

Linux

UBNetDef Spring 2022

Vasu Baldwa

Ethan Viapiano

What is Linux?

- You may have heard of Linux being talked about by upper level CS grads in the context of “kernel space memory management”.
- It's not that complicated.

What is a Linux?

- Specifically: Linux is a kernel, the bit of software that communicates between the hardware and the operating system.
- It's found everywhere.
 - Operating systems
 - Embedded devices
 - Supercomputers
 - My car runs linux.
- More generally: Linux is a group of operating systems (called "distributions") that all use the linux kernel.

Distributions

- There are countless different distributions (shortened to "distros")
- 2 major families:
 - Debian based
 - Includes Debian, Ubuntu, Kali, Mint, Pop
 - Red Hat based
 - Includes Red Hat, Fedora, CentOS, Rocky
- Other distributions include:
 - RedstarOS (리눅스가 최고다)
 - Arch
 - OpenSuse
 - Gentoo
 - Feel free to ask SecDev what they use!

The Terminal

- Another way to interact with your system.
- Most GUI activity can be done here faster.
- When have we used a terminal in class?

The Terminal

- Running without a GUI (headless) mean systems can be more lightweight
- There are several common command line interpreters, or **shells**
 - bash, zsh, sh, csh, fish, (and many more)
- Typically, you will see a prompt in your shell that gives you some information about your current session, often including your current directory
 - You can customize your prompt via a configuration file (such as `~/.bashrc`)

```
vasu@DESKTOP-04D01ET:/mnt/d/Documents$ Hello SysSec!
```

User

Hostname

Current Directory

Type Here

“Command Line” “CLI”

“Shell” “Bash”

“Terminal”

“Hacking Window”

Terminal

- sysadmin: The username of the current user logged in
- VasuKali: The hostname of the machine

sysadmin@VasuKali: ~

File Actions Edit View Help

sysadmin@VasuKali:~\$ ls -al Documents/

```
total 12
drwxr-xr-x  3 sysadmin sysadmin 4096 Apr 30 21:45 .
drwxr-xr-x 17 sysadmin sysadmin 4096 Sep  1 08:50 ..
drwxr-xr-x  3 sysadmin sysadmin 4096 Apr 30 21:45 Ansible
```

Terminal

■ ~ :Home directory shortcut

sysadmin@VasuKali: ~

File Actions Edit View Help

sysadmin@VasuKali:~\$ ls -al Documents/

total 12

drwxr-xr-x 3 sysadmin sysadmin 4096 Apr 30 21:45 .

drwxr-xr-x 17 sysadmin sysadmin 4096 Sep 1 08:50 ..

drwxr-xr-x 3 sysadmin sysadmin 4096 Apr 30 21:45 Ansible

Terminal

- \$:The prompt symbol.
- Denotes the end of the command prompt
 - User's keyboard input will appear next

sysadmin@VasuKali: ~

File Actions Edit View Help

sysadmin@VasuKali:~\$ ls -al Documents/

total 12

drwxr-xr-x 3 sysadmin sysadmin 4096 Apr 30 21:45 .

drwxr-xr-x 17 sysadmin sysadmin 4096 Sep 1 08:50 ..

drwxr-xr-x 3 sysadmin sysadmin 4096 Apr 30 21:45 Ansible

Commands

- `ls` :A command
 - An instruction given by a user telling a computer to do something

sysadmin@VasuKali: ~

File Actions Edit View Help

sysadmin@VasuKali:~\$ **ls** -al Documents/

total 12

drwxr-xr-x	3	sysadmin	sysadmin	4096	Apr	30	21:45	.
drwxr-xr-x	17	sysadmin	sysadmin	4096	Sep	1	08:50	..
drwxr-xr-x	3	sysadmin	sysadmin	4096	Apr	30	21:45	Ansible

Commands

- -a_l: A flag
 - A way to set options and pass in arguments to the commands you run.
 - Commands change their behavior based on what flags are set.

```
sysadmin@VasuKali:~$ ls -al Documents/
total 12
drwxr-xr-x  3 sysadmin sysadmin 4096 Apr 30 21:45 .
drwxr-xr-x 17 sysadmin sysadmin 4096 Sep  1 08:50 ..
drwxr-xr-x  3 sysadmin sysadmin 4096 Apr 30 21:45 Ansible
```

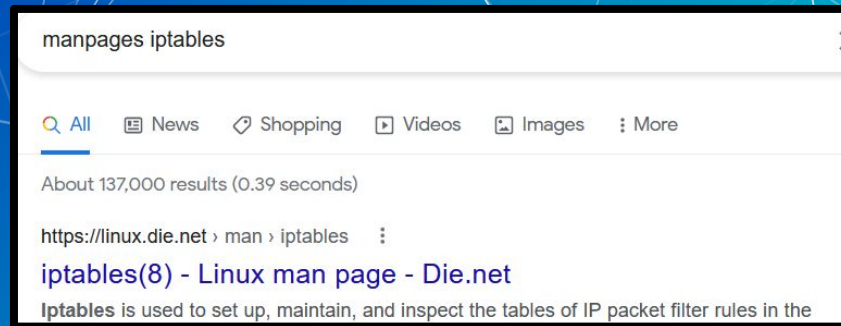
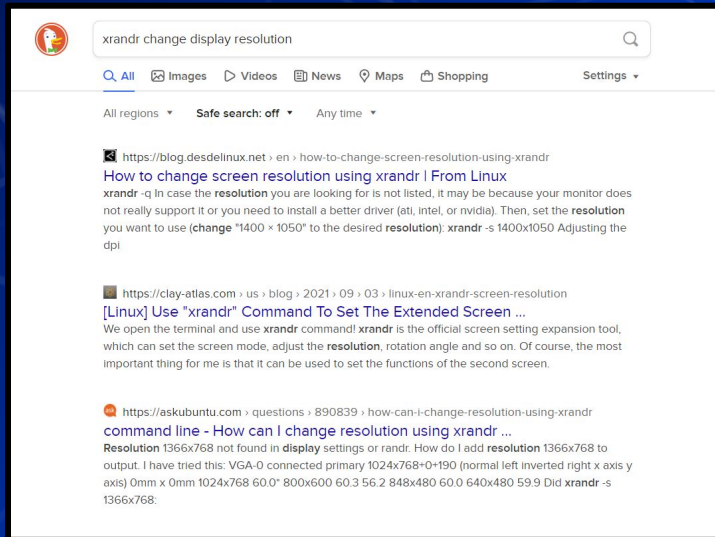
Commands

- Documents/:An argument
 - File name referenced

```
sysadmin@VasuKali:~$ ls -al Documents/  
total 12  
drwxr-xr-x  3 sysadmin sysadmin 4096 Apr 30 21:45 .  
drwxr-xr-x 17 sysadmin sysadmin 4096 Sep  1 08:50 ..  
drwxr-xr-x  3 sysadmin sysadmin 4096 Apr 30 21:45 Ansible
```


Commands? Memorization?

- **Look it up.** It's what I do, it's what Ken Smith does, it's what everyone does.
 - Best way to learn/troubleshoot anything linux related
- This lecture covers ~20/30 of the most important/useful commands



```
sysadmin@VasuKali:~$ man man
```

Man pages

- If you're stuck and the suffix `--help` isn't helping, use the prefix `man`
- Fully detailed description of what each command suffix does.
- `man - Manual`

```
MAN(1)                                Manual pager utils                                MAN(1)

NAME
    man - an interface to the system reference manuals

SYNOPSIS
    man [man options] [[section] page ...] ...
    man -k [apropos options] regexp ...
    man -K [man options] [section] term ...
    man -f [whatis options] page ...
    man -l [man options] file ...
    man -w|-W [man options] page ...

DESCRIPTION
    man is the system's manual pager. Each page argument given to man
    is normally the name of a program, utility or function. The man-
    ual page associated with each of these arguments is then found and
    displayed. A section, if provided, will direct man to look only
    in that section of the manual. The default action is to search in
    all of the available sections following a pre-defined order (see
    DEFAULTS), and to show only the first page found, even if page ex-
    ists in several sections.

    The table below shows the section numbers of the manual followed
    by the types of pages they contain.

    Manual page man(1) line 1 (press h for help or q to quit)
```




showing [all](#), navigate: [← explain sort\(1\)](#) [→ explain shell syntax](#)

▾ [cut\(1\)](#) -d ' ' -f 1 /var/log/apache2/access_logs | ▾ [uniq\(1\)](#) -c | ▾ [sort\(1\)](#) -n

remove sections from each line of files

-d, --delimiter=DELIM
use DELIM instead of TAB for field delimiter

-f, --fields=LIST
select only these fields; also print any line that contains no delimiter character, unless the **-s** option is specified

With no FILE, or when FILE is -, read standard input.

