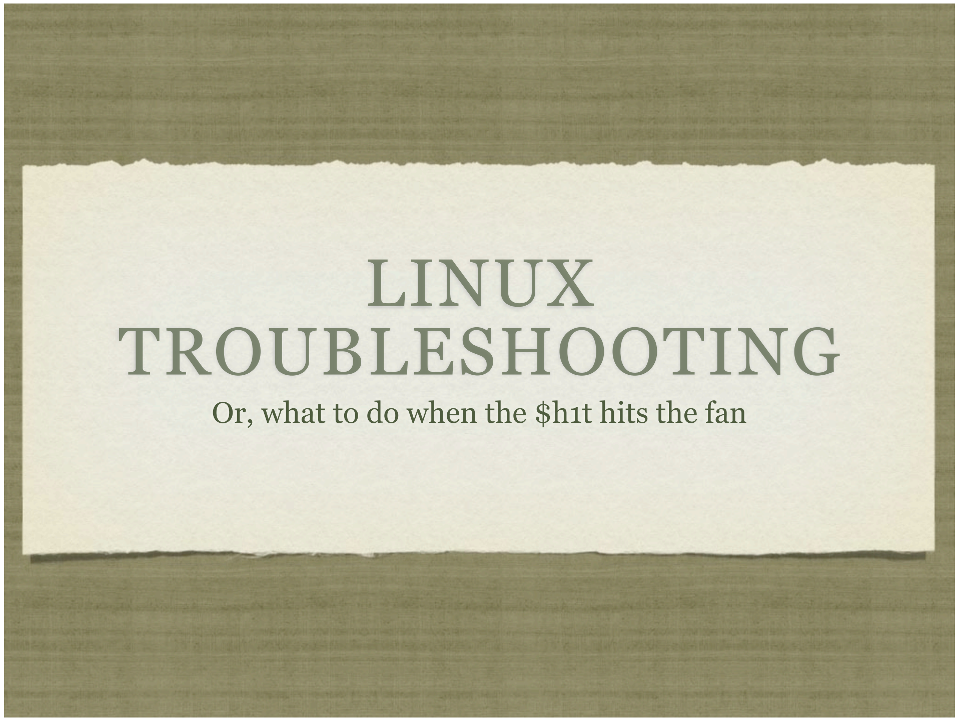
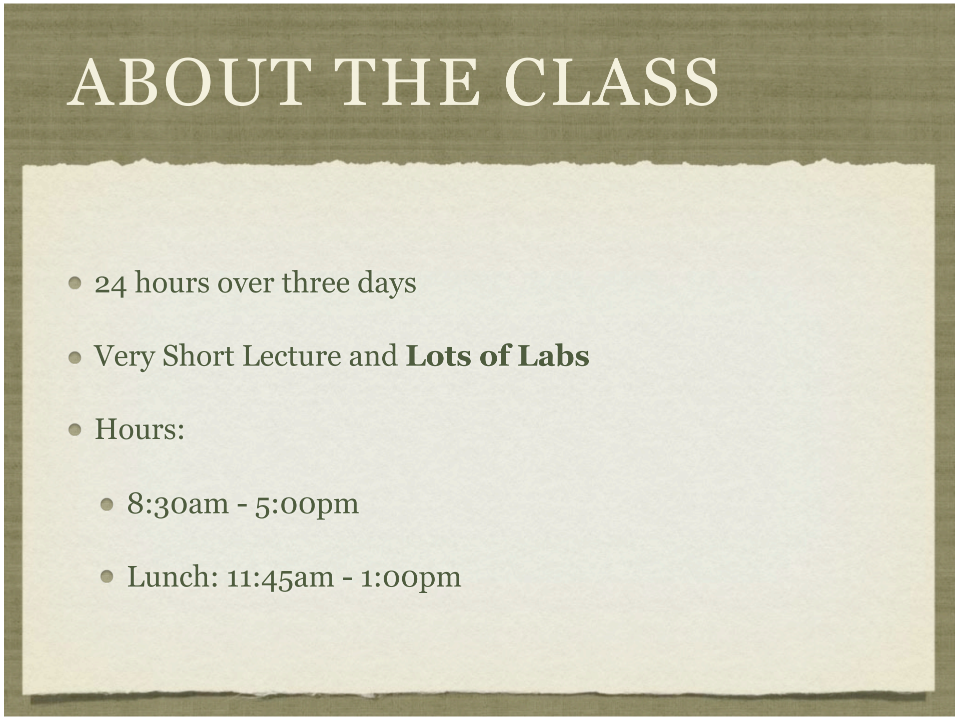


