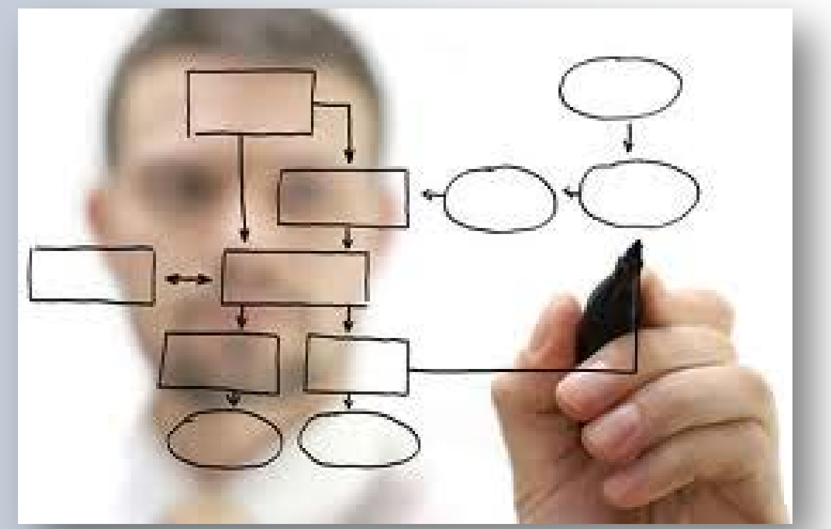
COMP 175

System Administration and Security



VIRTUAL MEMORY AND PROCESSES

Virtual Memory & Processes

- Objectives
- Upon completion you will:
  - Understand the concept of virtual memory, including swapping and paging
  - Understand the concept of a process and communication with other processes
  - Understand the concept of signals
  - Understand several ways to monitor and control processes and virtual memory, and their impact on performance

The Thing King and the Paging Game - Rules

1. Each player gets several million things.
2. Things are kept in crates that hold 4,096 things each. Things in the same crate are called crate-mates.
3. Crates are stored either in the workshop or warehouses. The workshop is almost always too small to hold all the crates.
4. There is only one workshop but there may be several warehouses. Everybody shares them.
5. Each thing has its own thing number.
6. What you do with a thing is to zark it. Everybody takes turns zarking.
7. You can only zark your things, not anybody else's.
8. Things can only be zarked when they are in the workshop.
9. Only the Thing King knows whether a thing is in the workshop or a warehouse.
10. The longer a thing goes without being zarked, the grubbier it becomes.

Virtual Memory

11. The way you get things is to ask the Thing King. He only gives out things by the crateful. This is to keep the royal overhead down.
12. The way you zark a thing is to give its thing number. If you give the number of a thing that happens to be in a workshop it gets zarked right away. If it is in a warehouse, the Thing King packs the crate containing your thing back into the workshop. If there is no room in the workshop, he first finds the grubbiest crate in the workshop, whether it be yours or somebody else's, and packs it off with all its crate-mates to a warehouse. In its place he puts the crate containing your thing. Your thing then gets zarked and you never know that it wasn't in the workshop all along.
13. Each player's stock of things have the same numbers as everybody else's. The Thing King always knows who owns what thing and whose turn it is, so you can't ever accidentally zark somebody else's thing even if it has the same thing number as one of yours.

Virtual Memory

Notes

1. Traditionally, the Thing King sits at a large, segmented table and is attended to by pages (the so-called “table pages”) whose job it is to help the king remember where all the things are and who they belong to.
2. One consequence of Rule 13 is that everybody’s thing numbers will be similar from game to game, regardless of the number of players.
3. The Thing King has a few things of his own, some of which move back and forth between workshop and warehouse just like anybody else’s, but some of which are just too heavy to move out of the workshop.
4. With the given set of rules, oft-zarked things tend to get kept mostly in the workshop while little-zarked things stay mostly in a warehouse. This is efficient stock control.

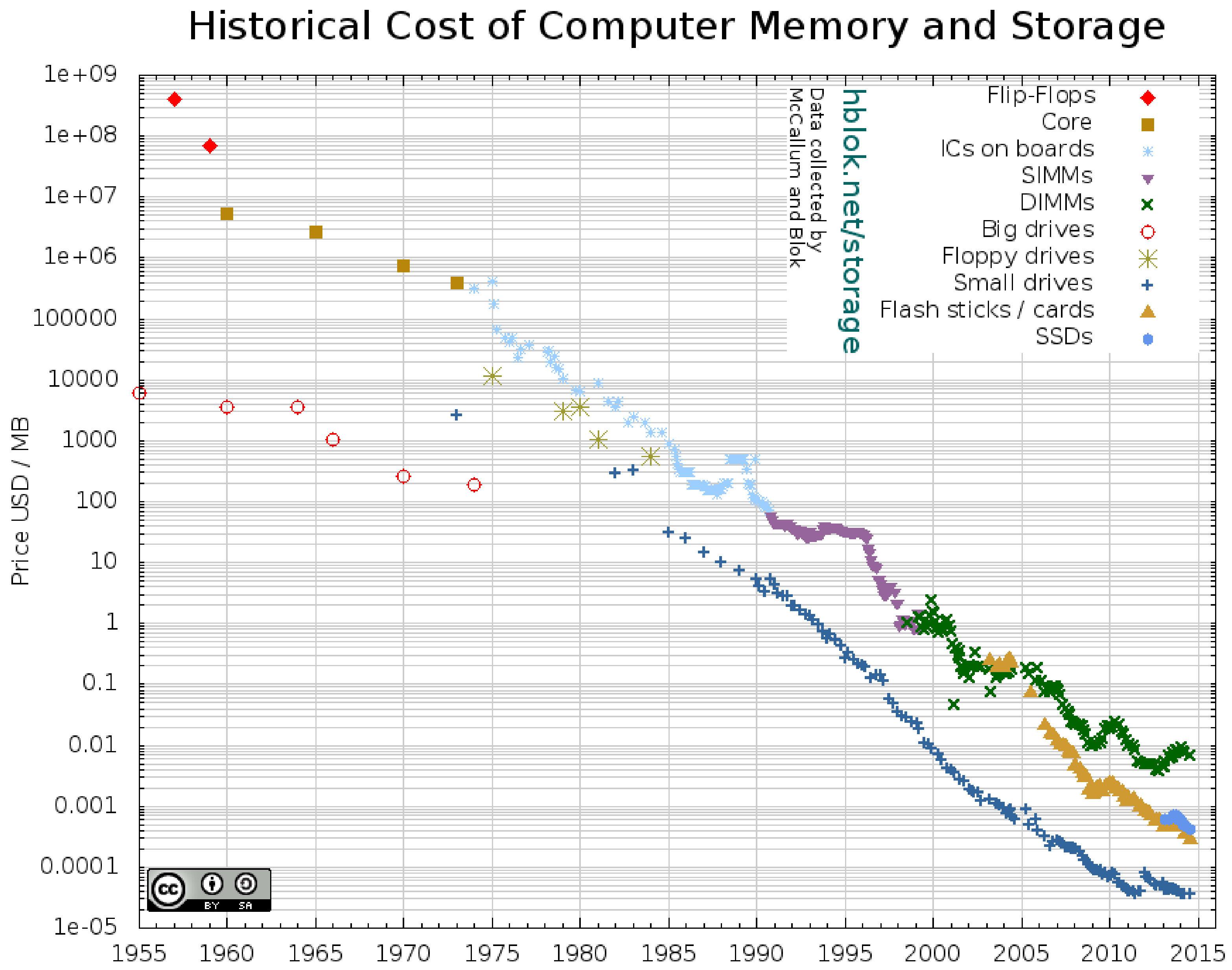
Long Live the Thing King!

-Jeff Berryman from Expert C Programming 1972

Virtual Memory

- VM Extends amount of Physical memory
- Linux: Total Memory = swap + RAM
- Swapping
  - Moving pages to and from memory
  - Page – block (unit) of RAM
- Memory used to be expen$ive

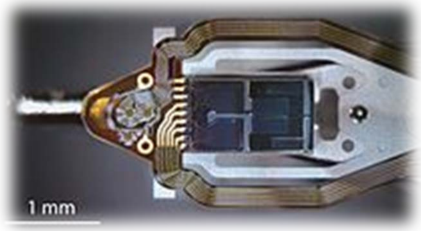
Memory & Storage Cost

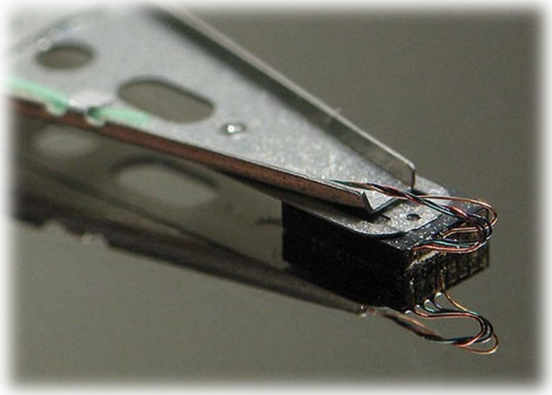


Cost Fluctuations

- RAM – prices doubled after 2012 Hynix fab plant fire
- Storage – 2011 Thailand flooding (WD, Samsung, Toshiba) raised prices through 2013

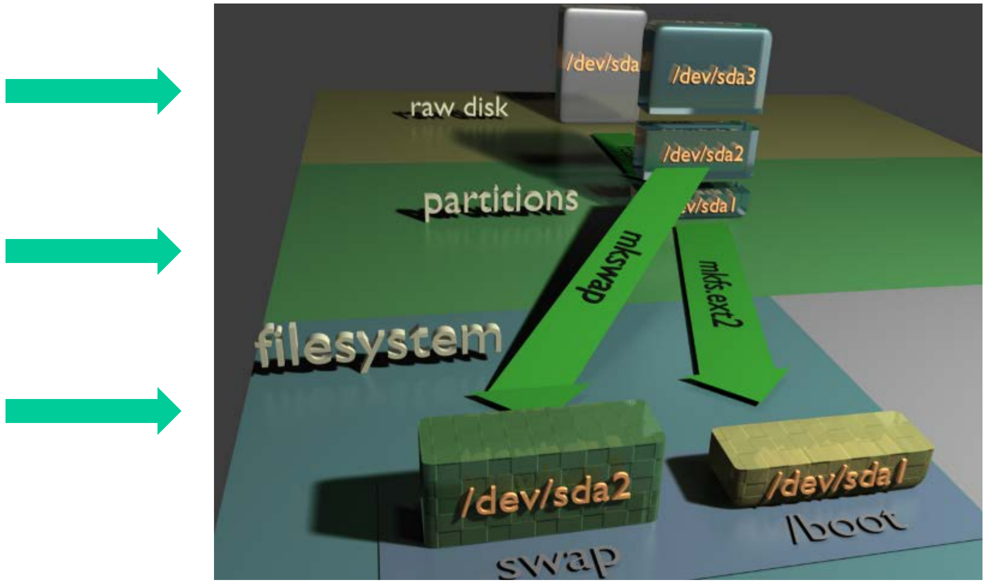
- Storage RPM
  - 4,200 – energy efficient portable devices
  - 5,400 – 100 Mbs r/w consumer grade
  - 7,200 – 120 Mbs r/w consumer grade
  - 10,000 – 15,000 server drives





Swap Space

- Used when system runs out of RAM for currently running programs
- Separate hard disk partition called swap partition
- Also called virtual memory
- Acts like extension of system's RAM
- Swapped-out process cannot run until swapped-in



Paged Virtual Memory

- Implementations of virtual memory divide a virtual address space into pages, blocks of contiguous virtual memory addresses.
- Pages are usually at least 4 kilobytes in size.
- Page tables are used to translate virtual addresses seen by the application into physical addresses used by hardware to process instructions.
- Each entry in the page table holds a flag indicating whether the corresponding page is in real memory or not.

