



## Rich's lesson module checklist

*Last updated 11/1/2017*

- Slides and lab posted
- WB converted from PowerPoint
- Print out agenda slide and annotate page numbers
  
- Flash cards
- Page numbers
- 1<sup>st</sup> minute quiz
- Web Calendar summary
- Web book pages
- Commands
  
- Lab 8 tested and published
  
- Real test published and scheduled on Canvas
- Real test servers startup and shutdown configured
- Real test accommodations (length and due time) made
  
- Practice test scheduled to prior to real test start
- Practice test servers scheduled to shutdown prior to real test
  
- 9V backup battery for microphone
- Backup slides, CCC info, handouts on flash drive
- Key card for door
  
- Update CCC Confer and 3C Media portals



### **Student Learner Outcomes**

1. Navigate and manage the UNIX/Linux file system by viewing, copying, moving, renaming, creating, and removing files and directories.
2. Use the UNIX features of file redirection and pipelines to control the flow of data to and from various commands.
3. With the aid of online manual pages, execute UNIX system commands from either a keyboard or a shell script using correct command syntax.

## Introductions and Credits



Jim Griffin

- Created this Linux course
- Created Opus and the CIS VLab
- Jim's site: <http://cabrillo.edu/~jgriffin/>



Rich Simms

- HP Alumnus
- Started teaching this course in 2008 when Jim went on sabbatical
- Rich's site: <http://simms-teach.com>

And thanks to:

- John Govsky for many teaching best practices: e.g. the First Minute quizzes, the online forum, and the point grading system (<http://teacherjohn.com/>)



## Student checklist for attending class

Rich's Cabrillo College CIS Classes  
CIS 90 Calendar

CIS 90 (Fall 2014) Calendar

Course Dates: [Genda](#) **Calendar**

**CIS 90**

Lesson	Date	Topics	Link
	9/2	<p><b>Class and Linux Overview</b></p> <ul style="list-style-type: none"> <li>Understand how the course will work</li> <li>High-level overview of computers, operating systems, and virtual machines</li> <li>Overview of LINUX/Linux market and architecture</li> <li>Using SSH for remote network logs</li> <li>Using terminals and the command line</li> </ul> <p><b>Methods</b></p> <p><a href="#">Presentation slides (download)</a></p> <p><b>Supplemental</b></p> <ul style="list-style-type: none"> <li>Howto #148: Logging into Opus (command)</li> </ul> <p><b>Assignments</b></p> <ul style="list-style-type: none"> <li>Student Survey</li> <li>Lab 1</li> </ul> <p><b>CCS Center</b></p> <p><a href="#">Enter virtual classroom</a></p>	
		<p>Quiz 1</p> <p>Commands</p>	

1. Browse to:  
**http://simms-teach.com**
2. Click the **CIS 90** link.
3. Click the **Calendar** link.
4. Locate today's lesson.
5. Find the **Presentation slides** for the lesson and **download** for easier viewing.
6. Click the **Enter virtual classroom** link to join CCC Confer.
7. Log into Opus-II with Putty or ssh command.

Note: Blackboard Collaborate Launcher only needs to be installed once. It has already been downloaded and installed on the classroom PC's.





# Student checklist for suggested screen layout

Google

CCC Confer

Downloaded PDF of Lesson Slides

The screenshot displays a virtual classroom interface with several overlapping windows. On the left is a Blackboard calendar page for 'Rich's Cabrillo College CIS 90 Classes'. In the center is a CCC Confer window showing a video feed of 'Rich Simms' and a participant list. Overlaid on the confer window is a Google map of the San Francisco Bay Area. To the right is an Adobe Acrobat Pro window displaying a PDF of lesson slides titled 'The CIS 90 System Playground'. Below the slides is a terminal window showing a login sequence for 'Opus-11' with a password prompt and a 'Welcome to Opus' message.

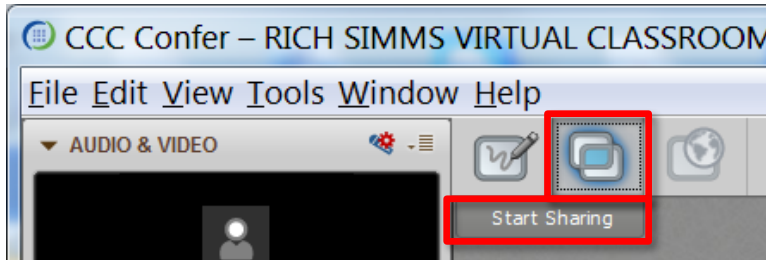
CIS 90 website Calendar page

One or more login sessions to Opus-II

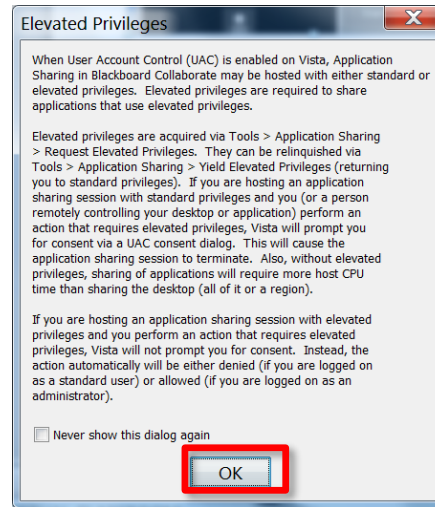


# Student checklist for sharing desktop with classmates

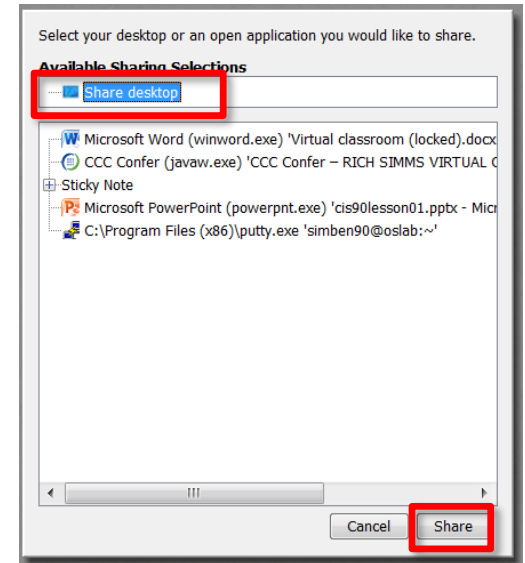
1) Instructor gives you sharing privileges



2) Click overlapping rectangles icon. If white "Start Sharing" text is present then click it as well.



3) Click OK button.



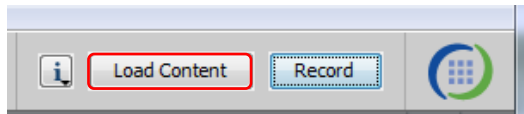
4) Select "Share desktop" and click Share button.



# Rich's CCC Confer checklist - setup

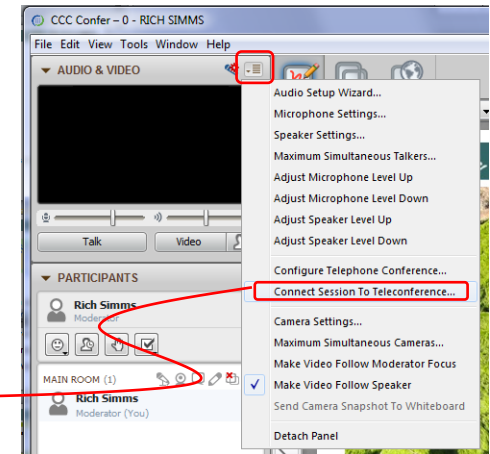
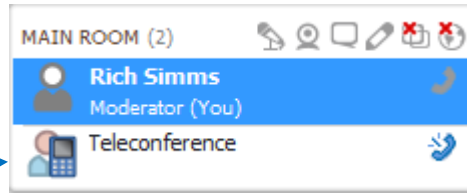


[ ] Preload White Board



[ ] Connect session to Teleconference

*Session now connected to teleconference*



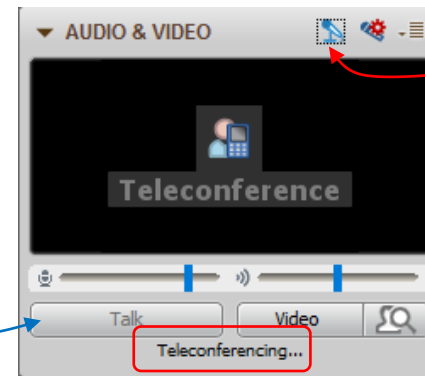
[ ] Is recording on?



*Red dot means recording*

[ ] Use teleconferencing, not mic

*Should be grayed out*



*Should change from phone handset icon to little Microphone icon and the Teleconferencing ... message displayed*



## Rich's CCC Confer checklist - screen layout



The screenshot displays a Windows desktop with several applications open:

- CCC Confer - 0 - RIC...:** A teleconference window showing a video feed of Rich Simms, a list of participants (Rich Simms as Moderator), and a chat window.
- foxit for slides:** A Foxit Reader window displaying a PDF document titled 'cis90lesson07.pdf'. A red box labeled 'foxit for slides' points to the document.
- chrome:** A Google Chrome browser window displaying a quiz page from 'simms-teach.com/docs/cis90/cis-90-TEST-1-Fall-12.pdf'. A red box labeled 'chrome' points to the browser window.
- putty:** A PuTTY terminal window showing a shell session for 'simben90@oslab:~'. The terminal displays a file tree with directories like 'boot', 'bin', 'etc', and 'sbin', and a prompt 'What command copies th...'. A red box labeled 'putty' points to the terminal window.
- vSphere Client:** A vSphere Client window showing the management interface for a vCenter environment, including a tree view of virtual machines and a 'Recent Tasks' table.

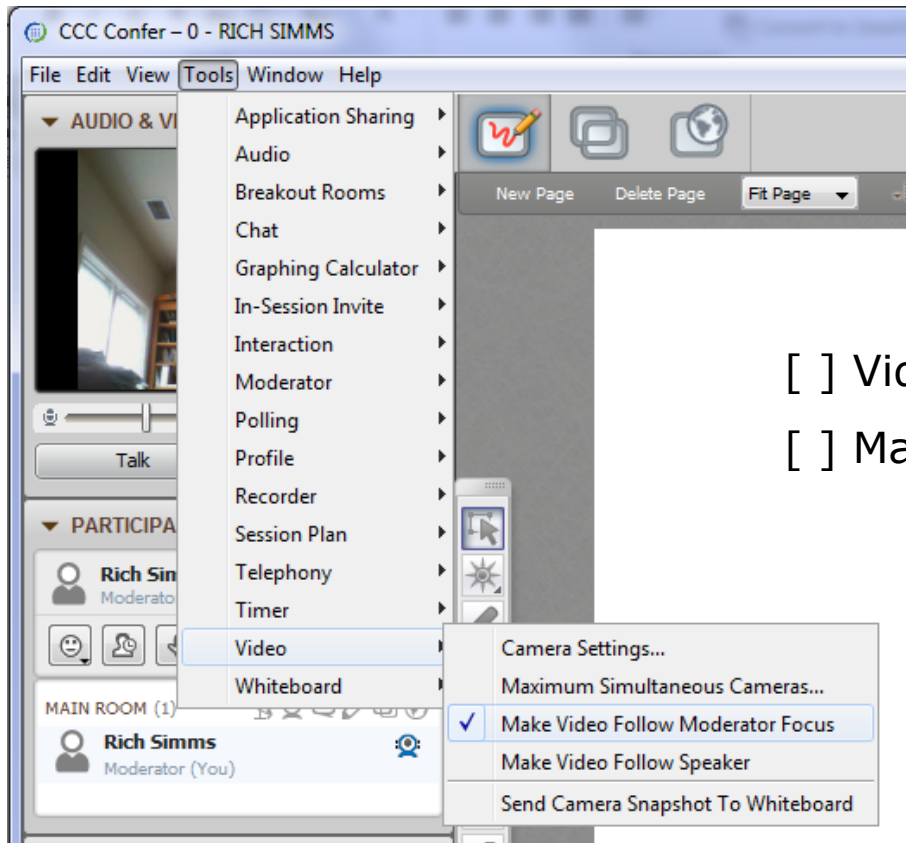
[ ] layout and share apps







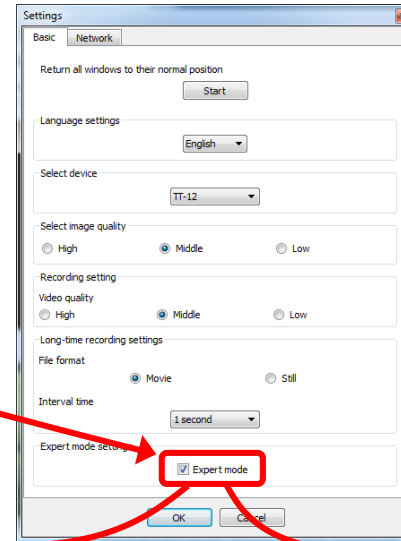
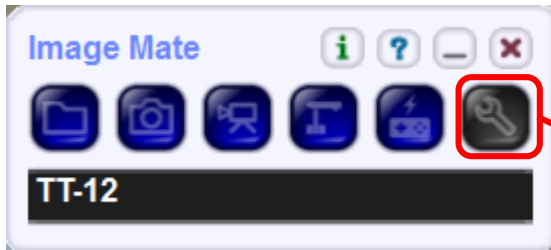
# Rich's CCC Confer checklist - webcam setup



- [ ] Video (webcam)
- [ ] Make Video Follow Moderator Focus



# Rich's CCC Confer checklist - Elmo



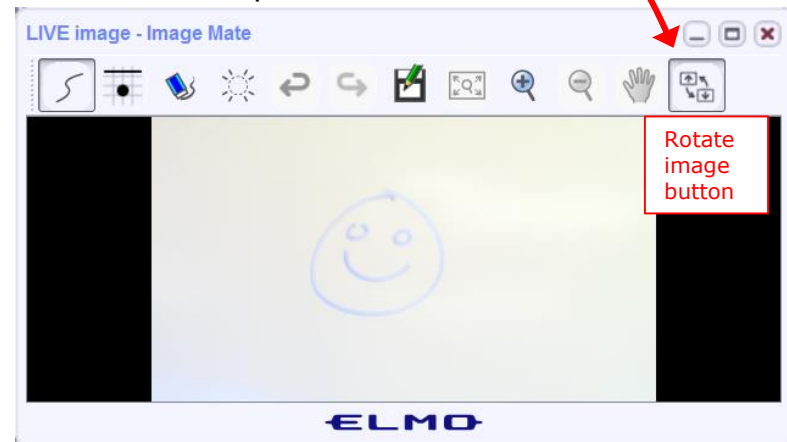
The "rotate image" button is necessary if you use both the side table and the white board.

Quite interesting that they consider you to be an "expert" in order to use this button!

Elmo rotated down to view side table



Elmo rotated up to view white board



Run and share the Image Mate program just as you would any other app with CCC Confer

## Rich's CCC Confer checklist - universal fixes

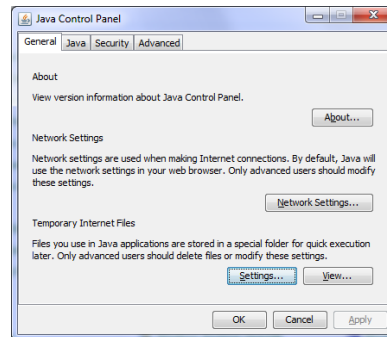
Universal Fix for CCC Confer:

- 1) Shrink (500 MB) and delete Java cache
- 2) Uninstall and reinstall latest Java runtime
- 3) <http://www.cccconfer.org/support/technicalSupport.aspx>

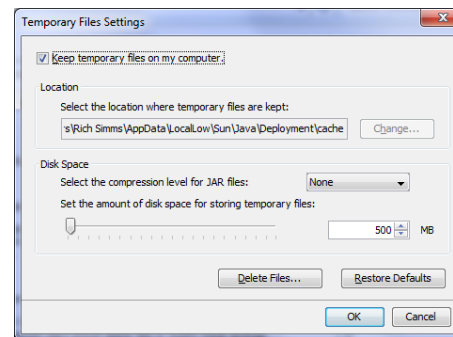
Control Panel (small icons)



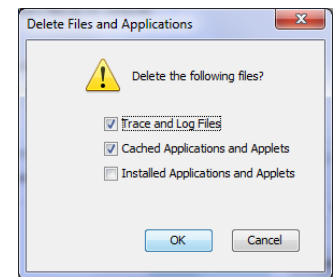
General Tab > Settings...



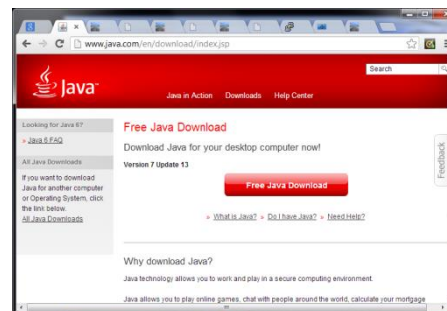
500MB cache size



Delete these

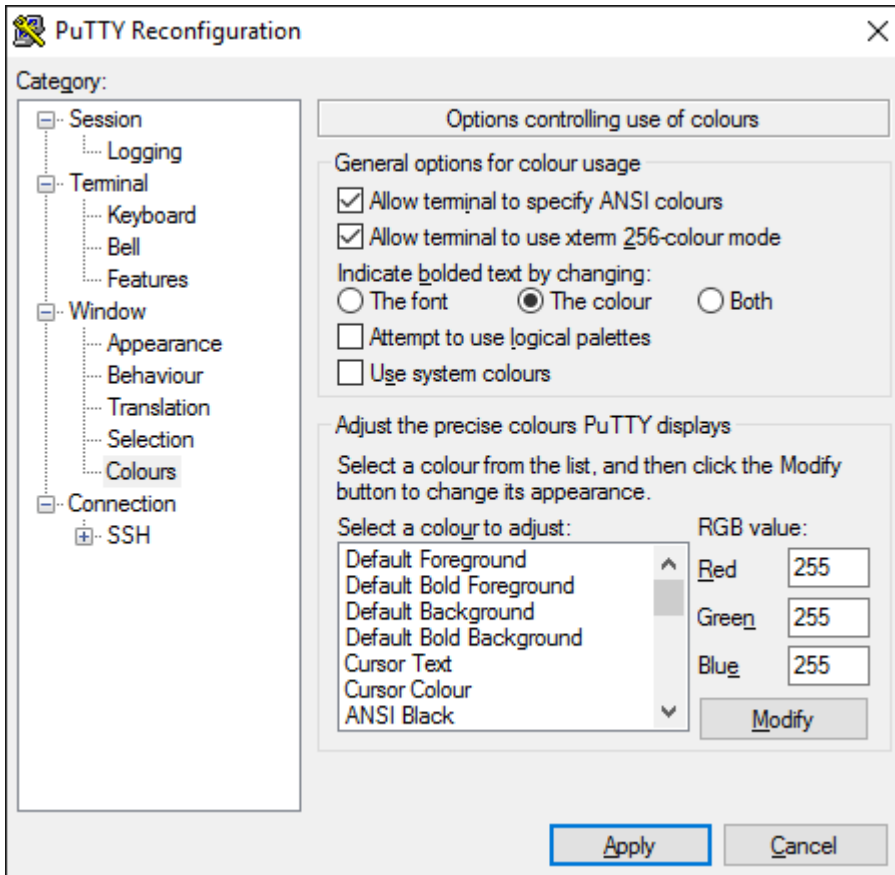


Google Java download





## Rich's CCC Confer checklist - Putty Colors



### Putty Colors

Default Foreground 255 255 255  
 Default Bold Foreground 255 255 255  
 Default Background 51 51 51  
 Default Bold Background 255 2 85  
 Cursor Text 0 0 0  
 Cursor Color 0 255 0  
 ANSI Black 77 77 77  
 ANSI Black Bold 85 85 85  
 ANSI Red 187 0 0  
 ANSI Red Bold 255 85 85  
 ANSI Green 152 251 152  
 ANSI Green Bold 85 255 85  
 ANSI Yellow 240 230 140  
 ANSI Yellow Bold 255 255 85  
 ANSI Blue 205 133 63  
 ANSI Blue Bold 135 206 235  
 ANSI Magenta 255 222 173  
 ANSI Magenta Bold 255 85 255  
 ANSI Cyan 255 160 160  
 ANSI Cyan Bold 255 215 0  
 ANSI White 245 222 179  
 ANSI White Bold 255 255 255

<http://looselytyped.blogspot.com/2013/02/zenburn-pleasant-color-scheme-for-putty.html>





# Start

# Sound Check

*Students that dial-in should mute their line using \*6 to prevent unintended noises distracting the web conference.*

*Instructor can use \*96 to mute all student lines.*

## *Volume*

*\*4 - increase conference volume.*

*\*7 - decrease conference volume.*

*\*5 - increase your voice volume.*

*\*8 - decrease your voice volume.*



Instructor: **Rich Simms**

Dial-in: **888-886-3951**

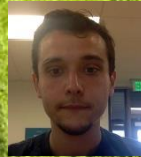
Passcode: **136690**



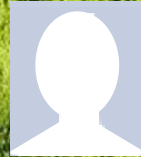
Marvin



William



Vinny



Hayden



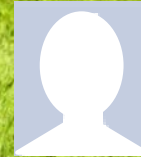
Nick



Ramon



Nicholas



Manuel



Damien



Oscar



Daniel P.



