

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 -
-
- Backup headset charged - done
 - CCC Confer wall paper - done
 - Slides & Lab uploaded - done
 - Email tech to class, turn on link -



Dennis



Sean



Christopher



Francisco



Rich

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



Salena



Abd



Sarah



Astitow



Mike D.



Alex



Christine



Steven



Richie



Nathan



Tony



James G.



Sergio



Anthony



Fernando



Miguel



Lars



Jennifer



Rudy



Laura P.



Nick



Juan



Jacob



Andrew



Luke



Saulius

Online Class Students



Edtson



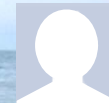
James B.



Liz



Casady



Jason



Dale



Aaron



Steve



Matt



Songul



Stephanie



Victor



Tanya



Mike P.



Adriana



Laura S.



Olivia



Janelle

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

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 is the difference between the fork and exec system calls?
3. What command shows the current running processes?

email answers to: risimms@cabrillo.edu

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



- [] Has the phone bridge been added?
- [] Is recording on?
- [] Does the phone bridge have the mike?
- [] Share slides, putty (rsimms, simmsben, roddyduk), and Chrome
- [] Disable spelling on PowerPoint

vi editor

Objectives	Agenda
<ul style="list-style-type: none">• Create and modify text files	<ul style="list-style-type: none">• Quiz• Questions from last week• Test results• 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

Test Results

Test 2 Results

Incorrect answer pareto

- 20 XXXXXXXXXXXXXXXXXXXXXXXXXXXX (27) [PT] [L9-99]
- 21 XXXXXXXXXXXXXXXXXXXXXXXXXXXX (25)
- 22 XXXXXXXXXXXXXXXXXXXXXXXXXXXX (23) [~PT]
- 07 XXXXXXXXXXXXXXXXXXXXXXXXXXXX (22) [~PT]
- 19 XXXXXXXXXXXXXXXXXXXXXXXXXXXX (22) [~PT]
- 12 XXXXXXXXXXXXXXXXXXXXXXXXXXXX (21) [L9-117]
- 13 XXXXXXXXXXXXXXXXXXXXXXXX (18) [~PT]
- 11 XXXXXXXXXXXXXXXXXXXXXXXX (17) [~PT]
- 14 XXXXXXXXXXXXXXXXXXXXXXXX (16) [~PT] [L9-50]
- 16 XXXXXXXXXXXXXXXXXXXXXXXX (16) [~PT]
- 17 XXXXXXXXXXXXXXXXXXXXXXXX (16) [~PT]
- 23 XXXXXXXXXXXXXXXXXXXXXXXX (16) [PT]
- 09 XXXXXXXXXXXXXXXXXXXXXXXX (15) [~PT]
- 15 XXXXXXXXXXXXXXXXXXXXXXXX (15) [PT] [L9-101]
- 18 XXXXXXXXXXXXXXXXXXXXXXXX (15) [PT]
- 24 XXXXXXXXXXXXXXXXXXXXXXXX (14) [~PT]
- 06 XXXXXXXXXXXXXXXXXXXXXXXX (13) [~PT]
- 10 XXXXXXXXXXXXXXXXXXXXXXXX (13) [~PT]
- 08 XXXXXXXXXXXXXXXXXXXXXXXX (12) [~PT]
- 04 XXXXXXXX (8) [F]
- 05 XXXXXXXX (8) [F]
- 25 XXXXXXXX (7) [PT]
- 02 XXXXXX (5) [F]
- 03 XXXXXX (5) [F]
- 01 XXX (3) [F]

Future Study Tips

- Take practice test and ask questions in class or on forum for any questions you don't understand
- Review Lesson slides, especially lessons that are reviews for an upcoming test
- Review labs - best way to really understand how things work.

PT - Same ? on practice text
~PT - Similar ? on practice test (modified)
L#-# - Similar ? covered in lesson
F - Flashcard ?

Test 2 Q20 (extra credit)

What complete command (with no ";"s) **counts** all the files belonging to you on the system, places a sorted list of them in the file *allmine*, and redirects error messages to the bit bucket?

Limits the files listed to just those owned by the user. The shell replaces \$LOGNAME with the actual username.

The tee sends the sorted files to both the file allmine and to the stdin of the wc command

```

find / -user $LOGNAME 2> /dev/null | sort | tee allmine | wc -l

```

find will list all files starting at / on the UNIX file tree

Permission errors are thrown away (from trying to list or traverse directories you don't have read or execute permission)

Test 2 Q21 (extra credit)

What are 2 files or directories on Opus that use inode number 1? Hint: see last slide in Lesson 10.

Answers: /proc , /sys, /dev/pts, /proc/sys/fs/binfmt_misc/status

```
/home/cis90/roddyduk $ ls -i /
2490369 bin      1474561 lib          11070 net          465 selinux    2523137 u
      2 boot          11 lost+found  720897 opt          393217 srv          1441793 usr
      942 dev      2129921 media          1 proc          1 sys          1507329 var
1277953 etc        11066 misc        2457601 root      2424833 tftpboot
1802241 home     1114113 mnt          1540097 sbin     1310721 tmp
/home/cis90/roddyduk $
```

```
/home/cis90/roddyduk $ ls -iLR / 2> /dev/null | grep "^ *1 "
1 -rw-r--r-- 1 root root 0 Sep 28 15:58 status
/home/cis90/roddyduk $ ls -iLR / 2> /dev/null | grep "^ *1 "
      1 dr-xr-xr-x 187 root root      0 Sep 28 15:58 proc
      1 drwxr-xr-x  11 root root      0 Sep 28 15:58 sys
      1 drwxr-xr-x  2 root root      0 Sep 28 15:58 pts
1 -rw-r--r-- 1 root root 0 Sep 28 15:58 status
```

To see the full path, use the -B <number> option on grep to list lines before the matched line

Test 2 Q22 (extra credit)

What complete command line (with ";"s) creates a file named hiro, changes that file's group to be users, then removes all permissions for others on the file?

Answer:

touch hiro; chgrp users hiro; chmod o-rwx hiro

```
/home/cis90/roddyduk $ touch hiro; chgrp users hiro; chmod o-rwx hiro
/home/cis90/roddyduk $ ls -l hiro
-rw-rw---- 1 roddyduk users 0 Nov  8 09:08 hiro
```

Test 2 Q7

If the contents of the file named states is:

Michigan
California
Ohio
New York

What is the result of doing the following command?

sort < states > states

Answer: The file states gets emptied!

```
/home/cis90/roddyduk $ cat states  
Michigan  
California  
Ohio  
New York  
/home/cis90/roddyduk $ sort < states > states  
/home/cis90/roddyduk $ cat states  
/home/cis90/roddyduk $
```

*This happens when bash hooks up the plumbing prior to the command being run. The > states will create if necessary and then empty the states file. This happens **before** sort runs.*

Test 2 Q19 (extra credit)

If the umask is 022 and the permissions on the file cinderella are rw--w--w-, then what would be the permissions of the file cinderella.bak after doing the following command?

cp cinderella cinderella.bak

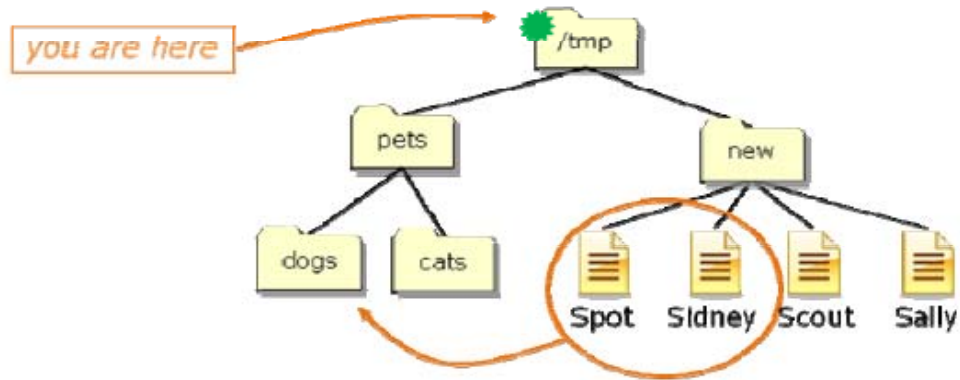
Answer: 600

```
rw- -w- -w- = 110 010 010 (permission on cinderella)
022          = 000 010 010 (mask)
-----
110 000 000 (mask applied)
 6   0   0  (permissions on cinderella.bak)
```

```
/home/cis90/roddyduk $ umask 022
/home/cis90/roddyduk $ touch cinderella
/home/cis90/roddyduk $ chmod 622 cinderella
/home/cis90/roddyduk $ cp cinderella cinderella.bak
/home/cis90/roddyduk $ ls -l cinder*
-rw--w--w- 1 roddyduk cis90 0 Nov  8 09:24 cinderella
-rw----- 1 roddyduk cis90 0 Nov  8 09:25 cinderella.bak
```

Test 2 Q12

12. Given this directory structure:



If your current working directory is */tmp*, what single command using filename expansion characters would move just the files *Spot* and *Sidney* to the *dogs* directory?

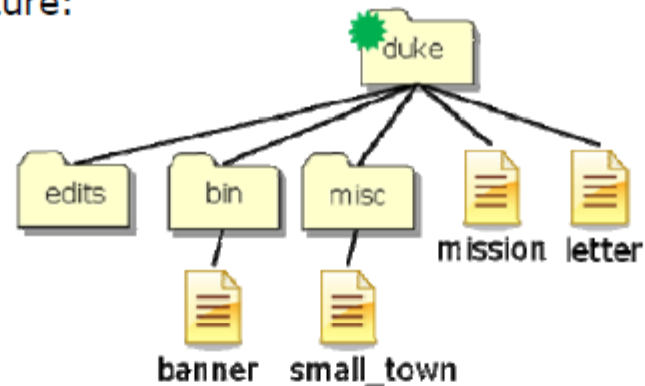
Answer: `mv new/S[pi]* pets/dogs`

```

/tmp $ ls pets/dogs
/tmp $ ls new/
Sally Scout Sidney Spot
/tmp $ mv new/S[pi]* pets/dogs
/tmp $ ls new
Sally Scout
/tmp $ ls pets/dogs/
Sidney Spot
  
```

Test 2 Q13

13. Given this directory structure:



If your current working directory is *duke*, what single command (no ";"s) would copy the *mission* and *banner* files to the */tmp* directory?

