

Lesson Module Checklist

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

- Lab tested and uploaded –
- Test02 uploaded, permissions set –
- Test02 mods made to Opus and Sun-Hwa -
- Real test uploaded and permissions set –
- CCC Confer wall paper – NA

- Materials uploaded –
- Backup slides, CCC info, handouts on flash drive -
- Check that backup room headset is charged –
- Spare 9v battery for mic



Instructor: **Rich Simms**

Dial-in: **888-450-4821**

Passcode: **761867**



Sean C.



Donald



Carlile



Andrew



Sean Fa.



Carter



Sean Fy.



Dajan



Bryn



Rita



Kelly



Ben



Ray



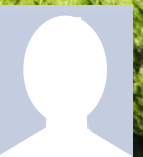
Fidel



Michael



Evan



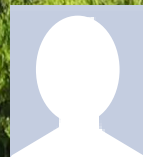
Josh



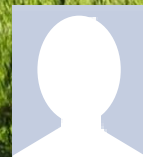
Carlos



Gustavo



Jessica



Evie



Jacob



Humberto



Chad

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

Quiz

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

No Quiz Today

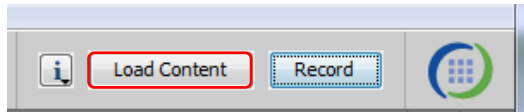
But we do have a test!

email answers to: risimms@cabrillo.edu

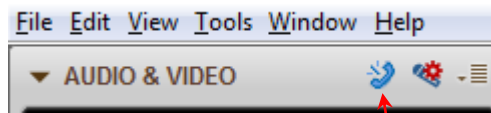
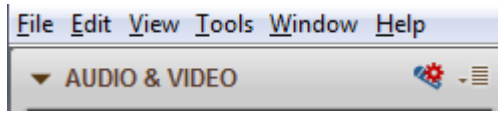
(answers must be emailed within the first few minutes of class for credit) 3



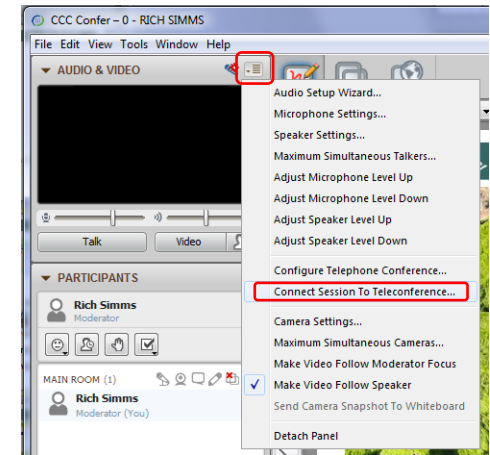
[] Load White Board with *cis*lesson??*-WB*



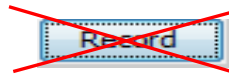
[] Connect session to Teleconference



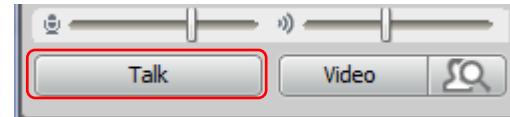
Connected to teleconference



[] Is recording on?



[] Toggle Talk button to not use Mic





- [] Video (webcam) optional
- [] layout and share apps

The screenshot shows a Windows desktop environment during a video conference. On the left is the 'CCC Confer' application window. In the center is a 'Foxit Reader' window displaying a PDF document titled 'cis90lesson07.pdf'. On the right is a 'Chrome' browser window showing a webpage with flashcard questions. In the foreground is a 'Putty' terminal window showing a login attempt for 'simben90' on 'oslab.cabrillo.edu'. Red boxes with white text label 'foxit for slides', 'chrome', and 'putty'. Red arrows point from these labels to the corresponding windows. The desktop taskbar at the bottom shows icons for various applications, and the system tray shows the time as 6:52 AM on 10/10/2012.

First Minute Quiz

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

No Quiz Today

But we do have a test!

**email answers to: 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 from last week
- Housekeeping
- Process definition
- Process lifecycle
- Process information
- Job control
- Signals
- Load balancing
- Wrap up
- Test #2



?'S

Previous material and assignment

1. Questions on previous material?

- File management (Lesson 6)
- Permissions (Lesson 7)
- Input/output (Lesson 8)
- Labs
- Practice test

2. Questions regarding the test today?

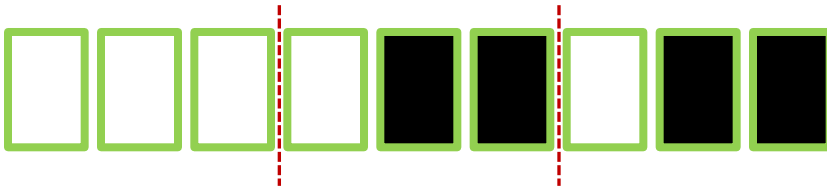
- Test will start during the last hour of class.
- Should take about 45-60 minutes
- If you wish, you can keep working on it till 11:59PM.
- You must do all the work on the test by yourself and not ask or give help to others regarding any of the test questions.

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***

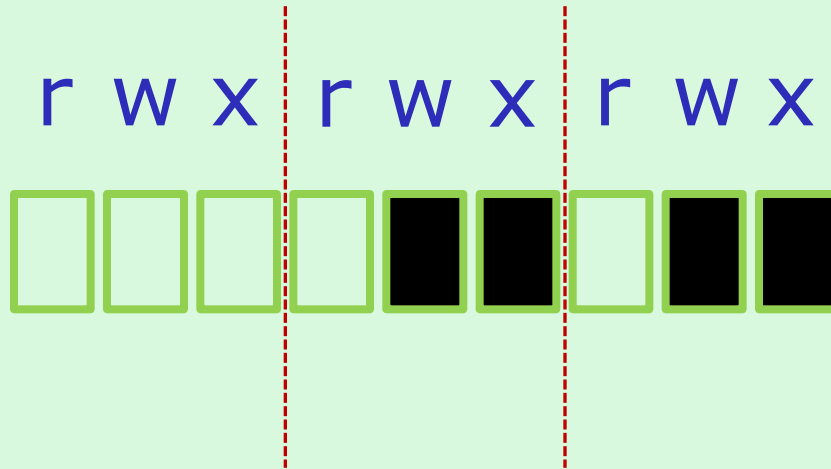
With a umask of 033 what permissions would a newly created directory have?



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?

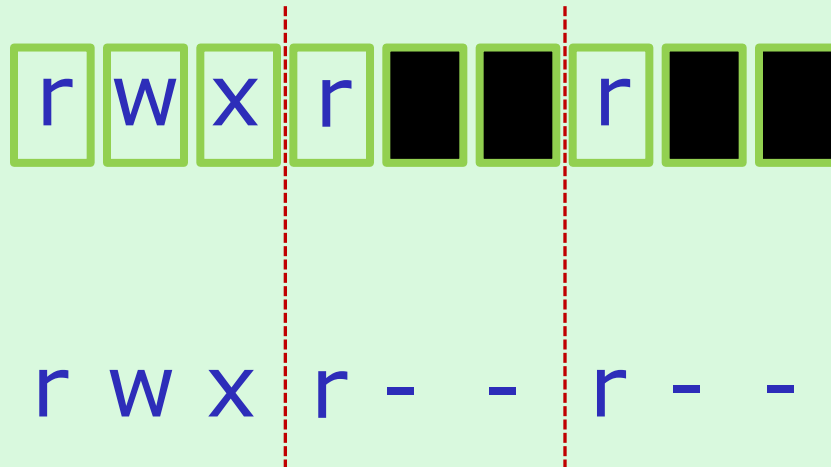


starting point = 777
(new directory)

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

Prove it to yourself on Opus as shown here

```
/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/
```

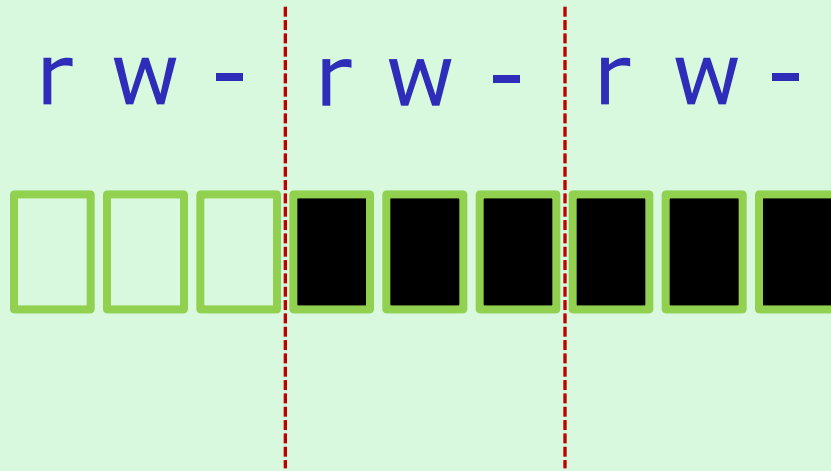
With a umask of 077 what permissions would a newly created file have?



From issuing **umask 077**

Example 2 – new file

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

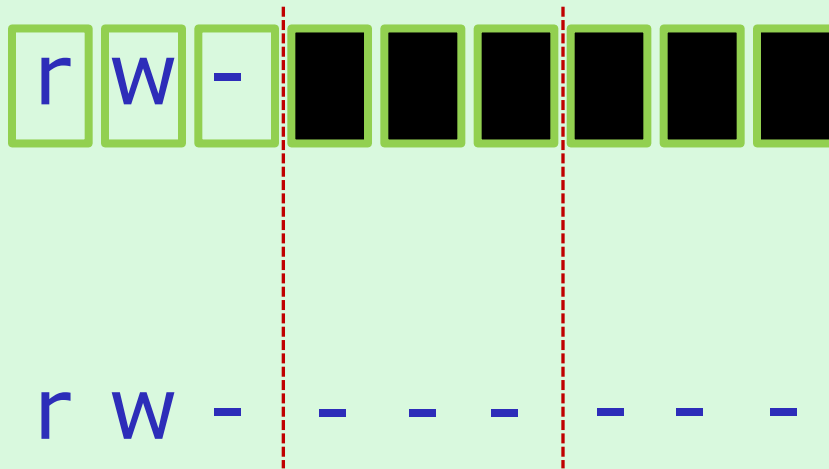


starting point = 666
(new file)

