

```
$ linux 101 |
```

Presented by: Shanelle Ileteo



\$ what are we learning today?

- An introduction to Linux
- Linux filesystem
- The terminal
- Basic Linux commands
- Some tips and tricks
- User and group management
- File and owner permissions
- Networking commands
- Services and processes commands
- Some Linux security tips along the way!



\$ a brief **introduction** to answer
all your burning questions |



\$ *what* is Linux?

- Open-source operating system
- Different distributions include...
 - Ubuntu
 - CentOS
 - Arch Linux
 - Debian
 - Fedora
 - Linux Mint
 - Red Hat Enterprise Linux
 - Slackware Linux
 - And many more!



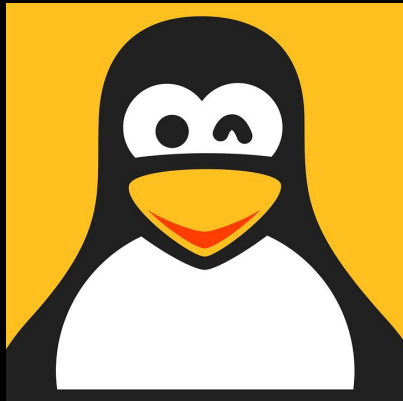
\$ **where** is Linux used?

- Software Development
 - Embedded Systems
 - Supercomputing
 - LAMP stack and web development
 - And much more!
-
- Used in both business settings and schools



\$ *when* did Linux start?

- **1991**: Linus Torvalds develops Linux as a personal project in Finland
- **1992**: Linux gets released online for free
- **1996**: Linux Mascot is created



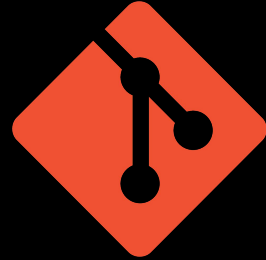
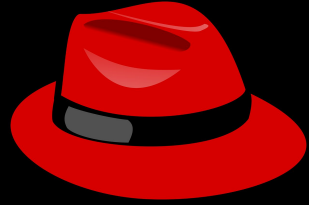
His name is...

Torvalds **U**nix aka **TUX**!



\$ **when** did Linux start?

- **2002:** Red Hat Enterprise Linux released
- **2005:** Linus Torvalds created Git to maintain Linux kernel
- **2009:** Google announced Chrome OS based on Linux kernel
- **2013:** Valve released SteamOS based on Debian (Linux distro)



\$ **why** use Linux?

PROS



- FREE!
- Open source community
- Highly secure

CONS



- Confusing for beginners / not UI friendly
- Games :- (



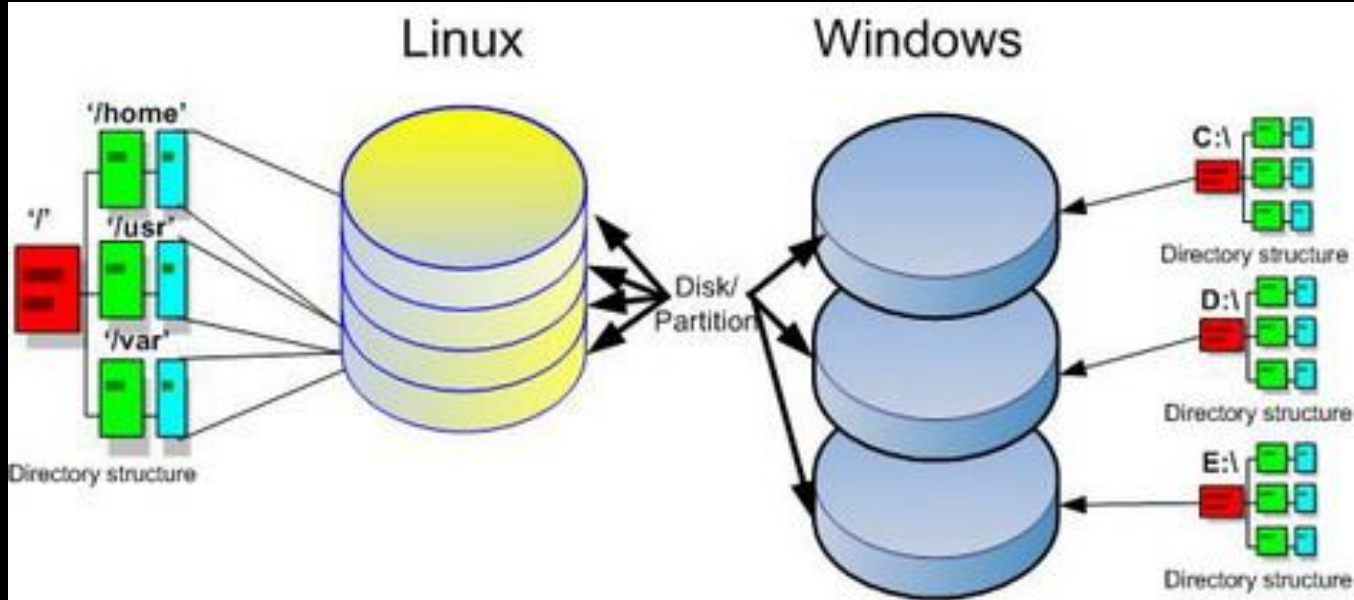
\$ *how* do I Linux?