LINUX TROUBLESHOOTING

Or, what to do when the $h1t hits the fan



ABOUT THE CLASS

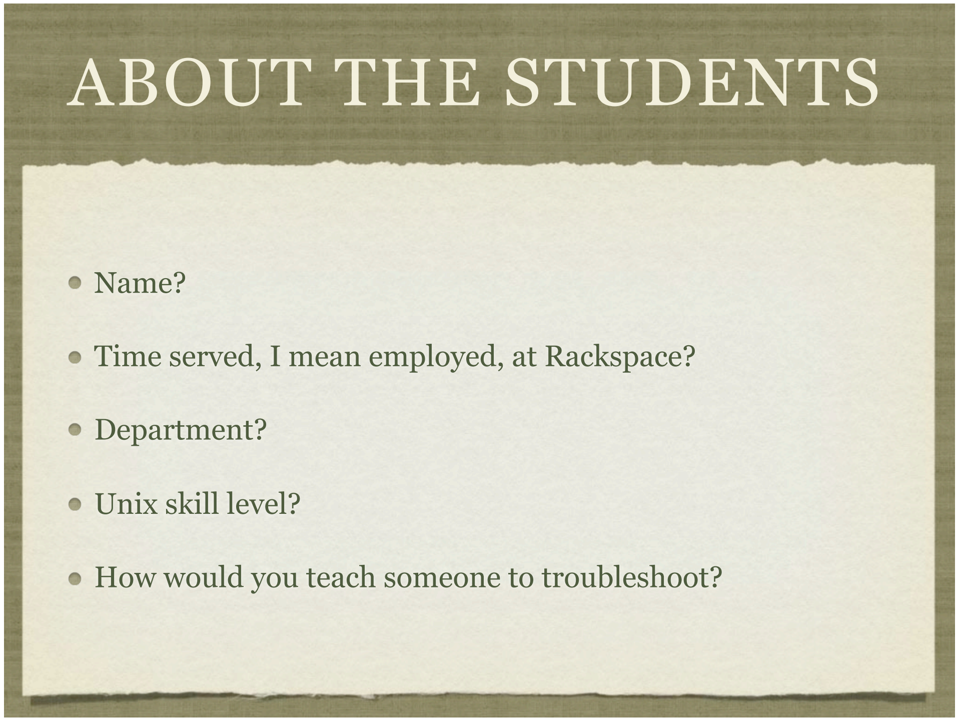
- 24 hours over three days
- Very Short Lecture and **Lots of Labs**
- Hours:
  - 8:30am - 5:00pm
  - Lunch: 11:45am - 1:00pm

ABOUT THE INSTRUCTOR

- Nathan Isburgh
  - instructor@edgecloud.com
  - Unix user 15+ years, teaching it 10+ years
  - RHCE, CISSP
  - Forgetful, goofy, patient :)

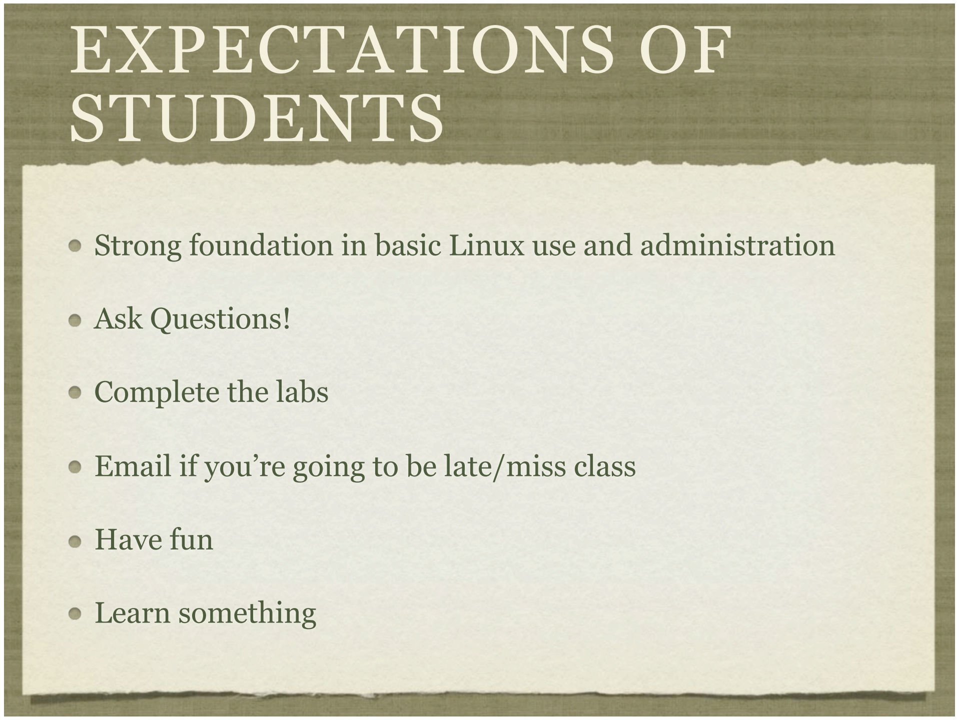
ABOUT THE COLLEGE

- Rackspace Parking Sticker = good to go
- Breaks between each lab
- Breakroom downstairs - labeled “Laundry”
- Sodas - bottles in machine ($1.25) or cans in mini-fridge ($0.50)
- Cafeteria
- Do not speed!
- No smoking anywhere. Can only smoke sitting in car.



ABOUT THE STUDENTS

- Name?
- Time served, I mean employed, at Rackspace?
- Department?
- Unix skill level?
- How would you teach someone to troubleshoot?



EXPECTATIONS OF STUDENTS

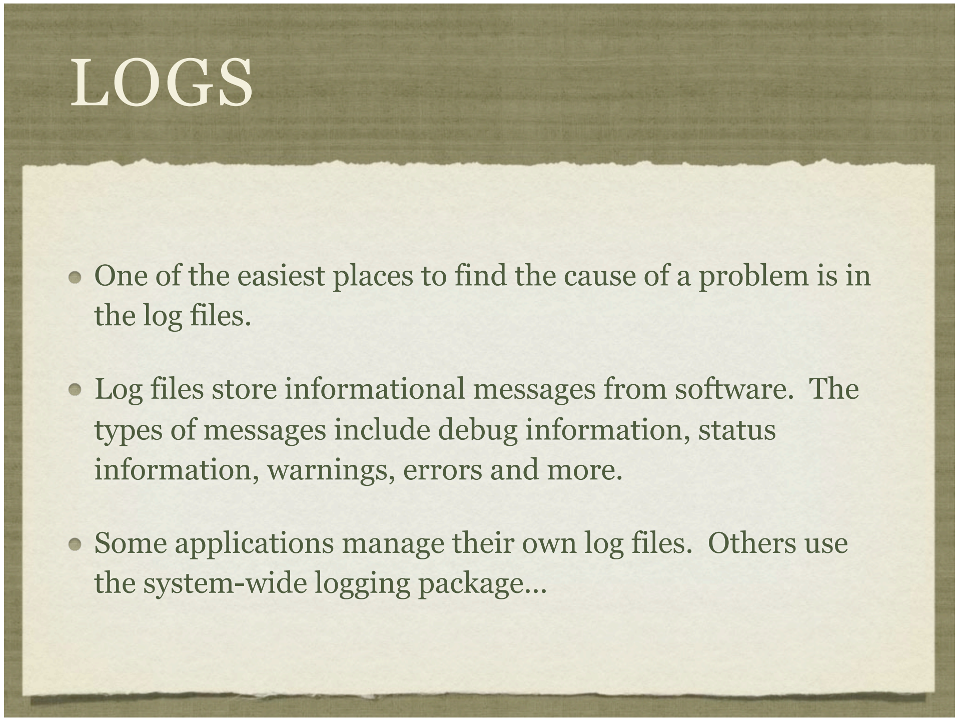
- Strong foundation in basic Linux use and administration
- Ask Questions!
- Complete the labs
- Email if you’re going to be late/miss class
- Have fun
- Learn something