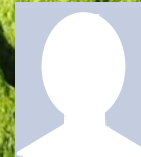
Jason



Brian



Vincent P.



Kyle



Sam X.



Jacobs



Tyler



Sean



Karina



Michael J.



Alejandro



Moises



Joseph



David



Emmanuel



Ben



## First Minute Quiz

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

**No Quiz today ... test instead**

For credit email answers to:

[risimms@cabrillo.edu](mailto:risimms@cabrillo.edu)

within the **first few minutes of class**





# UNIX Processes

## Objectives

- 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

## Agenda

- Questions
- FYI: shell debugging
- Housekeeping
- Process definition
- Process life cycle
- ps command
- Job control
- Signals
- Load balancing
- Assignment
- Wrap up
- Test #2



# Questions



# Questions?

Lesson material?

Labs? Tests?

How this course works?

- Graded work in home directories
- Answers in /home/cis90/answers

*Who questions much, shall learn much, and retain much.*

- Francis Bacon

*If you don't ask, you don't get.*

- Mahatma Gandhi

Chinese  
Proverb

他問一個問題，五分鐘是個傻子，他不問一個問題仍然是一個傻瓜永遠。

*He who asks a question is a fool for five minutes; he who does not ask a question remains a fool forever.*

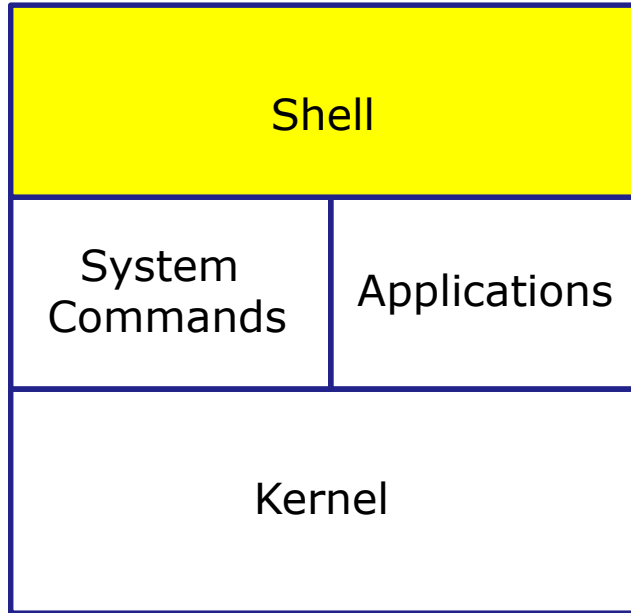
# FYI

shell debugging and { }





# The Shell **Parse** Step



- 1) **Prompt** for a command
- 2) **Parse** (interpret metacharacters, expand file names and dissect command line into options and arguments)
- 3) **Search** for program (along the path)
- 4) **Execute** program by loading into memory (becomes a process), hookup input and outputs, and pass along command line options and arguments.
- 5) **Nap** (wait till process is done)
- 6) **Repeat**

## Important Concept to Understand

- It's a **team effort** between the **shell** and the **command** to process what a user types after the prompt
- The shell does the initial work during the **parse step** and provides a list of options and arguments to the command
- The command may not see everything the user actually typed in

## FYI **set -x, set +x**



```
/home/cis90/rodduk $ set -x           Enable shell debugging
```

```
+ set -x
++ echo -ne '\033]0;rodduk@opus:~'
```

```
/home/cis90/rodduk $ type /bin/pi*
```

```
+ type /bin/ping /bin/ping6
/bin/ping is /bin/ping
/bin/ping6 is /bin/ping6
++ echo -ne '\033]0;rodduk@opus:~'
```

*Shows what arguments  
are actually passed to  
the command being run*

```
/home/cis90/rodduk $ 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;rodduk@opus:~'
```

```
/home/cis90/rodduk $ set +x           Disable shell debugging
```

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

## FYI set -x, set +x



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

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

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

*Shows variables in double (weak) quotes get expanded, while those in single (strong) quotes do not*

```
/home/cis90/rodduk $ set +x           Disable shell debugging
+ set +x
/home/cis90/rodduk $
```



## FYI set -x, set +x



```
/home/cis90/milhom $ set -x Enable shell debugging
++ printf '\033]0;%s@%s:%s\007' milhom90 oslab '~'
```

```
/home/cis90/milhom $ find . -name *treat*
+ find . -name treat1
find: './Hidden': Permission denied
./treat1
++ printf '\033]0;%s@%s:%s\007' milhom90 oslab '~'
```

```
/home/cis90/milhom $ find . -name *trick*
+ find . -name *trick*
find: './Hidden': Permission denied
./Miscellaneous/.trick6
./Poems/Shakespeare/.trick3
./Poems/Yeats/.trick2
./Poems/.trick5
./Poems/Blake/.trick4
./ssh/.trick1
++ printf '\033]0;%s@%s:%s\007' milhom90 oslab '~'
```

```
/home/cis90/milhom $ set +x Disable shell debugging
+ set +x
/home/cis90/milhom $
```

*Shows how filename expansion metacharacters are expanded or not depending on whether a match was found!*

## FYI using {}



*The braces {} are filename expansion metacharacters*

```
/home/cis90/simben $ mkdir fast
/home/cis90/simben $ ls fast
/home/cis90/simben $ touch fast/file{1,2,3,4,5}
/home/cis90/simben $ ls fast
file1  file2  file3  file4  file5
```

*Short hand for specifying multiple filenames at once*

```
/home/cis90/simben $ set -x
++ echo -ne '\033]0;simben90@opus:~'

/home/cis90/simben $ touch fast/file{1,2,3,4,5}
+ touch fast/file1 fast/file2 fast/file3 fast/file4 fast/file5
++ echo -ne '\033]0;simben90@opus:~'
```

*Showing  
how bash  
did the  
expansion  
above*

```
simben90@osla...
/home/cis90/simben $ ls -l
archives
bag
bigfile
bin
class
cmds
cruz
dead.letter
docs
dogs
dogs.inorder
dogs.tar
edits
errors
etc
Hidden
Hidden.tar
lab01-collection
lab01.graded
lab02-collection
lab02.graded
lab03.graded
lab04.graded
lab04-mydata
lab05.graded
lab06.graded
lab07
lab07.graded
labx2
letter
log
logs
myfiles
mylog
names
new
news
newer
old
olddir
poems
treat1
uh.bak
uhistory
what_am_i
whoami
wnoami
words
/home/cis90/simben $
```

```
/home/cis90/simben $ find -name *treat*
find: `./Hidden': Permission denied
./treat1
```

```
/home/cis90/simben $ find -name *trick*
find: `./Hidden': Permission denied
./poems/Shakespeare/.trick4
./poems/Yeats/.trick3
./poems/Neruda/.trick5
./poems/Dickenson/.trick6
./.testdir/.trick1
./.ssh/.trick2
/home/cis90/simben $
```



Why does the first command only find one of the six *treat* files ... yet the second command finds all six *trick* files?

Put your answer in the chat window



# Housekeeping





## Housekeeping

1. No labs due today!
2. Lab 8 is due next week.
3. Practice test & server will shut down shortly before the real test starts.
4. Test 2 during the last hour of class today
  - Canvas - timed test - 60 minutes
  - OPEN book, notes, computer
  - CLOSED mouths (work solo, don't ask for or give assistance to others)
  - Working students may take the test later in the day but it must be submitted by 11:59PM



# Test Instructions

## HONOR CODE:

This test is open book, open notes, and open computer. HOWEVER, you must work alone. You may not discuss the test questions or answers with others during the test period. You may not ask or receive assistance from anyone other than the instructor when doing this test. Likewise you may not give any assistance to anyone taking the test.

## INSTRUCTIONS:

Test system: sun-hwa-t2.cis.cabrillo.edu (port 22)

This test should be completed using the sun-hwa-t2 system only. Because this system is on a private network, log into Opus-II first, then ssh into sun-hwa-t2.

Grading will be based on your answers AND that you correctly implemented the "DO THIS FIRST" portion of each question.

**Highlighted text** indicates a portion of a question that differs from the practice test.

If you get stuck on a question and can't proceed you can ask the instructor for help and forfeit the point. The instructor will be available during class and available by email (risimms@cabrillo.edu) later in the evening from 8:00-10:00PM.

Please KEEP YOUR ANSWERS TO A SINGLE LINE ONLY !!

This test must be completed in one sitting. The submittal will be made automatically when the time is up. If you submit early by accident you will not be able to re-enter and continue. If that happens don't panic! Just email the instructor any remaining answers before the time is up.

## Heads up on Final Exam

Test #3 (final exam) is **MONDAY December 11th 1-3:50PM**

<b>Mon</b>	12/11	<b>Test #3 (the final exam)</b>	5 posts <a href="#">Lab X1</a> <a href="#">Lab X2</a>
		<b>Time</b> <ul style="list-style-type: none"> <li>MONDAY 1:00PM - 3:50PM in Room 828</li> </ul> <b>Materials</b> <ul style="list-style-type: none"> <li>Test (<a href="#">canvas</a>)</li> </ul> <b>CCC Confer</b> <ul style="list-style-type: none"> <li><a href="#">Enter virtual classroom</a></li> <li><a href="#">Class archives</a></li> </ul>	

*Extra credit  
labs and  
final posts  
due by  
11:59PM*

- All students will take the test at the same time. The test must be completed by **3:50PM**.
- Working and long distance students can take the test online via CCC Confer and Canvas.
- Working students will need to plan ahead to arrange time off from work for the test.
- Test #3 is **mandatory** (even if you have all the points you want)

## FALL 2017 FINAL EXAMINATIONS SCHEDULE DECEMBER 11 TO DECEMBER 16

### DAYTIME FINAL SCHEDULE

**Daytime Classes:** All times in bold refer to the beginning times of classes. **MW/Daily** means Monday alone, Wednesday alone, Monday and Wednesday **or any 3** or more days in any combination. **TTH** means Tuesday alone, Thursday alone, or Tuesday and Thursday. **Classes meeting other combinations of days and/or hours not listed must have a final schedule approved by the Division Dean.**

STARTING CLASS TIME / DAY(S)	EXAM HOUR	EXAM DATE
<i>Classes starting between:</i>		
6:30 am and 8:55 am, MW/Daily	7:00 am-9:50 am	Monday, December 11
9:00 am and 10:15 am, MW/Daily	7:00 am-9:50 am	Wednesday, December 13
10:20 am and 11:35 am, MW/Daily	10:00 am-12:50 pm	Monday, December 11
11:40 am and 12:55 pm, MW/Daily	10:00 am-12:50 pm	Wednesday, December 13
1:00 pm and 2:15 pm, MW/Daily	1:00 pm-3:50 pm	Monday, December 11
2:20 pm and 3:35 pm, MW/Daily	1:00 pm-3:50 pm	Wednesday, December 13
3:40 pm and 5:30 pm, MW/Daily	4:00 pm-6:50 pm	Monday, December 11

6:30 am and 8:55 am, TTh	7:00 am-9:50 am
9:00 am and 10:15 am, TTh	7:00 am-9:50 am
10:20 am and 11:35 am, TTh	10:00 am-12:50 pm
11:40 am and 12:55 pm, TTh	10:00 am-12:50 pm
1:00 pm and 2:15 pm, TTh	1:00 pm-3:50 pm
2:20 pm and 3:35 pm, TTh	1:00 pm-3:50 pm
3:40 pm and 5:30 pm, TTh	4:00 pm-6:50 pm
Friday am	9:00 am-11:50 am
Friday pm	1:00 pm-3:50 pm
Saturday am	9:00 am-11:50 am
Saturday pm	1:00 pm-3:50 pm

### CIS 90 Introduction to UNIX/Linux

Provides a technical overview of the UNIX/Linux operating system, including hands-on experience with commands, files, and tools. Recommended Preparation: CIS 1L or CIS 72.

Transfer Credit: Transfers to CSU/UC

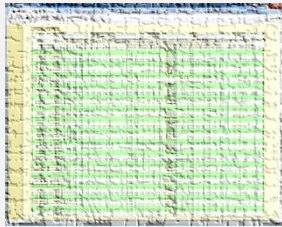
Section	Days	Times	Units	Instructor	Room
98169	W	1:00PM-4:05PM	3.00	R.Simms	OL
&	Arr.	Arr.		R.Simms	OL
Section 98169 is an ONLINE course. Meets weekly throughout the semester online during the scheduled times by remote technology with an additional 50 min online lab per week. For details, see instructor's web page at <a href="http://go.cabrillo.edu/online">go.cabrillo.edu/online</a> .					
98170	W	1:00PM-4:05PM	3.00	R.Simms	828
&	Arr.	Arr.		R.Simms	OL
Section 98170 is a Hybrid ONLINE course. Meets weekly throughout the semester at the scheduled times with an additional 50 min online lab per week. For details, see instructor's web page at <a href="http://go.cabrillo.edu/online">go.cabrillo.edu/online</a> .					

## Where to find your grades

*Send me your survey to get your LOR code name.*

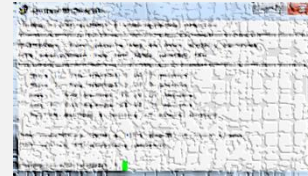
### The CIS 90 website Grades page

<http://simms-teach.com/cis90grades.php>



### Or check on Opus

**checkgrades** *codename*  
(where codename is your LOR codename)



Written by Jesse Warren a past CIS 90 Alumnus

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

**At the end of the term I'll add up all your points and assign you a grade using this table**

### Points that could have been earned:

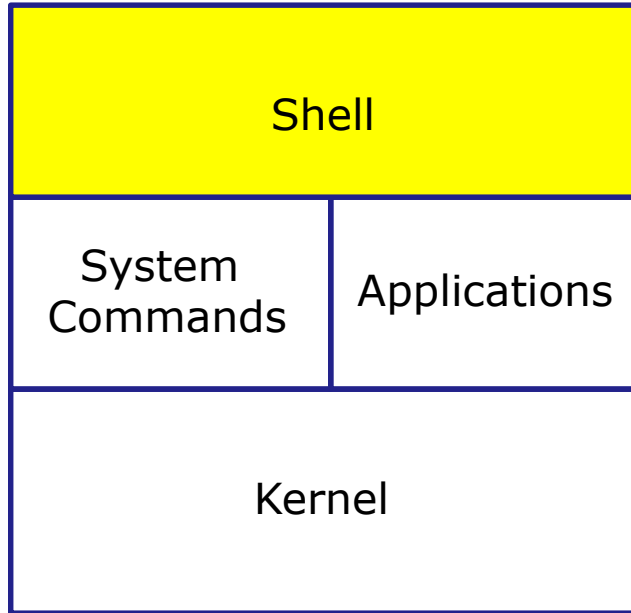
7 quizzes: 21 points  
 7 labs: 210 points  
 1 test: 30 points  
 2 forum quarters: 40 points  
**Total: 301 points**



# Process Definition



# The Shell **Execute** Step

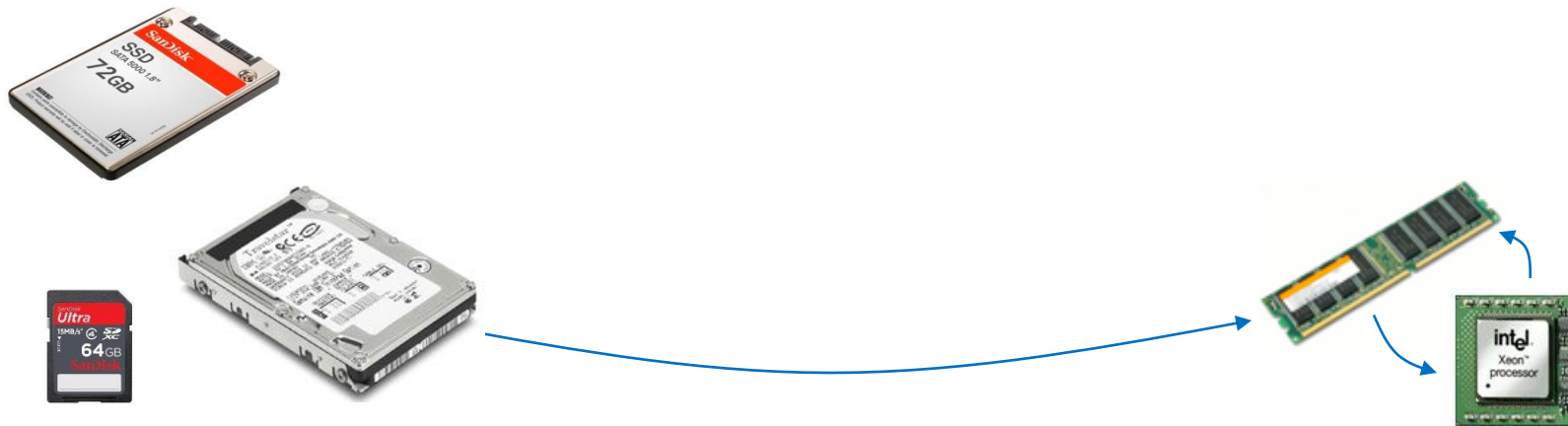


- 1) **Prompt** for a command
- 2) **Parse** (interpret metacharacters, expand file names and dissect command line into options and arguments)
- 3) **Search** for program (along the path)
- 4) **Execute** program by loading it into memory (as a process) and providing it with the parsed options/arguments. In addition hook up all inputs and outputs (stdin, stdout and stderr)
- 5) **Nap** (wait till process is done)
- 6) **Repeat**



## Definition of a process

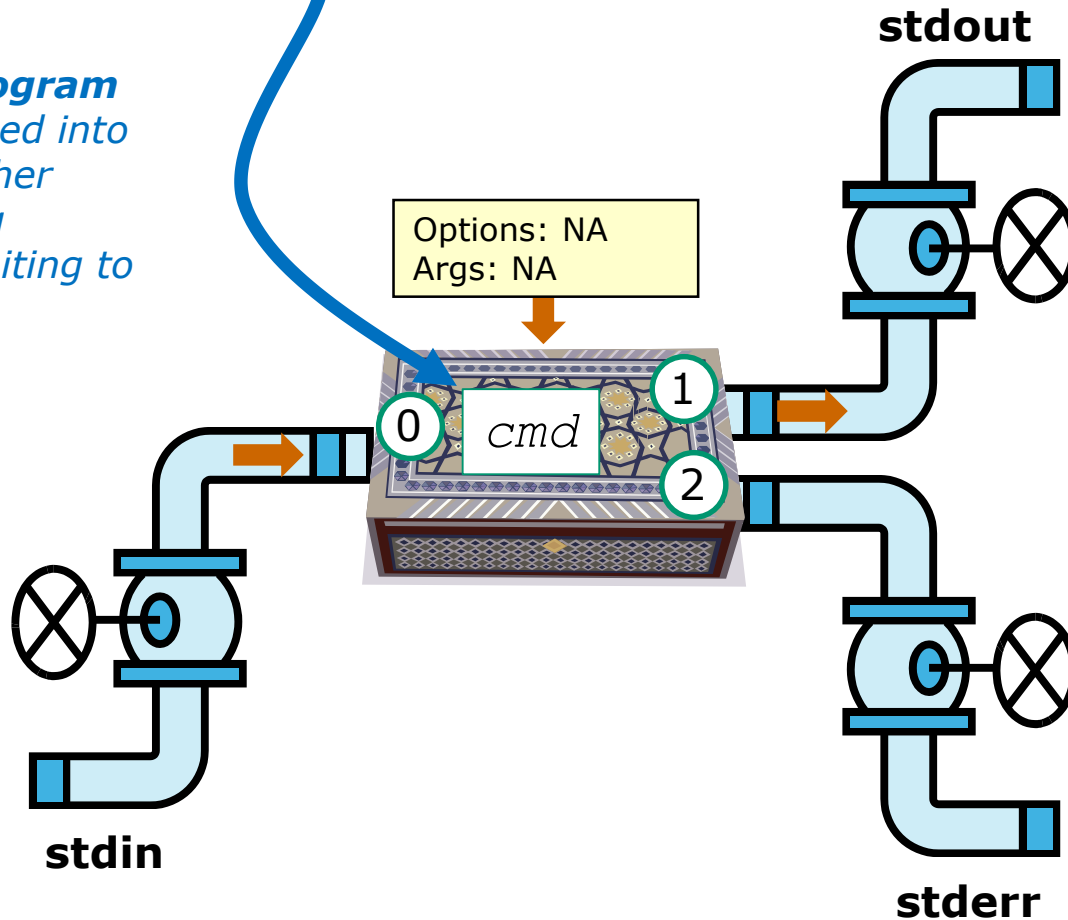
A **process** is a **program** that has been copied (loaded) into memory by the kernel and is either running (executing instructions) or waiting to run.



