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
- Supplies – chocolates

- Backup headset charged - done
- CCC Confer wall paper - done
- Slides & Lab uploaded - done
- Real test uploaded and permissions removed – done
- Mail script ready for test question

Cabrillo College est. 1959

CIS 90 - Lesson 10

Dennis

Sean

Christopher

Francisco

Salena

Abd

Sarah

Astitow

Mike D.

Alex

Christine

Steven

Richie

Nathan

Tony

James G.

Sergio

Anthony

Fernando

Miguel

Lars

Jennifer

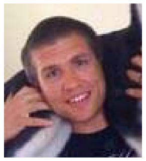
Rudy

Laura P.

Nick

Juan

Jacob


Andrew

Luke

Saulius

Rich

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

Online Class Students

Edtson

James B.

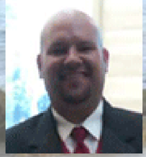
Liz

Casady

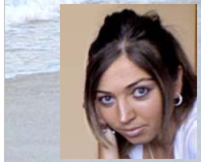
Jason

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

[ ] 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

UNIX Processes

Objectives	Agenda
• Know the process life cycle • Interpret ps command output • Run or schedule jobs to run in the background • Send signals to processes • Configure process load balancing	• Questions from last week • Housekeeping • Process definition • Process lifecycle • Process information • Job control • Signals • Load balancing • Test #2 • Wrap up

Previous material and assignment

1. Questions on previous material?
   - File management
   - Permissions
   - Input/output
   - Labs
   - Practice test

2. Questions regarding the test today?
   - Test will start during the last hour of class.
   - If you wish, you can keep working on it till midnight.
   - You must do all the work on the test by yourself and not ask or give help to others regarding any of the test questions.

FYI set –x, set +x

```
/home/cis90/roddyduk $ set -x
+ set -x
++ echo -ne '\033]0;roddyduk@opus:~'

/home/cis90/roddyduk $ type /bin/pi*
+ type /bin/ping /bin/ping6
/bin/ping is /bin/ping
/bin/ping6 is /bin/ping6
++ echo -ne '\033]0;roddyduk@opus:~'

/home/cis90/roddyduk $ type -af /usr/bin/p[ek]*[ct] 2> /dev/null
+ type -af /usr/bin/perlcc /usr/bin/perldoc /usr/bin/pkcs11_inspect
/usr/bin/perlcc is /usr/bin/perlcc
/usr/bin/perldoc is /usr/bin/perldoc
/usr/bin/pkcs11_inspect is /usr/bin/pkcs11_inspect
++ echo -ne '\033]0;roddyduk@opus:~'

/home/cis90/roddyduk $ set +x
+ set +x
/home/cis90/roddyduk $
```

Enable showing expanded arguments

set –x *shows you the actual expansion done by bash and what options and arguments are passed to the command/program being run*

Disable showing expanded arguments

FYI set –x, set +x

```
/home/cis90/roddyduk $ set -x
+ set -x
++ echo -ne '\033]0;roddyduk@opus:~'

/home/cis90/roddyduk $ find . -name '$LOGNAME'
+ find . -name '$LOGNAME'
find: ./Hidden: Permission denied
find: ./testdir: Permission denied
++ echo -ne '\033]0;roddyduk@opus:~'

/home/cis90/roddyduk $ find . -name "$LOGNAME"
+ find . -name roddyduk
find: ./Hidden: Permission denied
./roddyduk
find: ./testdir: Permission denied
++ echo -ne '\033]0;roddyduk@opus:~'

/home/cis90/roddyduk $ set +x
+ set +x
/home/cis90/roddyduk $
```

__set –x__ shows you the actual expansion done by bash and what options and arguments are passed to the command/program being run

set –x shows you the actual expansion done by bash and what is passed in to the process

FYI using {}

The braces { } are metacharacters

```
/home/cis90/roddyduk $ ls -ld test
drwxr-xr-x 2 roddyduk cis90 4096 Apr 22 09:46 test

/home/cis90/roddyduk $ ls test

/home/cis90/roddyduk $ touch test/file{1,2,3,4,5}

/home/cis90/roddyduk $ ls test
file1  file2  file3  file4  file5
```

Housekeeping

Housekeeping

- SJSU
- Chocolates
- Managing your grade

A message from …

SAN JOSE STATE
UNIVERSITY

Greetings Professor Simms,

My name is Tim Hill. I’m chair of the MIS department in the College of Business at SJSU. I’m reaching out to computer information systems faculty and advisers in the local community colleges to raise awareness of our program and the important benefits community college transfer students should consider when exploring major options at SJSU. For students interested in combining computer technology and business, MIS represents an exceptional choice of concentration within the Bachelors of Science in Business Administration.

Management Information Systems (MIS) is the BSBA concentration at SJSU that currently garners *the highest starting salaries and yields the highest placement rate upon graduation*. And MIS students are the *only ones on the SJSU campus formally recruited Google*! But unfortunately, far too few transfer students are aware of MIS and its unique advantages. Please consider making them aware of the 4 attachments and the 6 critical bullet points below before they choose their concentration within the BSBA at SJSU.

If you would be so kind as to share this information as appropriate, it would be greatly appreciated. And please let me know if I can be of assistance. I would be happy to visit your campus to speak with you and/or your colleagues and/or students about the MIS program and job opportunities. Just call (408) 924-3512 or reply to this email and I’ll get back to you promptly.

Recent developments to note:

The College of Business has just reduced the transfer GPA requirement from 3.4 down to 2.8, effective immediately.
SJSU is accepting applications for Fall 2011 through November 30.
(http://www.sjsu.edu/news/news_detail.jsp?id=3459)

SJSU has extended the Spring 2011 Admissions deadline to November 15 (http://www.sjsu.edu/news/news_detail.jsp?id=3468)

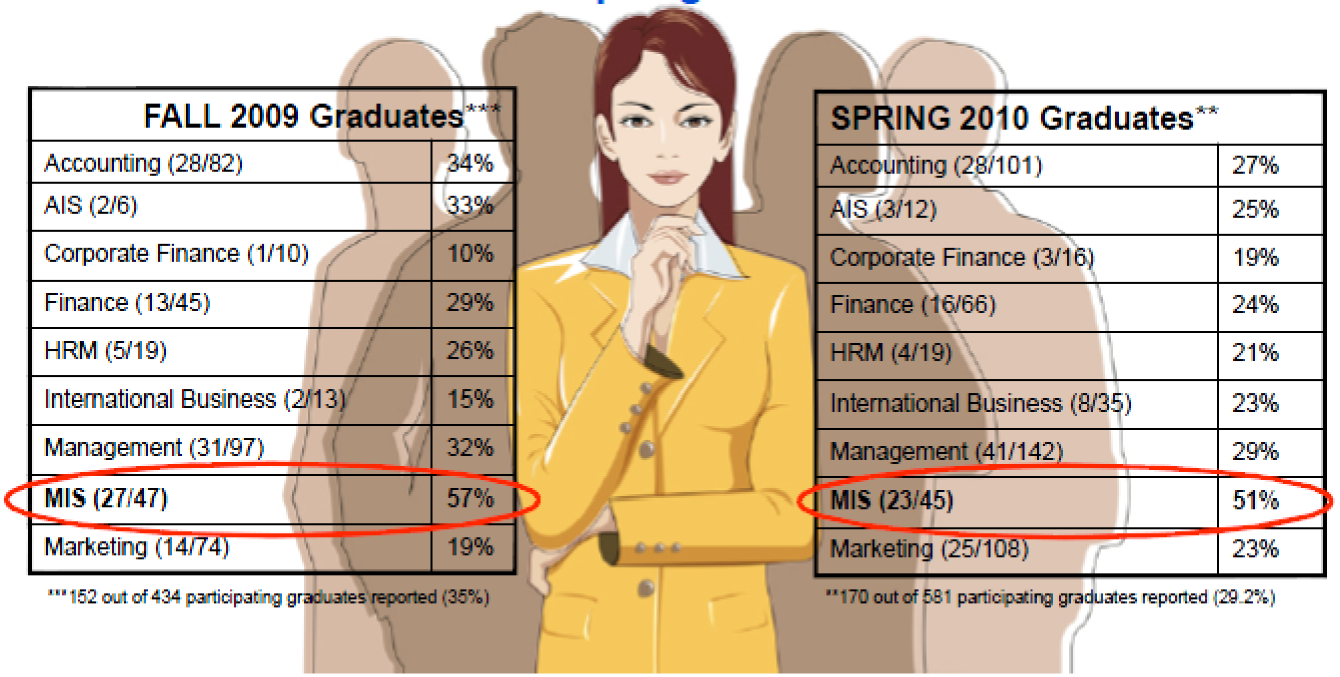
If you would be so kind as to share this information as appropriate, it would be greatly appreciated. And please let me know if I can be of assistance. I would be happy to visit your campus to speak with you and/or your colleagues and/or students about the MIS program and job opportunities. Just call (408) 924-3512 or reply to this email and I'll get back to you promptly.

Timothy R. Hill, Chair
Department of Management Information Systems
San Jose State University
One Washington Square
San Jose, CA 95192-0244
(408) 924-3512 (v)
http://www.cob.sjsu.edu/hill_t

College of Business Placement Rate at Graduation

(by concentration)

MIS Graduates lead in percentage of students with positions in career field upon graduation!



FALL 2009 Graduates***	
Accounting (28/82)	34%
AIS (2/6)	33%
Corporate Finance (1/10)	10%
Finance (13/45)	29%
HRM (5/19)	26%
International Business (2/13)	15%
Management (31/97)	32%
MIS (27/47)	**57%**
Marketing (14/74)	19%

***152 out of 434 participating graduates reported (35%)

SPRING 2010 Graduates**	
Accounting (28/101)	27%
AIS (3/12)	25%
Corporate Finance (3/16)	19%
Finance (16/66)	24%
HRM (4/19)	21%
International Business (8/35)	23%
Management (41/142)	29%
MIS (23/45)	**51%**
Marketing (25/108)	23%

**170 out of 581 participating graduates reported (29.2%)

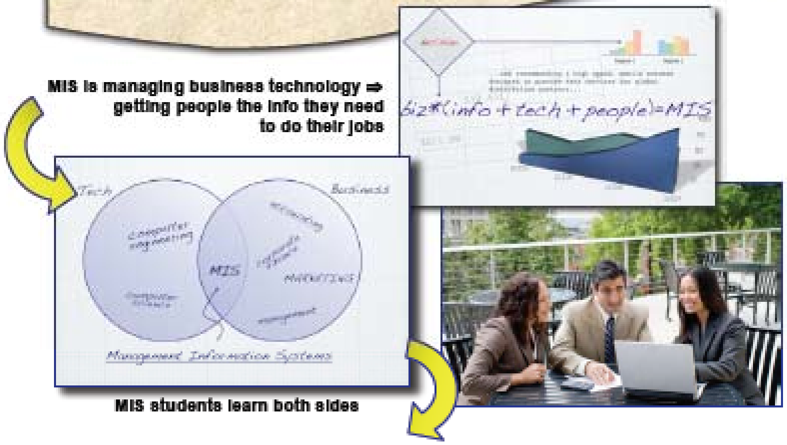
(Poll taken by SJSU Career Center)

* Data collected only from students who chose to participate in this survey at the College of Business Convocations.

SAN JOSE STATE UNIVERSITY

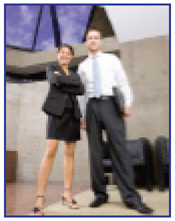
BS in Business Administration (avg new grad salary*)

Concentration:		
	☑ Management Information Systems	$58,555
	□ Accounting Information Systems	$53,797
	□ Finance	$52,733
	□ Management	$52,174
	□ International Business	$51,667
	□ Accounting	$51,438
	□ Corporate Financial Management	$48,750
	□ Marketing	$46,597
	□ Human Resources Management	$44,365


MIS is managing business technology ⇒ getting people the info they need to do their jobs

MIS students learn both sides

- ✓ interesting, dynamic, growing field – evolves with tech
- ✓ learn business + database, networking, web, security…
- ✓ award-winning faculty, academic & industry (IBM, Cisco, etc.)
- ✓ hands-on projects at local non-profits, eg. SJ City Parks Dept
- ✓ MIS grads work at HP, Cisco, Google, eBay…
- ✓ highest average salary of all SJSU Business concentrations*

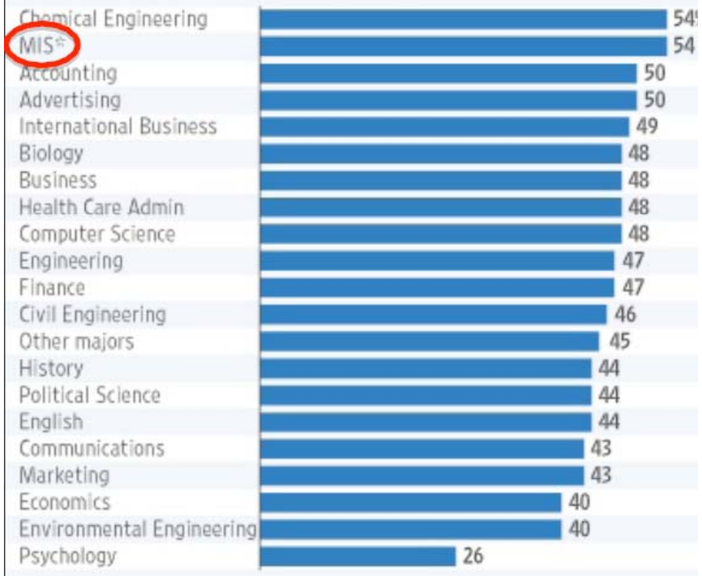


my ~~the~~ future is MIS
www.cob.sjsu.edu/mis

* according to 2009 SJSU Career Center survey

Satisfaction Not Guaranteed

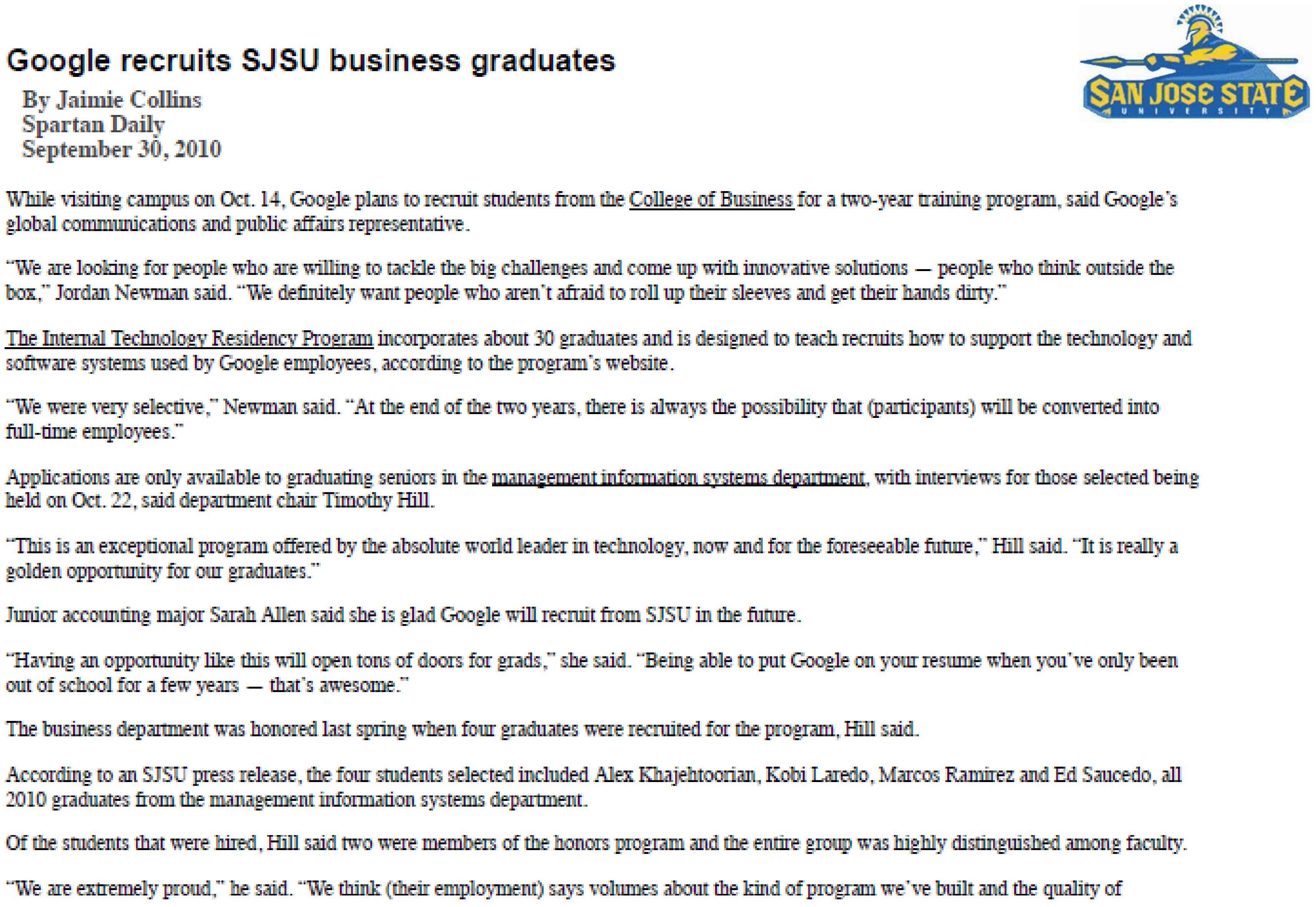
Percentage of college graduates, sorted by major, who answered 'satisfied' or 'very satisfied' to the question: 'Overall, how satisfied are you with your current career path up to now?'