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

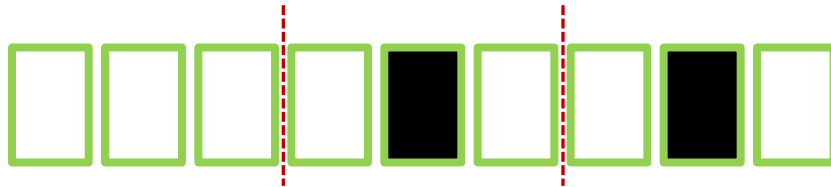
Answer: 600

Prove it to yourself on Opus as shown here

```
/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



r w - - W - - W -



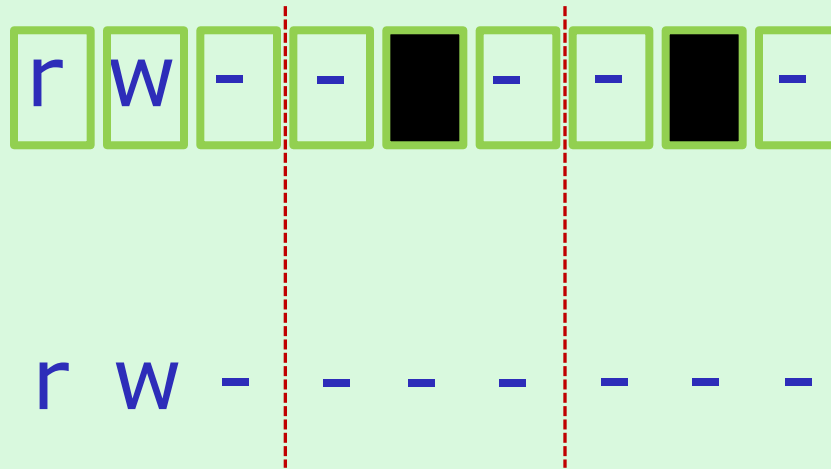
starting point = 622
(source file permissions)

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

Prove it to yourself on Opus as shown here

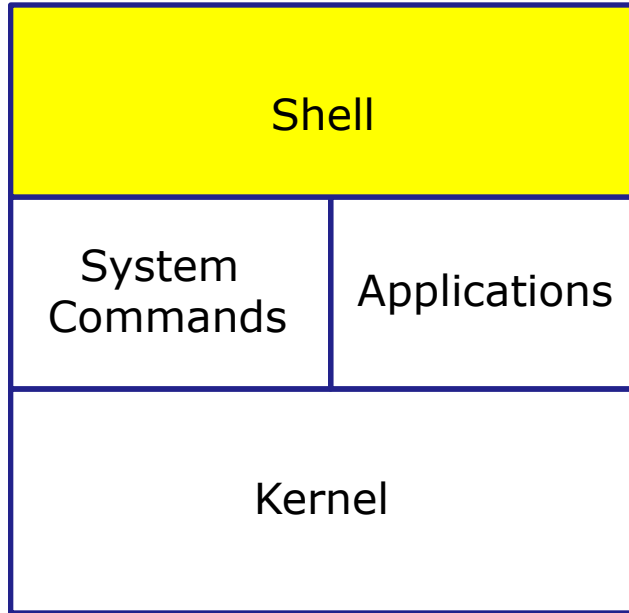
```
/home/cis90ol/simmsben $ touch cinderella
/home/cis90ol/simmsben $ chmod 622 cinderella
/home/cis90ol/simmsben $ umask 022
/home/cis90ol/simmsben $ cp cinderella cinderella.bak
/home/cis90ol/simmsben $ ls -l cinderella.bak
-rw----- 1 simmsben cis90ol 0 Apr 21 12:53 cinderella.bak
```

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 hands a clean 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 how bash expands pathnames

```
/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 $
```

shows what arguments are actually passed to the command when it's loaded

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:~'
```

Shows how quoted text strings get handled for variables

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

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

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

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

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

```
+ set -x
```

```
++ echo -ne '\033]0;simben90@opus:~'
```

```
/home/cis90/simben $ find . -name *.egg
```

```
+ find . -name 1991.egg
```

```
./1991.egg
```

```
++ echo -ne '\033]0;simben90@opus:~'
```

```
/home/cis90/simben $ find . -name "*.egg"
```

```
+ find . -name '*.egg'
```

```
./1991.egg
```

```
./basket/.1993.egg
```

```
< snipped >
```

```
./basket/.1969.egg
```

```
./basket/.1972.egg
```

```
++ echo -ne '\033]0;simben90@opus:~'
```

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

```
+ set +x
```

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

Shows how quoted text strings get handled

Shows filename expansion metacharacters without quotes are expanded and those in quotes are not

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*



Housekeeping

Managing your grade

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

Points gone by

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

301 points

Points yet to earn

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

259 points

- Plus extra credit - up to 90 points

Managing your grade

Rich's Cabrillo College CIS 90 Grades

Points can be earned from the following activities:

- 7% - Quizzes
- 16% - Tests
- 14% - Help forum participation
- 54% - Lab assignments
- 3% - Final project

How your grade is determined:
A student can earn up to 560 total points during the activities listed above. The course grade is based on the number of points earned.

Percentage	Total Points	Letter Grade	Pass/Fail
80% or higher	504 or higher	A	Pass
60% to 79.9%	448 to 503	B	Pass
40% to 59.9%	392 to 447	C	Pass
20% to 39.9%	336 to 391	D	No pass
0% to 19.9%	0 to 335	No pass	No pass

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

Choice of Grade or Pass/No Pass
You indicate your grading choice on the Student Survey form passed out during the first class. You can verify your grading choice options on the table below. Contact the instructor by email with any questions or to request a change in grading choice.

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

Current Progress

Code	Grading	Choice	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	Q23	Q24	Q25	Q26	Q27	Q28	Q29	Q30	Q31	Q32	Q33	Q34	Q35	Q36	Q37	Q38	Q39	Q40	Q41	Q42	Q43	Q44	Q45	Q46	Q47	Q48	Q49	Q50	Q51	Q52	Q53	Q54	Q55	Q56	Q57	Q58	Q59	Q60	Q61	Q62	Q63	Q64	Q65	Q66	Q67	Q68	Q69	Q70	Q71	Q72	Q73	Q74	Q75	Q76	Q77	Q78	Q79	Q80	Q81	Q82	Q83	Q84	Q85	Q86	Q87	Q88	Q89	Q90	Q91	Q92	Q93	Q94	Q95	Q96	Q97	Q98	Q99	Q100	Q101	Q102	Q103	Q104	Q105	Q106	Q107	Q108	Q109	Q110	Q111	Q112	Q113	Q114	Q115	Q116	Q117	Q118	Q119	Q120	Q121	Q122	Q123	Q124	Q125	Q126	Q127	Q128	Q129	Q130	Q131	Q132	Q133	Q134	Q135	Q136	Q137	Q138	Q139	Q140	Q141	Q142	Q143	Q144	Q145	Q146	Q147	Q148	Q149	Q150	Q151	Q152	Q153	Q154	Q155	Q156	Q157	Q158	Q159	Q160	Q161	Q162	Q163	Q164	Q165	Q166	Q167	Q168	Q169	Q170	Q171	Q172	Q173	Q174	Q175	Q176	Q177	Q178	Q179	Q180	Q181	Q182	Q183	Q184	Q185	Q186	Q187	Q188	Q189	Q190	Q191	Q192	Q193	Q194	Q195	Q196	Q197	Q198	Q199	Q200	Q201	Q202	Q203	Q204	Q205	Q206	Q207	Q208	Q209	Q210	Q211	Q212	Q213	Q214	Q215	Q216	Q217	Q218	Q219	Q220	Q221	Q222	Q223	Q224	Q225	Q226	Q227	Q228	Q229	Q230	Q231	Q232	Q233	Q234	Q235	Q236	Q237	Q238	Q239	Q240	Q241	Q242	Q243	Q244	Q245	Q246	Q247	Q248	Q249	Q250	Q251	Q252	Q253	Q254	Q255	Q256	Q257	Q258	Q259	Q260	Q261	Q262	Q263	Q264	Q265	Q266	Q267	Q268	Q269	Q270	Q271	Q272	Q273	Q274	Q275	Q276	Q277	Q278	Q279	Q280	Q281	Q282	Q283	Q284	Q285	Q286	Q287	Q288	Q289	Q290	Q291	Q292	Q293	Q294	Q295	Q296	Q297	Q298	Q299	Q300	Q301	Q302	Q303	Q304	Q305	Q306	Q307	Q308	Q309	Q310	Q311	Q312	Q313	Q314	Q315	Q316	Q317	Q318	Q319	Q320	Q321	Q322	Q323	Q324	Q325	Q326	Q327	Q328	Q329	Q330	Q331	Q332	Q333	Q334	Q335	Q336	Q337	Q338	Q339	Q340	Q341	Q342	Q343	Q344	Q345	Q346	Q347	Q348	Q349	Q350	Q351	Q352	Q353	Q354	Q355	Q356	Q357	Q358	Q359	Q360	Q361	Q362	Q363	Q364	Q365	Q366	Q367	Q368	Q369	Q370	Q371	Q372	Q373	Q374	Q375	Q376	Q377	Q378	Q379	Q380	Q381	Q382	Q383	Q384	Q385	Q386	Q387	Q388	Q389	Q390	Q391	Q392	Q393	Q394	Q395	Q396	Q397	Q398	Q399	Q400	Q401	Q402	Q403	Q404	Q405	Q406	Q407	Q408	Q409	Q410	Q411	Q412	Q413	Q414	Q415	Q416	Q417	Q418	Q419	Q420	Q421	Q422	Q423	Q424	Q425	Q426	Q427	Q428	Q429	Q430	Q431	Q432	Q433	Q434	Q435	Q436	Q437	Q438	Q439	Q440	Q441	Q442	Q443	Q444	Q445	Q446	Q447	Q448	Q449	Q450	Q451	Q452	Q453	Q454	Q455	Q456	Q457	Q458	Q459	Q460	Q461	Q462	Q463	Q464	Q465	Q466	Q467	Q468	Q469	Q470	Q471	Q472	Q473	Q474	Q475	Q476	Q477	Q478	Q479	Q480	Q481	Q482	Q483	Q484	Q485	Q486	Q487	Q488	Q489	Q490	Q491	Q492	Q493	Q494	Q495	Q496	Q497	Q498	Q499	Q500	Q501	Q502	Q503	Q504	Q505	Q506	Q507	Q508	Q509	Q510	Q511	Q512	Q513	Q514	Q515	Q516	Q517	Q518	Q519	Q520	Q521	Q522	Q523	Q524	Q525	Q526	Q527	Q528	Q529	Q530	Q531	Q532	Q533	Q534	Q535	Q536	Q537	Q538	Q539	Q540	Q541	Q542	Q543	Q544	Q545	Q546	Q547	Q548	Q549	Q550	Q551	Q552	Q553	Q554	Q555	Q556	Q557	Q558	Q559	Q560	Q561	Q562	Q563	Q564	Q565	Q566	Q567	Q568	Q569	Q570	Q571	Q572	Q573	Q574	Q575	Q576	Q577	Q578	Q579	Q580	Q581	Q582	Q583	Q584	Q585	Q586	Q587	Q588	Q589	Q590	Q591	Q592	Q593	Q594	Q595	Q596	Q597	Q598	Q599	Q600	Q601	Q602	Q603	Q604	Q605	Q606	Q607	Q608	Q609	Q610	Q611	Q612	Q613	Q614	Q615	Q616	Q617	Q618	Q619	Q620	Q621	Q622	Q623	Q624	Q625	Q626	Q627	Q628	Q629	Q630	Q631	Q632	Q633	Q634	Q635	Q636	Q637	Q638	Q639	Q640	Q641	Q642	Q643	Q644	Q645	Q646	Q647	Q648	Q649	Q650	Q651	Q652	Q653	Q654	Q655	Q656	Q657	Q658	Q659	Q660	Q661	Q662	Q663	Q664	Q665	Q666	Q667	Q668	Q669	Q670	Q671	Q672	Q673	Q674	Q675	Q676	Q677	Q678	Q679	Q680	Q681	Q682	Q683	Q684	Q685	Q686	Q687	Q688	Q689	Q690	Q691	Q692	Q693	Q694	Q695	Q696	Q697	Q698	Q699	Q700	Q701	Q702	Q703	Q704	Q705	Q706	Q707	Q708	Q709	Q710	Q711	Q712	Q713	Q714	Q715	Q716	Q717	Q718	Q719	Q720	Q721	Q722	Q723	Q724	Q725	Q726	Q727	Q728	Q729	Q730	Q731	Q732	Q733	Q734	Q735	Q736	Q737	Q738	Q739	Q740	Q741	Q742	Q743	Q744	Q745	Q746	Q747	Q748	Q749	Q750	Q751	Q752	Q753	Q754	Q755	Q756	Q757	Q758	Q759	Q760	Q761	Q762	Q763	Q764	Q765	Q766	Q767	Q768	Q769	Q770	Q771	Q772	Q773	Q774	Q775	Q776	Q777	Q778	Q779	Q780	Q781	Q782	Q783	Q784	Q785	Q786	Q787	Q788	Q789	Q790	Q791	Q792	Q793	Q794	Q795	Q796	Q797	Q798	Q799	Q800	Q801	Q802	Q803	Q804	Q805	Q806	Q807	Q808	Q809	Q810	Q811	Q812	Q813	Q814	Q815	Q816	Q817	Q818	Q819	Q820	Q821	Q822	Q823	Q824	Q825	Q826	Q827	Q828	Q829	Q830	Q831	Q832	Q833	Q834	Q835	Q836	Q837	Q838	Q839	Q840	Q841	Q842	Q843	Q844	Q845	Q846	Q847	Q848	Q849	Q850	Q851	Q852	Q853	Q854	Q855	Q856	Q857	Q858	Q859	Q860	Q861	Q862	Q863	Q864	Q865	Q866	Q867	Q868	Q869	Q870	Q871	Q872	Q873	Q874	Q875	Q876	Q877	Q878	Q879	Q880	Q881	Q882	Q883	Q884	Q885	Q886	Q887	Q888	Q889	Q890	Q891	Q892	Q893	Q894	Q895	Q896	Q897	Q898	Q899	Q900	Q901	Q902	Q903	Q904	Q905	Q906	Q907	Q908	Q909	Q910	Q911	Q912	Q913	Q914	Q915	Q916	Q917	Q918	Q919	Q920	Q921	Q922	Q923	Q924	Q925	Q926	Q927	Q928	Q929	Q930	Q931	Q932	Q933	Q934	Q935	Q936	Q937	Q938	Q939	Q940	Q941	Q942	Q943	Q944	Q945	Q946	Q947	Q948	Q949	Q950	Q951	Q952	Q953	Q954	Q955	Q956	Q957	Q958	Q959	Q960	Q961	Q962	Q963	Q964	Q965	Q966	Q967	Q968	Q969	Q970	Q971	Q972	Q973	Q974	Q975	Q976	Q977	Q978	Q979	Q980	Q981	Q982	Q983	Q984	Q985	Q986	Q987	Q988	Q989	Q990	Q991	Q992	Q993	Q994	Q995	Q996	Q997	Q998	Q999	Q1000
------	---------	--------	----	----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------

