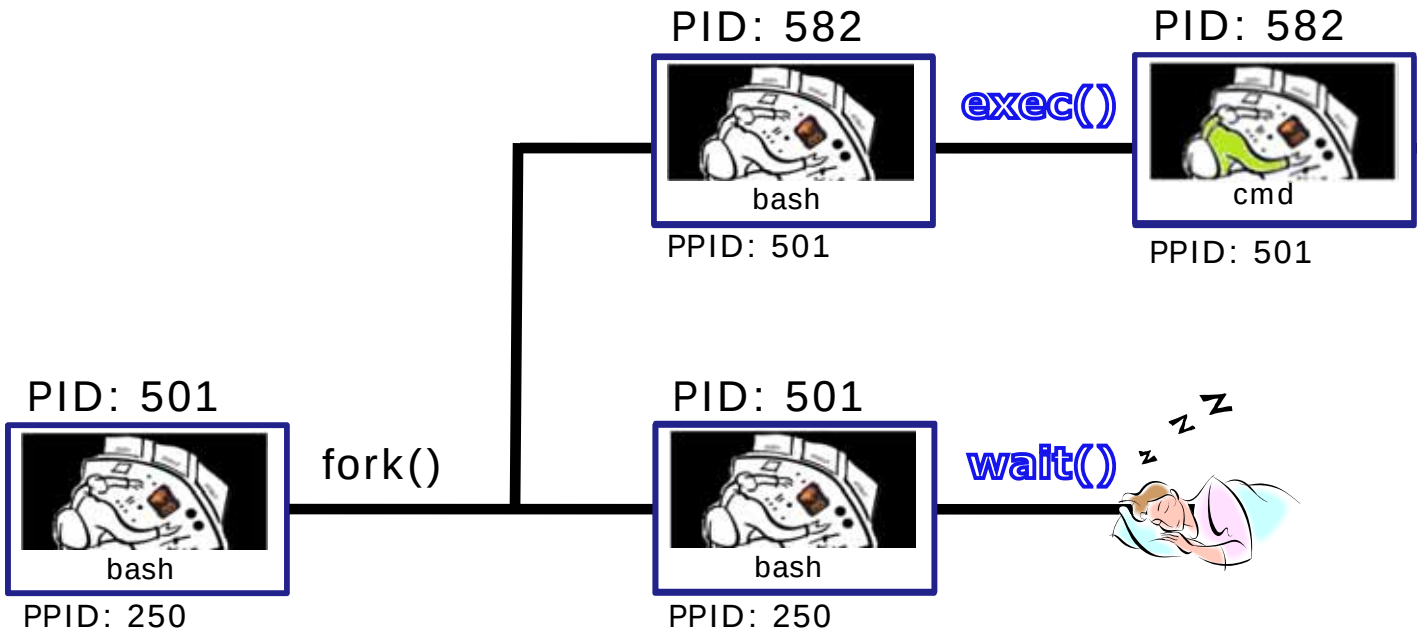


1) When a program is loaded into memory a new process must be created.