Paged Virtual Memory

- If it *is* in real memory, the page table entry will contain the real memory address at which the page is stored.
- If it is *not* in real memory, the hardware raises a page fault exception
- The paging supervisor then accesses secondary storage, returns the requested page, updates the page tables to reflect the physical location of the virtual address and signals to restart the request.
- When all physical memory is already in use, the paging supervisor must free a page in primary storage to hold the swapped-in page.

Virtual Memory

- Virtual memory – separation of user logical memory from physical memory.
- Only part of the program needs to be in memory for execution
- Logical address space can therefore be much larger than physical address space
- Allows address spaces to be shared by several processes
- Virtual memory typically implemented via:
  - Demand paging

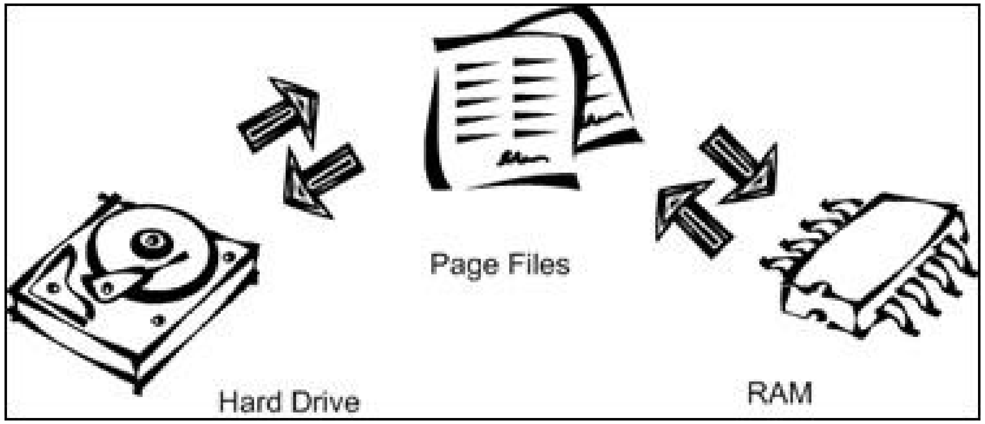
Demand Paging

- Bring a page into memory only when it is needed
  - Less I/O needed
  - Less memory needed
  - Faster response
  - More users

- Page is needed □reference to it
  - Invalid reference □abort
  - Not-in-memory □bring to memory

Paged Virtual Memory

- Thrashing is when the computer spends excessive time swapping pages to and from disk, slowing down useful work. Adding real memory is the simplest response, but improving application design, scheduling, and memory usage can help.
- Backing store is part of a hard disk used for paging or swapping – its cheaper but slower than memory. Lots slower.



Compare and Contrast

- A page fault occurs when the Cache Manager tries to access data not in cache memory
- Prepaging attempts to reduce page faults from process startup by prepaging

- Whole books/classes on this topic
- Zark, Thing King are as good a description

Code Suckage

Program structure

Int[128,128] data;

Each row is stored in one page

Program 1

```
for (j = 0; j <128; j++)
    for (i = 0; i < 128; i++)
        data[i,j] = 0;
```

128 x 128 = 16,384 page faults

Program 2

```
for (i = 0; i < 128; i++)
    for (j = 0; j < 128; j++)
        data[i,j] = 0;
```

128 page faults

Processes

- Key aspect of the kernel is process management
  - Creation and termination of processes
  - Process scheduler
- A process is the executing program code
- May include resources such as open files, pending signals, internal kernal data, processor state, address space, one or more threads of execution, and a data section containing global variables.
- Threads of execution are the objects of activity within the process. Each thread includes a unique program counter, process stack, and set of processor registers.

Processes

- Multiple, independent processes may be running at the same time.
- Each user has several active processes at once
- A large system may have hundreds or more
- Background processes (daemons) are also running
  - Example: cron daemon wakes up once a minute
  - Checks if there is any work for it to do
  - If so, it does the work
  - Then it goes back to sleep

Processes

- Kernel maintains information about processes
- Linux is multitasking operating system
- Single microprocessor can really only perform one task at a time
- Time slice
  - A few microseconds
  - Allocated to each process by kernel
  - Common to all operating systems that don’t have multiple microprocessors

Process Creation

- Processes started from
  - Command line or GUI graphical desktop
  - Kernel
  - Another process
- Linux processes are created by the fork system call creating an exact copy of the original process.
  - Forking process called the parent process
  - The new process called the child process
- The parent and child each have their own, private memory images.
- If parent changes any of its variables, changes are not visible to the child, and vice versa.

Process Files

- Open files are shared between parent and child.
- Files open to parent before the fork, will be open in both parent and child afterward
- Changes to file made by one are visible to other
- How do the processes know which should run parent code and which should run child code?
  - Fork system call returns a 0 to child
  - Nonzero value to parent
    - child’s PID (Process Identifier)
- Both processes normally check the return value
  - Act accordingly

Processes

- Processes are named by their PIDs
- If child wants to know its own PID
  - getpid system call provides it
- When a child terminates parent is given the PID
  - A parent may have many children
  - Children may also have children
  - Original process can build up an entire tree of children, grandchildren, and further descendants
- Daemon spawn!

Process Lifecycle

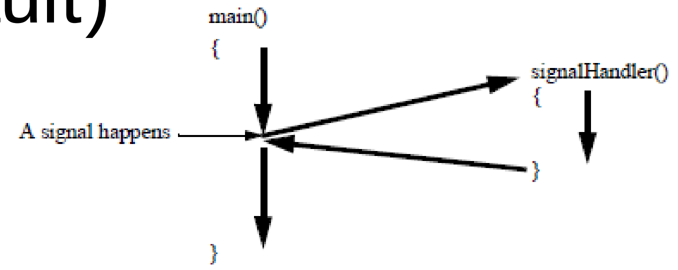
- Child finishes
  - Sends status message to parent
  - Zombie – waits for parent acknowledgment
  - Orphan – init (PID 1) becomes parent

Processes

- Linux processes communicate by creating a pipe between themselves where one process writes bytestream for other to read
- Synchronization: a process can't read from an empty pipe until data are available
- Shell pipelines are implemented with pipes
- `sort <f | head` causes the shell to create two processes, sort and head, with a pipe between them connecting sort’s standard output to head’s standard input
- Sort data goes directly to head instead of file
- If pipe fills up, system stops running sort until head removes some data from the pipe

Processes

- Processes can communicate via software interrupts
- Process sends a signal to another process
- Processes can tell system what to do when the signal arrives
  - Let signal kill process (default)
  - Ignore it
  - Catch it
    - Must specify a signal handling procedure.
    - Control abruptly switches to handler upon arrival
    - Control reverts when handler finishes and returns



Processes

Starting Processes from the Shell

- When you start program
  - Program takes control of command line
  - Parent process waits for new process to finish
- Type ampersand (&) after name of command
  - Shell forks new process without pausing itself
  - Runs the process in background
  - Can start another command immediately
  - `jobs` command shows child processes
  - `Ctl-Z` suspends job - see `bg` and `fg` commands

Signals

#	Name	Description
1.	SIGHUP	Hangup (POSIX)
2.	SIGINT	Terminal interrupt (ANSI)
3.	SIGQUIT	Terminal quit (POSIX)
4.	SIGILL	Illegal instruction (ANSI)
5.	SIGTRAP	Trace trap (POSIX)
6.	SIGIOT	IOT Trap (4.2 BSD) Synonym for SIGABRT (abort)
7.	SIGBUS	BUS error (4.2 BSD)
8.	SIGFPE	Floating point exception (ANSI)
9.	SIGKILL	Kill (can't be caught or ignored) (POSIX)

Signals

#	Name	Description
1.	SIGHUP	Hangup (POSIX)
2.	SIGINT	Te
3.	SIGQUIT	Te
4.	S	Ill
5.	GTRAP	Tr
6.	GIOT	IC
7.	IGBUS	BU
8.	SIGFPE	Flo
9.	SIGKILL	Kill (can't be caught or ignored) (POSIX)

`kill –HUP pid`
and
`kill -9 pid`
for killing processes

<Ctrl><c> - Terminate process
<Ctrl><z> - Suspend process

Monitoring Processes - Linux

- ps (process status)
  - BSD – Sorts by %CPU Usage
  - SVR4 – Sorts by PID
- top
  - Full terminal screen display
  - Sortable
  - Can kill and renice processes

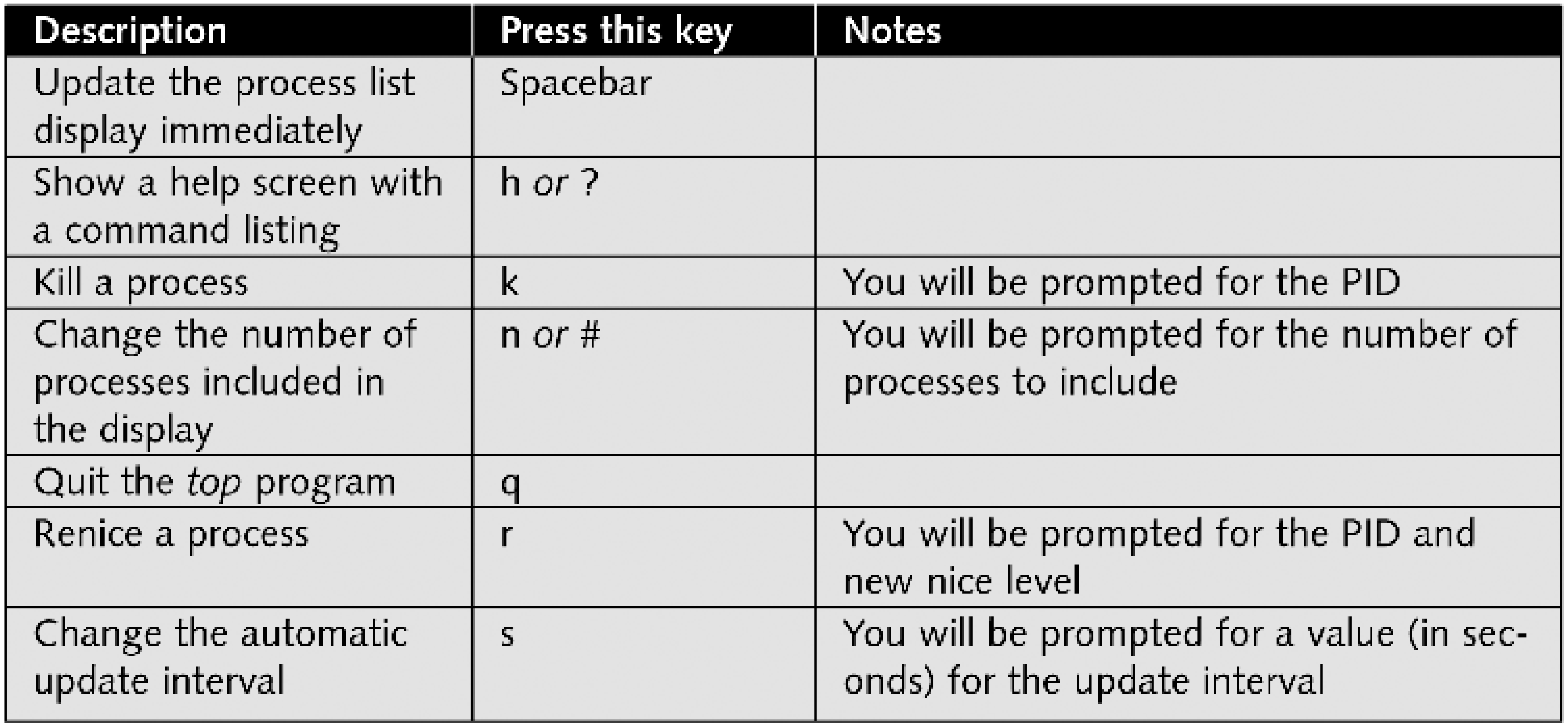
Monitoring Processes - Linux

top utility

- Displays list of running processes
- Arranged by how much CPU time each is using
- Process consuming greatest amount of CPU time shown at top of list
- Updated regularly
- Normally started without any options
- Cannot run in background