OVERVIEW

- Troubleshooting is a thorough methodology used to track down the cause of problem.
- Keywords: **thorough** and **methodology**
- Without a thorough and exhaustive approach, the issue might be overlooked
- Without a strong and methodical approach, the issue may be misdiagnosed

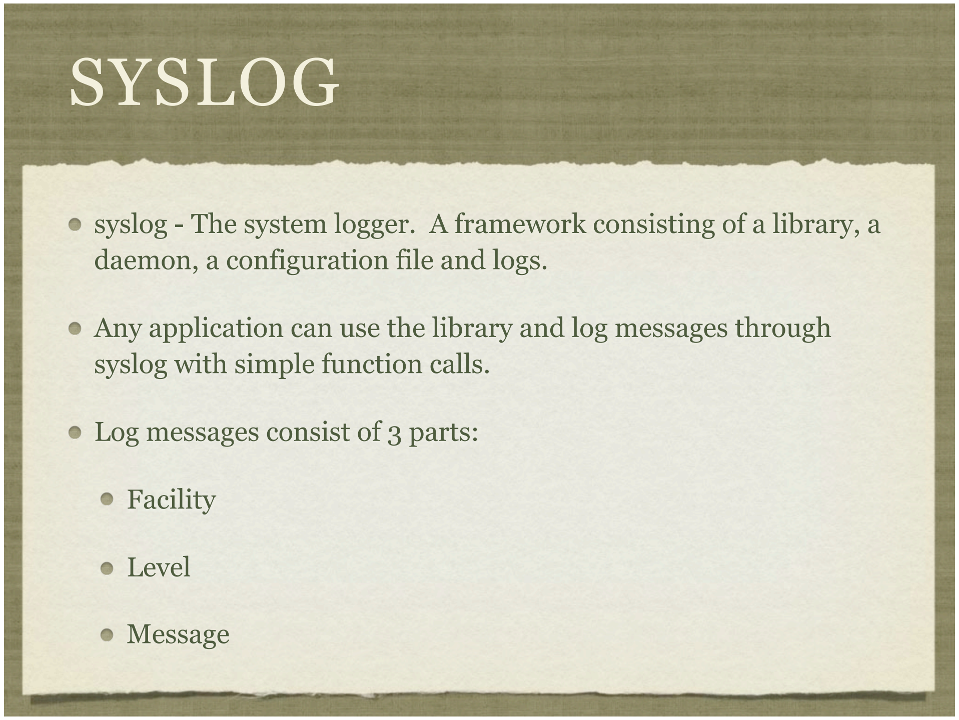
TROUBLESHOOTING KEYS

- Most Important: Only change one thing at a time
- Check #1 most likely cause: You
- Check logs for error messages
- After that, check configuration and permissions
- If all else fails, slowly, piece by piece, start removing complexity from the system to narrow down the problem area.
- DOCUMENT EVERYTHING



LOGS

- One of the easiest places to find the cause of a problem is in the log files.
- Log files store informational messages from software. The types of messages include debug information, status information, warnings, errors and more.
- Some applications manage their own log files. Others use the system-wide logging package...



SYSLOG

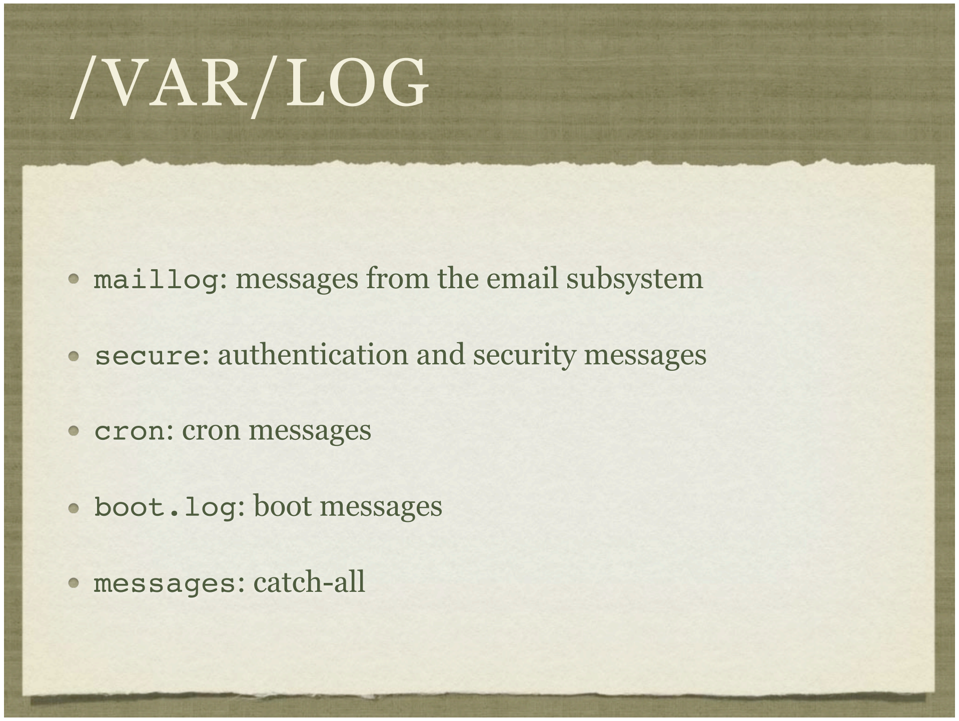
- syslog - The system logger. A framework consisting of a library, a daemon, a configuration file and logs.
- Any application can use the library and log messages through syslog with simple function calls.
- Log messages consist of 3 parts:
  - Facility
  - Level
  - Message