*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.*

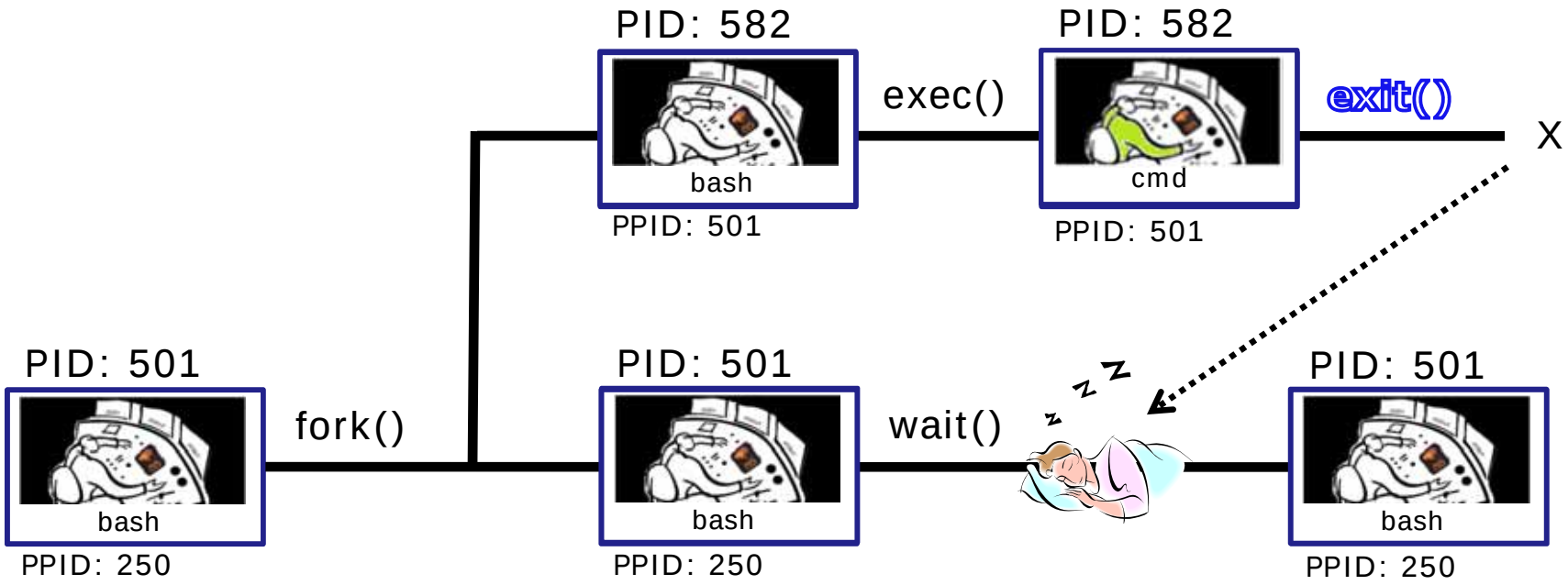
Process Lifecycle



*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.*

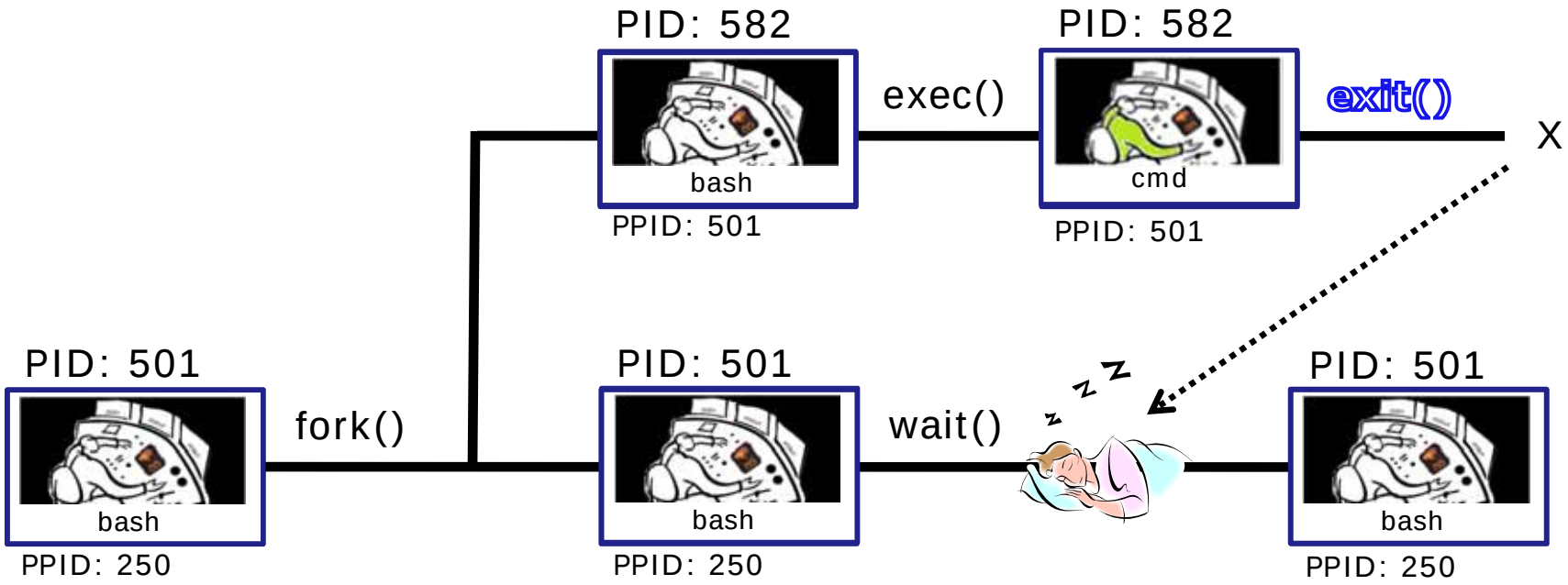
Process Lifecycle



*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

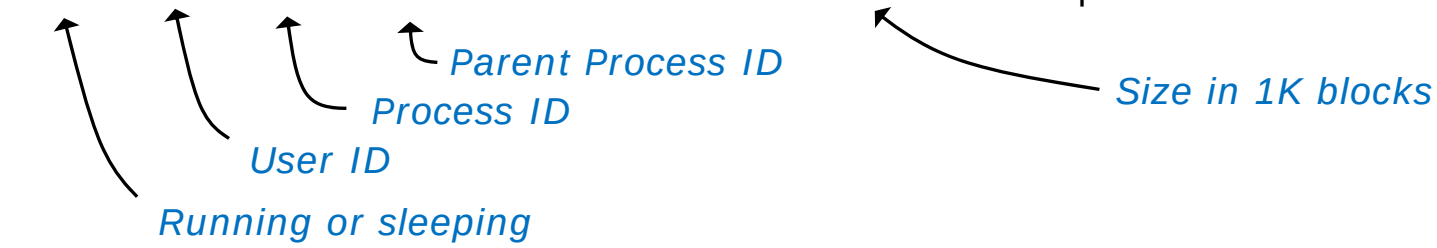


*3) If the **parent** process were to die before the **child**, the zombie will become an **orphan**. Fortunately the init process will adopt any orphaned **zombies**.*

Process Information

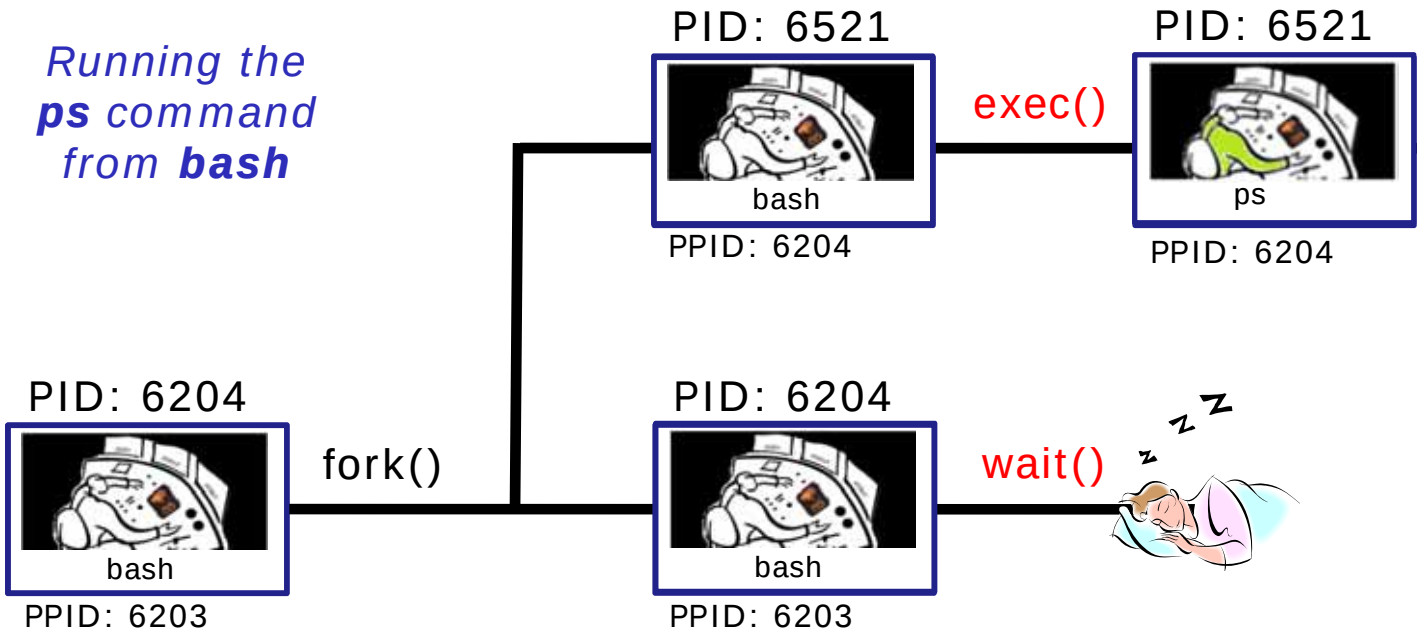
Use –l for additional options

```
[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
```



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

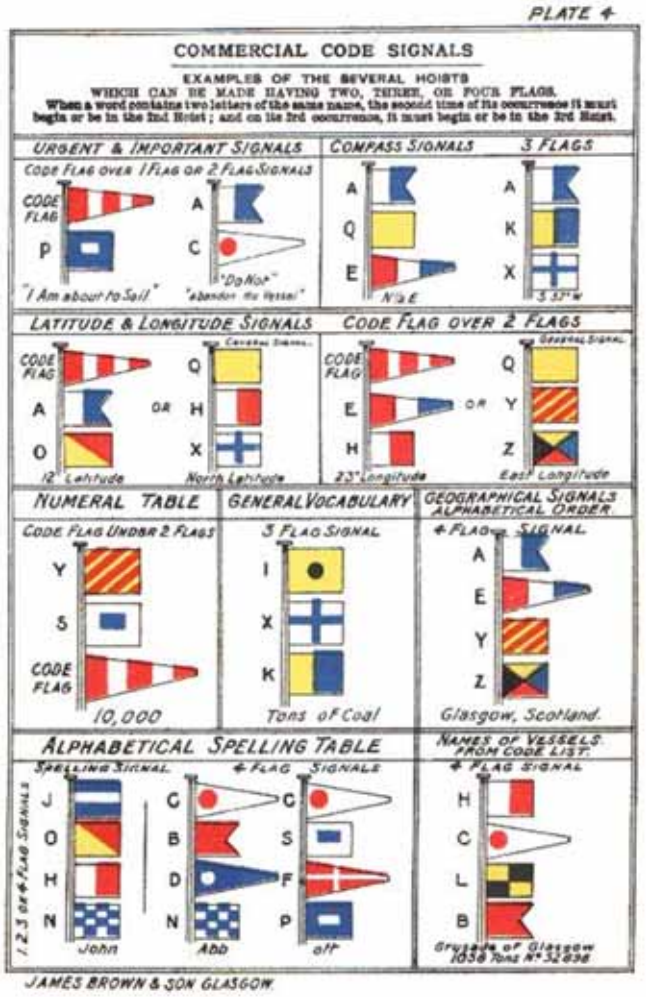
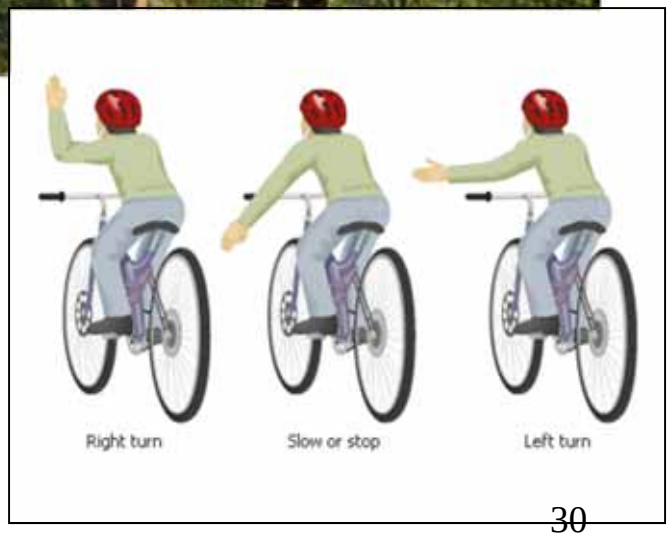