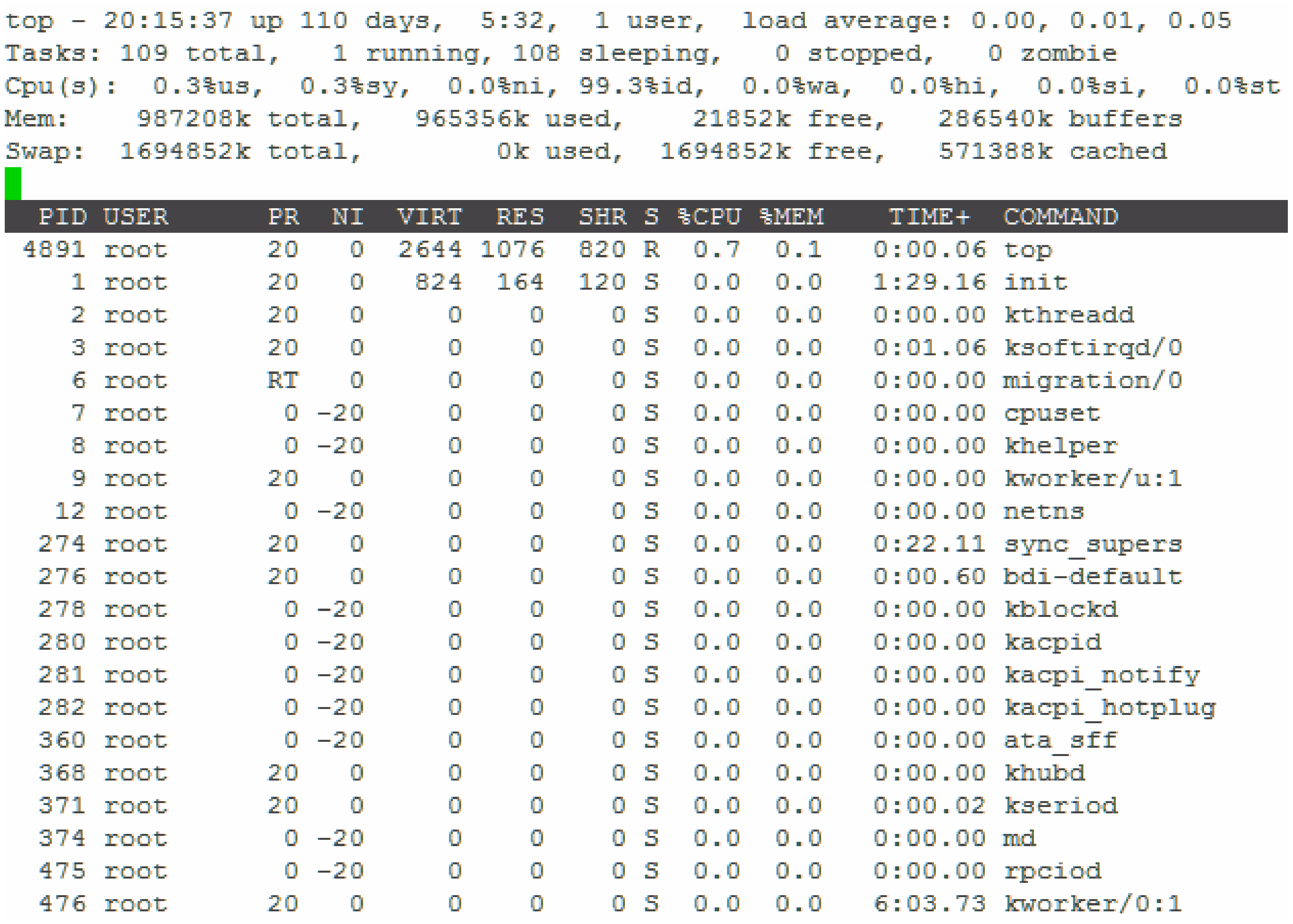
Monitoring Processes - Linux

Table 6-4 Interactive Commands in *top*



Description	Press this key	Notes
Update the process list display immediately	Spacebar	
Show a help screen with a command listing	h *or* ?	
Kill a process	k	You will be prompted for the PID
Change the number of processes included in the display	n *or* #	You will be prompted for the number of processes to include
Quit the *top* program	q	
Renice a process	r	You will be prompted for the PID and new nice level
Change the automatic update interval	s	You will be prompted for a value (in seconds) for the update interval

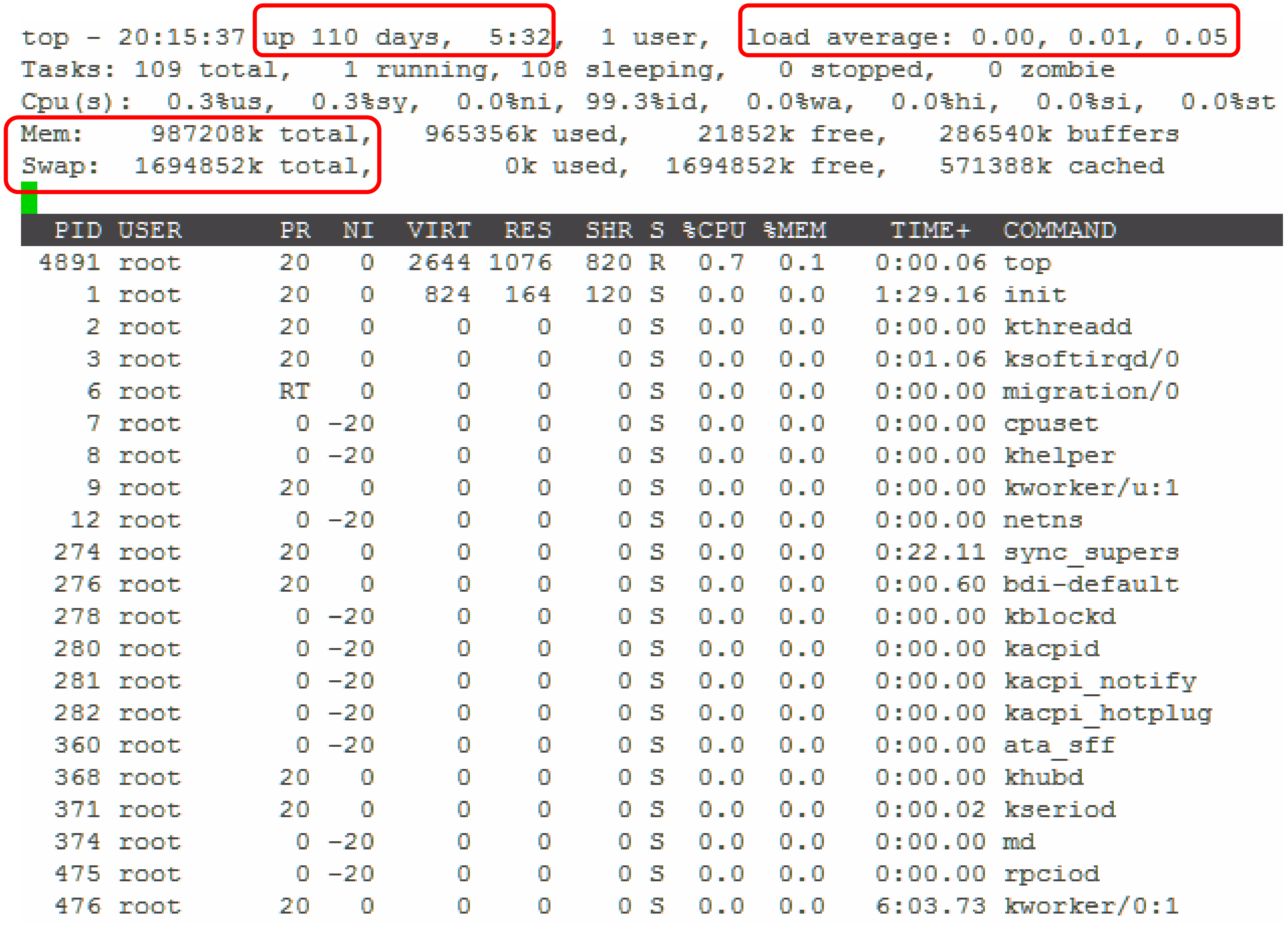
top



```
top - 20:15:37 up 110 days,  5:32,  1 user,  load average: 0.00, 0.01, 0.05
Tasks: 109 total,   1 running, 108 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.3%us,  0.3%sy,  0.0%ni, 99.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    987208k total,   965356k used,    21852k free,   286540k buffers
Swap:  1694852k total,        0k used,  1694852k free,   571388k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 4891 root      20   0  2644 1076  820 R  0.7  0.1   0:00.06 top
    1 root      20   0   824  164  120 S  0.0  0.0   1:29.16 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0     0    0    0 S  0.0  0.0   0:01.06 ksoftirqd/0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    7 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 cpuset
    8 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 khelper
    9 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kworker/u:1
   12 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 netns
  274 root      20   0     0    0    0 S  0.0  0.0   0:22.11 sync_supers
  276 root      20   0     0    0    0 S  0.0  0.0   0:00.60 bdi-default
  278 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kblockd
  280 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kacpid
  281 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kacpi_notify
  282 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kacpi_hotplug
  360 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 ata_sff
  368 root      20   0     0    0    0 S  0.0  0.0   0:00.00 khubd
  371 root      20   0     0    0    0 S  0.0  0.0   0:00.02 kseriod
  374 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 md
  475 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 rpciod
  476 root      20   0     0    0    0 S  0.0  0.0   6:03.73 kworker/0:1
```