Answer: `cp mission bin/banner /tmp`

Test 2 Q11

11. Benji messed up his *edits* directory. His long listings now look like the following:

```
/home/cis90/simmsben $ ls -l edits
total 0
?----- ? ? ? ?      ? 1995.egg
?----- ? ? ? ?      ? better_town
?----- ? ? ? ?      ? small_town
?----- ? ? ? ?      ? spellk
?----- ? ? ? ?      ? text.err
?----- ? ? ? ?      ? text.fxd
/home/cis90/simmsben $
```

Note: No inode information is being displayed. It requires execute permission to access inode information.

What command should Benji issue so he can see all his file information again?

Answer: `chmod u+x edits`



Add execute permission to the user (owner) to allow user to access inode information

Test 2 Q11 (continued)

```

/home/cis90/roddyduk $ ls -l edits
total 40
-rw----r-- 1 roddyduk cis90 1382 Feb  1  2002 better_town
-rw----r-- 1 roddyduk cis90 1580 Nov 16  2004 small_town
-rw----r-- 1 roddyduk cis90  485 Aug 26  2003 spellk
-rw----r-- 1 roddyduk cis90  250 Jul 20  2001 text.err
-rw----r-- 1 roddyduk cis90  231 Jul 20  2001 text.fxd
/home/cis90/roddyduk $ chmod u-x edits
/home/cis90/roddyduk $ ls -l edits
total 0
?----- ? ? ? ?      ? better_town
?----- ? ? ? ?      ? small_town
?----- ? ? ? ?      ? spellk
?----- ? ? ? ?      ? text.err
?----- ? ? ? ?      ? text.fxd
/home/cis90/roddyduk $ chmod u+x edits
/home/cis90/roddyduk $ ls -l edits
total 40
-rw----r-- 1 roddyduk cis90 1382 Feb  1  2002 better_town
-rw----r-- 1 roddyduk cis90 1580 Nov 16  2004 small_town
-rw----r-- 1 roddyduk cis90  485 Aug 26  2003 spellk
-rw----r-- 1 roddyduk cis90  250 Jul 20  2001 text.err
-rw----r-- 1 roddyduk cis90  231 Jul 20  2001 text.fxd

```

Listing includes all information

Mess it up here

Now inode information is unavailable and shows as ?'s

Fix it here

Test 2 Q14

14. From your home directory, what single command line **using at least one filename expansion character** and an **environment variable**, could be used to mail yourself and rsimms, a count of the misspelled words in all of Yeat's poems, with a subject of "Yeats"?

Answer:

```
spell poems/Yeats/* | wc -l | mail -s "Yeats" $LOGNAME rsimms
```

*filename
expansion
character*

a count

*environment
variable*

Test 2 Q14 (continued)

```
/home/cis90/roddyduk $ spell poems/Yeats/* | wc -l | mail -s "Yeats" $LOGNAME rsimms
/home/cis90/roddyduk $ mail
Mail version 8.1 6/6/93.  Type ? for help.
"/var/spool/mail/roddyduk": 1 message 1 new
>N 1 roddyduk@Opus.cabril  Mon Nov  8 10:24  16/616  "Yeats"
&
Message 1:
From roddyduk@Opus.cabrillo.edu  Mon Nov  8 10:24:31 2010
Date: Mon, 8 Nov 2010 10:24:31 -0800
From: Duke Roddy <roddyduk@Opus.cabrillo.edu>
To: roddyduk@Opus.cabrillo.edu, rsimms@Opus.cabrillo.edu
Subject: Yeats

3

& quit
Saved 1 message in mbox
/home/cis90/roddyduk $
```

Test 2 - Q16

Extra Credit (1 point each)

16. You have been hired as a parser. You will need to parse command lines and identify the command, options, arguments and any redirection. This includes doing any filename expansion. Parse the command below and complete the section that follows:

stat -tZ /usr/bin/sta*[ts] > myreport

command:	stat
option(s):	-tZ
argument(s):	/usr/bin/stap-report
	/usr/bin/stat
	/usr/bin/states
redirection:	stdout redirected to myreport

```
/home/cis90/roddyduk $ echo /usr/bin/sta*[ts]  
/usr/bin/stap-report /usr/bin/stat /usr/bin/states
```

Test 2 - Q16 (continued)

Extra Credit (1 point each)

16. You have been hired as a parser. You will need to parse command lines and identify the command, options, arguments and any redirection. This includes doing any filename expansion. Parse the command below and complete the section that follows:

stat -tZ /usr/bin/sta*[ts] > myreport

command:	stat
option(s):	-tZ
argument(s):	/usr/bin/stap-report
	/usr/bin/stat
	/usr/bin/states
redirection:	stdout redirected to myreport

```
/home/cis90/roddyduk $ echo /usr/bin/sta*[ts]
/usr/bin/stap-report /usr/bin/stat /usr/bin/states
```

Test 2 - Q16 (continued)

Extra Credit (1 point each)

16. You have been hired as a parser. You will need to parse command lines and identify the command, options, arguments and any redirection. This includes doing any filename expansion. Parse the command below and complete the section that follows:

`stat -tZ /usr/bin/sta*[ts] > myreport`

command:	<code>stat</code>
option(s):	<code>-tZ</code>
argument(s):	<code>/usr/bin/stap-report</code>
	<code>/usr/bin/stat</code>
	<code>/usr/bin/states</code>
redirection:	<code>stdout</code> redirected to <code>myreport</code>

```
/home/cis90/roddyduk $ echo /usr/bin/sta*[ts]
/usr/bin/stap-report /usr/bin/stat /usr/bin/states
```

Test 2 - Q16 (continued)

Extra Credit (1 point each)

16. You have been hired as a parser. You will need to parse command lines and identify the command, options, arguments and any redirection. This includes doing any filename expansion. Parse the command below and complete the section that follows:

stat -tZ /usr/bin/sta*[ts]>myreport

command:	<u>stat</u>
option(s):	<u>-tZ</u>
	<u></u>
	<u></u>
argument(s):	<u>/usr/bin/stap-report</u>
	<u>/usr/bin/stat</u>
	<u>/usr/bin/states</u>
redirection:	<u>stdout</u> redirected to <u>myreport</u>

```
/home/cis90/roddyduk $ echo /usr/bin/sta*[ts]
/usr/bin/stap-report /usr/bin/stat /usr/bin/states
```


Test 2 - Q16 (continued)

Extra Credit (1 point each)

16. You have been hired as a parser. You will need to parse command lines and identify the command, options, arguments and any redirection. This includes doing any filename expansion. Parse the command below and complete the section that follows:

stat -tZ /usr/bin/sta*[ts] > myreport

command:	stat
option(s):	-tZ
argument(s):	/usr/bin/stap-report
	/usr/bin/stat
	/usr/bin/states
redirection:	stdout redirected to myreport

```
/home/cis90/roddyduk $ echo /usr/bin/sta*[ts]
/usr/bin/stap-report /usr/bin/stat /usr/bin/states
```

Test 2 Q17

17. On Opus, how many directories and sub-directories are in the /lib portion of the file tree?

Answer: 332

```
/home/cis90/simmsben $ find /lib -type d | wc -l  
332
```

Start listing files in /lib

Only show directories

Count them

Test 2 23

23. Using only **echo** commands and file redirection, how could you create a file named *characters* so it would contain the following lines?

Hiro
Ando
Sylar
Nikki

Alternate answer:

```
echo Hiro > characters  
echo Ando >> characters  
echo Sylar >> characters  
echo Nikki >> characters
```

```
/home/cis90/simmsben $ echo Hiro > characters  
/home/cis90/simmsben $ echo Ando >> characters  
/home/cis90/simmsben $ echo Sylar >> characters  
/home/cis90/simmsben $ echo Nikki >> characters  
/home/cis90/simmsben $ cat characters  
Hiro  
Ando  
Sylar  
Nikki
```

Test 2 Q23 Continued

23. Using only **echo** commands and file redirection, how could you create a file named *characters* so it would contain the following lines?

Hiro
Ando
Sylar
Nikki

Answer:

```
echo "Hiro  
Ando  
Sylar  
Nikki" > characters
```

```
/home/cis90/simmsben $ echo "Hiro  
> Ando  
> Sylar  
> Nikki" > characters  
/home/cis90/simmsben $ cat characters  
Hiro  
Ando  
Sylar  
Nikki
```

Test 2 Q23 Continued

23. Using only **echo** commands and file redirection, how could you create a file named *characters* so it would contain the following lines?

Hiro
Ando
Sylar
Nikki

Alternate answer:

echo -e "Hiro\nAndo\nSylar\nNikki" > characters

```
/home/cis90/simmsben $ echo -e "Hiro\nAndo\nSylar\nNikki" > characters
/home/cis90/simmsben $ cat characters
Hiro
Ando
Sylar
Nikki
```

Test 2 Q9 Continued

9. What are the (numeric) permissions on the /usr/bin file with the inode rsimms emailed you?

Answer: 755

Message 61:

```
From rsimms@Opus.cabrillo.edu Fri Nov 5 22:47:14 2010
Date: Fri, 5 Nov 2010 22:47:14 -0700
From: Rich Simms <rsimms@Opus.cabrillo.edu>
To: simmsben@Opus.cabrillo.edu
Subject: Test 2 - Question 9
```

Hello Benji,

Please use inode 1452431 for question 9 on the test.