Pipelines

A [pipeline](#) is a sequence of one or more commands separated by one of the control operators `|` or `|&`. The format for a pipeline is:

```
[time [-p]] [ ! ] command [ [|&] command2 ... ]
```

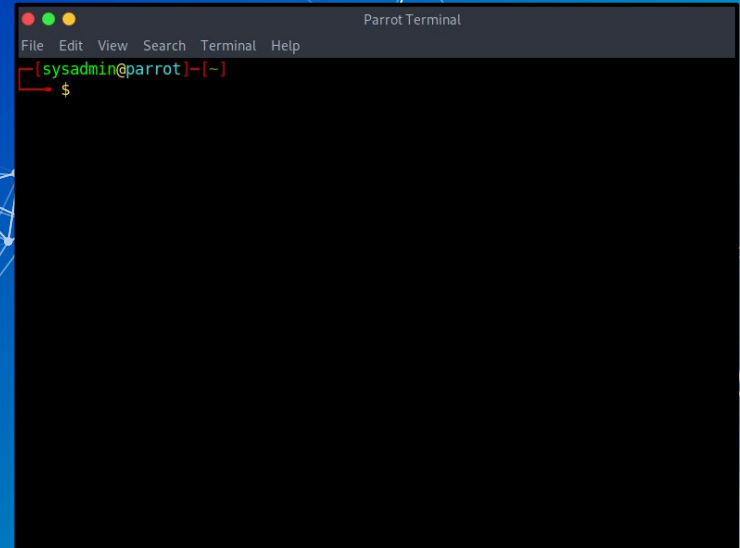
The standard output of [command](#) is connected via a pipe to the standard input of [command2](#). This connection is performed before any redirections specified by the command (see **REDIRECTION** below). If `|&` is used, the standard error of [command](#) is connected to [command2](#)'s standard input through the pipe; it is shorthand for `2>&1 |`. This implicit redirection of the standard error is performed after any redirections specified by the command.

Tab Tab Tab Tab Tab Tab Tab Tab Tab Tab...

- Many shells use tab to autocomplete or suggest autocompletion
- This is so useful it gets its own slide

What am I?

- Now that we've opened up the terminal, we can start to get our bearings on the system
- `whoami` : Current user
- `pwd` : Where you are
- `hostname` : Name of system you are on
- `ip a` : What is your network information
- `ps -aux` : What is running
- `clear` : clears the screen

A screenshot of a Parrot Terminal window. The title bar at the top says "Parrot Terminal". Below the title bar is a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal content shows a green prompt character followed by the text "(sysadmin@parrot)~" and a red cursor. Below this, a green dollar sign "\$" is visible, indicating the shell prompt.

```
(sysadmin@parrot)~  
$
```

What am I?

- Now that we've opened up the terminal, we can start to get our bearings on the system
- `whoami`

```
sysadmin@Vasukali:~$ whoami  
sysadmin  
sysadmin@Vasukali:~$
```

What am I?

- Now that we've opened up the terminal, we can start to get our bearings on the system
- `pwd` : Print Working Directory

```
sysadmin@VasuKali:~$ pwd
/home/sysadmin
sysadmin@VasuKali:~$
```

What am I?

- Now that we've opened up the terminal, we can start to get our bearings on the system
- `hostname` : Name of system you are on

```
sysadmin@VasuKali:~$ hostname  
VasuKali  
sysadmin@VasuKali:~$
```


What am I?

- Now that we've opened up the terminal, we can start to get our bearings on the system
- `ip a`: What is your network information

```
sysadmin@VasuKali:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group d
efault qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:50:56:86:03:a8 brd ff:ff:ff:ff:ff:ff
    inet 192.168.13.174/20 brd 192.168.15.255 scope global dynamic noprefix
route eth0
        valid_lft 6330sec preferred_lft 6330sec
    inet6 fe80::250:56ff:fe86:3a8/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue stat
e DOWN group default
    link/ether 02:42:4a:74:b3:92 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
sysadmin@VasuKali:~$
```

What am I?

- Now that we've opened up the terminal, we can start to get our bearings on the system
- `ps -aux` : process status
 - Shows **(a)**ll the processes
 - With **(u)**sernames
 - Including those not started from the terminal **(x)**

```
demo@mx1:~$ ps -aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.1	0.0	3292	2056	?	Ss	22:23	0:00	init [5]
root	2	0.0	0.0	0	0	?	S	22:23	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	I<	22:23	0:00	[rcu_gp]
root	4	0.0	0.0	0	0	?	I<	22:23	0:00	[rcu_par_gp]
root	6	0.0	0.0	0	0	?	I<	22:23	0:00	[kworker/0:0H-events_highpri]
root	8	0.0	0.0	0	0	?	I<	22:23	0:00	[mm_percpu_wq]
root	9	0.0	0.0	0	0	?	S	22:23	0:00	[rcu_tasks_rude_]
root	10	0.0	0.0	0	0	?	S	22:23	0:00	[rcu_tasks_trace]
root	11	0.0	0.0	0	0	?	S	22:23	0:00	[ksoftirqd/0]
root	12	0.0	0.0	0	0	?	I	22:23	0:00	[rcu_sched]
root	13	0.0	0.0	0	0	?	S	22:23	0:00	[migration/0]
root	14	0.0	0.0	0	0	?	I	22:23	0:00	[kworker/0:1-events]
root	15	0.0	0.0	0	0	?	S	22:23	0:00	[cpuhp/0]
root	16	0.0	0.0	0	0	?	S	22:23	0:00	[cpuhp/1]
root	17	0.0	0.0	0	0	?	S	22:23	0:00	[migration/1]
root	18	0.0	0.0	0	0	?	S	22:23	0:00	[ksoftirqd/1]
root	20	0.0	0.0	0	0	?	I<	22:23	0:00	[kworker/1:0H-kblockd]
root	21	0.0	0.0	0	0	?	S	22:23	0:00	[cpuhp/2]
root	22	0.0	0.0	0	0	?	S	22:23	0:00	[migration/2]
root	23	0.0	0.0	0	0	?	S	22:23	0:00	[ksoftirqd/2]
root	25	0.0	0.0	0	0	?	I<	22:23	0:00	[kworker/2:0H-kblockd]

What am I?

- Now that we've opened up the terminal, we can start to get our bearings on the system
- `clear` : clears the screen
 - Does not clear the history

```
Tasks: 178 total, 1 running, 177 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.1 us, 0.1 sy, 0.0 ni, 99.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.
MiB Mem : 7965.8 total, 6194.7 free, 622.7 used, 1148.4 buff/cache
MiB Swap: 975.0 total, 975.0 free, 0.0 used, 7047.6 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+
451	root	20	0	237772	8116	6780	S	0.3	0.1	10:19.86
4378	sysadmin	20	0	1293540	83904	64308	S	0.3	1.0	0:00.63
32049	sysadmin	20	0	9064	3572	3136	R	0.3	0.0	0:00.02
1	root	20	0	168188	11332	8412	S	0.0	0.1	0:12.62
2	root	20	0	0	0	0	S	0.0	0.0	0:00.40
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00
9	root	20	0	0	0	0	S	0.0	0.0	0:00.07
10	root	20	0	0	0	0	I	0.0	0.0	0:30.11
11	root	rt	0	0	0	0	S	0.0	0.0	0:03.77
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00
15	root	rt	0	0	0	0	S	0.0	0.0	0:03.54
16	root	20	0	0	0	0	S	0.0	0.0	0:01.22
18	root	0	-20	0	0	0	I	0.0	0.0	0:00.00
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00
20	root	rt	0	0	0	0	S	0.0	0.0	0:03.58
21	root	20	0	0	0	0	S	0.0	0.0	0:00.05

```
sysadmin@VasuKali:~$
```