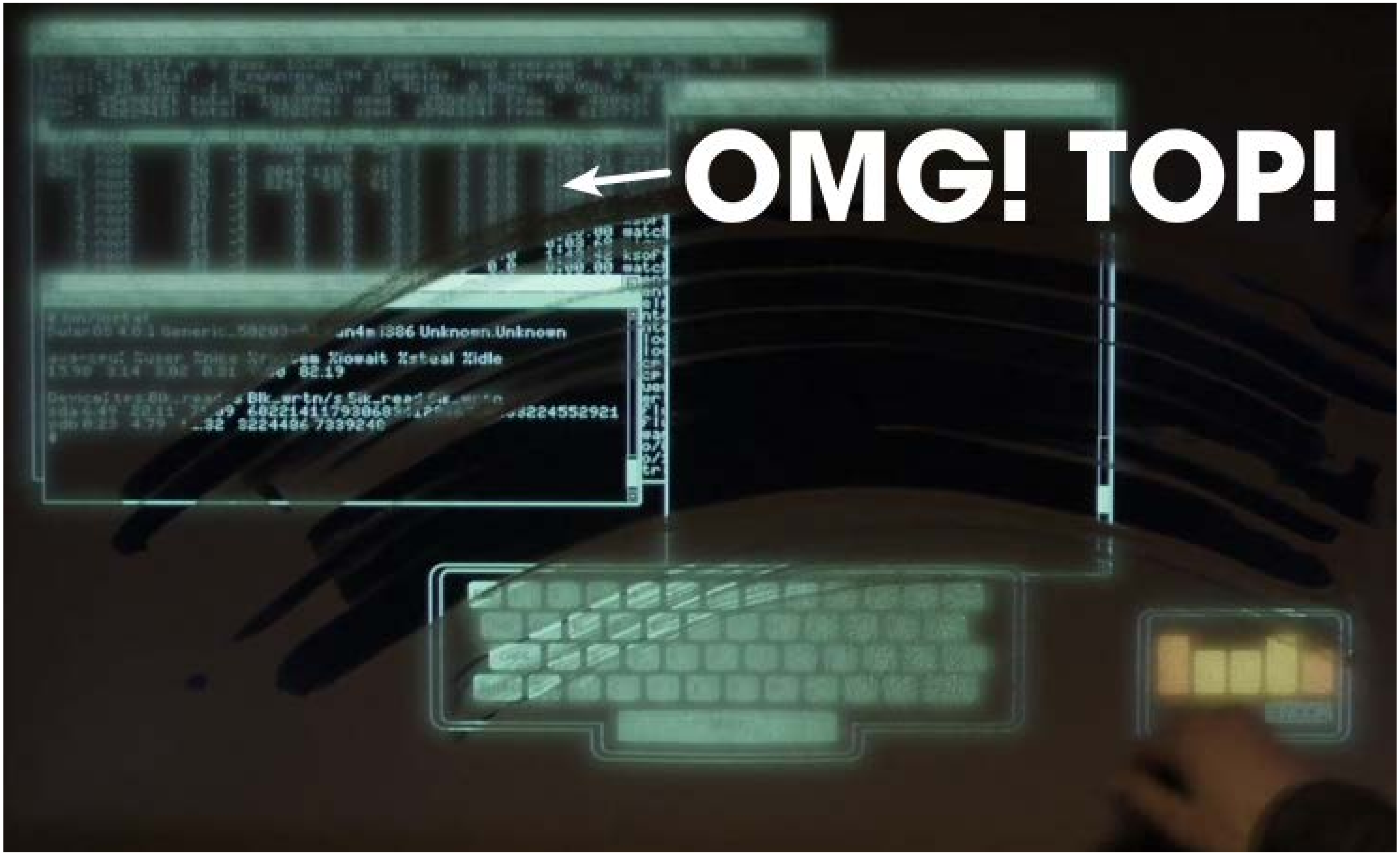
top



```
top - 20:15:37 up 110 days,  5:32,  1 user,  load average: 0.00, 0.01, 0.05
Tasks: 109 total,   1 running, 108 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.3%us,  0.3%sy,  0.0%ni, 99.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:     987208k total,   965356k used,    21852k free,   286540k buffers
Swap:   1694852k total,        0k used,  1694852k free,   571388k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 4891 root      20   0  2644 1076  820 R  0.7  0.1   0:00.06 top
    1 root      20   0   824  164  120 S  0.0  0.0   1:29.16 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0     0    0    0 S  0.0  0.0   0:01.06 ksoftirqd/0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    7 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 cpuset
    8 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 khelper
    9 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kworker/u:1
   12 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 netns
  274 root      20   0     0    0    0 S  0.0  0.0   0:22.11 sync_supers
  276 root      20   0     0    0    0 S  0.0  0.0   0:00.60 bdi-default
  278 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kblockd
  280 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kacpid
  281 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kacpi_notify
  282 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kacpi_hotplug
  360 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 ata_sff
  368 root      20   0     0    0    0 S  0.0  0.0   0:00.00 khubd
  371 root      20   0     0    0    0 S  0.0  0.0   0:00.02 kseriod
  374 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 md
  475 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 rpciod
  476 root      20   0     0    0    0 S  0.0  0.0   6:03.73 kworker/0:1
```

Tron: Legacy Trailer



```
[illegible]
SolarOS 4.0.1 Generic_50203-[illegible]un4m i386 Unknown.Unknown

[illegible] %user %nice %system %iowait %steal %idle
15.90 [illegible] 82.19

Device: [illegible] kB_read/s kB_wrtn/s kB_read kB_wrtn
sda [illegible] 6022141179306[illegible]3224552921
sdb [illegible] 8224486 7339240
#
```

Information from ps

- PID
- USER
- PRI/NI - Priority/Nice level
- RSS/SIZE - Resident/Total Memory used
- STAT - Process State
- %CPU/%MEM - % of System CPU/Memory
- TIME - CPU time used
- Command
- Remember from earlier slides – option flags
  - ps –ef
  - ps -efx

ps

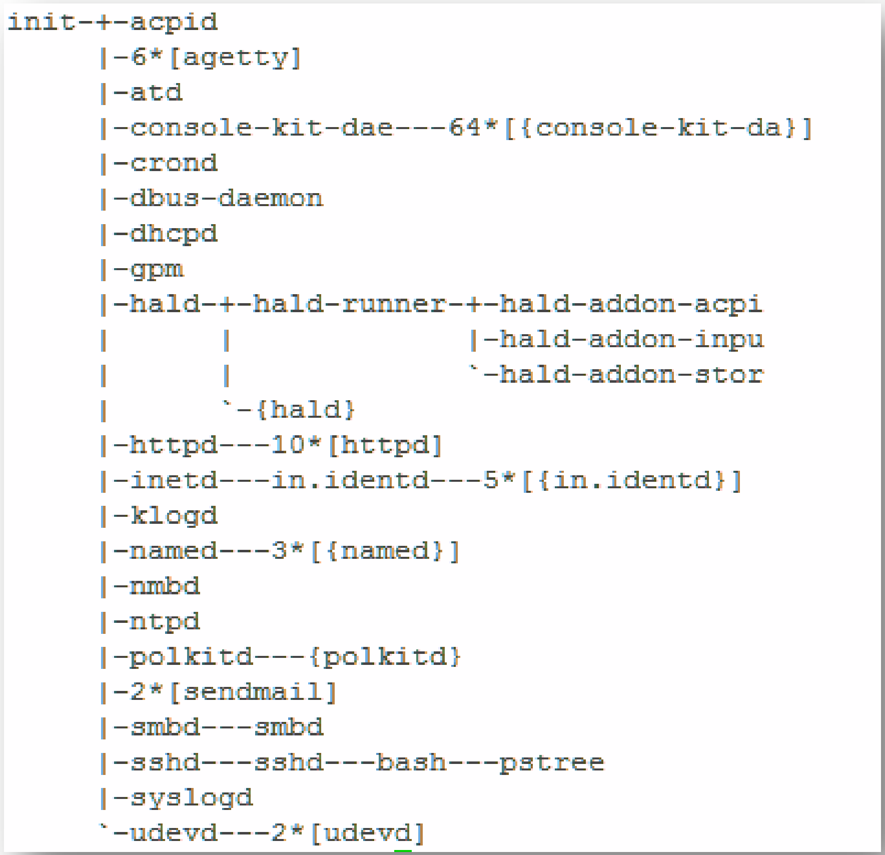
ps your processes from current session

ps –e | grep named

ps – eLf   get thread information

pstree   process tree

$ pstree



```
init-+-acpid
     |-6*[agetty]
     |-atd
     |-console-kit-dae---64*[{console-kit-da}]
     |-crond
     |-dbus-daemon
     |-dhcpd
     |-gpm
     |-hald-+-hald-runner-+-hald-addon-acpi
     |      |             |-hald-addon-inpu
     |      |             `-hald-addon-stor
     |      `-{hald}
     |-httpd---10*[httpd]
     |-inetd---in.identd---5*[{in.identd}]
     |-klogd
     |-named---3*[{named}]
     |-nmbd
     |-ntpd
     |-polkitd---{polkitd}
     |-2*[sendmail]
     |-smbd---smbd
     |-sshd---sshd---bash---pstree
     |-syslogd
     `-udevd---2*[udevd]
```

Management Tools

Viewing Virtual Memory Information

- vmstat command
  - View detailed information about how swap space is being used
  - Displayed information based on information averaged over time since system was started
  - Can also run to be continuously updated

Monitoring Processes (Windows)

- Task Manager
  - Graphical viewer
  - Can also sort or kill processes
- tasklist
  - Command line Viewer
  - Can view associated dll files
- wmic process [options]
  - Can view, start, and kill processes
  - Remote connection capabilities

SysInternals Tools

- Command Line tools
  - Pslist – display running processes
  - Pskill – Kill processes by name or PID
  - Psexec – Run programs remotely
- GUI Tools
  - Procmon – Detailed process info
  - RAMMap – View RAM usage
  - VMMap – View Virtual Memory usage