*As of October 27, 2012
(using Jesse's
checkgrades script)*

```

anborn: 74% (225 of 301 points)
arador: 62% (189 of 301 points)
aragorn: 68% (205 of 301 points)
balrog: 54% (165 of 301 points)
bombadil: 90% (273 of 301 points)
boromir: 62% (189 of 301 points)
celeborn: 105% (317 of 301 points)
dori: 51% (156 of 301 points)
elrond: 65% (196 of 301 points)
eomer: 83% (251 of 301 points)
gimli: 40% (123 of 301 points)
goldberry: 54% (165 of 301 points)
huan: 101% (307 of 301 points)
ingold: 101% (305 of 301 points)
marhari: 46% (140 of 301 points)
pallando: 78% (237 of 301 points)
quickbeam: 27% (84 of 301 points)
samwise: 74% (223 of 301 points)
saruman: 95% (288 of 301 points)
sauron: 105% (317 of 301 points)
shadowfax: 106% (320 of 301 points)
smeagol: 99% (298 of 301 points)
theoden: 92% (279 of 301 points)
tulkas: 81% (244 of 301 points)

```

Managing your grade Getting extra help for CIS 90

Rich's Cabrillo College CIS Classes
CIS 90 Grades

Home Resources Forums **CIS Lab** CTC

CIS 90 (Fall 2010) Grades
Course Home Calendar

Points can be earned from the following activities:

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

How your grade is determined:
A student can earn up to 560 total points doing the activities listed above. The course grade is number of points earned.

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

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

Choice of Grade or Pass/No Pass
You indicate your grading choice on the Student Survey form passed out during the first class grading choice selection on the table below. Contact the instructor by email with any question

Come by the lab and get help from instructors and student assistants

Cabrillo Network & Systems Technology Lab
Aptos Campus

Home Resources NETLAB Location

Fall 2012 Instructor and Lab Assistant Hours

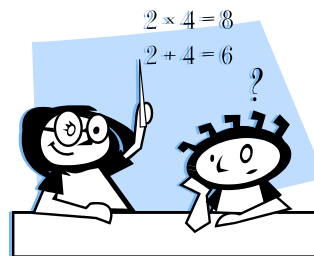
Note: The CIS Lab is closed on holidays and spring break (Sep 3, Nov 12, Nov 22-23)

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

Gerlinde=Gerlinde Brady, Jim=Jim Griffin, Rich=Rich Simms

Managing your grade Getting extra help for CIS 90

- Rich's Office Hours Wed 4:20PM-5:10PM in Room 2501 (right after class) or TBA
- Ask questions on the Forum at:
<http://opus.cabrillo.edu/forum/>

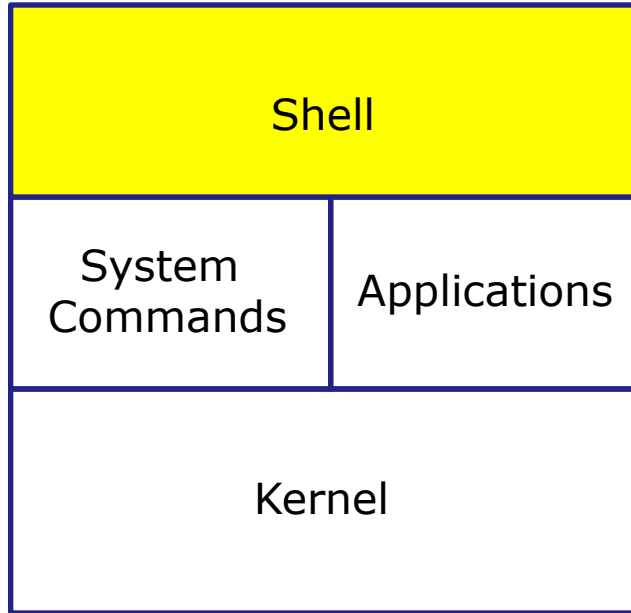




Process Definition



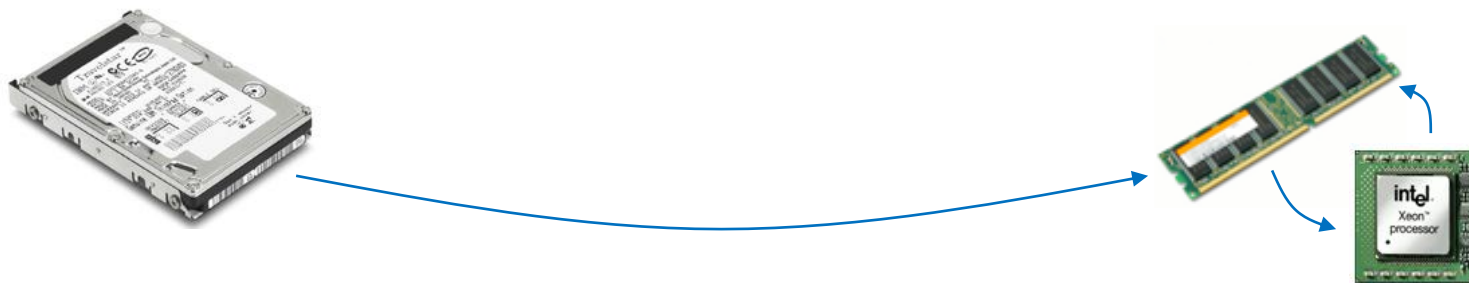
The Shell **Execute** Step



- 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 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**

Definition of a process

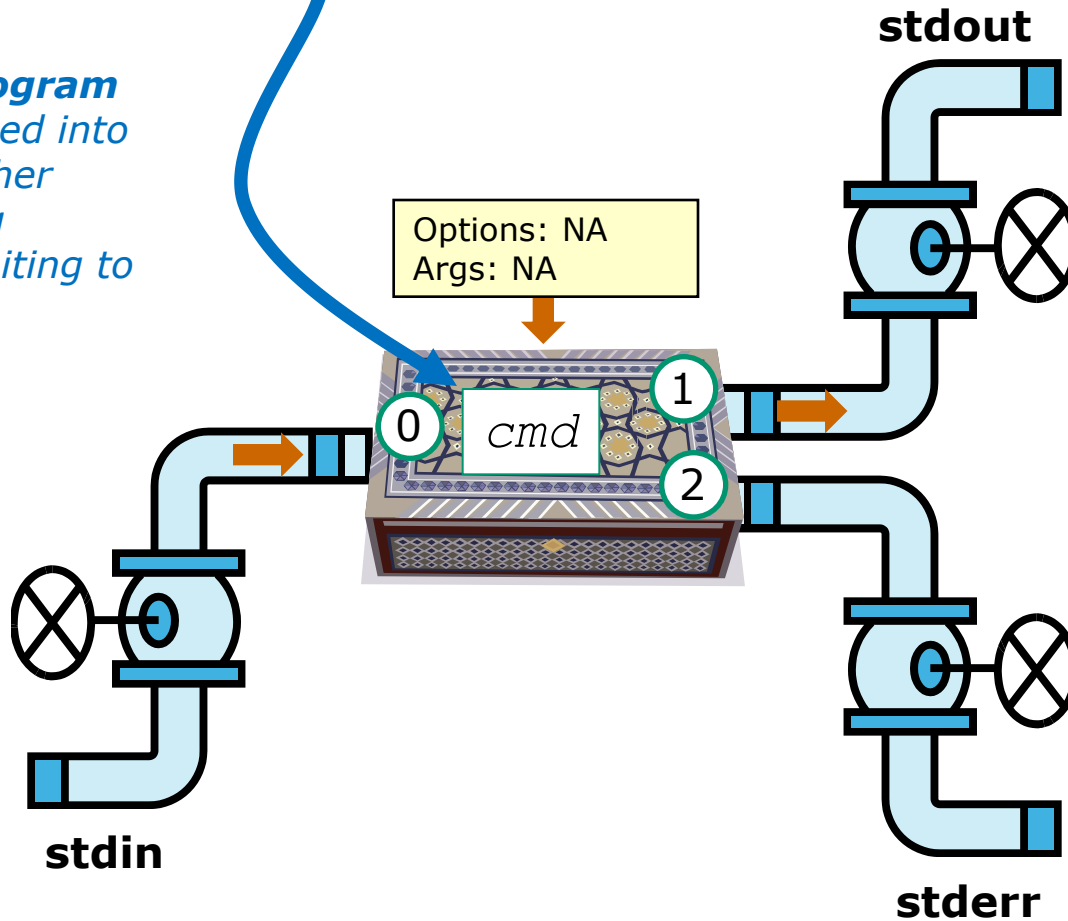
*A **process** is a **program** that has been copied (loaded) into memory by the kernel and is either running (executing instructions) or waiting to run.*



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

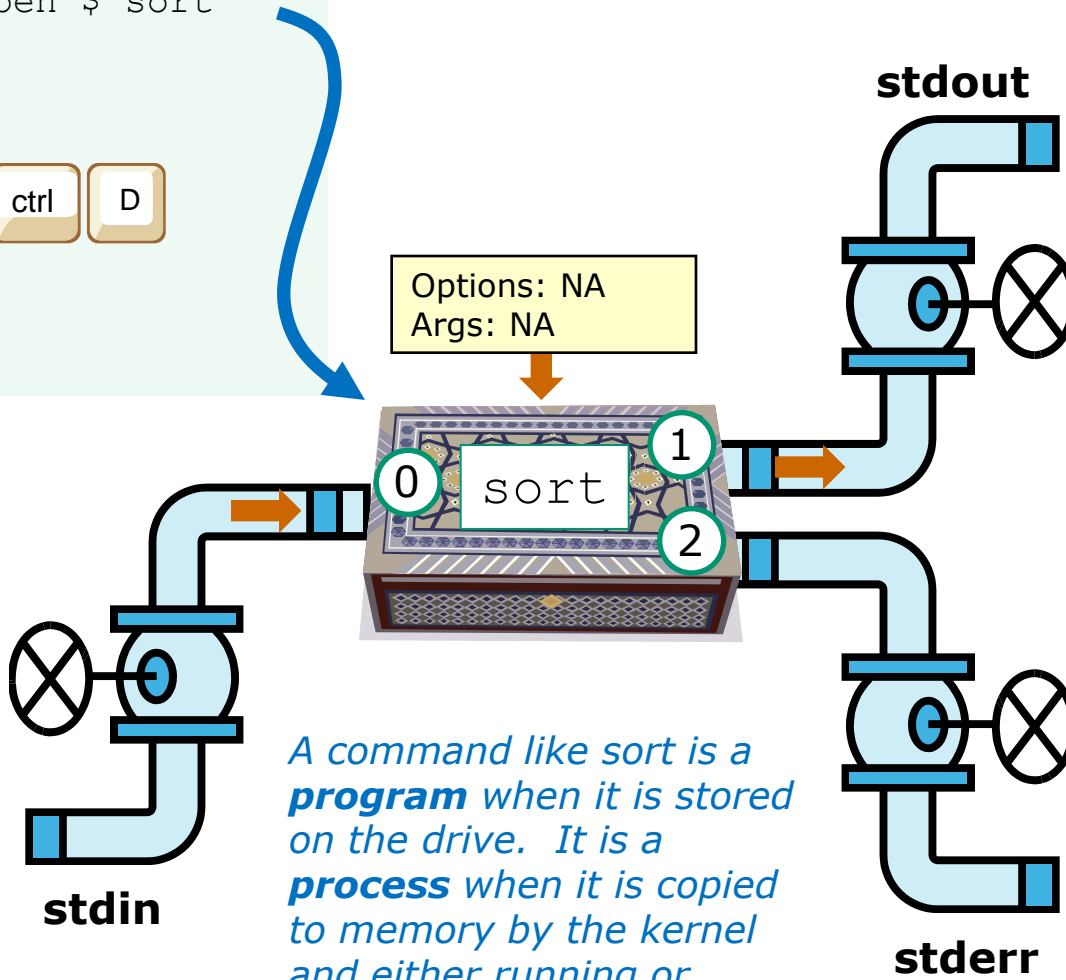
```
/home/cis90/simben $ sort  
duke  
benji  
star  
homer  
benji  
duke  
homer  
star
```