# Program to process

```
/home/cis90/simben $cmd
```

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



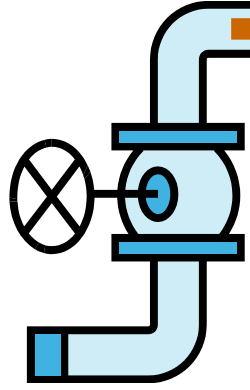
# Example program to process: sort command

```

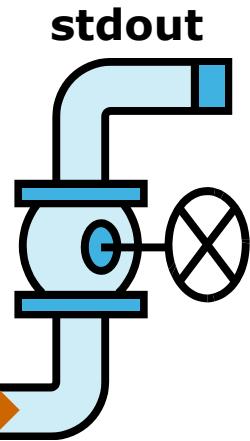
/home/cis90/simben $ sort
duke
benji
star
homer
benji
duke
homer
star
    
```



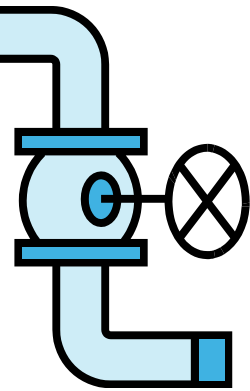
Options: NA  
Args: NA



**stdin**



**stdout**



**stderr**

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

/dev/pts/0



benji  
duke  
homer  
star

/dev/pts/0



duke  
benji  
star  
homer

**A simple example:**

```
CODE
void function1() {
    int A = 10;
    A += 66;
}

compiles to...
function1:
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

*Many programs are written in the C language*

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

## Example program to process: sort command

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

Use **type** to find where the sort program is located

```
[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 ~]$
```

Use **file** to see sort is a binary executable

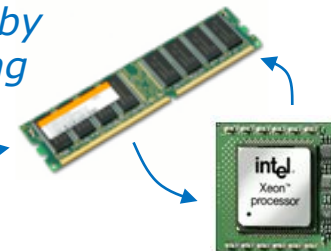
```
[rsimms@opus ~]$ xxd /bin/sort | more
```

```
00000000: 7f45 4c46 0101 0100 0000 0000 0000 0000  .ELF.....
00000010: 0200 0300 0100 0000 e093 0408 3400 0000  .....4...
00000020: 2cdb 0000 0000 0000 3400 2000 0800 2800  ,.....4. ...(.
00000030: 1f00 1e00 0600 0000 3400 0000 3480 0408  .....4...4...
00000040: 3480 0408 0001 0000 0001 0000 0500 0000  4.....
00000050: 0400 0000 0300 0000 3401 0000 3481 0408  .....4...4...
00000060: 3481 0408 1300 0000 1300 0000 0400 0000  4.....
```

Use **xxd** to produce a hexadecimal dump of the sort file

< *snipped* >

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



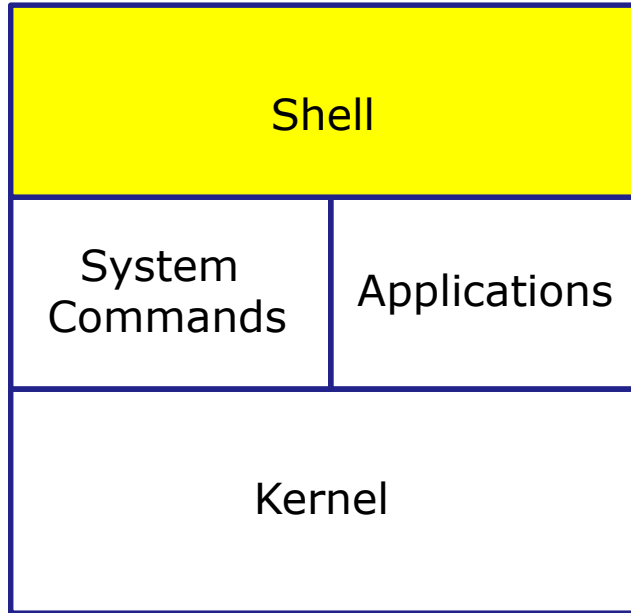


# Process Life Cycle





# The Shell **Execute** Step

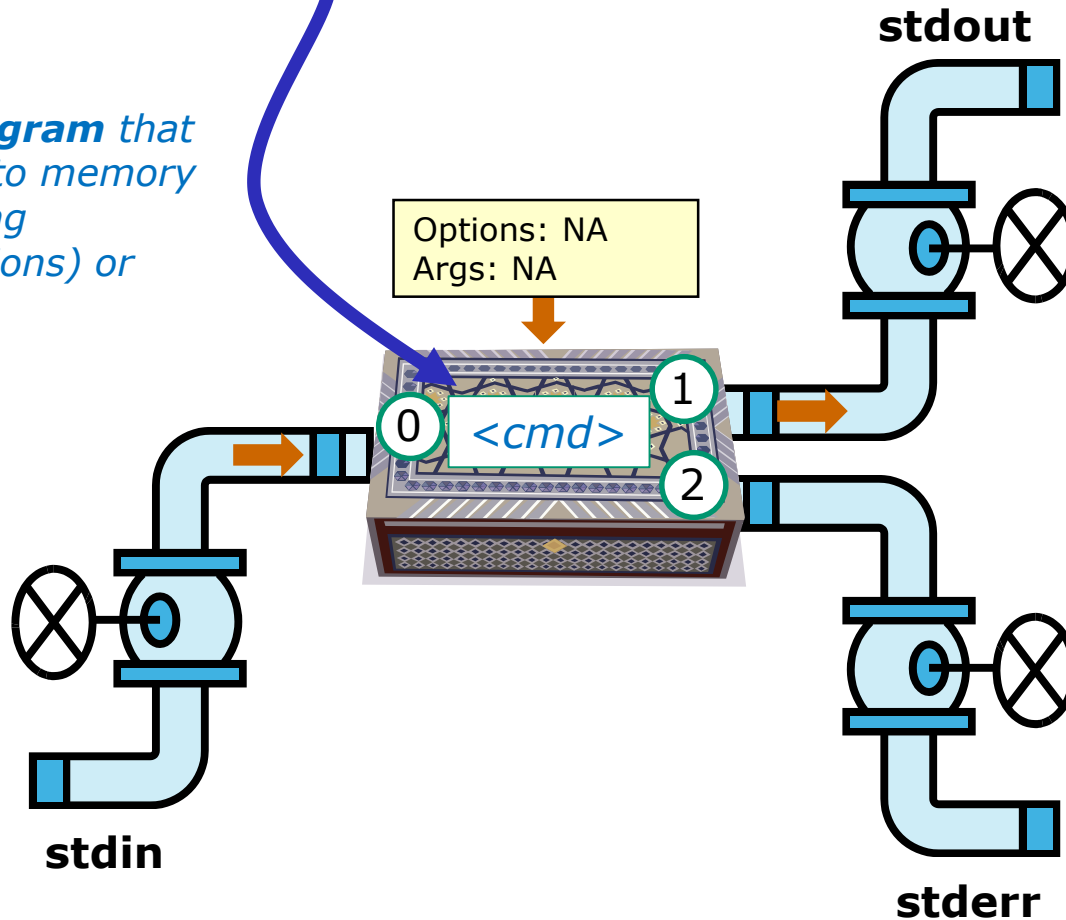


- 1) **Prompt** for a command
- 2) **Parse** (interpret metacharacters, expand file names and dissect command line into options and arguments)
- 3) **Search** for program (along the path)
- 4) **Execute** program by loading it into memory (as a process) and providing it with the parsed options/arguments. In addition hook up all inputs and outputs (stdin, stdout and stderr)
- 5) **Nap** (wait till process is done)
- 6) **Repeat**

## Executing a command `<cmd>`

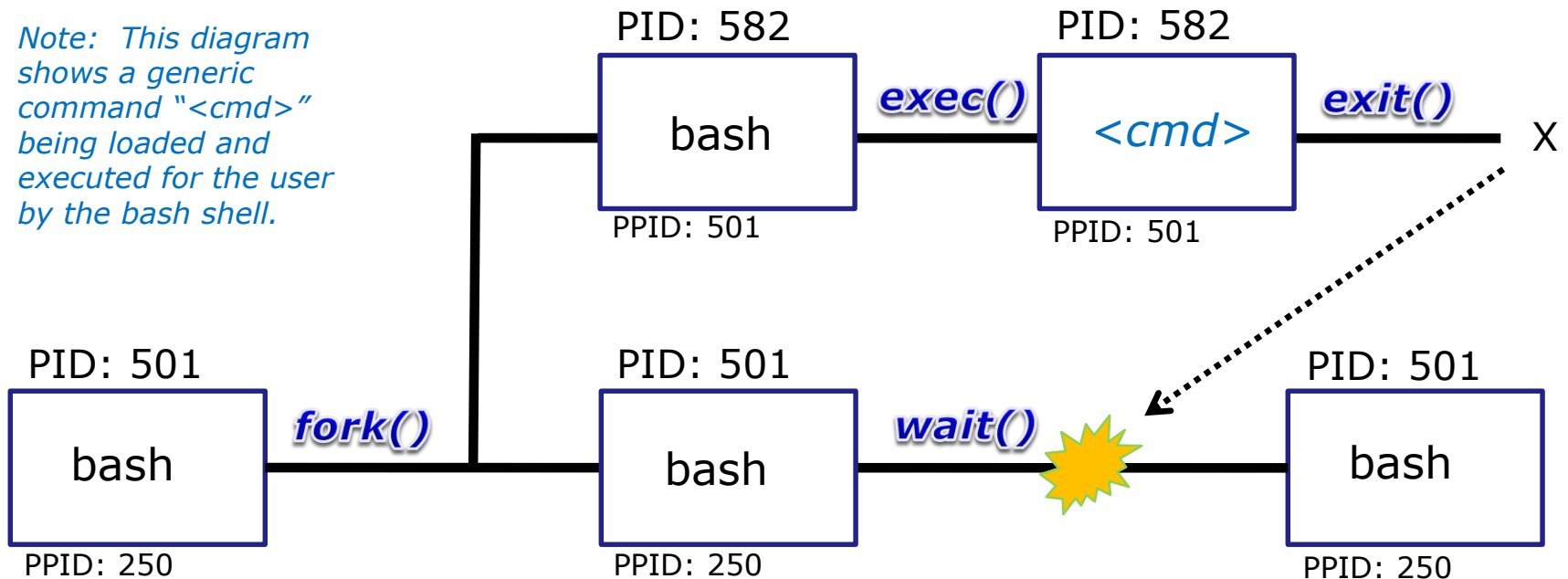
```
/home/cis90/simben $ <cmd>
```

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



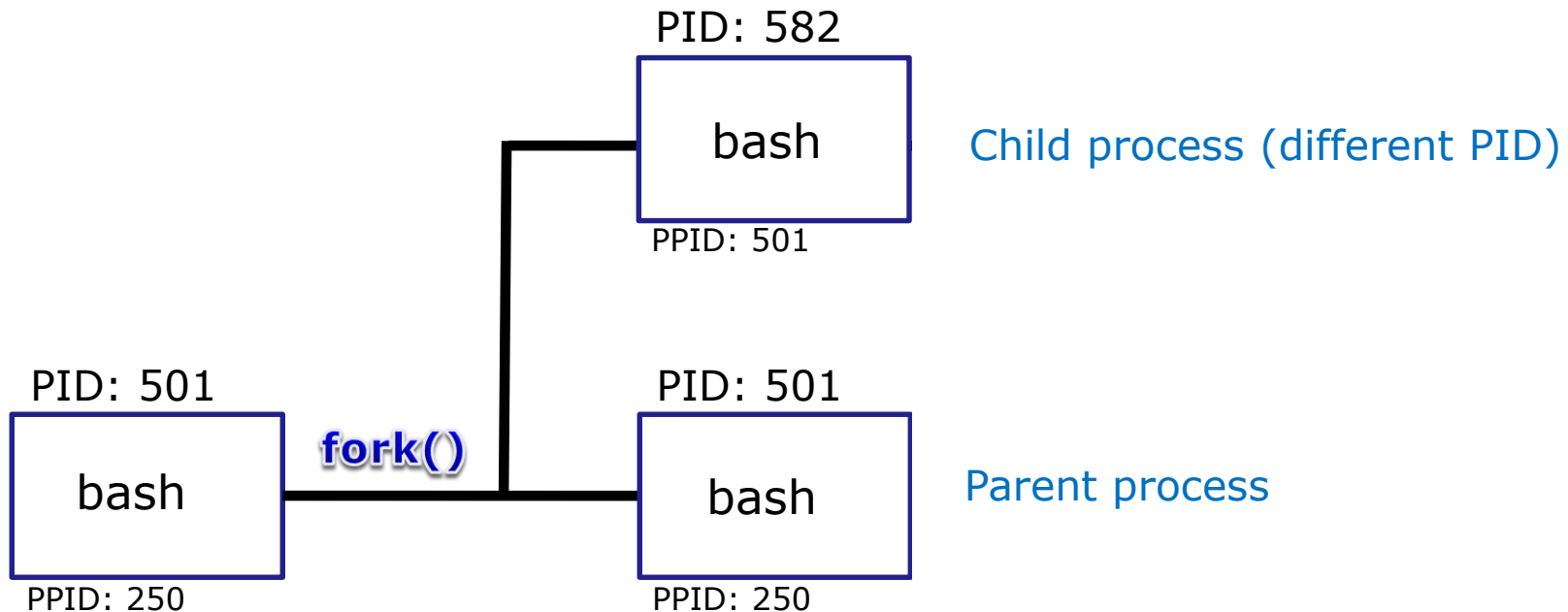
# Process Lifecycle

Note: This diagram shows a generic command "<cmd>" being loaded and executed for the user by the bash shell.



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

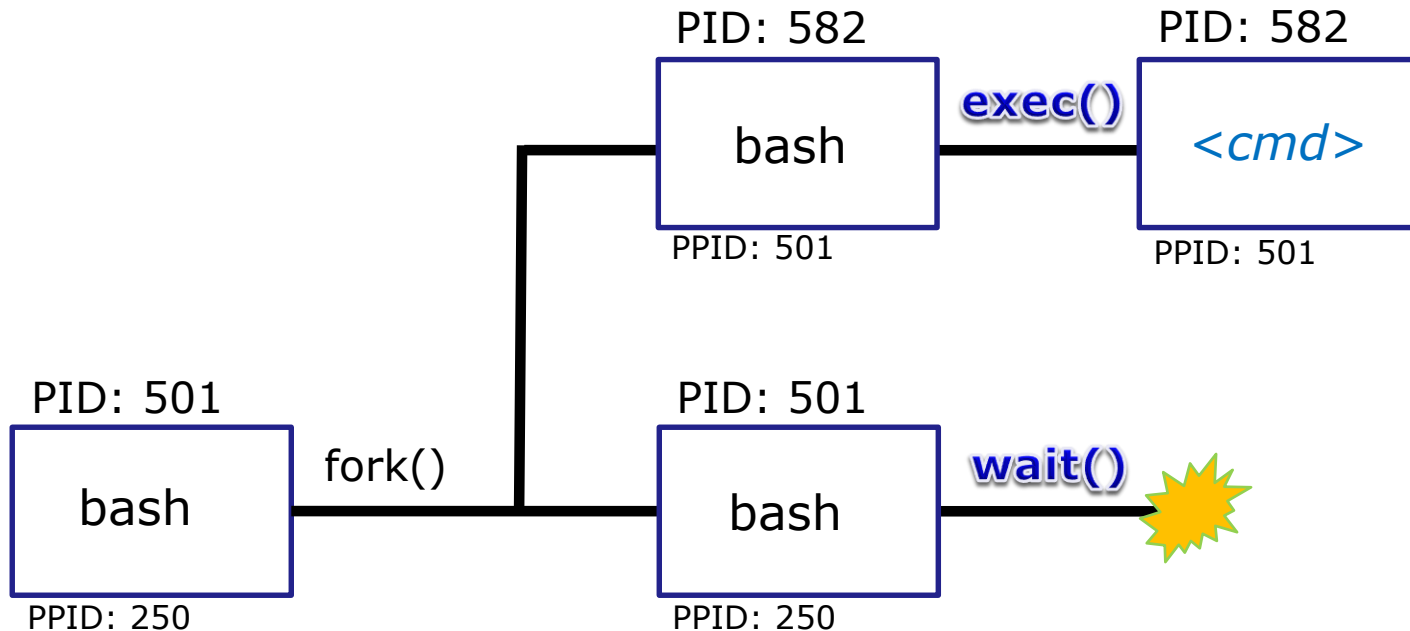
# Process Lifecycle - fork child process



1) The first step in executing a command is to create a new child process

- 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

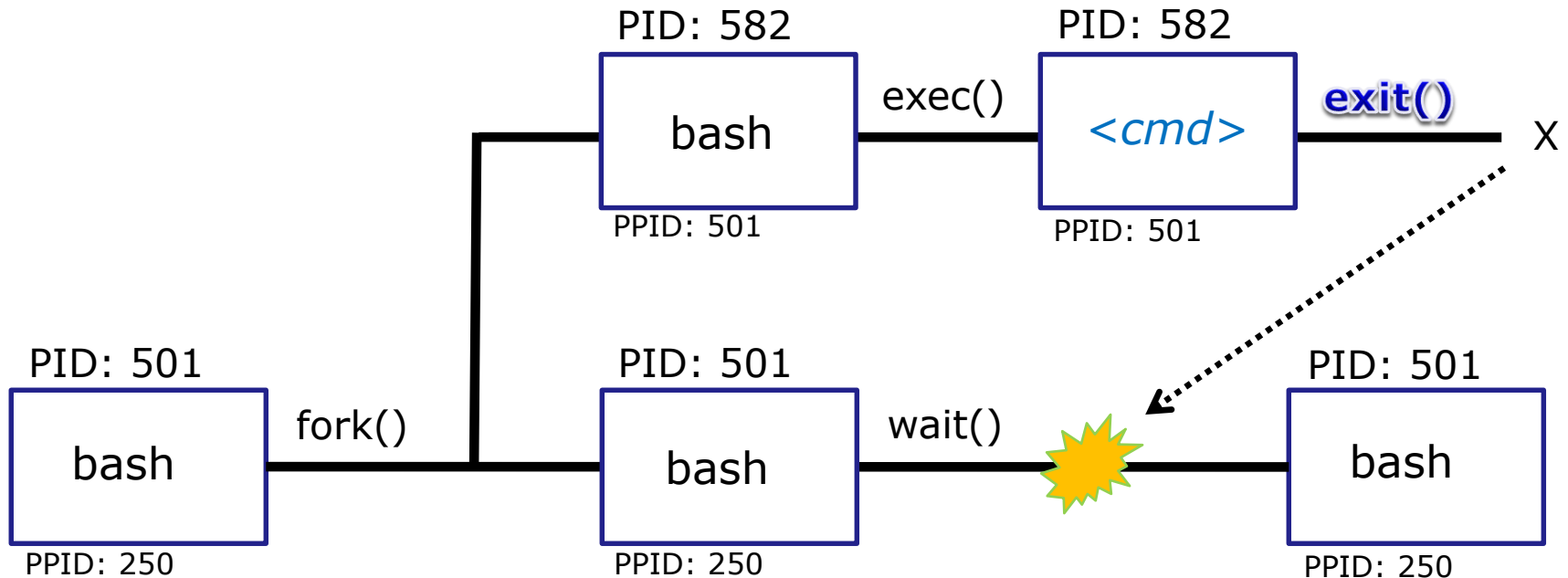


2) The next step is to load the command into the new child process

- 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.



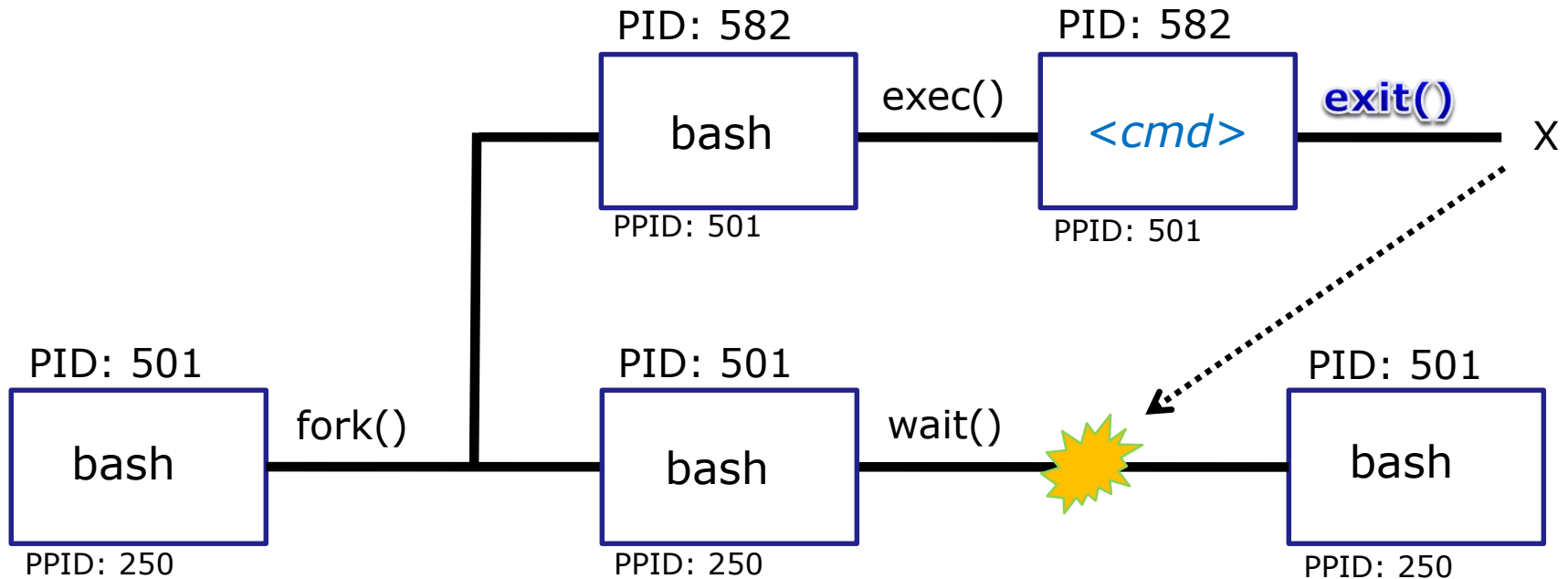
# Process Lifecycle



3) The final step is to terminate the new child process after it has finished

- 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. 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



*Note: 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 ps command



## Tools for your toolbox

**ps** - report a snapshot of the current processes

# ps command

## Basic syntax

(see man page for the rest of the story)

**ps** *<options>*

## Examples

**ps** *(shows your shell and ps processes in current session)*

**ps -a** *(show all processes you are running on all sessions)*

**ps -u simben90** *(shows sshd, shell and current processes all login sessions)*

**ps -l** *(shows your shell and ps processes using long format)*

**ps -ef** *(shows every process on system using full format)*



Column Header	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 (state) of the process: S=Sleeping, R=Running, T=Stopped, Z=Zombie, D=uninterruptable sleep (usually IO)
PRI	Process priority
SZ	Process size in pages
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

## Column headers on ps command output

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

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



# ps command