Chemical Engineering	54
MIS*	54
Accounting	50
Advertising	50
International Business	49
Biology	48
Business	48
Health Care Admin	48
Computer Science	48
Engineering	47
Finance	47
Civil Engineering	46
Other majors	45
History	44
Political Science	44
English	44
Communications	43
Marketing	43
Economics	40
Environmental Engineering	40
Psychology	26

*Management Information Systems

Survey was conducted between April and June 2010, of people who graduated college between 1999 and 2010, with 10,800 respondents. Margin of error ranges between 2% and 7%, depending on major. Survey was limited to grads in a set of jobs deemed satisfying, well-paid and with growth potential. Source: PayScale.com



Google recruits SJSU business graduates

By Jaimie Collins
Spartan Daily
September 30, 2010

While visiting campus on Oct. 14, Google plans to recruit students from the College of Business for a two-year training program, said Google's global communications and public affairs representative.

"We are looking for people who are willing to tackle the big challenges and come up with innovative solutions — people who think outside the box," Jordan Newman said. "We definitely want people who aren't afraid to roll up their sleeves and get their hands dirty."

The Internal Technology Residency Program incorporates about 30 graduates and is designed to teach recruits how to support the technology and software systems used by Google employees, according to the program's website.

"We were very selective," Newman said. "At the end of the two years, there is always the possibility that (participants) will be converted into full-time employees."

Applications are only available to graduating seniors in the management information systems department, with interviews for those selected being held on Oct. 22, said department chair Timothy Hill.

"This is an exceptional program offered by the absolute world leader in technology, now and for the foreseeable future," Hill said. "It is really a golden opportunity for our graduates."

Junior accounting major Sarah Allen said she is glad Google will recruit from SJSU in the future.

"Having an opportunity like this will open tons of doors for grads," she said. "Being able to put Google on your resume when you've only been out of school for a few years — that's awesome."

The business department was honored last spring when four graduates were recruited for the program, Hill said.

According to an SJSU press release, the four students selected included Alex Khajehtoorian, Kobi Laredo, Marcos Ramirez and Ed Saucedo, all 2010 graduates from the management information systems department.

Of the students that were hired, Hill said two were members of the honors program and the entire group was highly distinguished among faculty.

"We are extremely proud," he said. "We think (their employment) says volumes about the kind of program we've built and the quality of

Chocolate Awards

Debian (2nd)	Redhat (3rd)	SUSE (1st)	Ubuntu (4th)
hamiljas botoschr dahlicas enriqste husemat messison orozcmig antiden perezrud redmanic fouric valadand zilissau	pirklla henrydal beltredt brownliz derriale galbrnat komicser millehom palmilar rochajuau velasliv dakkaabd	martiant birmijam cardefra daviesa salinjac dingechrr garciton hernaaar mottste parrijen pitzemik wattsluk	srecklau blacksea delfimik garibjam hrdinste menafer ojedavic dawadast pennitan castrsal plastadr woodjan

4 chocolates will go to 1st place finishers
3 chocolates will go to 2nd place finishers
2 chocolates will go to 3rd place finishers
1 chocolates will go to 4th place finishers

(Available in class, CIS Lab (Mondays 1-3:30) or TBD

Managing your grade

Points gone by

- 7 quizzes - 21 points
- 1 tests - 30 points
- 2 forum periods - 40 points
- 7 labs - 210 points

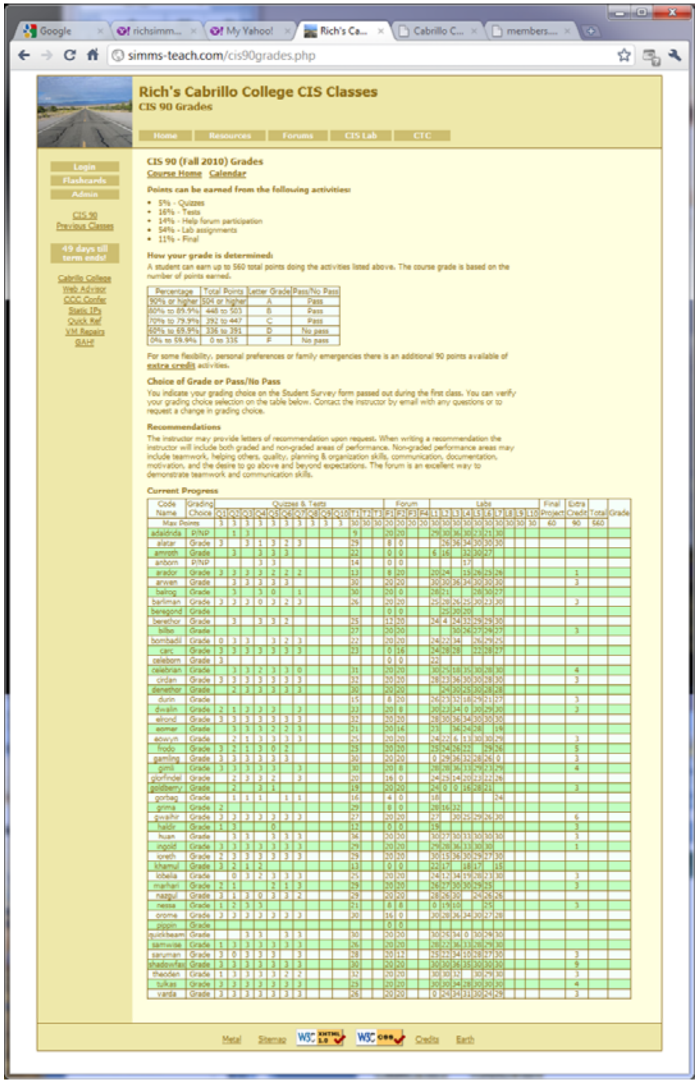
301 points

Points yet to earn

- 3 quizzes - 9 points
- 2 tests - 60 points
- 2 forum periods - 40 points
- 3 labs - 90 points
- 1 final project - 60 points

259 points

- Plus extra credit - up to 90 points



Rich's Cabrillo College CIS Classes

CIS 90 Grades

CIS 90 (Fall 2010) Grades

Course Home Calendar

Points can be earned from the following activities:

- 5% - Quizzes
- 16% - Tests
- 14% - Help forum participation
- 54% - Lab assignments
- 11% - Final

How your grade is determined:

A student can earn up to 560 total points doing the activities listed above. The course grade is based on the number of points earned.

Percentage	Total Points	Letter Grade	Pass/No Pass
90% or higher	504 or higher	A	Pass
80% to 89.9%	448 to 503	B	Pass
70% to 79.9%	392 to 447	C	Pass
60% to 69.9%	336 to 391	D	No pass
0% to 59.9%	0 to 335	F	No pass

For some flexibility, personal preferences or family emergencies there is an additional 90 points available of **extra credit** activities.

Choice of Grade or Pass/No Pass

You indicate your grading choice on the Student Survey form passed out during the first class. You can verify your grading choice selection on the table below. Contact the instructor by email with any questions or to request a change in grading choice.

Recommendations

The instructor may provide letters of recommendation upon request. When writing a recommendation the instructor will include both graded and non-graded areas of performance. Non-graded performance areas may include teamwork, helping others, quality, planning & organization skills, communication, documentation, motivation, and the desire to go above and beyond expectations. The forum is an excellent way to demonstrate teamwork and communication skills.

Current Progress

shadowfax	321	106.6%	
elrond	311	103.3%	
arwen	308	102.3%	
huan	304	101.0%	
tulkas	302	100.3%	
cirdan	301	100.0%	
samwise	291	96.7%	
gimli	286	95.0%	
ioreth	286	95.0%	
celebrian	285	94.7%	
orome	280	93.0%	
ingold	277	92.0%	
barliman	273	90.7%	
theoden	273	90.7%	
varda	262	87.0%	
gwaihir	261	86.7%	
quickbeam	260	86.4%	
dwalin	255	84.7%	
denethor	255	84.7%	
saruman	254	84.4%	
adaldrida	252	83.7%	
lobelia	252	83.7%	
marhari	248	82.4%	
nazgul	244	81.1%	

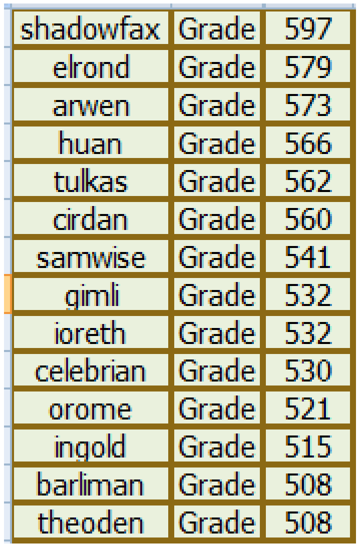
gamling	242	80.4%
berethor	240	79.7%
alatar	238	79.1%
eowyn	237	78.7%
bombadil	236	78.4%
frodo	233	77.4%
durin	222	73.8%
carc	217	72.1%
bilbo	209	69.4%
eomer	203	67.4%
glorfindel	203	67.4%
arador	196	65.1%
balrog	191	63.5%
goldberry	157	52.2%
amroth	145	48.2%
grima	115	38.2%
khamul	110	36.5%
nessa	103	34.2%
beregond	75	24.9%
gorbag	67	22.3%
haldir	38	12.6%
anborn	37	12.3%
celeborn	25	8.3%
pippin	0	0.0%

Current total points and completion rates on 10/31/2010

Estimate using current points and scoring percentages

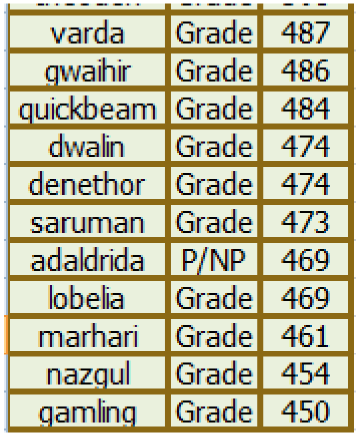
*estimatedPoints = currentPoints + (currentPoints/301 * 259)*

ALPHA ***(Hypothetical)***



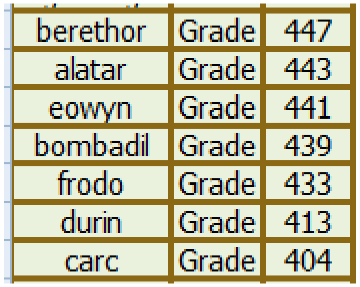
shadowfax	Grade	597
elrond	Grade	579
arwen	Grade	573
huan	Grade	566
tulkas	Grade	562
cirdan	Grade	560
samwise	Grade	541
gimli	Grade	532
ioreth	Grade	532
celebrian	Grade	530
orome	Grade	521
ingold	Grade	515
barliman	Grade	508
theoden	Grade	508

BRAVO ***(Hypothetical)***



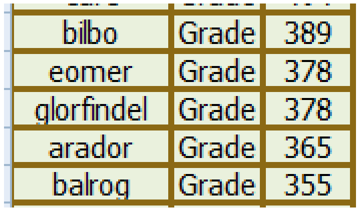
varda	Grade	487
gwaihir	Grade	486
quickbeam	Grade	484
dwalin	Grade	474
denethor	Grade	474
saruman	Grade	473
adaldrida	P/NP	469
lobelia	Grade	469
marhari	Grade	461
nazgul	Grade	454
gamling	Grade	450

CHARLIE ***(Hypothetical)***



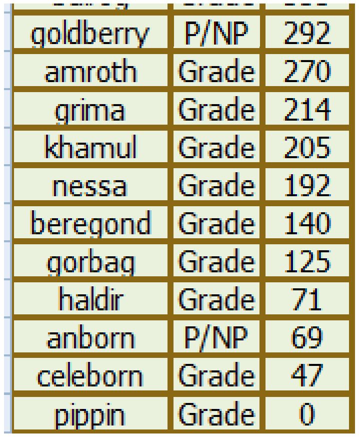
berethor	Grade	447
alatar	Grade	443
eowyn	Grade	441
bombadil	Grade	439
frodo	Grade	433
durin	Grade	413
carc	Grade	404

DELTA ***(Hypothetical)***

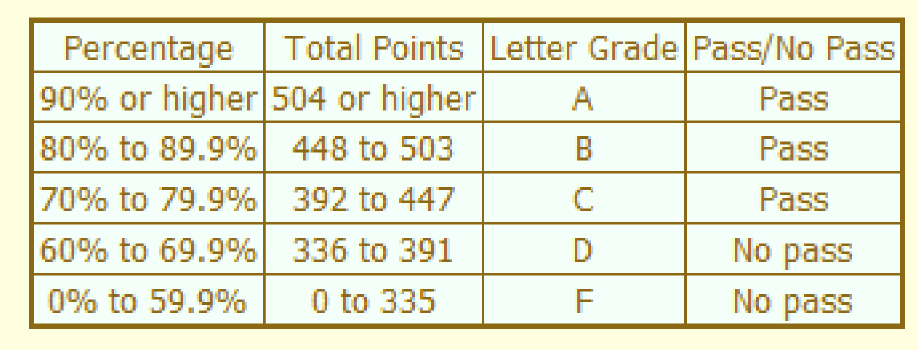


bilbo	Grade	389
eomer	Grade	378
glorfindel	Grade	378
arador	Grade	365
balrog	Grade	355

FOXTROT ***(Hypothetical)***



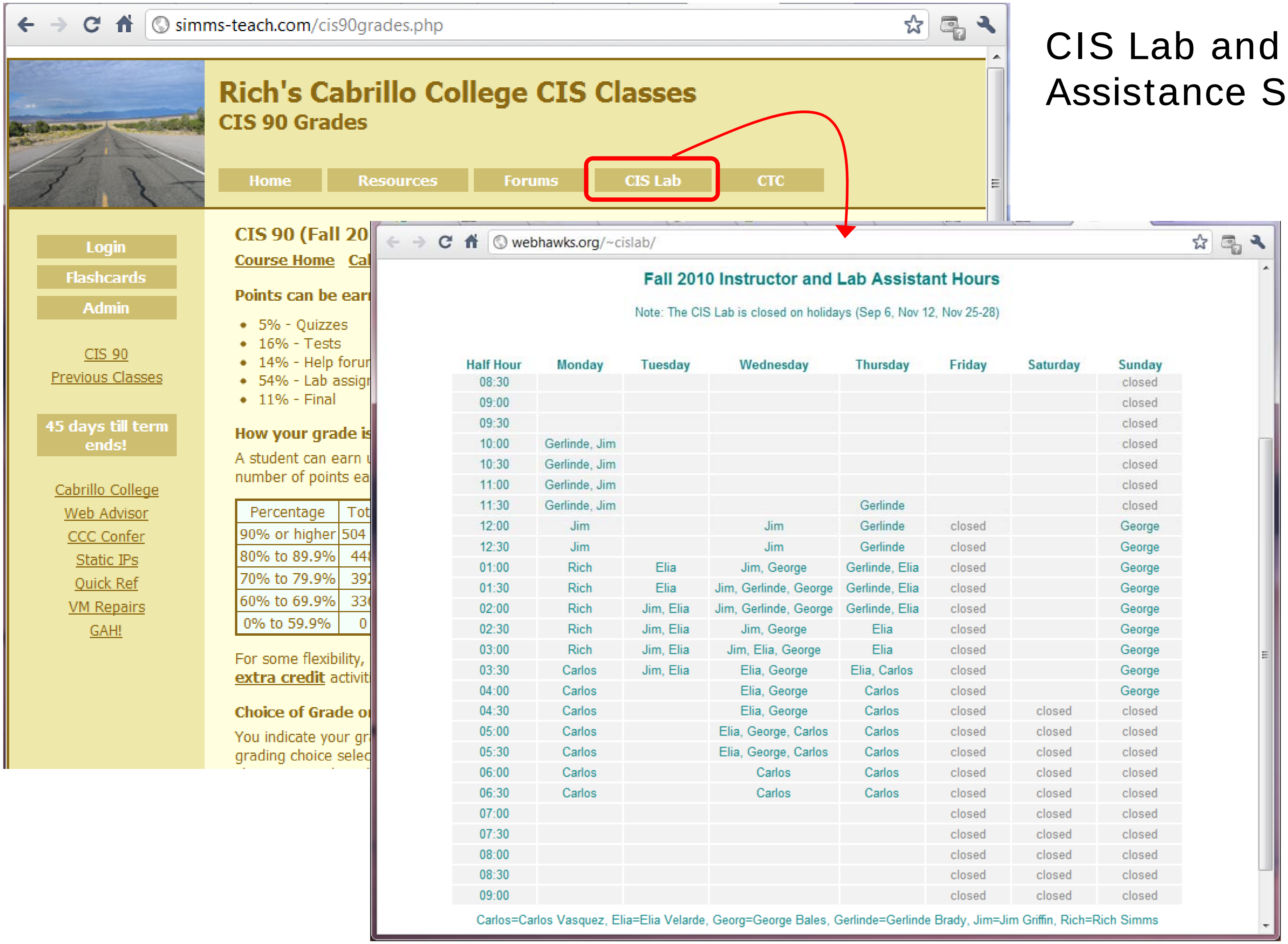
goldberry	P/NP	292
amroth	Grade	270
grima	Grade	214
khamul	Grade	205
nessa	Grade	192
beregond	Grade	140
gorbag	Grade	125
haldir	Grade	71
anborn	P/NP	69
celeborn	Grade	47
pippin	Grade	0



Percentage	Total Points	Letter Grade	Pass/No Pass
90% or higher	504 or higher	A	Pass
80% to 89.9%	448 to 503	B	Pass
70% to 79.9%	392 to 447	C	Pass
60% to 69.9%	336 to 391	D	No pass
0% to 59.9%	0 to 335	F	No pass

Don't like what you see?
There are many options: extra credit, tutoring, grading choice, office hours, using TAs in CIS Lab and/or contacting your instructor for some 1:1 time.

