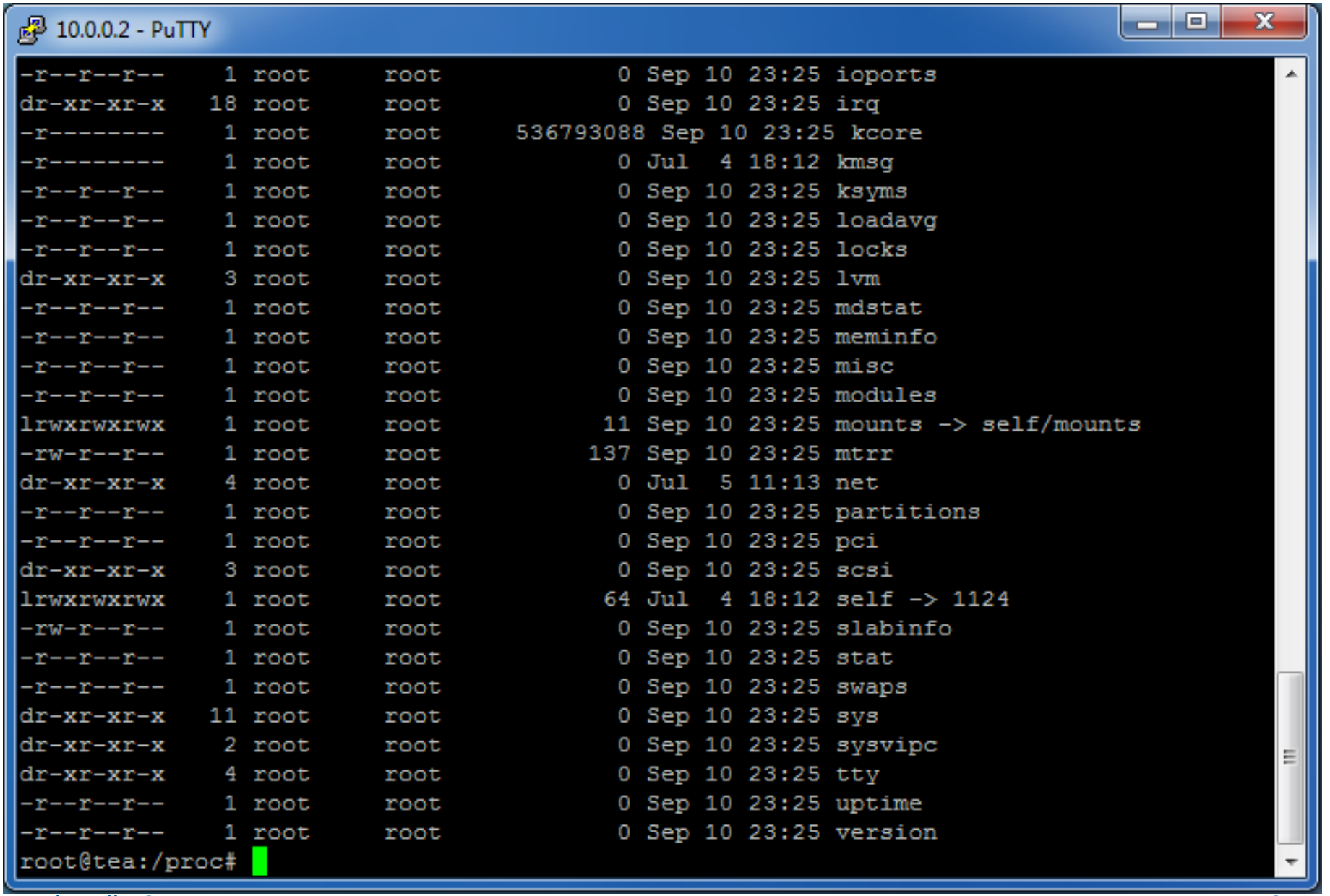
COMP 175

System Administration and Security

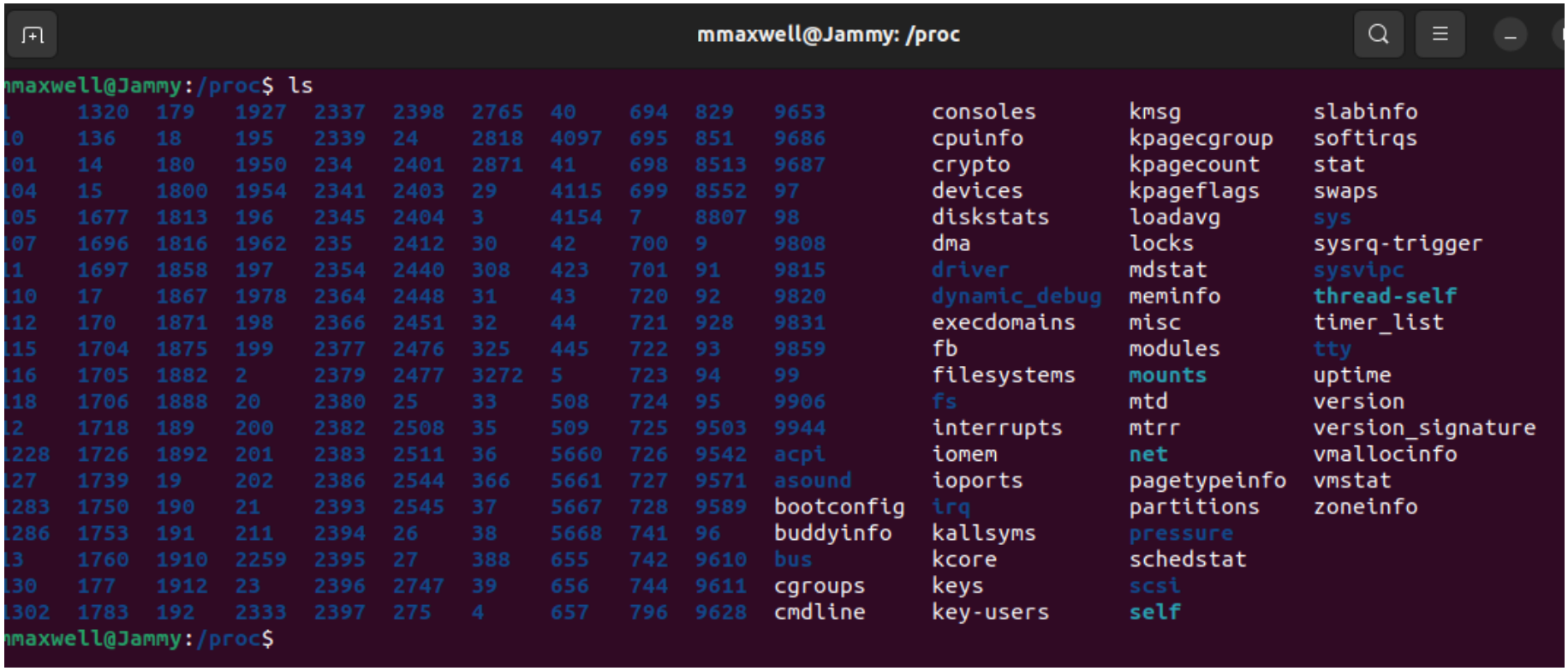
The /proc File System, Directories, Permissions

001000001 001000001 01010010 01010010 01010010

/proc


10.0.0.2 - PuTTY
-r--r--r-- 1 root root 0 Sep 10 23:25 ioports
dr-xr-xr-x 18 root root 0 Sep 10 23:25 irq
-r-------- 1 root root 536793088 Sep 10 23:25 kcore
-r-------- 1 root root 0 Jul 4 18:12 kmsg
-r--r--r-- 1 root root 0 Sep 10 23:25 ksyms
-r--r--r-- 1 root root 0 Sep 10 23:25 loadavg
-r--r--r-- 1 root root 0 Sep 10 23:25 locks
dr-xr-xr-x 3 root root 0 Sep 10 23:25 lvm
-r--r--r-- 1 root root 0 Sep 10 23:25 mdstat
-r--r--r-- 1 root root 0 Sep 10 23:25 meminfo
-r--r--r-- 1 root root 0 Sep 10 23:25 misc
-r--r--r-- 1 root root 0 Sep 10 23:25 modules
lrwxrwxrwx 1 root root 11 Sep 10 23:25 mounts -> self/mounts
-rw-r--r-- 1 root root 137 Sep 10 23:25 mtrr
dr-xr-xr-x 4 root root 0 Jul 5 11:13 net
-r--r--r-- 1 root root 0 Sep 10 23:25 partitions
-r--r--r-- 1 root root 0 Sep 10 23:25 pci
dr-xr-xr-x 3 root root 0 Sep 10 23:25 scsi
lrwxrwxrwx 1 root root 64 Jul 4 18:12 self -> 1124
-rw-r--r-- 1 root root 0 Sep 10 23:25 slabinfo
-r--r--r-- 1 root root 0 Sep 10 23:25 stat
-r--r--r-- 1 root root 0 Sep 10 23:25 swaps
dr-xr-xr-x 11 root root 0 Sep 10 23:25 sys
dr-xr-xr-x 2 root root 0 Sep 10 23:25 sysvipc
dr-xr-xr-x 4 root root 0 Sep 10 23:25 tty
-r--r--r-- 1 root root 0 Sep 10 23:25 uptime
-r--r--r-- 1 root root 0 Sep 10 23:25 version
root@tea:/proc#