-Rich

```
~/cis90/simmsben $ ls -il /usr/bin | grep 1452431
-1452431 -rwxr-xr-x 1 root root 26380 Jul 13 2009 pinky
~/cis90/simmsben $
```

Test 2 Q15

15. Given the file *expressions* contains these two lines:

```
5+5
9/0
```

What complete command using **bc** would input the math problems in *expressions*, **append** the calculated answers to the file *results* and write any errors to the file *errors*?

stdin redirected
from keyboard to
file *expressions*

stderr redirected
from terminal to
file *errors*

```
bc < expressions >> results 2> errors
```

stdout redirected
from terminal to
append to file *results*

Test 2 Q15 verification

```
/home/cis90/roddyduk $ echo 5+5 > expressions
/home/cis90/roddyduk $ echo 9/0 >> expressions
/home/cis90/roddyduk $ bc < expressions >> results 2> errors
/home/cis90/roddyduk $ cat results errors
10
Runtime error (func=(main), adr=5): Divide by zero
/home/cis90/roddyduk $
```


Test 2 Q18

18. **cd** into `/home/cis90/roddyduk`, why does

find -name treat?

only find one treat file, yet

find -name 'treat?'

finds multiple treat files?

Answer:

In the first example bash expands **treat?** to **treat1** and passes **treat1** as an argument to the find command. Because find only looks for **treat1**, that is all it finds.

In the second example bash doesn't expand **'treat?'** and passes **treat?** as an argument to the find command. Because find is now using a wildcard (**treat?**) it finds multiple treat files.

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.e15  
samba-client-3.0.28-1.e15_2.1  
samba-3.0.28-1.e15_2.1  
samba-common-3.0.28-1.e15_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/cis90/simmsben $ cat /etc/passwd | grep :90: | wc -l  
54
```

```
/home/cis90/simmsben $ cat /etc/passwd | grep cis90 | wc -l  
54
```

*There are 54. The cis90 group is GID 90, the home directories are /home/cis90/**

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