CIS Lab and Assistance Schedule



Rich's Cabrillo College CIS Classes

CIS 90 Grades

CIS 90 (Fall 20

Course Home Cal

Points can be ear

- 5% - Quizzes
- 16% - Tests
- 14% - Help forur
- 54% - Lab assigr
- 11% - Final

How your grade is

A student can earn u number of points ea

Percentage	Tot
90% or higher	504
80% to 89.9%	448
70% to 79.9%	392
60% to 69.9%	330
0% to 59.9%	0

For some flexibility, extra credit activiti

Choice of Grade or

You indicate your gr grading choice selec

Fall 2010 Instructor and Lab Assistant Hours

Note: The CIS Lab is closed on holidays (Sep 6, Nov 12, Nov 25-28)

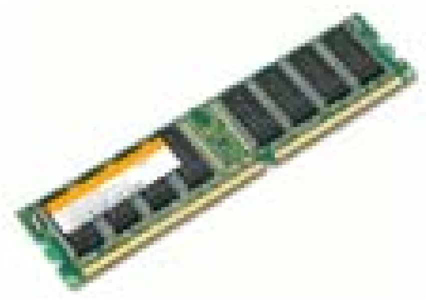
Half Hour	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
08:30							closed
09:00							closed
09:30							closed
10:00	Gerlinde, Jim						closed
10:30	Gerlinde, Jim						closed
11:00	Gerlinde, Jim						closed
11:30	Gerlinde, Jim			Gerlinde			closed
12:00	Jim		Jim	Gerlinde	closed		George
12:30	Jim		Jim	Gerlinde	closed		George
01:00	Rich	Elia	Jim, George	Gerlinde, Elia	closed		George
01:30	Rich	Elia	Jim, Gerlinde, George	Gerlinde, Elia	closed		George
02:00	Rich	Jim, Elia	Jim, Gerlinde, George	Gerlinde, Elia	closed		George
02:30	Rich	Jim, Elia	Jim, George	Elia	closed		George
03:00	Rich	Jim, Elia	Jim, Elia, George	Elia	closed		George
03:30	Carlos	Jim, Elia	Elia, George	Elia, Carlos	closed		George
04:00	Carlos		Elia, George	Carlos	closed		George
04:30	Carlos		Elia, George	Carlos	closed	closed	closed
05:00	Carlos		Elia, George, Carlos	Carlos	closed	closed	closed
05:30	Carlos		Elia, George, Carlos	Carlos	closed	closed	closed
06:00	Carlos		Carlos	Carlos	closed	closed	closed
06:30	Carlos		Carlos	Carlos	closed	closed	closed
07:00					closed	closed	closed
07:30					closed	closed	closed
08:00					closed	closed	closed
08:30					closed	closed	closed
09:00					closed	closed	closed

Carlos=Carlos Vasquez, Elia=Elia Velarde, Georg=George Bales, Gerlinde=Gerlinde Brady, Jim=Jim Griffin, Rich=Rich Simms

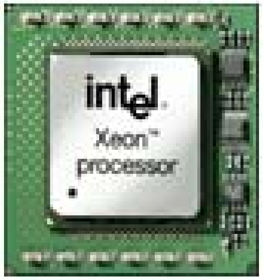
Process Definition

Definition of a process

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



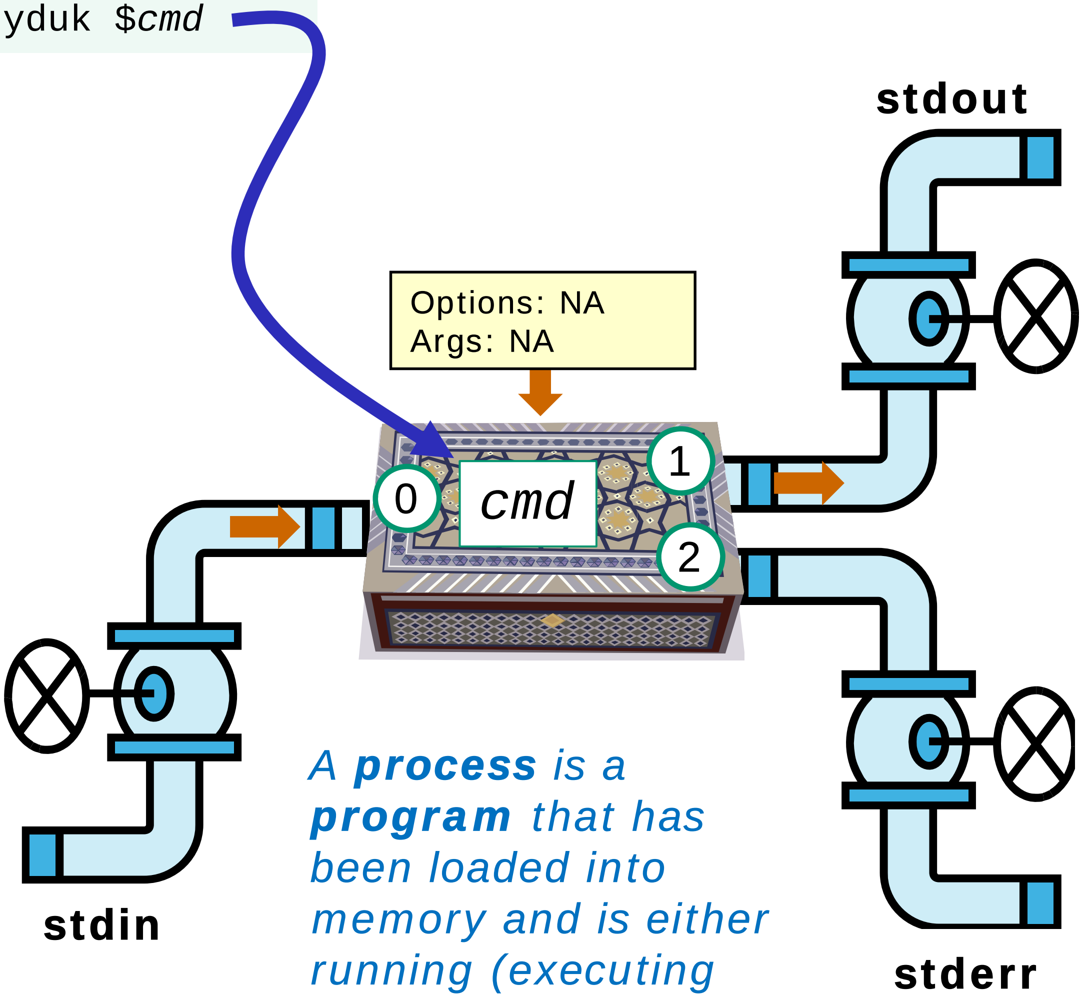
RAM (Random Access Memory) contains the instructions



The CPU executes the instructions in RAM

Program to process

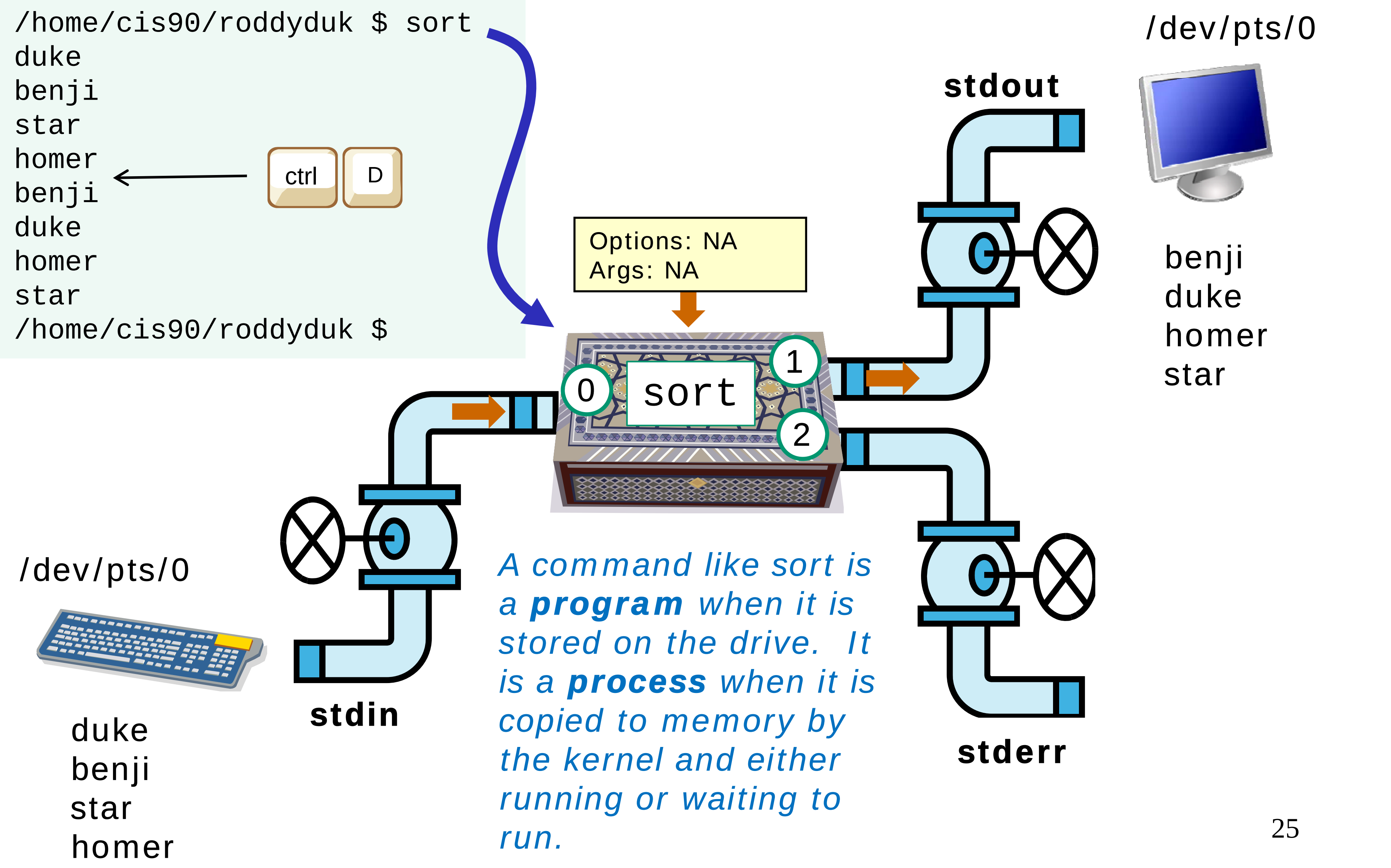
```
/home/cis90/roddyduk $cmd
```



Options: NA
Args: NA

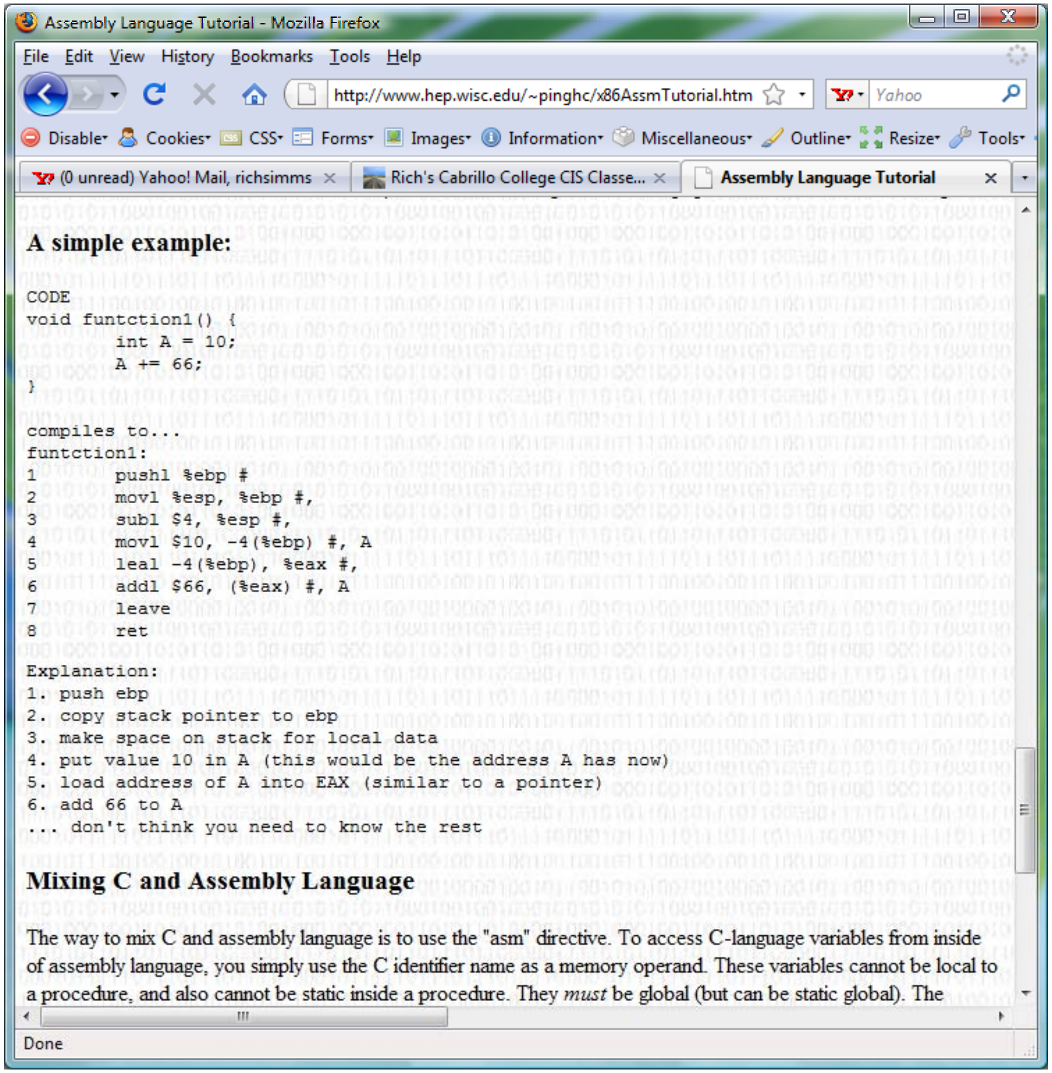
*A **process** is a **program** that has been loaded into memory and is either running (executing instructions) or waiting to run*

Example program to process: sort command



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

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


Assembly Language Tutorial - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.hep.wisc.edu/~pinghc/x86AssmTutorial.htm
Yahoo
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools
(0 unread) Yahoo! Mail, richsimms
Rich's Cabrillo College CIS Classe...
Assembly Language Tutorial
Done

A simple example:

```
CODE
void funtction1() {
        int A = 10;
        A += 66;
}

compiles to...
funtction1:
1       pushl %ebp #
2       movl %esp, %ebp #,
3       subl $4, %esp #,
4       movl $10, -4(%ebp) #, A
5       leal -4(%ebp), %eax #,
6       addl $66, (%eax) #, A
7       leave
8       ret

Explanation:
1. push ebp
2. copy stack pointer to ebp
3. make space on stack for local data
4. put value 10 in A (this would be the address A has now)
5. load address of A into EAX (similar to a pointer)
6. add 66 to A
... don't think you need to know the rest
```

Mixing C and Assembly Language

The way to mix C and assembly language is to use the "asm" directive. To access C-language variables from inside of assembly language, you simply use the C identifier name as a memory operand. These variables cannot be local to a procedure, and also cannot be static inside a procedure. They *must* be global (but can be static global). The

Most programs are written in the C language

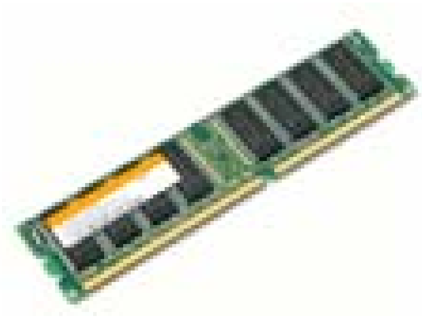
The C compiler translates the C code into binary machine code instructions the CPU can execute.

http://www.hep.wisc.edu/~pinghc/x86AssmTutorial.htm

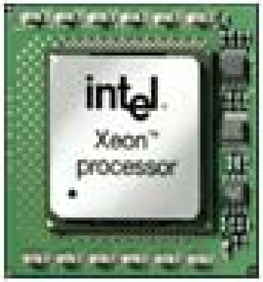
Example program to process: sort command

```
[rsimms@opus ~]$ type sort
sort is /bin/sort

[rsimms@opus ~]$ file /bin/sort
/bin/sort: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux
2.6.9, dynamically linked (uses shared libs), for GNU/Linux 2.6.9, stripped
[rsimms@opus ~]$

[rsimms@opus ~]$ xxd /bin/sort | more
```

```
0000000: 7f45 4c46 0101 0100 0000 0000 0000 0000  .ELF............
0000010: 0200 0300 0100 0000 e093 0408 3400 0000  ............4...
0000020: 2cdb 0000 0000 0000 3400 2000 0800 2800  ,.......4. ...(.
0000030: 1f00 1e00 0600 0000 3400 0000 3480 0408  ........4...4...
0000040: 3480 0408 0001 0000 0001 0000 0500 0000  4...............
0000050: 0400 0000 0300 0000 3401 0000 3481 0408  ........4...4...
0000060: 3481 0408 1300 0000 1300 0000 0400 0000  4...............
0000070: 0100 0000 0100 0000 0000 0000 0080 0408  ................
0000080: 0080 0408 caca 0000 caca 0000 0500 0000  ................
0000090: 0010 0000 0100 0000 00d0 0000 0050 0508  .............P..
00000a0: 0050 0508 9404 0000 e80a 0000 0600 0000  .P..............
00000b0: 0010 0000 0200 0000 a0d1 0000 a051 0508  .............Q..
```