SYSLOG

- The facility describes what part of the operating system generated the message, and is selected by the software:
  - auth, authpriv, cron, daemon, ftp, kern, lpr, mail, news, security, syslog, user, uucp, local0-local7
- The level represents the importance of the message, and is also chosen by the software:
  - emergency, alert, critical, error, warning, notice, info, debug

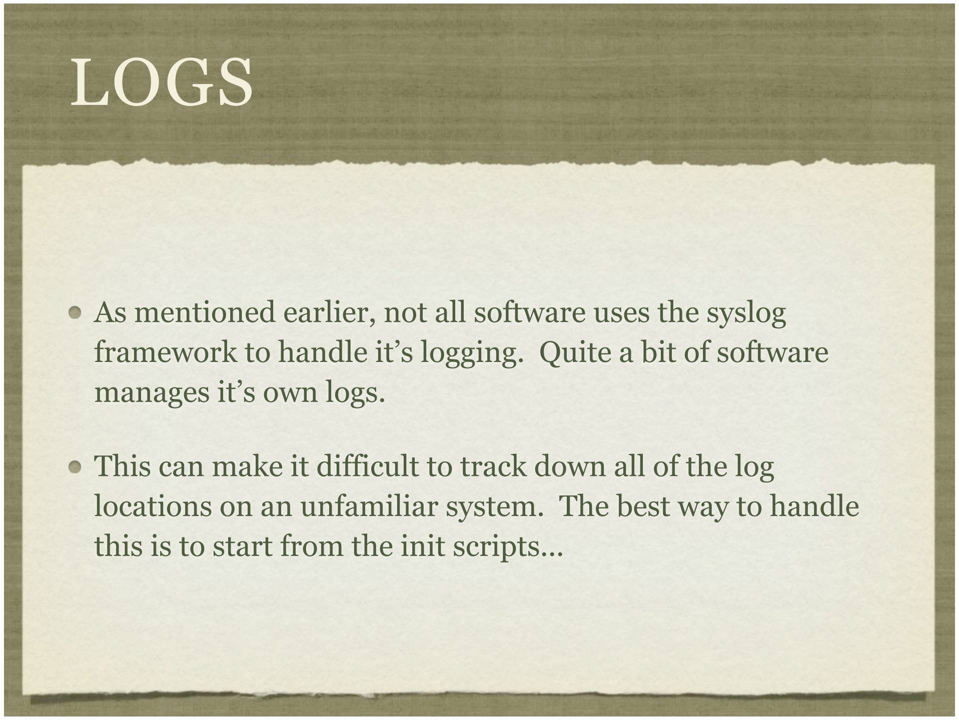
/ETC/SYSLOG.CONF

- `/etc/syslog.conf` defines where all of the log messages should go. Destinations include files, screens of logged in users, console, other syslog servers.
- Basic file format:
  - `facility.level      destination`
- Examples:
  - `*.err      /dev/console`
  - `mail.*      /var/log/maillog`
  - `*.info;mail.none;authpriv.none     /var/log/messages`



/VAR/LOG

- `maillog`: messages from the email subsystem
- `secure`: authentication and security messages
- `cron`: cron messages
- `boot.log`: boot messages
- `messages`: catch-all



LOGS

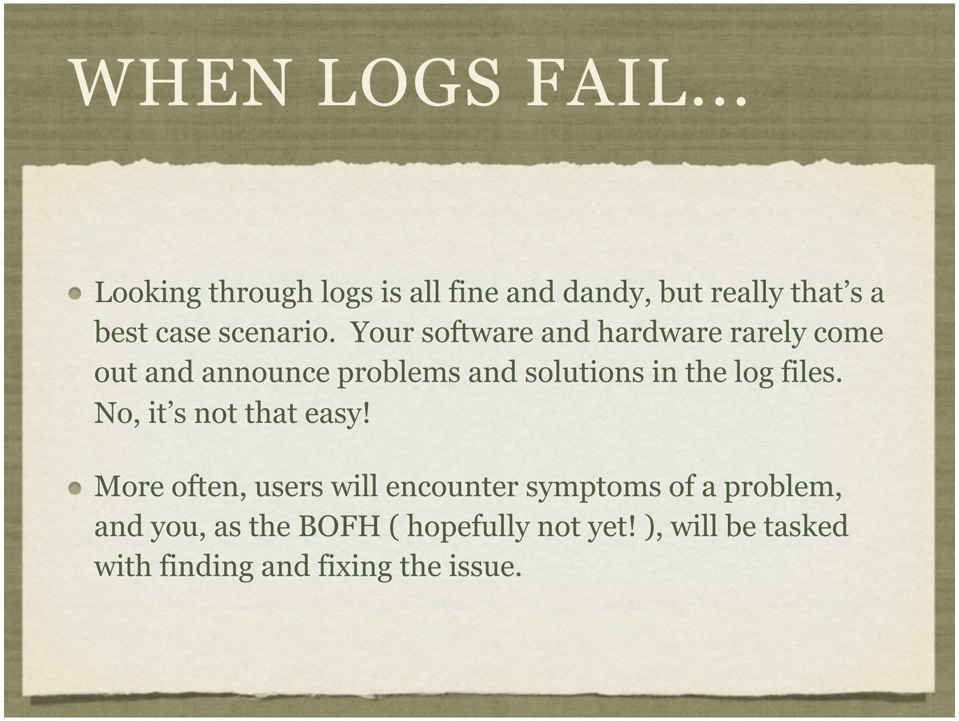
- As mentioned earlier, not all software uses the syslog framework to handle it’s logging. Quite a bit of software manages it’s own logs.
- This can make it difficult to track down all of the log locations on an unfamiliar system. The best way to handle this is to start from the init scripts...