/dev/pts/0



duke
benji
star
homer



/dev/pts/0



benji
duke
homer
star

*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.*

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

Done

Most 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      Use type to find where the
sort is /bin/sort                sort program is located
```

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

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

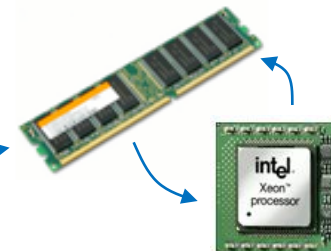
```
[rsimms@opus ~]$ xxd /bin/sort | more
```

```
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* >

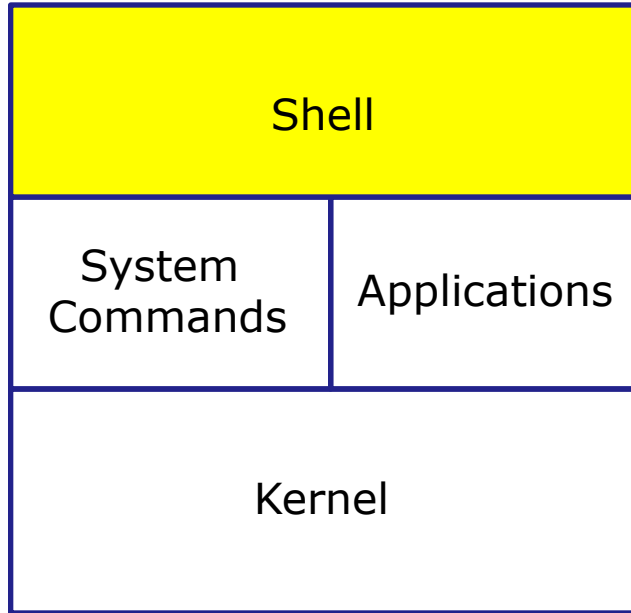
A command like **sort** is a **program**
when it is stored on the drive. It is a
process when it is copied to memory
by the kernel and either running or
waiting to run by the CPU



Process Life Cycle



The Shell **Execute** Step

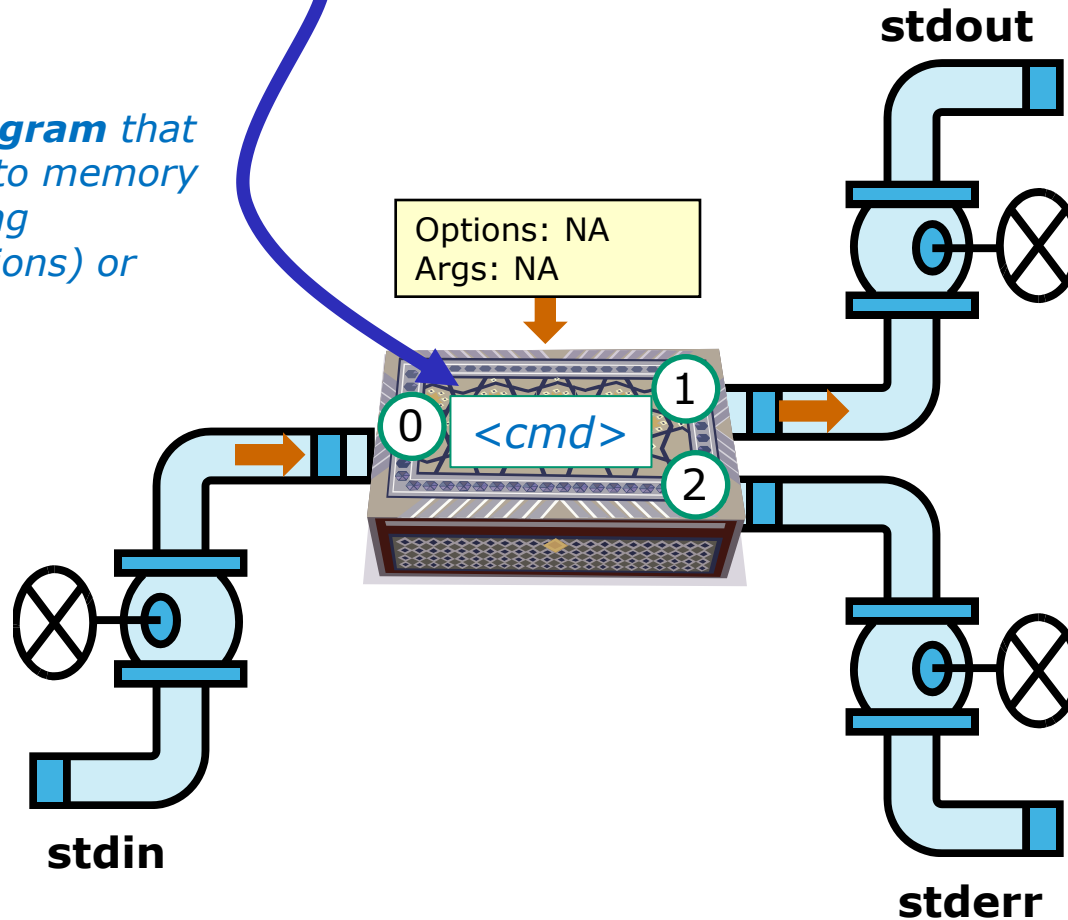


- 1) **Prompt** for a command
- 2) **Parse** (interpret metacharacters, expand file names and dissect command line into options and arguments)
- 3) **Search** for program (along the path)
- 4) **Execute** program by loading into memory (becomes a process), hookup input and outputs, and pass along command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