PLATE 4
COMMERCIAL CODE SIGNALS
EXAMPLES OF THE SEVERAL HOISTS
WHICH CAN BE MADE HAVING TWO, THREE, OR FOUR FLAGS.
URGENT & IMPORTANT SIGNALS
COMPASS SIGNALS
3 FLAGS
LATITUDE & LONGITUDE SIGNALS
CODE FLAG OVER 2 FLAGS
NUMERAL TABLE
GENERAL VOCABULARY
GEOGRAPHICAL SIGNALS ALPHABETICAL ORDER
ALPHABETICAL SPELLING TABLE
NAMES OF VESSELS FROM CODE LIST.
10,000
Tons of Coal
Glasgow, Scotland.
JAMES BROWN & SON GLASGOW


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

Use kill –l to see all of them

```
/home/cis90/simmsben $ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGSTKFLT
17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU
25) SIGXFSZ     26) SIGVTALRM   27) SIGPROF     28) SIGWINCH
29) SIGIO       30) SIGPWR      31) SIGSYS      34) SIGRTMIN
35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3  38) SIGRTMIN+4
39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12
47) SIGRTMIN+13 48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14
51) SIGRTMAX-13 52) SIGRTMAX-12 53) SIGRTMAX-11 54) SIGRTMAX-10
55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7  58) SIGRTMAX-6
59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
/home/cis90/simmsben $
```

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          39 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          40 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          All
```

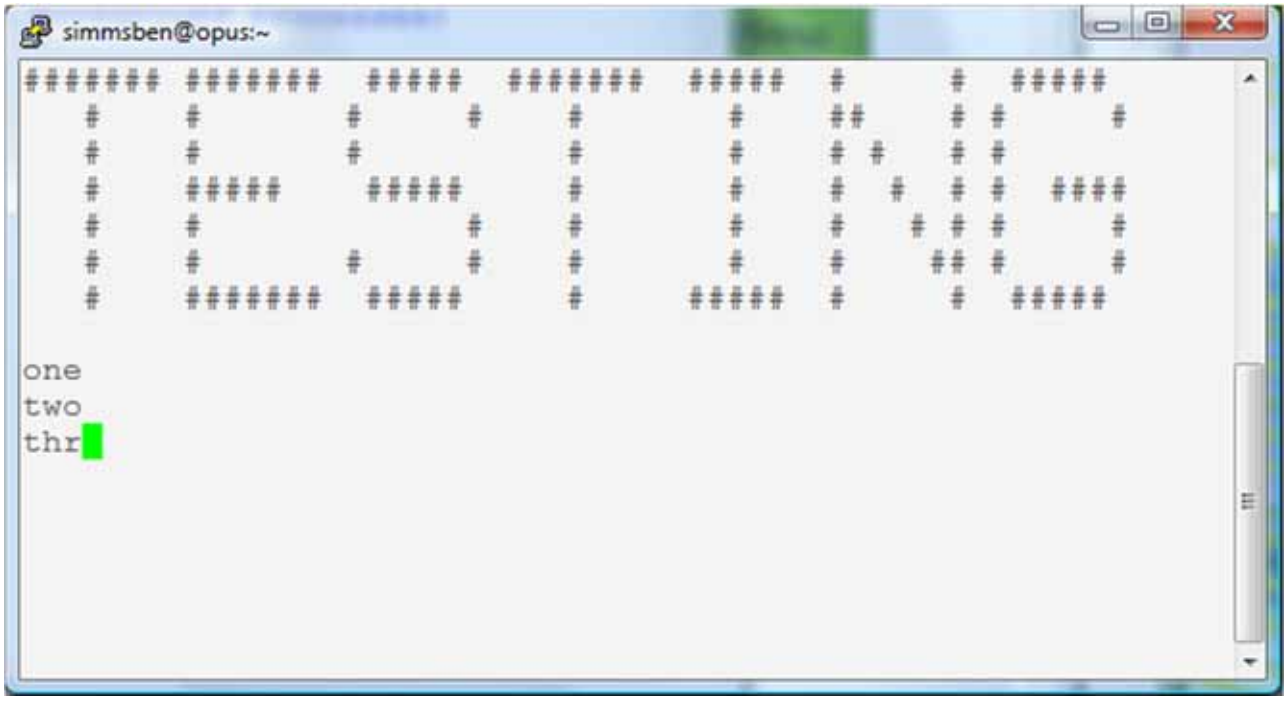
Jim's app script

Endless loop

rsimms@opus:/home/cis90/depot

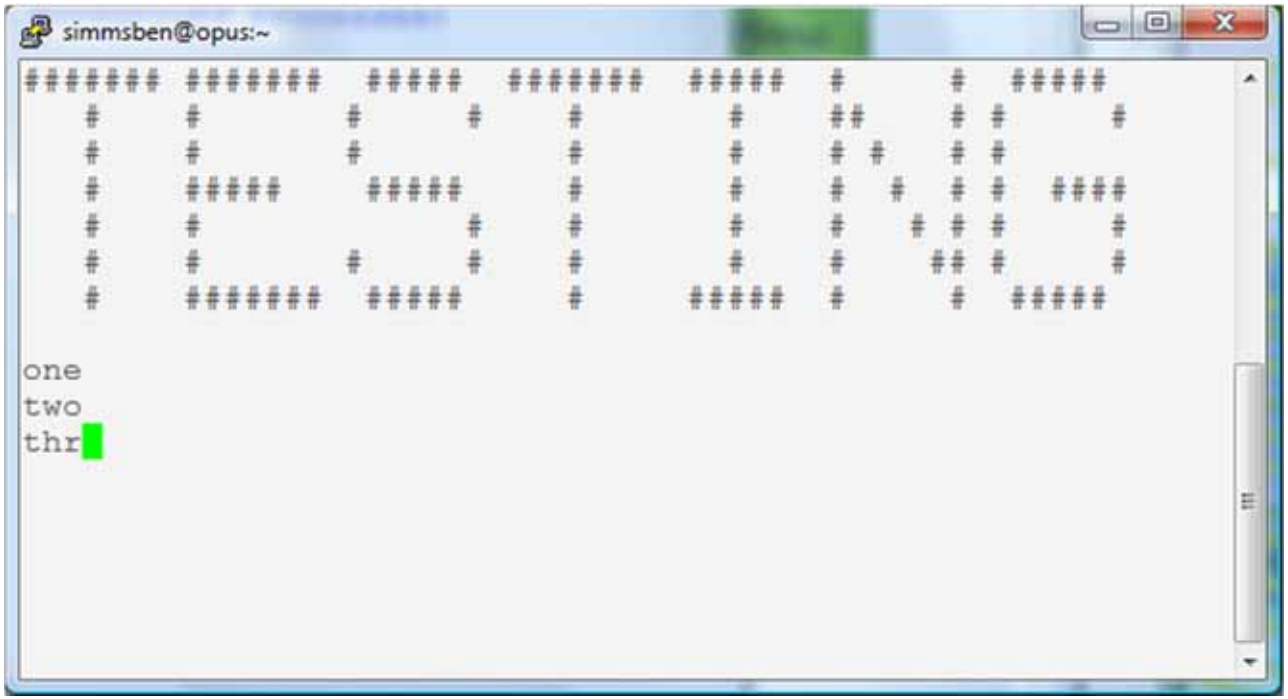
```
#!/bin/sh
#
# app - script to demostrate use of signals
#
# Usage:  run app with no options or parameters
#
# Send signals to it with keystrokes or kill command
#
# Notes:
# stty -echo stop the display of characters typed
# stty echo makes typed characters visible again
# stty susp ^Z sets suspend 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          43 All
```

Signals
Benji runs app



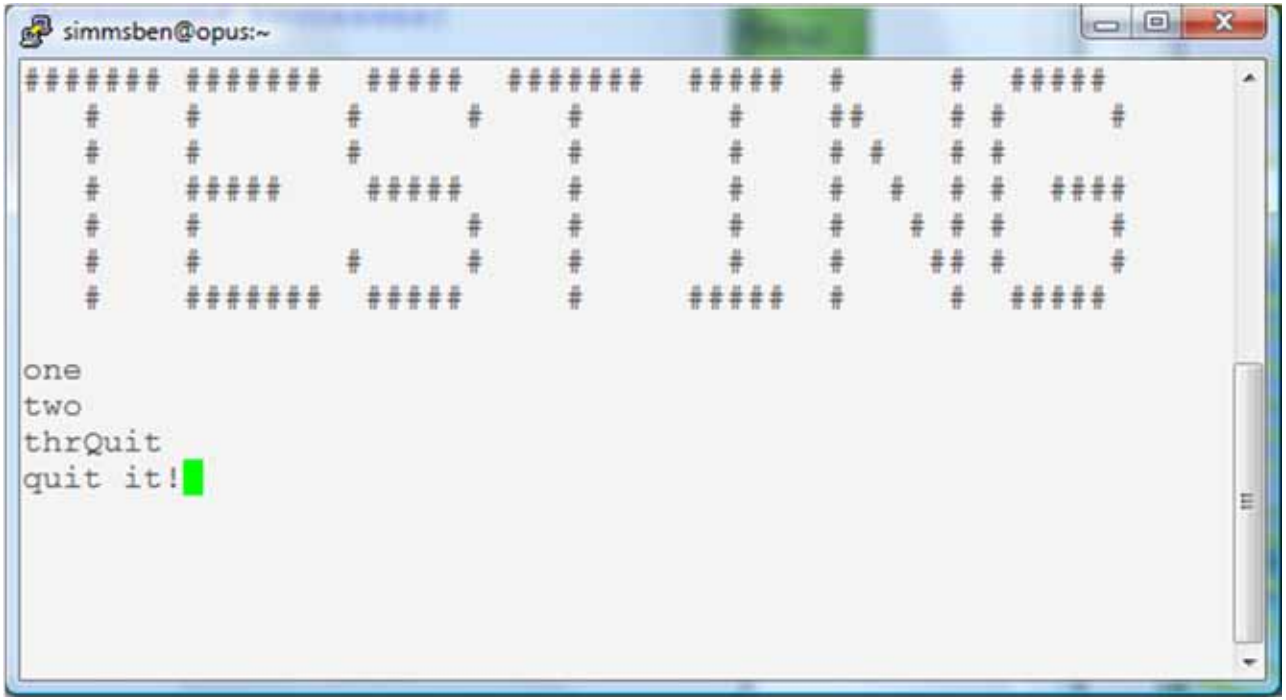
Benji logs in and runs app … uh oh, its stuck !

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/3s but the app process shrugs them off with two "quit it!" messages

Signals

Benji runs app

simmsben@opus:~

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

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