LOCATING APPLICATION LOGS

- To track down the log file location for an application, you need to find it’s configuration file so you can see where the logs are being written.
- Of course, finding the configuration file might be just as difficult, so it’s best to start at the source.
- init starts all of the system services, and so there is an init script somewhere that is starting up the application in question.
- The init script almost always references the configuration file

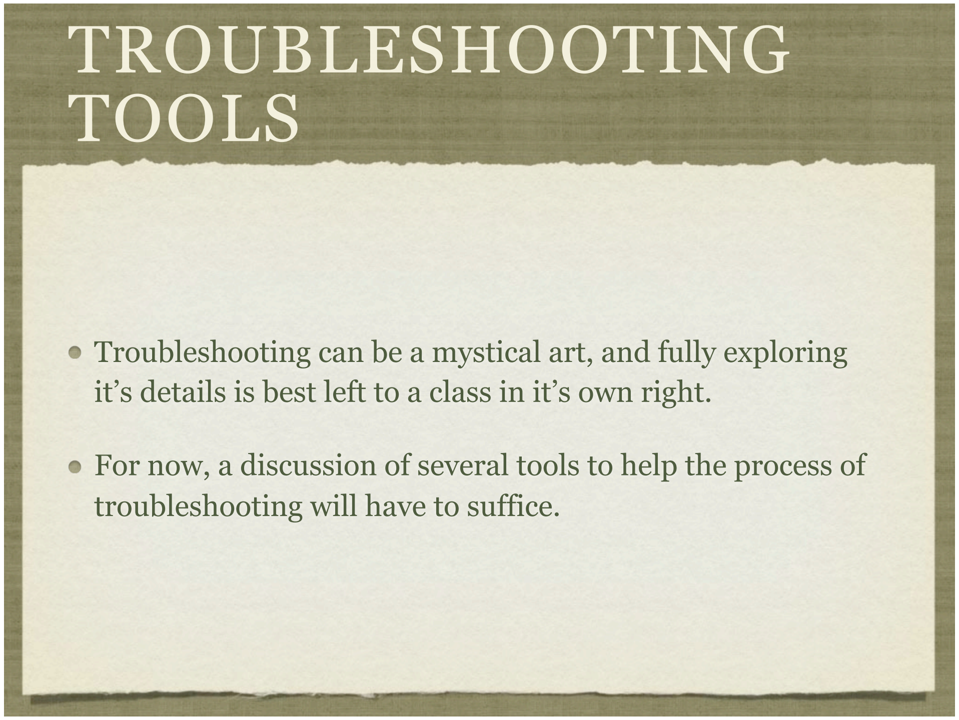
LOCATING APPLICATION LOGS

- Now that the configuration file location is known, it only takes a few moments to scan through it and find out where logs are being written.
- As for the format of the log file, that’s completely dependent on the application. Some will be similar to syslog, others, like Apache or Qmail, will be completely foreign looking.
- Fortunately, a little common sense and judicious application of Google Ointment will get the information you seek.



WHEN LOGS FAIL...

- Looking through logs is all fine and dandy, but really that's a best case scenario. Your software and hardware rarely come out and announce problems and solutions in the log files. No, it's not that easy!
- More often, users will encounter symptoms of a problem, and you, as the BOFH (hopefully not yet!), will be tasked with finding and fixing the issue.

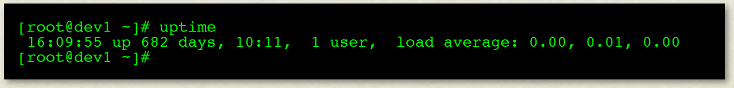


TROUBLESHOOTING TOOLS

- Troubleshooting can be a mystical art, and fully exploring it's details is best left to a class in it's own right.
- For now, a discussion of several tools to help the process of troubleshooting will have to suffice.

UPTIME

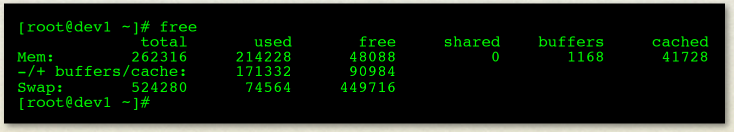
- `uptime`: Reports system uptime along with load averages.
  - Load Average: Average number of processes in run queue that are blocked.
  - `uptime` reports three values: the load averaged over the last 1 minute, 5 minutes and 15 minutes. This is useful to get an idea of the load trend on the system.
  - Example:



```
[root@dev1 ~]# uptime
 16:09:55 up 682 days, 10:11,  1 user,  load average: 0.00, 0.01, 0.00
[root@dev1 ~]#
```

FREE

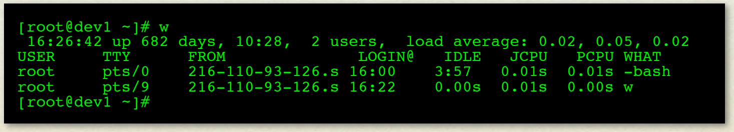
- `free`: reports on memory and swap usage
  - buffers: I/O buffers, directory cache
  - cached: filesystem cache (data)
  - Example:



```
[root@dev1 ~]# free
             total       used       free     shared    buffers     cached
Mem:        262316     214228      48088          0       1168      41728
-/+ buffers/cache:     171332      90984
Swap:       524280      74564     449716
[root@dev1 ~]#
```