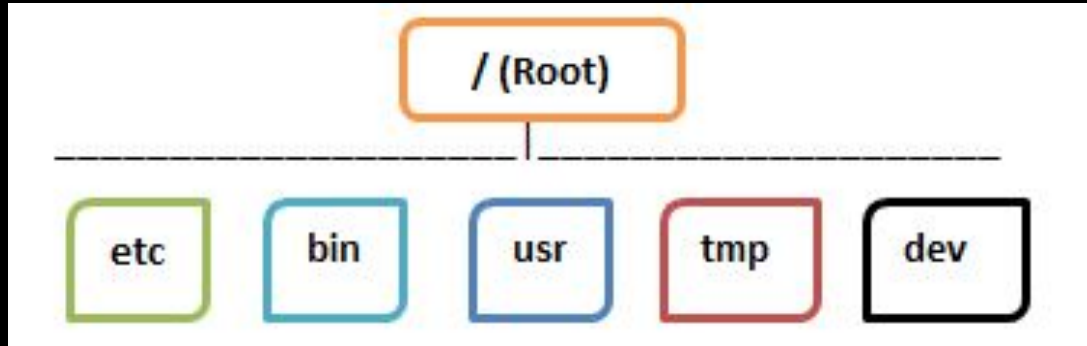
\$ understanding the **filesystem** |



\$ filesystem comparison



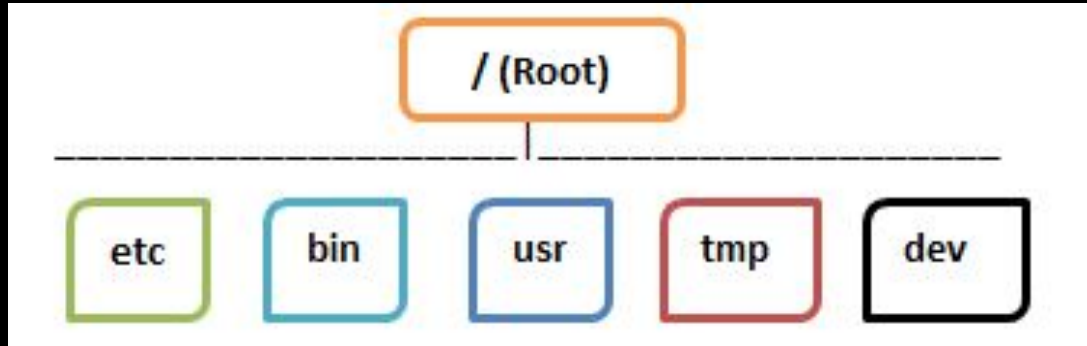
\$ an overview of the filesystem



- `/ (root)` - root directory of the entire system hierarchy
- `/etc` - host-specific system-wide configuration files



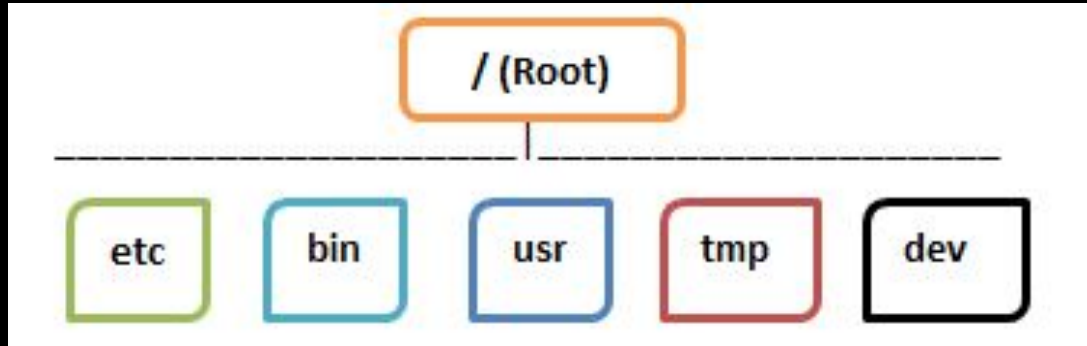
\$ an overview of the filesystem



- `/bin` - essential user command binaries
- `/usr` - user utilities and applications