```
one
two
thr
```

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 $
```

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 this time ... and app never even sees it coming ... poof ... app is gone

Review of Job Control

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 & to run any command or script 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
```

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

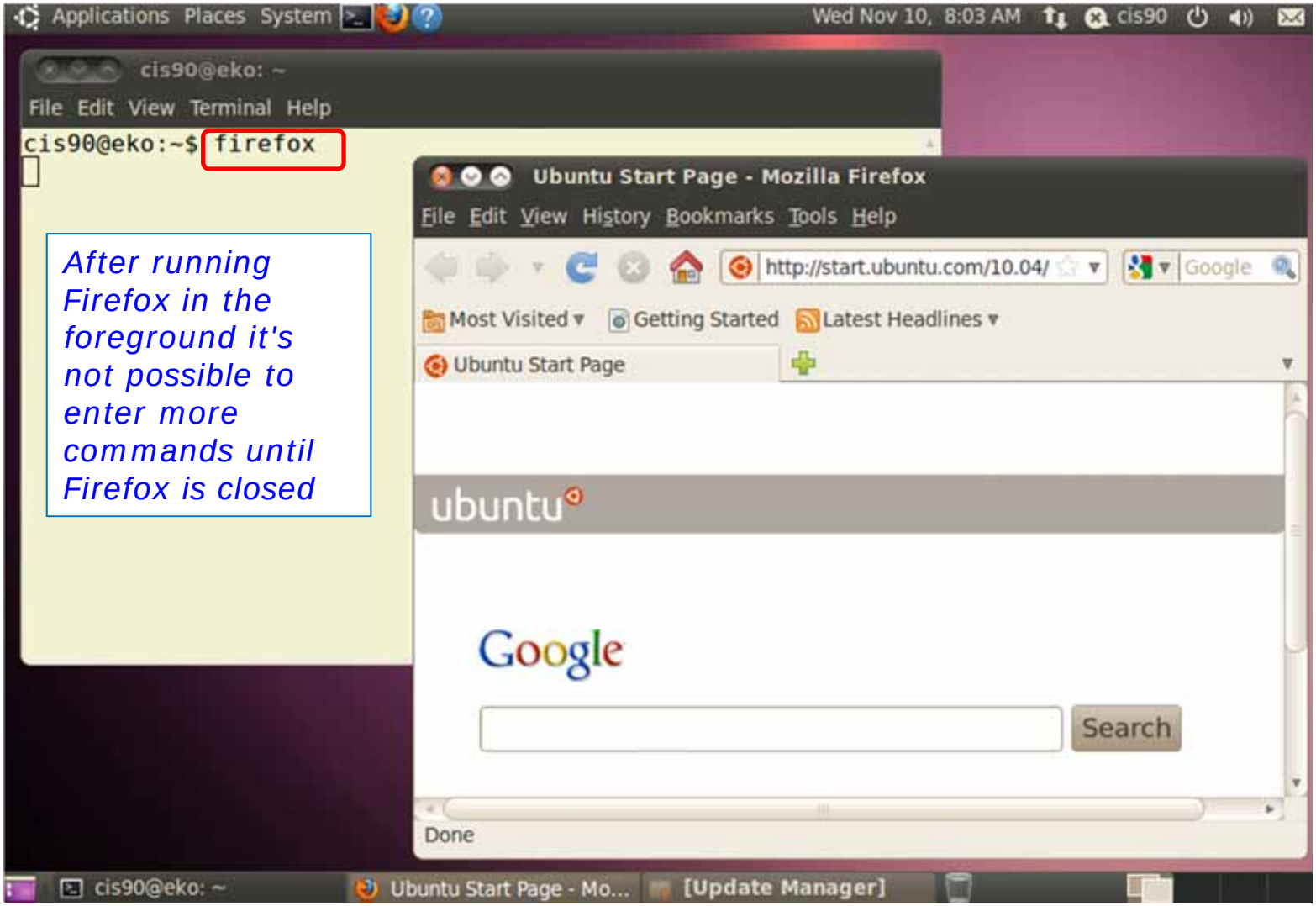
Example 2