/proc



- ls (not –al)
- Ubuntu desktop not (Slackware)

/proc

- /proc directory is a “pseudo-filesystem”
- /proc is the parent of a virtual filesystem
- The pseudo-files do not really exist on disk
- Information read from proc is generated by the kernel at boot time
- Files contain current settings used by kernel
- Easy way to exchange info
- Good for system reports
- Easy to create (see web)

Next: /proc and the file system

Linux /proc directory

- change to root:
  - sudo –I
  - su
- /proc/partitions contains partition settings
  - # clear ; cat /proc/partitions
  - # cat /proc/partitions

1. Examples are relative or absolute? (early)
2. Advantage of the above pathing
3. Examples differ in what ways?

cat/proc/ide/hda/

- /proc/ide information about IDE devices present
- /proc/ide/hda info on 1st bootable IDE hard disk
- cat /proc/ide/hda/ <- autospell /had %&!@
- media disk
- model FUJITSU MPD3084AT (8.4 GB)
- capacity 16514064 (in blocks)
- geometry physical 16383/16/63 (chs)
  logical 1027/255/63
- cache 512

- MTBF: 500,000 powered-on hours/24/365=57 yr.
- Listed on eBay as ‘Vintage’ (drive circa 1998)

/proc

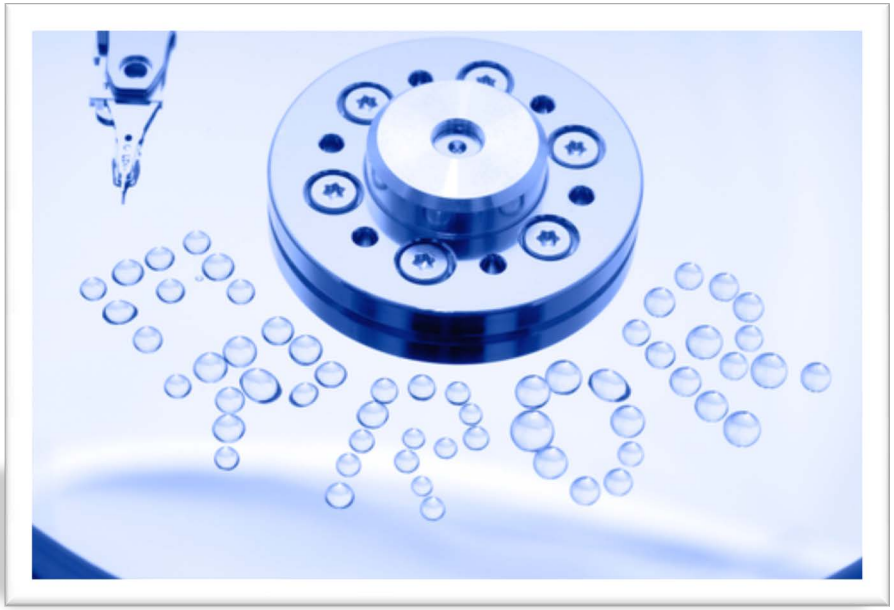
```
# cat /proc/partitions
major minor       #blocks  name
   3     0       8257032 hda
   3     1       8000338 hda1
   3     2        249007 hda2
```

Add the blocks=16506377

capacity said=16514064 - where are 7687 blocks?

fsck

- Overhead
- Boundary alignment
- Spares for recovery
- Marked as bad blocks
  - Not to be used



ide disks

- Geometry shows size in **c**ylinder/**h**ead/**s**ector
- Capacity shows size in **l**ogical **b**lock **a**ddressing
- Older CHS format limited to 1024 cylinders
- Old BIOS limited drives to 528 MB
- Can only describe disks up to 8G
- Modern disks almost always described using LBA
- Linux kernel uses LBA
- CHS parameters informational only
- Since OS uses LBA, PC BIOS must also use LBA

/proc

```
# cat /proc/swaps
Filename        Type            Size    Used    Priority
/dev/hda2       partition       248996  0       -1
```

- hda2 is the swap partition
- df shows:

```
# df -k
Filesystem    1K-blocks      Used  Available Use% Mounted on
/dev/hda1      7749536   3651968 3697552  50%   /
```

/proc/ide/hda

- -r--r--r-- 1 root root 0 Sep 11 11:25 cache
- -r--r--r-- 1 root root 0 Sep 11 11:25 capacity
- -r--r--r-- 1 root root 0 Sep 11 11:25 capacity
- -r--r--r-- 1 root root 0 Sep 11 11:25 driver
- -r--r--r-- 1 root root 0 Sep 11 11:25 geometry
- -r-------- 1 root root 0 Sep 11 11:25 identify
- -r--r--r-- 1 root root 0 Sep 11 11:25 media
- -r--r--r-- 1 root root 0 Sep 11 11:25 model
- -rw------- 1 root root 0 Sep 11 11:25 settings
- -r-------- 1 root root 0 Sep 11 11:25 smart_thresholds
- -r-------- 1 root root 0 Sep 11 11:25 smart_values

- BIOS Cylinders/Heads/Sectors for translation
- DMA 32-bit IO max kb request write cache acoustic

/proc

- # cat /proc/mounts (reformatted)

```
rootfs     /              rootfs rw 0 0
/dev/root  /                ext2 rw 0 0
devpts     /dev/pts       devpts rw 0 0
proc       /proc           proc  rw 0 0
usbfs      /proc/bus/usb usbfs rw 0 0
```

- Device that is mounted
- mount point
- file system type
- read-only (ro) or read-write (rw)
- dummy values to match format of /etc/mtab

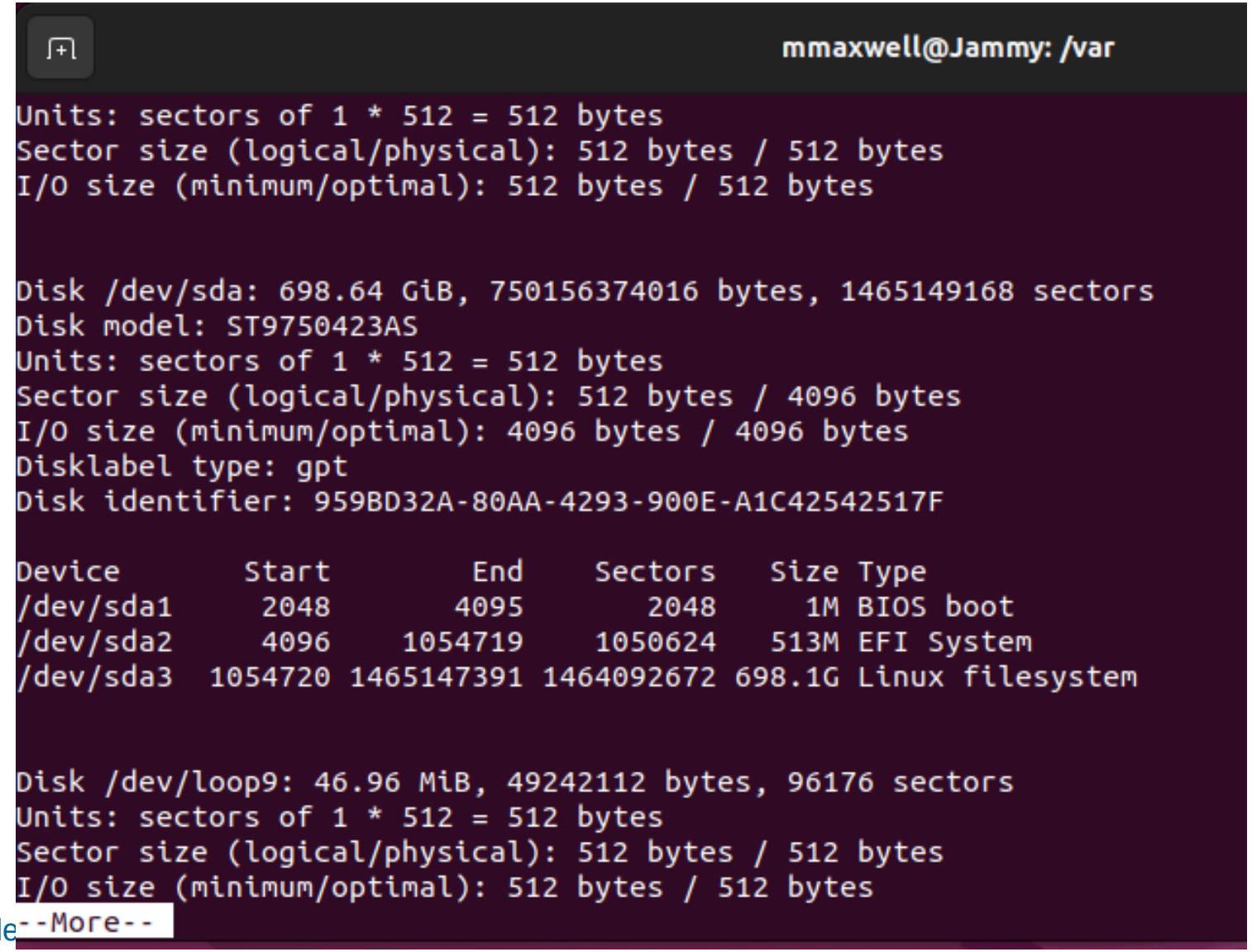
BTW - 8 GB for a Server?