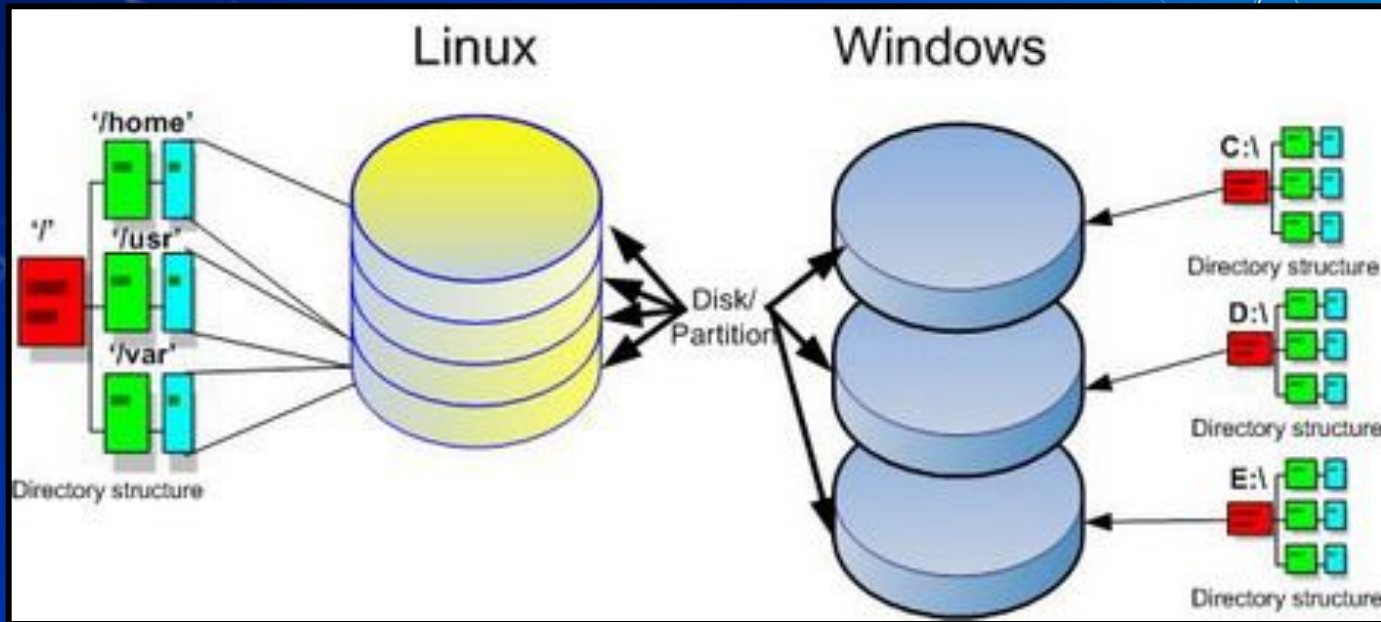
```
sysadmin@VasuKali: ~
File Actions Edit View Help
sysadmin@VasuKali:~$
```

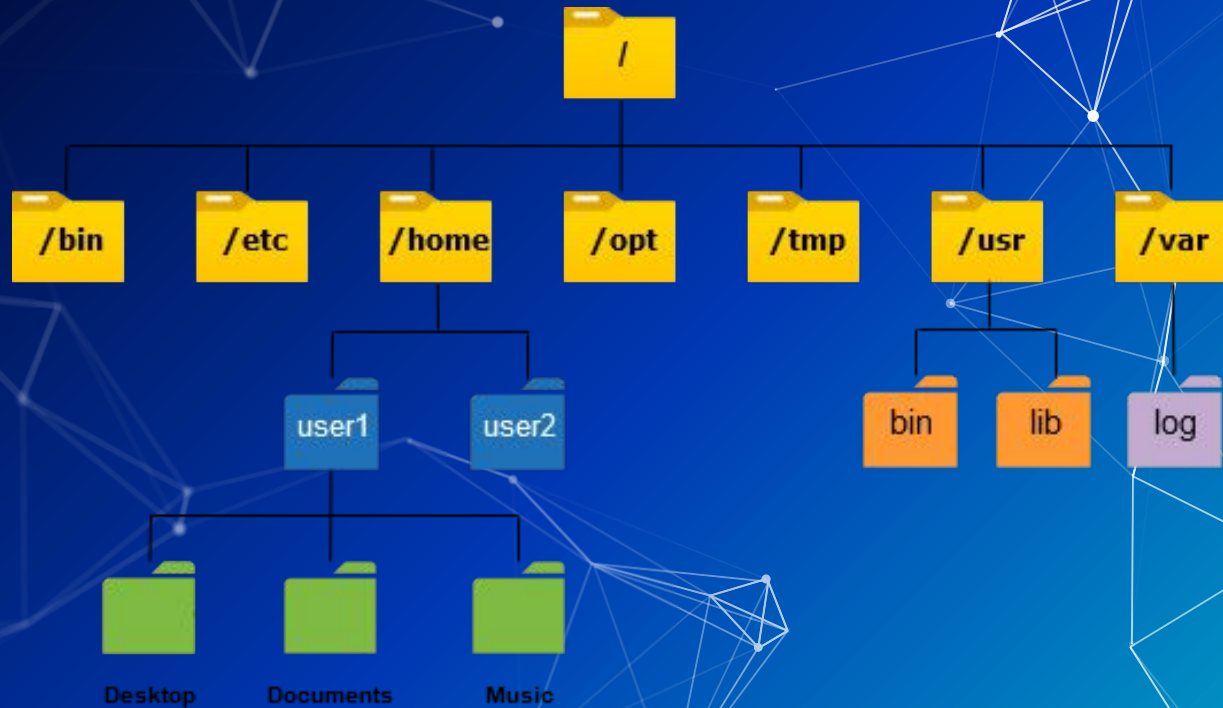
Questions (Question mark)

Demo!