W

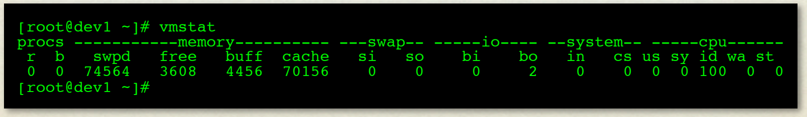
- w: Displays an uptime report, followed by a breakdown of all logged-in users and what process they are running
    - JCPU: Combined CPU time of all processes attached to the terminal (foreground and background)
    - PCPU: CPU time of foreground process, listed in “what” column
    - Example:



```
[root@dev1 ~]# w
 16:26:42 up 682 days, 10:28,  2 users,  load average: 0.02, 0.05, 0.02
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0    216-110-93-126.s 16:00    3:57   0.01s  0.01s -bash
root     pts/9    216-110-93-126.s 16:22    0.00s  0.01s  0.00s w
[root@dev1 ~]#
```

VMSTAT

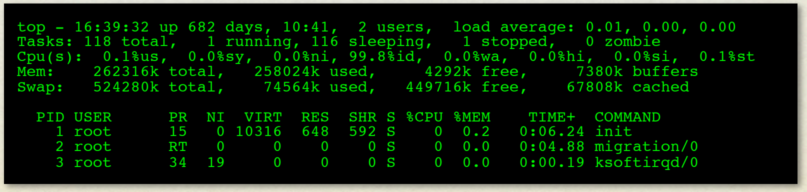
- `vmstat`: Snapshot report covering several primary statistics.
    - procs: number of running and blocked processes
    - swap: swapped in and swapped out blocks of memory, per second
    - io: blocks in and blocks out read/written per second
    - system: interrupts and context switches per second
    - cpu: user, system, idle, wait and time-stolen from a VM



```
[root@dev1 ~]# vmstat
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu------
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0  74564   3608   4456  70156    0    0     0     2    0    0  0  0 100  0  0
[root@dev1 ~]#
```

TOP

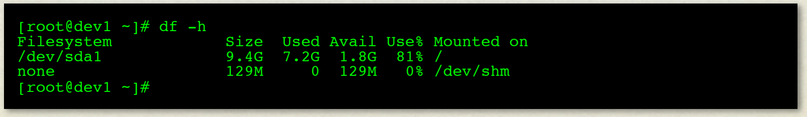
- `top`: Self-updating tool displays combination summary at top, followed by ordered list of processes. Fully customizable.
  - The summary includes uptime information, memory breakdowns, CPU utilization and process state summaries
  - The process display can be customized and sorted to suit need



```
top - 16:39:32 up 682 days, 10:41,  2 users,  load average: 0.01, 0.00, 0.00
Tasks: 118 total,   1 running, 116 sleeping,   1 stopped,   0 zombie
Cpu(s):  0.1%us,  0.0%sy,  0.0%ni, 99.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.1%st
Mem:    262316k total,   258024k used,     4292k free,     7380k buffers
Swap:   524280k total,    74564k used,   449716k free,    67808k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root      15   0 10316  648  592 S    0  0.2   0:06.24 init
    2 root      RT   0     0    0    0 S    0  0.0   0:04.88 migration/0
    3 root      34  19     0    0    0 S    0  0.0   0:00.19 ksoftirqd/0
```

DF

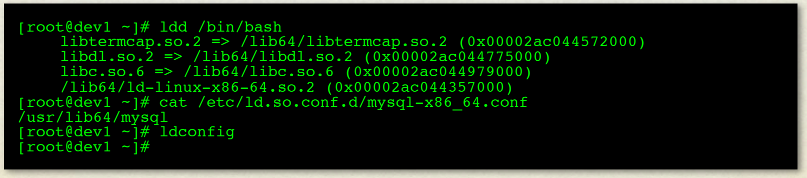
- `df`: lists filesystem utilization
  - Breaks down size and use information for each mounted filesystem
  - -h is useful option to display in “human-friendly” format



```
[root@dev1 ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda1             9.4G  7.2G  1.8G  81% /
none                  129M     0  129M   0% /dev/shm
[root@dev1 ~]#
```

LDD, LDCONFIG

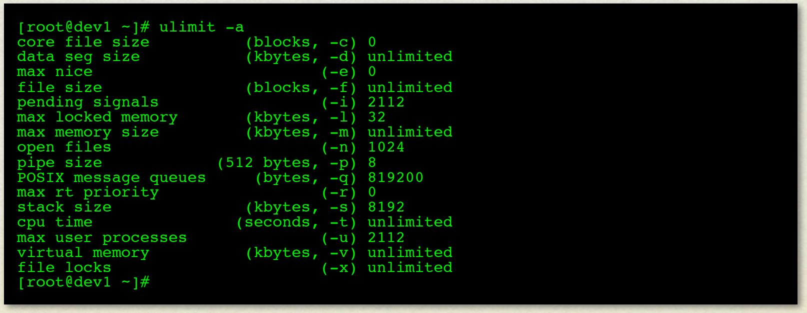
- `ldd`: List library dependencies
- `ldconfig`: Update library location database
  - `/etc/ld.so.conf` and `/etc/ld.so.conf.d/*.conf` for list of pathnames to search for libraries, creates database for dynamic linker



```
[root@dev1 ~]# ldd /bin/bash
     libtermcap.so.2 => /lib64/libtermcap.so.2 (0x00002ac044572000)
     libdl.so.2 => /lib64/libdl.so.2 (0x00002ac044775000)
     libc.so.6 => /lib64/libc.so.6 (0x00002ac044979000)
     /lib64/ld-linux-x86-64.so.2 (0x00002ac044357000)
[root@dev1 ~]# cat /etc/ld.so.conf.d/mysql-x86_64.conf
/usr/lib64/mysql
[root@dev1 ~]# ldconfig
[root@dev1 ~]#
```