```
/home/cis90/simben $ ps
  PID TTY          TIME CMD
 11424 pts/3        00:00:00 bash
 12006 pts/3        00:00:00 ps
```

*With no options it shows my shell and ps processes for the terminal device I'm using*



man ps

- PID Process Identification Number, a unique number identifying the process
- TTY The terminal that the process's stdin and stdout are connected to
- CMD The name of the process (the command being run)
- TIME Cumulative CPU time

# ps command with -a option

```

/home/cis90/simben $ ps -a
  PID TTY          TIME CMD
12098 pts/2    00:00:00 man
12101 pts/2    00:00:00 sh
12102 pts/2    00:00:00 sh
12106 pts/2    00:00:00 less
12139 pts/3    00:00:00 ps
/home/cis90/simben $
  
```

*The -a option shows all processes being run by all users (does not include shell or sshd processes)*

/dev/pts/3

```

simben90@oslab:~/
/home/cis90/simben $ ps -a
  PID TTY          TIME CMD
12098 pts/2    00:00:00 man
12101 pts/2    00:00:00 sh
12102 pts/2    00:00:00 sh
12106 pts/2    00:00:00 less
12139 pts/3    00:00:00 ps
/home/cis90/simben $
  
```

/dev/pts/2

```

PS (1)                                Linux User's Manual                                PS (1)
NAME
  ps - report a snapshot of the current processes.

SYNOPSIS
  ps [options]

DESCRIPTION
  ps displays information about a selection of the active processes. If
  you want a repetitive update of the selection and the displayed
  information, use top(1) instead.

  This version of ps accepts several kinds of options:
  1  UNIX options, which may be grouped and must be preceded by a dash.
  2  BSD options, which may be grouped and must not be used with a dash.
  3  GNU long options, which are preceded by two dashes.

  Options of different types may be freely mixed, but conflicts can
  appear. There are some synonymous options, which are functionally
  identical, due to the many standards and ps implementations that this
  ps is compatible with.
  
```

man ps

- PID     Process Identification Number, a unique number identifying the process
- TTY     The terminal that the process's stdin and stdout are connected to
- CMD     The name of the process (the command being run)
- TIME    Cumulative CPU time

# ps command with -u option

```
/home/cis90/simben $ ps -u simben90
```

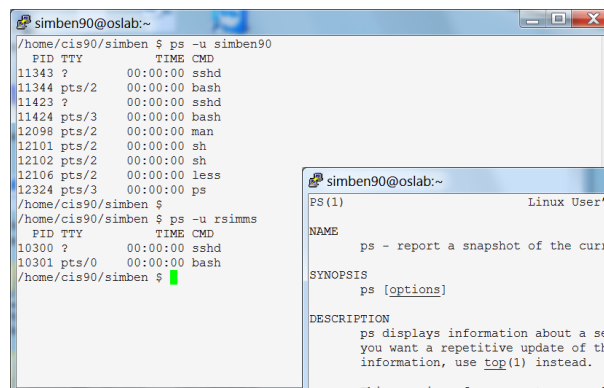
```

PID TTY          TIME CMD
11343 ?            00:00:00 sshd
11344 pts/2        00:00:00 bash
11423 ?            00:00:00 sshd
11424 pts/3        00:00:00 bash
12098 pts/2        00:00:00 man
12101 pts/2        00:00:00 sh
12102 pts/2        00:00:00 sh
12106 pts/2        00:00:00 less
12324 pts/3        00:00:00 ps

```

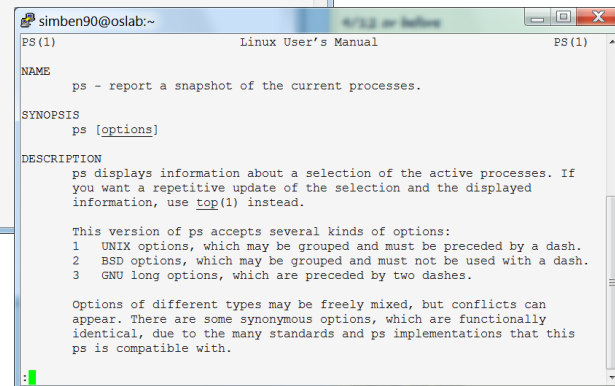
```
/home/cis90/simben $
```

*Use the -u (user) option to look at processes owned by a specific user (includes shell and sshd processes)*



/dev/pts/3

/dev/pts/2



man ps

- PID Process Identification Number, a unique number identifying the process
- TTY The terminal that the process's stdin and stdout are connected to
- CMD The name of the process (the command being run)
- TIME Cumulative CPU time

# ps command with -l option

Use **-l** (long format) to show additional process information

*11424 is sleeping*  
*12438 is running*

```

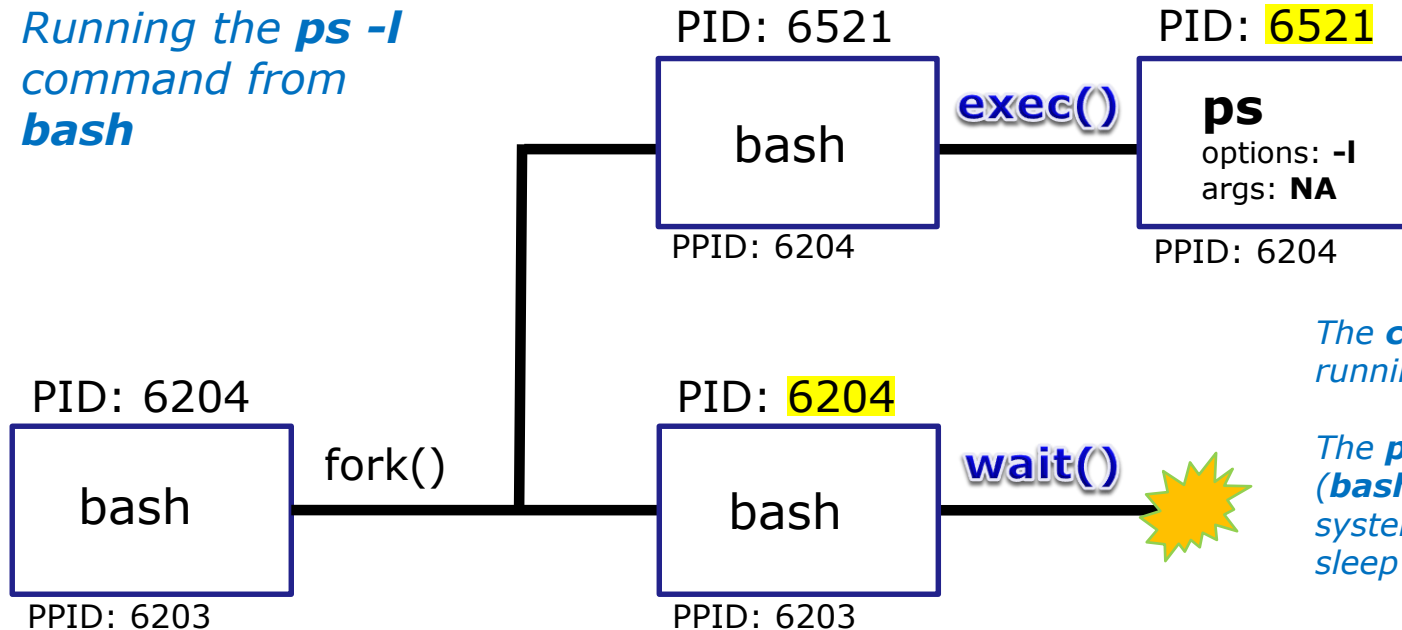
/home/cis90/simben $ ps -l
F S  UID  PID  PPID  C  PRI  NI ADDR SZ  WCHAN  TTY          TIME CMD
0 S  1201 11424 11423  0  80   0 - 1315 -      pts/3      00:00:00 bash
0 R  1201 12438 11424  0  80   0 - 1220 -      pts/3      00:00:00 ps
    
```



- UID      The user running the process
- S        The status of the process: S=Sleeping, R=Running, T=Stopped, Z=Zombie, D=uninterruptable sleep (usually IO)
- PRI      Process priority
- 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

# Deep Dive View of **ps -l** command

Running the **ps -l** command from **bash**



The **child** process (**ps**) is running (status=R)

The **parent** process (**bash**) issues the **wait** system call and goes to sleep (status=S)

6204 is sleeping  
6521 is running

```
[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

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

## ps command with **-ef** options (page 1)

```
/home/cis90/simben $ ps -ef
UID          PID    PPID  C  STIME TTY          TIME CMD
root          1      0   0  Aug27 ?           00:00:36 /sbin/init
root          2      0   0  Aug27 ?           00:00:00 [kthreadd]
root          3      2   0  Aug27 ?           00:00:14 [migration/0]
root          4      2   0  Aug27 ?           00:00:04 [ksoftirqd/0]
root          5      2   0  Aug27 ?           00:00:00 [migration/0]
root          6      2   0  Aug27 ?           00:00:35 [watchdog/0]
root          7      2   0  Aug27 ?           00:00:10 [migration/1]
root          8      2   0  Aug27 ?           00:00:00 [migration/1]
root          9      2   0  Aug27 ?           00:00:18 [ksoftirqd/1]
root         10      2   0  Aug27 ?           00:00:30 [watchdog/1]
root         11      2   0  Aug27 ?           00:00:10 [migration/2]
root         12      2   0  Aug27 ?           00:00:00 [migration/2]
root         13      2   0  Aug27 ?           00:00:07 [ksoftirqd/2]
root         14      2   0  Aug27 ?           00:00:30 [watchdog/2]
root         15      2   0  Aug27 ?           00:00:12 [migration/3]
root         16      2   0  Aug27 ?           00:00:00 [migration/3]
root         17      2   0  Aug27 ?           00:00:10 [ksoftirqd/3]
root         18      2   0  Aug27 ?           00:00:30 [watchdog/3]
root         19      2   0  Aug27 ?           00:03:37 [events/0]
root         20      2   0  Aug27 ?           00:04:37 [events/1]
root         21      2   0  Aug27 ?           00:03:50 [events/2]
root         22      2   0  Aug27 ?           00:04:42 [events/3]
root         23      2   0  Aug27 ?           00:00:00 [cgroup]
root         24      2   0  Aug27 ?           00:00:00 [khelper]
```

*Use **-ef** option to see everything with full format*



## ps command with -ef options (page 2)

```

root      25      2    0 Aug27 ?           00:00:00 [netns]
root      26      2    0 Aug27 ?           00:00:00 [async/mgr]
root      27      2    0 Aug27 ?           00:00:00 [pm]
root      28      2    0 Aug27 ?           00:00:28 [sync_supers]
root      29      2    0 Aug27 ?           00:00:31 [bdi-default]
root      30      2    0 Aug27 ?           00:00:00 [kintegrityd/0]
root      31      2    0 Aug27 ?           00:00:00 [kintegrityd/1]
root      32      2    0 Aug27 ?           00:00:00 [kintegrityd/2]
root      33      2    0 Aug27 ?           00:00:00 [kintegrityd/3]
root      34      2    0 Aug27 ?           00:01:18 [kblockd/0]
root      35      2    0 Aug27 ?           00:00:17 [kblockd/1]
root      36      2    0 Aug27 ?           00:00:22 [kblockd/2]
root      37      2    0 Aug27 ?           00:00:33 [kblockd/3]
root      38      2    0 Aug27 ?           00:00:00 [kacpid]
root      39      2    0 Aug27 ?           00:00:00 [kacpi_notify]
root      40      2    0 Aug27 ?           00:00:00 [kacpi_hotplug]
root      41      2    0 Aug27 ?           00:00:00 [ata_aux]
root      42      2    0 Aug27 ?           00:00:00 [ata_sff/0]
root      43      2    0 Aug27 ?           00:00:00 [ata_sff/1]
root      44      2    0 Aug27 ?           00:00:00 [ata_sff/2]
root      45      2    0 Aug27 ?           00:00:00 [ata_sff/3]
root      46      2    0 Aug27 ?           00:00:00 [ksuspend_usbd]
root      47      2    0 Aug27 ?           00:00:00 [khubd]
root      48      2    0 Aug27 ?           00:00:00 [kseriod]
root      49      2    0 Aug27 ?           00:00:00 [md/0]
root      50      2    0 Aug27 ?           00:00:00 [md/1]
root      51      2    0 Aug27 ?           00:00:00 [md/2]
root      52      2    0 Aug27 ?           00:00:00 [md/3]

```

## ps command with -ef options (page 3)

```

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

```

## ps command with -ef options (page 4)

```

root      53      2   0  Aug27 ?           00:00:00 [md_misc/0]
root      54      2   0  Aug27 ?           00:00:00 [md_misc/1]
root      55      2   0  Aug27 ?           00:00:00 [md_misc/2]
root      56      2   0  Aug27 ?           00:00:00 [md_misc/3]
root      57      2   0  Aug27 ?           00:00:00 [linkwatch]
root      58      2   0  Aug27 ?           00:00:02 [khungtaskd]
root      59      2   0  Aug27 ?           00:00:03 [kswapd0]
root      60      2   0  Aug27 ?           00:00:00 [ksmd]
root      61      2   0  Aug27 ?           00:00:00 [aio/0]
root      62      2   0  Aug27 ?           00:00:00 [aio/1]
root      63      2   0  Aug27 ?           00:00:00 [aio/2]
root      64      2   0  Aug27 ?           00:00:00 [aio/3]
root      65      2   0  Aug27 ?           00:00:00 [crypto/0]
root      66      2   0  Aug27 ?           00:00:00 [crypto/1]
root      67      2   0  Aug27 ?           00:00:00 [crypto/2]
root      68      2   0  Aug27 ?           00:00:00 [crypto/3]
root      73      2   0  Aug27 ?           00:00:00 [kthrotld/0]
root      74      2   0  Aug27 ?           00:00:00 [kthrotld/1]
root      75      2   0  Aug27 ?           00:00:00 [kthrotld/2]
root      76      2   0  Aug27 ?           00:00:00 [kthrotld/3]
root      77      2   0  Aug27 ?           00:00:00 [pciehp]
root      79      2   0  Aug27 ?           00:00:00 [kpsmoused]
root      80      2   0  Aug27 ?           00:00:00 [usbhid_resumer]
root     110      2   0  Aug27 ?           00:00:00 [kstriped]
root     194      2   0  Aug27 ?           00:00:00 [scsi_eh_0]
root     195      2   0  Aug27 ?           00:00:00 [scsi_eh_1]
root     209      2   0  Aug27 ?           00:00:00 [scsi_eh_2]