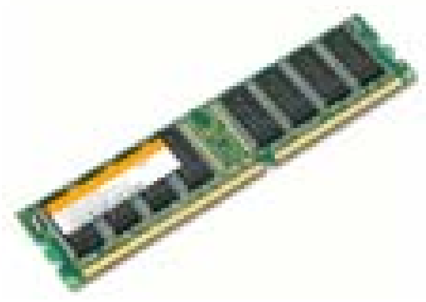


*A command like sort is a **program** when it is stored on the drive. It is a **process** when it is copied to memory by the kernel and either running or waiting to run by the CPU*

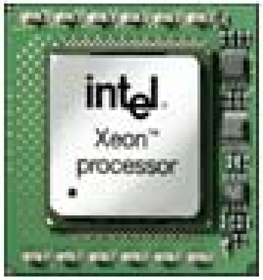


Definition of a process

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



RAM (Random Access Memory) contains the instructions

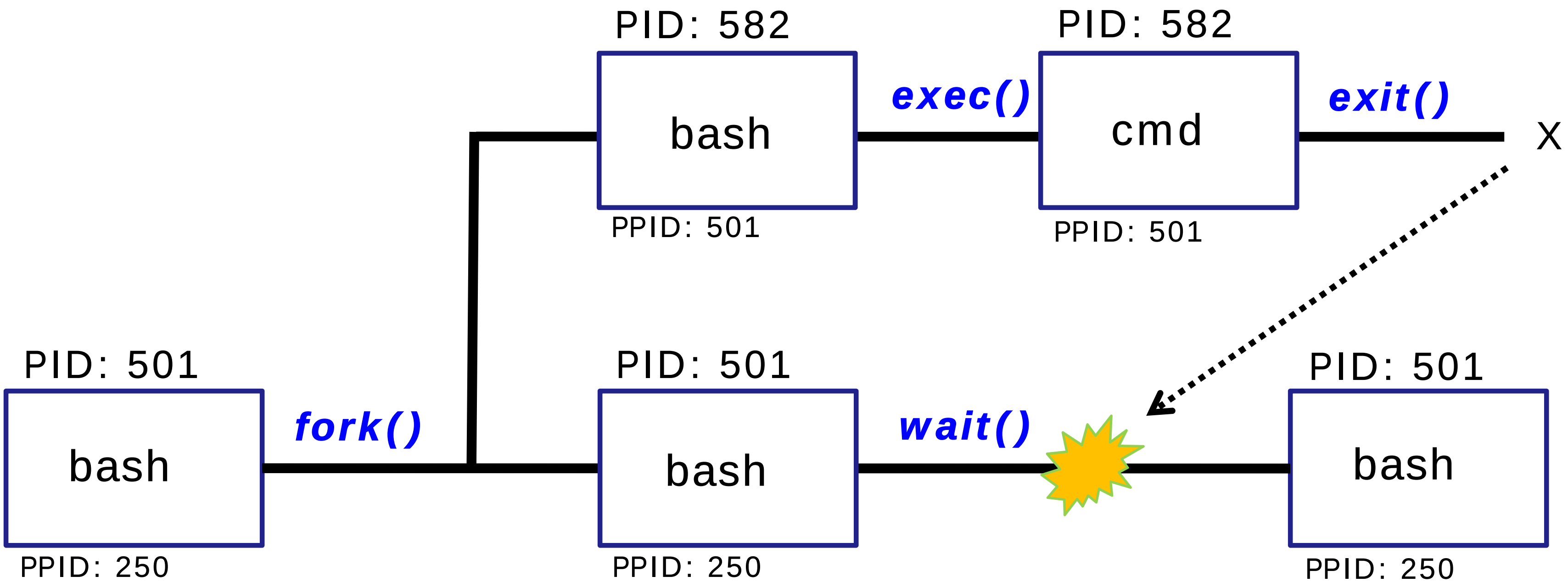


The CPU executes the instructions in RAM

Process Life Cycle

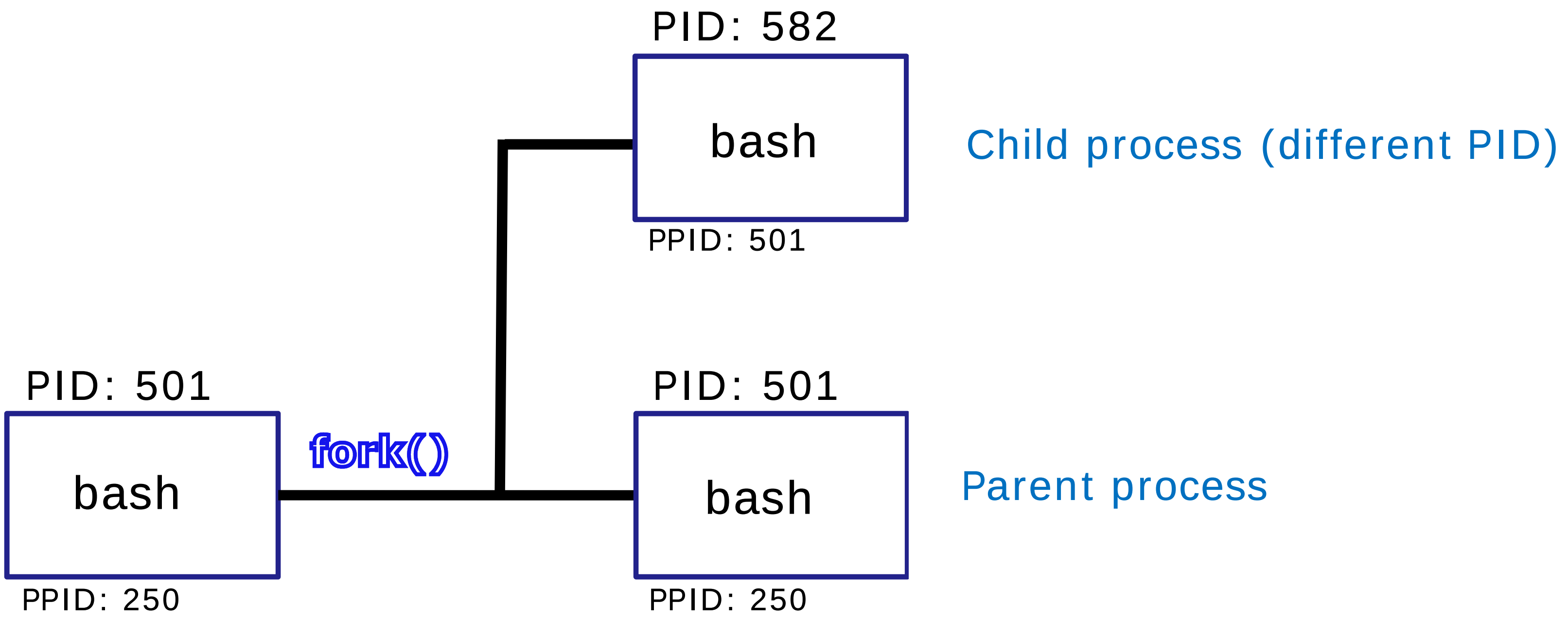
Process Lifecycle

Note: This diagram shows a generic command "cmd" being loaded and run by a user using the bash shell.



Note: A process uses system calls (e.g. fork, exec, wait, exit) to requests services from the kernel

Process Lifecycle

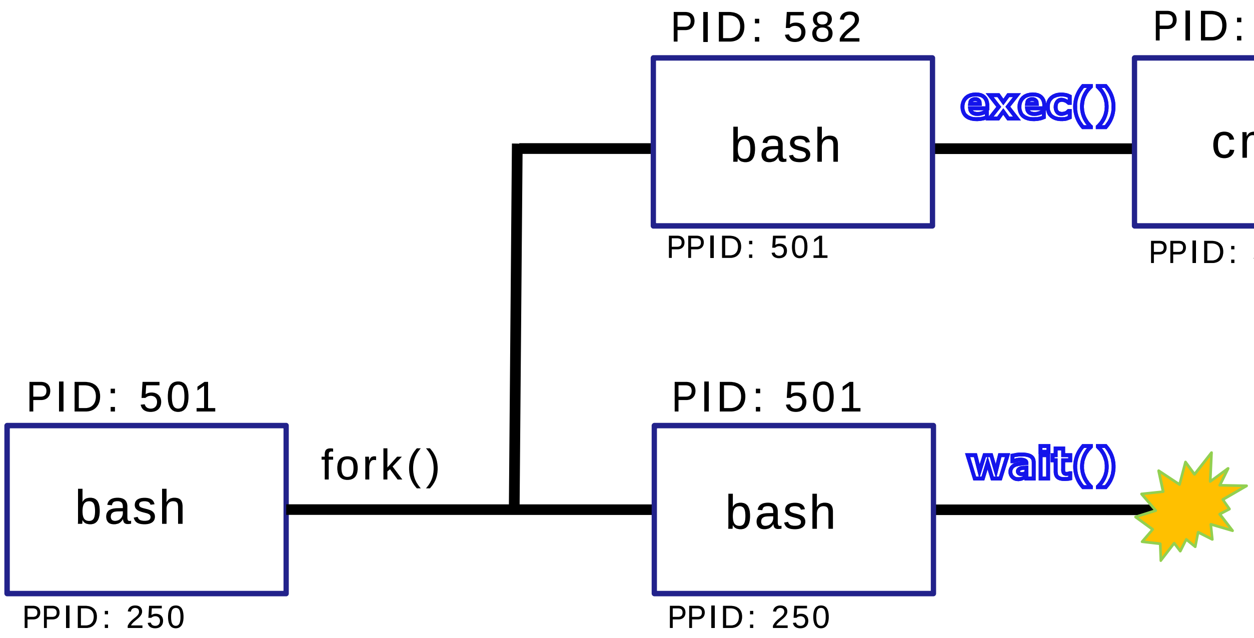


1) When a program is loaded into memory a new process must be created.

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.

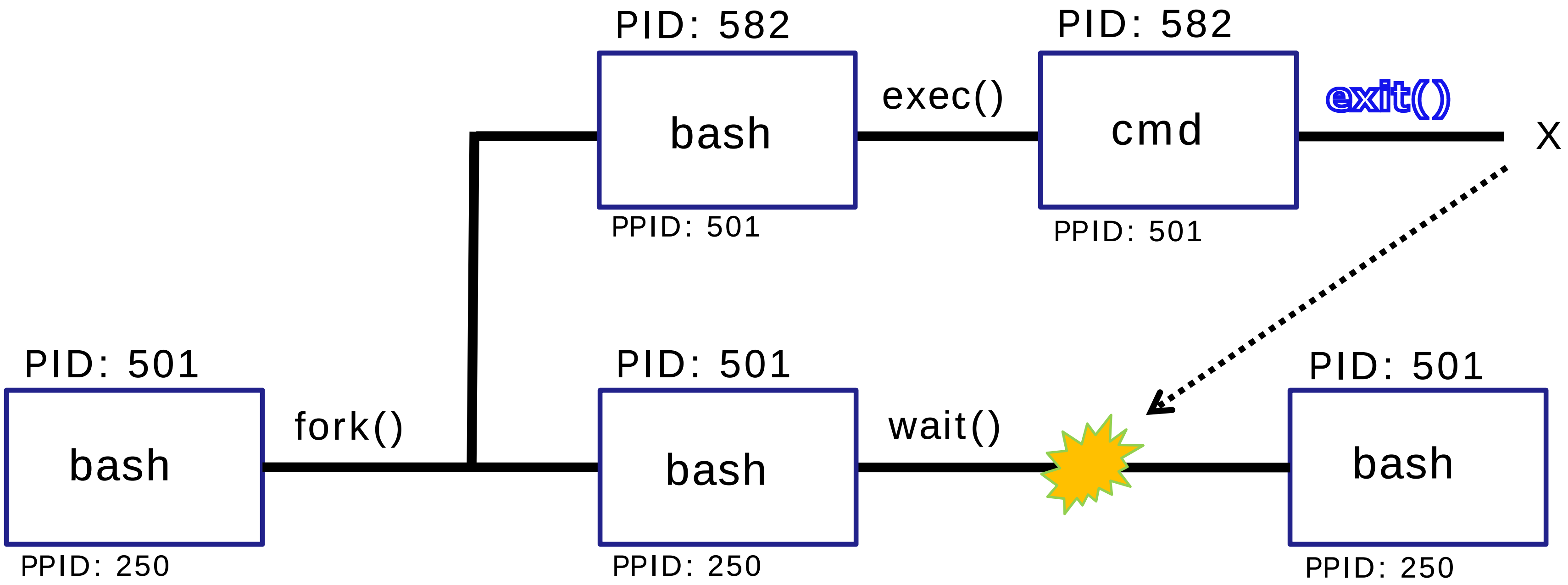
Process Lifecycle



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.

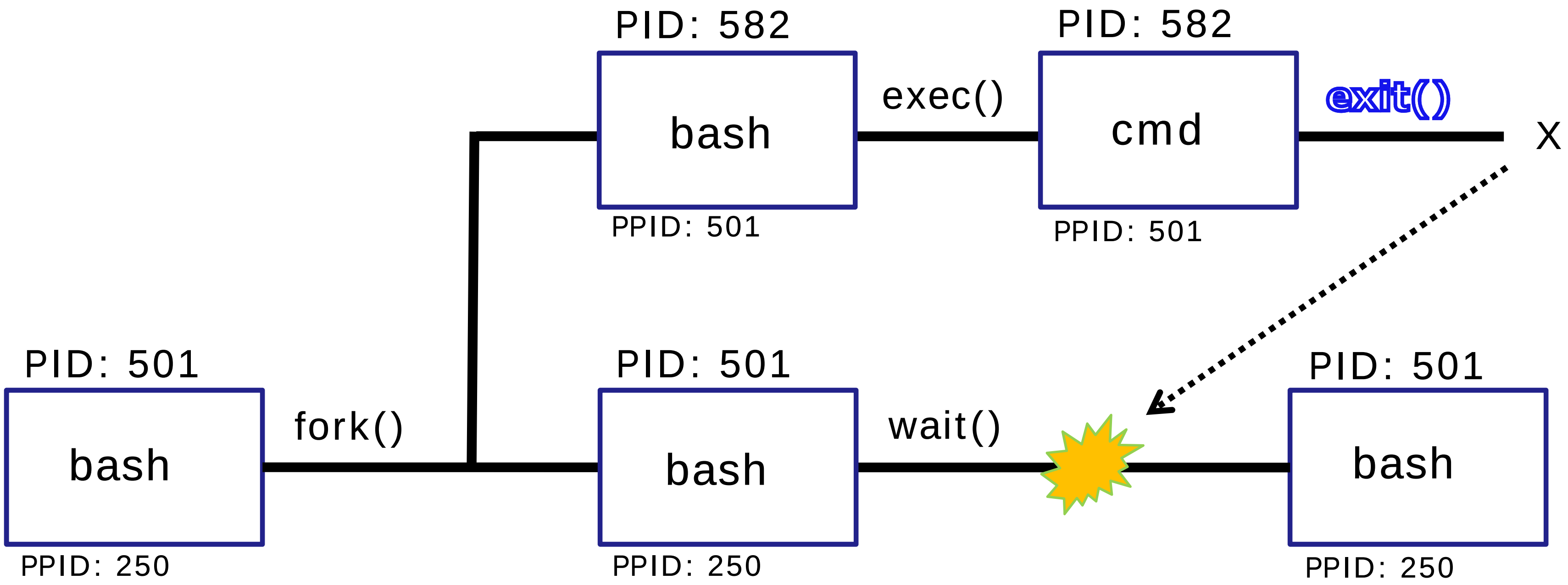
Process Lifecycle



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

Process Information

Information	Description
PID	Process Identification Number, a unique number identifying the process
PPID	Parent PID, the PID of the parent process (like … in the file hierarchy)
UID	The user running the process
TTY	The terminal that the process's stdin and stdout are connected to
S	The status of the process: S=Sleeping, R=Running, T=Stopped, Z=Zombie
PRI	Process priority
SZ	Process size
CMD	The name of the process (the command being run)
C	The CPU utilization of the process
WCHAN	Waiting channel (name of kernel function in which the process is sleeping)
F	Flags (1=forked but didn't exit, 4=used superuser privileges)
TIME	Cumulative CPU time
NI	Nice value

Just a few of the types of information kept on a process.

*Use **man ps** to see a lot more.*

Process Information

```
[rsimms@opus ~]$ ps
  PID TTY          TIME CMD
 6204 pts/6    00:00:00 bash
 6285 pts/6    00:00:00 ps
[rsimms@opus ~]$
```

Show just my processes. Note bash was started for me when I started my terminal session. ps is showing because it is running as this output is printed.

Process Information

Process ID number

```
[rsimms@opus ~]$ ps -a
  PID TTY          TIME CMD
 6173 pts/0    00:00:00 man
 6176 pts/0    00:00:00 sh
 6177 pts/0    00:00:00 sh
 6182 pts/0    00:00:00 less
 6294 pts/6    00:00:00 ps
[rsimms@opus ~]$
```

-a option shows all my processes not associated with a terminal. This includes my other login session where I'm doing a man command on ps.

```
[rsimms@opus ~]$ ps x
  PID TTY      STAT   TIME COMMAND
 5368 ?        S      0:00 sshd: rsimms@pts/0
 5369 pts/0    Ss     0:00 -bash
 6173 pts/0    S+     0:00 man ps
 6176 pts/0    S+     0:00 sh -c (cd /usr/share/man && (echo ".ll 7.5i"; echo ".nr L
 6177 pts/0    S+     0:00 sh -c (cd /usr/share/man && (echo ".ll 7.5i"; echo ".nr L
 6182 pts/0    S+     0:00 /usr/bin/less -is
 6203 ?        S      0:00 sshd: rsimms@pts/6
 6204 pts/6    Ss     0:00 -bash
 6312 pts/6    R+     0:00 ps x
```

The x option shows full commands being run and states (most are asleep).