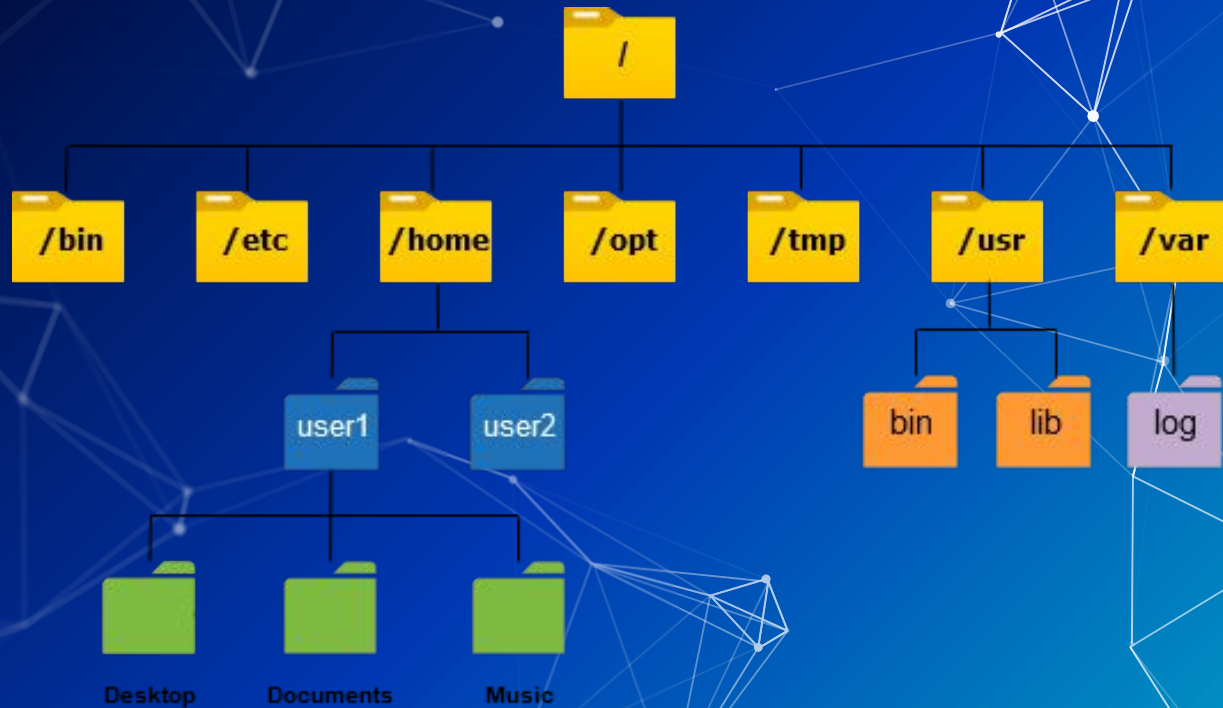
Understanding the filesystem

- Everything is built of the root or / directory
- Everything is a file

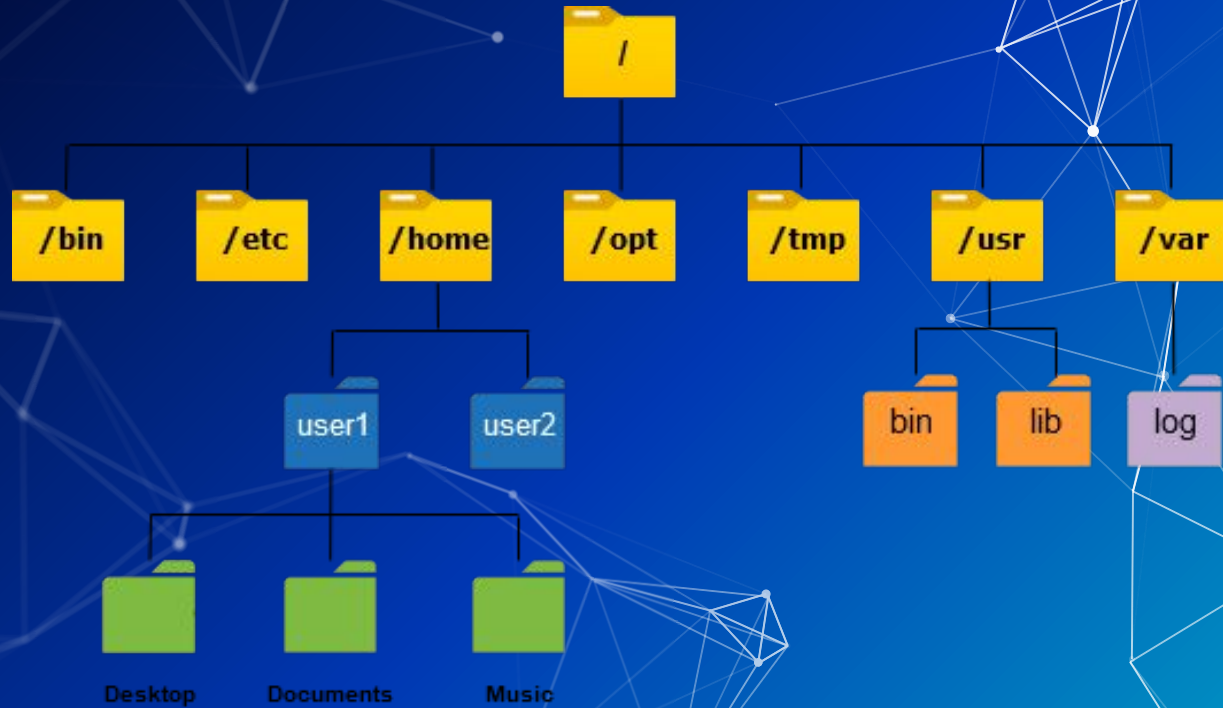




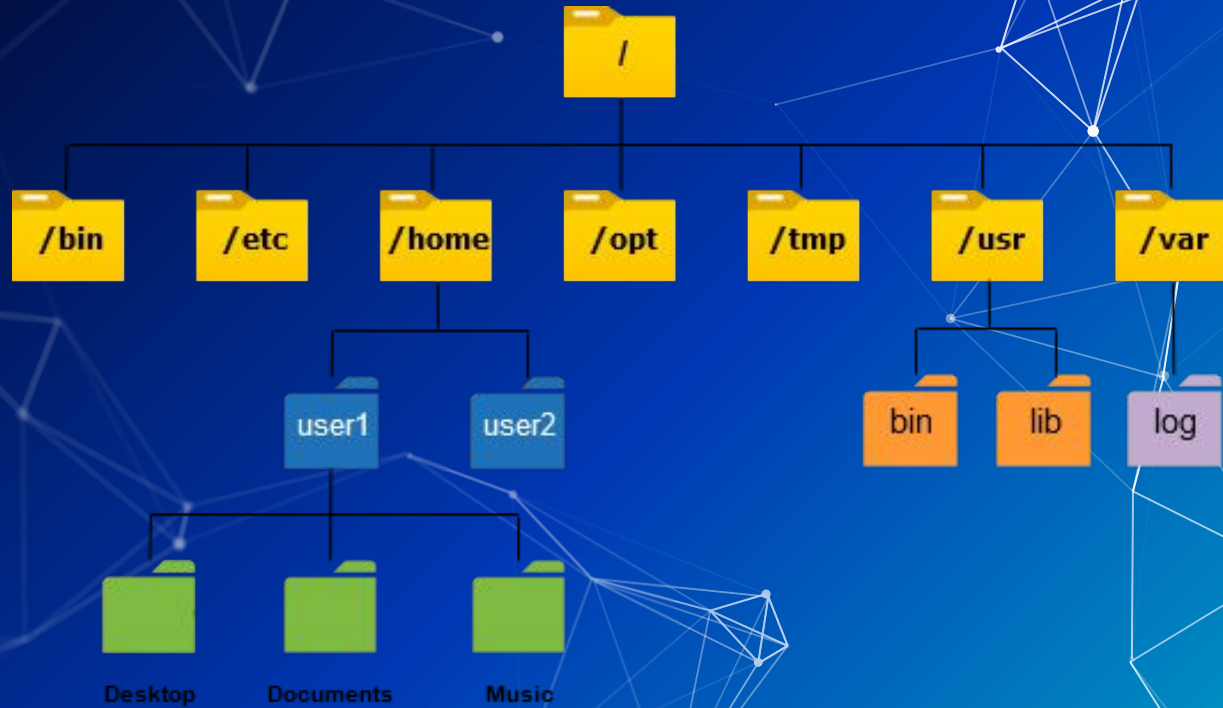
- / (root) root directory of the entire system hierarchy.
 - Everything starts at root.
 - Nothing is higher than root.



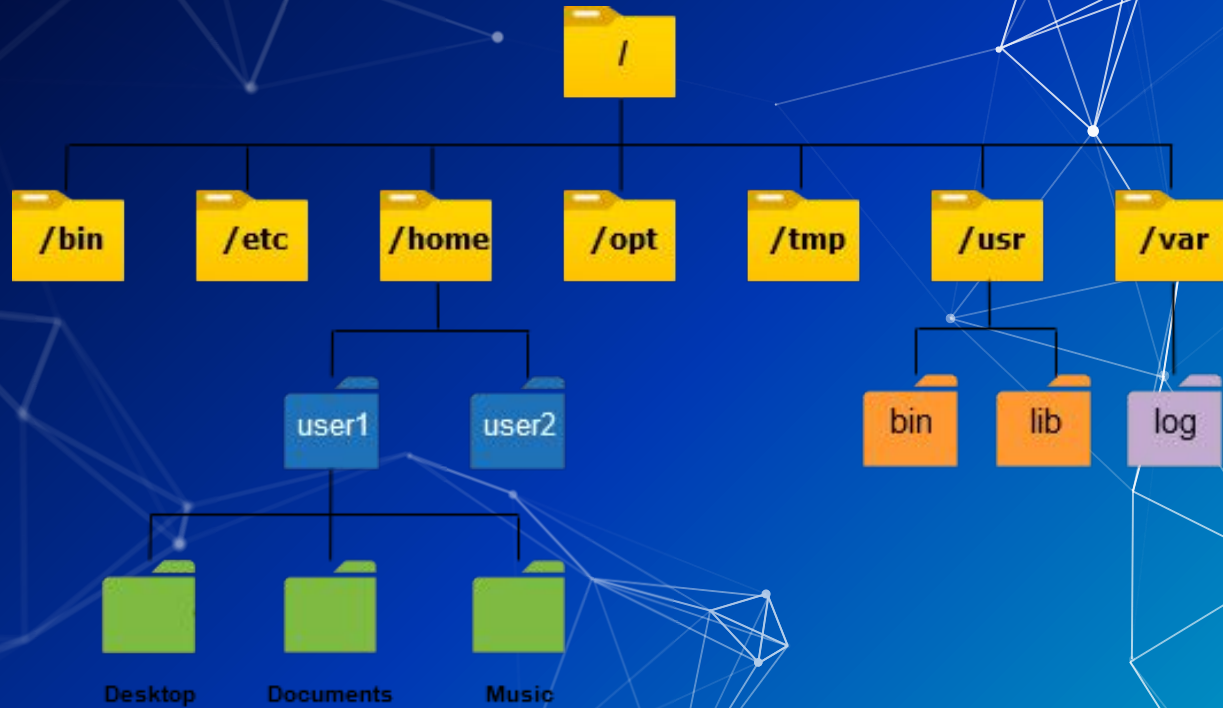
- /bin/ essential command binaries
 - whoami, pwd, cp are all stored here