Processes and Signals

...although the *kill* part

that sounded good

Trouble ticket: Computer seems slow

Blame game – what is the real issue

Worse: Your system seems slow

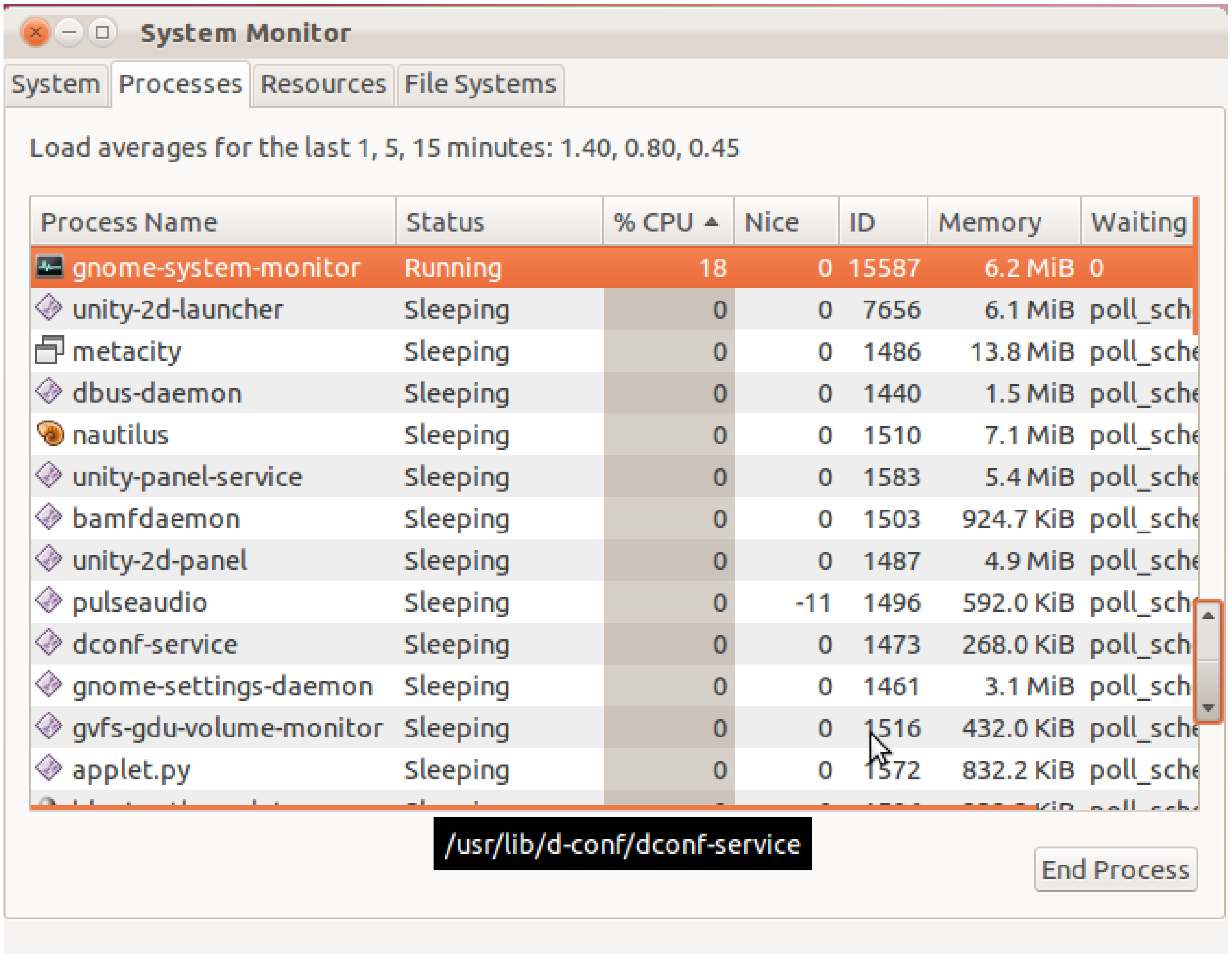
Even worse: Your system seems infected

Motivated now...?!

System Performance

Slow....

Tool ate CPU

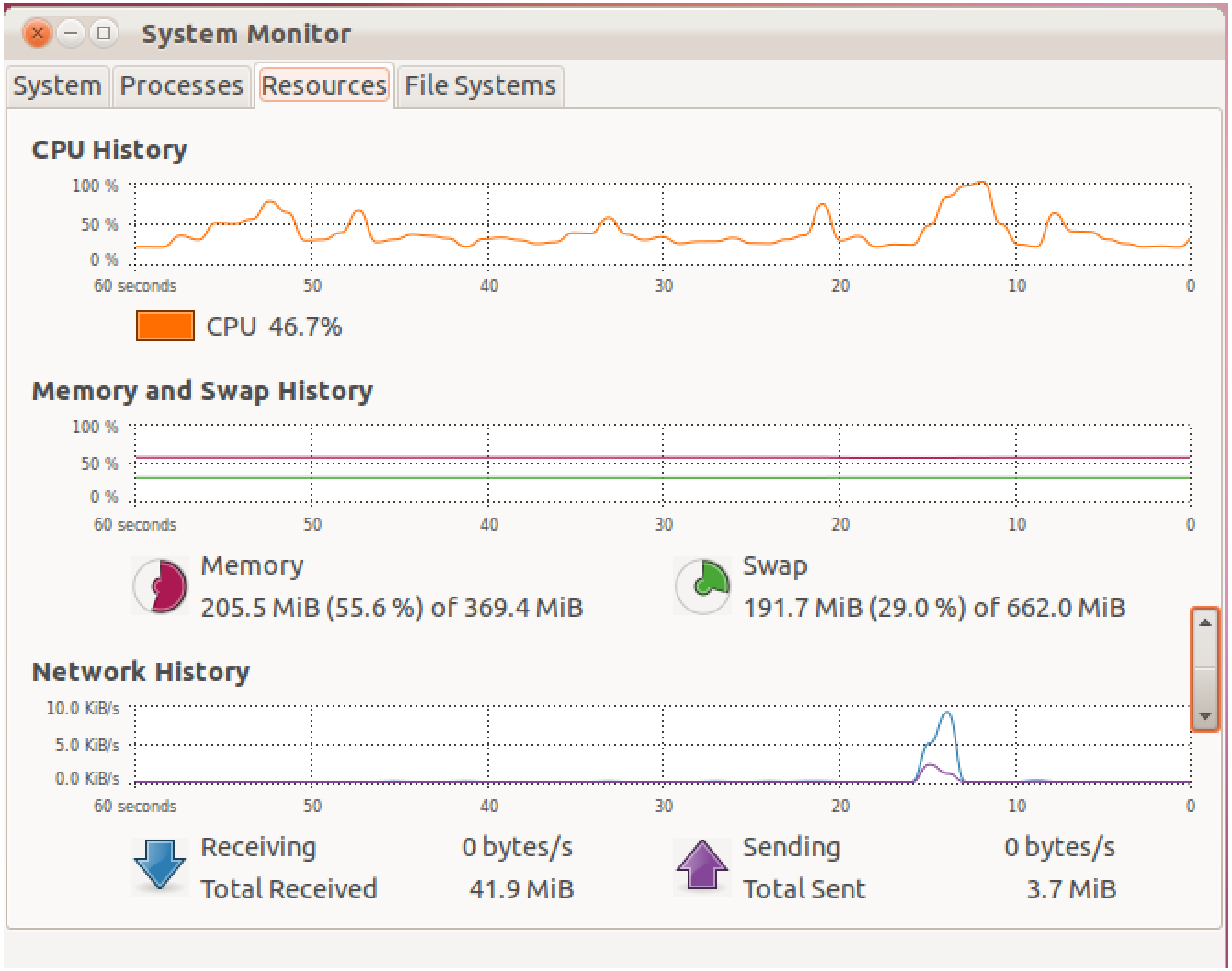


Load averages for the last 1, 5, 15 minutes: 1.40, 0.80, 0.45

Process Name	Status	% CPU ▲	Nice	ID	Memory	Waiting
gnome-system-monitor	Running	18	0	15587	6.2 MiB	0
unity-2d-launcher	Sleeping	0	0	7656	6.1 MiB	poll_sch
metacity	Sleeping	0	0	1486	13.8 MiB	poll_sche
dbus-daemon	Sleeping	0	0	1440	1.5 MiB	poll_sche
nautilus	Sleeping	0	0	1510	7.1 MiB	poll_sche
unity-panel-service	Sleeping	0	0	1583	5.4 MiB	poll_sche
bamfdaemon	Sleeping	0	0	1503	924.7 KiB	poll_sche
unity-2d-panel	Sleeping	0	0	1487	4.9 MiB	poll_sche
pulseaudio	Sleeping	0	-11	1496	592.0 KiB	poll_sch
dconf-service	Sleeping	0	0	1473	268.0 KiB	poll_sch
gnome-settings-daemon	Sleeping	0	0	1461	3.1 MiB	poll_sch
gvfs-gdu-volume-monitor	Sleeping	0	0	1516	432.0 KiB	poll_sche
applet.py	Sleeping	0	0	1572	832.2 KiB	poll_sche