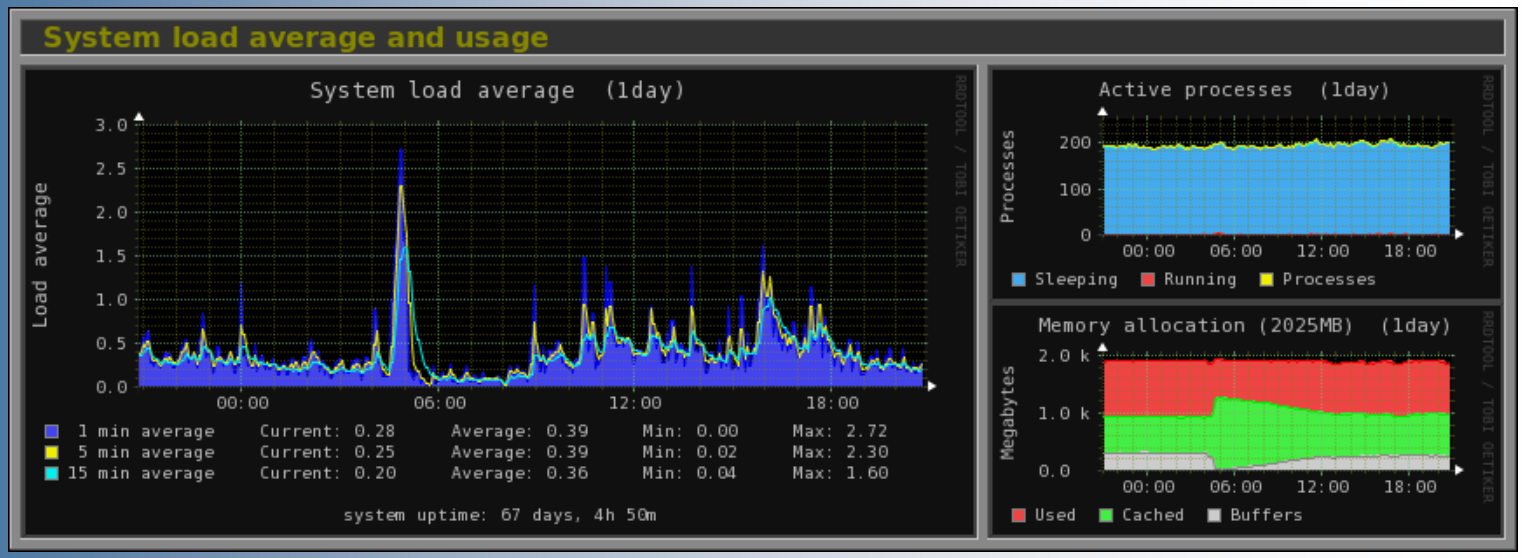
- firewall
- email
- web
- dns
- dhcp
- syslog

- This system replaced a 486
- Text-based services are not CPU intensive
- IDE? (See next slide)

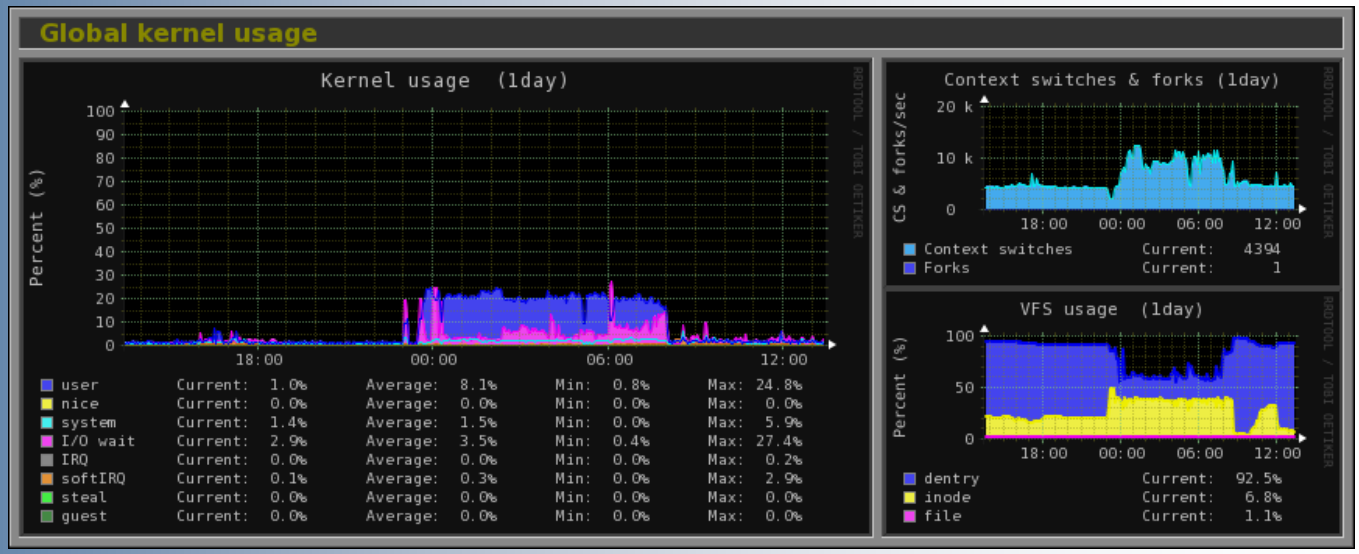
sda

- SATA connected drive (or SCSI)



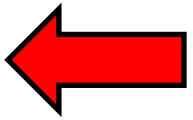


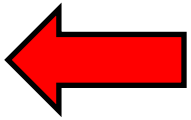
/proc Revisited

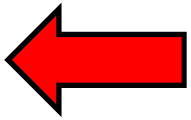


clear ; cat /proc/cpuinfo

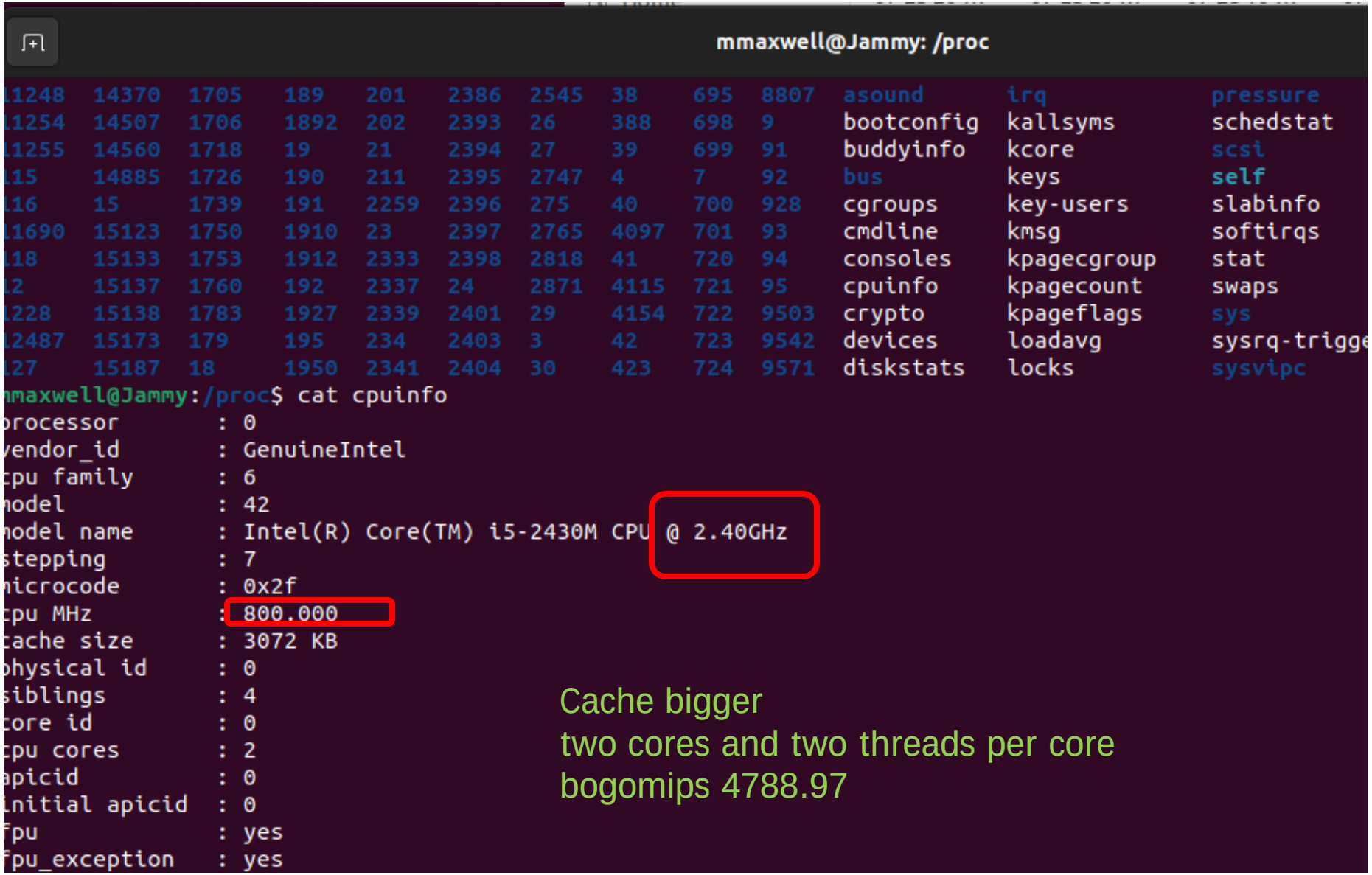
```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 15
model           : 2
model name      : Intel(R) Pentium(R) 4 CPU 1.80GHz
stepping        : 4
cpu MHz         : 1816.230
cache size      : 512 KB
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 2
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep
   mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2
   ss ht tm
bogomips        : 3617.58
```