```

## ps command with -ef options (page 5)

```

root      210      2   0 Aug27 ?           00:00:00 [vmw_pvscsi_wq_2]
root      321      2   0 Aug27 ?           00:00:19 [jbd2/sda1-8]
root      322      2   0 Aug27 ?           00:00:00 [ext4-dio-unwrit]
root      414      1   0 Aug27 ?           00:00:00 /sbin/udevd -d
root      530      2   0 Aug27 ?           00:02:17 [vmmemctl]
root      776      2   0 Aug27 ?           00:00:29 [jbd2/sda5-8]
root      777      2   0 Aug27 ?           00:00:00 [ext4-dio-unwrit]
root      778      2   0 Aug27 ?           00:05:28 [jbd2/sda3-8]
root      779      2   0 Aug27 ?           00:00:00 [ext4-dio-unwrit]
root      822      2   0 Aug27 ?           00:00:43 [kauditd]
root     1457      1   0 Aug27 ?           00:02:13 auditd
root     1475      1   0 Aug27 ?           00:00:00 /sbin/portreserve
root     1482      1   0 Aug27 ?           00:00:45 /sbin/rsyslogd -i /var/run/syslo
root     1511      1   0 Aug27 ?           00:28:03 irqbalance --pid=/var/run/irqbal
rpc      1525      1   0 Aug27 ?           00:00:09 rpcbind
rpcuser  1543      1   0 Aug27 ?           00:00:00 rpc.statd
root     1555      1   0 Aug27 ?           00:00:12 mdadm --monitor --scan -f --pid-
dbus     1681      1   0 Aug27 ?           00:00:07 dbus-daemon --system
root     1698      1   0 Aug27 ?           00:00:42 cupsd -C /etc/cups/cupsd.conf
root     1723      1   0 Aug27 ?           00:00:00 /usr/sbin/acpid
68       1732      1   0 Aug27 ?           00:00:42 hald
root     1733     1732   0 Aug27 ?           00:00:00 hald-runner
root     1765     1733   0 Aug27 ?           00:00:00 hald-addon-input: Listening on /
68       1773     1733   0 Aug27 ?           00:00:00 hald-addon-acpi: listening on ac
root     1800      1   0 Aug27 ?           00:02:50 automount --pid-file /var/run/au
root     1863      1   0 Aug27 ?           00:00:00 /bin/sh /usr/bin/mysqld_safe --d
mysql    1965     1863   0 Aug27 ?           01:42:39 /usr/libexec/mysqld --basedir=/u

```

## ps command with -ef options (page 6)

```

root      1997      1  0 Aug27 ?          00:03:33 sendmail: accepting connections
smmsp    2006      1  0 Aug27 ?          00:00:01 sendmail: Queue runner@01:00:00
root     2028      1  0 Aug27 ?          00:00:00 abrt-dump-oops -d /var/spool/abr
root     2036      1  0 Aug27 ?          00:04:06 /usr/sbin/httpd
root     2044      1  0 Aug27 ?          00:02:17 crond
root     2055      1  0 Aug27 ?          00:00:02 /usr/sbin/atd
root     2076      1  0 Aug27 tty1        00:00:00 /sbin/mingetty /dev/tty1
root     2078      1  0 Aug27 tty2        00:00:00 /sbin/mingetty /dev/tty2
root     2080      1  0 Aug27 tty3        00:00:00 /sbin/mingetty /dev/tty3
root     2082      1  0 Aug27 tty4        00:00:00 /sbin/mingetty /dev/tty4
root     2088      1  0 Aug27 tty5        00:00:00 /sbin/mingetty /dev/tty5
root     2090      1  0 Aug27 tty6        00:00:00 /sbin/mingetty /dev/tty6
apache   3716    2036  0 Nov02 ?          00:01:22 /usr/sbin/httpd
apache   5550    2036  0 Nov02 ?          00:01:15 /usr/sbin/httpd
apache   5551    2036  0 Nov02 ?          00:01:20 /usr/sbin/httpd
apache   5552    2036  0 Nov02 ?          00:01:17 /usr/sbin/httpd
apache   5554    2036  0 Nov02 ?          00:01:16 /usr/sbin/httpd
apache   6611    2036  0 Nov02 ?          00:01:18 /usr/sbin/httpd
root    10295  18067  0 07:28 ?          00:00:00 sshd: rsimms [priv]
rsimms  10300  10295  0 07:28 ?          00:00:00 sshd: rsimms@pts/0
rsimms  10301  10300  0 07:28 pts/0      00:00:00 -bash
apache  10326    2036  0 Nov02 ?          00:01:07 /usr/sbin/httpd
root    11088  18067  0 08:06 ?          00:00:00 sshd: lamnav90 [priv]
lamnav90 11092  11088  0 08:06 ?          00:00:01 sshd: lamnav90@pts/1
lamnav90 11093  11092  0 08:06 pts/1      00:00:00 -bash
root    11336  18067  0 08:12 ?          00:00:00 sshd: simben90 [priv]
simben90 11343  11336  0 08:12 ?          00:00:00 sshd: simben90@pts/2
simben90 11344  11343  0 08:12 pts/2      00:00:00 -bash

```

## ps command with -ef options (page 6)

```

root      11415 18067  0 08:13 ?          00:00:00 sshd: simben90 [priv]
simben90  11423 11415  0 08:13 ?          00:00:00 sshd: simben90@pts/3
simben90  11424 11423  0 08:13 pts/3    00:00:00 -bash
root      11767     2  0 Sep17 ?          00:00:00 [rpciod/0]
root      11768     2  0 Sep17 ?          00:00:00 [rpciod/1]
root      11769     2  0 Sep17 ?          00:00:00 [rpciod/2]
root      11770     2  0 Sep17 ?          00:00:00 [rpciod/3]
root      11772     2  0 Sep17 ?          00:00:00 [kslowd000]
root      11773     2  0 Sep17 ?          00:00:00 [kslowd001]
root      11774     2  0 Sep17 ?          00:00:00 [nfsiod]
lamnav90  12591 11093  0 08:57 pts/1    00:00:00 ssh sun-hwa-p2
root      12613     2  0 Sep08 ?          00:05:57 [flush-8:0]
simben90  12684 11344  0 08:59 pts/2    00:00:00 ssh sun-hwa-p2
root      12824 18067  0 09:05 ?          00:00:00 sshd: smimat90 [priv]
smimat90  12845 12824  0 09:06 ?          00:00:00 sshd: smimat90@pts/4
smimat90  12846 12845  0 09:06 pts/4    00:00:00 -bash
root      12875 18067  0 09:06 ?          00:00:00 sshd: pikann90 [priv]
pikann90  12879 12875  0 09:06 ?          00:00:00 sshd: pikann90@pts/5
pikann90  12880 12879  0 09:06 pts/5    00:00:00 -bash
root      12906 18067  0 09:06 ?          00:00:00 sshd: pikann90 [priv]
pikann90  12925 12906  0 09:07 ?          00:00:00 sshd: pikann90@pts/6
pikann90  12926 12925  0 09:07 pts/6    00:00:00 -bash
pikann90  12957 12926  0 09:07 pts/6    00:00:00 ssh sun-hwa-p2
root      13008 18067  0 09:09 ?          00:00:00 sshd: smimat90 [priv]
smimat90  13013 13008  0 09:10 ?          00:00:00 sshd: smimat90@pts/7
smimat90  13014 13013  0 09:10 pts/7    00:00:00 -bash
root      13330 18067  0 09:20 ?          00:00:00 sshd: quifra90 [priv]

```



## ps command with -ef options (page 7)

```

quifra90 13355 13330 0 09:21 ? 00:00:00 sshd: quifra90@pts/8
quifra90 13356 13355 0 09:21 pts/8 00:00:00 -bash
apache 13456 2036 0 09:24 ? 00:00:00 /usr/sbin/httpd
apache 13458 2036 0 09:24 ? 00:00:00 /usr/sbin/httpd
apache 13459 2036 0 09:24 ? 00:00:00 /usr/sbin/httpd
smimat90 13548 13014 0 09:28 pts/7 00:00:00 man grep
smimat90 13551 13548 0 09:28 pts/7 00:00:00 sh -c (cd "/usr/share/man" && (e
smimat90 13552 13551 0 09:28 pts/7 00:00:00 sh -c (cd "/usr/share/man" && (e
smimat90 13557 13552 0 09:28 pts/7 00:00:00 /usr/bin/less -is
simben90 13640 11424 0 09:30 pts/3 00:00:00 ps -ef
tinsam90 14869 1 0 Sep09 ? 00:00:00 SCREEN
tinsam90 14870 14869 0 Sep09 pts/20 00:00:00 /bin/bash
tinsam90 14886 14869 0 Sep09 pts/21 00:00:00 /bin/bash
tinsam90 14932 14869 0 Sep09 pts/23 00:00:00 /bin/bash
root 15152 414 0 Sep30 ? 00:00:00 /sbin/udevd -d
root 15153 414 0 Sep30 ? 00:00:00 /sbin/udevd -d
root 18067 1 0 Sep25 ? 00:00:04 /usr/sbin/sshd
root 18962 2 0 Sep09 ? 00:00:00 [bluetooth]
ntp 25613 1 0 Sep29 ? 00:00:16 ntpd -u ntp:ntp -p /var/run/ntpd
apache 32671 2036 0 Nov02 ? 00:01:37 /usr/sbin/httpd
apache 32674 2036 0 Nov02 ? 00:01:34 /usr/sbin/httpd
apache 32675 2036 0 Nov02 ? 00:01:35 /usr/sbin/httpd
apache 32676 2036 0 Nov02 ? 00:01:34 /usr/sbin/httpd
apache 32677 2036 0 Nov02 ? 00:01:35 /usr/sbin/httpd
apache 32678 2036 0 Nov02 ? 00:01:33 /usr/sbin/httpd
apache 32679 2036 0 Nov02 ? 00:01:34 /usr/sbin/httpd
apache 32680 2036 0 Nov02 ? 00:01:36 /usr/sbin/httpd

```



# Job Control

```
find / -user simben90 2> /dev/null
```

*Some commands, like the one we used in Lab 7, take 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.

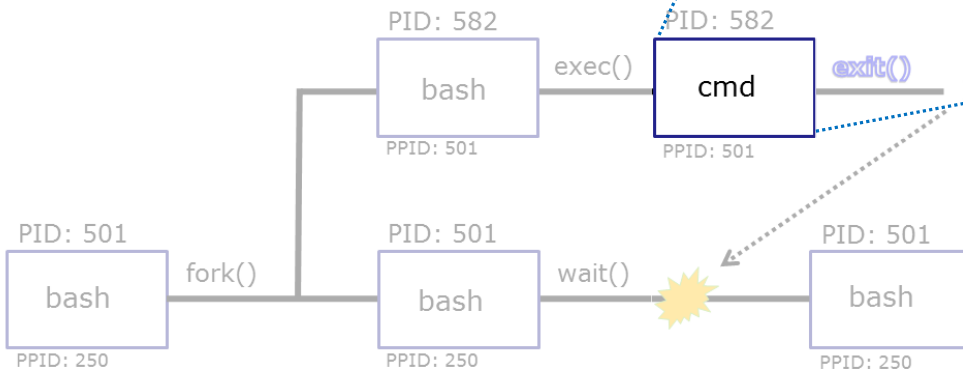
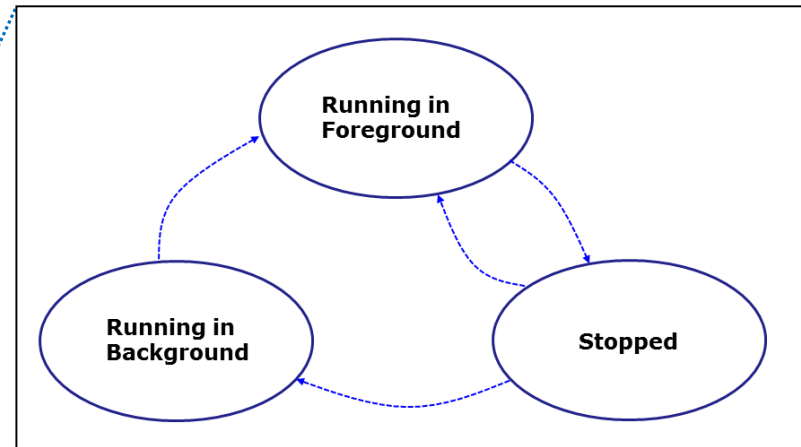
### Background Processes

- 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

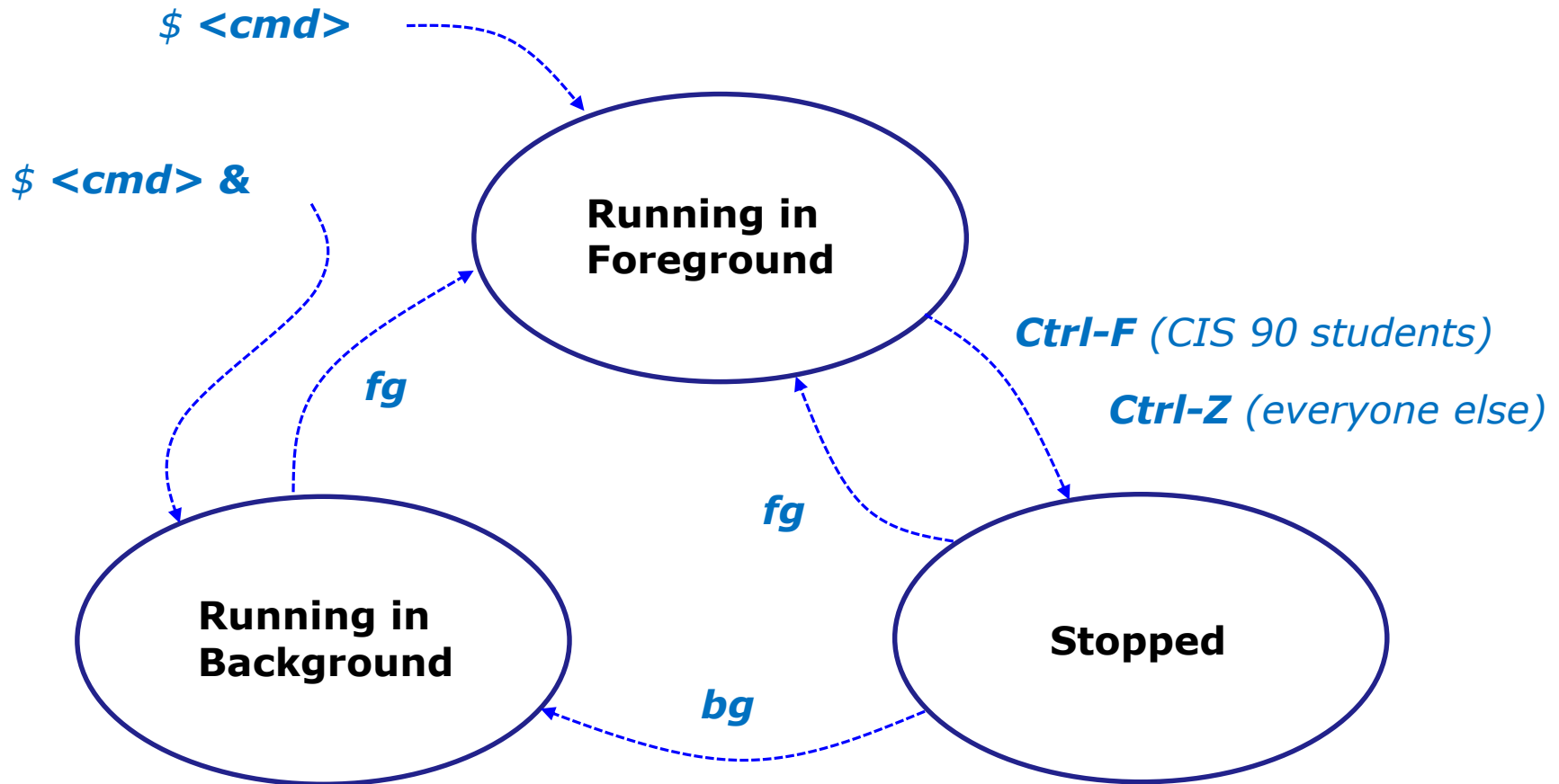
## A feature of the bash shell

When a process is **running** the user can **stop** it and choose whether it runs in the **background** or **foreground**



# Job Control

A feature of the bash shell



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



# Job Control

## Suspending and Resuming

### **Ctrl-F**

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

*Note, CIS 90 students will be using Ctrl-F which has been configured in their shell environment. Normally Ctrl-Z is used.*

### **bg**

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

# Job Control

## Keyboard customization for CIS 90

### Ctrl-Z or Ctrl-F

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

```
/home/cis90/simben $ 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 shell environment for the CIS 90 accounts was customized to use a different keystroke for sending a SIGTSTP signal*

## Example - suspending a **find** command

```
$ find / -name "stage[12]" 2> /dev/null
```

*Suspend a long find command, then resume it in the background*

**Running in Foreground**

*Ctrl-F (CIS 90 students)*

*Ctrl-Z (everyone else)*

**Running in Background**

**Stopped**

*bg*

## Example - suspending a **find** command

```

/home/cis90/simben $ find / -name "stage[12]" 2> /dev/null
/home/cis90/bownic/bin/stage1
/home/cis90/bownic/bin/stage2
/home/cis90/zemric/stage1
/home/cis90/zemric/stage2
/home/cis90/boyjef/bin/stage1
/home/cis90/boyjef/bin/stage2
/home/cis90/porrya/bin/stage1
/home/cis90/porrya/bin/stage2
/home/cis90/isoric/stage1
/home/cis90/isoric/stage2
^F
[1]+  Stopped                  find / -name "stage[12]" 2> /dev/null
/home/cis90/simben $
  
```

**Ctrl-F** (CIS 90 accounts) Or  
**Ctrl-Z** (other accounts) is  
tapped to suspend the  
find command

*Notice, we can type more commands again after  
the find command was stopped*

*In the same session we can monitor the find process*

*Process ID 25907  
(find) is stopped  
(status = T)*

```

/home/cis90/simben $ ps -l
F S  UID  PID  PPID  C  PRI  NI  ADDR  SZ  WCHAN  TTY  TIME  CMD
0 S  1201 11344 11343  0  80   0  -  1315  -  pts/2  00:00:00 bash
0 T  1201 25907 11344  4  80   0  -  1219  -  pts/2  00:00:00 find
0 R  1201 25925 11344  0  80   0  -  1219  -  pts/2  00:00:00 ps
/home/cis90/simben $
  
```

## Example - suspending a **find** command

```

/home/cis90/simben $ bg
[1]+ find / -name "stage[12]" 2> /dev/null &
/home/cis90/simben $ /usr/share/grub/i386-redhat/stage1
/usr/share/grub/i386-redhat/stage2
/boot/grub/stage1
/boot/grub/stage2

[1]+  Exit 1                  find / -name "stage[12]" 2> /dev/null
/home/cis90/simben $
    