```

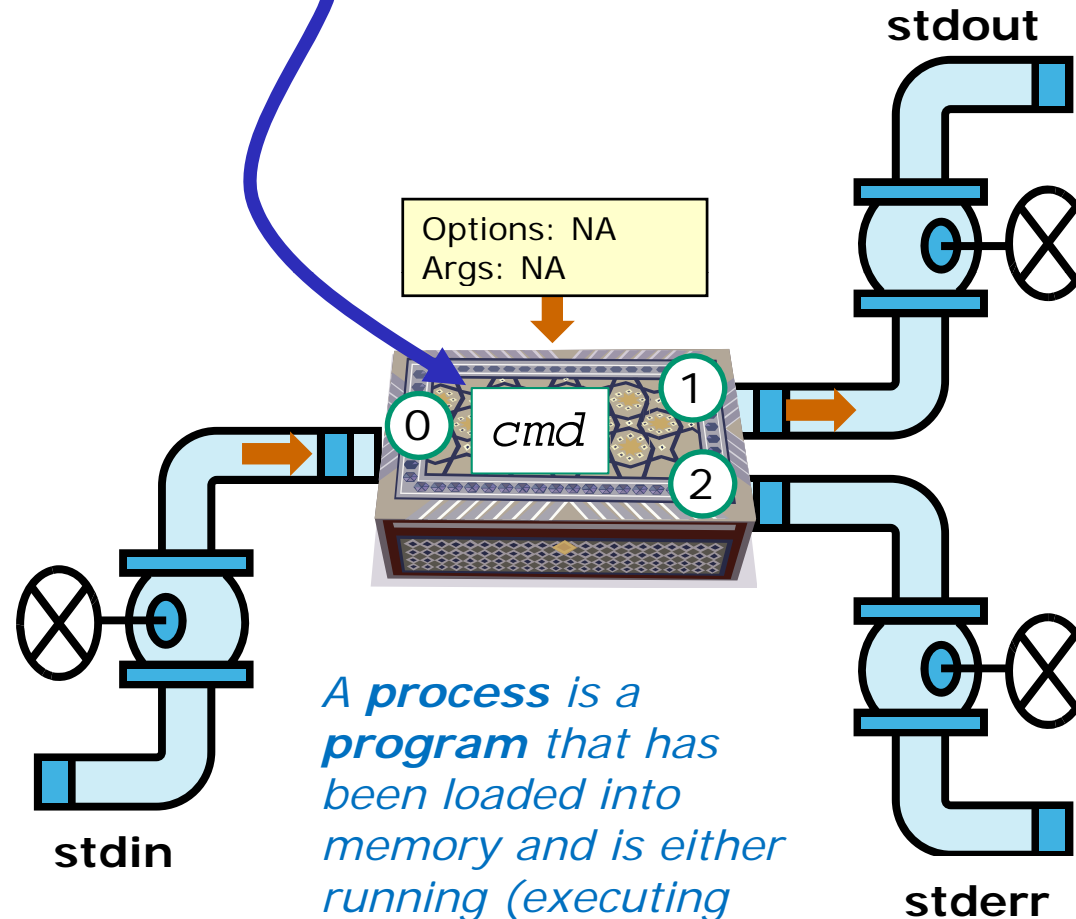
grep practice

- How many CIS191 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

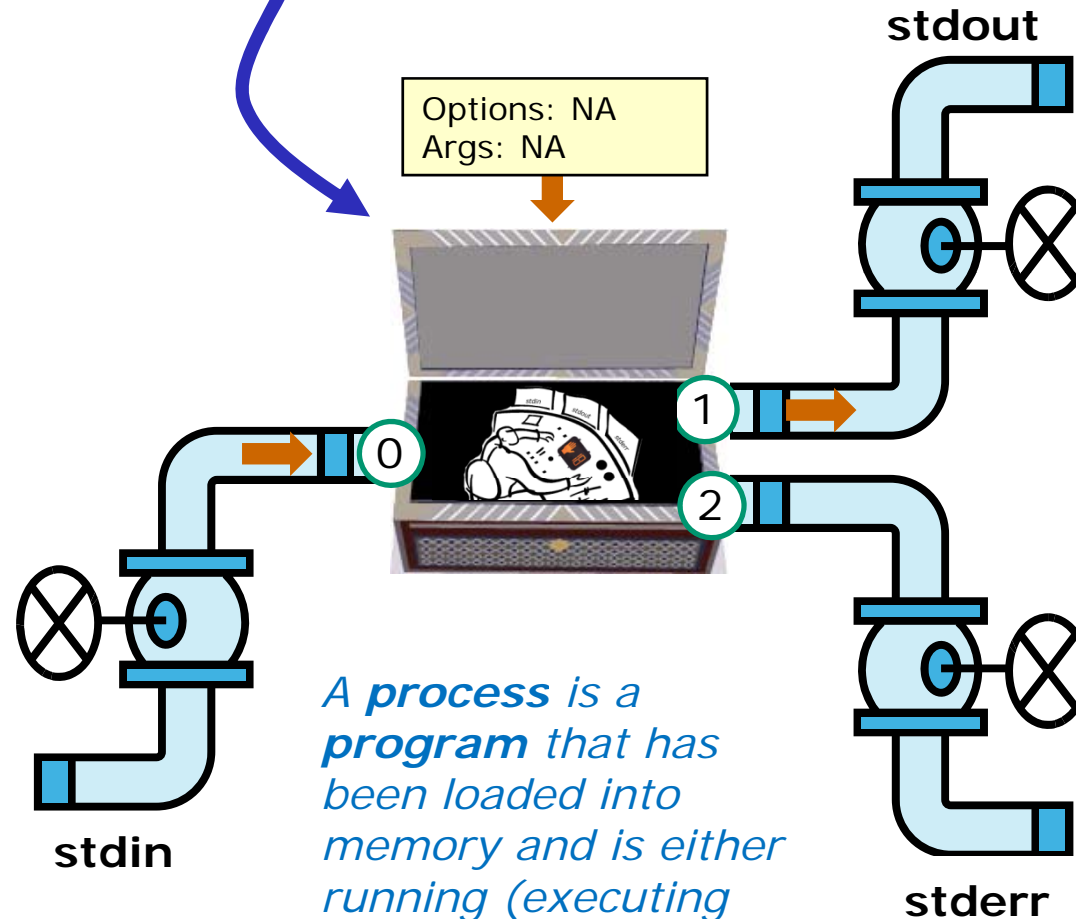
```
/home/cis90/roddyduk $cmd
```



*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 $
    
```



/dev/pts/0

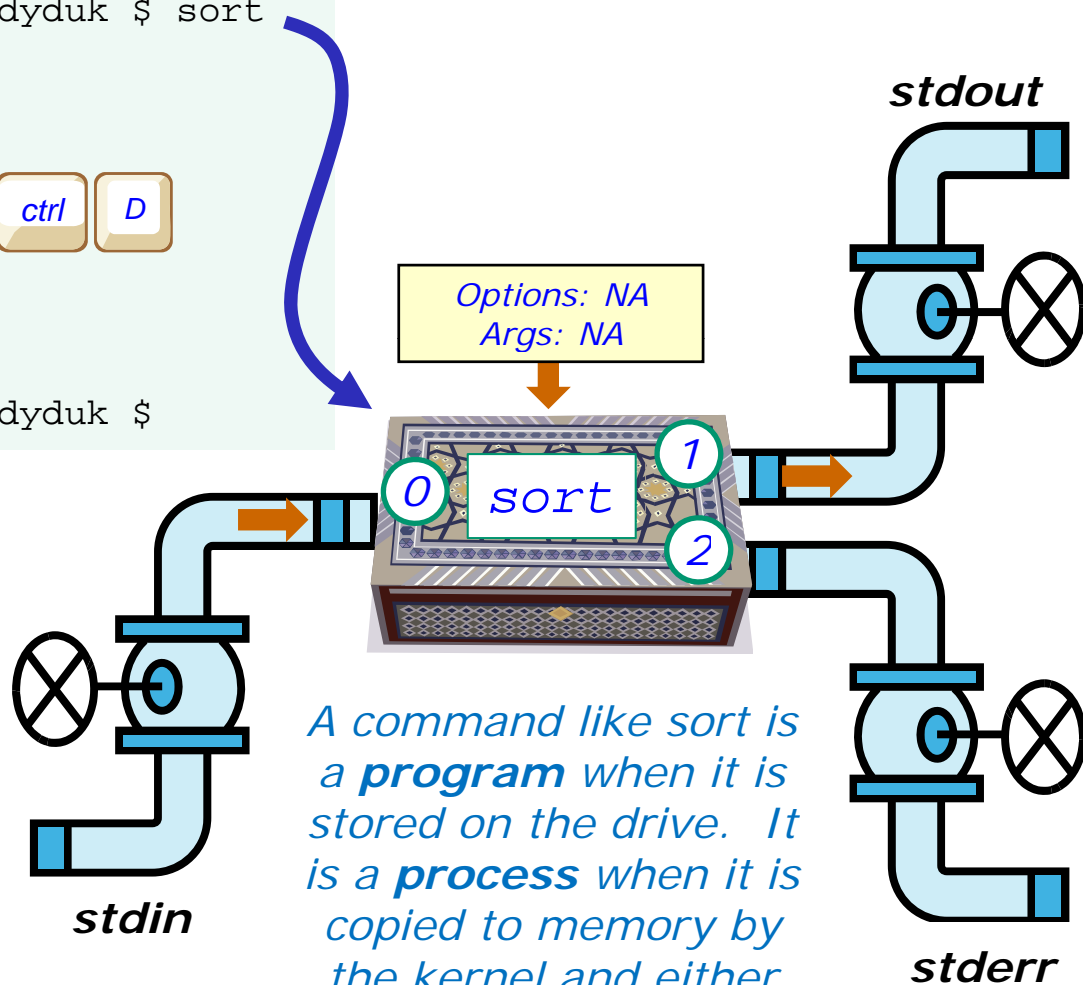


benji
duke
homer
star

/dev/pts/0



duke
benji
star
homer

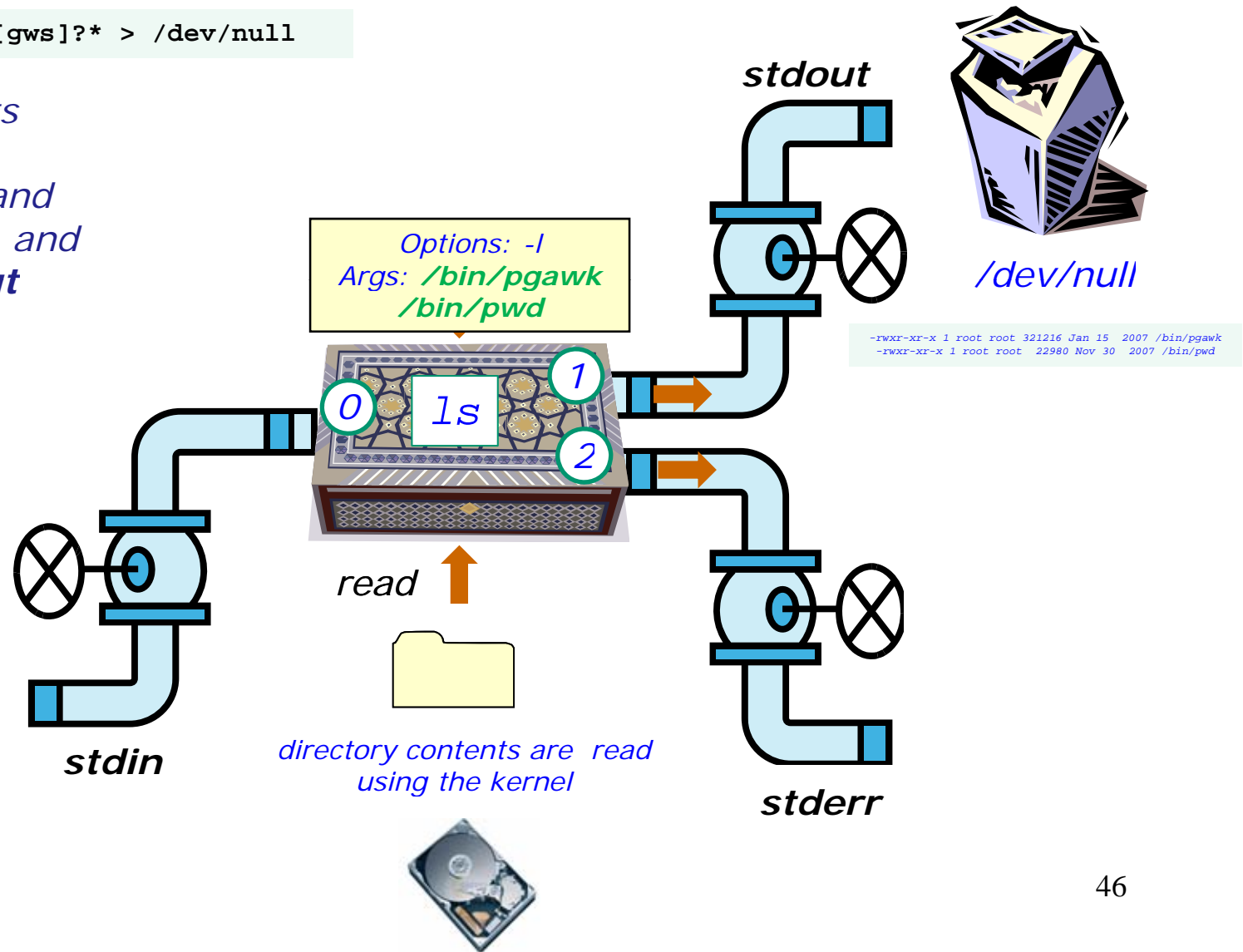


A command like sort is a **program** when it is stored on the drive. It is a **process** when it is copied to memory by the kernel and either running or waiting to run.

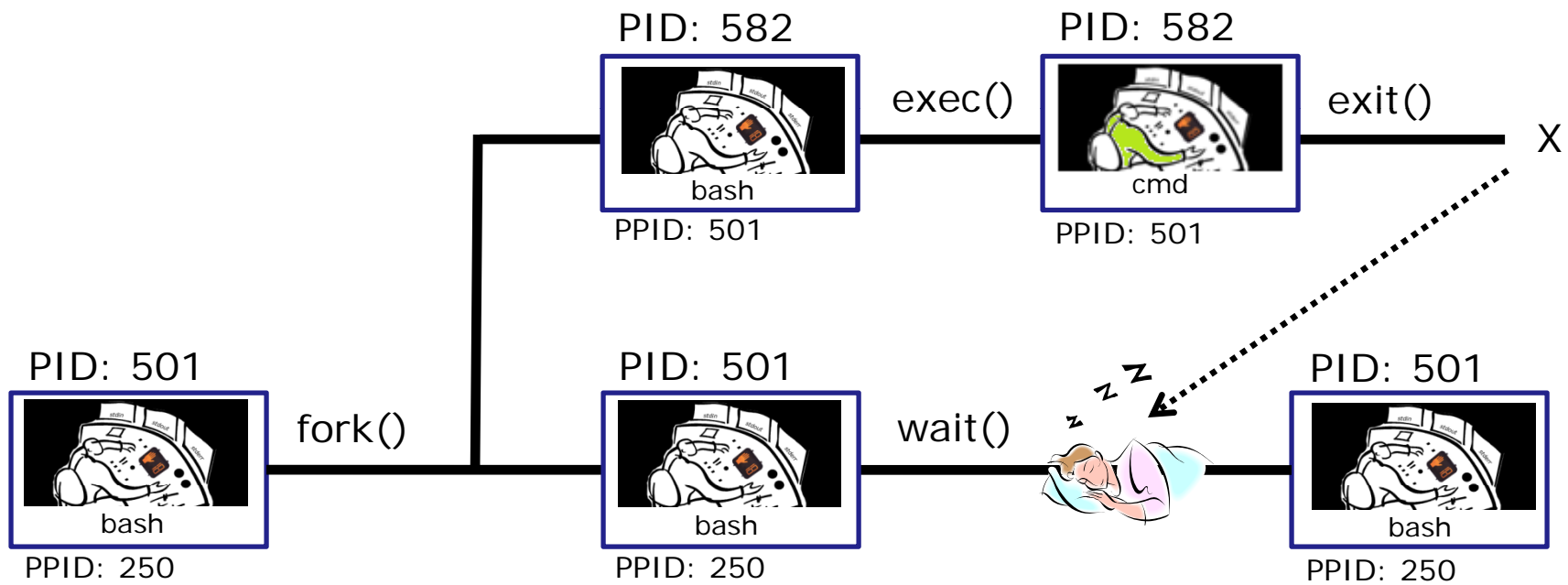
example program to process

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