Executing a command <cmd>

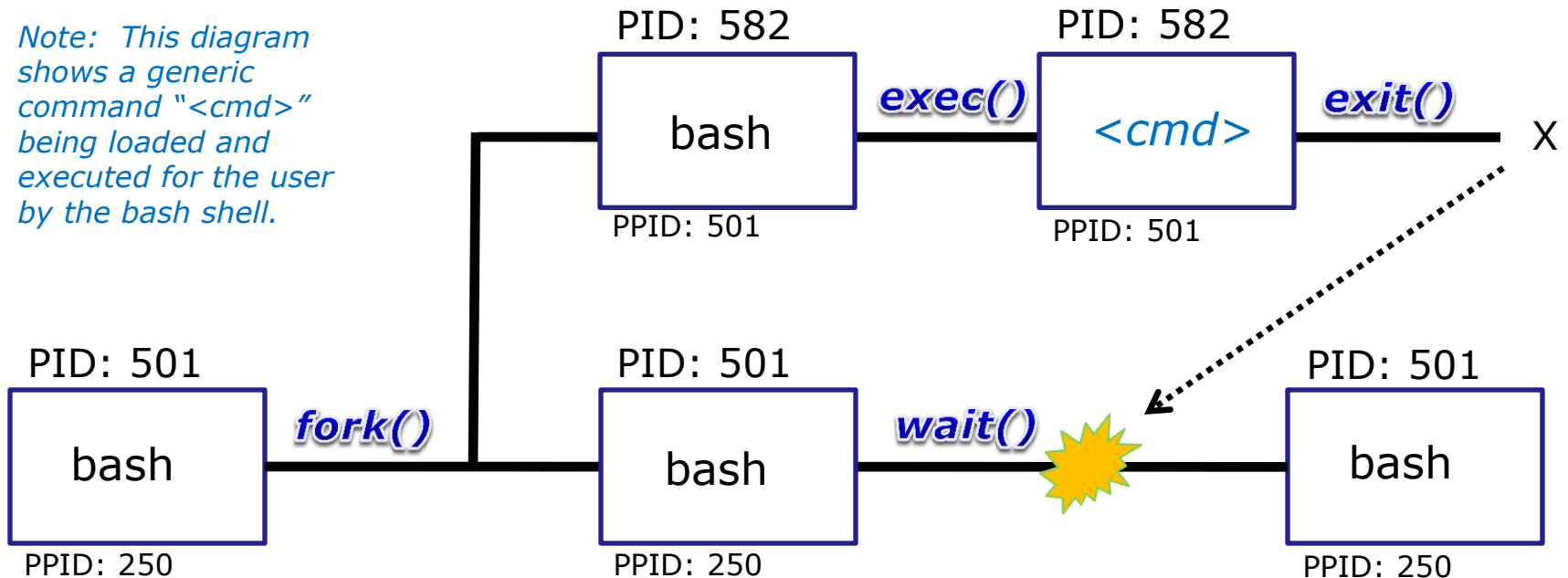
```
/home/cis90/simben $ <cmd>
```

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



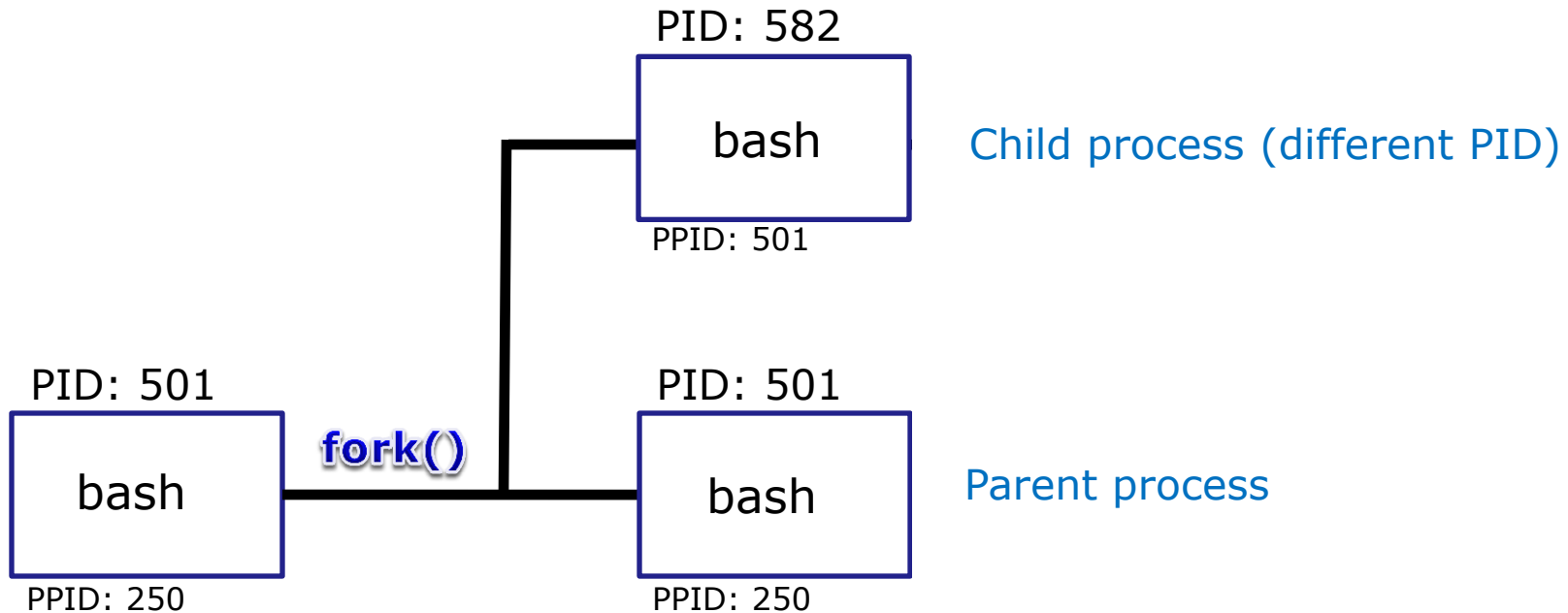
Process Lifecycle

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

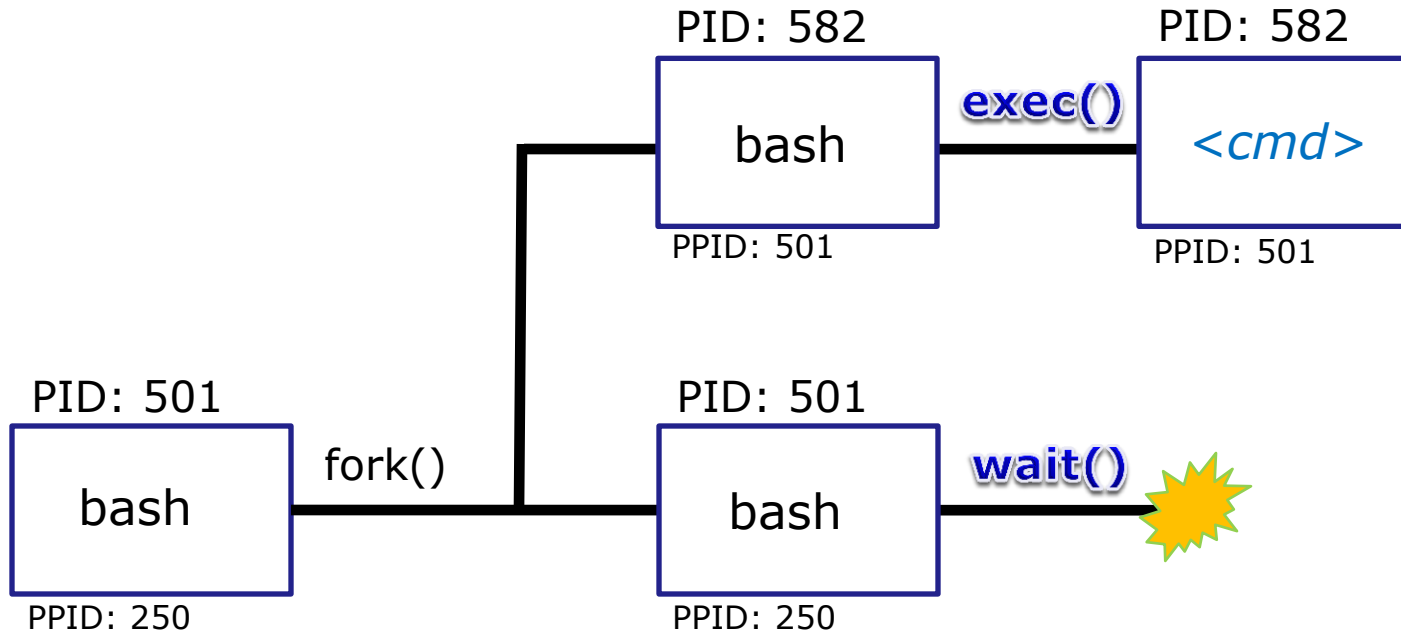
Process Lifecycle – fork child process



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.

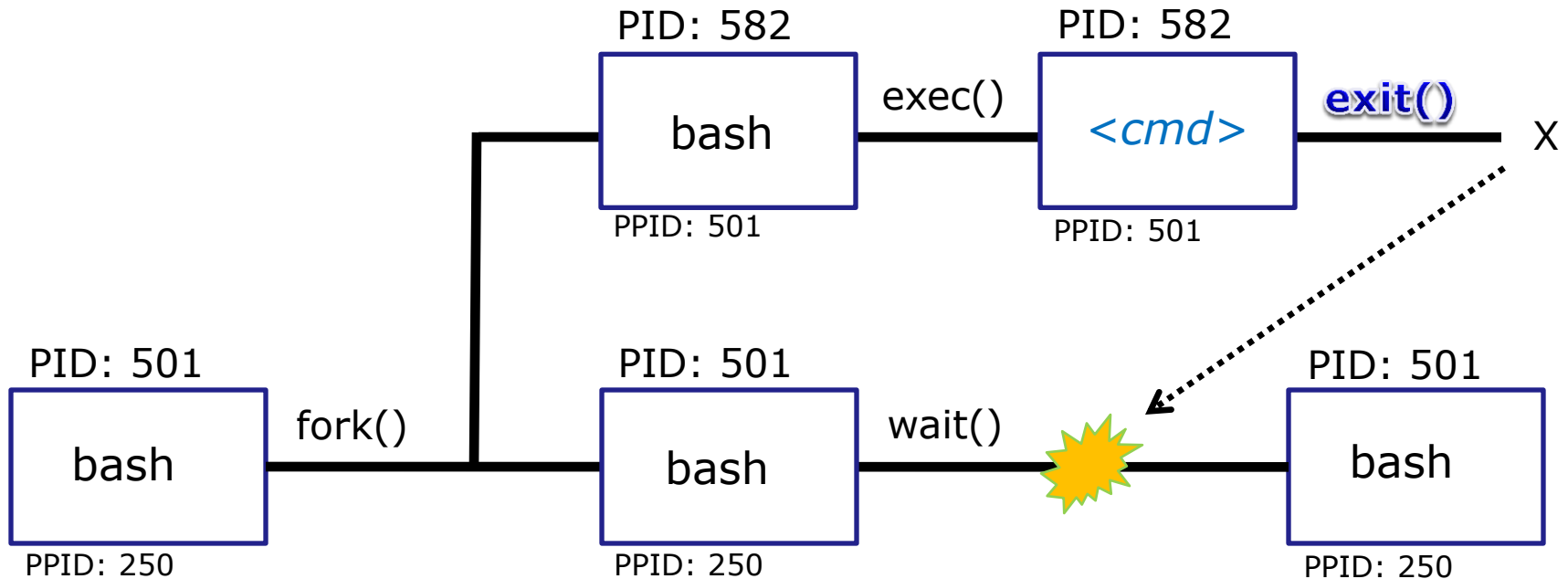
Process Lifecycle



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.

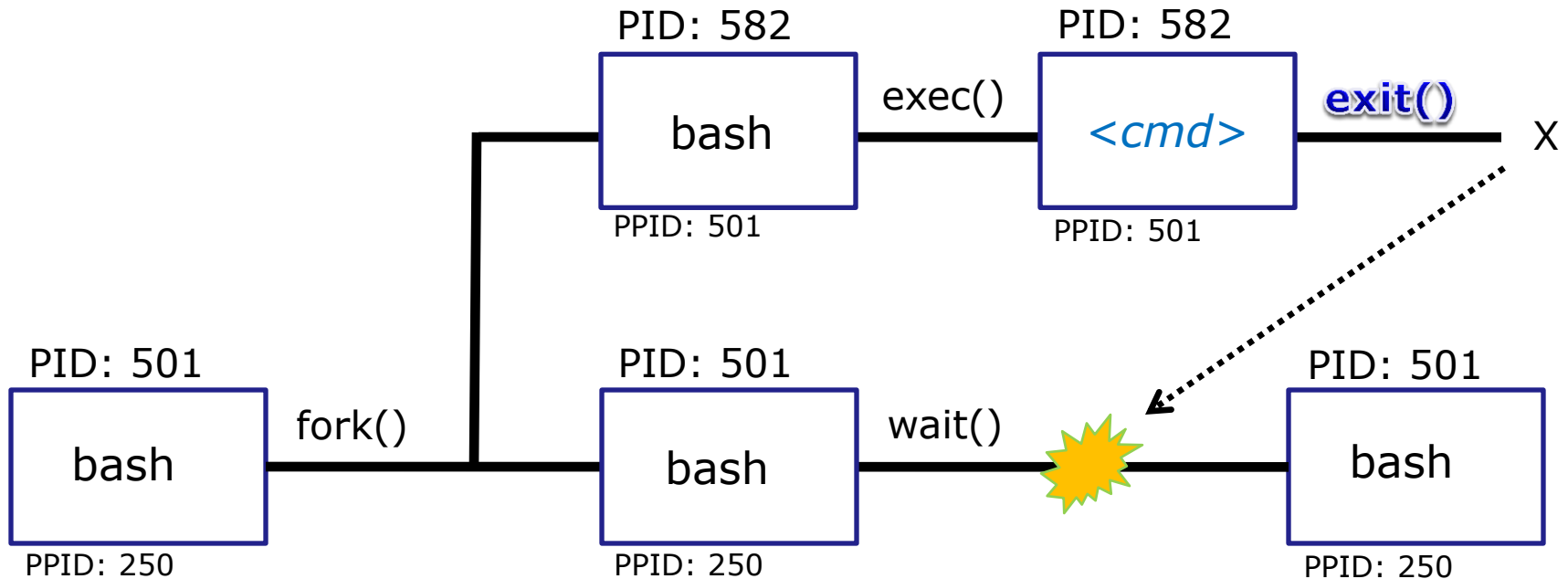
Process Lifecycle



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



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

Process Information

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

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

ps command

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

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

ps command with -u option

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

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

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

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

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

ps command with -l option

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

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

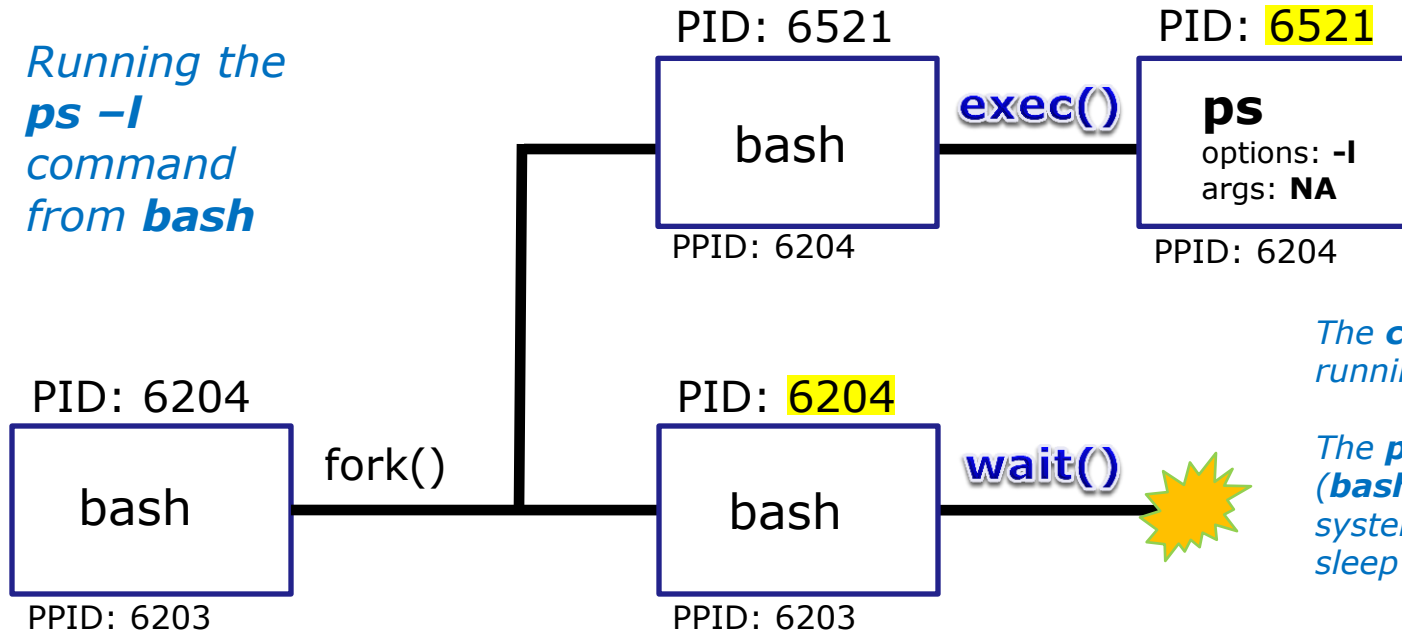
F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
0	S	201	6204	6203	0	75	0	-	1165	wait	pts/6	00:00:00	bash
0	R	201	6521	6204	0	77	0	-	1050	-	pts/6	00:00:00	ps