Sleeping

Running (+ means running in the foreground)

I'm using two Putty sessions, in one session I have the man page open for ps, the other I'm issuing ps commands

Process Information

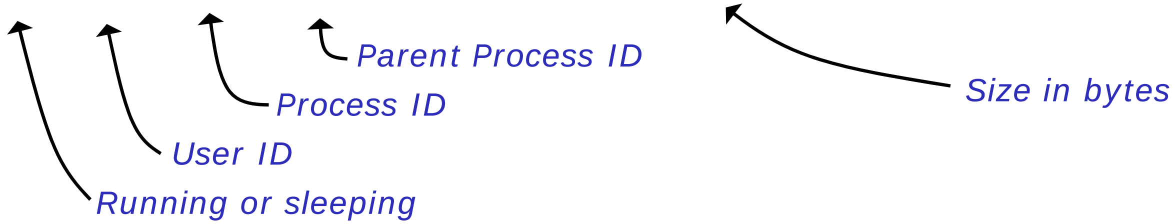
```
[rsimms@opus ~]$ cat /etc/passwd | grep Marcos
valdemar:x:1200:103:Marcos Valdebenito:/home/cis90/valdemar:/bin/bash

[rsimms@opus ~]$ ps -u 1200
  PID TTY          TIME CMD
 5971 ?        00:00:00 sshd
 5972 pts/5    00:00:00 bash
[rsimms@opus ~]$

[rsimms@opus ~]$ ps -u dymesdia
  PID TTY          TIME CMD
 6418 ?        00:00:00 sshd
 6419 pts/1    00:00:00 bash
[rsimms@opus ~]$

[rsimms@opus ~]$ ps -u rsimms
  PID TTY          TIME CMD
 5368 ?        00:00:00 sshd
 5369 pts/0    00:00:00 bash
 6173 pts/0    00:00:00 man
 6176 pts/0    00:00:00 sh
 6177 pts/0    00:00:00 sh
 6182 pts/0    00:00:00 less
 6203 ?        00:00:00 sshd
 6204 pts/6    00:00:00 bash
 6510 pts/6    00:00:00 ps
[rsimms@opus ~]$
```

Use the u option to look at processes owned by a specific user

Process Information

Use –l for additional options

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



Parent Process ID

Size in bytes

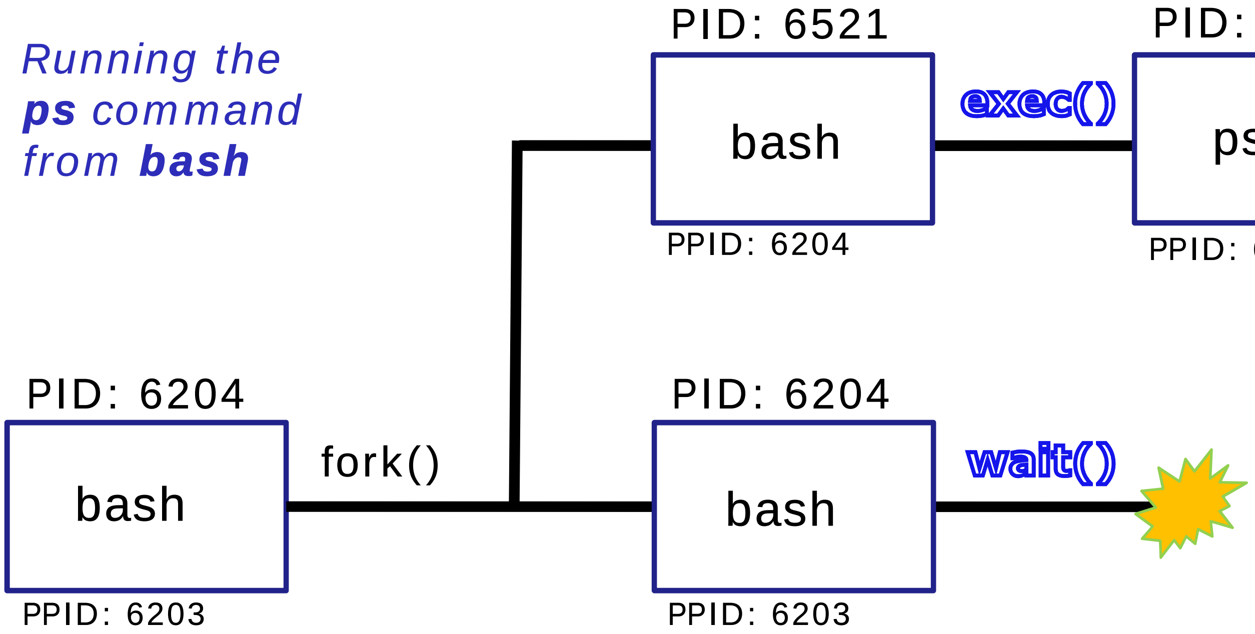
Process ID

User ID

Running or sleeping

Process Lifecycle

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



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

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

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

Process Information

```
[rsimms@opus ~]$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Sep10 ?        00:00:05 init [3]
root         2     1  0 Sep10 ?        00:00:00 [migration/0]
root         3     1  0 Sep10 ?        00:00:00 [ksoftirqd/0]
root         4     1  0 Sep10 ?        00:00:00 [watchdog/0]
root         5     1  0 Sep10 ?        00:00:02 [migration/1]
root         6     1  0 Sep10 ?        00:00:00 [ksoftirqd/1]
root         7     1  0 Sep10 ?        00:00:00 [watchdog/1]
root         8     1  0 Sep10 ?        00:00:00 [events/0]
root         9     1  0 Sep10 ?        00:00:00 [events/1]
root        10     1  0 Sep10 ?        00:00:00 [khelper]
root        11     1  0 Sep10 ?        00:00:00 [kthread]
root        15    11  0 Sep10 ?        00:00:00 [kblockd/0]
root        16    11  0 Sep10 ?        00:00:00 [kblockd/1]
root        17    11  0 Sep10 ?        00:00:00 [kacpid]
root       109    11  0 Sep10 ?        00:00:00 [cqueue/0]
root       110    11  0 Sep10 ?        00:00:00 [cqueue/1]
root       113    11  0 Sep10 ?        00:00:00 [khubd]
root       115    11  0 Sep10 ?        00:00:00 [kseriod]
root       181    11  0 Sep10 ?        00:00:00 [pdflush]
root       182    11  0 Sep10 ?        00:00:07 [pdflush]
root       183    11  0 Sep10 ?        00:00:01 [kswapd0]
root       184    11  0 Sep10 ?        00:00:00 [aio/0]
root       185    11  0 Sep10 ?        00:00:00 [aio/1]
root       341    11  0 Sep10 ?        00:00:00 [kpsmoused]
root       371    11  0 Sep10 ?        00:00:00 [ata/0]
```

Use ***–ef*** *option to see every process running*

Process Information

```
root       372    11  0 Sep10 ?        00:00:00 [ata/1]
root       373    11  0 Sep10 ?        00:00:00 [ata_aux]
root       377    11  0 Sep10 ?        00:00:00 [scsi_eh_0]
root       378    11  0 Sep10 ?        00:00:00 [scsi_eh_1]
root       379    11  0 Sep10 ?        00:01:25 [kjournald]
root       412    11  0 Sep10 ?        00:00:00 [kauditd]
root       446     1  0 Sep10 ?        00:00:00 /sbin/udevd -d
root       869    11  0 Sep10 ?        00:00:01 [kedac]
root      1420    11  0 Sep10 ?        00:00:00 [kmpathd/0]
root      1421    11  0 Sep10 ?        00:00:00 [kmpathd/1]
root      2082     1  0 Sep10 ?        00:00:05 /usr/sbin/restorecond
root      2098     1  0 Sep10 ?        00:00:11 auditd
root      2100  2098  0 Sep10 ?        00:00:05 /sbin/audispd
root      2120     1  0 Sep10 ?        00:00:23 syslogd -m 0
root      2123     1  0 Sep10 ?        00:00:00 klogd -x
root      2160     1  0 Sep10 ?        00:00:20 mcstransd
rpc       2183     1  0 Sep10 ?        00:00:00 portmap
root      2201     1  0 Sep10 ?        00:01:18 /usr/bin/python -E /usr/sbin/setroub
rpcuser   2227     1  0 Sep10 ?        00:00:00 rpc.statd
root      2275     1  0 Sep10 ?        00:00:00 rpc.idmapd
root      2345     1  0 Sep10 ?        00:00:00 /usr/bin/vmnet-bridge -d /var/run/vm
root      2364     1  0 Sep10 ?        00:00:00 /usr/bin/vmnet-natd -d /var/run/vmne
dbus      2383     1  0 Sep10 ?        00:00:15 dbus-daemon --system
root      2434     1  0 Sep10 ?        00:00:51 pcscd
root      2472     1  0 Sep10 ?        00:00:00 /usr/bin/hidd --server
root      2493     1  0 Sep10 ?        00:00:02 automount
```

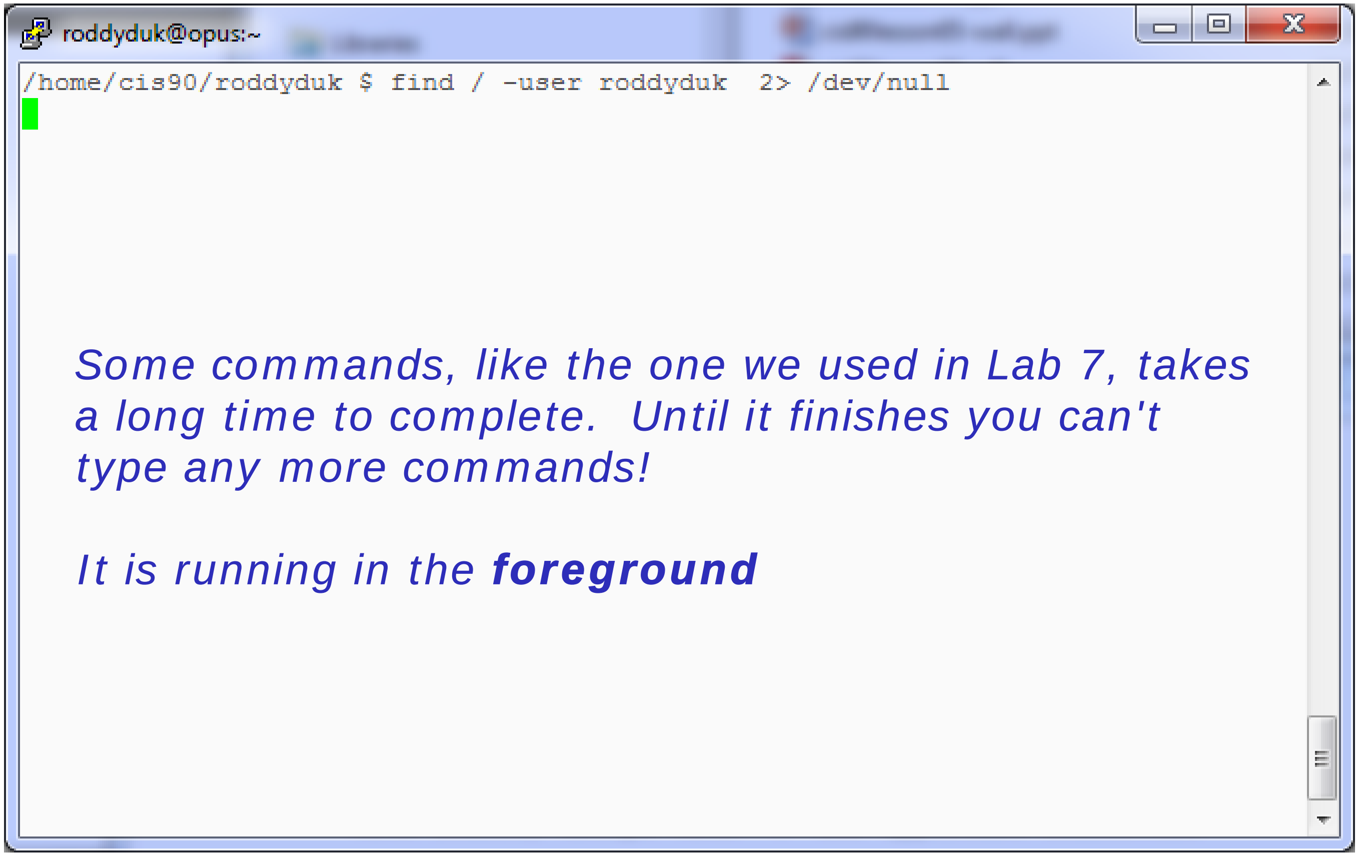
Process Information

```
root       2534     1  0 Sep10 ?        00:00:00 ./hpiod
root       2539     1  0 Sep10 ?        00:00:00 python ./hpssd.py
root       2556     1  0 Sep10 ?        00:00:00 cupsd
root       2575     1  0 Sep10 ?        00:00:11 /usr/sbin/sshd
root       2600     1  0 Sep10 ?        00:00:01 sendmail: accepting connections
smmsp      2609     1  0 Sep10 ?        00:00:00 sendmail: Queue runner@01:00:00 for
root       2626     1  0 Sep10 ?        00:00:00 crond
xfs        2662     1  0 Sep10 ?        00:00:00 xfs -droppriv -daemon
root       2693     1  0 Sep10 ?        00:00:00 /usr/sbin/atd
root       2710     1  0 Sep10 ?        00:00:00 rhnsd --interval 240
root       2743     1  0 Sep10 ?        00:01:33 /usr/bin/python -tt /usr/sbin/yum-up
root       2745     1  0 Sep10 ?        00:00:00 /usr/libexec/gam_server
root       2749     1  0 Sep10 ?        00:00:00 /usr/bin/vmnet-netifup -d /var/run/v
root       2758     1  0 Sep10 ?        00:00:00 /usr/bin/vmnet-netifup -d /var/run/v
root       2768     1  0 Sep10 ?        00:00:00 /usr/bin/vmnet-netifup -d /var/run/v
root       2827     1  0 Sep10 ?        00:00:00 /usr/bin/vmnet-dhcpd -cf /etc/vmware
root       2858     1  0 Sep10 ?        00:00:00 /usr/bin/vmnet-dhcpd -cf /etc/vmware
root       2859     1  0 Sep10 ?        00:00:00 /usr/bin/vmnet-dhcpd -cf /etc/vmware
68         2875     1  0 Sep10 ?        00:00:01 hald
root       2876  2875  0 Sep10 ?        00:00:00 hald-runner
68         2883  2876  0 Sep10 ?        00:00:00 hald-addon-acpi: listening on acpid
68         2886  2876  0 Sep10 ?        00:00:00 hald-addon-keyboard: listening on /d
68         2890  2876  0 Sep10 ?        00:00:00 hald-addon-keyboard: listening on /d
root       2898  2876  0 Sep10 ?        00:02:46 hald-addon-storage: polling /dev/hda
root       2944     1  0 Sep10 ?        00:00:00 /usr/sbin/smartd -q never
root       2949     1  0 Sep10 tty2     00:00:00 /sbin/mingetty tty2
```

Process Information

```
root      2950     1  0 Sep10 tty3     00:00:00 /sbin/mingetty tty3
root      5365  2575  0 08:19 ?        00:00:00 sshd: rsimms [priv]
rsimms    5368  5365  0 08:19 ?        00:00:00 sshd: rsimms@pts/0
rsimms    5369  5368  0 08:19 pts/0    00:00:00 -bash
root      5969  2575  0 10:14 ?        00:00:00 sshd: valdemar [priv]
valdemar  5971  5969  0 10:14 ?        00:00:00 sshd: valdemar@pts/5
valdemar  5972  5971  0 10:14 pts/5    00:00:00 -bash
rsimms    6173  5369  0 10:36 pts/0    00:00:00 man ps
rsimms    6176  6173  0 10:36 pts/0    00:00:00 sh -c (cd /usr/share/man && (echo ".
rsimms    6177  6176  0 10:36 pts/0    00:00:00 sh -c (cd /usr/share/man && (echo ".
rsimms    6182  6177  0 10:36 pts/0    00:00:00 /usr/bin/less -is
root      6200  2575  0 10:37 ?        00:00:00 sshd: rsimms [priv]
rsimms    6203  6200  0 10:37 ?        00:00:00 sshd: rsimms@pts/6
rsimms    6204  6203  0 10:37 pts/6    00:00:00 -bash
root      6408  2575  0 11:07 ?        00:00:00 sshd: dymesdia [priv]
dymesdia  6418  6408  0 11:08 ?        00:00:00 sshd: dymesdia@pts/1
dymesdia  6419  6418  0 11:08 pts/1    00:00:00 -bash
rsimms    6524  6204  0 11:15 pts/6    00:00:00 ps -ef
lyonsrob 12891     1  0 Oct01 ?        00:00:00 SCREEN
lyonsrob 12892 12891  0 Oct01 pts/3    00:00:00 /bin/bash
root     29218     1  0 Oct15 tty1     00:00:00 /sbin/mingetty tty1
[rsimms@opus ~]$
```

Job Control



```
/home/cis90/roddyduk $ find / -user roddyduk  2> /dev/null
█
```

Some commands, like the one we used in Lab 7, takes a long time to complete. Until it finishes you can't type any more commands!

*It is running in the **foreground***

Job Control
A feature of the bash shell

Foreground processes

- Processes that receive their input and write their output to the terminal.
- The parent shell waits on these processes to die.

Background Processes

- Processes that do not get their input from a user keyboard.
- The parent shell does not wait on these processes; it re-prompts the user for next command.