cat /proc/cpuinfo



Cache bigger
two cores and two threads per core
bogomips 4788.97

/proc/meminfo

```
root@tea:/proc# cat meminfo
        total:    used:    free:  shared: buffers:  cached:
Mem:  527835136 455454720 72380416         0 28106752 270102528
Swap: 254971904         0 254971904
MemTotal:         515464 kB
MemFree:           70684 kB
MemShared:             0 kB
Buffers:           27448 kB
Cached:           263772 kB
SwapCached:            0 kB
Active:           227924 kB
Inactive:          81556 kB
HighTotal:             0 kB
HighFree:              0 kB
LowTotal:         515464 kB
LowFree:           70684 kB
SwapTotal:        248996 kB
SwapFree:         248996 kB
root@tea:/proc#
```

/proc/net

10.0.0.2 - PuTTY

```
dr-xr-xr-x   64 root     root            0 Jul  4 11:12 ..
-r--r--r--    1 root     root            0 Sep 10 23:30 arp
-r--r--r--    1 root     root            0 Sep 10 23:30 dev
-r--r--r--    1 root     root            0 Sep 10 23:30 dev_mcast
dr-xr-xr-x    2 root     root            0 Sep 10 23:30 drivers
-r--r--r--    1 root     root            0 Sep 10 23:30 ip_conntrack
-r--r--r--    1 root     root            0 Sep 10 23:30 ip_tables_matches
-r--r--r--    1 root     root            0 Sep 10 23:30 ip_tables_names
-r--r--r--    1 root     root            0 Sep 10 23:30 ip_tables_targets
-r--r--r--    1 root     root            0 Sep 10 23:30 mcfilter
-r--r--r--    1 root     root            0 Sep 10 23:30 netlink
-r--r--r--    1 root     root            0 Sep 10 23:30 netstat
-r--r--r--    1 root     root            0 Sep 10 23:30 packet
-r--r--r--    1 root     root            0 Sep 10 23:30 psched
-r--r--r--    1 root     root            0 Sep 10 23:30 raw
-r--r--r--    1 root     root            0 Sep 10 23:30 route
dr-xr-xr-x    2 root     root            0 Sep 10 23:30 rpc
-r--r--r--    1 root     root            0 Sep 10 23:30 rt_cache
-r--r--r--    1 root     root            0 Sep 10 23:30 rt_cache_stat
-r--r--r--    1 root     root            0 Sep 10 23:30 snmp
-r--r--r--    1 root     root            0 Sep 10 23:30 sockstat
-r--r--r--    1 root     root            0 Sep 10 23:30 softnet_stat
-r--r--r--    1 root     root            0 Jul  5 11:13 tcp
-r--r--r--    1 root     root            0 Sep 10 23:30 tr_rif
-r--r--r--    1 root     root            0 Sep 10 23:30 udp
-r--r--r--    1 root     root            0 Sep 10 23:30 unix
-r--r--r--    1 root     root            0 Sep 10 23:30 wireless
root@tea:/proc/net#
```

proc

- 0 byte file sizes? proc registers itself to the Virtual File System layer (VFS). When VFS make calls to it requesting i-nodes for files/directories, the /proc file system creates those files/directories from information within the kernel.

```
# ls -al cpuinfo
-r--r--r-- 1 root root 0 Sep 10 00:27 cpuinfo

# file cpuinfo
cpuinfo: empty

# cat cpuinfo
processor       : 0
vendor_id       : GenuineIntel
model name      : Intel(R) Pentium(R) 4 CPU 2.00GHz
```

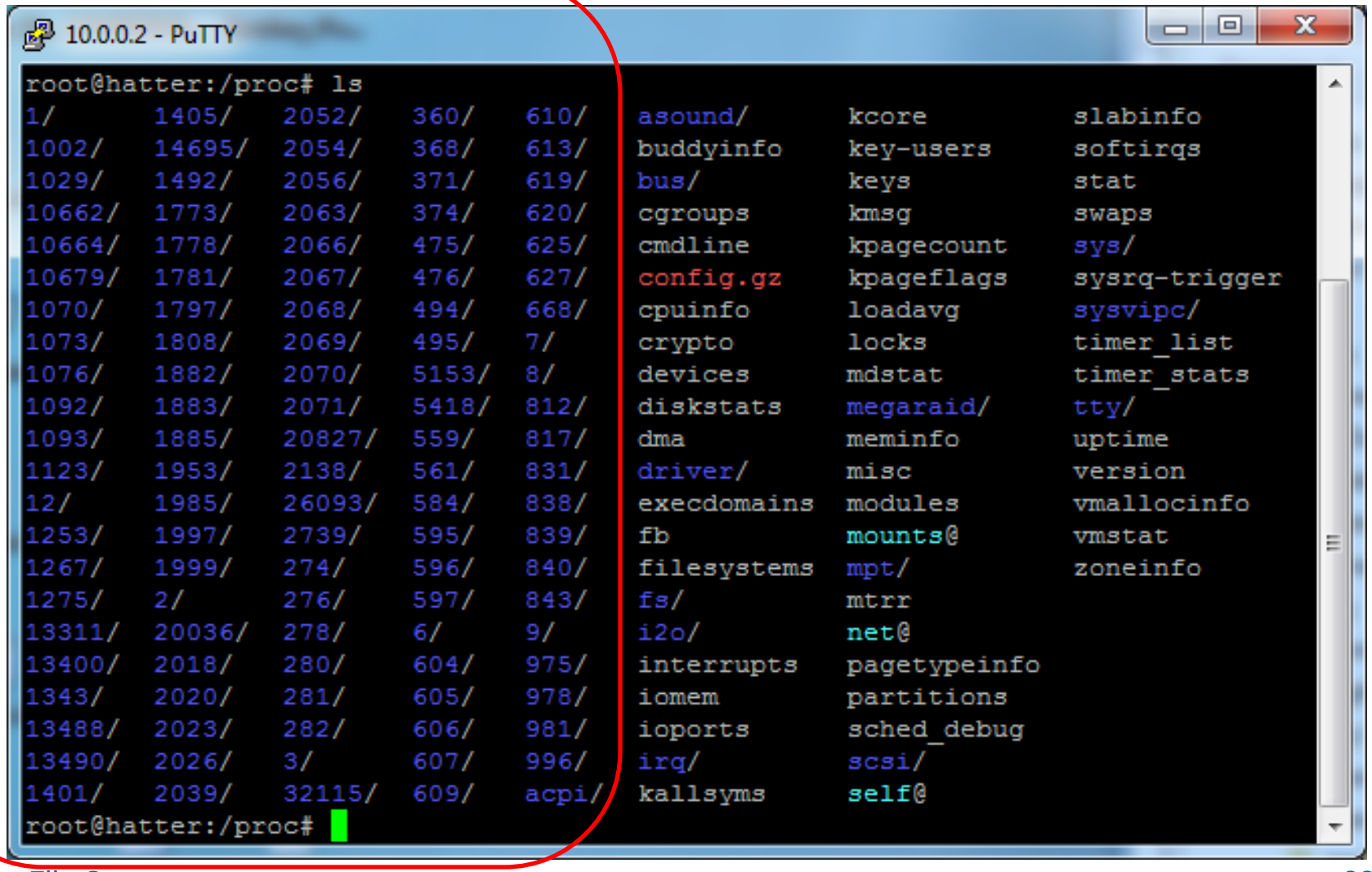
proc/sys

```
dr-xr-xr-x   0 root root 0 Sep 10 00:34 debug/
dr-xr-xr-x   0 root root 0 Sep 10 00:34 dev/
dr-xr-xr-x   0 root root 0 Sep 10 00:34 fs/
dr-xr-xr-x   0 root root 0 Jun 10 07:42 kernel/
dr-xr-xr-x   0 root root 0 Sep 10 00:34 net/
dr-xr-xr-x   0 root root 0 Sep 10 00:34 sunrpc/
dr-xr-xr-x   0 root root 0 Sep 10 00:34 vm/
```