```
/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

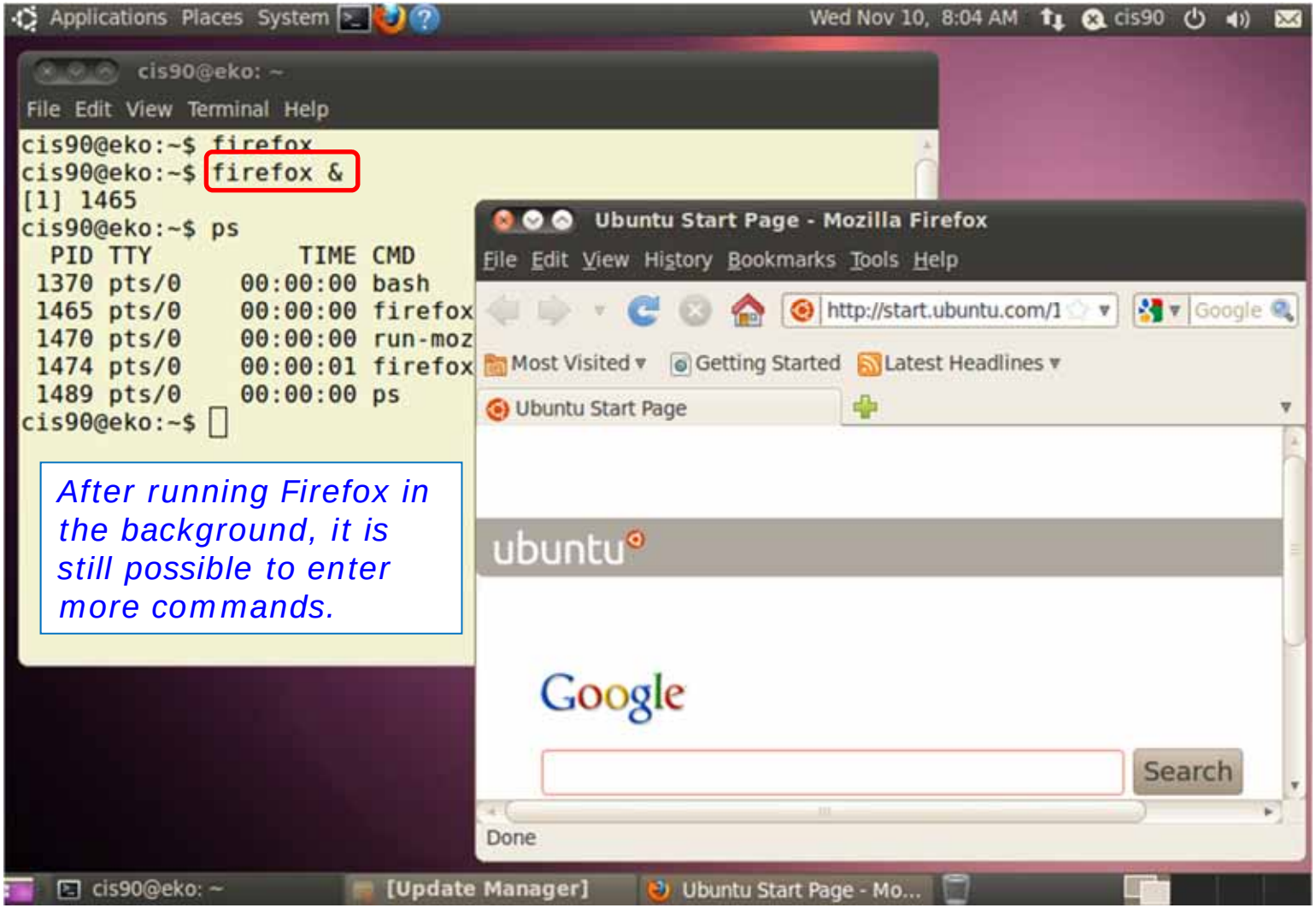
Job Control

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



Job Control

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



Job Control

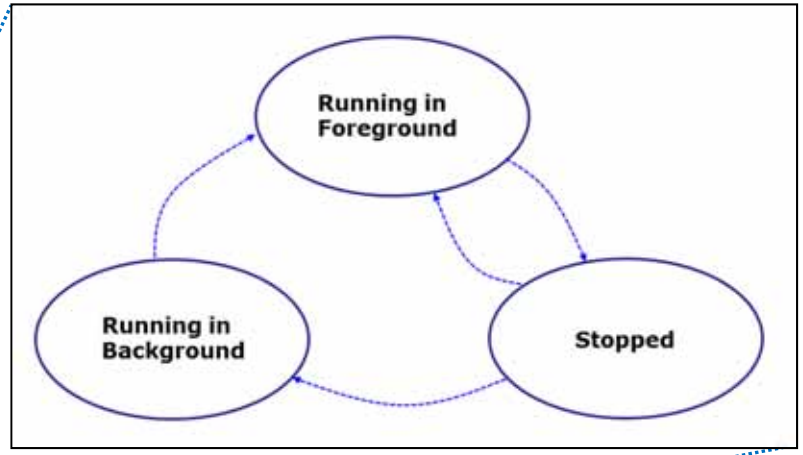
A feature of the bash shell

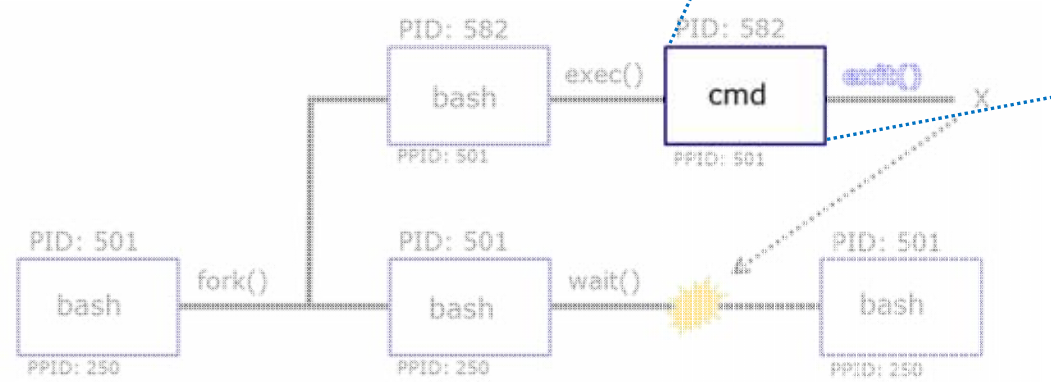
&	Append to a command to run it in the background
bg	Resumes a suspended job in the background
fg	Brings the most recent background process to the foreground
jobs	Lists all background jobs