Job Control

A feature of the bash shell

Ctrl-Z *(non CIS 90 accounts)*
or Ctrl-F *(CIS 90 student accounts)*

- Stops (suspends) a foreground process by sending it a "TTY Stop" (SIGTSTP) signal

bg

- resumes the currently suspended process and runs it in the background

Job Control

A feature of the bash shell

Ctrl-Z or Ctrl-F

- To send a SIGTSTP signal from the keyboard
- Stops (suspends) a foreground process

```
/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;
```

CIS 90 accounts use Ctrl-F)

```
[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;
```

Other Opus accounts use Ctrl-Z)

The bash environment for the CIS 90 accounts was customized to use a different keystroke for sending a SIGTSTP signal than other Opus accounts.

*Example: Suspending a **find** command*

Job Control

A feature of the bash shell

Ctrl-Z or Ctrl-F (SIGTSTP)

- Stops (suspends) a foreground process

```
[rsimms@opus ~]$ find / -name "stage[12]"  2> /dev/null
/boot/grub/stage1
/boot/grub/stage2
/usr/share/grub/i386-redhat/stage1
/usr/share/grub/i386-redhat/stage2

[1]+  Stopped                 find / -name "stage[12]" 2> /dev/null
[rsimms@opus ~]$
```

***Ctrl-F** (CIS 90 accounts) or **Ctrl-Z** (other accounts) is tapped while find is running*

PID 7587 is stopped (S=Sleeping, R=Running, T=sTopped)

```
[rsimms@opus ~]$ ps -l  -u rsimms
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME
5 S   201  5368  5365  0  75   0 -  2460 -      ?        00:00:00
0 S   201  5369  5368  0  75   0 -  1165 wait   pts/0    00:00:00
5 S   201  6203  6200  0  75   0 -  2491 -      ?        00:00:00
0 S   201  6204  6203  0  75   0 -  1165 -      pts/6    00:00:00
0 T   201  7587  6204  5  77   0 -  1145 finish pts/6    00:00:00
0 R   201  7588  5369  0  77   0 -  1062 -      pts/0    00:00:00
[rsimms@opus ~]$
```

Example: Resuming a ***find*** *command*

Job Control

A feature of the bash shell

bg

- Starts the currently suspended process and runs it in the background

```
[rsimms@opus ~]$ find / -name "stage[12]"  2> /dev/null
/boot/grub/stage1
/boot/grub/stage2
/usr/share/grub/i386-redhat/stage1
/usr/share/grub/i386-redhat/stage2

[1]+  Stopped                 find / -name "stage[12]" 2> /dev/null
[rsimms@opus ~]$ bg
[1]+ find / -name "stage[12]" 2> /dev/null &
[rsimms@opus ~]$
```

bg *resumes the find command which then finishes*

PID 7587 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  75   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  7696  5369  0  77   0 -  1062 -      pts/0    00:00:00 ps
[rsimms@opus ~]$
```

Example:
Suspending a
***sleep** command*

Job Control

A feature of the bash shell

Ctrl-Z or Ctrl-F (SIGTSTP)

- Stops (suspends) a foreground process

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

[1]+  Stopped                 sleep 5
```

***Ctrl-F** (CIS 90 accounts)*
*or **Ctrl-Z** (other accounts)*
is tapped while find
is running

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 ~]$
```

Example:
Resuming a
***sleep** command*

Job Control

A feature of the bash shell

bg

- Starts the currently suspended process running in the foreground

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

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

***bg** resumes the sleep command and it finishes*

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 ~]$
```

Job Control
A feature of the bash shell

&

- Append to a command to run it in the background

fg

- Brings the most recent background process to the foreground

jobs

- Lists all background jobs

Job Control
A feature of the bash shell

The & has sleep run in the background and jobs shows the shows it as the one and only background job

```
[rsimms@opus ~]$ sleep 10 &
[1] 7761
[rsimms@opus ~]$ jobs
[1]+  Running                 sleep 10 &
[rsimms@opus ~]$ fg
sleep 10
```

After fg, sleep now runs in the foreground. The prompt is gone. Need to wait until sleep finishes for prompt to return.

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

& is often used when running GUI tools like Firefox or Wireshark from the command line and you want to keep using the terminal for more commands while those applications run.

Signals

Signals

Signals are asynchronous messages sent to processes

They can result in one of three courses of action:

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

Signals are sent:

- Using the kill command: **`$ kill -# PID`**
  - Where # is the signal number and PID is the process id.
  - if no number is specified, SIGTERM is sent.
- Using special keystrokes
  - limited to just a few signals

Use kill –l to see all signals

Signals

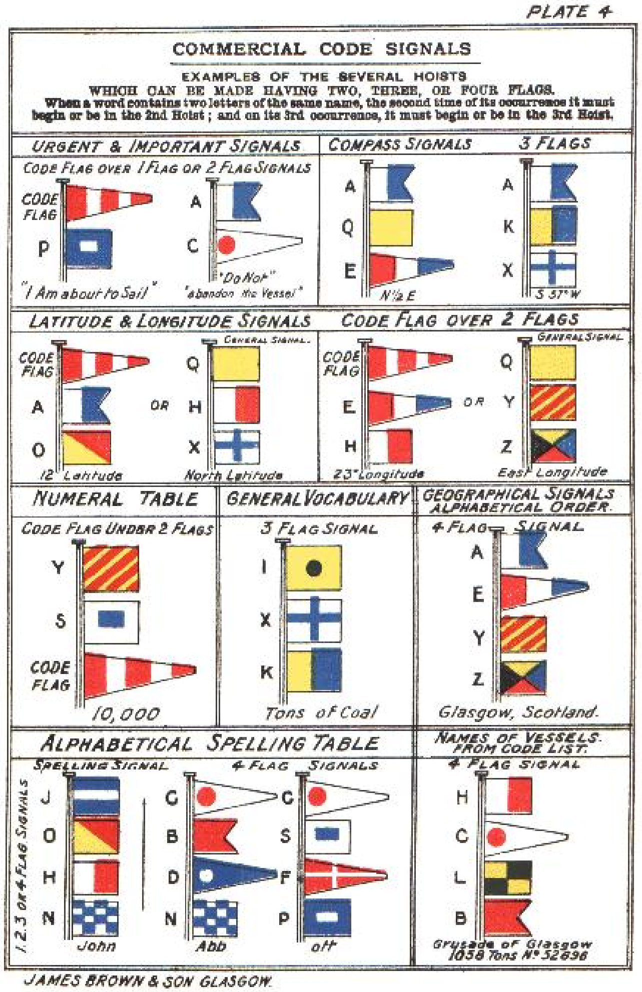
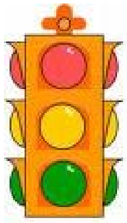


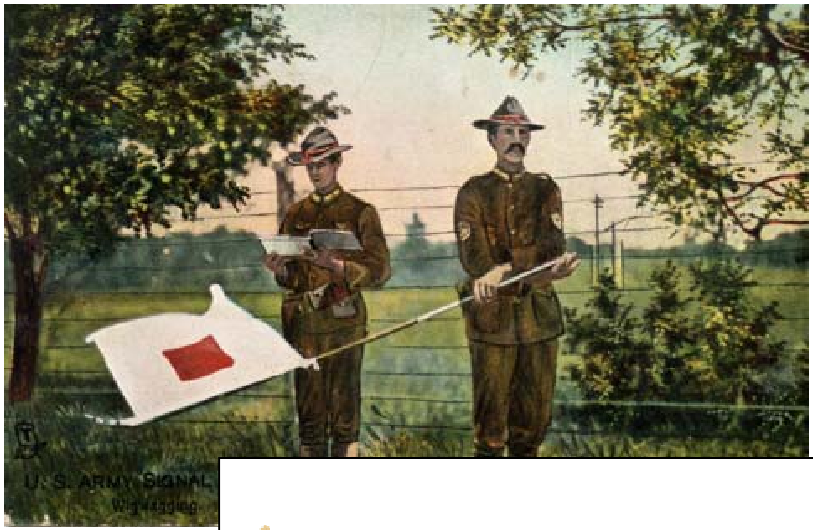
PLATE 4

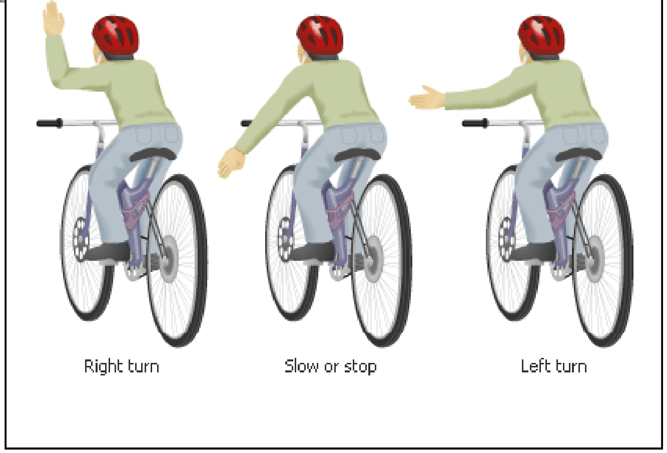
COMMERCIAL CODE SIGNALS

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

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

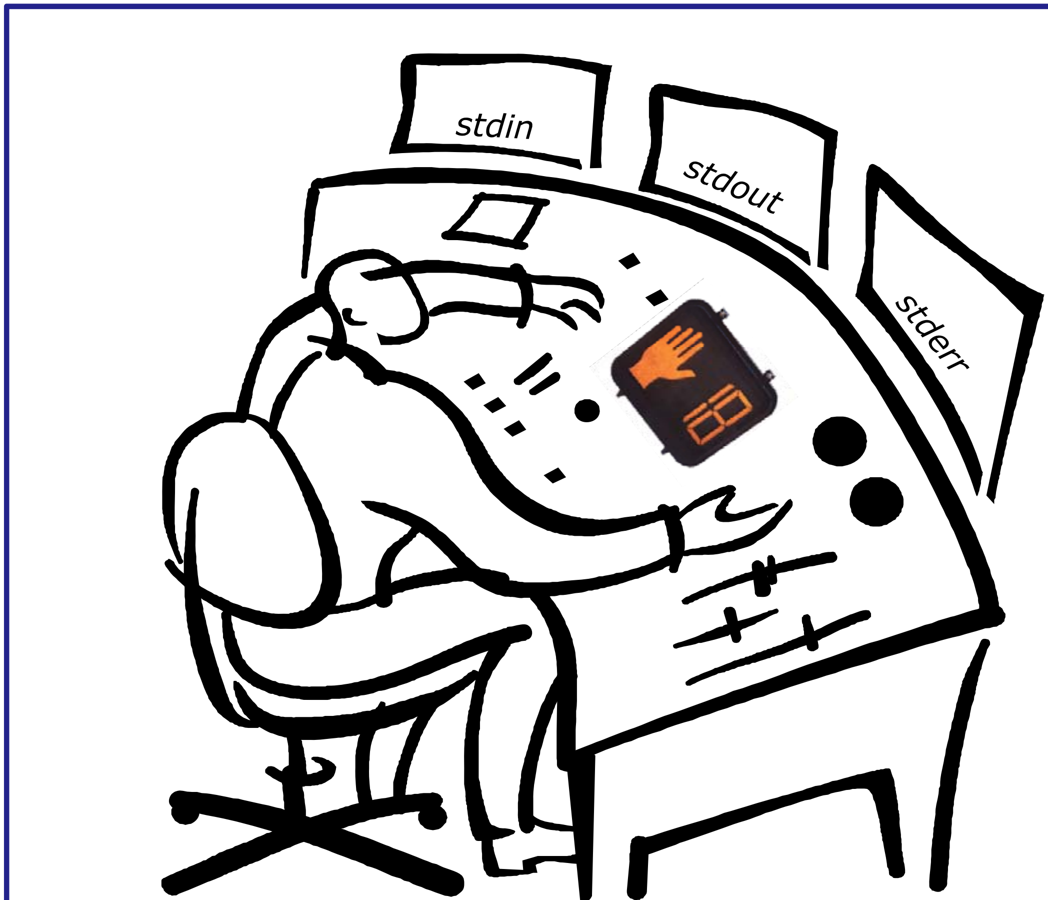






Signals

Signals are asynchronous messages sent to processes



Running process getting a signal

Asynchronous means it can happen at any time

Signals

Signals are asynchronous messages sent to processes

Running process gets a signal

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

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
/home/cis90/roddyduk $
```

Signals
Special keystrokes

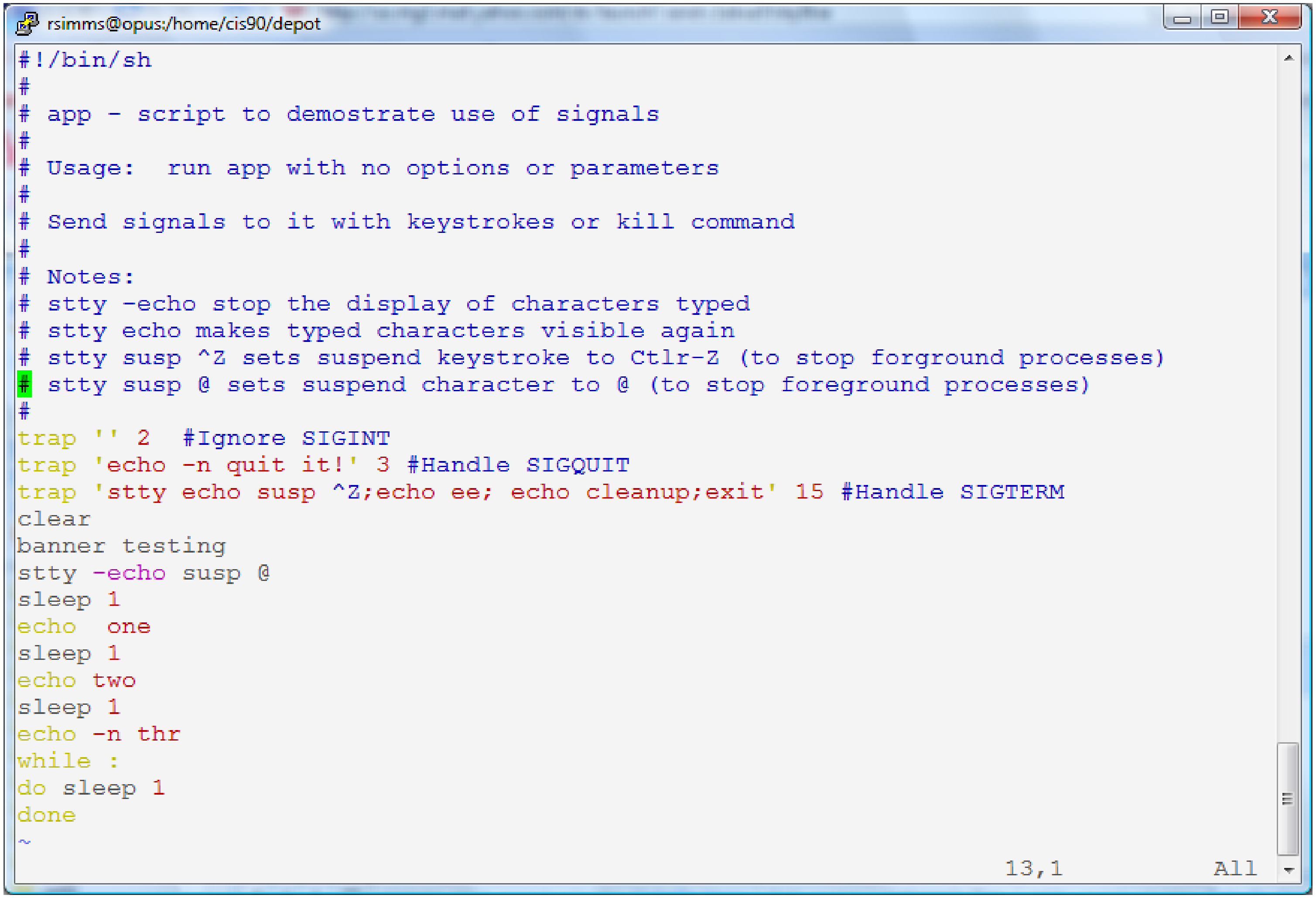
```
/home/cis90/roddyduk $ stty -a
speed 38400 baud; rows 26; columns 78; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^F; rprnt = ^R;
werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
```

```
[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;
```

use stty –a to see special keystrokes

Signals

Jim's app script

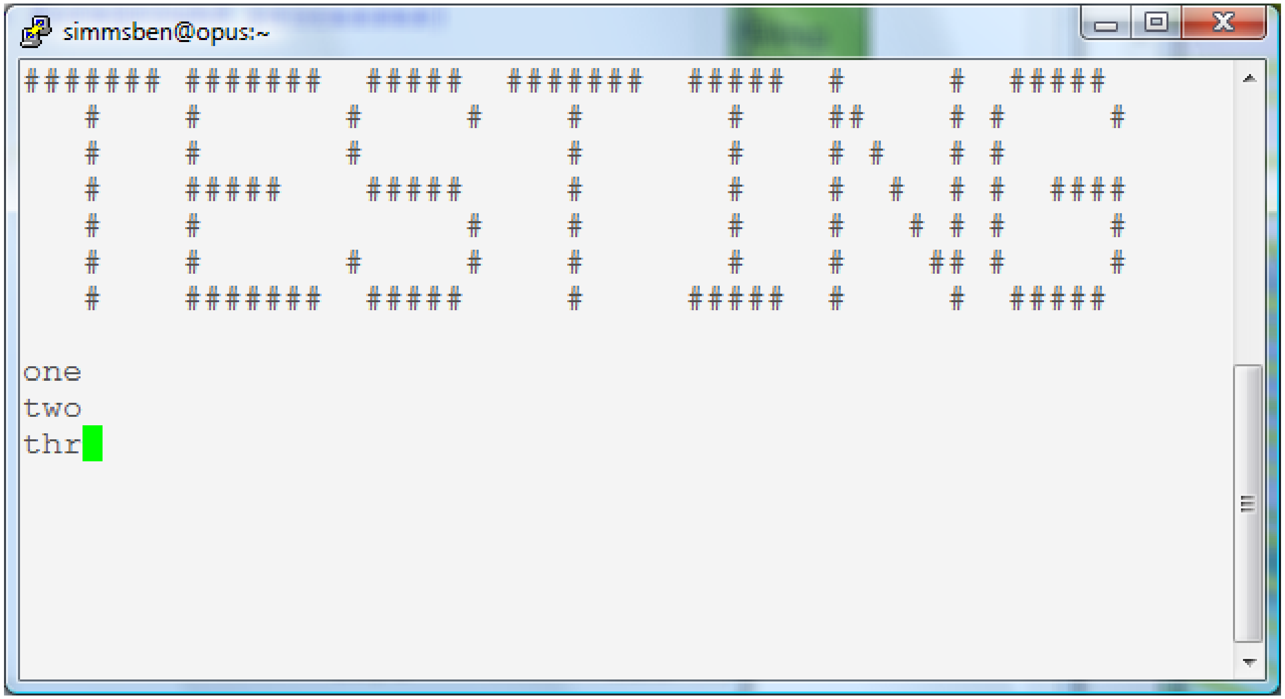


rsimms@opus:/home/cis90/depot