6204 is sleeping
6521 is running

Running or sleeping (points to S/R)
 User ID (points to UID)
 Process ID (points to PID)
 Parent Process ID (points to PPID)
 Size in 1K blocks (points to SZ)

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)

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

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

ps command with -ef options (page 2)

```

root      372      11    0 Sep10 ?           00:00:00 [ata/1]
root      373      11    0 Sep10 ?           00:00:00 [ata_aux]
root      377      11    0 Sep10 ?           00:00:00 [scsi_ah_0]
root      378      11    0 Sep10 ?           00:00:00 [scsi_ah_1]
root      379      11    0 Sep10 ?           00:01:25 [kjournald]
root      412      11    0 Sep10 ?           00:00:00 [kauditd]
root      446        1    0 Sep10 ?           00:00:00 /sbin/udev -d
root      869      11    0 Sep10 ?           00:00:01 [kedac]
root     1420      11    0 Sep10 ?           00:00:00 [kmpathd/0]
root     1421      11    0 Sep10 ?           00:00:00 [kmpathd/1]
root     2082        1    0 Sep10 ?           00:00:05 /usr/sbin/restorecond
root     2098        1    0 Sep10 ?           00:00:11 auditd
root     2100    2098    0 Sep10 ?           00:00:05 /sbin/audispd
root     2120        1    0 Sep10 ?           00:00:23 syslogd -m 0
root     2123        1    0 Sep10 ?           00:00:00 klogd -x
root     2160        1    0 Sep10 ?           00:00:20 mcstransd
rpc      2183        1    0 Sep10 ?           00:00:00 portmap
root     2201        1    0 Sep10 ?           00:01:18 /usr/bin/python -E /usr/sbin/setroub
rpcuser  2227        1    0 Sep10 ?           00:00:00 rpc.statd
root     2275        1    0 Sep10 ?           00:00:00 rpc.idmapd
root     2345        1    0 Sep10 ?           00:00:00 /usr/bin/vmnet-bridge -d /var/run/vm
root     2364        1    0 Sep10 ?           00:00:00 /usr/bin/vmnet-natd -d /var/run/vmne
dbus     2383        1    0 Sep10 ?           00:00:15 dbus-daemon --system
root     2434        1    0 Sep10 ?           00:00:51 pcsd
root     2472        1    0 Sep10 ?           00:00:00 /usr/bin/hidd --server
root     2493        1    0 Sep10 ?           00:00:02 automount

```

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)

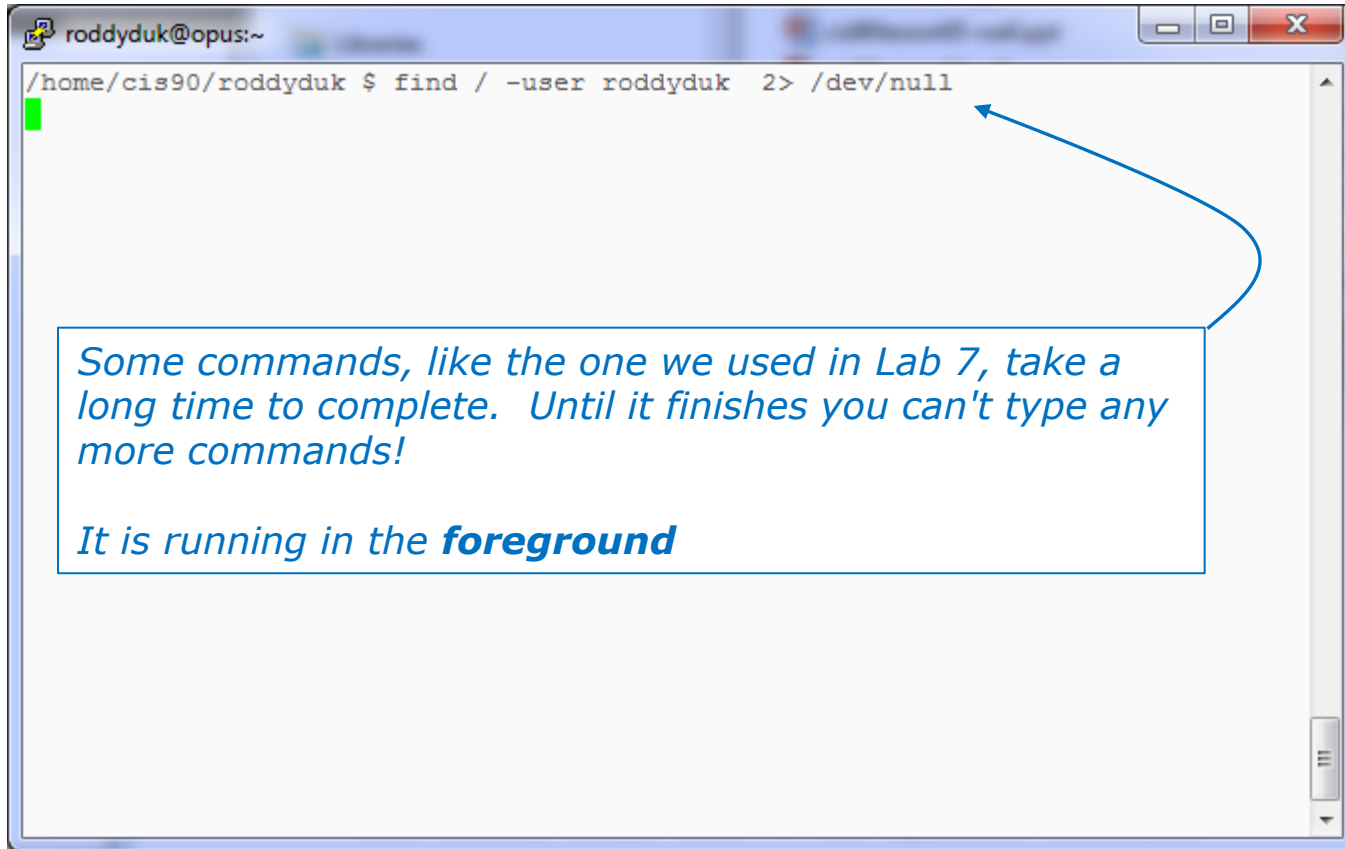
```

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

```



Job Control





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.

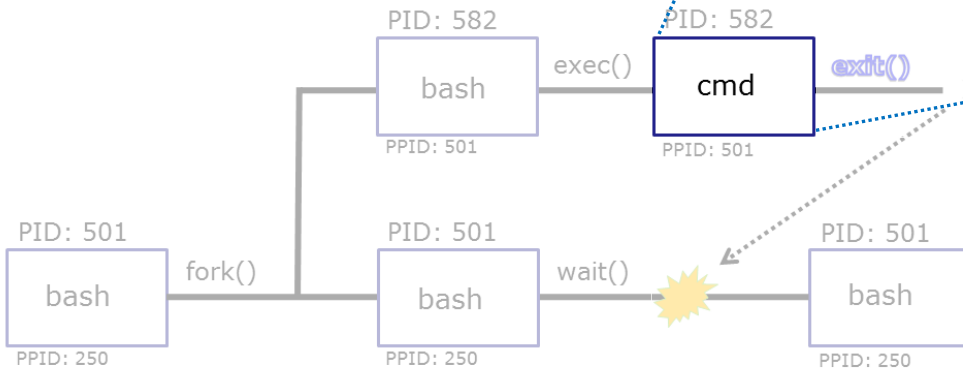
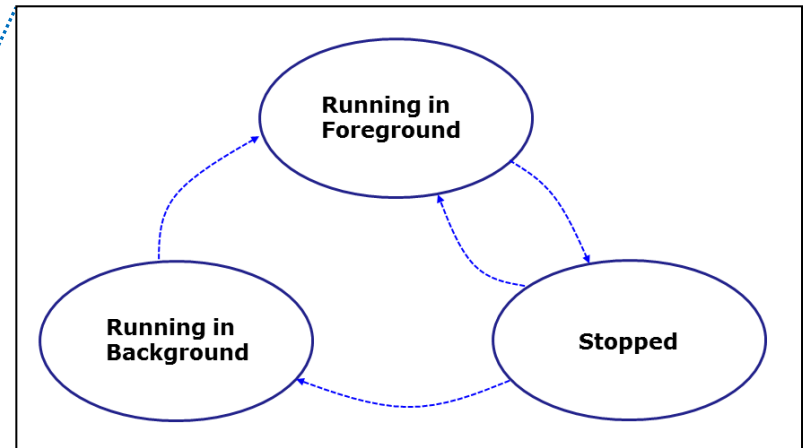
Background Processes