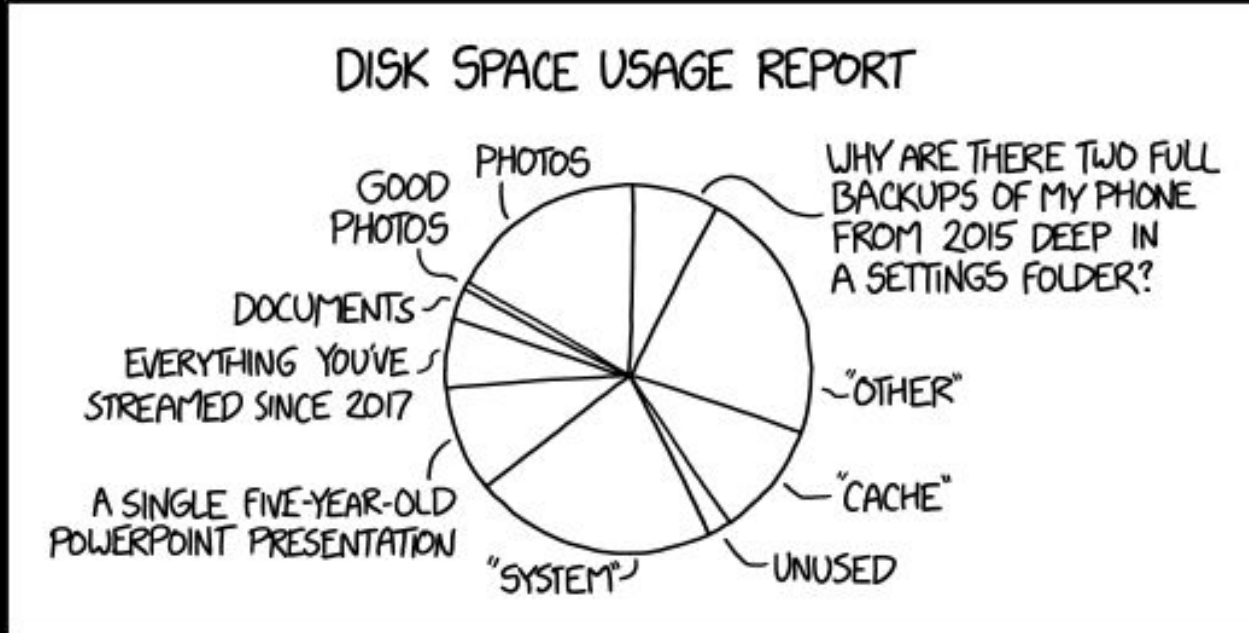
\$ an overview of the filesystem



- `/tmp` - temporary files
- `/dev` - essential device files attached to the system



\$ Security Tip: Follow partitions and use backups

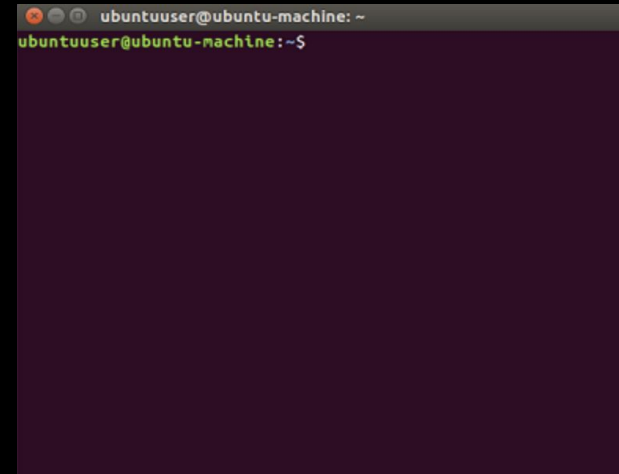


\$ navigating your way through the
terminal |



\$ the what now?