```
#!/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
```

Signals

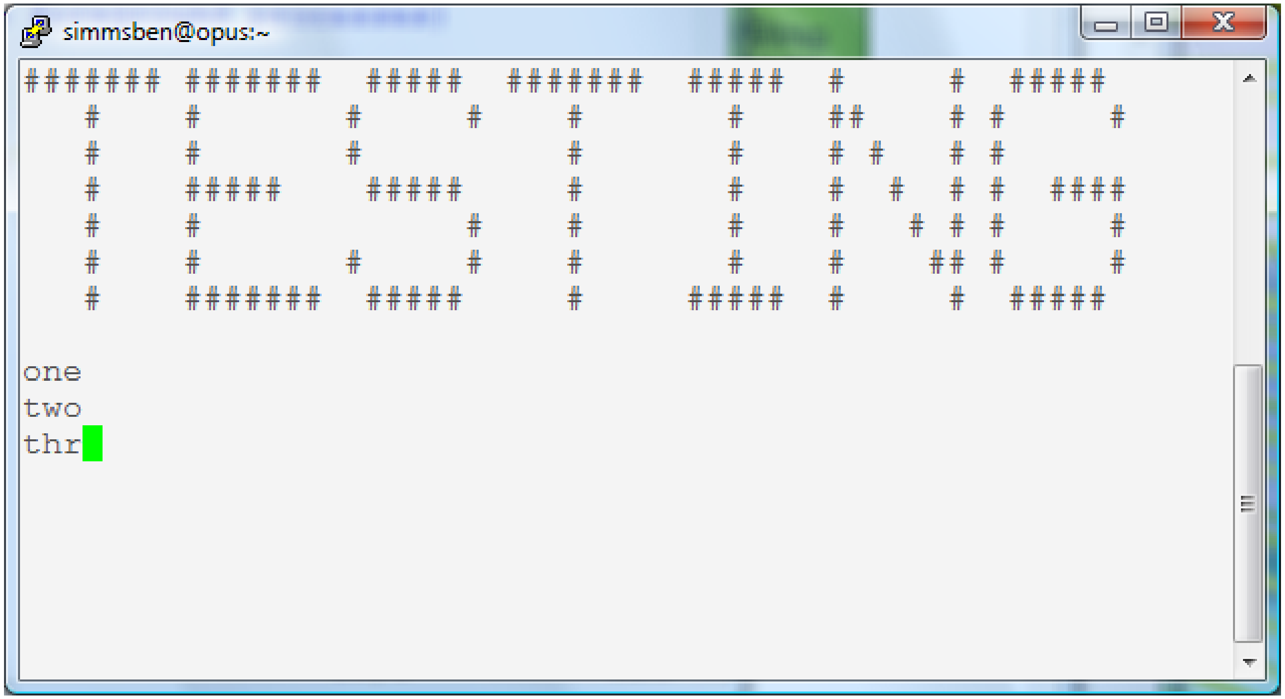
Benji runs app



```
####### #######  #####  #######  #####  #     #  #####
   #    #       #     #    #       #    ##    # #     #
   #    #       #          #       #    # #   # #
   #    #####    #####     #       #    #  #  # #   ####
   #    #             #    #       #    #   # # #     #
   #    #       #     #    #       #    #    ## #     #
   #    #######  #####     #     #####  #     #  #####

one
two
thr
```

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

Signals
Benji runs app

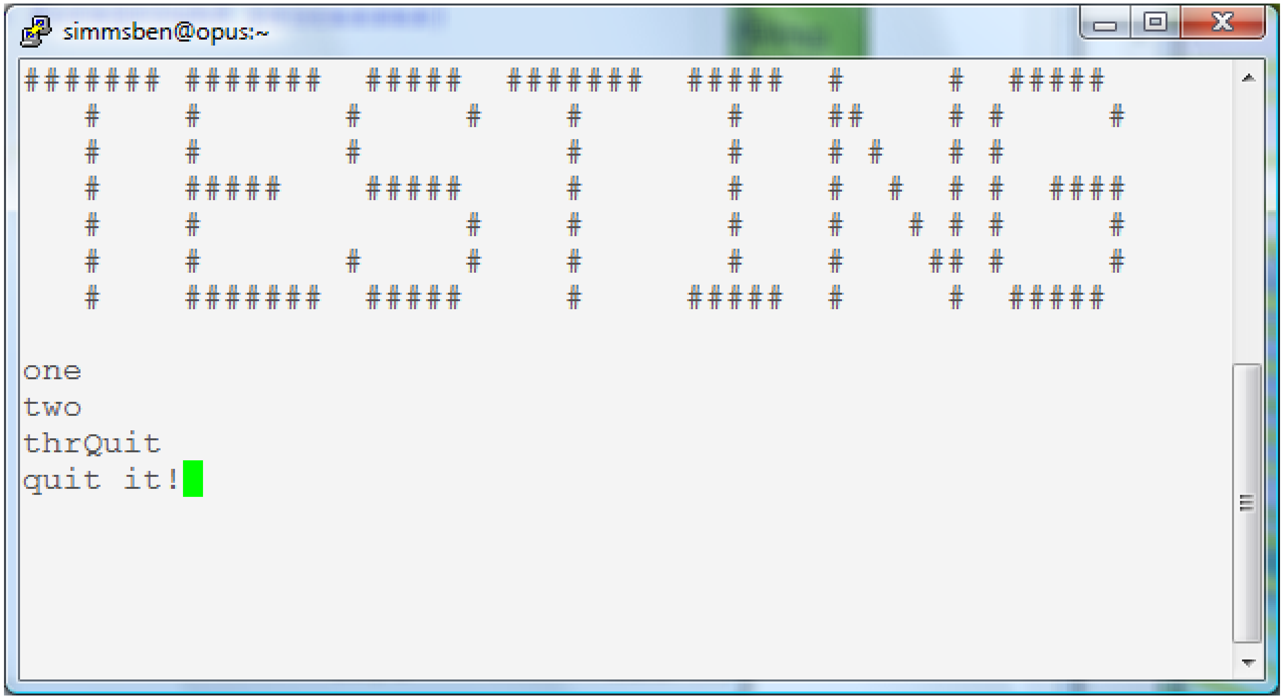


```
####### #######  #####  #######  #####  #     #  #####
   #    #       #     #    #       #    ##    # #     #
   #    #       #          #       #    # #   # #
   #    #####    #####     #       #    #  #  # #  ####
   #    #             #    #       #    #   # # #     #
   #    #       #     #    #       #    #    ## #     #
   #    #######  #####     #     #####  #     #  #####

one
two
thr
```

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

Signals

Benji runs app

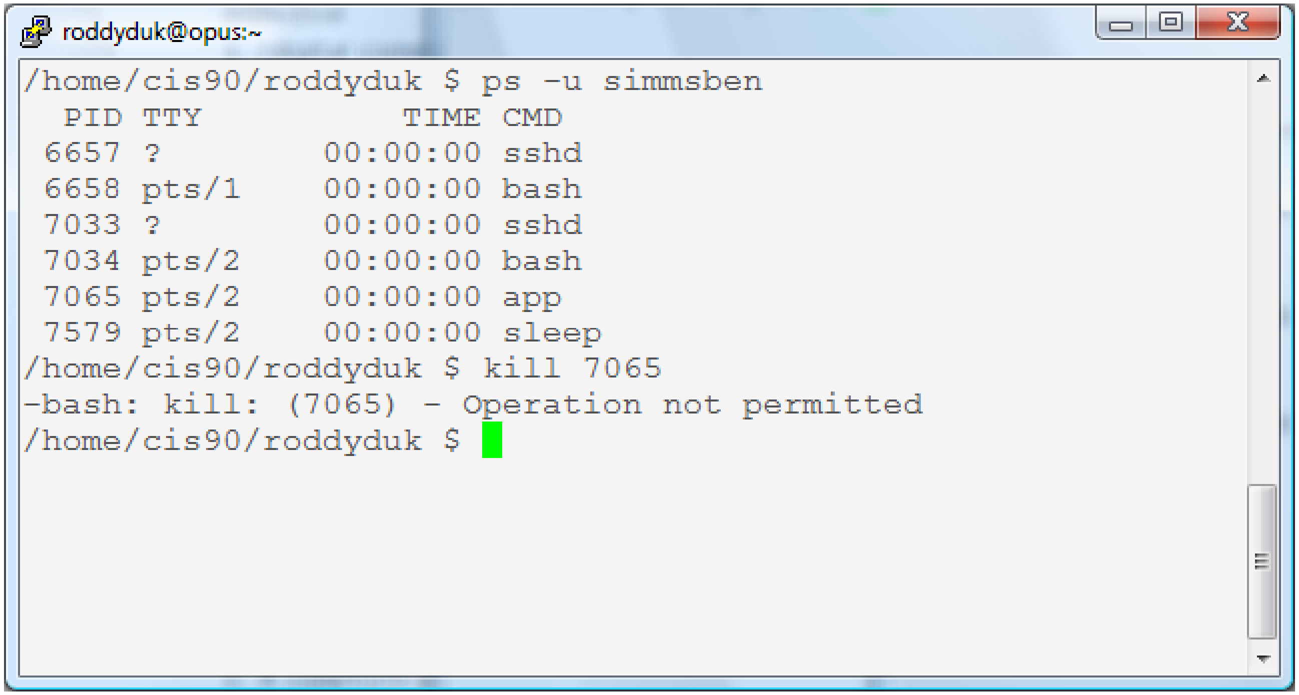


```
####### #######  #####  #######  #####  #     #  #####
   #    #       #     #    #       #    ##    # #     #
   #    #       #          #       #    # #   # #
   #    #####    #####     #       #    #  #  # #  ####
   #    #             #    #       #    #   # # #     #
   #    #       #     #    #       #    #    ## #     #
   #    #######  #####     #     #####  #     #  #####

one
two
thrQuit
quit it!
```

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

Signals

Benji runs app

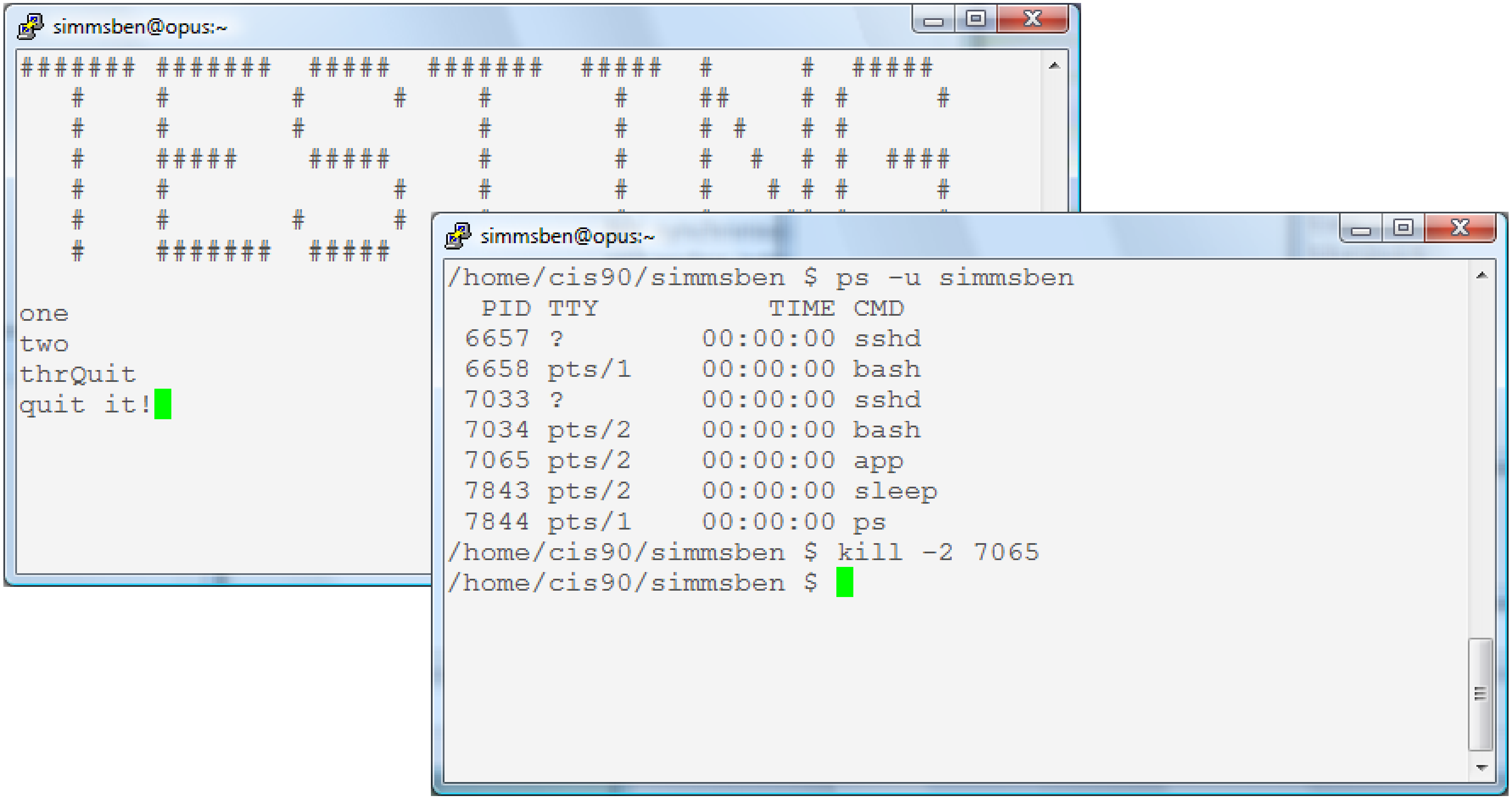


```
/home/cis90/roddyduk $ ps -u simmsben
  PID TTY          TIME CMD
 6657 ?        00:00:00 sshd
 6658 pts/1    00:00:00 bash
 7033 ?        00:00:00 sshd
 7034 pts/2    00:00:00 bash
 7065 pts/2    00:00:00 app
 7579 pts/2    00:00:00 sleep
/home/cis90/roddyduk $ kill 7065
-bash: kill: (7065) - Operation 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