```

*bg resumes the find command in the background*

*Notice, we can't type more commands again in this session until the find command finishes*

*In a different session we can monitor the find process*

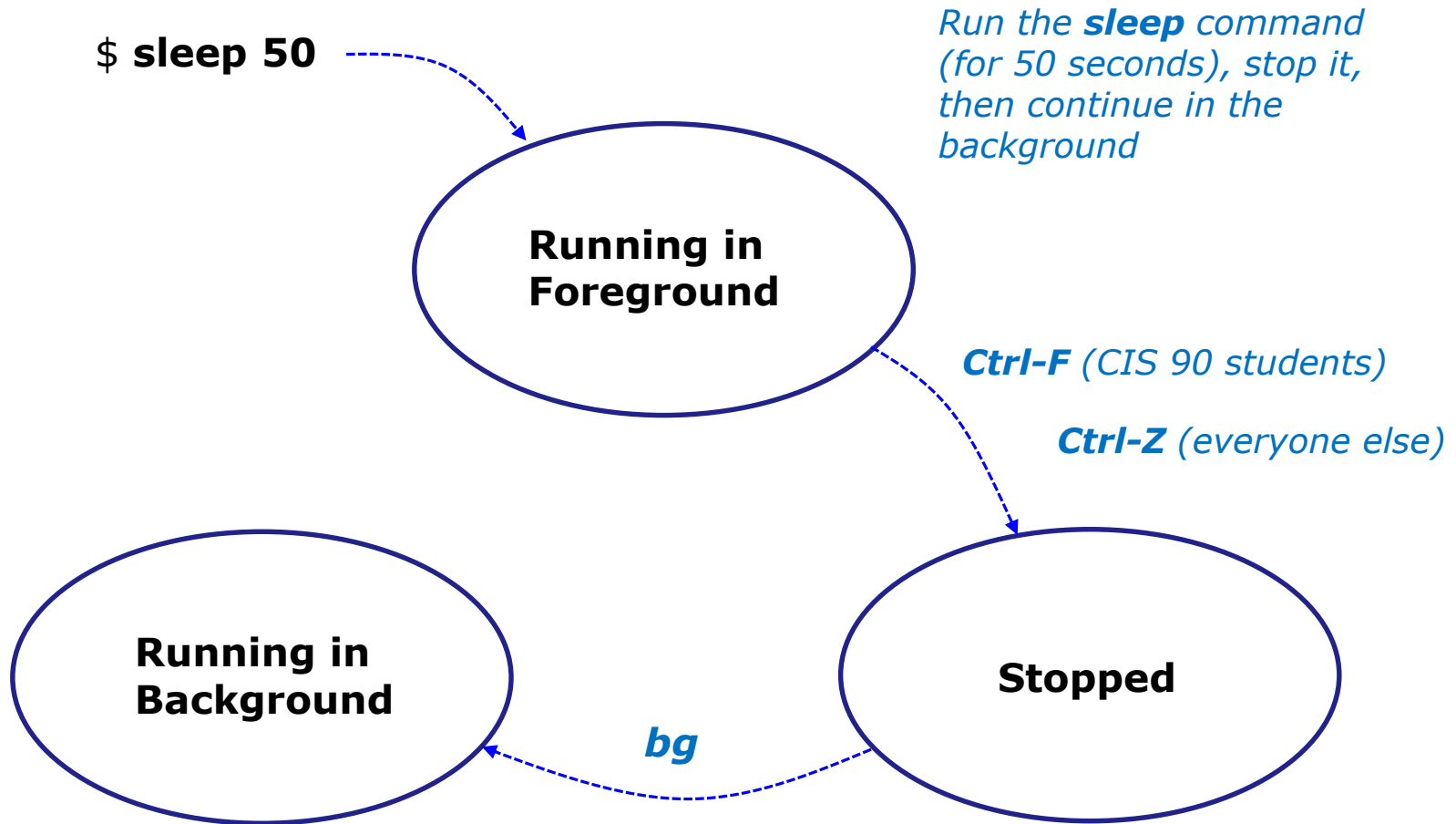
```

/home/cis90/simben $ ps -l -u simben90
F S  UID  PID  PPID  C  PRI  NI  ADDR  SZ  WCHAN  TTY          TIME CMD
5 S  1201 11343 11336  0  80   0  -   3010  ?      ?          00:00:01 sshd
0 S  1201 11344 11343  0  80   0  -   1315  -      pts/2      00:00:00 bash
5 R  1201 11423 11415  0  80   0  -   3200  ?      ?          00:00:01 sshd
0 S  1201 11424 11423  0  80   0  -   1315  -      pts/3      00:00:00 bash
0 R  1201 25907 11344  0  80   0  -   1186  -      pts/2      00:00:01 find
0 R  1201 25956 11424  0  80   0  -   1234  -      pts/3      00:00:00 ps
/home/cis90/simben $
    
```

*Process ID 25907  
(find) is running  
(status=R)*

## Job Control

Example - suspending a **sleep** command





# Job Control

## Example - suspending a **sleep** command

```
[rsimms@opus ~]$ sleep 50
[1]+  Stopped                  sleep 50
[rsimms@opus ~]$
```

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

```
[rsimms@opus ~]$ ps -l -u rsimms
```

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
5	S	201	25055	25044	0	75	0	-	2481	stext	?	00:00:00	sshd
0	S	201	25056	25055	0	76	0	-	1168	-	pts/3	00:00:00	bash
5	S	201	25087	25084	0	75	0	-	2481	stext	?	00:00:00	sshd
0	S	201	25088	25087	0	75	0	-	1168	wait	pts/4	00:00:00	bash
0	T	201	25389	25056	0	76	0	-	929	finish	pts/3	00:00:00	sleep
0	R	201	25391	25088	0	77	0	-	1065	-	pts/4	00:00:00	ps

**PID 25389**  
(sleep) is  
stopped

## Job Control

### Example - suspending a **sleep** command

```
[rsimms@opus ~]$ bg
[1]+ sleep 50 &
```

*bg resumes the sleep command and it finishes*

*PID 25389 is sleeping and no longer stopped (status=S)*

```
[rsimms@opus ~]$ ps -l -u rsimms
F S  UID  PID  PPID  C  PRI  NI  ADDR  SZ  WCHAN  TTY          TIME CMD
5 S   201 25055 25044  0  75   0  -   2481 stext  ?           00:00:00 sshd
0 S   201 25056 25055  0  75   0  -   1168 -      pts/3       00:00:00 bash
5 R   201 25087 25084  0  81   0  -   2481 stext  ?           00:00:00 sshd
0 S   201 25088 25087  0  75   0  -   1168 wait   pts/4       00:00:00 bash
0 S   201 25389 25056  0  75   0  -   929 322807 pts/3       00:00:00 sleep
0 R   201 25394 25088  0  77   0  -   1065 -      pts/4       00:00:00 ps
[rsimms@opus ~]$
```

# Job Control

## Additional Control Options

**&**

- Append to a command to run it in the background

**fg**

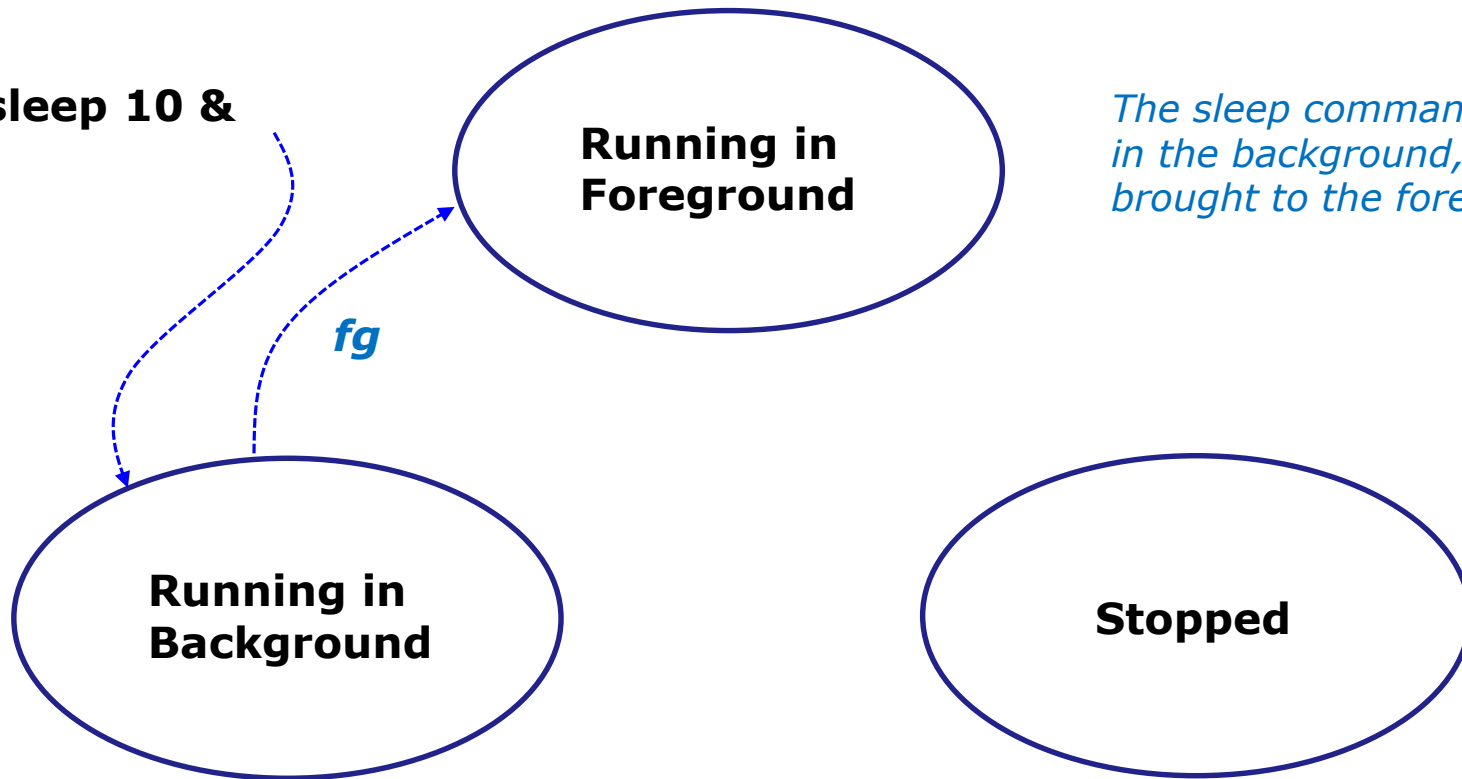
- Brings the most recent background process to the foreground

**jobs**

- Lists all background jobs

# Job Control Example

\$ **sleep 10 &**



*The sleep command is started in the background, then brought to the foreground*

## Job Control Example

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

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

```
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. This allows you to keep using the terminal for more commands while those applications run.*

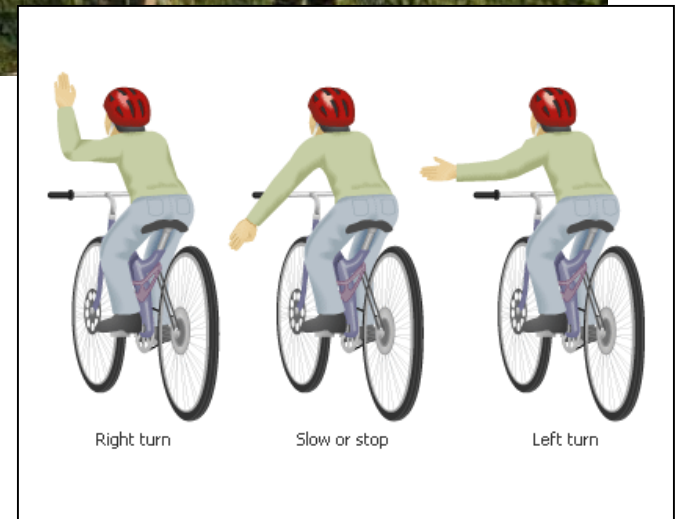
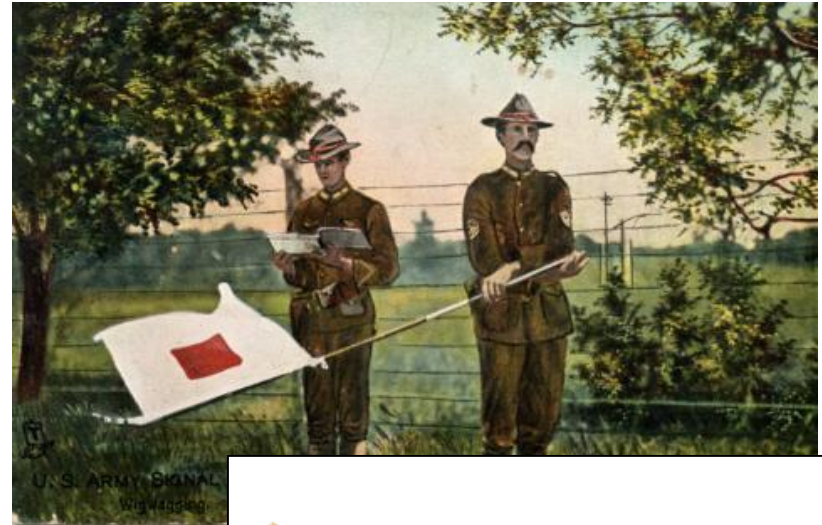
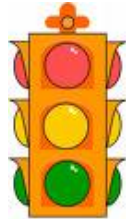


# Signals

# Signals

PLATE 4

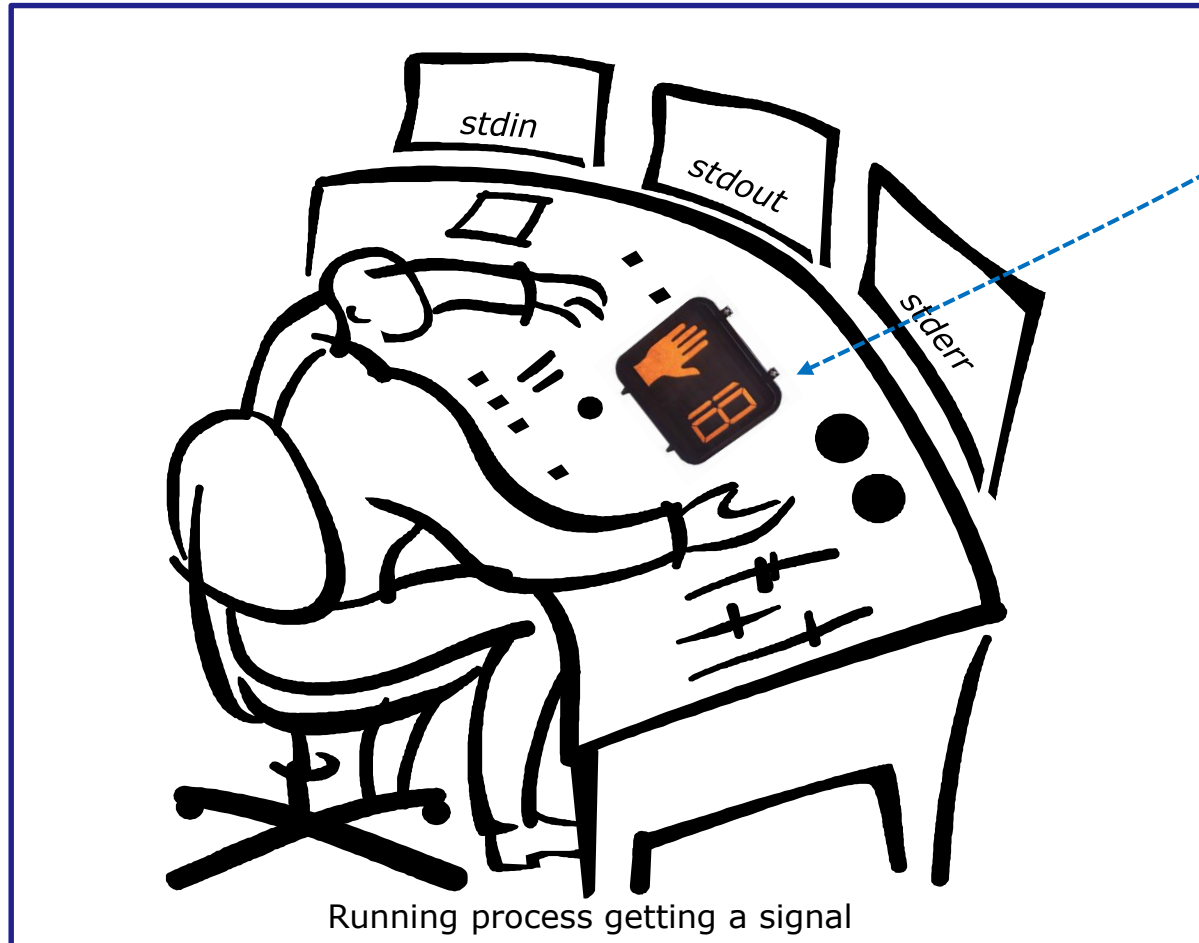
COMMERCIAL CODE SIGNALS		
<p>EXAMPLES OF THE SEVERAL HOISTS WHICH CAN BE MADE HAVING TWO, THREE, OR FOUR FLAGS. When a word contains two letters of the same name, the second time of its occurrence it must begin or be in the 2nd Hoist; and on its 3rd occurrence, it must begin or be in the 3rd Hoist.</p>		
<b>URGENT &amp; IMPORTANT SIGNALS</b>		<b>COMPASS SIGNALS 3 FLAGS</b>
<p>CODE FLAG OVER 1 FLAG OR 2 FLAG SIGNALS</p> <p>CODE FLAG P: "I Am about to Sail"</p> <p>CODE FLAG C: "Do Not" "abandon the Vessel"</p>		<p>A: N 1/2 E</p> <p>K: S 57° W</p>
<b>LATITUDE &amp; LONGITUDE SIGNALS</b>		<b>CODE FLAG OVER 2 FLAGS</b>
<p>CODE FLAG A: 12° Latitude</p> <p>CODE FLAG O: North Latitude</p>	<p>GENERAL SIGNAL Q: OR H</p> <p>GENERAL SIGNAL X: OR Y</p>	<p>CODE FLAG E: 23° Longitude</p> <p>GENERAL SIGNAL Z: East Longitude</p>
<b>NUMERAL TABLE</b>	<b>GENERAL VOCABULARY</b>	<b>GEOGRAPHICAL SIGNALS ALPHABETICAL ORDER</b>
<p>CODE FLAG UNDER 2 FLAGS Y: 10,000</p> <p>CODE FLAG S: Tons of Coal</p>	<p>3 FLAG SIGNAL I: Tons of Coal</p> <p>3 FLAG SIGNAL X: Tons of Coal</p> <p>3 FLAG SIGNAL K: Tons of Coal</p>	<p>4 FLAG SIGNAL A: Glasgow, Scotland.</p> <p>4 FLAG SIGNAL E: Glasgow, Scotland.</p> <p>4 FLAG SIGNAL Y: Glasgow, Scotland.</p> <p>4 FLAG SIGNAL Z: Glasgow, Scotland.</p>
<b>ALPHABETICAL SPELLING TABLE</b>		<b>NAMES OF VESSELS FROM CODE LIST</b>
<p>1, 2, 3 OR 4 FLAG SIGNALS</p> <p>J: John</p> <p>O: Abb</p> <p>H: Abb</p> <p>N: Abb</p>	<p>4 FLAG SIGNALS</p> <p>C: Abb</p> <p>B: Abb</p> <p>D: Abb</p> <p>N: Abb</p> <p>F: Abb</p> <p>P: Abb</p>	<p>4 FLAG SIGNAL H: Graysa of Glasgow</p> <p>4 FLAG SIGNAL C: Graysa of Glasgow</p> <p>4 FLAG SIGNAL L: Graysa of Glasgow</p> <p>4 FLAG SIGNAL B: Graysa of Glasgow</p>





# Signals

*Signals are asynchronous messages sent to processes*



*Asynchronous means it can happen at any time*

# Signals

## Signals are asynchronous messages sent to processes

They can result in one of three courses of action:

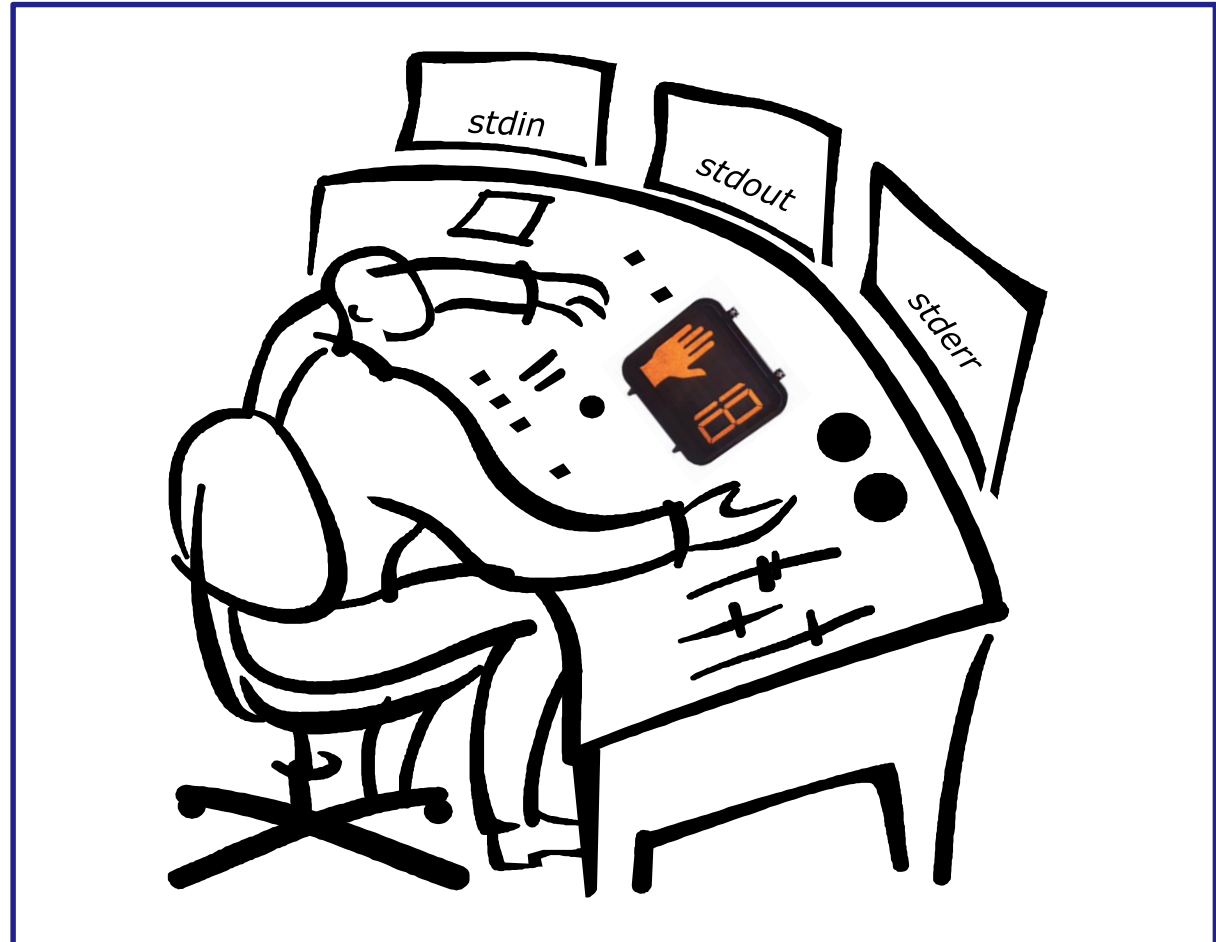
1. be ignored,
2. default action (die)
3. execute some predefined function.

Signals are sent:

- Using the **kill** command: `$ kill -# PID`
  - Where `#` is the signal number and PID is the process id.
  - if no signal number is specified, SIGTERM is sent.
- Using special **keystrokes** (e.g. Ctrl-Z for SIGTSTP/20)
  - limited to just a few signals
  - sent to the process running in the foreground

# Signals

*Signals are asynchronous messages sent to processes*



Running process gets a signal



## Tools for your toolbox



**kill** - send signal to process (by PID)



**killall** - send signal to process (by name)

# kill command

## Basic syntax

(see man page for the rest of the story)

**kill** *<signal>* *<PID>*

## Examples

```
kill -s sigquit 14151 (Send signal SIGQUIT/3 to process 14151)
```

```
kill -s 3 14151 (Send signal SIGQUIT/3 to process 14151)
```

```
kill -3 14151 (Send signal SIGQUIT/3 to process 14151)
```

```
kill -9 14151 (Send signal SIGKILL/9 to process 14151)
```

```
kill -l (list all signal numbers)
```

# killall command

## Basic syntax

(see man page for the rest of the story)

**killall** *<signal>* *<process>*

## Examples

**killall -s sigquit app** *(Send signal 3 to process named app)*

**killall -s 3 app** *(Send signal 3 to process named app)*

**killall -3 app** *(Send signal 3 to process named app)*

**killall -9 app** *(Send signal 9 to process named app)*

# Signals

*Use kill -l to see all signals*

```
/home/cis90/rodduk $ 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/rodduk $
```



# Signals

SIGHUP	1	Hangup (POSIX)
SIGINT	2	Terminal interrupt (ANSI) <b>Ctrl-C</b>
SIGQUIT	3	Terminal quit (POSIX) <b>Ctrl-\</b>
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 ( <b>can't be caught or ignored</b> ) (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) ( <b>default kill signal when not specified</b> )

# 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) <b>Ctrl-Z or Ctrl-F</b>
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)

# Signals

## Special keystrokes

```
/home/cis90/rodduk $ 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 Ctrl-C to send a SIGINT/2 "Terminal Interrupt"*

*or Ctrl-\ to send a SIGQUIT/3 "Terminal Quit"*

# 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
~
```

13,1 All

# Signals

## Class Exercise

- View Jim's script with: **cat bin/app**
- Look for the three trap handlers
  - Signal 2 (SIGINT)
  - Signal 3 (SIGQUIT)
  - Signal 15 (SIGTERM)

# Signals

## Benji runs app



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

one  
two  
thr█

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

# Signals

## Benji runs app



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

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



# Signals

## Benji runs app



```
simmsben@opus:~  
#####  #####  #####  #####  #####  #  #  #####  
#  #  #  #  #  #  ##  #  #  #  
#  #  #  #  #  #  #  #  #  #  
#  #####  #####  #  #  #  #  #  #####  
#  #  #  #  #  #  #  #  #  #  
#  #  #  #  #  #  #  #  #  #  
#  #####  #####  #  #####  #  #####  
  
one  
two  
thrQuit  
quit it! █
```

*Benji tries using the keyboard to send a SIGQUIT/3 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

```

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

one
two
thrQuit
quit it!

```

```

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

```



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

# Signals

## Benji runs app

```

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

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

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

```



*Benji ups the anty and sends two SIGQUIT/3's but the app process shrugs them off with "quit it!" messages*

# Signals

## Benji runs app

```

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

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

/home/cis90/simmsben $ kill -3 7065
/home/cis90/simmsben $ kill -15 7065
/home/cis90/simmsben $ █

```



*Benji decides to send a SIGTERM/15 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 program that prints a grid of '#' characters. The bottom window shows the output of the 'ps' command, listing running processes.

```

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

one
two
thr█

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



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

## Signals

### Benji runs app

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

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

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



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



# Signals

## Class Exercise

- 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)



# Load Balancing (scheduling)

## Load Balancing with **at** command

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

Any output sent to **stdout** or **stderr** by the list of commands will be emailed to the user unless redirected elsewhere.



## Tools for your toolbox

NEW

**at** - schedule a job to run in the future

NEW

**atq** - list queue of pending jobs

NEW

**atrm** - remove a pending job



# at command

## Basic syntax

(see man page for the rest of the story)

## **at <time>**

*(the at command will then read commands from stdin)*

## Examples

```
at 3:00pm wednesday
```

```
at> echo Meet with Sarah | mail -s 'Reminder' simben90
```

```
at> Ctrl-D
```

*End of file means no more commands to process*

*at prompt (you don't type this, the at command does)*

# at command

Specifying future time examples:

`at now + 5 minutes`

`at now + 2 hours`

`at now + 1 week`

`at 1:00AM`

`at 3:00PM wednesday`

`at 12:00AM 12/25/2014`

`at teatime`

# Activity

You try it:

```
/home/cis90/simben $ at now + 1 minute
```

```
at> banner Hola Benji
```

```
at> <EOT>
```

*Use Ctrl-D for End of File*

```
job 875 at 2014-11-03 14:11
```

```
/home/cis90/simben $ mail
```

Then read your mail a minute later

```
simben90@oslab:~/
/home/cis90/simben $ mail
Heirloom Mail version 12.4 7/29/08. Type ? for help.
"/var/spool/mail/simben90": 1 message 1 new
>N 1 Benji Simms      Mon Nov  3 14:11 30/1211 "Output from your job "
& 1
Message 1:
From: simben90@oslab.cis.cabrillo.edu  Mon Nov  3 14:11:01 2014
Return-Path: <simben90@oslab.cis.cabrillo.edu>
Date: Mon, 3 Nov 2014 14:11:01 -0800
From: Benji Simms <simben90@oslab.cis.cabrillo.edu>
Subject: Output from your job      875
To: simben90@oslab.cis.cabrillo.edu
Status: R

#####
# HOLA #
#####
# BENJI #
#####

& q
Held 1 message in /var/spool/mail/simben90
You have mail in /var/spool/mail/simben90
/home/cis90/simben $
```

*Write in the chat window the name of the sender of the email sent to you*



# Activity

You try it:

```
/home/cis90/simben $ tty
/dev/pts/2
/home/cis90/simben $ at now + 1 minute
at> echo > /dev/pts/2
at> banner Hola Benji > /dev/pts/2
at> <EOT>
job 873 at 2014-11-03 14:04
```

```
simben90@oslab:~
/home/cis90/simben $
# # # # # # # # # #
# # # # # # # # # #
# # # # # # # # # #
##### # # # # # # #
# # # # # # # # # #
# # # # # # # # # #
# # # # # # # # # #
# # # # # # # # # #
# # # # # # # # # #
##### # # # # # # #

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

*Write in the chat window the reason for doing an echo command before the banner command when writing to the terminal device*

## at examples

```
at 12:00 am wednesday  
  chmod 700 /home/rsimms/turnin
```

```
at 9:00 am wednesday  
  chmod 750 /home/rsimms/turnin
```

```
at 11:59pm  
  cat files.out bigshell > lab08  
  cp lab08 /home/rsimms/turnin/cis90/lab08.$LOGNAME
```

```
at 2:50pm tuesday  
  cp /etc/nologin.bak /etc/nologin  
  shutdown -P +10
```

## at job management

```
/home/cis90/simben $ echo chmod 000 letter | at 3:00pm
```

```
job 878 at 2014-11-03 15:00
```

```
/home/cis90/simben $ echo chmod 644 letter | at 3:05pm
```

```
job 879 at 2014-11-03 15:05
```

```
/home/cis90/simben $ echo chmod 640 letter | at 1:00am friday
```

```
job 880 at 2014-11-07 01:00
```

```
/home/cis90/simben $ atq
```

```
879      2014-11-03 15:05 a simben90
```

```
880      2014-11-07 01:00 a simben90
```

```
878      2014-11-03 15:00 a simben90
```

*The **atq** command lists the queue of pending jobs scheduled to run in the future.*

```
/home/cis90/simben $ atrm 879
```

```
/home/cis90/simben $ atq
```

```
880      2014-11-07 01:00 a simben90
```

```
878      2014-11-03 15:00 a simben90
```

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

```
/home/cis90/simben $ atrm 878 880
```

```
/home/cis90/simben $ atq
```

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

## at command error handling

```
/home/cis90/simben $ at now + 1 minute
at> kitty letter
at> <EOT>
job 150 at 2011-04-20 10:47
```

*Oops, specified a non-existent command to run in the future (**kitty** should have been **cat**)*

```
/home/cis90/simben $ atq
150      2011-04-20 10:47 a simmsben
/home/cis90ol/simmsben $ atq
```

```
/home/cis90/simben $ mail
Mail version 8.1 6/6/93.  Type ? for help.
"/var/spool/mail/simben": 1 message 1 new
>N 1 simben@Opus.cabrillo.edu Wed Apr 20 10:47 16/709 "Output from your job "
& 1
Message 1:
From simben@Opus.cabrillo.edu Wed Apr 20 10:47:01 2011
Date: Wed, 20 Apr 2011 10:47:01 -0700
From: Benji Simms <simben@Opus.cabrillo.edu>
Subject: Output from your job 150
To: simben@Opus.cabrillo.edu
```

*Because, you may not be online when the command runs, any error messages are mailed to you.*

```
/bin/bash: line 2: kitty: command not found
```

# Viewing an at jobs

```
/home/cis90/simben $ atq
882      2014-11-03 15:05 a simben90
881      2014-11-03 15:00 a simben90
883      2014-11-07 01:00 a simben90
```

```
/home/cis90/simben $ at -c 883
```

Use the -c option to view the contents of an at job

```
#!/bin/sh
# atrun uid=1201 gid=190
# mail simben90 0
umask 2
HOSTNAME=oslab.cis.cabrillo.edu; export HOSTNAME
SELINUX_ROLE_REQUESTED=; export SELINUX_ROLE_REQUESTED
SHELL=/bin/bash; export SHELL
HISTSIZE=1000; export HISTSIZE
SSH_CLIENT=2601:9:6680:53b:8d5f:4722:4af4:186e\ 59885\ 2220; export SSH_CLIENT
SELINUX_USE_CURRENT_RANGE=; export SELINUX_USE_CURRENT_RANGE
QTDIR=/usr/lib/qt-3.3; export QTDIR
QTINC=/usr/lib/qt-3.3/include; export QTINC
SSH_TTY=/dev/pts/2; export SSH_TTY
USER=simben90; export USER
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33:cd=40;33:or=40;31:oi=01;05;37;41:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44;
ex=01;32;*.tar=01;31;*.tgz=01;31;*.arj=01;31;*.taz=01;31;*.lzh=01;31;*.lha=01;31;*.zip=01;31;*.z=01;31;*.d=01;31;*.gz=01;31;*.l
z=01;31;*.xz=01;31;*.bz2=01;31;*.tbz=01;31;*.tbz2=01;31;*.bz=01;31;*.rz=01;31;*.deb=01;31;*.rpm=01;31;*.jar=01;31;*.ace=01;31;*.zoo=01;31;*.cpio=01;31;
*.7z=01;31;*.rz=01;31;*.jpg=01;35;*.jpeg=01;35;*.gif=01;35;*.bmp=01;35;*.pnm=01;35;*.pgm=01;35;*.ppm=01;35;*.tga=01;35;*.xpm=01;35;*.xpm=01;35;*.tif=01;35;*.tiff=01
;35;*.png=01;35;*.svg=01;35;*.svgz=01;35;*.mng=01;35;*.pcc=01;35;*.mov=01;35;*.mpg=01;35;*.mpeg=01;35;*.m2v=01;35;*.mkv=01;35;*.ogm=01;35;*.mp4=01;35;*.m4v=01;35;*.
mp4=01;35;*.vob=01;35;*.qt=01;35;*.nuv=01;35;*.wmv=01;35;*.asf=01;35;*.rm=01;35;*.rmvb=01;35;*.flc=01;35;*.avi=01;35;*.flv=01;35;*.flv=01;35;*.gl=01;35;*.dl=01;35;
*.act=01;35;*.xwd=01;35;*.xps=01;35;*.cgm=01;35;*.emf=01;35;*.axv=01;35;*.ans=01;35;*.ogv=01;35;*.aac=01;36;*.au=01;36;*.flac=01;36;*.mid=01;36;*.midi=0
1;36;*.mka=01;36;*.mp3=01;36;*.mpc=01;36;*.ogg=01;36;*.ra=01;36;*.wav=01;36;*.axa=01;36;*.oga=01;36;*.spx=01;36;*.xspf=01;36; export LS_COLORS
USERNAME=; export USERNAME
MAIL=/var/spool/mail/simben90; export MAIL
PATH=/usr/lib/qt-3.3/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/cis90/simben/../../bin:/home/cis90/simben/bin:.. export PATH
PWD=/home/cis90/simben; export PWD
LANG=en_US.UTF-8; export LANG
SELINUX_LEVEL_REQUESTED=; export SELINUX_LEVEL_REQUESTED
HISTCONTROL=ignoresdups; export HISTCONTROL
SHLVL=1; export SHLVL
HOME=/home/cis90/simben; export HOME
BASH_ENV=/home/cis90/simben/.bashrc; export BASH_ENV
LOGNAME=simben90; export LOGNAME
QTLIB=/usr/lib/qt-3.3/lib; export QTLIB
CVS_RSH=ssh; export CVS_RSH
SSH_CONNECTION=2601:9:6680:53b:8d5f:4722:4af4:186e\ 59885\ 2607:f380:80f:f425:230\ 2220; export SSH_CONNECTION
LESSOPEN=|/usr/bin/lesspipe.sh %s; export LESSOPEN
G_BROKEN_FILENAMES=1; export G_BROKEN_FILENAMES
cd /home/cis90/simben | |
echo 'Execution directory inaccessible' >2
exit 1
```

reduced in size to fit on slide

All these environment variables must be set to appropriate values so your commands since you may be no longer logged in

```
/${SHELL:-/bin/sh} << 'marcinDELIMITER7acf33a1'
```

```
chmod 640 letter
```

This is where you will see your own commands

```
marcinDELIMITER7acf33a1
/home/cis90/simben $
```

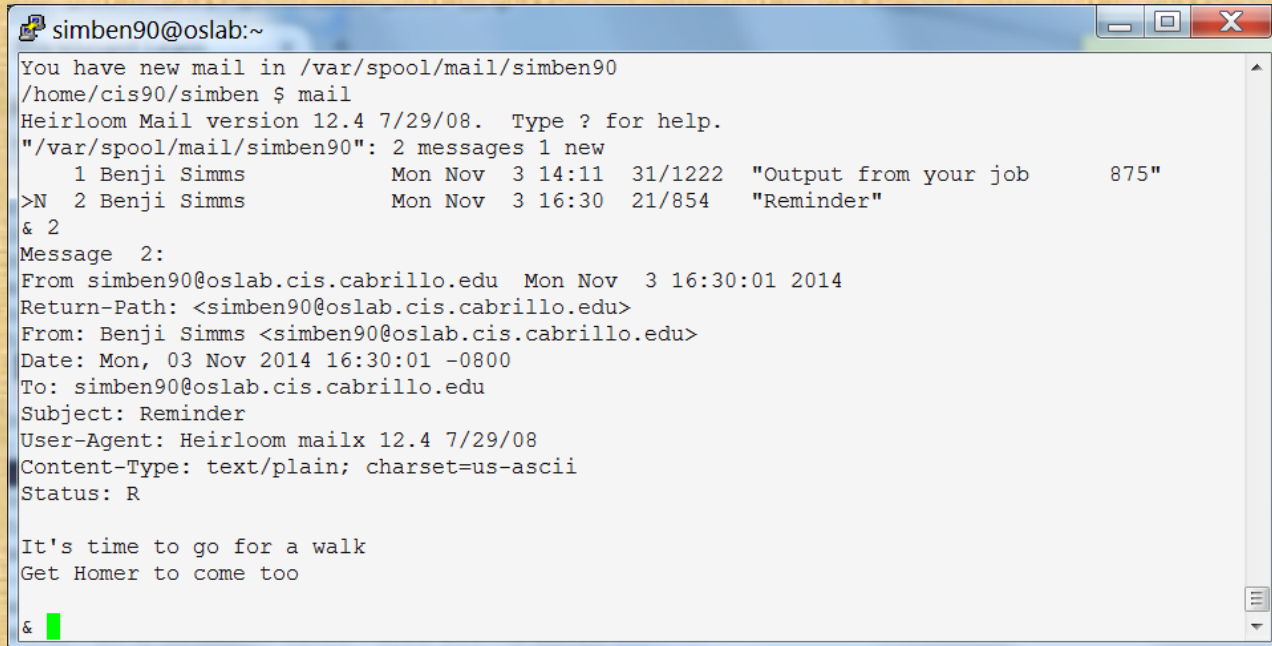
## Activity

### Schedule an email reminder

```

/home/cis90/simben $ at 16:30
at> echo "It's time to go for a walk" > /tmp/message.benji
at> echo "Get Homer to come too" >> /tmp/message.benji
at> cat /tmp/message.benji | mail -s "Reminder" simben90
at> rm /tmp/message.benji
at> <EOT>
/home/cis90/simben $

```



```

simben90@oslab:~
You have new mail in /var/spool/mail/simben90
/home/cis90/simben $ mail
Heirloom Mail version 12.4 7/29/08. Type ? for help.
"/var/spool/mail/simben90": 2 messages 1 new
  1 Benji Simms      Mon Nov  3 14:11  31/1222  "Output from your job      875"
>N 2 Benji Simms      Mon Nov  3 16:30  21/854   "Reminder"
& 2
Message 2:
From simben90@oslab.cis.cabrillo.edu Mon Nov  3 16:30:01 2014
Return-Path: <simben90@oslab.cis.cabrillo.edu>
From: Benji Simms <simben90@oslab.cis.cabrillo.edu>
Date: Mon, 03 Nov 2014 16:30:01 -0800
To: simben90@oslab.cis.cabrillo.edu
Subject: Reminder
User-Agent: Heirloom mailx 12.4 7/29/08
Content-Type: text/plain; charset=us-ascii
Status: R

It's time to go for a walk
Get Homer to come too

&

```



# Assignment





**Lab 8**

In this lab you will use the `ps` command to monitor processes as you create them using UNIX commands.