ULIMIT

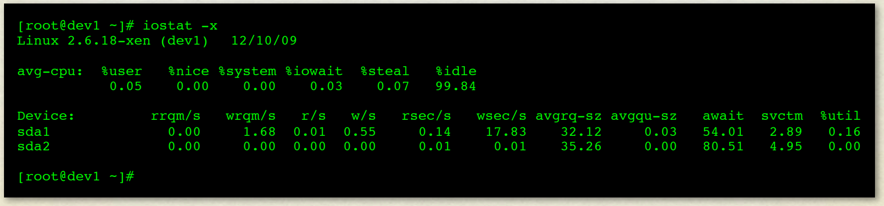
- ulimit: Sets resource limits
  - Can limit open files, memory use, cpu time, subprocesses and more.



```
[root@dev1 ~]# ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
max nice                        (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 2112
max locked memory       (kbytes, -l) 32
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
max rt priority                 (-r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 2112
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
[root@dev1 ~]#
```

IOSTAT

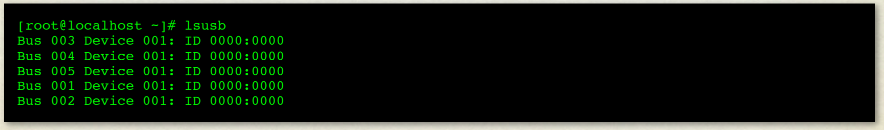
- `iostat`: IO statistics report
  - Part of the `sysstat` package; not always installed
  - Allows for drilldown into the IO system to view real time metrics on IO operations per filesystem



```
[root@dev1 ~]# iostat -x
Linux 2.6.18-xen (dev1)   12/10/09

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.05    0.00    0.00    0.03    0.07   99.84

Device:         rrqm/s   wrqm/s   r/s   w/s   rsec/s   wsec/s avgrq-sz avgqu-sz   await  svctm  %util
sda1              0.00     1.68  0.01  0.55     0.14    17.83    32.12     0.03   54.01   2.89   0.16
sda2              0.00     0.00  0.00  0.00     0.01     0.01    35.26     0.00   80.51   4.95   0.00

[root@dev1 ~]#
```

LSUSB

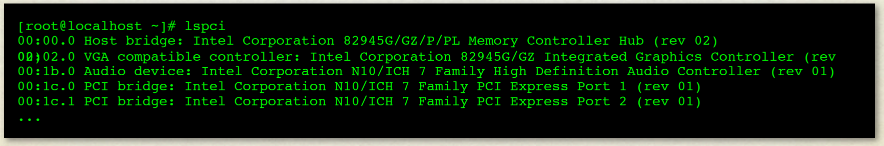
- `lsusb`: List USB bus
  - Generates a listing of devices on the USB bus
  - Consider `-v` option for *detailed* information



```
[root@localhost ~]# lsusb
Bus 003 Device 001: ID 0000:0000
Bus 004 Device 001: ID 0000:0000
Bus 005 Device 001: ID 0000:0000
Bus 001 Device 001: ID 0000:0000
Bus 002 Device 001: ID 0000:0000
```

LSPCI

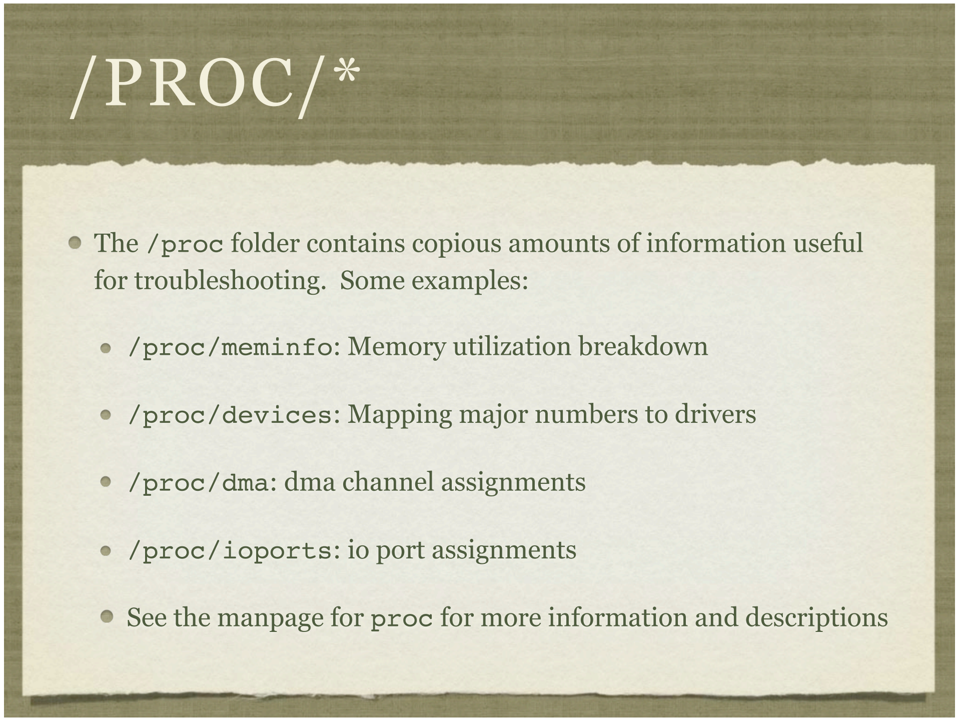
- `lspci`: List PCI bus
  - Generates a listing of devices on the PCI bus
  - Consider `-v` option for *detailed* information



```
[root@localhost ~]# lspci
00:00.0 Host bridge: Intel Corporation 82945G/GZ/P/PL Memory Controller Hub (rev 02)
00:02.0 VGA compatible controller: Intel Corporation 82945G/GZ Integrated Graphics Controller (rev
00:1b.0 Audio device: Intel Corporation N10/ICH 7 Family High Definition Audio Controller (rev 01)
00:1c.0 PCI bridge: Intel Corporation N10/ICH 7 Family PCI Express Port 1 (rev 01)
00:1c.1 PCI bridge: Intel Corporation N10/ICH 7 Family PCI Express Port 2 (rev 01)
...
```