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

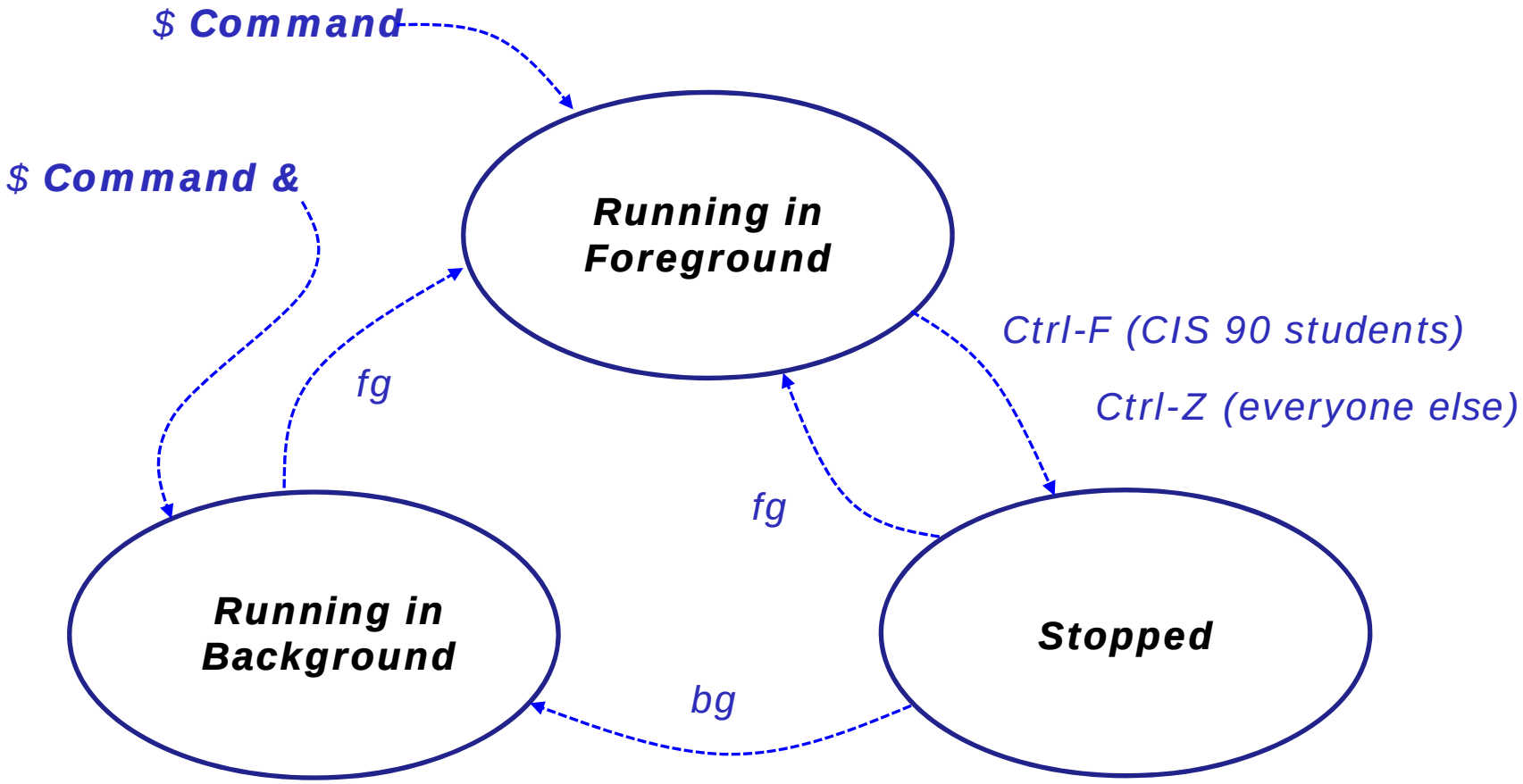
Job Control
A feature of the bash shell

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

*Jobs can be resumed in the background using **bg** or in the foreground using **fg***

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!

Job Control

- Run and suspend two jobs
  sleep 125
  Ctrl-F or Ctrl-Z
  sleep 120
  Ctrl-F or Ctrl-Z
- Use **jobs** to see them
- Resume one job with the **bg** command
- Use **jobs** to see change
- Bring the other to the foreground with **fg**
- Use **jobs** when control returns to see that every process finished
- Use **sleep 15 &** to run in the background
- Use **jobs** to check on progress

Review of Load Balancing

Load Balancing

The **at** command reads from stdin or a file for a list of commands to run, and begins running them at the time of day specified as the first argument:

```
$ cat job1
cp bin/myscript bin/myscript.bak
$ at 10:30pm < job1
```

This will run the cp command in the file job1 at 10:30 PM

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

*This will run the commands entered after the **at** command at 11:59 PM*

*Hold down the **Ctrl** key, then tap the **D** key on the keyboard for an EOF (end of file)*

Load Balancing

Managing queued jobs

This job makes a backup of myscript and sends an email when finished

```
/home/cis90/roddyduk $ cat job1
cp bin/myscript bin/myscript.bak
echo "Job 1 - finished, myscript has been backed up" | mail -s "Job 1" roddyduk
/home/cis90/roddyduk $ at now + 5 minutes < job1
job 24 at 2008-11-12 12:14
/home/cis90/roddyduk $ at now + 2 hours < job1
job 25 at 2008-11-12 14:09
/home/cis90/roddyduk $ at teatime < job1
job 26 at 2008-11-12 16:00
/home/cis90/roddyduk $ at now + 1 week < job1
job 27 at 2008-11-19 12:10
/home/cis90/roddyduk $ at 3:00 12/12/2011 < job1
job 28 at 2011-12-12 03:00
/home/cis90/roddyduk $ jobs
/home/cis90/roddyduk $ atq
25      2008-11-12 14:09 a roddyduk
28      2008-12-12 03:00 a roddyduk
27      2008-11-19 12:10 a roddyduk
26      2008-11-12 16:00 a roddyduk
24      2008-11-12 12:14 a roddyduk
/home/cis90/roddyduk $
```

Several ways to specify a future time to run

*Use the **atq** command to show queued jobs*

Load Balancing

Managing queued jobs

```
/home/cis90/roddyduk $ jobs
/home/cis90/roddyduk $ atq
25      2008-11-12 14:09 a roddyduk
28      2008-12-12 03:00 a roddyduk
27      2008-11-19 12:10 a roddyduk
26      2008-11-12 16:00 a roddyduk
24      2008-11-12 12:14 a roddyduk
/home/cis90/roddyduk $ atrm 24
/home/cis90/roddyduk $ atq
25      2008-11-12 14:09 a roddyduk
28      2008-12-12 03:00 a roddyduk
27      2008-11-19 12:10 a roddyduk
26      2008-11-12 16:00 a roddyduk
/home/cis90/roddyduk $
```

*The **jobs** command lists processes running or suspended in the background.*

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

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

vi

vi

Making a script

In your bin directory, create a file called color and add the following lines:

```
echo -n "What is your name? "
read NAME
echo -n "What is your favorite color? "
read COLOR
echo "Hi $NAME, your favorite color is $COLOR"
```

Save the file, and give it execute permissions with ***chmod +x color***

Now run your script by typing its name

vi

Moving around in a file

Note: to execute any of the following commands from vi, you must be in command mode. Press the Esc key to enter 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
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
^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

vi

Reading and Writing out files

Note: to execute any of the following commands from vi, you must be in command mode. Press the Esc key to enter command mode.

:q exits vi if you have saved your changes
:q! exits vi even if you have not saved your changes
:w saves any changes you've made to the file you are editing
: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 Input mode

i Ready to insert characters immediately before the current cursor position
a Ready to append characters immediately after the current cursor position
I Ready to insert characters at the start of the current line
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
r Ready to replace the current character with the character you type next
R Ready to Replace (overwrite) characters starting at the current cursor position
s Ready to replace the current character with the string you type next
cw Ready to replace the current word with the string you type next

vi

Cut, Copy, Pasting Commands

Note: to execute any of the following commands from vi, you must be in command mode. Press the Esc key to enter command mode.

x Deletes the current character
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

Note: to execute any of the following commands from vi, you must be in command mode. Press the Esc key to enter 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

vi activity

```
HOW SMALL IS SMALL?

YOU KNOW WHEN YOU'RE IN A SMALL TOWN WHEN...
   The airportttttt runaway is terraced.
   The polka is more popular ththanan a mashpit on on Saturday noght.
   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.

YOU KNOW YOU'RE IN A SMALL TOWN WHEN...
   You are born on June 13 and
your family receives gifts from the local merchants because you are the first
baby of the year.
   You speak to each dogg you pass by name and he wags at you.
   You dial a wrong number and talk for 15 minutes anyway.
   You are run off Main Street by a combine.

                              Reprinter from the Ayshire Empire News

xxxxx
   You cna't walk for exercise because every car that passes you offers you a
      ride.
   You get married and the local paper devotes a quarter page to the story.
   You drive into a ditch 5 miles out of town and the word gets back to town
before you do.

YOU KNOW YOU'RE IN A SMALL TOWN WHEN...
YOU KNOW YOU'RE IN A SMALL TOWN WHEN...
YOU KNOW YOU'RE IN A SMALL TOWN WHEN...
   The biggest bussssssiness on town sells farm machinery.
   You write a check on the wrng bank and it covers you anyway.
   The pickups on Main Street outnumber the cars three to one.
   You miss church on Sunday and someone sends youa get well card.
   Someone asks you how you feel and then listens to what you say.

Thank God for small towns... and the people who live in them.
```