- An interface where you can type and execute text-based commands
- Can be used to make your life easier!
 - Not always given the UI (ie remote connecting)
 - Powerful commands that can execute tasks faster and more efficiently



\$ some basics about the terminal

```
ubuntuuser@ubuntu-machine: ~  
ubuntuuser@ubuntu-machine:~$
```

- ubuntuuser = username
- ubuntu-machine = hostname
- ~ = current working directory
- \$ (No) = superuser

Wait...superuser?



\$ some basics about the terminal

```
ubuntuuser@ubuntu-machine: ~  
ubuntuuser@ubuntu-machine:~$
```

- ubuntuuser = username
- ubuntu-machine = hostname
- ~ = current working directory
- \$ (No) = superuser

Change to superuser with **sudo su**



\$ some basics about the terminal

```
root@ubuntucient: /home/ubuntucient
File Edit View Search Terminal Help
ubuntucient@ubuntucient:~$ sudo su
root@ubuntucient:/home/ubuntucient#
```

- root = username
- ubuntucient = hostname
- /home/ubuntucient = current working directory
- # (Yes) = superuser



\$ ~~Security~~ Tip: Don't always run
as root

I AM ROOT.

OK.

I AM ROOT.

OK.

rm -rf /



NO!!

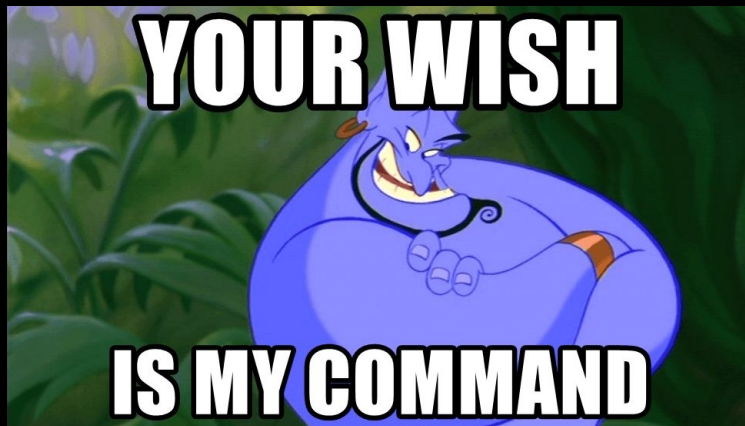


\$ learning the **basics** |



\$ about commands

- Commands are your way of communicating with your computer
- Three components to a command...
 - Utility (required)
 - Flag
 - Argument



```
$ pwd
```

- **pwd** = "Print working directory"
- It tells you where you are

```
$ pwd  
/home/ubuntuclient
```




```
$ ls
```

- **ls** = "List"
- It lists out what's in your folder
- Use flags to list more things...
 - **-a** : hidden files (starting with ".")
 - **-l** : long format (with permissions)
- Can combine flags (ie **-la**)
- Can also list parent directory (**ls ..**), root directory (**ls /**) and user's home directory (**ls ~**)



```
$ cd
```

- **cd** = "change directory"
- It lets you move from one folder to another
- Can change to the parent directory, root directory, and user's home directory
 - Anyone remember how?



\$ cd

- **cd** = "change directory"
- It lets you move from one folder to another
- Can change to the parent directory, root directory, and user's home directory
 - Anyone remember how?
 - Using `cd ..` `cd /` and `cd ~`



\$ echo and cat

- **echo** lets you display text in the terminal

```
$ echo hello world  
hello world
```

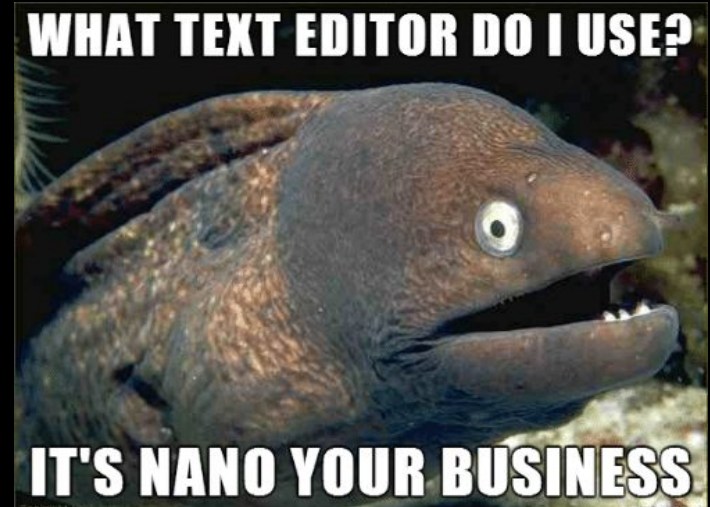
- **cat** = "concatenate"
- It lets you display text from files

```
$ cat example.txt  
This is an example file.
```