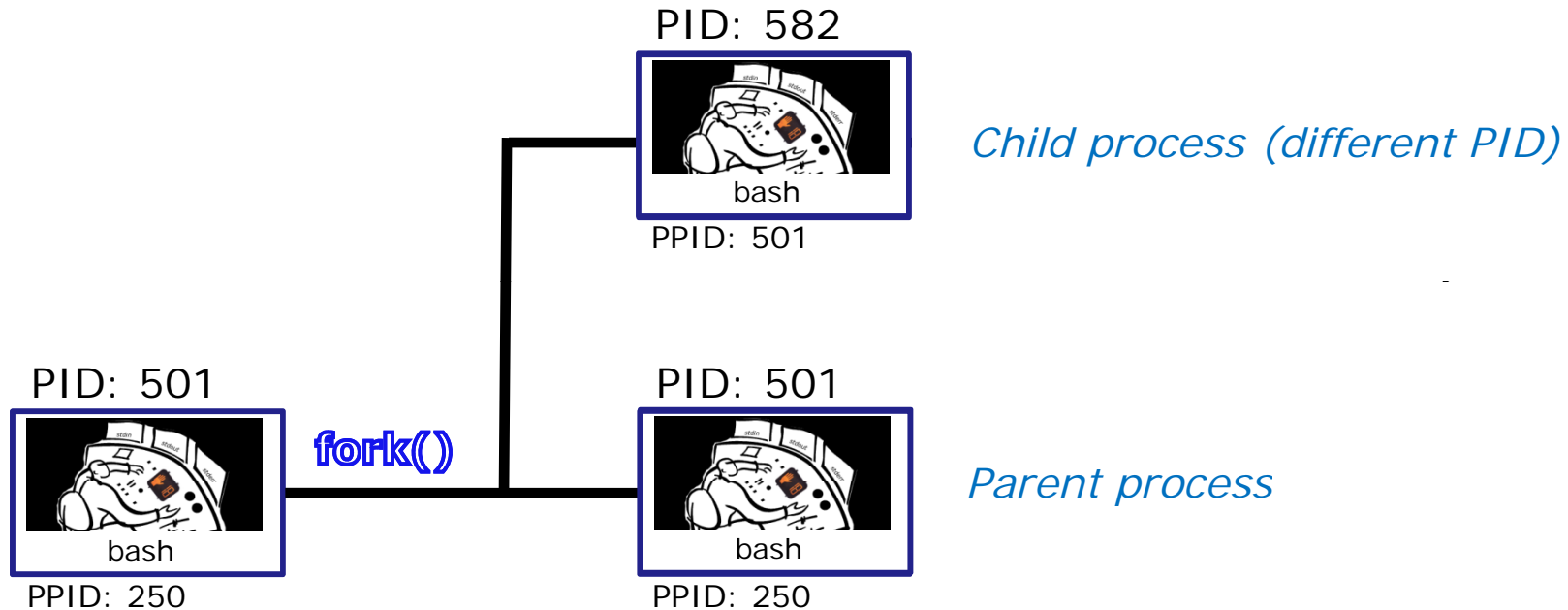
Note: *ls* gets its input from the command line and the OS (kernel) and writes to **stdout** (redirected to **/dev/null**) and **stderr**.



Process Lifecycle



Process Lifecycle

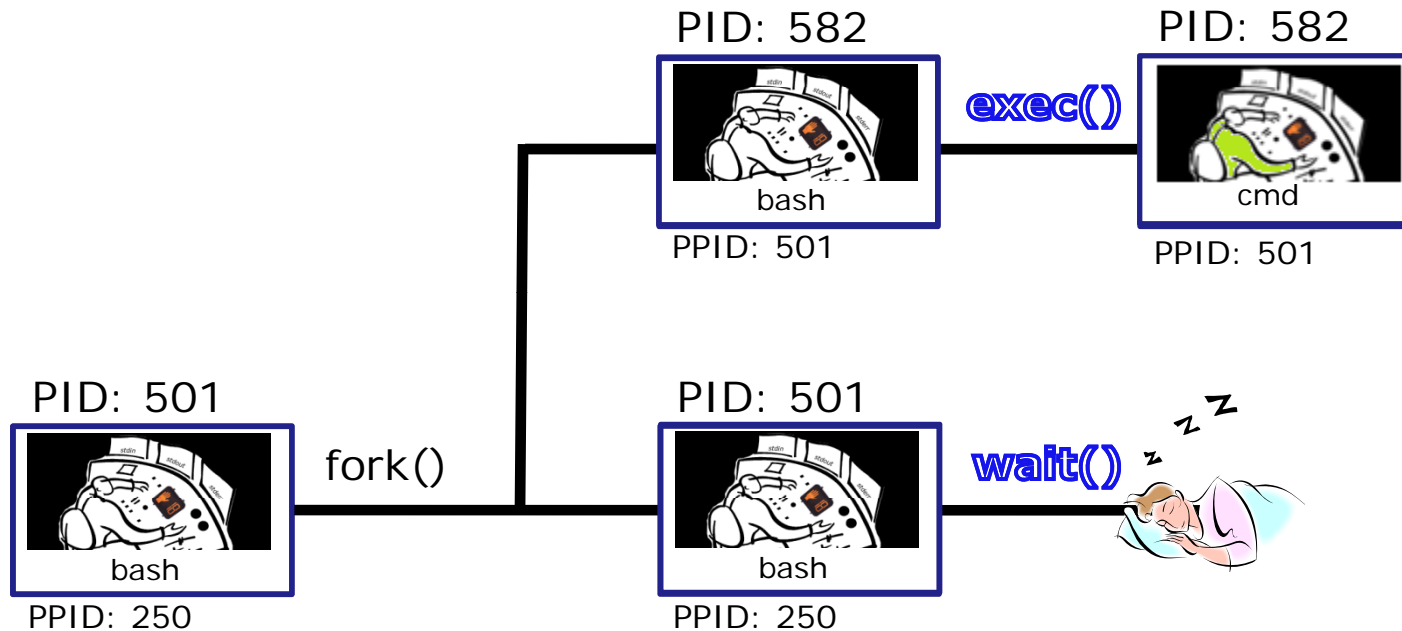


1) When a program is loaded into memory a new process must be created.

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.

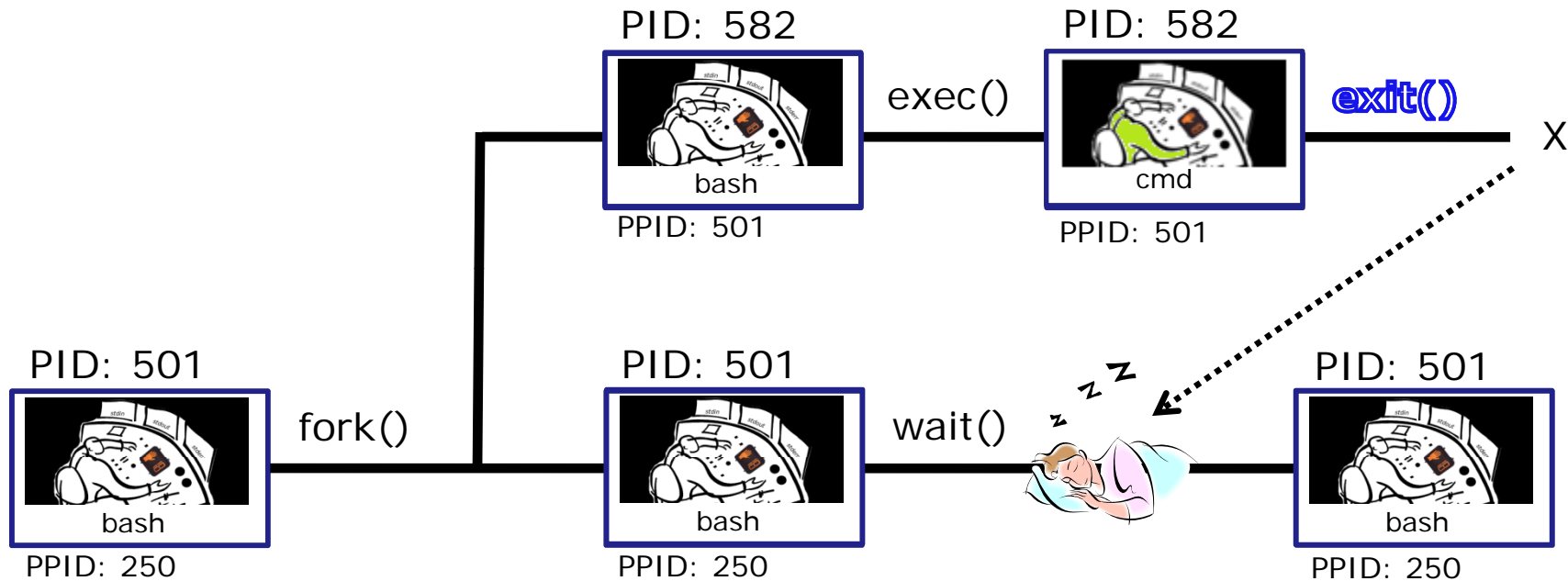
Process Lifecycle



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.

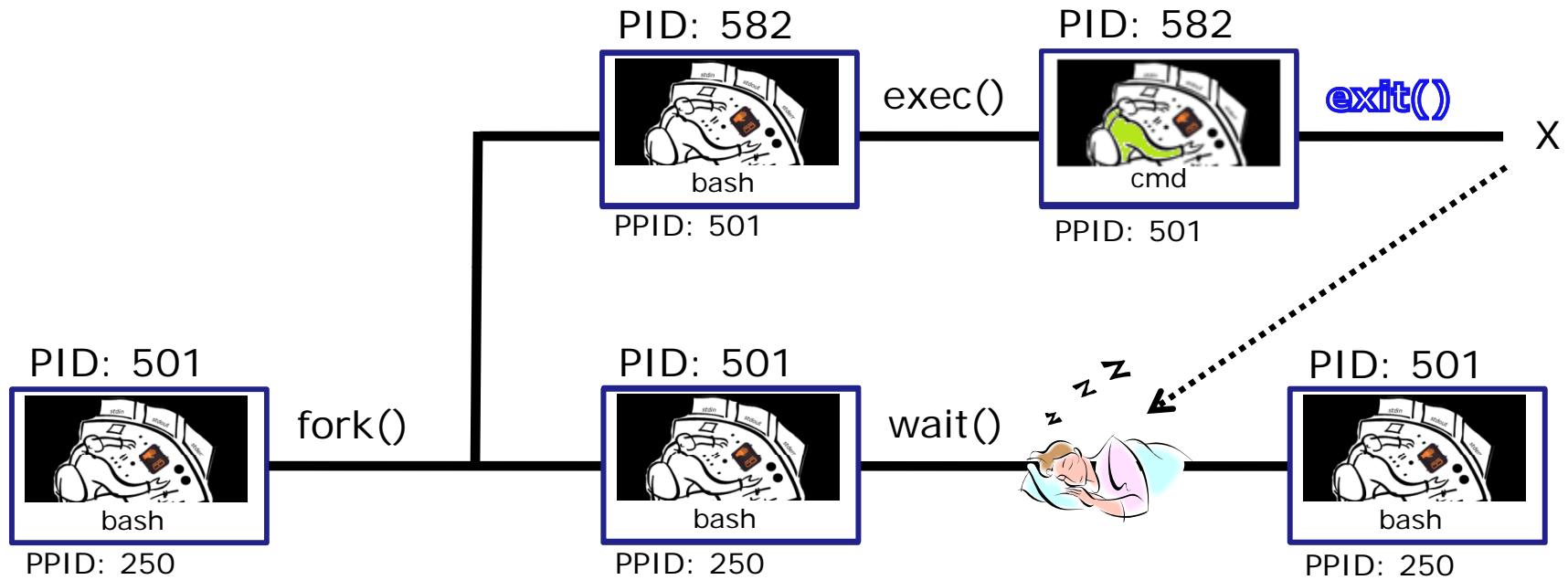
Process Lifecycle



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**.

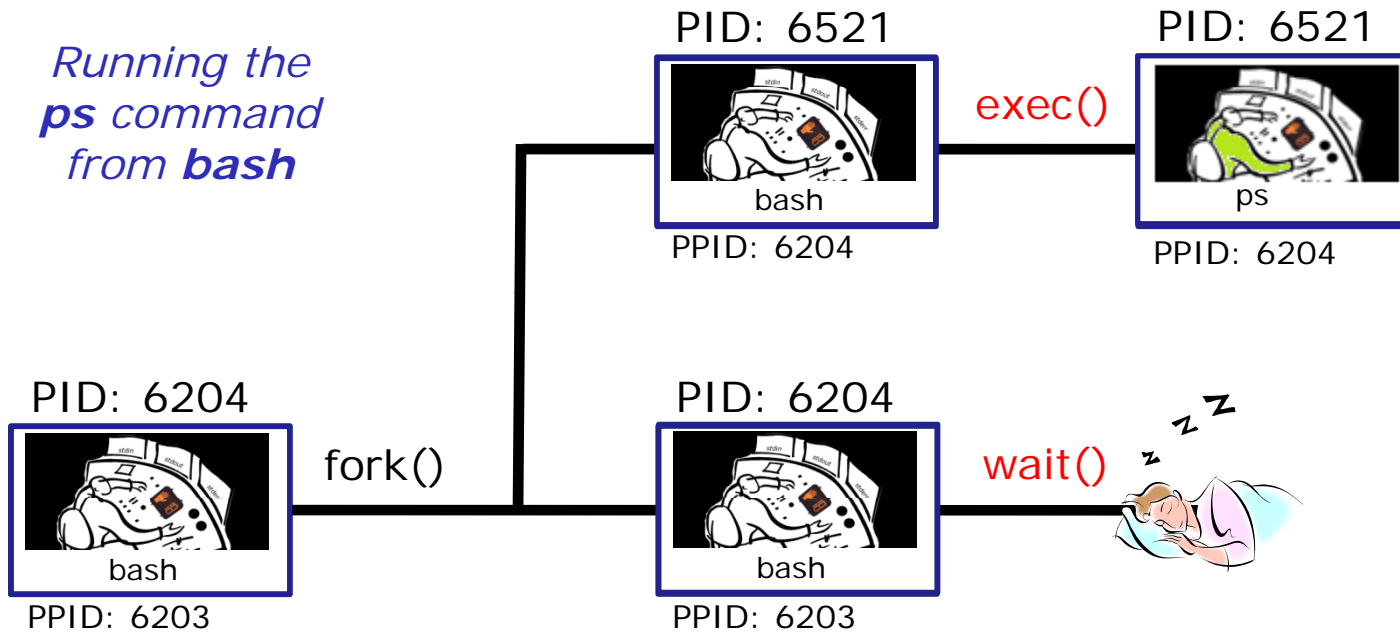
Process Information

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