In Lab 9 you clean up this text file

vi activity

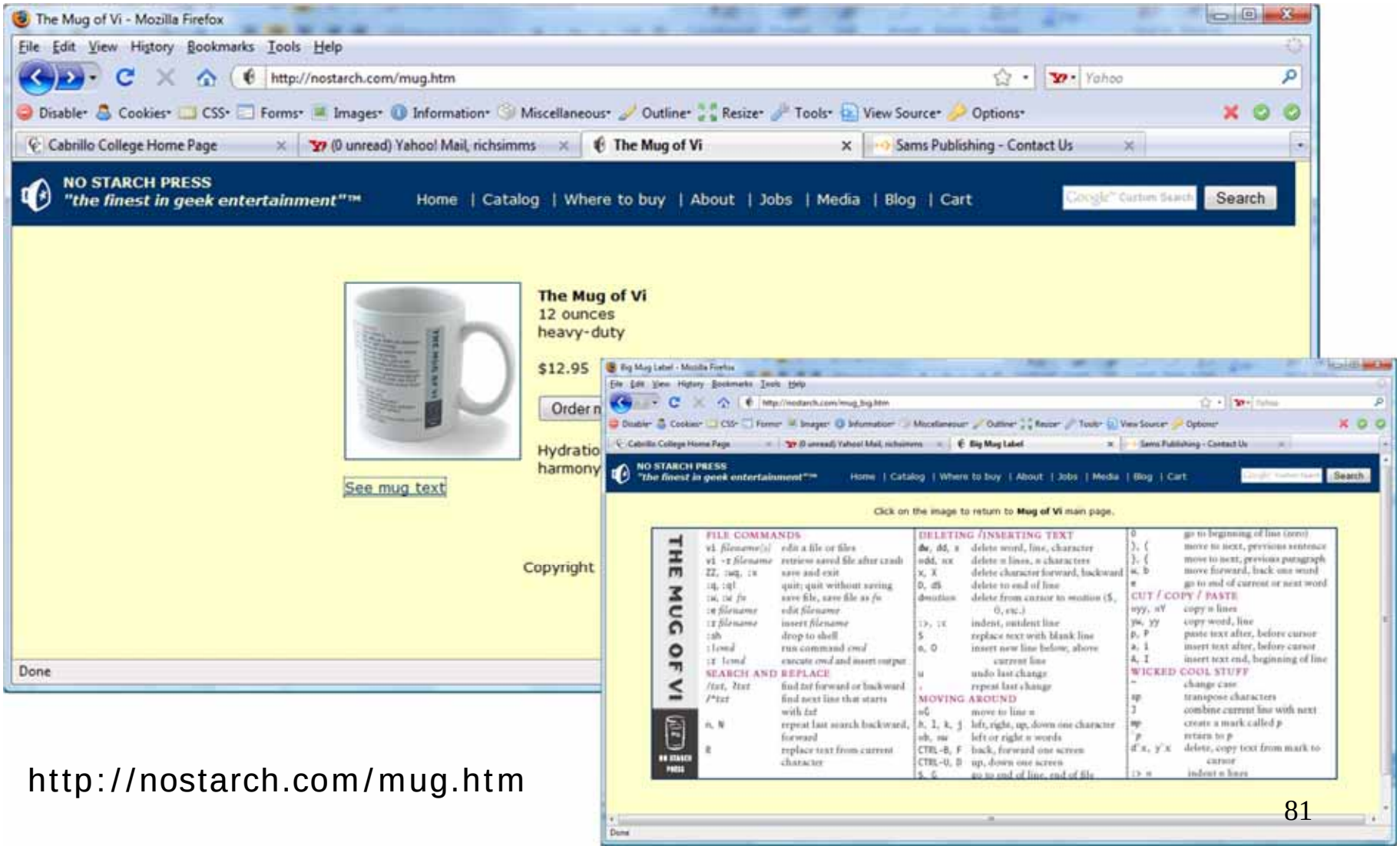
```
Technology for Mountain Folk

1.   LOG ON:            Makin a wood stove hotter.
2.   LOG OFF:           Don't add no more wood.
3.   MONITOR:           Keepin an eye on the wood stove.
4.   DOWNLOAD:          Gettin the farwood off the truk.
5.   MEGA HERTZ:        When yer not kerful gettin the farwood.
6.   FLOPPY DISC:       Whatcha git from tryin to carry too much farwood.
7.   RAM:               That thar thing whut splits the farwood.
8.   HARD DRIVE:        Gettin home in the winter time.
9.   PROMPT:            Whut the mail ain't in the winter time.
10. WINDOWS:            Whut to shut when it's cold outside.
11. SCREEN:             Whut to shut when it's blak fly season.
12. BYTE:               Whut them dang flys do.
13. CHIP:               Munchies fer the TV.
14. MICRO CHIP:         Whut's in the bottom of the munchie bag.
15. MODEM:              Whut cha did to the hay fields.
16. DOT MATRIX:         Old Dan Matrix's wife.
17. LAP TOP:            Whar the kitty sleeps.
18. KEYBOARD:           Whar ya hang the dang keys.
19. SOFTWARE:           Them dang plastic forks and knifs.
20. MOUSE:              Whut eats the grain in the barn.
21. MAINFRAME:          Holds up the barn roof.
22. PORT:               Fancy Flatlander wine.
23. ENTER:              Northerner talk few "C'mon in y'all"
24. RANDOM ACCESS MEMORY: Wen ya cain't 'member whut ya paid fer the rifle
                                        when yore wife asks.
25. MOUSE PAD:          That hippie talk fer the rat hole.
```

In Lab 9 you clean up and sort this text file

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

The Mug of vi



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
```

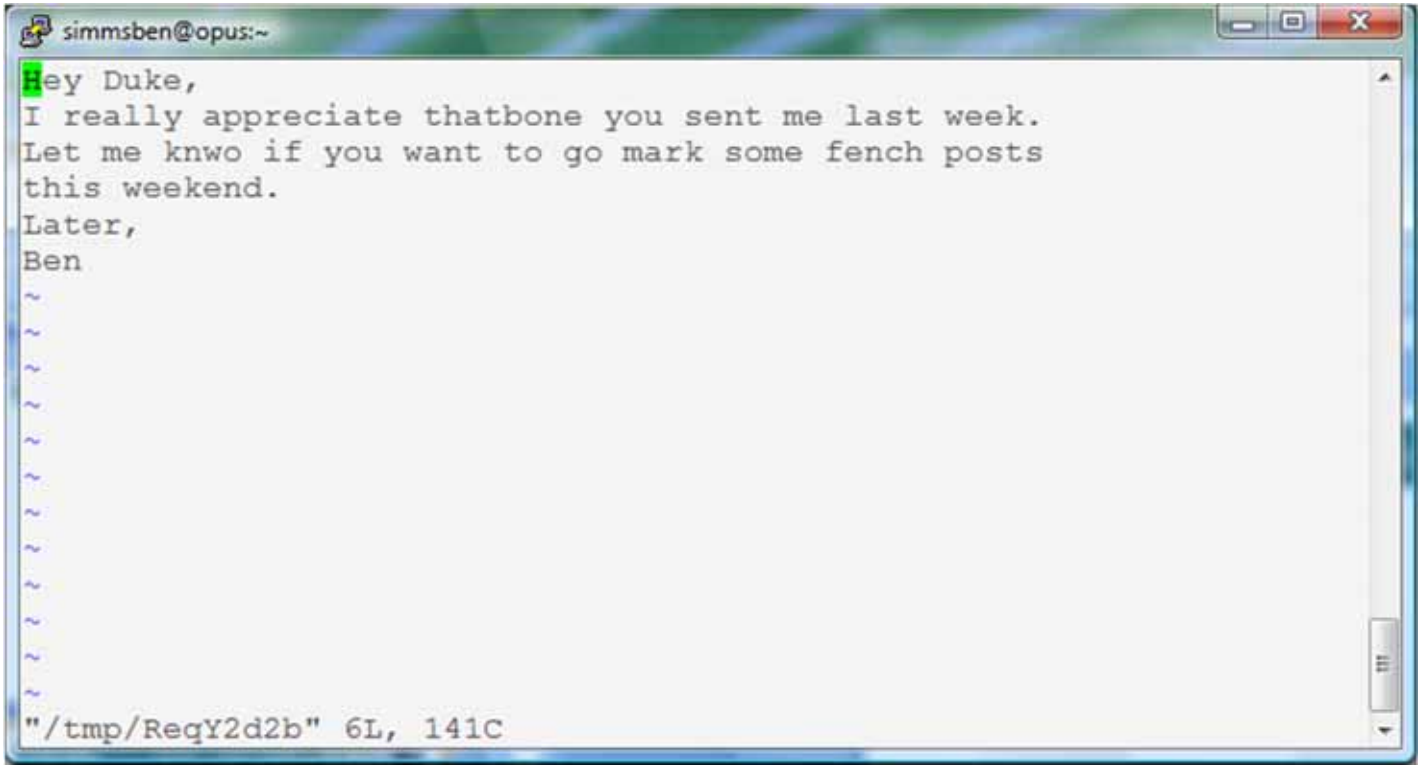
You are composing a message and you spot some typos …
CRUD … what can you do?

/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



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.

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

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