/usr/lib/d-conf/dconf-service

System Performance



CPU History

🟧 CPU 46.7%

Memory and Swap History

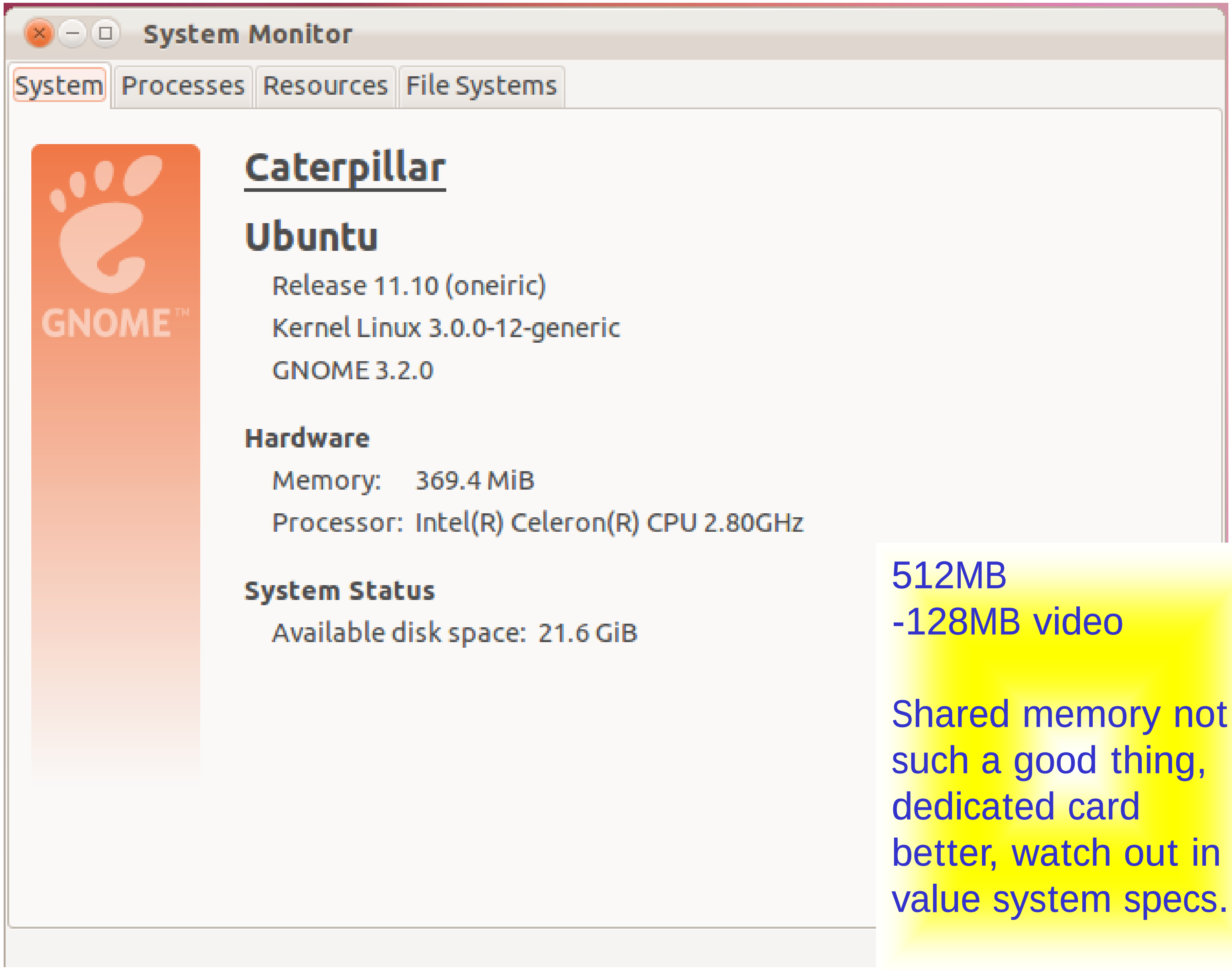
Memory
205.5 MiB (55.6 %) of 369.4 MiB

Swap
191.7 MiB (29.0 %) of 662.0 MiB

Network History

Receiving	0 bytes/s	Sending	0 bytes/s
Total Received	41.9 MiB	Total Sent	3.7 MiB

System Performance



Caterpillar

Ubuntu

Release 11.10 (oneiric)
Kernel Linux 3.0.0-12-generic
GNOME 3.2.0

Hardware

Memory:	369.4 MiB
Processor:	Intel(R) Celeron(R) CPU 2.80GHz

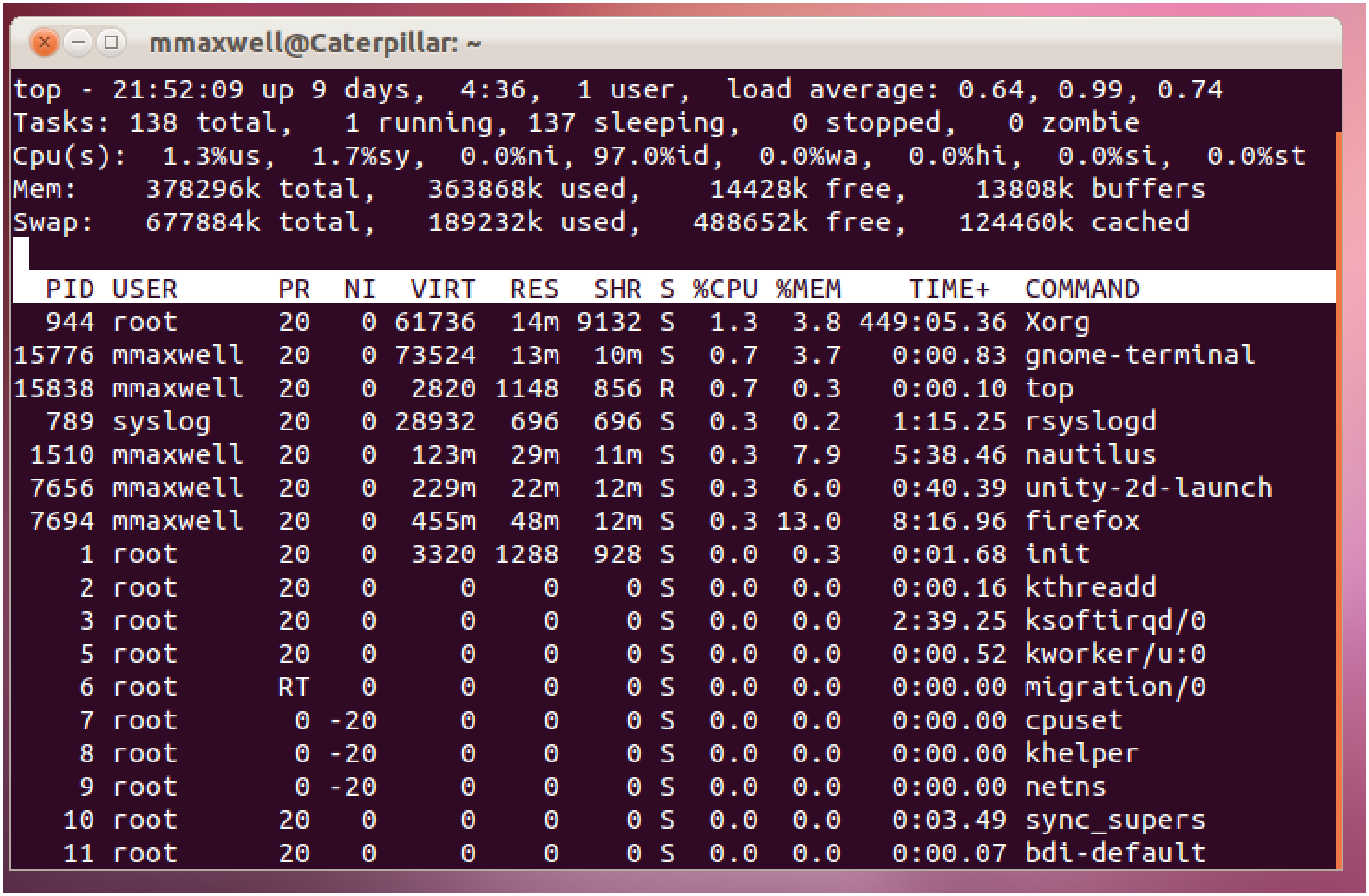
System Status

Available disk space: 21.6 GiB

512MB
-128MB video

Shared memory not such a good thing, dedicated card better, watch out in value system specs.

top

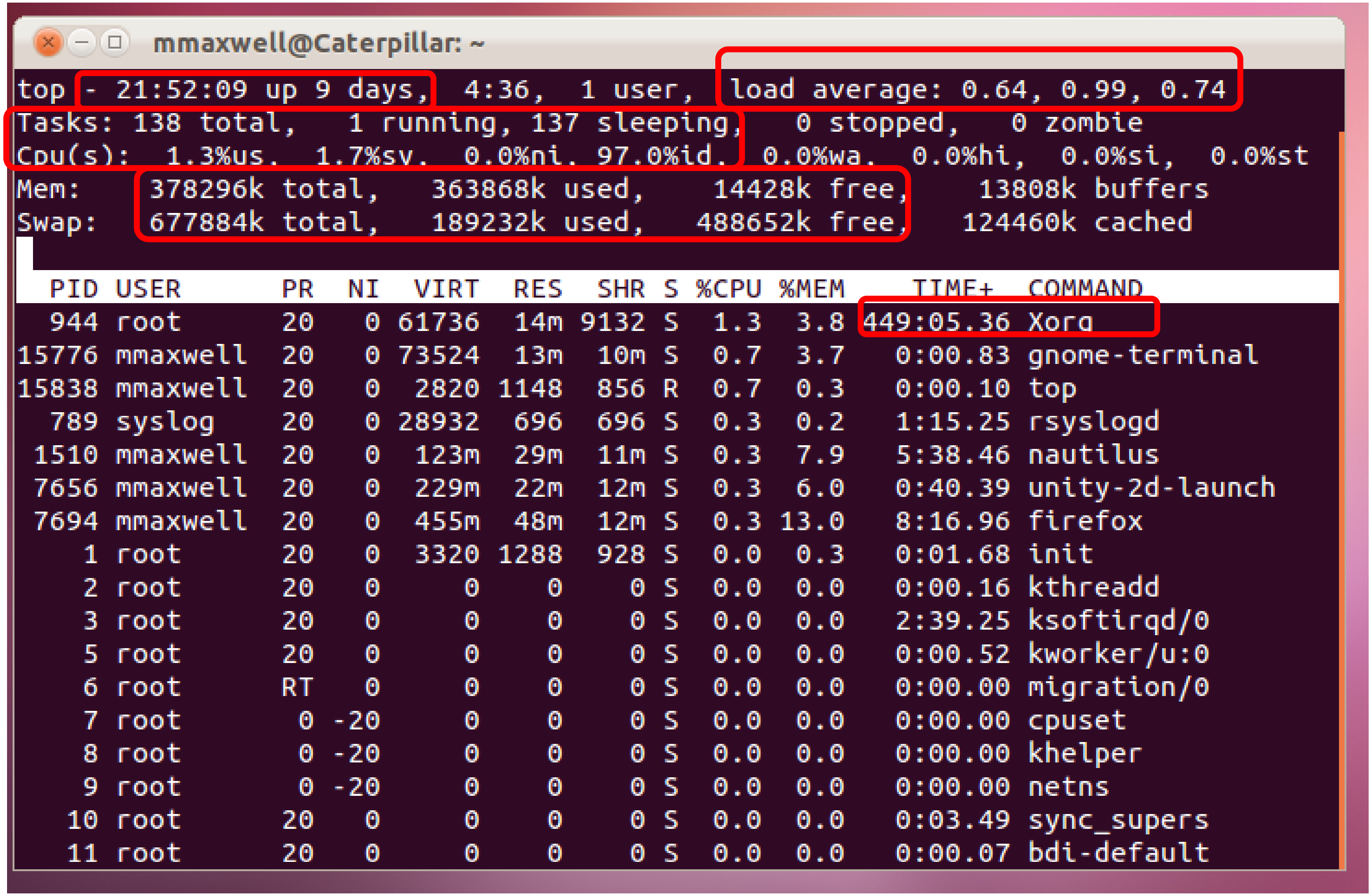


mmaxwell@Caterpillar: ~

```
top - 21:52:09 up 9 days,  4:36,  1 user,  load average: 0.64, 0.99, 0.74
Tasks: 138 total,   1 running, 137 sleeping,   0 stopped,   0 zombie
Cpu(s):  1.3%us,  1.7%sy,  0.0%ni, 97.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    378296k total,   363868k used,    14428k free,    13808k buffers
Swap:   677884k total,   189232k used,   488652k free,   124460k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
  944 root      20   0 61736  14m 9132 S  1.3  3.8 449:05.36 Xorg
15776 mmaxwell  20   0 73524  13m  10m S  0.7  3.7   0:00.83 gnome-terminal
15838 mmaxwell  20   0  2820 1148  856 R  0.7  0.3   0:00.10 top
  789 syslog    20   0 28932  696  696 S  0.3  0.2   1:15.25 rsyslogd
 1510 mmaxwell  20   0  123m  29m  11m S  0.3  7.9   5:38.46 nautilus
 7656 mmaxwell  20   0  229m  22m  12m S  0.3  6.0   0:40.39 unity-2d-launch
 7694 mmaxwell  20   0  455m  48m  12m S  0.3 13.0   8:16.96 firefox
    1 root      20   0  3320 1288  928 S  0.0  0.3   0:01.68 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.16 kthreadd
    3 root      20   0     0    0    0 S  0.0  0.0   2:39.25 ksoftirqd/0
    5 root      20   0     0    0    0 S  0.0  0.0   0:00.52 kworker/u:0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    7 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 cpuset
    8 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 khelper
    9 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 netns
   10 root      20   0     0    0    0 S  0.0  0.0   0:03.49 sync_supers
   11 root      20   0     0    0    0 S  0.0  0.0   0:00.07 bdi-default
```

What does it tell us?



mmaxwell@Caterpillar: ~

```
top - 21:52:09 up 9 days,  4:36,  1 user,  load average: 0.64, 0.99, 0.74
Tasks: 138 total,   1 running, 137 sleeping,   0 stopped,   0 zombie
Cpu(s):  1.3%us,  1.7%sy,  0.0%ni, 97.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    378296k total,   363868k used,    14428k free,    13808k buffers
Swap:   677884k total,   189232k used,   488652k free,   124460k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
  944 root      20   0 61736  14m 9132 S  1.3  3.8 449:05.36 Xorg
15776 mmaxwell  20   0 73524  13m  10m S  0.7  3.7   0:00.83 gnome-terminal
15838 mmaxwell  20   0  2820 1148  856 R  0.7  0.3   0:00.10 top
  789 syslog    20   0 28932  696  696 S  0.3  0.2   1:15.25 rsyslogd
 1510 mmaxwell  20   0  123m  29m  11m S  0.3  7.9   5:38.46 nautilus
 7656 mmaxwell  20   0  229m  22m  12m S  0.3  6.0   0:40.39 unity-2d-launch
 7694 mmaxwell  20   0  455m  48m  12m S  0.3 13.0   8:16.96 firefox
    1 root      20   0  3320 1288  928 S  0.0  0.3   0:01.68 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.16 kthreadd
    3 root      20   0     0    0    0 S  0.0  0.0   2:39.25 ksoftirqd/0
    5 root      20   0     0    0    0 S  0.0  0.0   0:00.52 kworker/u:0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    7 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 cpuset
    8 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 khelper
    9 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 netns
   10 root      20   0     0    0    0 S  0.0  0.0   0:03.49 sync_supers
   11 root      20   0     0    0    0 S  0.0  0.0   0:00.07 bdi-default
```

Top Legend

- Average system load over 1, 5, and 15 minutes
- Amount of time CPU spent, displayed as %
- Running a User's Process
- Handling a kernel call, fault, or interrupt
- Memory – with virtual usually shown as swap
- Column
  - Virtual memory allocated each process
  - Physical memory allocated each process

  - So – what do the numbers suggest?