```
[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?

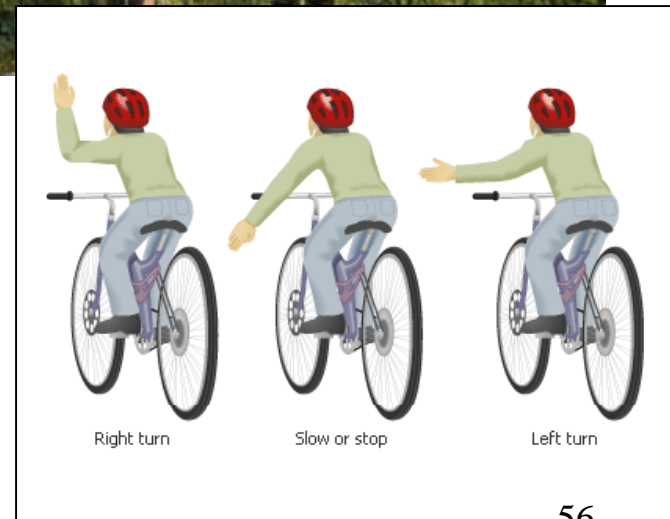
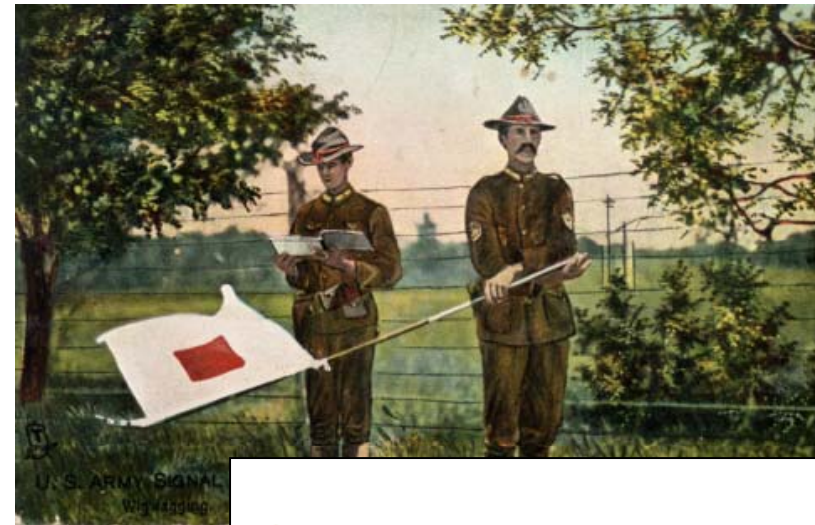
Review of Signals

Signals

PLATE 4

COMMERCIAL CODE SIGNALS		
<p>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.</p>		
URGENT & IMPORTANT SIGNALS		COMPASS SIGNALS 3 FLAGS
CODE FLAG OVER 1 FLAG OR 2 FLAG SIGNALS		
CODE FLAG P	A C	A Q E
"I Am about to Sail"	"Do Not" "abandon the Vessel"	N 1/2 E S 3/4 W
LATITUDE & LONGITUDE SIGNALS		CODE FLAG OVER 2 FLAGS
CODE FLAG A O	OR H X	CODE FLAG E H
12° Latitude North Latitude	General Signal	OR Y Z
		23° Longitude East Longitude
NUMERAL TABLE	GENERAL VOCABULARY	GEOGRAPHICAL SIGNALS ALPHABETICAL ORDER
CODE FLAG UNDER 2 FLAGS Y S	3 FLAG SIGNAL I X K	4 FLAG SIGNAL A E Y Z
CODE FLAG 10,000	Tons of Coal	Glasgow, Scotland.
ALPHABETICAL SPELLING TABLE		NAMES OF VESSELS FROM CODE LIST
J O H N	4 FLAG SIGNALS C B D N	4 FLAG SIGNAL H C L B
John	Abb ott	Glasgow of Glasgow 1058 Tons No 52636

JAMES BROWN & SON GLASGOW



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:



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 demonstrate 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 Ctrl-Z (to stop foreground 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 65 All

```

Jim's app script

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

```

rsimms@opus:/home/cis90/depot
#!/bin/sh
#
# app - script to demonstrate 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 foreground 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 66 All

```

Jim's app script

*Signal 15's
close
gracefully*

```

rsimms@opus:/home/cis90/depot
#!/bin/sh
#
# app - script to demonstrate 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 Ctrl-Z (to stop foreground 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 67 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 demonstrate 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 Ctrl-Z (to stop foreground 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 68 All

Jim's app script

```

rsimms@opus:/home/cis90/depot
#!/bin/sh
#
# app - script to demonstrate 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 Ctrl-Z (to stop foreground 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 69 All