- 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

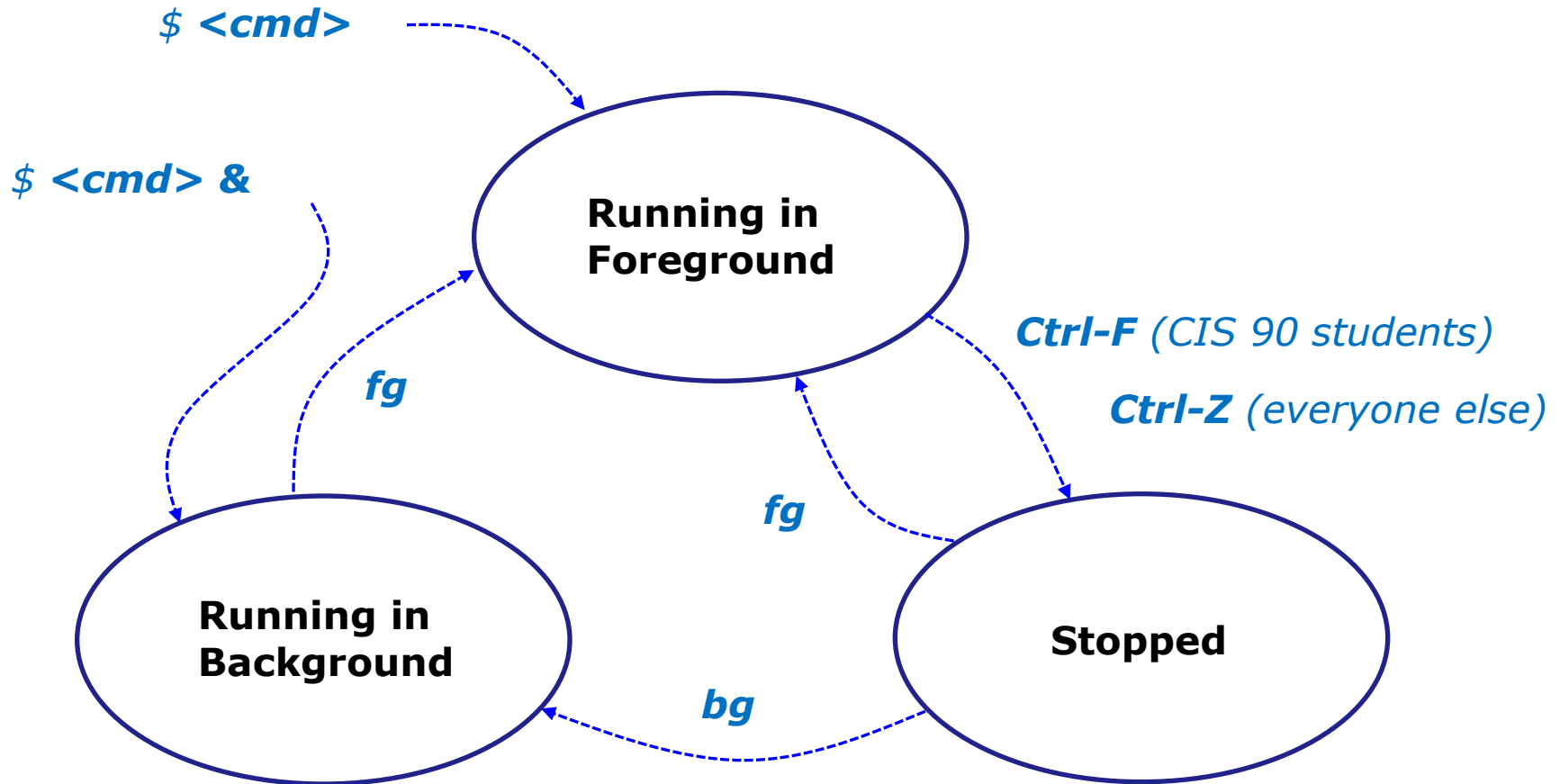
A feature of the bash shell

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

Job Control

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

Job Control

Example - suspending a **find** command

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

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

Process ID 25124 (find) is stopped (status =T)

```
[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	78	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	25124	25056	2	78	0	-	1098	finish	pts/3	00:00:00	find
0	R	201	25127	25088	0	77	0	-	1065	-	pts/4	00:00:00	ps

Job Control

Example - suspending a **find** command

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

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

bg resumes the find command in the background

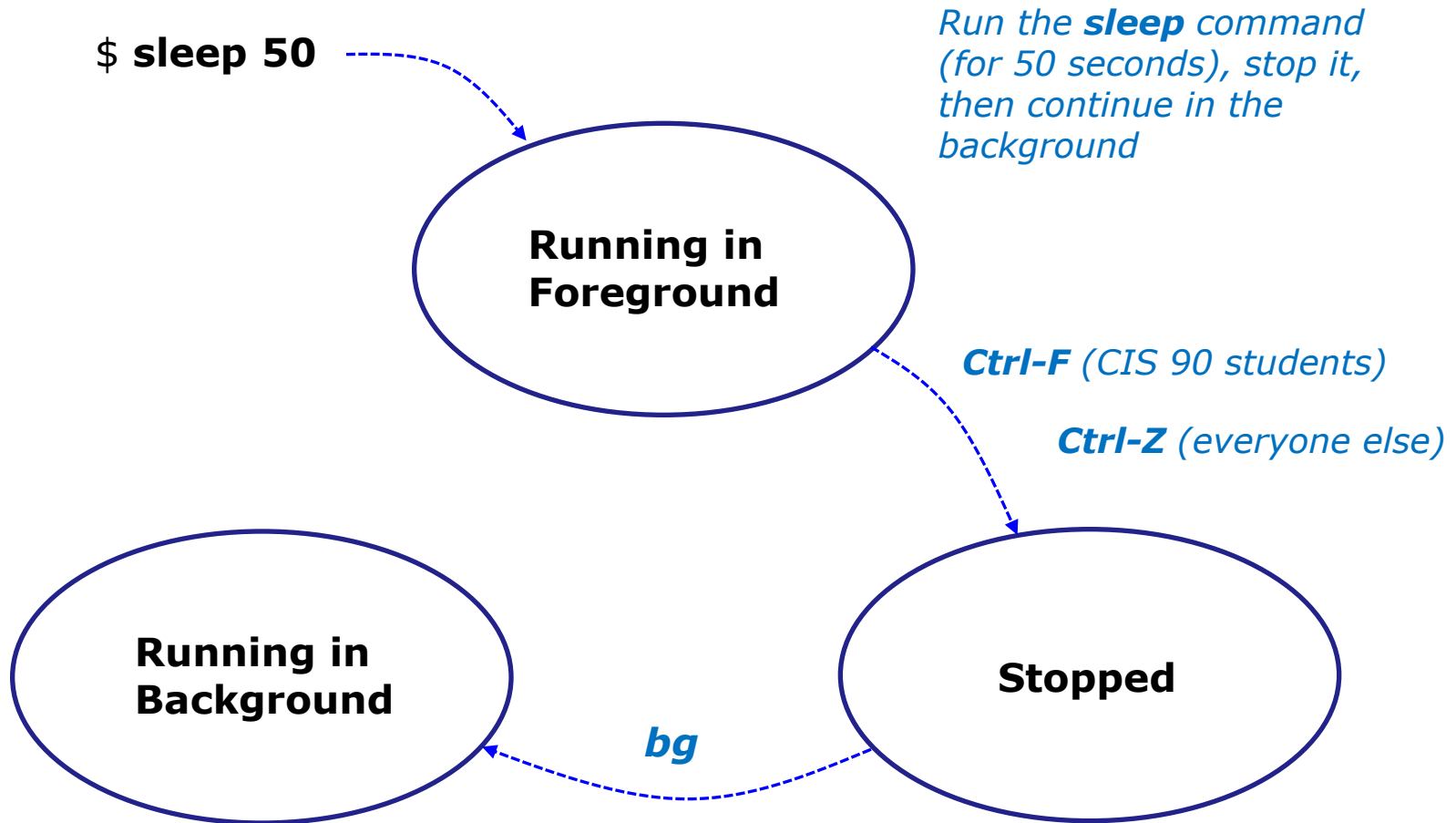
```
[rsimms@opus ~]$ ps -l -u rsimms
```

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	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	R	201	25124	25056	1	78	0	-	1099	-	pts/3	00:00:00	find
0	R	201	25129	25088	0	77	0	-	1065	-	pts/4	00:00:00	ps

*Process ID 25124
(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 ~]$ sleep 50

[1]+  Stopped                  sleep 50
[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

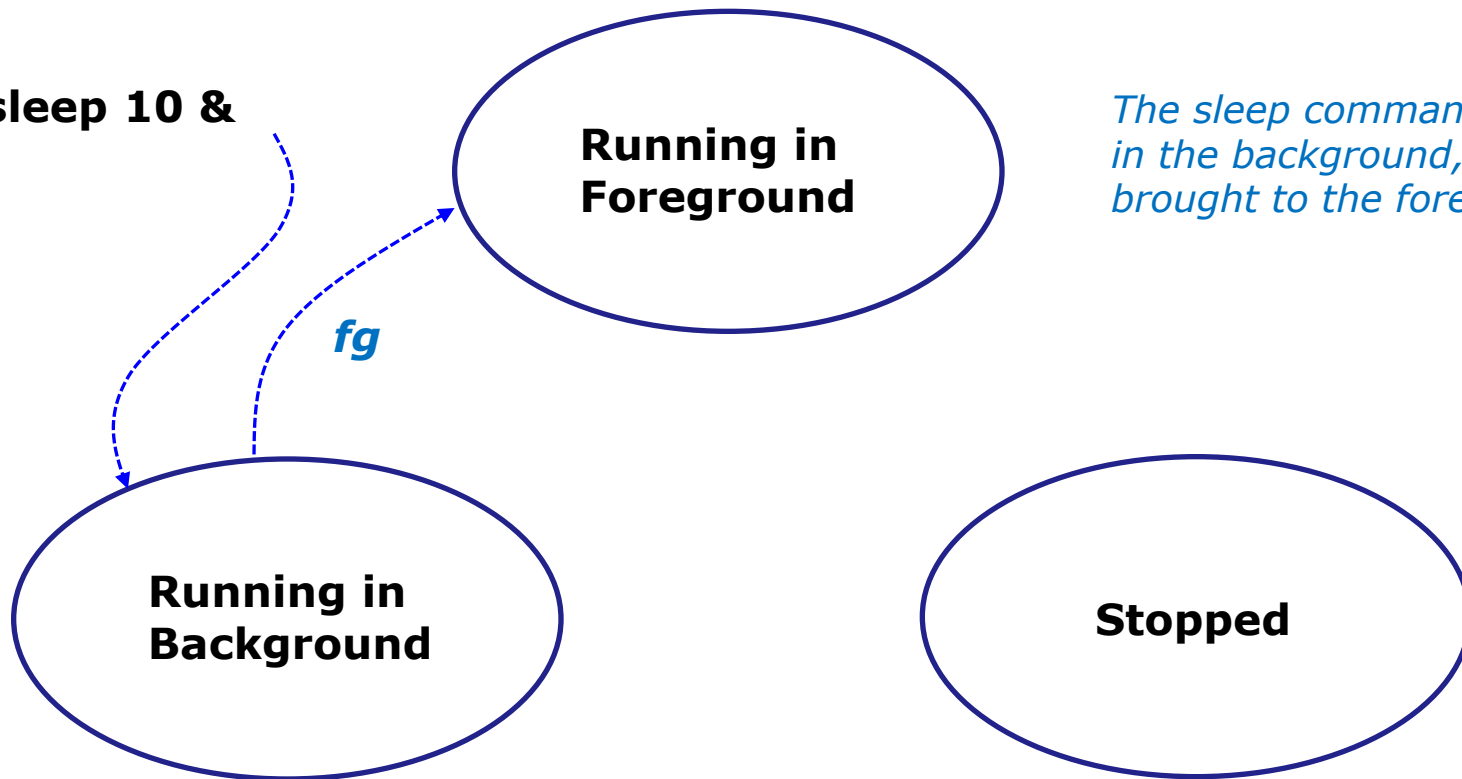
- Brings the most recent background process to the foreground

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 ~]$
```

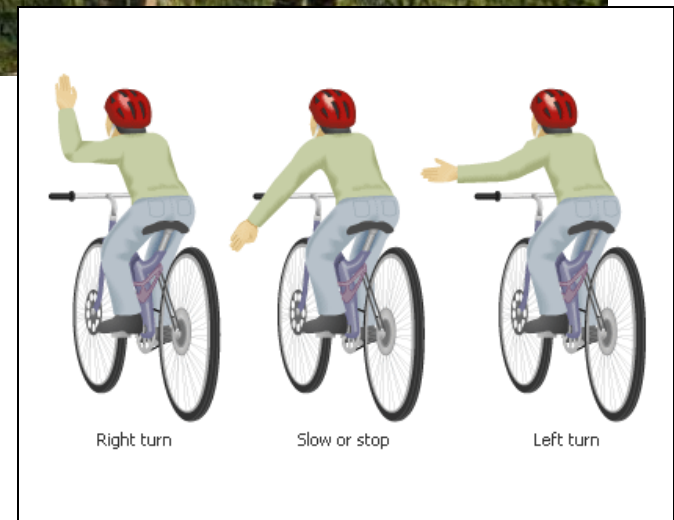
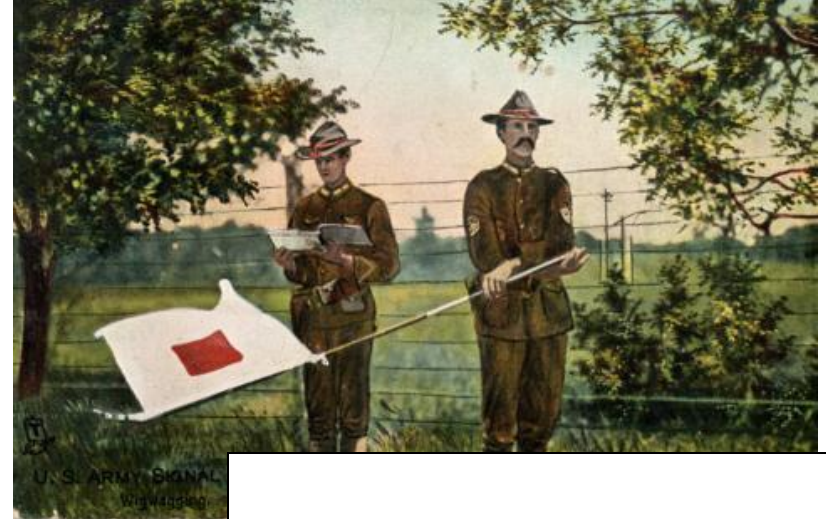
***&** is often used when running GUI tools like **firefox** or **wireshark** from the command line. This allows you to keep using the terminal for more commands while those applications run.*