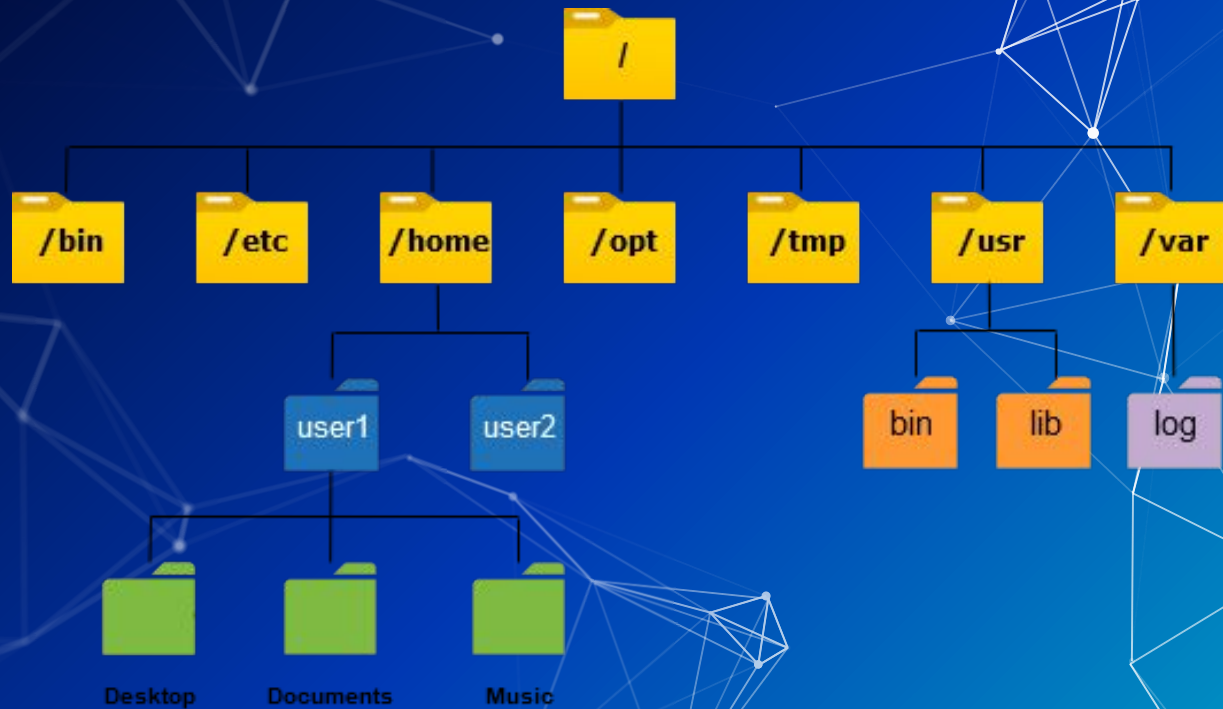
- `/etc/` specific system-wide configuration files
 - We edited the network configuration file in here for HW02



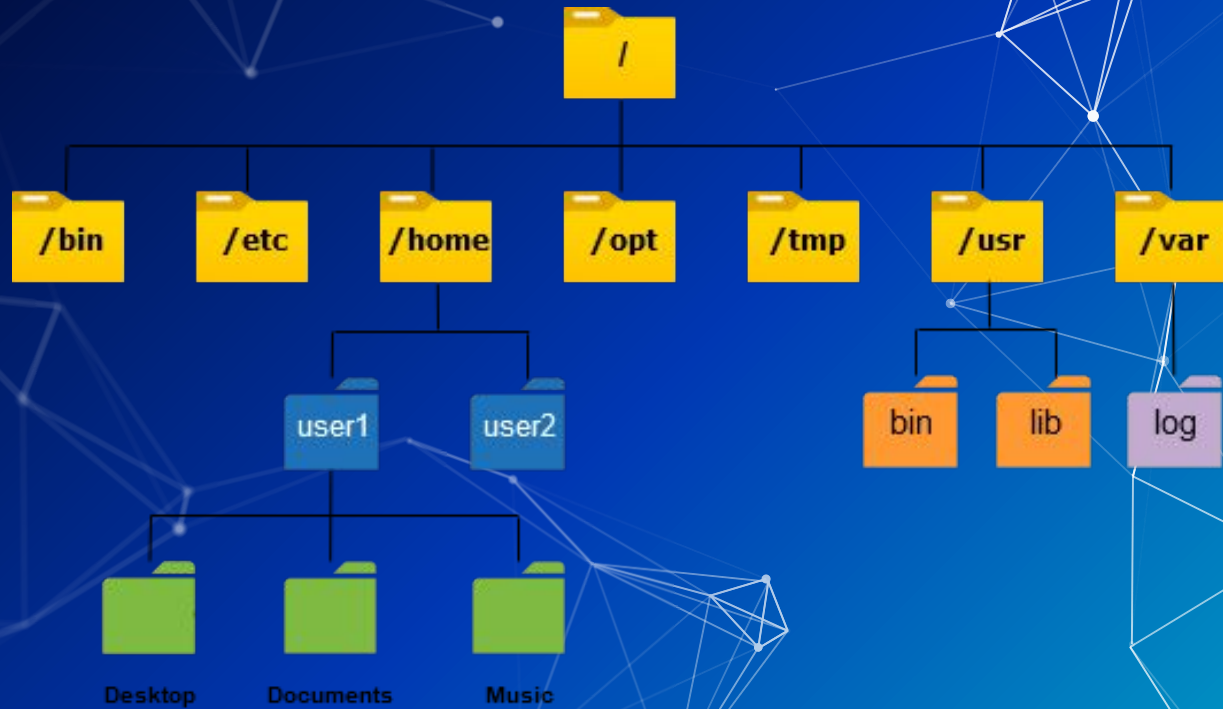
- `/home/` Users' home directories, containing saved files, personal settings, etc.