Writing to these files can change the state of the kernel
Changes to these files should be made with caution

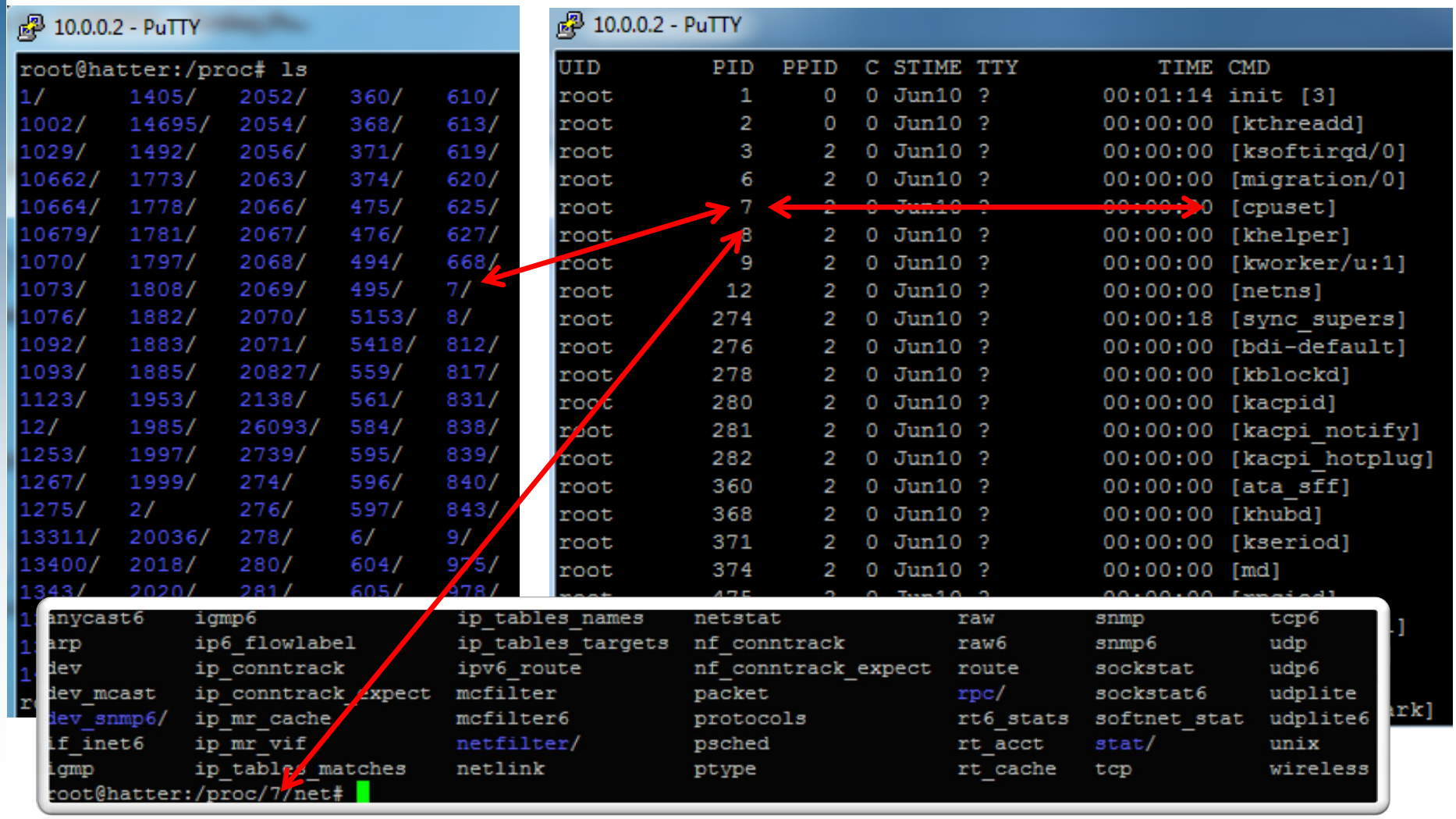
/proc

- Numbers?



/proc

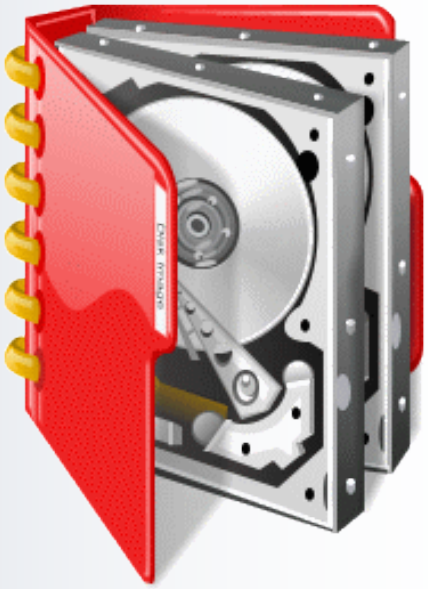
- PID's Process Identifier



/proc

- Can be used for system monitoring monitorix.org

The File System: Directories



I've miles and miles of files
Pretty files of your forefather's fruit
and now to suit our great computer,
Your magnetic ink.
- In the Beginning, the Moody Blues

File Systems

"On a UNIX system, everything is a file;
if something is not a file, it is a process."

Linux systems make no difference between a file and a directory, a directory is just a file containing names of other files. Programs, services, texts, images, etc., are all files

Input and output devices, generally all devices, also considered to be files

The file system is comprised of:

- User data - the actual data contained in files
- Metadata - structural information such as superblock, inodes, directories