```

*Endless
loop*

Signals

Benji runs app

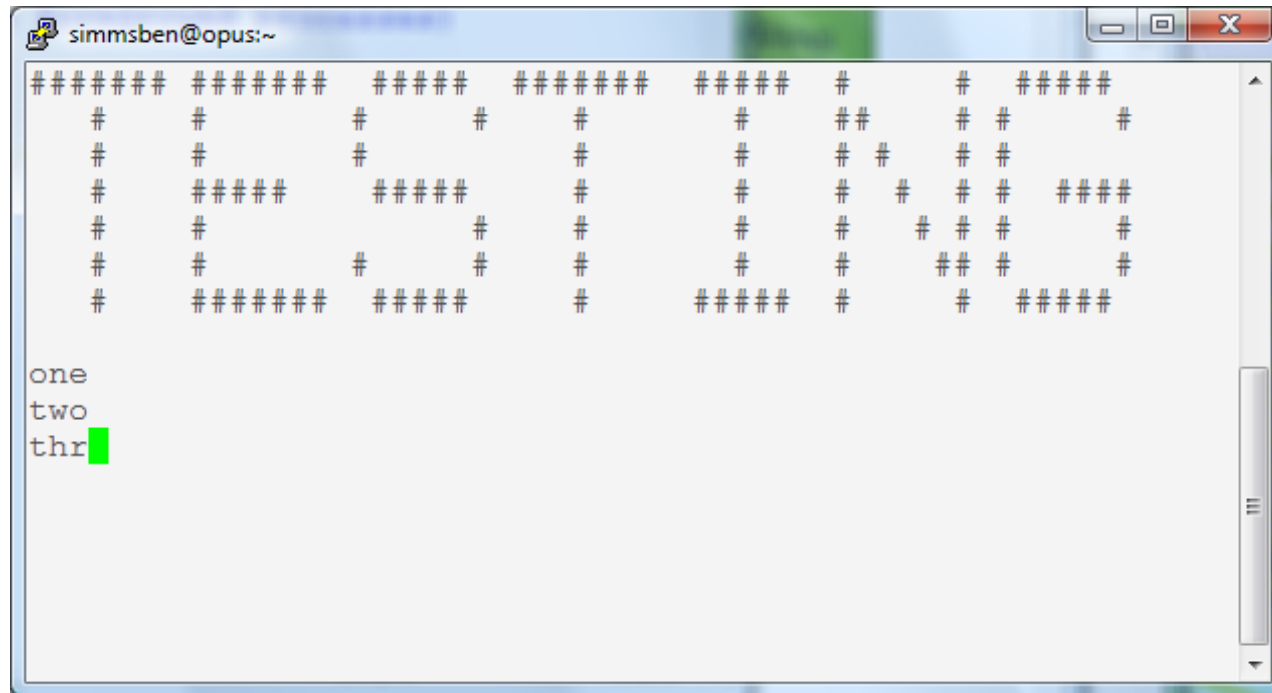


```
simmsben@opus:~  
##### # # # # # # # # # # # # # #  
# # # # # # # # # # # # # # # # # # # #  
# # # # # # # # # # # # # # # # # # # #  
# # # # # # # # # # # # # # # # # # # #  
# # # # # # # # # # # # # # # # # # # #  
# # # # # # # # # # # # # # # # # # # #  
# # # # # # # # # # # # # # # # # # # #  
  
one  
two  
thr █
```

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

Signals

Benji runs app



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

Signals

Benji runs app



```
simmsben@opus:~  
#####  #####  #####  #####  #####  #  #  #####  
#  #  #  #  #  #  ##  #  #  #  
#  #  #  #  #  #  #  #  #  #  
#  #####  #####  #  #  #  #  #  #####  
#  #  #  #  #  #  #  #  #  #  
#  #  #  #  #  #  #  #  ##  #  #  
#  #####  #####  #  #####  #  #####
```

one
two
thrQuit
quit it! █

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

Signals

Benji runs app



```

rododyduk@opus:~
/home/cis90/rododyduk $ 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/rododyduk $ kill 7065
-bash: kill: (7065) - Operation not permitted
/home/cis90/rododyduk $ █

```

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

The image shows two terminal windows. The top window displays a grid of '#' characters and the text 'one', 'two', 'thrQuit', and 'quit it!'. The bottom window shows the output of the 'ps -u simmsben' command, with the process '7065 pts/2 00:00:00 app' highlighted by a red box. Below that, the command 'kill -2 7065' is entered and also highlighted by a red box.

```

simmsben@opus:~
#####  #####  #####  #####  #####  #  #  #####
#      #      #      #      #      #      ##  #      #
#      #      #      #      #      #      #  #      #
#      #####  #####  #      #      #      #      #####
#      #      #      #      #      #      #      #      #
#      #      #      #      #      #      #      #      #
#      #####  #####

one
two
thrQuit
quit it!

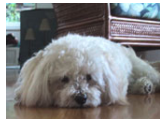
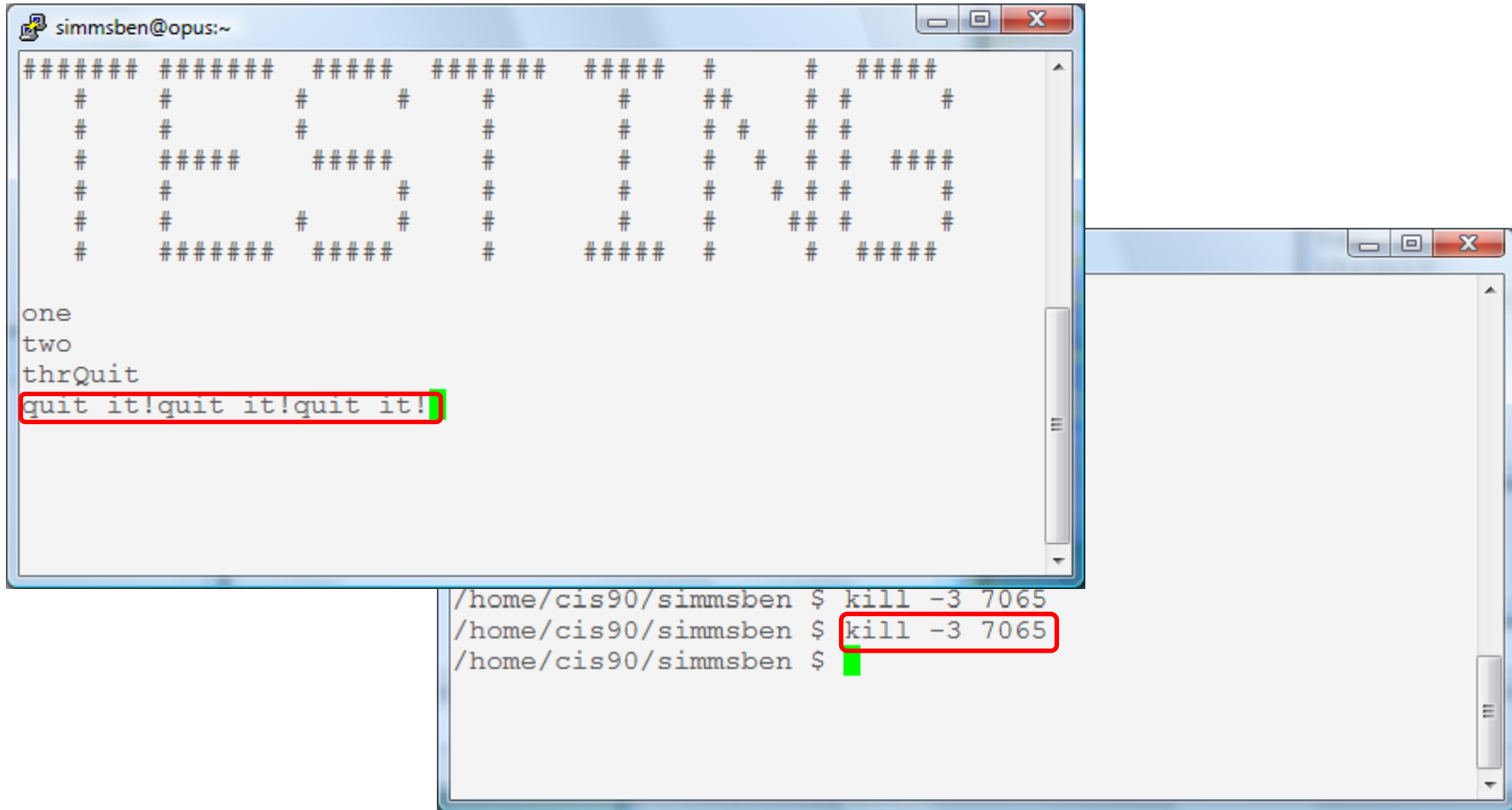
/home/cis90/simmsben $ ps -u simmsben
  PID TTY          TIME CMD
 6657 ?            00:00:00 sshd
 6658 pts/1        00:00:00 bash
 7033 ?            00:00:00 sshd
 7034 pts/2        00:00:00 bash
 7065 pts/2        00:00:00 app
 7843 pts/2        00:00:00 sleep
 7844 pts/1        00:00:00 ps
/home/cis90/simmsben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



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

Signals

Benji runs app



Benji ups the ante and sends several SIGQUITs but the app process shrugs them off with two "quit it!" messages ⁷⁵

Signals

Benji runs app

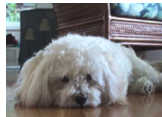
```