Initial Conclusions

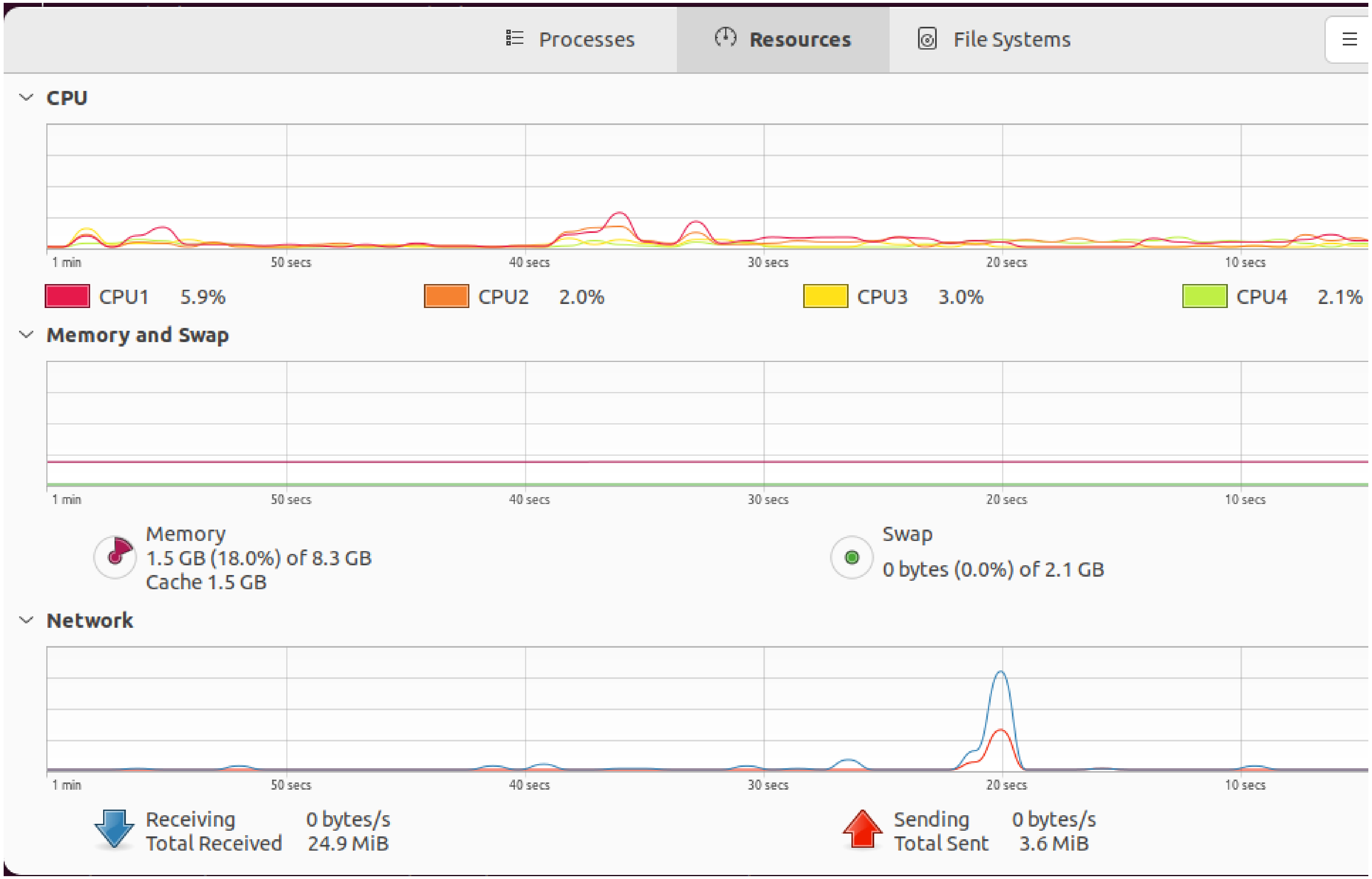
- CPU overworked – Celeron is challenged
- More memory needed
- Ubuntu workstation ate the system resources

Donor HP Pavilion laptop

- Intel Celeron 2.8 GHz
- ATI Mobility Radeon 9000 Shared video memory
- 512MB DDR SDRAM 330Mhtz PC2700

A Better Computer

- Core I5 2.4GHz 8GB Memory Intel Mesa Graphics



🟥 CPU1 5.9%
🟧 CPU2 2.0%
🟨 CPU3 3.0%
🟩 CPU4 2.1%

Memory
1.5 GB (18.0%) of 8.3 GB
Cache 1.5 GB

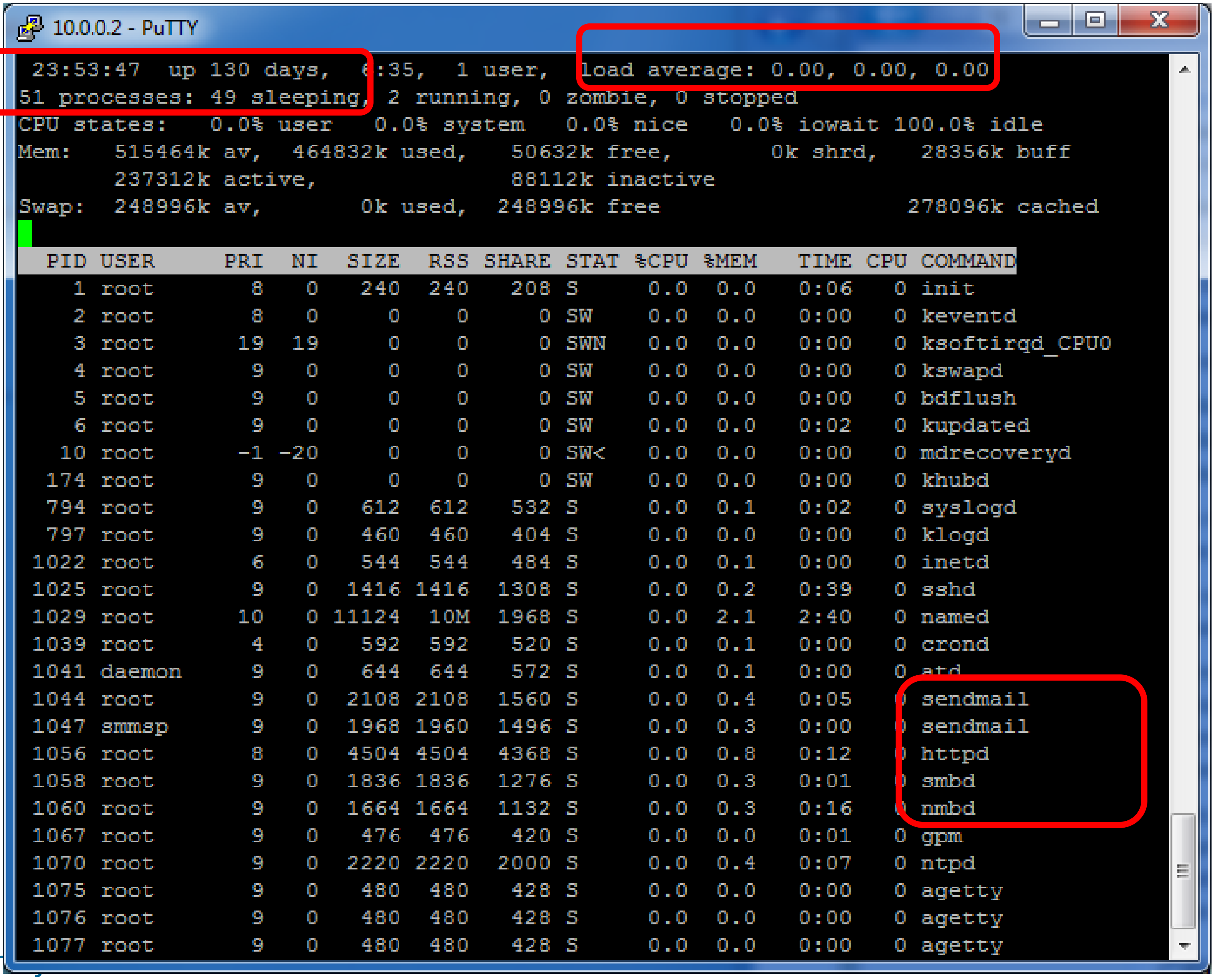
Swap
0 bytes (0.0%) of 2.1 GB

Receiving	0 bytes/s		Sending	0 bytes/s	
Total Received	24.9 MiB		Total Sent	3.6 MiB	

An Internet Server

DNS
DHCP
Firewall
Sendmail
Webserver
NTP
Linux 2.4
Slackware
Pentium4
1.8Ghtz
512MB
400Mz FSB
~10/2001