Directories, Files, & Inodes

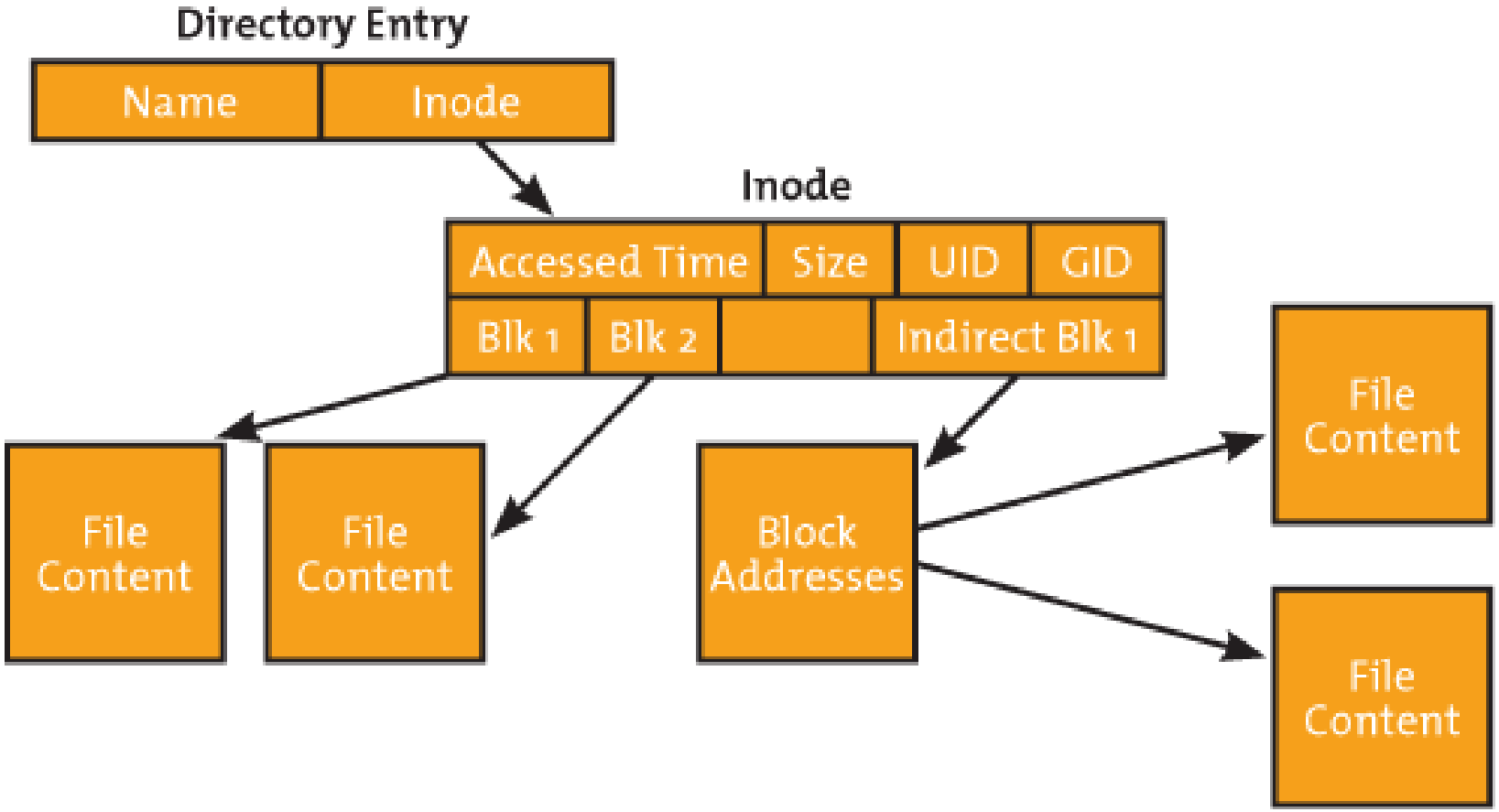
- Every directory & file is listed in its parent directory
- Root directory, that parent is itself
- Directory is file with a table listing its' files, giving file names to the inode numbers in the list
- Information about all files and directories is maintained in INODE TABLE (Index Node)
- An Inode is a table entry containing information about a file (metadata) including file permissions, UID, GID, size, time stamp, pointers to files data blocks on the disk etc.
- Inodes are all the same size, size is defined in superblock
- One inode allocated to each file/directory
- Everything has at least one primary inode
- Each inode has a unique sequential number

inodes

Metadata inside the inode (Index Node):

- File size
- Ownership indicator (like GUID number)
- MAC (Modify, Access, Change) & deletion times
- Mode field — typically contains file or link type
- Basic permissions rwx for ugo (more later)
- Link Count — number of hard linked directory entries with this inode as head node for a file
- Pointers to file data blocks on the disk

inodes



Relationship between a Directory Entry, an INODE, and the allocated blocks of a file

File System Structure

Deleting

- Given that Unix stores a file's administrative information (its physical location on disk, permissions, ownership, and modification times) in an inode
- *And* the file name (link) is stored in the contents of a directory entry
- *Then* deleting a file consists of removing the link to the inode (the inode itself is not deleted)

File System Structure

Data Recovery?

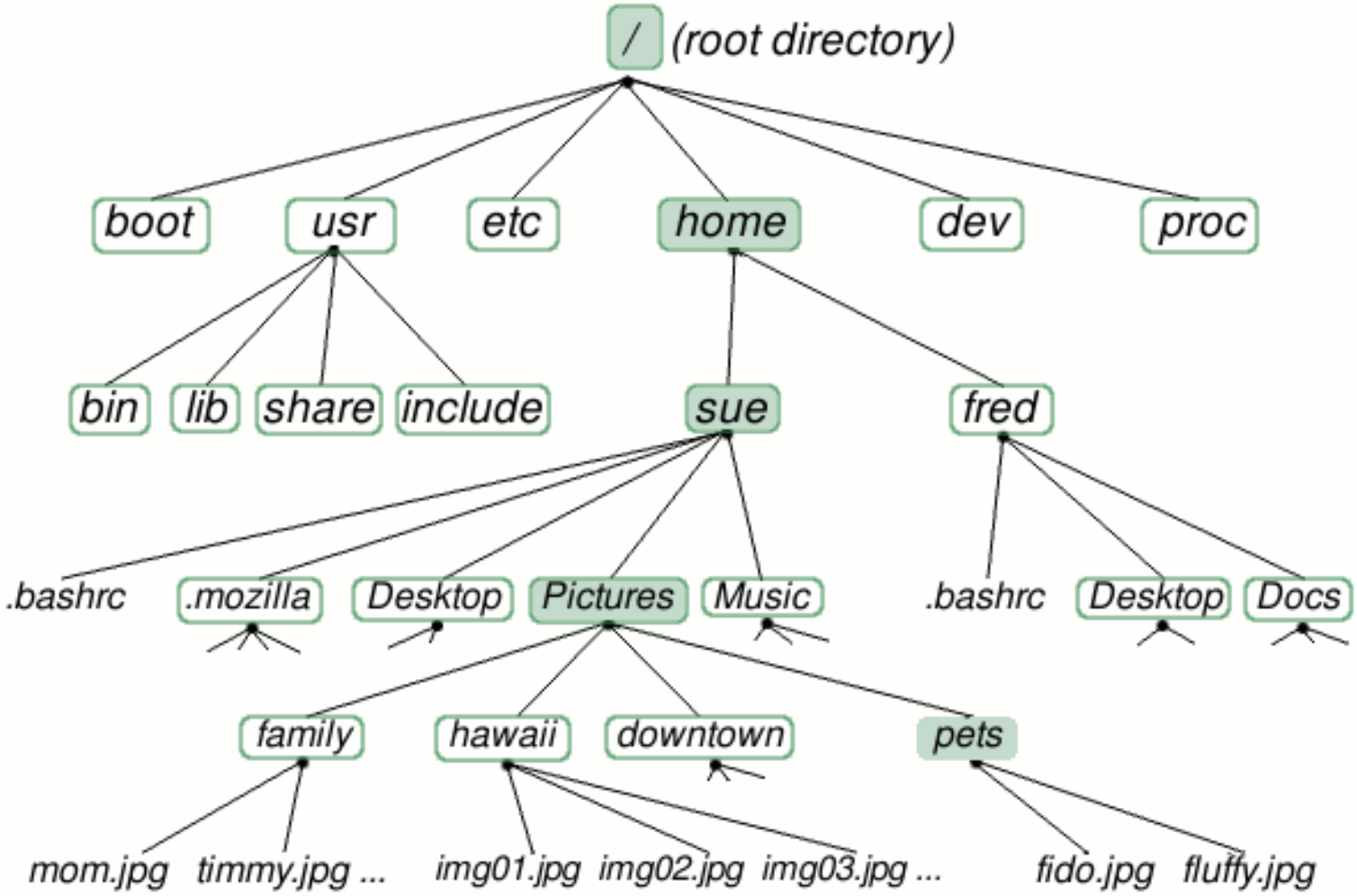
- When a file is deleted - the number of links to the inode is reduced by one
  - inode may have more than one link (or name)
- If number of links becomes zero, kernel may reuse the disk space making recovery difficult
- Magnetic Force Microscopy (MFM) can recover most data unless wipe is used

```
$ wipe -kD /dev/sdb1
```

Linux Directory Structure

- File system is similar to inverted tree structure
- The directory tree, a hierarchical structure, begins at a special directory called the root, or ***/***
- Root directory typically shown at the top
- Sub-directories branch out below ***/***
- Each node is either a file or a directory of files
- Directory paths separated by a forward slash: ***/***
- Exact opposite of DOS and Windows ****

Upside Down Tree



Don't confuse root (directory) with root (superuser)

Linux Directory Structure

- Specify a file or directory by its path name, either:
  - The full, or absolute, path name, or
  - The path relative to a location
- Full path name starts with the root **/**
  - and uses / as separaters
  - absolute example: /usr/local/bin/xntp
- Relative paths start in the current directory
  - example: log/syslog (you're currently in /var)

- Understand! absolute vs relative addressing

Not to be confused with..