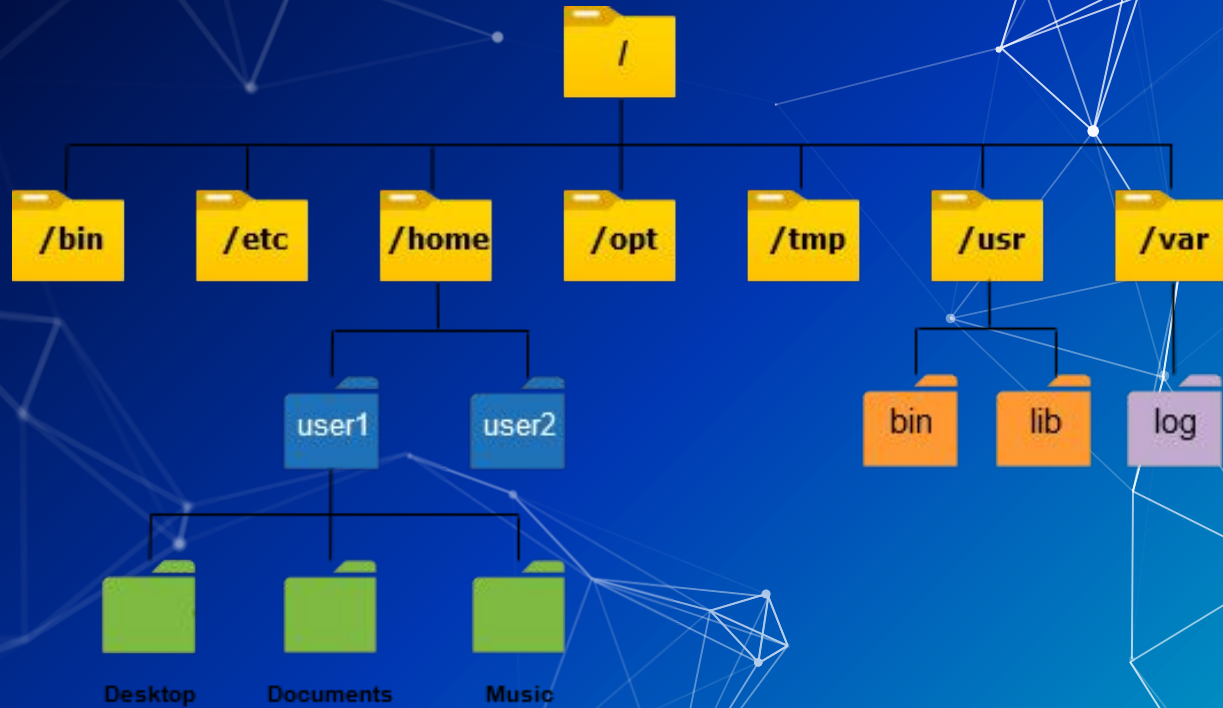
- /opt/ Additional software and addons



- /tmp/ Temporary files
 - Typically not saved after reboots



- `/usr/` user level binaries and applications



- `/var/` Variable files - content of the file is expected to continually change during normal operation of the system
 - System logs are stored here

Linux FHS

- There are more key paths on the filesystem that we haven't covered
- These are specified in the Filesystem Hierarchy Standard (FHS)
- You can access that information from your terminal with `man hier`
- <https://refspecs.linuxfoundation.org/fhs.shtml>

```
HIER(7)                                Linux Programmer's Manual                                HIER(7)
NAME
    hier - description of the filesystem hierarchy
DESCRIPTION
    A typical Linux system has, among others, the following directories:

    /      This is the root directory. This is where the whole tree
           starts.

    /bin   This directory contains executable programs which are needed in
           single user mode and to bring the system up or repair it.

    /boot  Contains static files for the boot loader. This directory holds
           only the files which are needed during the boot process. The
           map installer and configuration files should go to /sbin and
           /etc. The operating system kernel (initrd for example) must be
           located in either / or /boot.

    /dev   Special or device files, which refer to physical devices. See
           mknod(1).
```

Questions (Question mark)

**How do we navigate the file
system?**

Navigating Directories

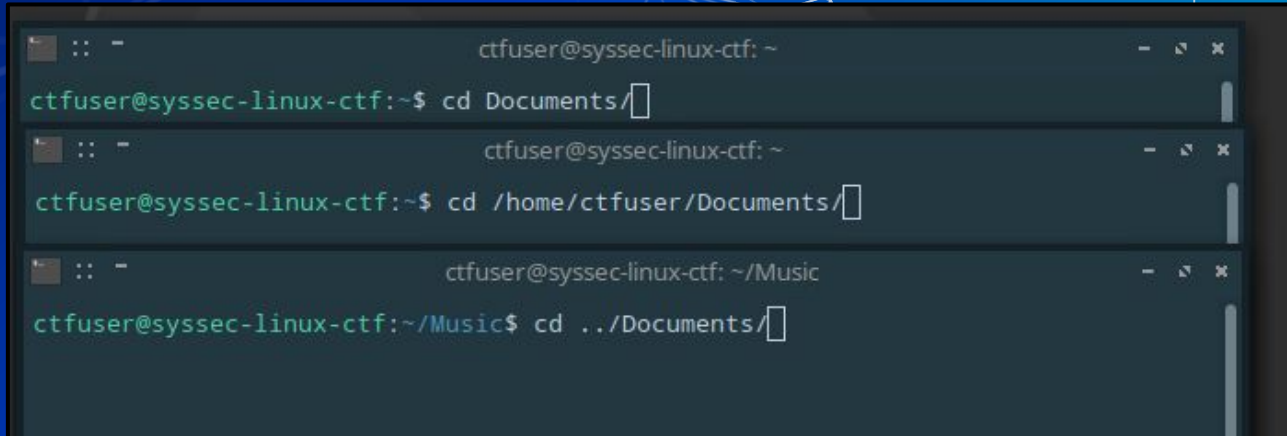
- `cd` - change directory: changes working directory
 - Usage: `cd <relative/absolute path>`
- `ls` - list files in a directory: shows files in a given directory
 - Files or directories that start with "." are hidden.
 - `ls -a`: shows hidden files and directories

```
vasu@DESKTOP-04D01ET:/mnt/d/Documents/College/UBNetDef/LinuxExample$ ls
ThisFileIsVisible.txt  YouCanAlsoSeeThisOne.txt
vasu@DESKTOP-04D01ET:/mnt/d/Documents/College/UBNetDef/LinuxExample$
```

```
vasu@DESKTOP-04D01ET:/mnt/d/Documents/College/UBNetDef/LinuxExample$ ls -a
.  ..  .soIsThisFile.txt  .ThisFileIsHidden.txt  ThisFileIsVisible.txt  YouCanAlsoSeeThisOne.txt
vasu@DESKTOP-04D01ET:/mnt/d/Documents/College/UBNetDef/LinuxExample$
```


Relative vs Absolute Paths

- Relative Locations
 - ~ Current user's "home" directory (shortcut)
 - . The current directory
 - .. The parent to your current directory
 - - The last directory you went to
- File Paths can be defined from your current directory (relative), or from the root directory(absolute).



The image shows three terminal windows stacked vertically, illustrating path navigation. Each window has a title bar with a standard Linux window icon and a close button.

```
ctfuser@syssec-linux-ctf: ~  
ctfuser@syssec-linux-ctf:~$ cd Documents/  
  
ctfuser@syssec-linux-ctf: ~  
ctfuser@syssec-linux-ctf:~$ cd /home/ctfuser/Documents/  
  
ctfuser@syssec-linux-ctf: ~/Music  
ctfuser@syssec-linux-ctf:~/Music$ cd ../Documents/
```

Demo!

Interacting with files

- cat
 - Syntax: `cat <filename>`
 - Displays the contents of the file in the terminal.

```
[sysadmin@parrot]-[~/Documents/NetDef/LinuxExamples]
└─$ cat ExampleText.txt
This is some random text. Radhika likes pineapples

Anthony sucks.

CCDC will go to RIT.

Baby Enzo wants his tendies.

[sysadmin@parrot]-[~/Documents/NetDef/LinuxExamples]
└─$
```

Interacting with files

- less
 - Syntax: `less <filename>`
 - Provides a scrollable version of `cat`
- touch
 - Syntax: `touch <filename>`
 - Creates an empty file with the filename provided
- wc: Word Count
 - Syntax: `wc <filename>`
 - Counts the number of lines, words and bytes in each file
- file
 - Syntax: `file <filename>`
 - Provides metadata about each file

Interacting with files

- cp: Copy
 - Syntax: `cp </path/to/source> </path/to/destination>`
- mv: Move
 - Syntax: `mv </path/to/source> </path/to/destination>`
 - You can use this to rename files as well
- rm: remove
 - Syntax: `rm <filename>`
 - Deletes the file for good. No recovery.
- mkdir: Make Directory
 - Syntax: `mkdir <folder name>`

Text Editors

- Syntax is <text editor name> <file> for anything

Editors

- vim - Very powerful editor with an unconventional workflow, can be hard for beginners
 - There are many good tutorials
 - Often times the default text editor
- nano - Pretty standard text editor, easier to use
 - Arrow keys to move and you can type, ctrl + x to exit and save
- emacs / gedit - Use the built in GUI text editor
 - Just like good ol' notepad
 - Emacs does have a CLI interface

find

- Find is very powerful, useful, and complex for finding files
- Basic syntax:
 - `find <search directory> <options>`
 - `-name <name>` or `-iname <name>` (case insensitive)
 - supports wildcards such as `"hello*"` which might match `"hello_world.txt"`

grep

- grep is also a really powerful tool for searching **inside files**
 - `grep <pattern> <file>`
- It uses the power of regular expressions (regex) to do its magic
- Find text in large files
 - Log files...?
 - Filter unwanted text away
 - You can send output of other commands to it!