No GUI

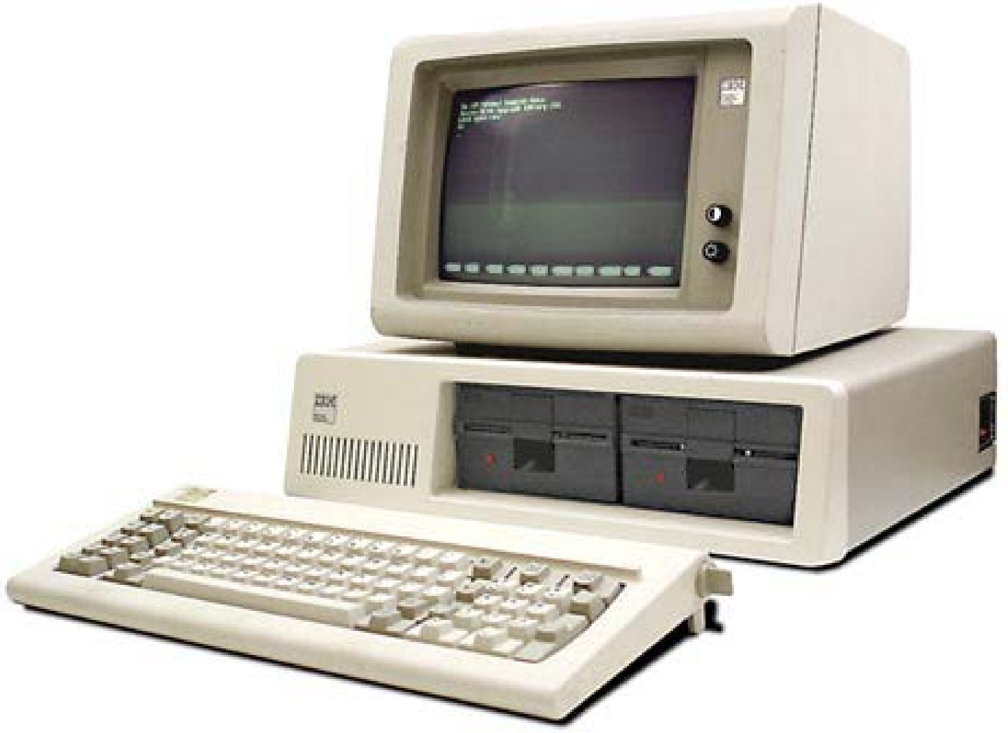


```
 23:53:47  up 130 days,  6:35,  1 user,  load average: 0.00, 0.00, 0.00
51 processes: 49 sleeping, 2 running, 0 zombie, 0 stopped
CPU states:   0.0% user   0.0% system   0.0% nice   0.0% iowait 100.0% idle
Mem:    515464k av,  464832k used,   50632k free,       0k shrd,   28356k buff
        237312k active,              88112k inactive
Swap:   248996k av,       0k used,  248996k free                  278096k cached

  PID USER      PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME CPU COMMAND
    1 root        8   0   240  240   208 S     0.0  0.0   0:06   0 init
    2 root        8   0     0    0     0 SW    0.0  0.0   0:00   0 keventd
    3 root       19  19     0    0     0 SWN   0.0  0.0   0:00   0 ksoftirqd_CPU0
    4 root        9   0     0    0     0 SW    0.0  0.0   0:00   0 kswapd
    5 root        9   0     0    0     0 SW    0.0  0.0   0:00   0 bdflush
    6 root        9   0     0    0     0 SW    0.0  0.0   0:02   0 kupdated
   10 root       -1 -20     0    0     0 SW<   0.0  0.0   0:00   0 mdrecoveryd
  174 root        9   0     0    0     0 SW    0.0  0.0   0:00   0 khubd
  794 root        9   0   612  612   532 S     0.0  0.1   0:02   0 syslogd
  797 root        9   0   460  460   404 S     0.0  0.0   0:00   0 klogd
 1022 root        6   0   544  544   484 S     0.0  0.1   0:00   0 inetd
 1025 root        9   0  1416 1416  1308 S     0.0  0.2   0:39   0 sshd
 1029 root       10   0 11124  10M  1968 S     0.0  2.1   2:40   0 named
 1039 root        4   0   592  592   520 S     0.0  0.1   0:00   0 crond
 1041 daemon      9   0   644  644   572 S     0.0  0.1   0:00   0 atd
 1044 root        9   0  2108 2108  1560 S     0.0  0.4   0:05   0 sendmail
 1047 smmsp       9   0  1968 1960  1496 S     0.0  0.3   0:00   0 sendmail
 1056 root        8   0  4504 4504  4368 S     0.0  0.8   0:12   0 httpd
 1058 root        9   0  1836 1836  1276 S     0.0  0.3   0:01   0 smbd
 1060 root        9   0  1664 1664  1132 S     0.0  0.3   0:16   0 nmbd
 1067 root        9   0   476  476   420 S     0.0  0.0   0:01   0 gpm
 1070 root        9   0  2220 2220  2000 S     0.0  0.4   0:07   0 ntpd
 1075 root        9   0   480  480   428 S     0.0  0.0   0:00   0 agetty
 1076 root        9   0   480  480   428 S     0.0  0.0   0:00   0 agetty
 1077 root        9   0   480  480   428 S     0.0  0.0   0:00   0 agetty
```

Performance

- Older systems have sufficient performance to run CLI-based operating systems
- Useful as test & production Internet servers
  - 80486 or newer
  - SBC
  - Embedded



Too old, insufficient performance

You Live In Interesting Times

- Processors speed used to double in 18 months (Moore’s Law), but bumped against the upper bound (due to thermal problems) went towards multi-core processors, i.e., parallelism to increase the processing power
- Disk storage doubling every 12 months
- Global bandwidth every 6 months
- What will the future OS be?
- Little incentive for efficiency
  - Peephole optimizer? What's that?

Windows Administration

Bring up a Command Prompt window

- whoami
- where
- where dwm.exe
    - [up arrow] [left arrows] where –T dwm.exe
    - (command history)
- pathping www.cisco.com
    - wait for n seconds

Windows Administration

- Fixable
- Right click on Window Title Bar
- Left click on Properties
- Screen Buffer Size
- Width 140 Height 200
- Window Size
- Width 110 Height 57
- OK

pathping www.cisco.com
    wait for n seconds

Windows Administration

- And just because we can....
- title [phrase of your choice]
- What changed?

Windows Administration

- Provides information about network latency and network loss at intermediate hops between a source and destination.
- Pathping sends multiple Echo Request messages to each router between a source and destination over a period of time and then computes results based on the packets returned from each router.
- Because pathping displays the degree of packet loss at any given router or link, you can determine which routers or subnets might be having network problems.

Windows Administration

- systeminfo
- Displays detailed configuration information about a computer and its operating system, including operating system configuration, security information, product ID, and hardware properties, such as RAM, disk space, and network cards
- Remote computer (with credential)
  - Hotfixes
  - VM's

Windows Administration

- tasklist
- Displays a list of applications and services with their Process ID (PID) for all tasks running on either a local or a remote computer.
- tasklist /m (module name)
- [up arrow] [space] | more
- tasklist /svc   List service information

- taskkill
- driverquery

Windows Administration

Windows Management Instrumentation Command-line

WMIC is based on aliases.

wmic /?	Get list of aliases
wmic	cpu
	pagefile
	process

exit

Spend some time exploring the wmic CLI.

Windows Administration

- netsh is a command-line scripting utility that allows you to, either locally or remotely, display or modify the network configuration of a computer that is currently running.
- netsh also provides a scripting feature that allows you to run a group of commands in batch mode against a specified computer.
- netsh can also save a configuration script in a text file for archival purposes or to help you configure other servers.
- If you have a DOS shell on a remote computer you can reconfigure the system as needed.

- netsh ENTER
- interface ipv4 ENTER
- show ENTER
- show tcpconnections ENTER
- show tcpstats ENTER
- quit ENTER

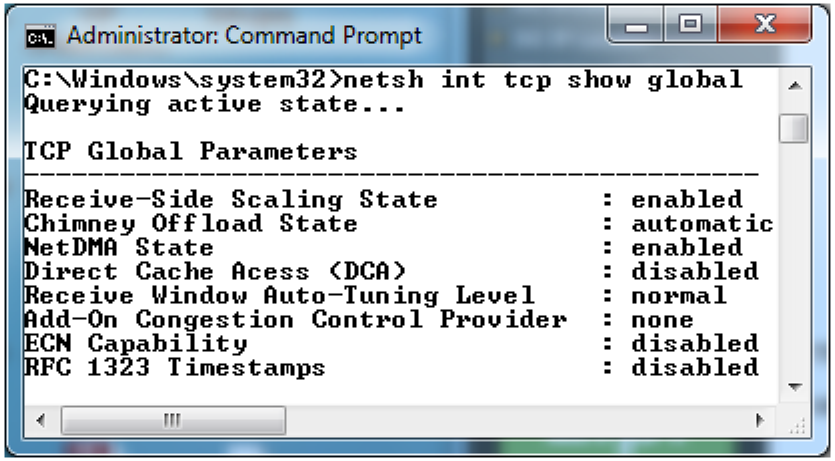
netsh is one of the reasons to make the command window larger.

Spend some time exploring netsh

System Tuning via DOS

- Some system parameters can be tuned from the CLI. One example:

  netsh int tcp show global



Administrator: Command Prompt

```
C:\Windows\system32>netsh int tcp show global
Querying active state...

TCP Global Parameters
----------------------------------------------
Receive-Side Scaling State          : enabled
Chimney Offload State               : automatic
NetDMA State                        : enabled
Direct Cache Acess (DCA)            : disabled
Receive Window Auto-Tuning Level    : normal
Add-On Congestion Control Provider  : none
ECN Capability                      : disabled
RFC 1323 Timestamps                 : disabled
```

- Add-On Congestion Control Provider

  Traditional slow-start and congestion avoidance algorithms in TCP help avoid network congestion by gradually increasing the TCP window at the beginning of transfers until either the TCP Receive Window boundary is reached or packet loss occurs. For broadband internet connections that combine high TCP Window with higher latency (high Bandwidth Delay Product), these algorithms do not increase the TCP windows fast enough to fully utilize the bandwidth of the connection.
- Compound TCP (CTCP) available since Vista and Server 2008 increases the TCP send window more aggressively for broadband connections (with a large TCP Receive Window and BDP) attempting to maximize throughput by monitoring delay variations and packet loss. It also ensures that its behavior does not impact other TCP connections negatively.
- By default, CTCP is turned off except for Server 2008. Turning this option on can significantly increase throughput and packet loss recovery.
- To enable CTCP, in elevated command prompt type:

```
netsh int tcp set global congestionprovider=ctcp
```

Windows Administration

- msconfig
  - Services
  - Startup
  - Tools
- Powershell
- A task-based command-line shell and scripting language designed especially for system administration

Windows Administration

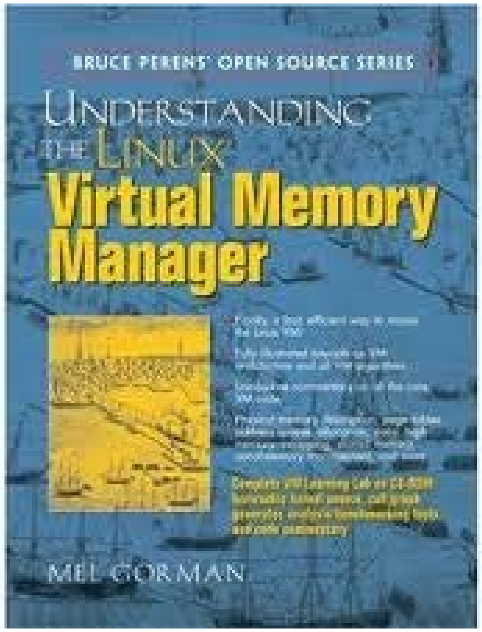
- Robocopy, or "Robust File Copy", is a command-line directory replication command.
- Can tolerate network interruptions and resume
- Skip junction points
- Multithreaded copying
- Folder oriented

- GUI versions are also available

Additional Resources

- http://sourceforge.net/projects/unixtop/
- http://technet.microsoft.com/en-us/sysinternals

- Books
- Understanding the Linux Virtual Memory Manager
- 729 pages 2004
- http://ptgmedia.pearsoncmg.com/images/0131453483/downloads/gorman_book.pdf


BRUCE PERENS' OPEN SOURCE SERIES
UNDERSTANDING THE LINUX Virtual Memory Manager
MEL GORMAN

Remember

- Virtual memory extends amount of Physical memory
- Linux: Total Memory = swap + RAM
- Swapping
  - Moving pages to and from memory
  - Page – block (unit) of RAM
- Page fault – data requested not in cache memory
- Kernel handles process management
- Process may have multiple threads
- Processes created by forking