\$ vi, gedit, emacs, nano, etc

- Text editors to edit files
- All programmers have different preferences
 - vi is pretty powerful, but nano or gedit recommended for beginners
 - Some OS's might not have your preferred text editor, so good to learn others



\$ touch

- **touch** lets you create, change and modify timestamps of files
- Can create multiple files
- Can use flags for additional specifications



```
$ mkdir
```

- **mkdir** = "make directory"
- It lets you create folders



\$ **rm**, **cp**, and **mv**

- **rm** = "remove"
- It removes a file (use **rm -r** or **rmdir** to remove directories)
- **cp** = "copy"
- It copies the contents from one file to another
- **mv** = "move"
- It moves the contents from one file to another



\$ grep and find

- **grep** lets you search for patterns in a file

```
$ grep helloworld complicatedfile.txt
```

- **find** lets you search for files and directories

```
$ find *.conf
```



\$ ~~Security~~ Tip: Use man, tldr, or google!

When you dunno how
to do your homework



\$ learning **tips** and **tricks** |



\$ some general tips

- Use the **up** and **down** keys to run previous commands
- Use **TAB** for autocompletion
- **!!** - run the previous command
- **!\$** - gives you access to previous command arguments
- Use **CTRL X**, **CTRL C** or **q** for exiting



\$ clear and history

- **clear** lets you clear up the terminal
- You can also use **CTRL L**
- **history** lists out the commands you've previously used
- Clear history with **-c**
- You can use **CTRL R** for an interactive history search



\$ redirection and pipes

- Redirect a command to a file or vice versa

```
$ echo some text > file
```

```
$ cat < file
```

- Pipes effectively chain commands together

```
$ cat file | less
```



\$ **user** and **group** management |



\$ what info does a user have?

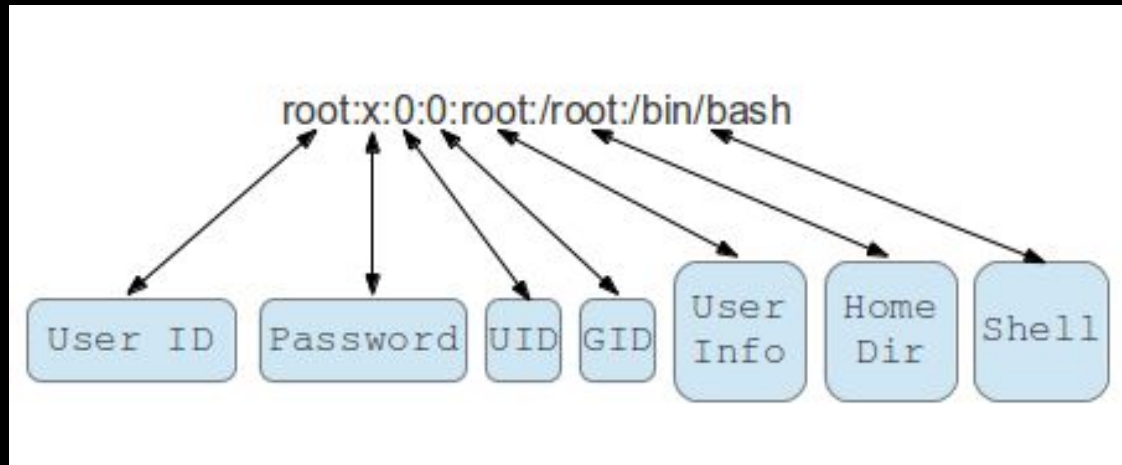
- Username
- UID (user ID)
- Default group
- Comments
- Shell
- Home directory location

And where exactly is all this stuff stored?



\$ /etc/passwd

- User info is stored in passwd file wherein the format is...



- What's up with that x though?



\$ /etc/shadow

- Encrypted passwords formally stored in **/etc/passwd**
- Now stored in **/etc/shadow** which is only readable by root
- Increase security as to reduce brute-force attacks



\$ useradd and adduser

- **useradd** takes the form...

```
$ useradd -c "<comment>" -m (create  
homdir) -s <shell> -g <primary group> -G  
<other groups> <username>
```

- Need to create password with **passwd <username>**

- **adduser** is interactive

- Handles creating the home directory, shell, password, etc



\$ userdel and deluser

- **userdel** and **deluser**
delete the user...

```
$ userdel <username>
```

```
$ deluser <username>
```

- **-r** flag can be
used to remove
the user's home
directory



\$ what info does a group have?