```
/home/cis90/roddyduk $ bash
[roddyduk@opus ~]$ bash
[roddyduk@opus ~]$ bash
[roddyduk@opus ~]$ ps
  PID TTY          TIME CMD
11198 pts/6    00:00:00 bash
11233 pts/6    00:00:00 bash
11257 pts/6    00:00:00 bash
11284 pts/6    00:00:00 bash
11309 pts/6    00:00:00 ps
[roddyduk@opus ~]$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000 11198 11197  0  75   0 -  1165 wait   pts/6    00:00:00 bash
0 S  1000 11233 11198  0  75   0 -  1166 wait   pts/6    00:00:00 bash
0 S  1000 11257 11233  0  75   0 -  1166 wait   pts/6    00:00:00 bash
0 S  1000 11284 11257  0  75   0 -  1165 wait   pts/6    00:00:00 bash
0 R  1000 11312 11284  0  77   0 -  1051 -      pts/6    00:00:00 ps
[roddyduk@opus ~]$ exit
exit
[roddyduk@opus ~]$ exit
exit
[roddyduk@opus ~]$ exit
exit
/home/cis90/roddyduk $ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000 11198 11197  0  75   0 -  1165 wait   pts/6    00:00:00 bash
0 R  1000 11363 11198  0  77   0 -  1051 -      pts/6    00:00:00 ps
/home/cis90/roddyduk $
```

Parent and child

Resume stopped jobs with bg and kill -18

```
[roddyduk@opus ~]$ sleep 60

[1]+  Stopped                 sleep 60
[roddyduk@opus ~]$ sleep 90

[2]+  Stopped                 sleep 90
[roddyduk@opus ~]$ ps -lf
F S UID        PID  PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S roddyduk 11529 11528  0  75   0 -  1165 wait   09:36 pts/6    00:00:00 -bash
0 S roddyduk 11560 11529  0  75   0 -  1165 wait   09:36 pts/6    00:00:00 bash
0 S roddyduk 11584 11560  0  75   0 -  1166 wait   09:36 pts/6    00:00:00 bash
0 S roddyduk 11608 11584  0  75   0 -  1166 wait   09:36 pts/6    00:00:00 bash
0 T roddyduk 11796 11608  0  75   0 -   926 finish 09:49 pts/6    00:00:00 sleep
60
0 T roddyduk 11798 11608  0  75   0 -   926 finish 09:49 pts/6    00:00:00 sleep
90
0 R roddyduk 11803 11608  0  77   0 -  1062 -      09:49 pts/6    00:00:00 ps -lf
[roddyduk@opus ~]$ jobs
[1]-  Stopped                 sleep 60
[2]+  Stopped                 sleep 90
[roddyduk@opus ~]$ bg
[2]+ sleep 90 &
[roddyduk@opus ~]$ jobs
[1]+  Stopped                 sleep 60
[2]-  Running                 sleep 90 &
[roddyduk@opus ~]$ kill -18 11796
[roddyduk@opus ~]$ jobs
[1]-  Done                    sleep 60
[2]+  Running                 sleep 90 &
```

```
/home/cis90/roddyduk $ sleep 60
Ctrl-F 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  75   0 -  1165 wait   pts/0    00:00:00 bash
0 T  1000 10737 10705  0  84   0 -   927 finish pts/0    00:00:00 sleep
0 R  1000 10739 10705  0  77   0 -  1051 -      pts/0    00:00:00 ps
/home/cis90/roddyduk $ kill -18 10737
/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 S  1000 10737 10705  0  78   0 -   927 322800 pts/0    00:00:00 sleep
0 R  1000 10741 10705  0  78   0 -  1051 -      pts/0    00:00:00 ps
/home/cis90/roddyduk $ jobs
[1]+  Done                    sleep 60
```

*Instead of using **bg** to resume a stopped process in the backgroud, lets use a kill signal instead*

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

Use `kill -l` to see all of them

```
/home/cis90/roddyduk $ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGSTKFLT
17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU
25) SIGXFSZ     26) SIGVTALRM   27) SIGPROF     28) SIGWINCH
29) SIGIO       30) SIGPWR      31) SIGSYS      34) SIGRTMIN
35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3  38) SIGRTMIN+4
39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12
47) SIGRTMIN+13 48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14
51) SIGRTMAX-13 52) SIGRTMAX-12 53) SIGRTMAX-11 54) SIGRTMAX-10
55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7  58) SIGRTMAX-6
59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

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)?

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)

Note Signal 20 is used to stop a process and moves it to the background

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

Signals

- Run and suspend two jobs
  - sleep 60
  - Ctrl-F (or Ctrl-Z)
  - sleep 90
  - Ctrl-F (or Ctrl-Z)
- Use **jobs** to see them
- Use **ps –lf** to get their PIDs
- Resume one job with the **bg** command
- Resume the other job with the kill -18 signal
- Use **jobs** to see if they complete

vi practice

- Bring up the vi reference page at:
  http://simms-teach.com/docs/vi-ref.html

- Create a directory called *practice*
  mkdir practice

- Copy in sample text files
  cp /home/cis90ol/depot/* practice

vi

Practice using these commands

Note: to execute any of the following commands from vi, you must be in command mode. Press the Esc key to enter 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
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
^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

vi

Now practice these commands

Note: to execute any of the following commands from vi, you must be in command mode. Press the Esc key to enter command mode.

:q exits vi if you have saved your changes
:q! exits vi even if you have not saved your changes
:w saves any changes you've made to the file you are editing
: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

Now practice these commands

i Ready to insert characters immediately before the current cursor position
a Ready to append characters immediately after the current cursor position
I Ready to insert characters at the start of the current line
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
r Ready to replace the current character with the character you type next
R Ready to Replace (overwrite) characters starting at the curent cursor position
s Ready to replace the current character with the string you type next
cw Ready to replace the current word with the string you type next

vi

Now practice these commands

Note: to execute any of the following commands from vi, you must be in command mode. Press the Esc key to enter command mode.

x Deletes the current character
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

Now practice these commands

Note: to execute any of the following commands from vi, you must be in command mode. Press the Esc key to enter command mode.

^g Tells you the filename you are editing and what line your cursor is on
u Undoes the last command you executed
. 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