**Prerequisites**

- Simms Lesson 10 slides: <http://simms-teach.com/cis90calendar.php>
- Check the forum for notes on this lab: <http://oslab.cis.cabrillo.edu/forum/>
- For additional assistance come to the CIS Lab: <http://webhawks.org/~cislab/>

**Preparation**

Log on to Opus so that you have a confirmed live shell at your service. Be sure you are in your home directory to start this lab.

1. Run the C shell program `ccsh`. Did your prompt change?
2. Now run the Bourne shell `sh`. Different prompt again?
3. Run the `ps` command to see that you have three shell processes running.
4. Run the `ps` command with the `-l` option (l for long format). Look at the column headed by the symbol `PPID`. This is the size of the process in 1K blocks. Which of the three shells that you are running is the largest? Redirect the file of output to the file `ps.out`.
5. Now terminate the Bourne and C shells by typing the `exit` command twice.
6. Run the `ps` command with the `-f` option. What is the parent (PPID) of your shell process? The Grandparent? The Great Grandparent? How far can you go?
7. What is the name of the program with the PID of 1? What is its parent?
8. Run the `top` command in the foreground.
9. Notice that you are stuck. Bring up another window on Opus and kill this process. Hint: use the command `ps -f -o ppid,ppppid` to find the PID number.
10. Run the `top` command in the background by adding `&` on this command line. (Hit the `ctrl` key to get your prompt back)

## Lab 8

*Doesn't take too long but don't wait till the last minute on this lab!*



# Wrap up

New commands:

Ctrl-Z or F  
bg

Suspends a foreground process  
Resumes suspended process

&  
fg

Runs command in the background  
Brings background job to foreground

jobs

show background jobs

kill  
killall

Send a signal to a process by PID  
Send a signal to a process by name

at  
atq  
atrm

Run job once in the future  
Show all *at* jobs queued to run  
Remove *at* jobs from queue

sleep

Sleep for specified amount of time

stty

Terminal control



## Next Class

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

Lab 8 due

Quiz #8 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?



# Test 2





## Notes to instructor

### Practice Test System

[ ] Start: `echo "/root/unlock-cis90; rm /etc/nologin" | at [T-1wk+1hr]`

[ ] End: `echo "/root/lock-cis90; cp /etc/nologin.bak /etc/nologin" | at [T-30]`

### Canvas Practice Test:

[ ] availability from = `[T-1wk+1hr]`, due & available until = `[T-30]`

[ ] remove password on real test on Canvas `[T-1wk]`

[ ] moderate any accommodations

### Real Test system

[ ] Start: `echo "/root/unlock-cis90; rm /etc/nologin" | at [T-0]`

[ ] End: `echo "/root/lock-cis90; cp /etc/nologin.bak /etc/nologin" | at [splashdown]`

### Canvas Real Test:

[ ] availability from = `[T-0]`, due & available until = `[splashdown]`

[ ] remove password on real test on Canvas `[T-0]`

[ ] moderate any accommodations

# Test Instructions

## HONOR CODE:

This test is open book, open notes, and open computer. HOWEVER, you must work alone. You may not discuss the test questions or answers with others during the test period. You may not ask or receive assistance from anyone other than the instructor when doing this test. Likewise you may not give any assistance to anyone taking the test.

## INSTRUCTIONS:

Test system: sun-hwa-t2.cis.cabrillo.edu (port 22)

This test should be completed using the sun-hwa-t2 system only. Because this system is on a private network, log into Opus first, then ssh into sun-hwa-t2.

Grading will be based on your answers AND that you correctly implemented the "DO THIS FIRST" portion of each question.

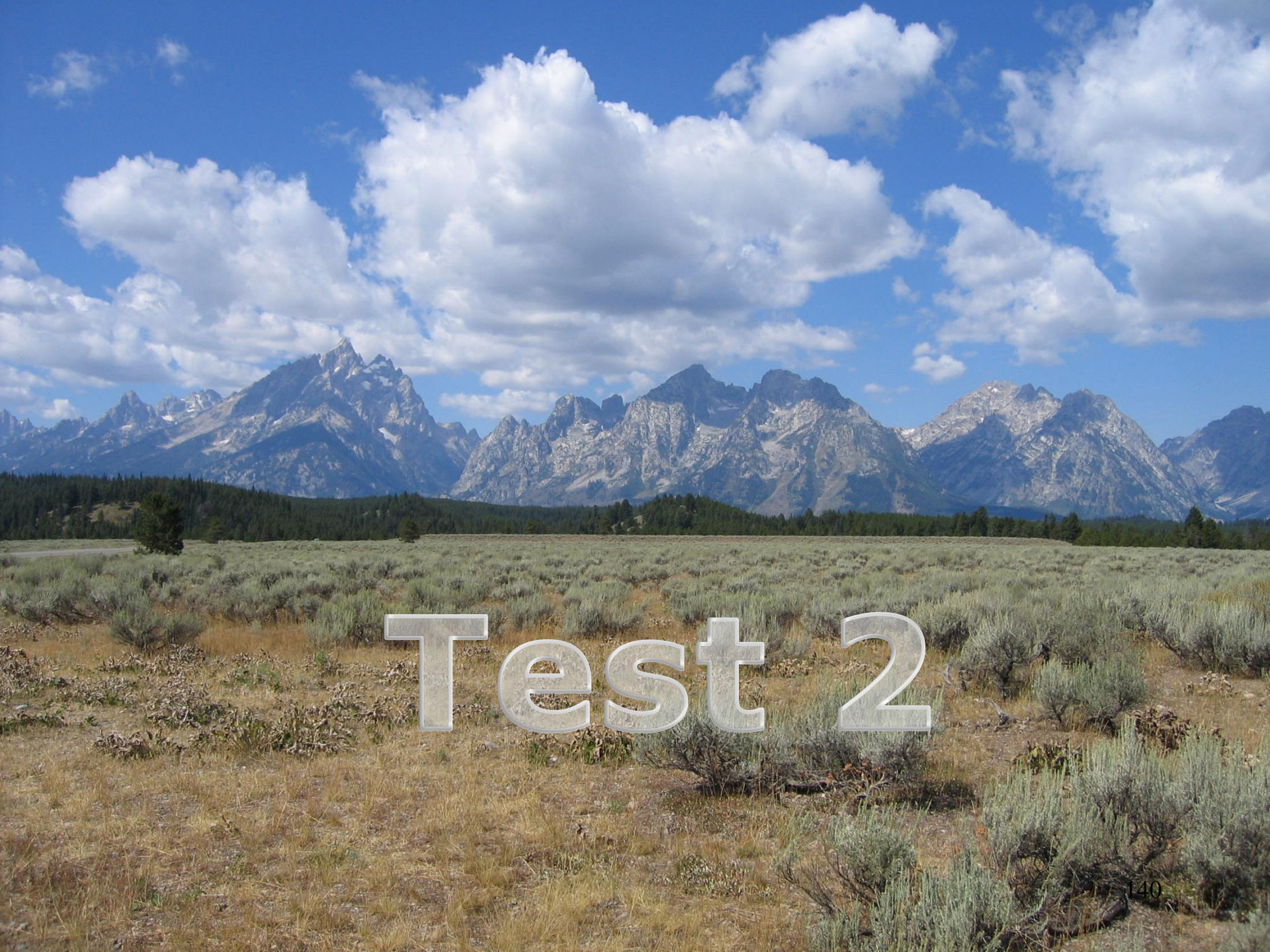
**Highlighted text** indicates a portion of a question that differs from the practice test.

If you get stuck on a question and can't proceed you can ask the instructor for help and forfeit the point. The instructor will be available during class and available by email (risimms@cabrillo.edu) later in the evening from 8:00-10:00PM.

Please KEEP YOUR ANSWERS TO A SINGLE LINE ONLY !!

This test must be completed in one sitting. The submittal will be made automatically when the time is up. If you submit early by accident you will not be able to re-enter and continue. If that happens don't panic! Just email the instructor any remaining answers before the time is up.





# Test 2



# Backup





# umask

# Review

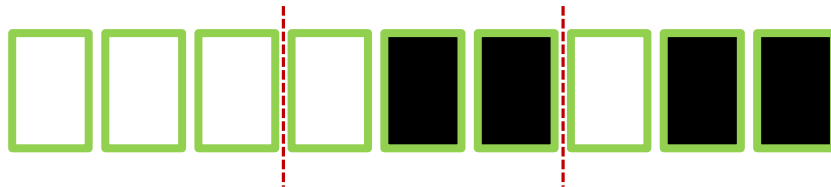
## umask summary

- Use the **umask** command to specify the permissions you want stripped from future new files and directories
- Does not change permissions on existing files

To determine permissions on a new file or directory apply the umask to the initial permission starting point:

- For new files, start with **666**
- For new directories, start with **777**
- *For file copies, start with **the permission on the source file being copied***

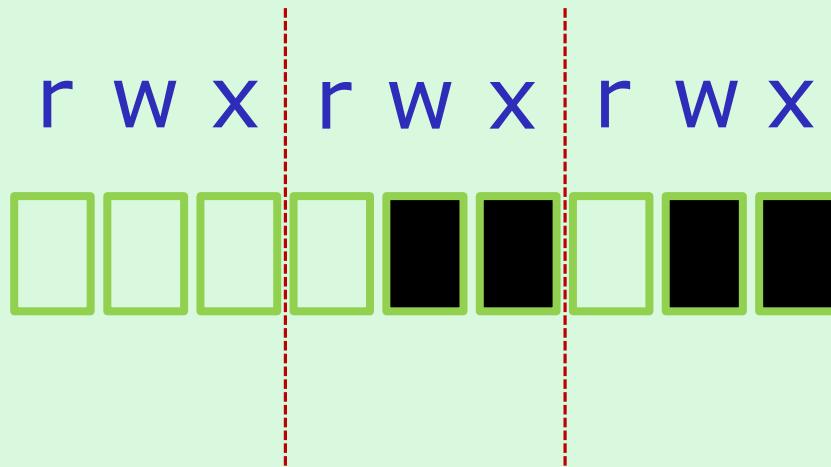
With a umask of 033 what permissions would a newly created directory have?



**umask setting of 033 strips these bits: --- -wx -wx**

## Example 1 - new directory

With a umask of 033 what permissions would a newly created directory have?

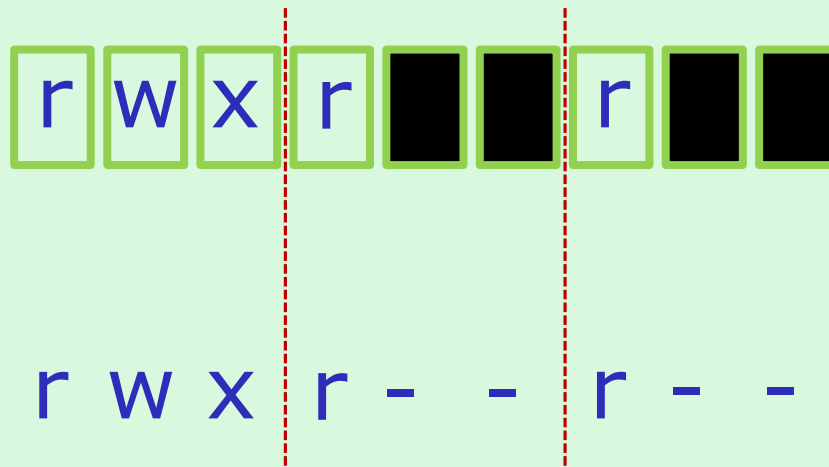


starting point = 777  
(new directory)

umask setting of 033 strips  
these bits: --- -wx -wx

## Example 1 - new directory

With a umask of 033 what permissions would a newly created directory have?



starting point = 777  
(new directory)

umask setting of 033 strips  
these bits: --- -wx -wx

Answer: 744

Verify your answer on Opus:

```
/home/cis90ol/simmsben $ umask 033
/home/cis90ol/simmsben $ mkdir brandnewdir
/home/cis90ol/simmsben $ ls -ld brandnewdir/
drwxr--r-- 2 simmsben cis90ol 4096 Apr 21 12:46 brandnewdir/
```



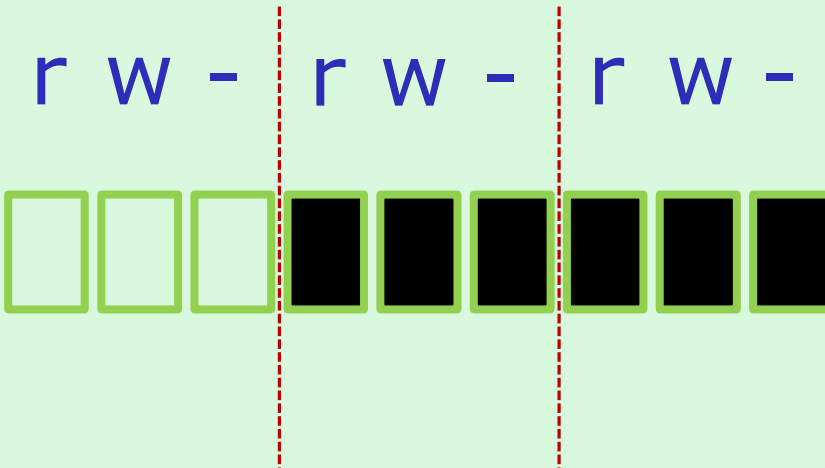
With a umask of 077 what permissions would a newly created file have?



From issuing **umask 077**

## Example 2 - new file

With a umask of 077 what permissions would a newly created file have?



starting point = 666  
(new file)

umask setting of 077 strips  
these bits: --- rwx rwx

## Example 2 - new file

With a umask of 077 what permissions would a newly created file have?



starting point = 666  
(new file)

umask setting of 077 strips  
these bits: --- rwx rwx

r w - - - - -

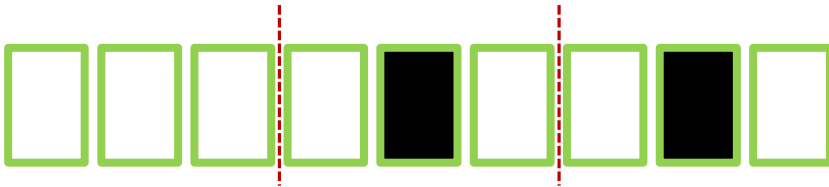
Answer: 600

*Verify your answer on Opus:*

```
/home/cis90ol/simmsben $ umask 077
/home/cis90ol/simmsben $ touch brandnewfile
/home/cis90ol/simmsben $ ls -l brandnewfile
-rw----- 1 simmsben cis90ol 0 Apr 21 12:50 brandnewfile
```

If `umask=022` and *cinderella* file permissions=`622`

What would the permissions be on the file *cinderella.bak* after:  
**`cp cinderella cinderella.bak`**

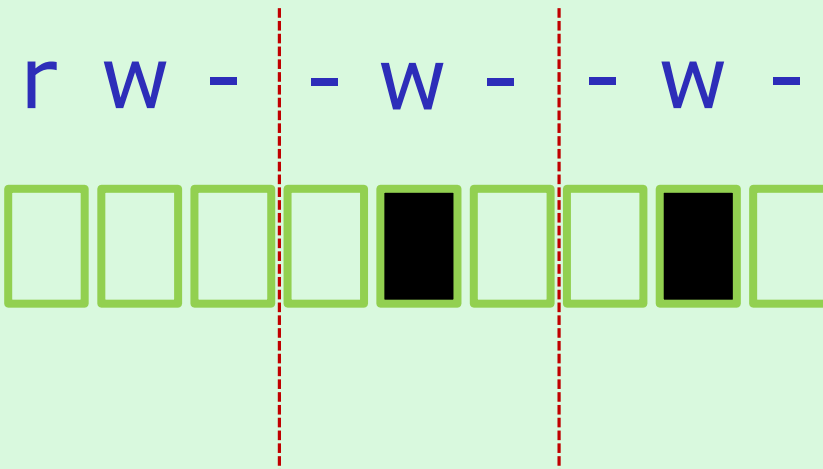


From issuing **`umask 022`**

## Example 2 - file copy

If `umask=022` and the `cinderella` file permissions=`622`

What would the permissions be on the file `cinderella.bak` after:  
**`cp cinderella cinderella.bak`**



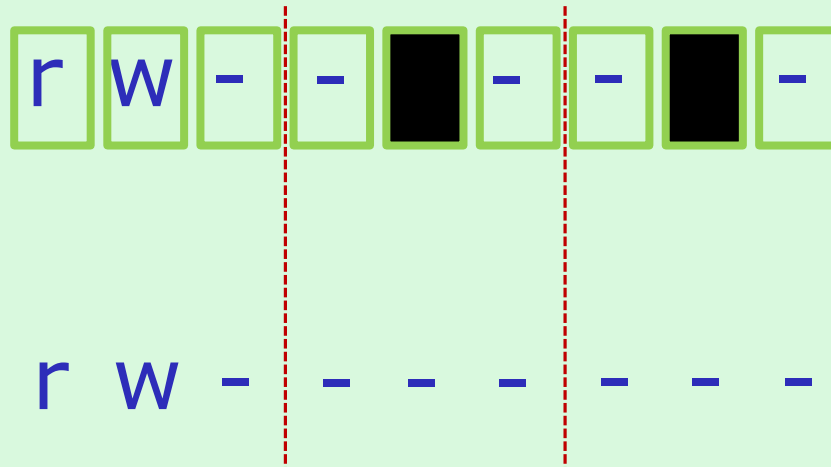
starting point = 622  
(source file permissions)

umask setting of 022 strips  
these bits: `--- -w- -w-`

## Example 2 - file copy

If `umask=022` and the `cinderella` file permissions=`622`

What would the permissions be on the file `cinderella.bak` after:  
**`cp cinderella cinderella.bak`**



starting point = 622  
(source file permissions)

umask setting of 022 strips  
these bits: --- -w- -w-

Answer: 600

Verify your answer on Opus:

```
/home/cis90ol/simmsben $ touch cinderella
/home/cis90ol/simmsben $ chmod 622 cinderella
/home/cis90ol/simmsben $ umask 022
/home/cis90ol/simmsben $ cp cinderella cinderella.bak
/home/cis90ol/simmsben $ ls -l cinderella.bak
-rw----- 1 simmsben cis90ol 0 Apr 21 12:53 cinderella.bak
```