- Group name
- Password (usually unused)
- GID (Group ID)
- List of accounts which belong to the group
- All groups found in **/etc/group**



\$ groupadd, groupdel and usermod

- **groupadd** and **groupdel** add/delete groups

```
$ groupadd <group name>
```

```
$ groupdel <group name>
```

- **usermod** lets you add users to a group

```
$ usermod -g <primary> -G <alt1>, <altN>
```

```
$ usermod -aG <newgrp1>, <newgrp2>,  
<newgrpN>
```



\$ id, groups, and passwd

- **id** and **groups** check the id and group the user belongs to

```
$ id <user>
```

```
$ groups <user>
```

- **passwd** changes the user's password

```
$ passwd <user>
```

- Note: root always has UID and GID of 0



\$ **Security Tip:** Implement password policy!



\$ sudo and su

- `sudo <command>` - run command as root
- `su <username>` - changes your user id to become superuser
- Access to sudo is defined in the **`/etc/sudoers`** file



\$ sudo and su

- Fun fact!



A screenshot of a Google search interface. The search bar at the top contains the text "who created sudo". Below the search bar, there are navigation links for "All", "Images", "News", "Videos", "Shopping", "More", "Settings", and "Tools". The search results show "About 26,600,000 results (0.57 seconds)". The main result is a featured snippet for "Bob Cogheshall". The text of the snippet states: "Sudo was first conceived and implemented by Bob Cogheshall and Cliff Spencer around 1980 at the Department of Computer Science at SUNY/Buffalo. It ran on a VAX-11/750 running 4.1BSD. An updated version, credited to Phil Betchel, Cliff Spencer, Gretchen Phillips, John LoVerso and Don Gworek, was posted to the net." Below this text, there is a link to "www.sudo.ws > history" and a title "A Brief History of Sudo". At the bottom of the search results, there are links for "About Featured Snippets" and "Feedback".

who created sudo

About 26,600,000 results (0.57 seconds)

Bob Cogheshall

Sudo was first conceived and implemented by Bob Cogheshall and Cliff Spencer around 1980 at the Department of Computer Science at SUNY/Buffalo. It ran on a VAX-11/750 running 4.1BSD. An updated version, credited to Phil Betchel, Cliff Spencer, Gretchen Phillips, John LoVerso and Don Gworek, was posted to the net.

[www.sudo.ws > history](http://www.sudo.ws/history)

A Brief History of Sudo

About Featured Snippets Feedback



\$ file and owner **permissions** |



\$ file permissions

```
# ls -l file
-rw-r--r-- 1 root root 0 Nov 19 23:49 file
```

Diagram illustrating the permissions for the file `file`:

- File type** (indicated by the first character `-`): Regular file.
- Owner (rw-)** (indicated by the next three characters `rw-`): Readable and Writeable.
- Group (r- -)** (indicated by the next three characters `r- -`): Readable.
- Other (r - -)** (indicated by the last three characters `r - -`): Readable.

Legend:

- `r` = Readable
- `w` = Writeable
- `x` = Executable
- `-` = Denied



\$ file permissions

owner			group			others		
r	w	x	r	w	x	r	w	x
(4)	(2)	(1)	(4)	(2)	(1)	(4)	(2)	(1)



\$ chmod

- chmod lets you change file permissions
- 4 = **R**ead
- 2 = **W**rite
- 1 = **E**Xecute

\$ chmod <permission> <filename>



\$ owner permissions

```
shum@sol1:~$ ls -l
total 20
drwx----- 2 shum  staff  4096 Jan 16 22:04 Mail
drwx----- 3 shum  staff  4096 Jan 16 14:15 csc128
drwxr-xr-x  2 shum  staff  4096 Jan 13 16:42 public
drwxr-xr-x  2 shum  staff  4096 Jan 16 14:07 public_html
-rw-r--r--  1 shum  staff   628 Jan 15 20:04 verse
```

Diagram illustrating the components of the `ls -l` output:

- file type**: Indicated by the first character of the permissions (e.g., `d` for directory, `-` for regular file).
- permissions**: Indicated by the next nine characters (e.g., `rw` for owner, `rx` for group, `rx` for others).
- number of hard links**: Indicated by the number following the permissions (e.g., `2`).
- user (owner) name**: Indicated by the user name (e.g., `shum`).
- group name**: Indicated by the group name (e.g., `staff`).
- size**: Indicated by the file size in bytes (e.g., `4096`).
- date/time last modified**: Indicated by the date and time (e.g., `Jan 16 22:04`).
- filename**: Indicated by the file name (e.g., `Mail`).