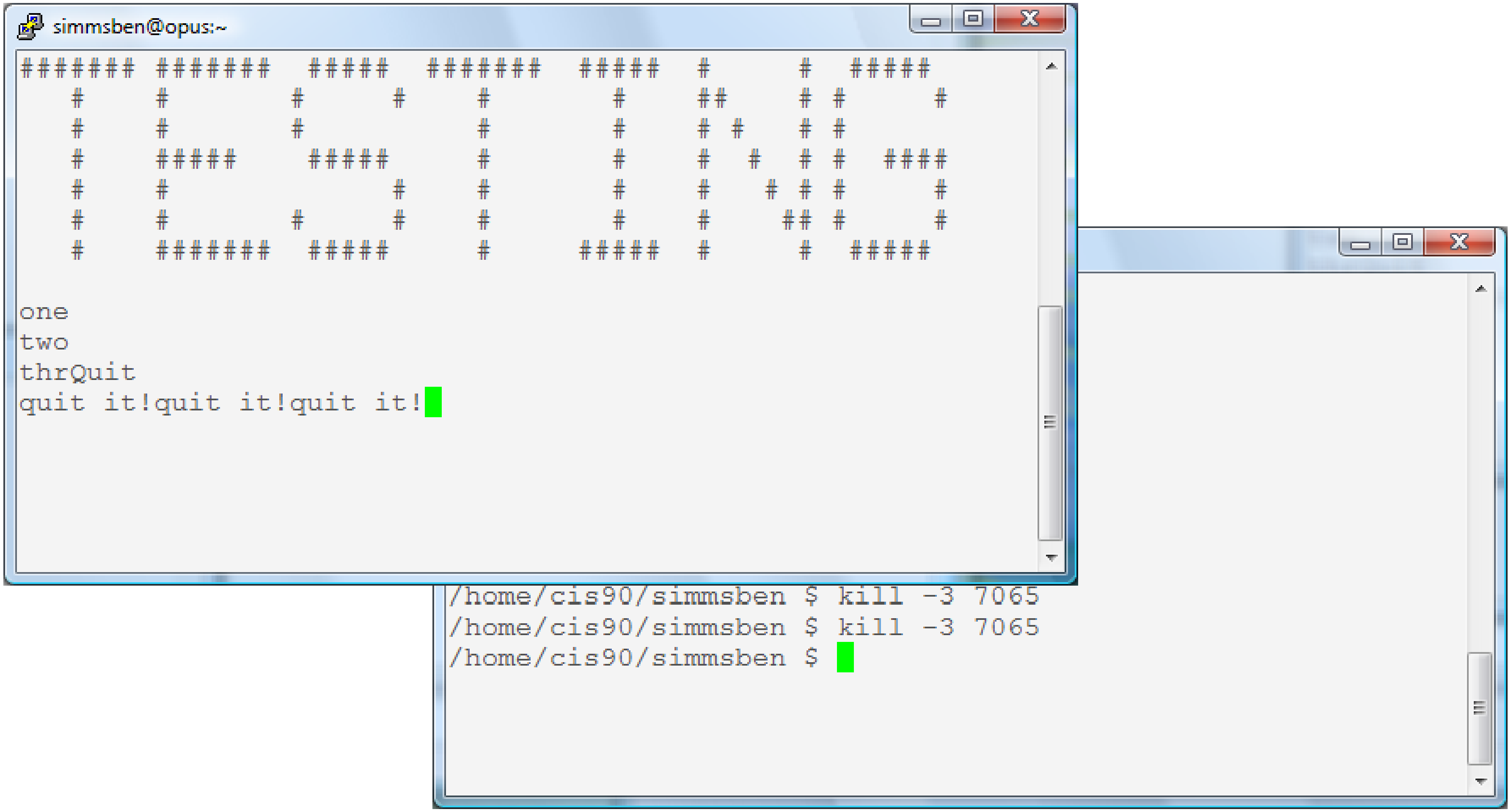
```
####### #######  #####  #######  #####  #     #  #####
   #    #       #     #    #       #    ##    # #     #
   #    #       #          #       #    # #   # #
   #    #####     #####     #       #    #  #  # #  ####
   #    #             #    #       #    #   # # #     #
   #    #       #     #
   #    #######  #####

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



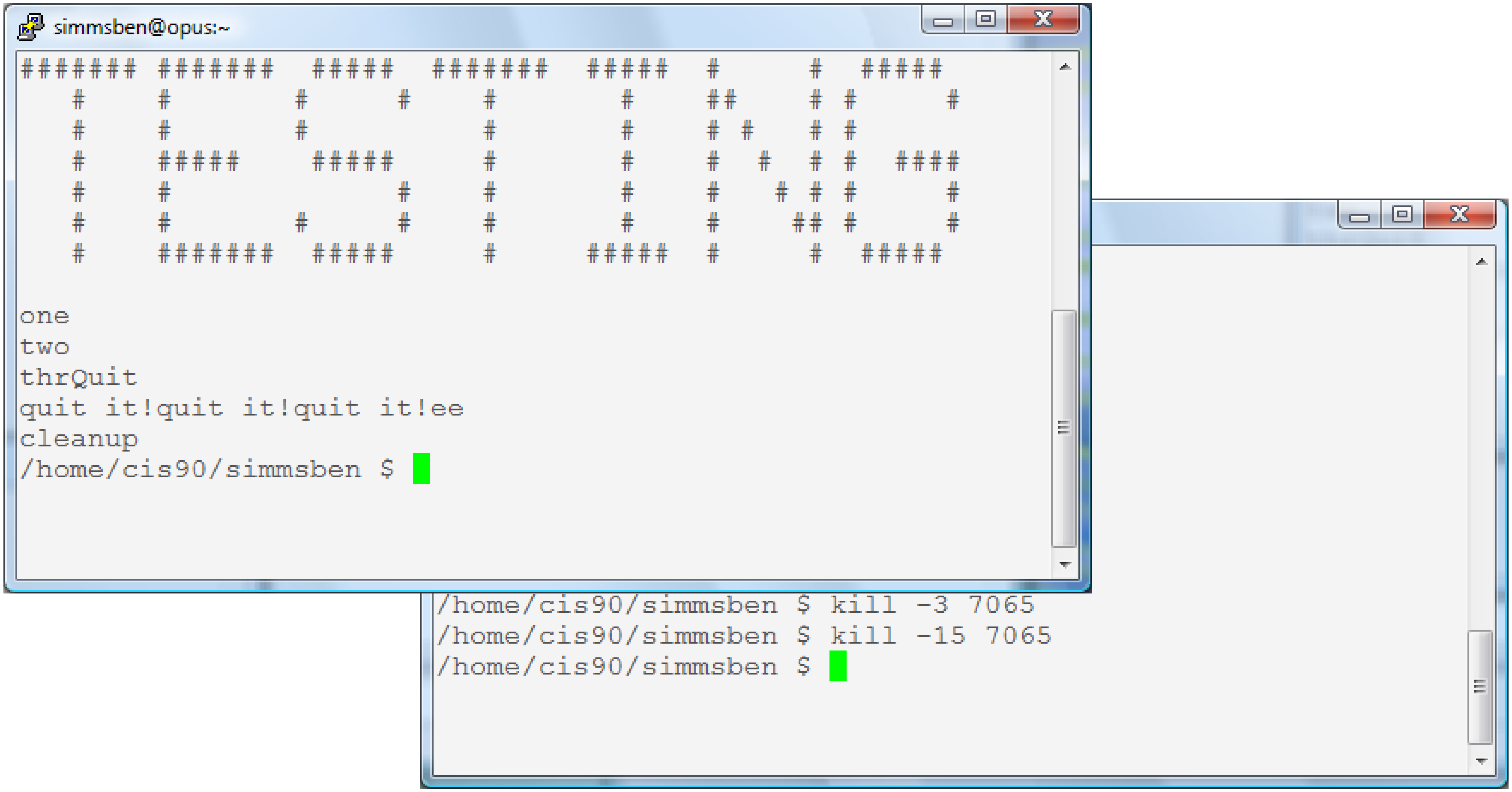
```
####### #######  #####  #######  #####  #     #  #####
   #    #       #     #    #       #    ##    # #     #
   #    #       #          #       #    # #   # #
   #    #####    #####     #       #    #  #  # #  ####
   #    #             #    #       #    #   # # #     #
   #    #       #     #    #       #    #    ## #     #
   #    #######  #####     #     #####  #     #  #####

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 anty and sends two SIGQUITs (-3) but the app process shrugs them off with "quit it!" messages

Signals

Benji runs app



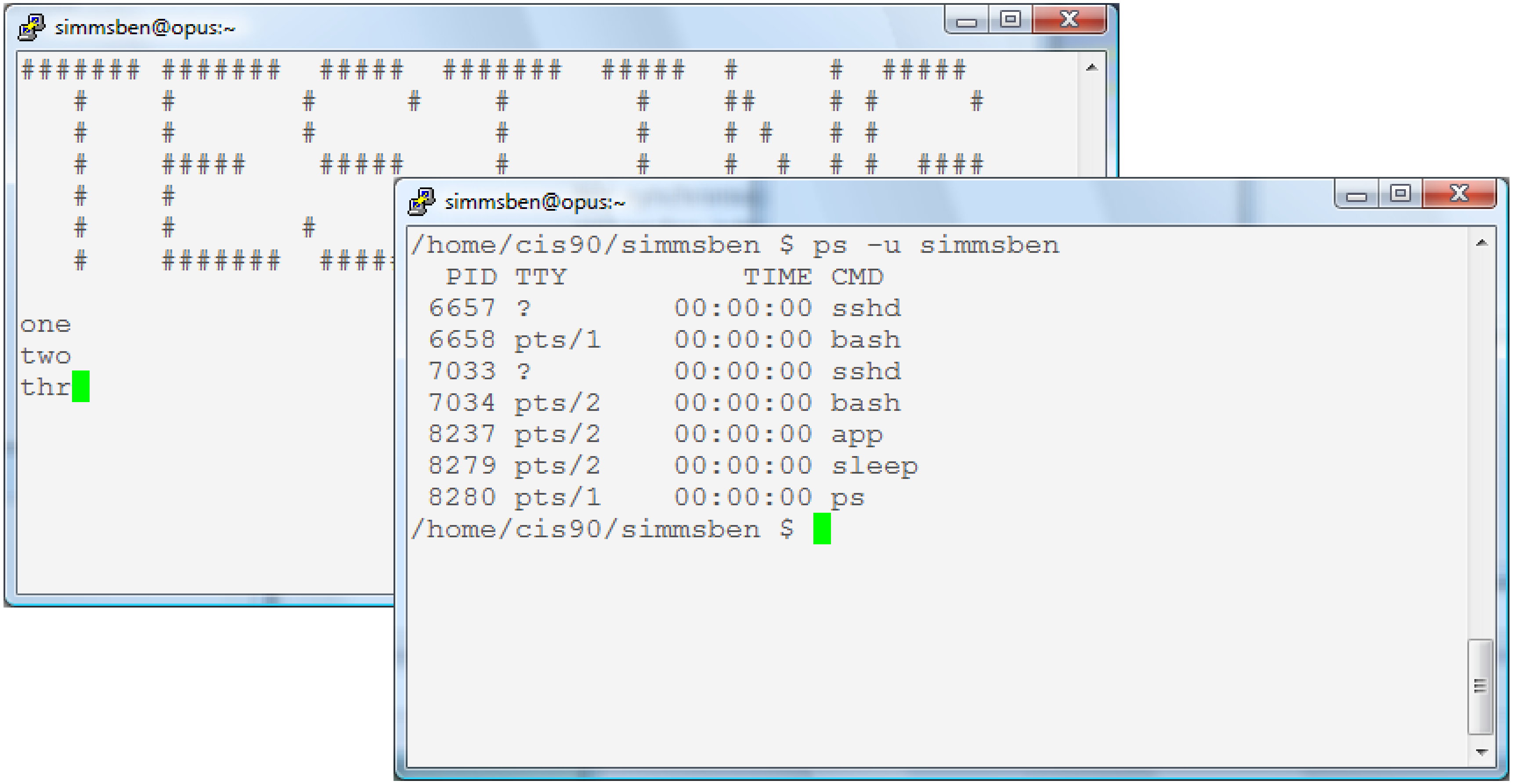
```
####### #######  #####  #######  #####  #     #  #####
   #    #       #     #    #       #    ##    # #     #
   #    #       #          #       #    # #   # #
   #    #####    #####     #       #    #  #  # #  ####
   #    #             #    #       #    #   # # #     #
   #    #       #     #    #       #    #    ## #     #
   #    #######  #####     #     #####  #     #  #####

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 (-15) this time and the app process finishes, cleans up and exits

Signals

Benji runs app



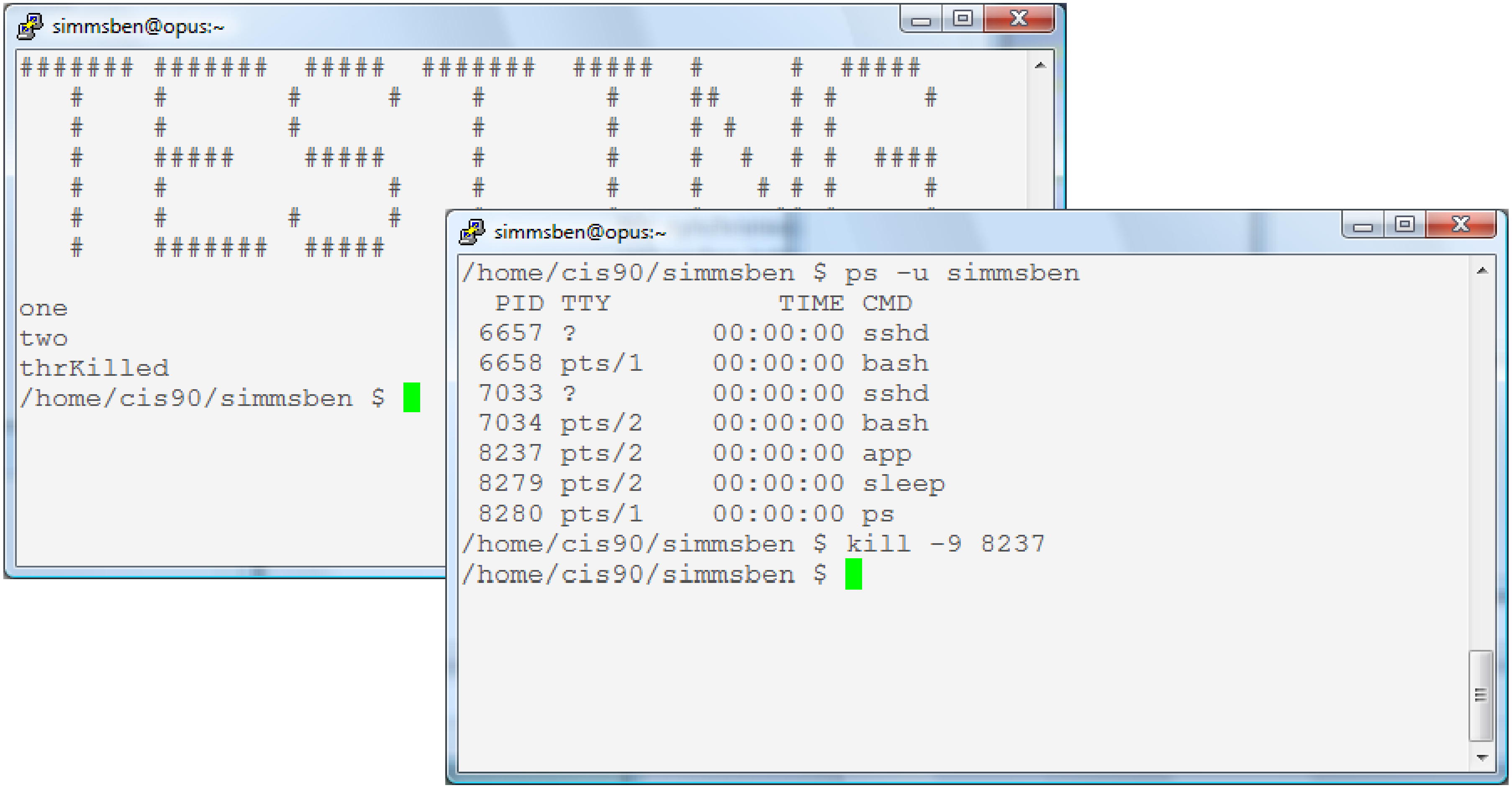
```
####### #######  #####  #######  #####  #     #  #####
   #    #       #     #    #       #    ##    # #     #
   #    #       #          #       #    # #   # #
   #    #####    #####     #       #    #  #  # #   ####
   #    #
   #    #       #
   #    ####### ####


one
two
thr
```

```
/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



```
####### #######  #####  #######  #####  #     #  #####
   #    #       #     #    #       #    ##    # #     #
   #    #       #          #       #    # #   # #
   #    #####    #####     #       #    #  #  # #  ####
   #    #             #    #       #    #   # # #     #
   #    #       #     #
   #    #######  #####


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

```
/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

Signals

Class Exercise

- View the ../depot/app program
- Look for the three trap handlers
  - Signal 2 (SIGINT)
  - Signal 3 (SIGQUIT)
  - Signal 15 (SIGTERM)
- Run app
- Try sending it a SIGINT from the keyboard (Ctrl-C)
- Try sending it a SIGQUIT from the keyboard (Ctrl-\)
- Login to another Putty session
  - Use the ps –u $LOGNAME to find the app PID
  - Send it a SIGINT (kill -2 PID)
  - Send it a SIGQUIT (kill -3 PID)
  - Now send either a SIGKILL (9) or SIGTERM (15) but first decide if app can clean up or not when it gets your signal.

Load Balancing

Load Balancing

So that the multiprocessing CPU on a UNIX system does not get overloaded, some processes need to be run during low peak hours such as early in the morning or later in the day.

The at command is for this purpose.

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

```
$ at 10:30pm < batch_file

$ at 11:59pm
at> cat files.out bigshell > lab08
at> cp lab08 /home/rsimms/cis90/$LOGNAME
at> Ctrl-D
$
Note: the Ctrl-d must be entered as the first character on the last
line.
```

Load Balancing

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 12/12/2010 < job1
job 28 at 2008-12-12 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

```
/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 futures that were scheduled by at command*

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

The Test

Add read permission on test2
Run mail-q9-all script in /home/rsimms/cis90/test02

Wrap up

New commands:

Ctrl-Z or F	Suspends a foreground process
bg	Resumes suspended process
&	Runs command in the background
fg	Brings background job to foreground
jobs	show background jobs
kill	Send a signal to a process
at	Run job once in the future
atq	Show all *at* jobs queued to run
atrm	Remove *at* jobs from queue

Next Class

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

Quiz questions for next class:

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

Backup

find command

Find all directories starting in my home directory that start with a capital B, S, Y or A.

```
[roddyduk@opus ~]$ find . -type d -name "[BSYA]*"
find: ./Hidden: Permission denied
./poems/Blake
./poems/Shakespeare
./poems/Yeats
./poems/Anon
[roddyduk@opus ~]$
```

Find all files starting in my home directory that contain town

```
[roddyduk@opus ~]$ find .  -name "\*town\*"
find: ./Hidden: Permission denied
[roddyduk@opus ~]$ find .  -name "*town*"
find: ./Hidden: Permission denied
./edits/small_town
./edits/better_town
[roddyduk@opus ~]$
```

grep command tips

*Use the "^ *1" to match all lines that start with zero or more blanks, a 1, followed by a blank.*

```
/home/cis90/roddyduk $ cat testfile
671 buster
 99 scout
125 benji
  1 homer
934 duke
100 lucy
 10 smokey
322 sky

/home/cis90/roddyduk $ grep "^ *1 " testfile
  1 homer
```

Use the B option to list lines preceding the matched line

```
/home/cis90/roddyduk $ grep -B 3  "^ *1 " testfile
671 buster
 99 scout
125 benji
  1 homer
```