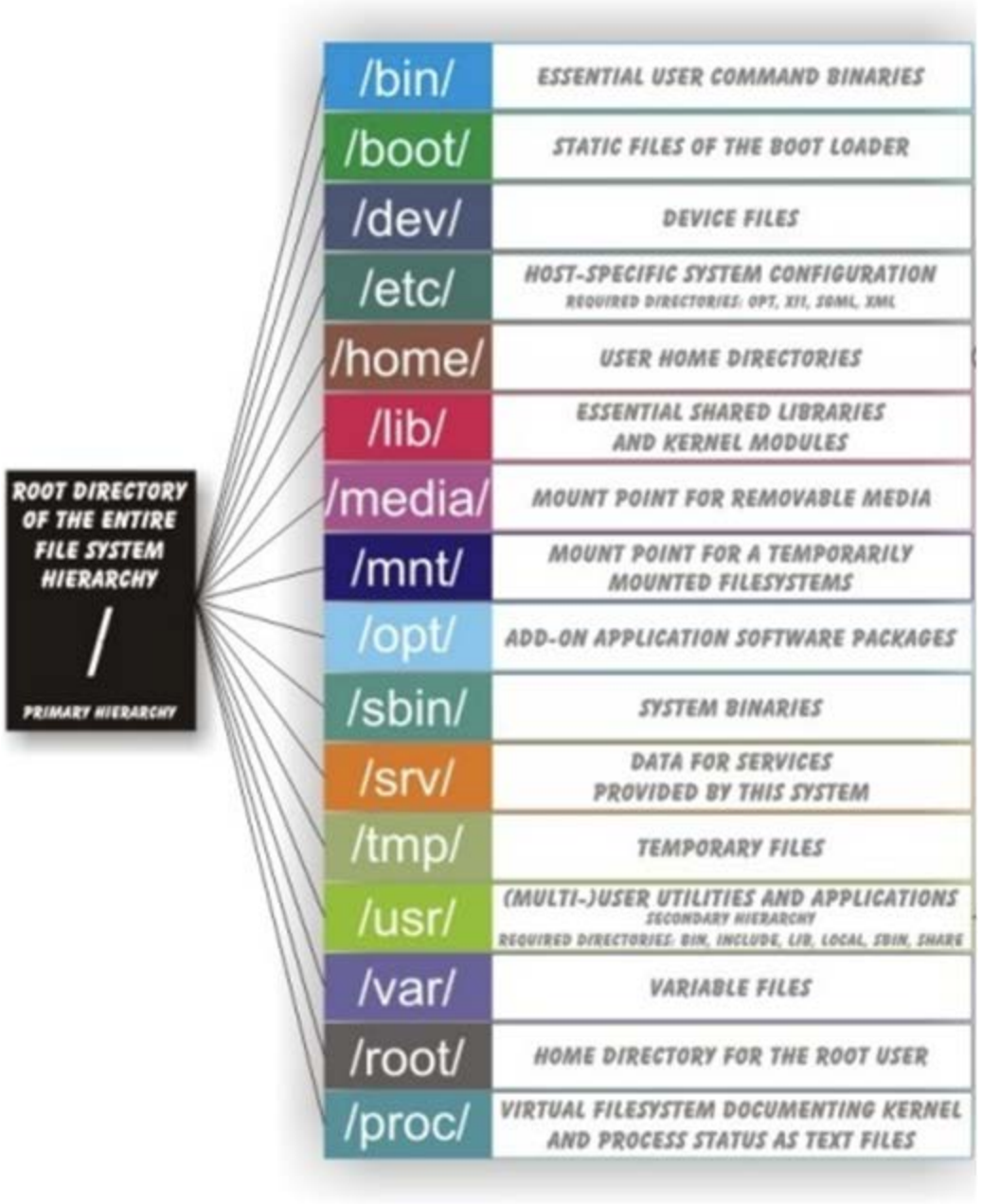
Understanding the difference between absolute and relative addressing is critical to running the system and services.

Hackers exploit system pathing errors on a regular basis.

Linux Directory Structure

There are other ways of looking at the file system structure than the inverted tree.

This graphic shows the convention behind the directory names.



Linux Directory Structure

- A **relative path name** specifies the path relative to another, usually the current working directory
  Two special directories:
  - . the current directory
  - .. the parent of the current directory
- At /usr/home/me a path in a relative way is:
  - ../usr/src/xntp
- e.g first go up one directory level, then down through the usr directory, followed by src directory, then to xntp

Linux Directory Structure

Other directory navigational conventions

- Your home directory "**~**" is where your personal files are located, and where you start at logon
  - Example: /home/mmaxwell
  - Path shortcut ~mmaxwell/
- Directory shortcuts to remember
  - ~ Your home directory
  - .. The parent directory
  - . The current directory

Linux File Types

- Unlike Windows, Unix file types (e.g. executable, data, text etc,) are not determined by file extension (e.g. exe, .dat, .txt)

There are seven basic types of file types in Linux.

- Regular Files -
- Directories d
- Character Device Files c
- Block Device Files b
- Local Domain Sockets s
- Named Pipes p
- Symbolic Links l

Linux File and Directory Names

- Up to 256 characters long
- Case sensitive – (a mixed blessing/curse)
- Filename permitted characters:
- letters and numbers
- hyphens - underscores _ dots .
  - No file called . or ..
    - . current directory .. parent directory
- Can't/not recommended: * ? $ & {} [] () etc.
- Highly not recommended: Spaces
- Can't use / (is a path separator)

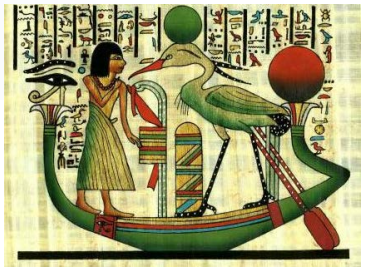
Metacharacters

*	Matches any sequence of zero/more characters except . at beginning of filename
?	Matches any single character
[a-z]	Matches any lower-case letter
[A-F]	Matches any upper-case letter from A to F
[0-9]	Matches any single digit
[a-zA-Z0-9]	Matches any digit or letter (any case)

Hyphens In Names - Bad

- ls > –junk
- rm –junk
- invalid option –j
  - whoops
  - how to get rid of?
- rm ./ –junk
- ReMoves a file, without a possibility of “undelete!”

Characters From Heck



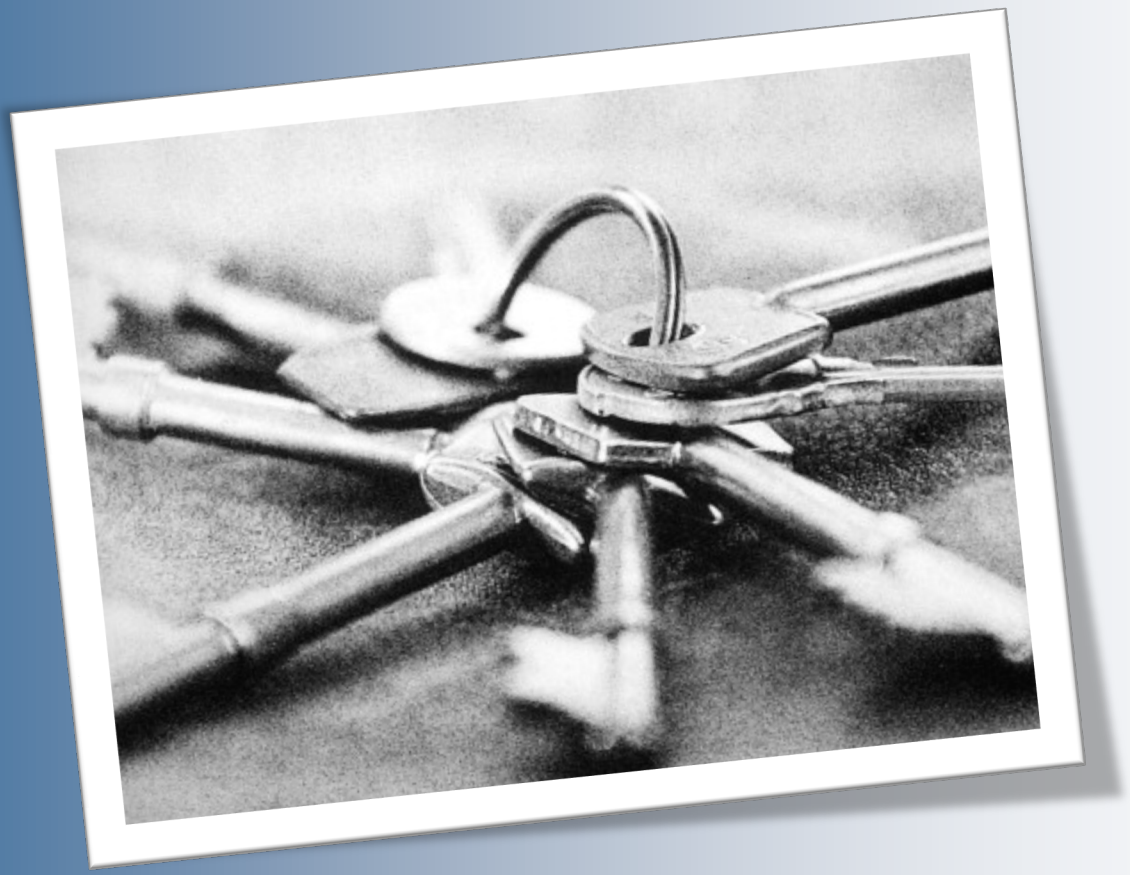
File names

- Ancient time: 8 characters.3 char extension
- All upper case, case insensitive
- Windows since NT: 127 characters, case sensitive
- Unix: 255 characters, case sensitive

- Moving files between systems can present some interesting issues:

```
-rwxr--r--  91964315 Turner\ Road\ v4\ on\ Vimeo.mp4
```

The \ escapes the significance of the space



PERMISSIONS

ROOT, MAY I?