Legend for permissions:

- rwx**: readable, writeable, executable
- rw**: readable, writeable
- r**: readable



\$ chown and chgrp

- **chown** lets you change the user who owns the file

```
$ chown <user> <path_to_file>
```

- **chgrp** lets you change the group who owns the file



```
$ chgrp <group> <path_to_file>
```



\$ **networking** commands |



\$ ip addr and ifconfig

- **ip addr** and **ifconfig** let you display the network specifications

```
$ ip addr
```

```
$ ip a
```

```
$ ip r
```

```
$ ifconfig
```



```
$ ping
```

- **ping** lets you send an ICMP echo request packet to network hosts to check connectivity

```
$ ping <IP address>
```



\$ nslookup and dig

- **nslookup** and **dig** let you query DNS nameservers

\$ nslookup <domain name>

\$ dig <domain name>



\$ netstat and netcat

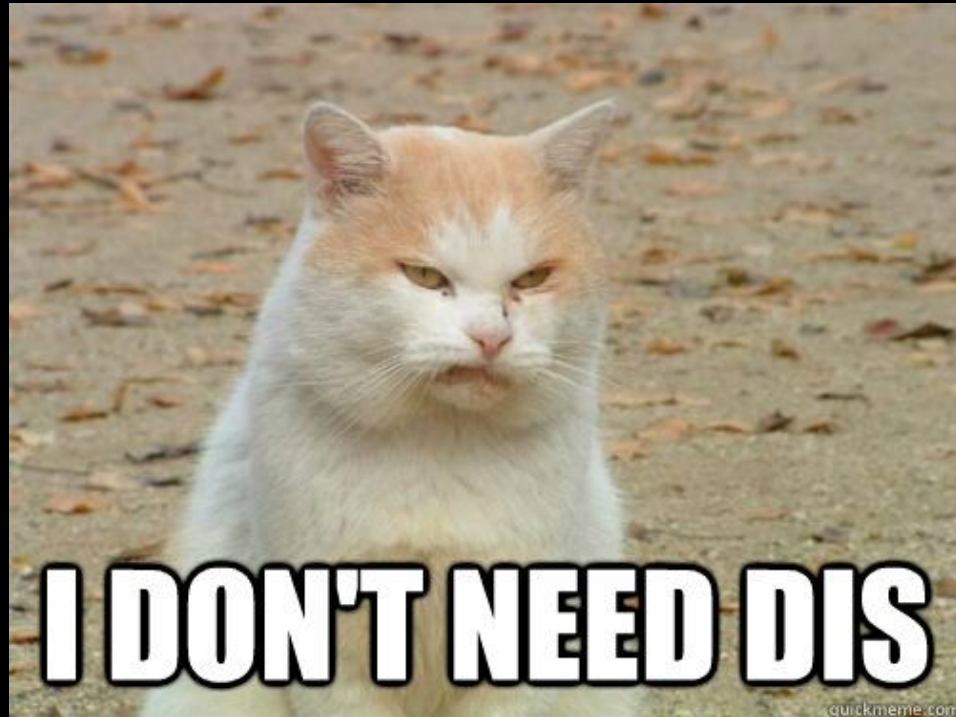
- **netstat** lets you see which applications are listening to current traffic
- **netcat** lets you connect connectivity to a TCP or UDP port
 - **-v** : verbose
 - **-z** : Scan without sending data

\$ netstat

\$ nc -vz <domain> <port>



\$ **Security Tip:** Only open ports that you need!



\$ nmap and traceroute

- **nmap** = "network mapper"
- It lets you scan a host to see what ports the host is listening to

```
$ nmap <IP address>
```

- **traceroute** lets you trace the path of the network
 - Useful for determining latency and network issues

```
$ traceroute <IP address>
```



```
$ ssh
```

- **ssh** = "secure shell"
- It lets you remote connect securely to another machine (replaced by Telnet)

```
$ ssh username@hostname
```