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 $ █

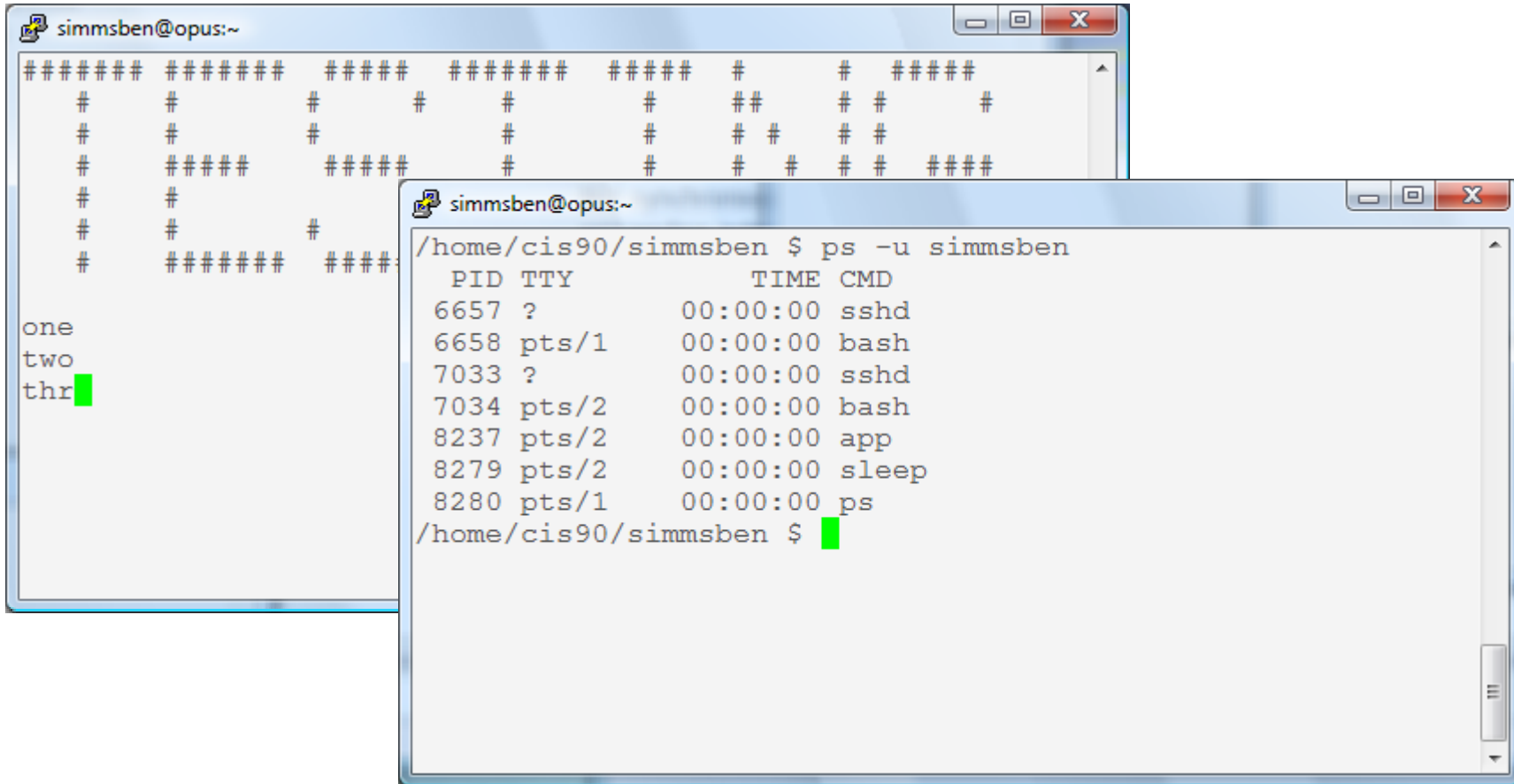
```



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

Signals

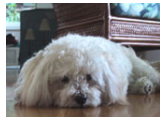
Benji runs app



The image shows two terminal windows. The top window displays a signal test where the user enters 'one', 'two', and 'thr'. The bottom window shows the output of the 'ps -u simmsben' command, listing running processes.

```
#####  
# # # # # # # # # #  
# # # # # # # # # #  
# #####  
# #  
# # #  
# #####  
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:~$
```

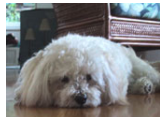
#####	#####	#####	#####	#####	#####	#####
#	#	#	#	#	#	#
#	#	#	#	#	#	#
#	#####	#####	#	#	#	#####
#	#	#	#	#	#	#
#	#	#	#	#	#	#
#	#####	#####	#	#	#	#

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

```
simmsben@opus:~$ 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

& 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

The screenshot shows a Linux desktop environment. In the foreground, a terminal window titled 'cis90@eko: ~' is open. The command 'firefox' is entered at the prompt and is highlighted with a red box. To the right, a Mozilla Firefox browser window titled 'Ubuntu Start Page - Mozilla Firefox' is open, displaying the Ubuntu Start Page. The browser's address bar shows 'http://start.ubuntu.com/10.04/'. The desktop background is a dark purple color. The system tray at the bottom shows the terminal window, the browser window, and the Update Manager icon.

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: ~  
File Edit View Terminal Help  
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.

Ubuntu Start Page - Mozilla Firefox
http://start.ubuntu.com/1
Most Visited Getting Started Latest Headlines
ubuntu
Google
Search
Done

Job Control Managing jobs

```
/home/cis90/simmsben $ sleep 120
```

```
Ctrl-Z or Ctrl-F (to suspend process)
```

```
[1]+ Stopped
```

```
sleep 120
```

```
/home/cis90/simmsben $ sleep 110
```

```
Ctrl-Z or Ctrl-F (to suspend process)
```

```
[2]+ Stopped
```

```
sleep 110
```

```
/home/cis90/simmsben $ sleep 100
```

```
Ctrl-Z or Ctrl-F (to suspend process)
```

```
[3]+ Stopped
```

```
sleep 100
```

```
/home/cis90/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/cis90/simmsben $ bg 2
[2]- sleep 110 &
/home/cis90/simmsben $ jobs
[1]- Stopped
sleep 120
[2] Running
sleep 110 &
[3]+ Stopped
sleep 100
/home/cis90/simmsben $ bg 1
[1]- sleep 120 &
/home/cis90/simmsben $ bg 3
[3]+ sleep 100 &
/home/cis90/simmsben $ jobs
[1] Running
sleep 120 &
[2]- Running
sleep 110 &
[3]+ Running
sleep 100 &
/home/cis90/simmsben $
```

*Suspended jobs can
be listed and
selectively resumed
using an argument
on the **bg** command*

Job Control

- Run and suspend two jobs
sleep 75
Ctrl-F or Ctrl-Z
sleep 85
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 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/cis90/depot/* practice

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

O (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*⁹³

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

O (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

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

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

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 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

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

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

. 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

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

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

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

Order now

Hydration harmony

[See mug text](#)

Copyright

Done

http://nostarch.com/mug.htm

Big Mug Label - Mozilla Firefox
http://nostarch.com/mug_big.htm

NO STARCH PRESS
"the finest in geek entertainment"™

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

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

THE MUG OF VI NO STARCH PRESS	FILE COMMANDS		DELETING / INSERTING TEXT		0	go to beginning of line (zero)
	vi filename(s)	edit a file or files	de, dd, x	delete word, line, character), (move to next, previous sentence
	vi -x filename	retrieve saved file after crash	n dd, n x	delete n lines, n characters	}, {	move forward, back one paragraph
	ZZ, :wq, :x	save and exit	x, X	delete character forward, backward	w, b	go forward, back one word
	:q, :q!	quit; quit without saving	D, d\$	delete to end of line	e	go to end of current or next word
	:w, :w fn	save file, save file as fn	dmotion	delete from cursor to motion (\$, 0, etc.)	CUT / COPY / PASTE	
	:e filename	edit filename			yy, nY	copy n lines
	:r filename	insert filename			yw, yy	copy word, line
	:sh	drop to shell			p, P	paste text after, before cursor
	:!cmd	run command cmd			a, i	insert text after, before cursor
	:r !cmd	execute cmd and insert output			A, I	insert text end, beginning of line
	SEARCH AND REPLACE		MOVING AROUND		WICKED COOL STUFF	
	/txt, ?txt	find txt forward or backward	.	undo last change	~	change case
	/*txt	find next line that starts with txt	u	repeat last change	xp	transpose characters
	n, N	repeat last search backward, forward	h, l, k, j	left or right n words	J	combine current line with next
	R	replace text from current character	nb, nW	back, forward one screen	mp	create a mark called p
			CTRL-U, D	up, down one screen	~p	return to p
			\$, G	go to end of line, end of file	d'x, y'x	delete, copy text from mark to cursor
					:> n	indent n lines

Done

/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

```
/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.cabrill 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
```

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

aspell list mimicks the standard unix spell program, roughly.

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

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

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

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

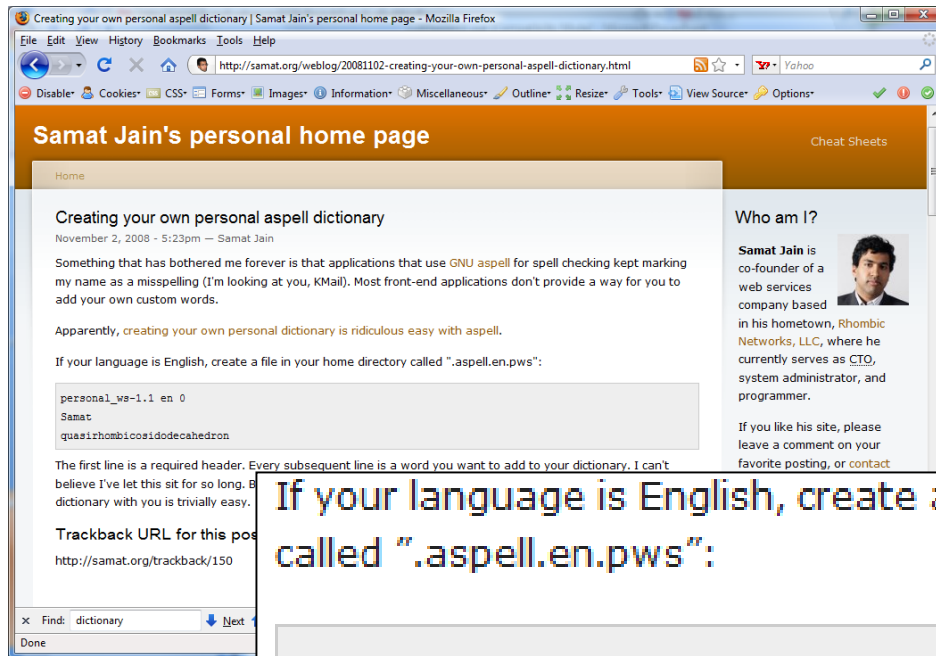
display the help message

-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.

Quiz questions for next class:

Lab 9
Five Posts

- 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

Parent and child

```

/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 $

```



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

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

*Resume stopped jobs with
bg and kill -18*

```
[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 background, 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) <i>Ctrl-Z or Ctrl-F</i>
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) <i>Ctrl-Z or Ctrl-F</i>
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 demonstrate 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 Ctrl-Z (to stop foreground 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
~
```

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) <i>Ctrl-Z or Ctrl-F</i>
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