Signals

Signals

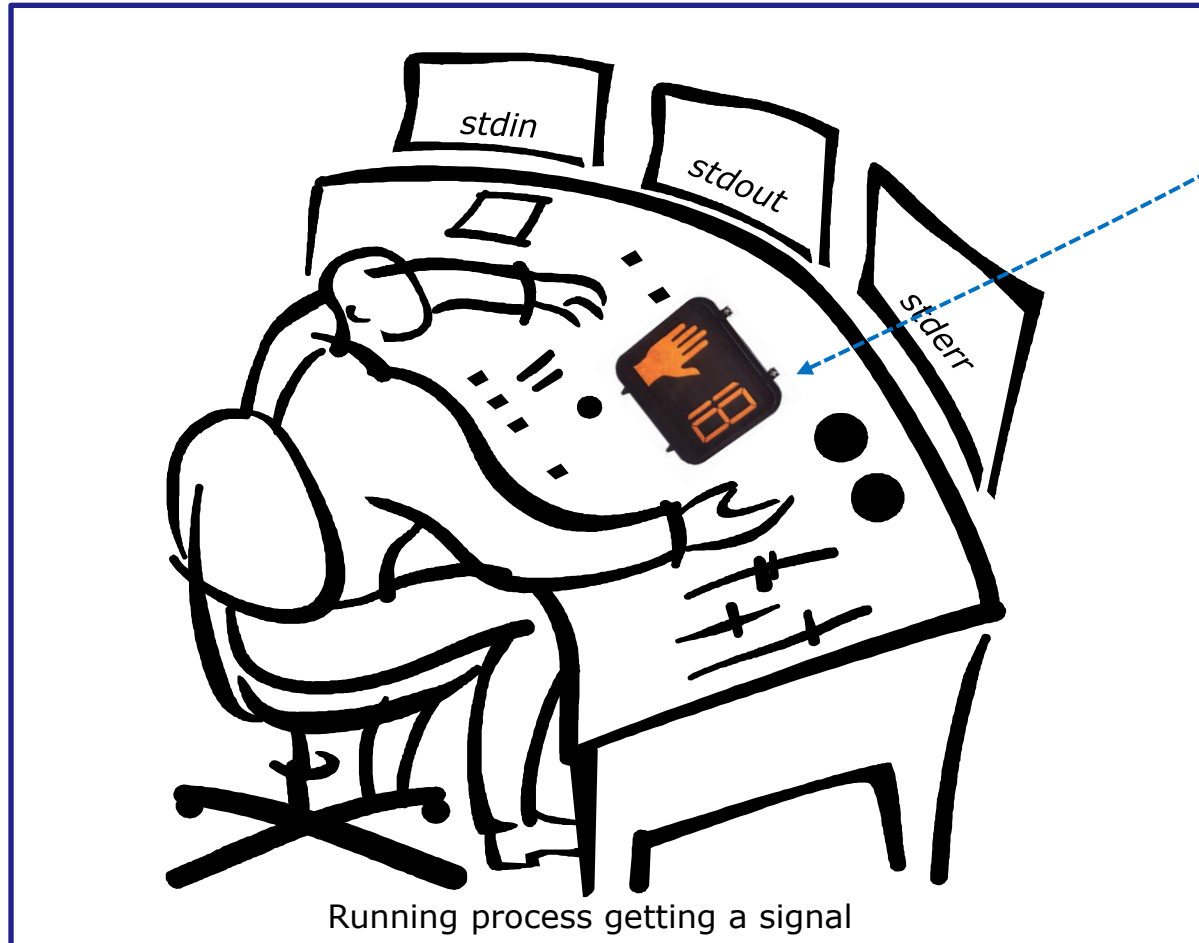
PLATE 4

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

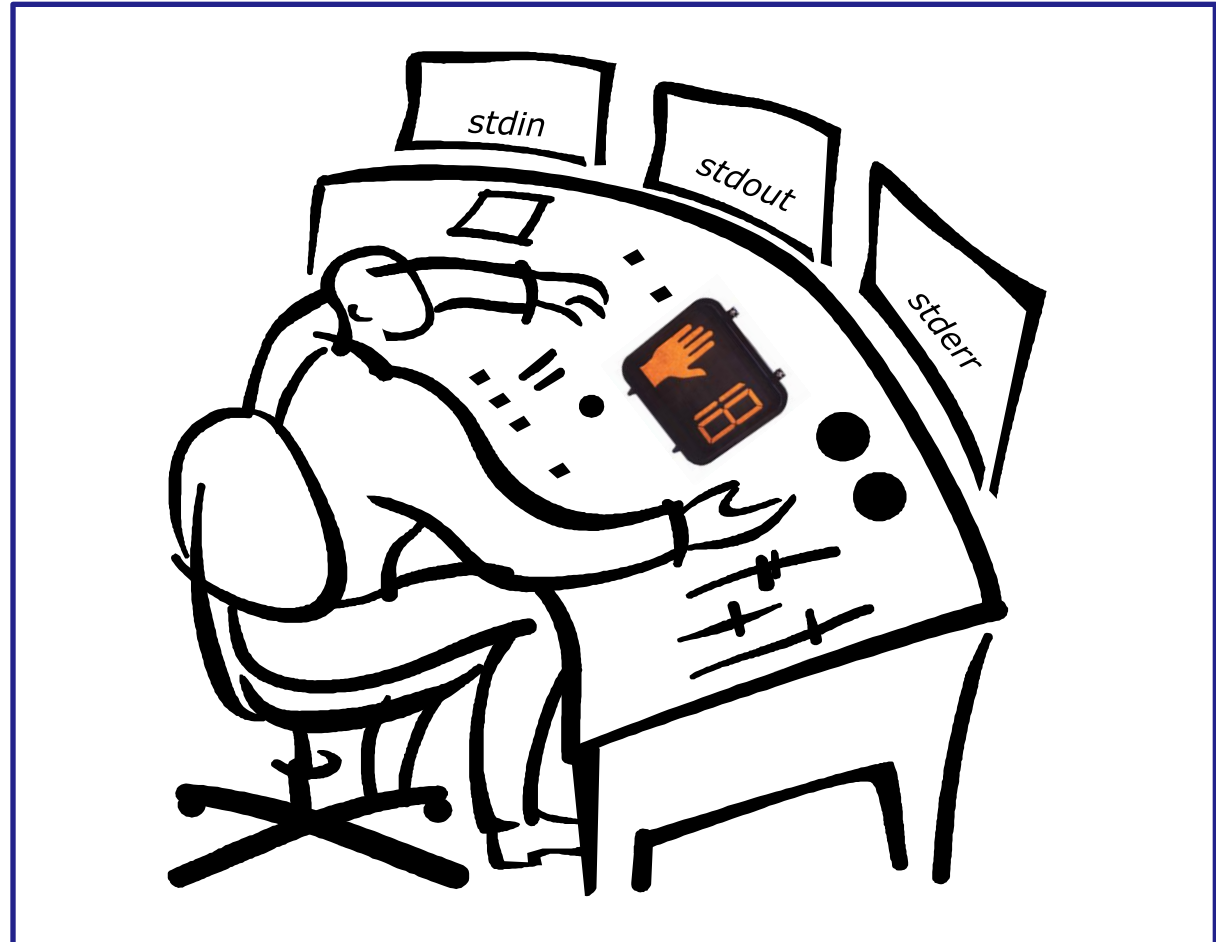
1. be ignored,
2. default action (die)
3. execute some predefined function.

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

Signals

SIGHUP	1	Hangup (POSIX)
SIGINT	2	Terminal interrupt (ANSI) Ctrl-C
SIGQUIT	3	Terminal quit (POSIX) Ctrl-\
SIGILL	4	Illegal instruction (ANSI)
SIGTRAP	5	Trace trap (POSIX)
SIGIOT	6	IOT Trap (4.2 BSD)
SIGBUS	7	BUS error (4.2 BSD)
SIGFPE	8	Floating point exception (ANSI)
SIGKILL	9	Kill (can't be caught or ignored) (POSIX)
SIGUSR1	10	User defined signal 1 (POSIX)
SIGSEGV	11	Invalid memory segment access (ANSI)
SIGUSR2	12	User defined signal 2 (POSIX)
SIGPIPE	13	Write on a pipe with no reader, Broken pipe (POSIX)
SIGALRM	14	Alarm clock (POSIX)
SIGTERM	15	Termination (ANSI) (default kill signal when not specified)

Use kill -l to see all signals

Signals

SIGSTKFLT	16	Stack fault
SIGCHLD	17	Child process has stopped or exited, changed (POSIX)
SIGCONT	18	Continue executing, if stopped (POSIX)
SIGSTOP	19	Stop executing(can't be caught or ignored) (POSIX)
SIGTSTP	20	Terminal stop signal (POSIX) Ctrl-Z or Ctrl-F
SIGTTIN	21	Background process trying to read, from TTY (POSIX)
SIGTTOU	22	Background process trying to write, to TTY (POSIX)
SIGURG	23	Urgent condition on socket (4.2 BSD)
SIGXCPU	24	CPU limit exceeded (4.2 BSD)
SIGXFSZ	25	File size limit exceeded (4.2 BSD)
SIGVTALRM	26	Virtual alarm clock (4.2 BSD)
SIGPROF	27	Profiling alarm clock (4.2 BSD)
SIGWINCH	28	Window size change (4.3 BSD, Sun)
SIGIO	29	I/O now possible (4.2 BSD)
SIGPWR	30	Power failure restart (System V)

Use kill -l to see all signals

Signals

Use **kill -l** to see all of them

```
/home/cis90/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

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

or Ctrl-\ to send a SIGQUIT/3

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 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

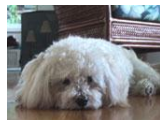
```

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
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

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

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

example 1

```
/home/cis9001/simmsben $ at 10:30pm < script_file
```

example 2

```
/home/cis9001/simmsben $ at 11:59pm  
at> cat files.out bigshell > lab08  
at> cp lab08 /home/rsimms/cis90/$LOGNAME  
at> Ctrl-D ←  
/home/cis9001/simmsben $
```

*Note: the **Ctrl-D** must be entered as the first character on the last line.*

at command scheduling examples

```
/home/cis90/rodduk $ cat job1
cp bin/myscript bin/myscript.bak
echo "Job 1 - finished, myscript has been backed up" | mail -s "Job 1" rodduk
```

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

```
/home/cis90/rodduk $ at now + 5 minutes < job1
job 24 at 2008-11-12 12:14
/home/cis90/rodduk $ at now + 2 hours < job1
job 25 at 2008-11-12 14:09
/home/cis90/rodduk $ at teatime < job1
job 26 at 2008-11-12 16:00
/home/cis90/rodduk $ at now + 1 week < job1
job 27 at 2008-11-19 12:10
/home/cis90/rodduk $ at 3:00 12/12/2010 < job1
job 28 at 2008-12-12 03:00
```

Many ways to specify a future time to run

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

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

at command management

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

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

```
25      2008-11-12 14:09 a rodduk
28      2008-12-12 03:00 a rodduk
27      2008-11-19 12:10 a rodduk
26      2008-11-12 16:00 a rodduk
24      2008-11-12 12:14 a rodduk
```

```
/home/cis90/rodduk $ atrm 24
```

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

```
25      2008-11-12 14:09 a rodduk
28      2008-12-12 03:00 a rodduk
27      2008-11-19 12:10 a rodduk
26      2008-11-12 16:00 a rodduk
/home/cis90/rodduk $
```

*The **jobs** command does not apply here. It lists processes running or suspended in the background.*

*The **atq** command lists jobs queued to run in the futures that were scheduled by at command*

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

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.cabril  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
```



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

Send a signal to a process

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?



The Test

- 10 minute break
- rocks/hiderocks T2 (sun-hwa)
- trouble-T2 (sun-hwa)
 - set permissions on activities (sun-hwa)
- trick and treats for Benji (opus)
- Add website read permission on test2



Test 2



Backup