Hands on 1 (Navigating linux ctf)

You have a vm called LinuxCTF. There are hidden files on it. You need to use the commands we just learned to find them. Remember Google is your friend, if you don't know how to do something try searching "How do I _____ in linux?"

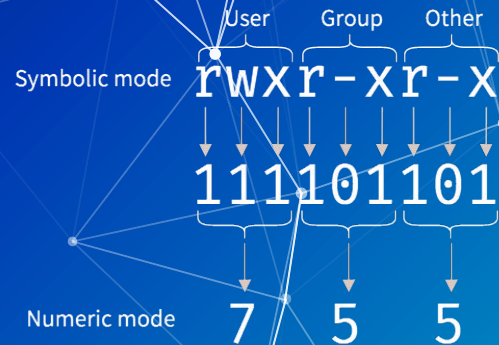
In MM: where to look, and a recap of the commands

Hands on 1 discussion

Let's talk (file) permissions

File permissions

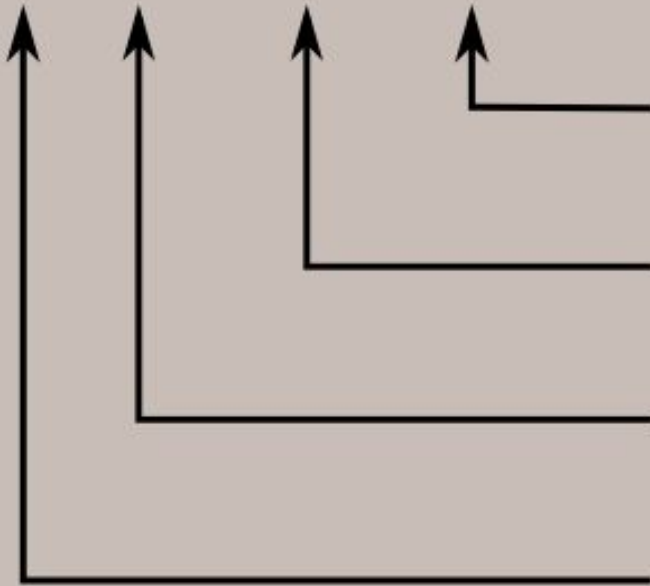
- Files owned by user and group
- File modes are read/write/execute
- Mode permissions granted to
 - owner, owning group, everyone
- Modifying
 - See permissions with `ls -l` command
 - Set modes with `chmod` command
 - Set owners with `chown` command



`-rwxrwxrwx`

```
[sysadmin@parrot]--[~/Documents/NetDef/Malware]
└─$ ls -l
total 0
drwxr-xr-x 1 sysadmin sysadmin 20 Feb 22 10:46 Bashark
drwxr-xr-x 1 sysadmin sysadmin 30 Feb 22 10:34 interject
drwxr-xr-x 1 sysadmin sysadmin 172 Feb 15 09:33 neko
[sysadmin@parrot]--[~/Documents/NetDef/Malware]
└─$
```

- rwxrwxrwx



Read, write, and execute permissions for all other users.

Read, write, and execute permissions for the group owner of the file.

Read, write, and execute permissions for the file owner.

File type:
- indicates regular file
d indicates directory

Reading a Permission Entry

- <type flag> <owner permissions> <group permissions> <world permissions>
- Default permissions = 644
 - Read and write for owner
 - Read for group and the world.
- What is 755?
- What about 245?

Octal	Binary	File Mode
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwX

chmod

- chmod = change file mode bits
- change file permissions
- chmod <permission> <filename>
 - Allow a file to be executable: `chmod +x myFile`
 - Grant all permissions to a file: `chmod 777 myFile`

```
vasu@DESKTOP-04D01ET:/mnt/d/Documents/College/UBNetDef/Lockdown/v11$ ls -l
total 500
-rwxrwxrwx 1 vasu vasu 6722 Oct 12 18:13 'Black Team Injects.docx'
-rwxrwxrwx 1 vasu vasu 42425 Oct 12 18:13 'Black Team Injects.pdf'
-rwxrwxrwx 1 vasu vasu 2606 Oct 13 02:40 gretzky-TCP4-1194-config.ovpn
-rwxrwxrwx 1 vasu vasu 11150 Oct 13 21:28 'Master Sheet.docx'
-rwxrwxrwx 1 vasu vasu 141715 Oct 13 21:28 'Master Sheet.pdf'
-rwxrwxrwx 1 vasu vasu 6047 Oct 13 02:21 "peter_gretzky-TCP4-1194-Pete's_config-config.ovpn"
-rwxrwxrwx 1 vasu vasu 6083 Oct 13 02:09 red_team_gretzky-TCP4-1194-lockdown-vpn-config.ovpn
-rwxrwxrwx 1 vasu vasu 19280 Oct 13 21:31 'RED TEAM PASSWORDS.docx'
-rwxrwxrwx 1 vasu vasu 83814 Oct 13 21:31 'RED TEAM PASSWORDS.pdf'
-rwxrwxrwx 1 vasu vasu 15455 Oct 10 15:32 'topology table.docx'
-rwxrwxrwx 1 vasu vasu 32049 Apr 25 2021 v10_REFERENCE.docx
-rwxrwxrwx 1 vasu vasu 3310 Oct 10 15:38 v11Topo.drawio
-rwxrwxrwx 1 vasu vasu 83137 Oct 10 15:38 v11Topo.drawio.png
-rwxrwxrwx 1 vasu vasu 33927 Oct 13 03:06 'v11 VPN RedTeam.pdf'
vasu@DESKTOP-04D01ET:/mnt/d/Documents/College/UBNetDef/Lockdown/v11$ |
```


Questions (Question mark)

Users and Groups

Users and Groups

- Linux systems have many users
 - One user per service
 - Stored in /etc/passwd
- Linux systems also have groups
 - Stored in /etc/group
- Every user has a User Identification number (UID)
- Groups also have unique Group Identification numbers (GIDs)
- The root user has a UID of 0
 - Root can do anything

/etc/passwd

```
testuser:x:1481:1482:This is a test user:/home/testuser:/bin/bash
```

[Username] | [Password] | [Userid] | [Groupid] | [User Information] | [User home path] | [User shell]

- Notice the x instead of the password?

/etc/shadow

- Encrypted passwords formally stored in /etc/passwd
- Now stored in /etc/shadow which is only readable by root

```
mark:$6$.n.:17736:0:99999:7:::  
[--] [----] [---] - [---] ----  
|      |      |      |      |||+-----> 9. Unused  
|      |      |      |      ||+-----> 8. Expiration date  
|      |      |      |      |+-----> 7. Inactivity period  
|      |      |      |      +-----> 6. Warning period  
|      |      |      +-----> 5. Maximum password age  
|      |      +-----> 4. Minimum password age  
|      +-----> 3. Last password change  
|  +-----> 2. Encrypted Password  
+-----> 1. Username
```

Adding users

- `useradd`: Add a user to the system
 - Syntax: `useradd -c "<comment>" -m (create homedir) -s <shell> -g <primary group> -G <other groups> <username>`
 - Need to create password with `passwd <username>`
 - This is complicated and sucky
- `adduser` is interactive!
 - It is a wrapper around `useradd`
 - Handles creating the home directory, shell, password, etc
 - Not available on all systems
 - Syntax: `adduser <username>`

userdel **and** deluser

- userdel and deluser delete the user
- Like useradd and adduser, deluser is a wrapper around userdel
- Syntax: `deluser <username>`
 - The `-r` flag will also delete the user's home directory