User & Group Permissions

- In UNIX/LINUX, there is a concept of user and an associated group
- The system determines whether or not a user or group can access a file or program based on the permissions assigned to them
- There is a special user called Super User or root which has permission to access any file and directory

Access Permissions

- There are three permissions for any file, directory or application program:

  r Indicates that a given category of user can read file

  w Indicates that a given category of user can write to file

  x Indicates that a given category of user can execute file

- Each of the three permissions are assigned to three defined categories of users:

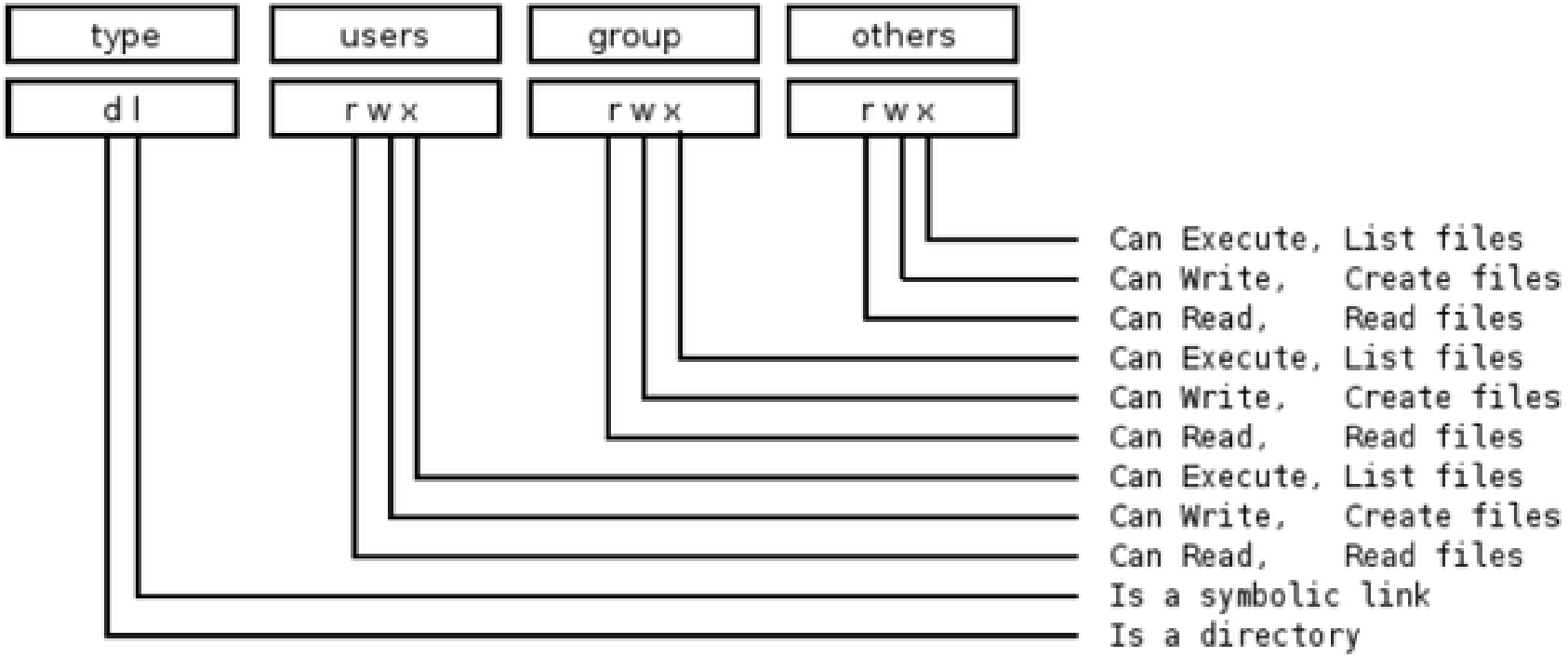
  user The user (owner) of the file or application

  group The group owning the file or application

  others All users with access to the system

Access Permissions

- The permissions for a file are listed are listed at the start of the line, starting with rwx
- The first set of symbols define owner access
- The next set of rwx symbols define group access
- The last set define access permitted all other users



Access Permissions

- One can easily view the permissions for a file by invoking a long format listing using the command: ls –l

```
drwxr-xr-x  8 root root 4096 Mar  2  2010 cache/
drwxr-xr-x  4 root root 4096 Mar 24  2011 db/
drwxr-xr-x  2 root root 4096 Feb  7  2011 empty/
```

Permissions

- Permissions on a Unix-like system are not inherited. Files created within a directory will not necessarily have same permissions as directory
- Octal notation

`-rwxrwxrwx` represented as 777

`-rwxr-xr-x` represented as 755

`-rw-rw-r--` represented as 664

`-r-x------` represented as 500

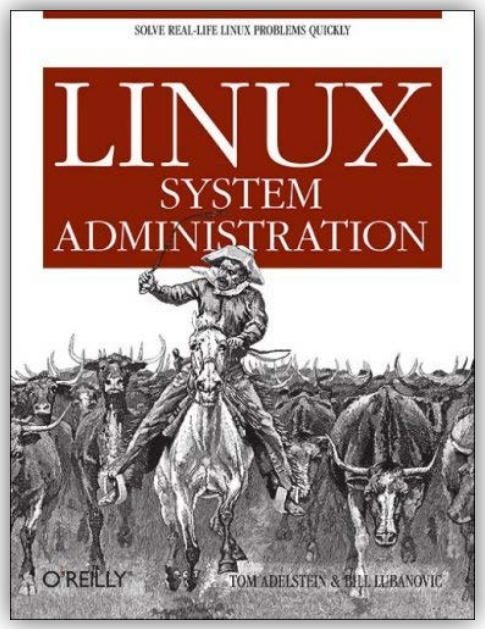
Octal

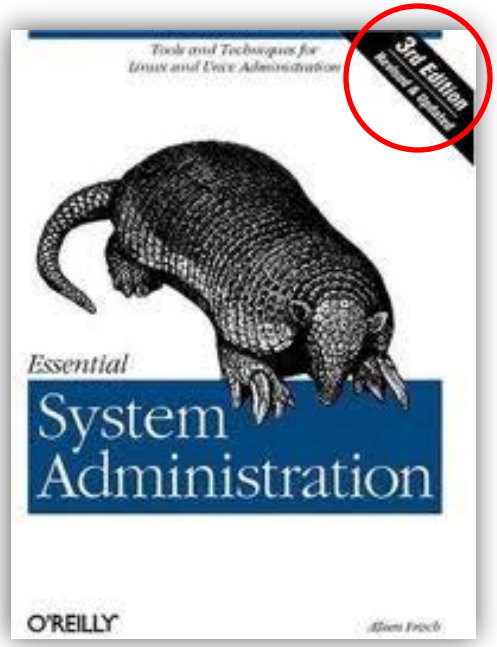
- `-rwxr-xr--`

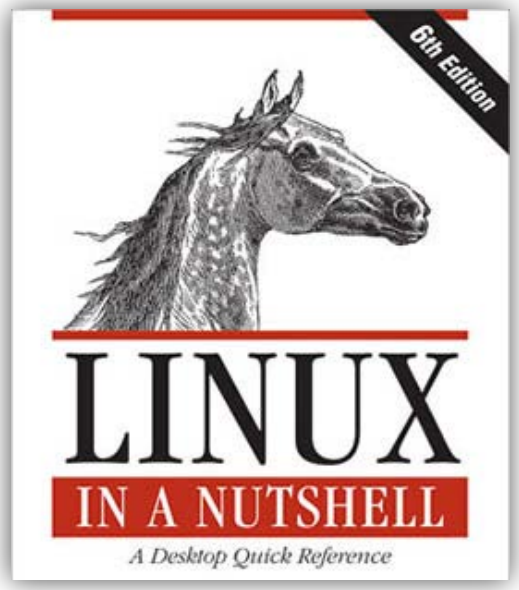
	u			g			o		
	7			5			4		
access	r	w	x	r	w	x	r	w	x
binary	4	2	1	4	2	1	4	2	1
enabled	1	1	1	1	0	1	1	0	0
result	4	2	1	4	0	1	4	0	0
total	7			5			4		

Additional References

- http://tldp.org/LDP/sag/html/index.html
- http://www.yolinux.com/TUTORIALS/LinuxTutorialSysAdmin.html







Remember

- proc – pseudo file system
- On a UNIX system, everything is a file
  - If something is not a file, it is a process
- File system comprised of user data and metadata
- PID Process Identifier
- Inodes (Index node) – info about a file system object
- absolute vs relative addressing
- ~ Your home directory
- .. The parent directory
- . The current directory
- rwx ugo