\$ **services** and **processes** commands |



\$ apt

- **apt** lets you install package managers

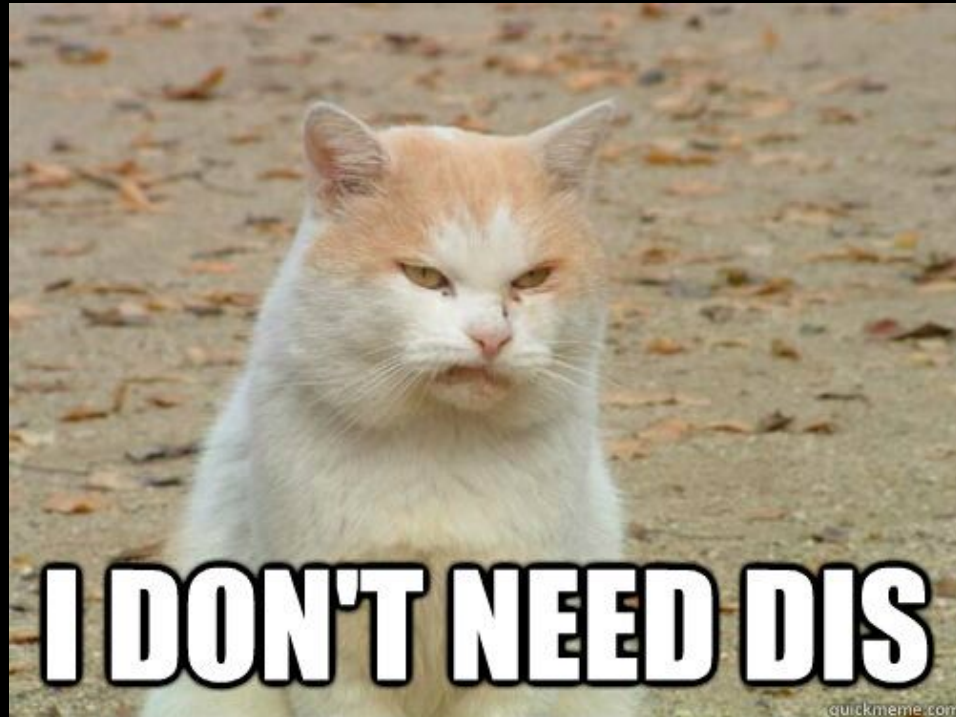
```
$ apt-get update
```

```
$ apt-get install <package>
```

```
$ apt upgrade <package>
```



\$ **Security Tip:** Only install what you need!



```
$ ps
```

- **ps** = "process status"
- It lets you see info about current processes
 - **a** : Shows processes for all users
 - **u** : Displays the process' user/owner
 - **x** : Shows processes not attached to a terminal

```
$ ps aux
```

```
$ ps aux | grep <search> | less
```



\$ top and htop

- **top** and **htop** let you see info about current processes interactively
 - htop needs to be installed first

```
$ top
```

```
$ htop
```



\$ service and systemctl

- Two main ways to control a service...
- System V uses service (older)

```
$ service <name> <start | stop | restart  
| reload | status>
```

- Systemd uses systemctl (newer)

```
$ systemctl <name> <start | stop |  
restart | reload | status>
```



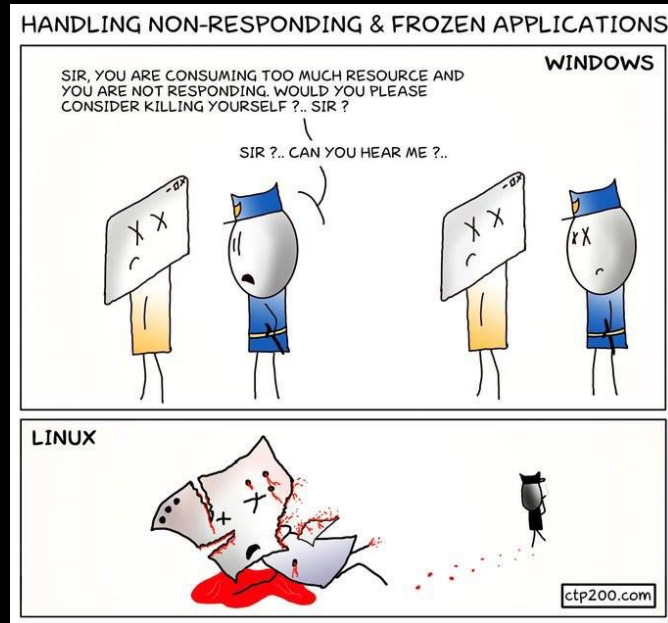
\$ kill

- **kill** lets you stop running a process

```
$ kill <process id>
```

```
$ kill %<job id>
```

```
$ kill -9 <process id>
```



\$ any **questions?** |