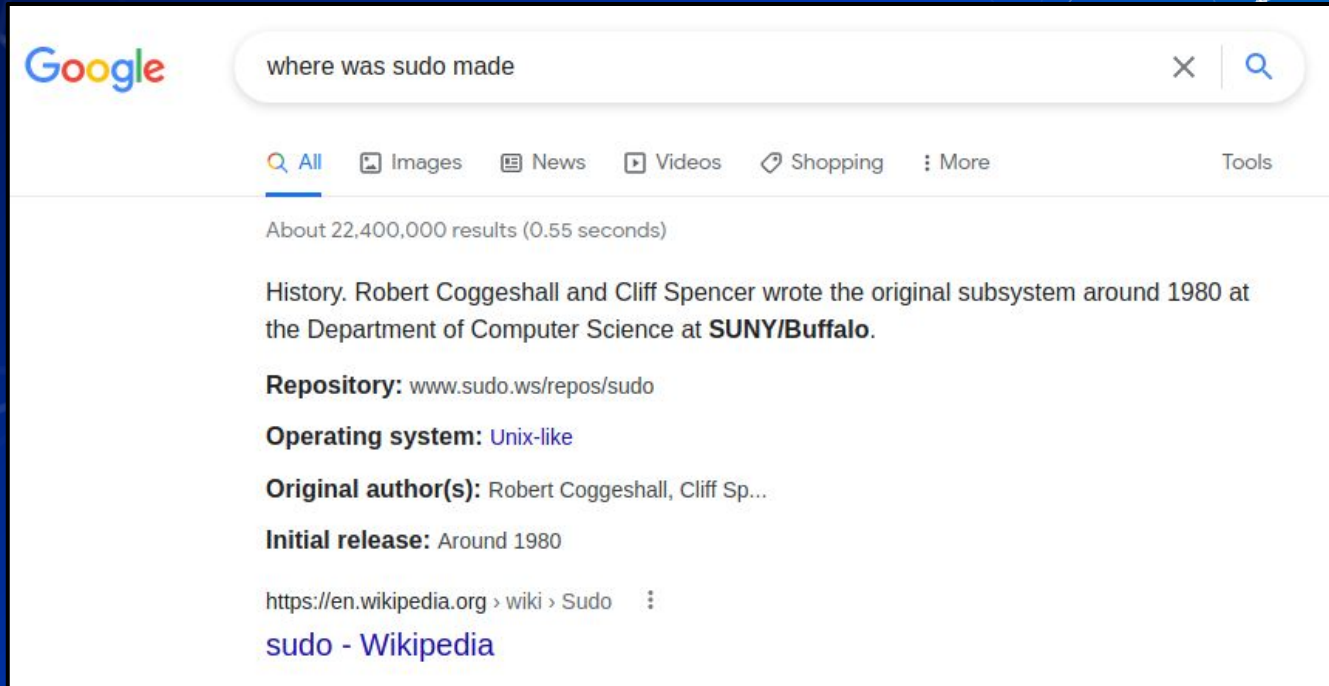
Administrative Right and Users

- The root user has full access to every part of the system
- Other users can access “root permissions” with the sudo command
- sudo: super user do
 - Syntax: `sudo <command>`
 - This will run the command with sudo permissions
 - To use sudo you must be in the sudo group
- Limit others users sudo access by editing the sudoers file
 - This is a special file, and must be edited with the `vi sudo` command

Administrative Right and Users

- You can switch users with su
- su: switch user
 - Syntax: su <username>
 - Typing su without a username will switch you into the root user

Fun fact about sudo:



A screenshot of a Google search interface. The search bar contains the text "where was sudo made". Below the search bar, there are tabs for "All", "Images", "News", "Videos", "Shopping", and "More". The "All" tab is selected. Below the tabs, it says "About 22,400,000 results (0.55 seconds)". The search results show a snippet of text: "History. Robert Coggeshall and Cliff Spencer wrote the original subsystem around 1980 at the Department of Computer Science at **SUNY/Buffalo**." Below this, there are several key-value pairs: "Repository: www.sudo.ws/repos/sudo", "Operating system: [Unix-like](#)", "Original author(s): Robert Coggeshall, Cliff Sp...", and "Initial release: Around 1980". At the bottom, there is a breadcrumb trail: "https://en.wikipedia.org > wiki > Sudo" followed by a vertical ellipsis and the text "sudo - Wikipedia".

Google

where was sudo made

Q All Images News Videos Shopping More Tools

About 22,400,000 results (0.55 seconds)

History. Robert Coggeshall and Cliff Spencer wrote the original subsystem around 1980 at the Department of Computer Science at **SUNY/Buffalo**.

Repository: www.sudo.ws/repos/sudo

Operating system: [Unix-like](#)

Original author(s): Robert Coggeshall, Cliff Sp...

Initial release: Around 1980

[https://en.wikipedia.org](https://en.wikipedia.org/wiki/Sudo) > wiki > Sudo

[sudo - Wikipedia](#)

Groups!

- Group name
- Password (usually unused)
- GID (Group ID)
- List of accounts which belong to the group
- All groups found in `/etc/group`
- Like security groups in Windows, Linux groups can also be used to grant users different privileges.

Fun with groups!

- groupadd and groupdel add/delete groups
 - Syntax: groupadd <group name>
 - Syntax: groupdel <group name>
- usermod lets you add/remove users to a group
 - Syntax: usermod -G <Group> <username>
- getent will let you see which users are part of a group
 - Syntax: getent group <groupname>

Package managers

- Used to install, uninstall, update and upgrade packages.
- Each distro has its own version
 - apt - Ubuntu, and Debian based
 - yum - CentOS and other Red Hat Enterprise
- To install a new package:
 - `sudo <package manager> install <package name>`

Update != Upgrade

- Update does not update your system!
 - It updates sources which keep track of new packages
- Upgrades actually downloads the new stuff
- Run update before upgrade

Remote connections (ssh)

- SSH is the most popular way of accessing and managing Linux systems remotely
- Usage: `ssh username@remote-host`
 - E.g., `ssh vasu@133.76.94.20`
- SSH can use public/private keys instead of/in conjunction with password based authentication
- Check out `ssh-keygen` and the man pages/google

Services

- Services on Linux are managed by the `systemd` service
 - Not all distros use `systemd`, but the major ones do
- `systemctl <command> <service name>`
 - `status`
 - `enable`
 - `start/stop`
- When have you used `systemctl` before?

```
[sysadmin@parrot]~  
$ sudo systemctl restart NetworkManager  
[sysadmin@parrot]~  
$
```

Environment variables

- Environment variables are a way to store information in a shell
- They can be set for the duration of a shell session with the `export` command
 - Syntax: `NEW_ENV=something`
 - Syntax: `export NEW_ENV=something`
- Environment variables can be put in shell configs and run every time a shell starts
- You can check the value of an environment variable with the `echo` command
 - `echo $NEW_ENV` would return "something"

Aliases

- Aliases are a great way to reduce repetitive and/or long commands
 - Because who doesn't like being lazy?
- The syntax is easy: `alias word='long command'`
 - Example: `alias errorlog='cat /var/log/system.log | grep error'`
- To see a list of all currently set aliases, just type `alias`
- To unset an alias, type `unalias <X>` where `<X>` is the alias you want to unset

```
# some more ls aliases
alias ll='ls -lh'
alias la='ls -lha'
alias l='ls -CF'
alias em='emacs -nw'
alias dd='dd status=progress'
alias _='sudo'
alias _i='sudo -i'
```


Pipes and redirecting things

- Redirect output to files
 - `command > outputfile.txt` (This will overwrite the file)
 - `command >> outfile.txt` (This will append to the file)
- Input file contents
 - `command < inputfile.txt`
- Pipe
 - `command | command2`
 - `cat log.txt | grep "success" | less`

Previous Commands

- `history` : Show your history on shells that keep track
 - `history -c` to clear your history
- `Ctrl + R` : Search command history
- `!!` : Rerun previous command
- `sudo !!` : Rerun as superuser (you will do this a lot)
- `<Up Arrow>` : Cycle through previous commands

Hands on 2 ctf boogaloo

The background of the slide is a solid blue gradient that transitions from a darker blue at the top to a lighter blue and then a hint of green at the bottom right. Overlaid on this background are several abstract geometric patterns. These consist of thin white lines connecting small white dots, forming a network of interconnected triangles and polygons of various sizes. Some of these shapes are more complex, resembling wireframe models of crystals or molecular structures. The overall effect is a modern, tech-oriented aesthetic.

Hands on 2 discussion

If you want to talk more about Linux, just message me, or
swing by my OH

That's all folks

Vasu will run OH on Tuesday 3/15 from 5:30 – 6:30 PM