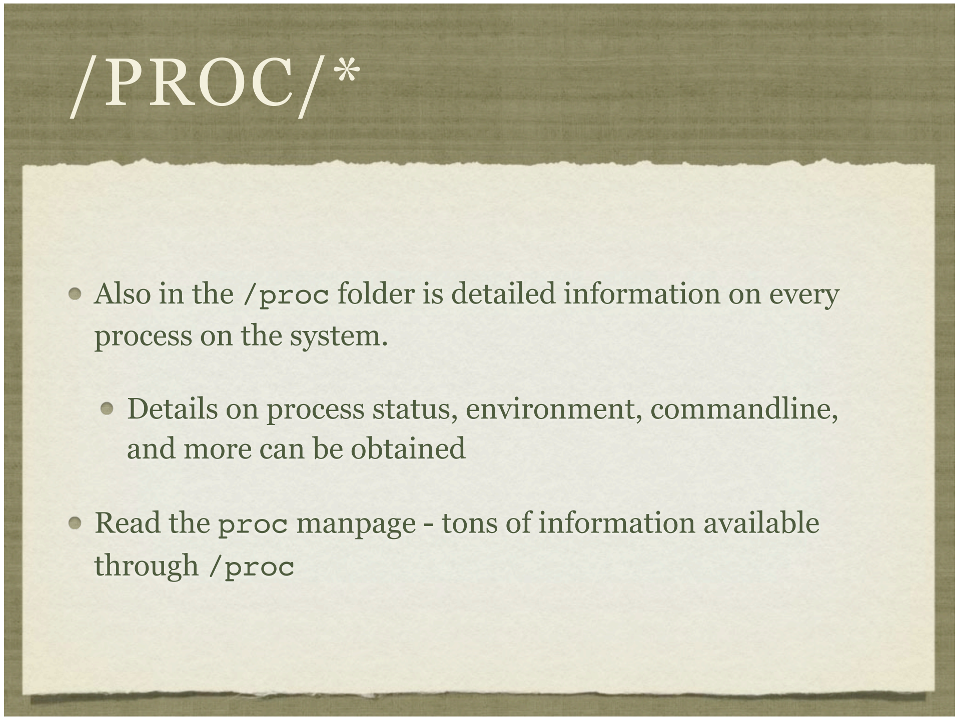
HEAVY ARTILLERY

- Now to discuss some of the more powerful troubleshooting tools
- *Not for the faint of heart* :)



/PROC/*

- The `/proc` folder contains copious amounts of information useful for troubleshooting. Some examples:
  - `/proc/meminfo`: Memory utilization breakdown
  - `/proc/devices`: Mapping major numbers to drivers
  - `/proc/dma`: dma channel assignments
  - `/proc/ioports`: io port assignments
  - See the manpage for `proc` for more information and descriptions



/PROC/*

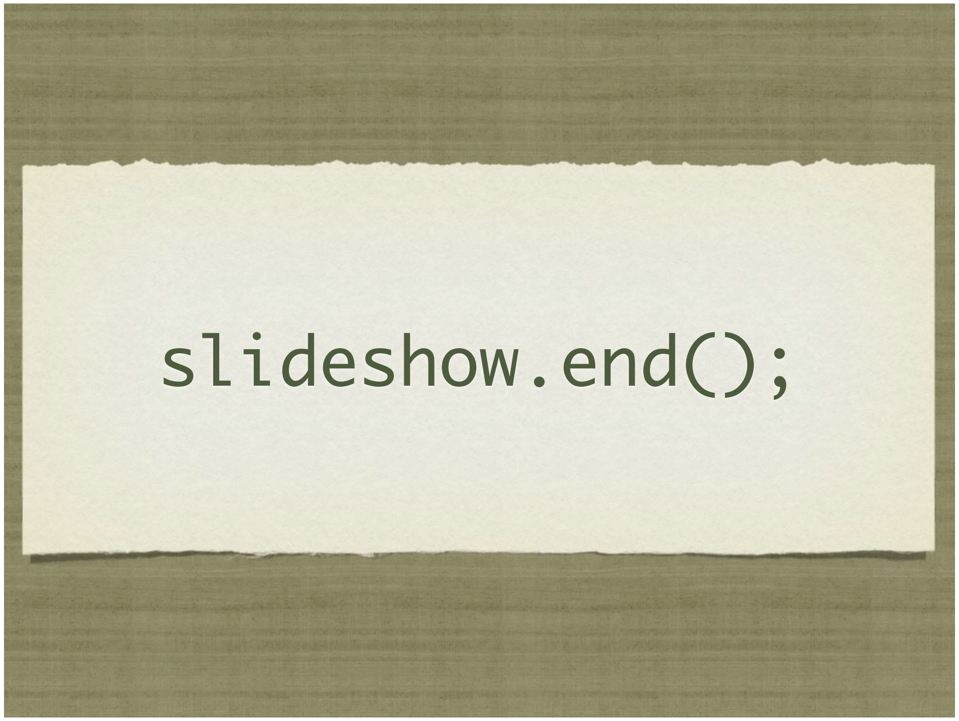
- Also in the `/proc` folder is detailed information on every process on the system.
  - Details on process status, environment, commandline, and more can be obtained
- Read the `proc` manpage - tons of information available through `/proc`

DEBUGFS

- debugfs: Very powerful filesystem debugging tool.
  - Allows direct visualization and manipulation of the filesystem internals
  - Extremely powerful, extremely dangerous. Duh!

STRACE

- strace: Traces each library call a process makes
  - Extremely useful to see what a process is doing
  - Can find errors, bugs, permission issues and more
  - Let’s play with it for a few minutes…



```
slideshow.end();
```