LINUX FUNDAMENTALS

PRESENTED BY DIANA WHITNEY – AUTOMATICPOODLE

TO CYBER AEGIS

SEPTEMBER 4, 2021

WHOAMI

Diana Whitney
@automaticpoodle

Cyber Security Specialist for six years

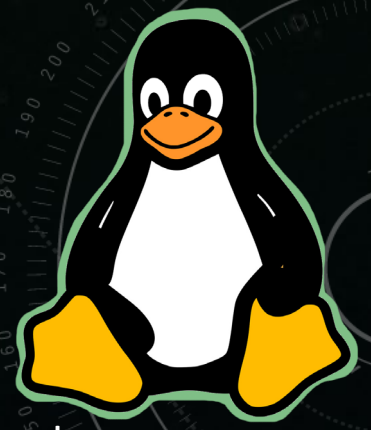
Payment Card Industry device compliance

Threat Intelligence Research

AGENDA

- Linux Overview
- The File System
  - Important files and folders
- Shells
  - Command Line
- Using Commands
  - Man pages, Help, history
- System Information
  - Whoami, pwd, network info
- Moving Around
  - cd, ls
- Files
  - Creating, editing, moving, copying
- Users and Permissions
  - chmod, sudo
- Finding Things
  - find, grep

LINUX OVERVIEW



- The Linux kernel was developed by Linus Torvalds as a free, open-source operating system kernel
- v0.01 was released in 1991 and was used in the GNU OS, a free alternative to UNIX
- The developer community comprises 5000-6000 members
- v5.13.11 is the current kernel version as of August 15, 2021
- Over 600 distributions – Ubuntu, Debian, Fedora, RedHat, Linux Mint, Kali, etc
- One of the most popular in the world – Android devices, smart cars, home appliances, enterprise servers

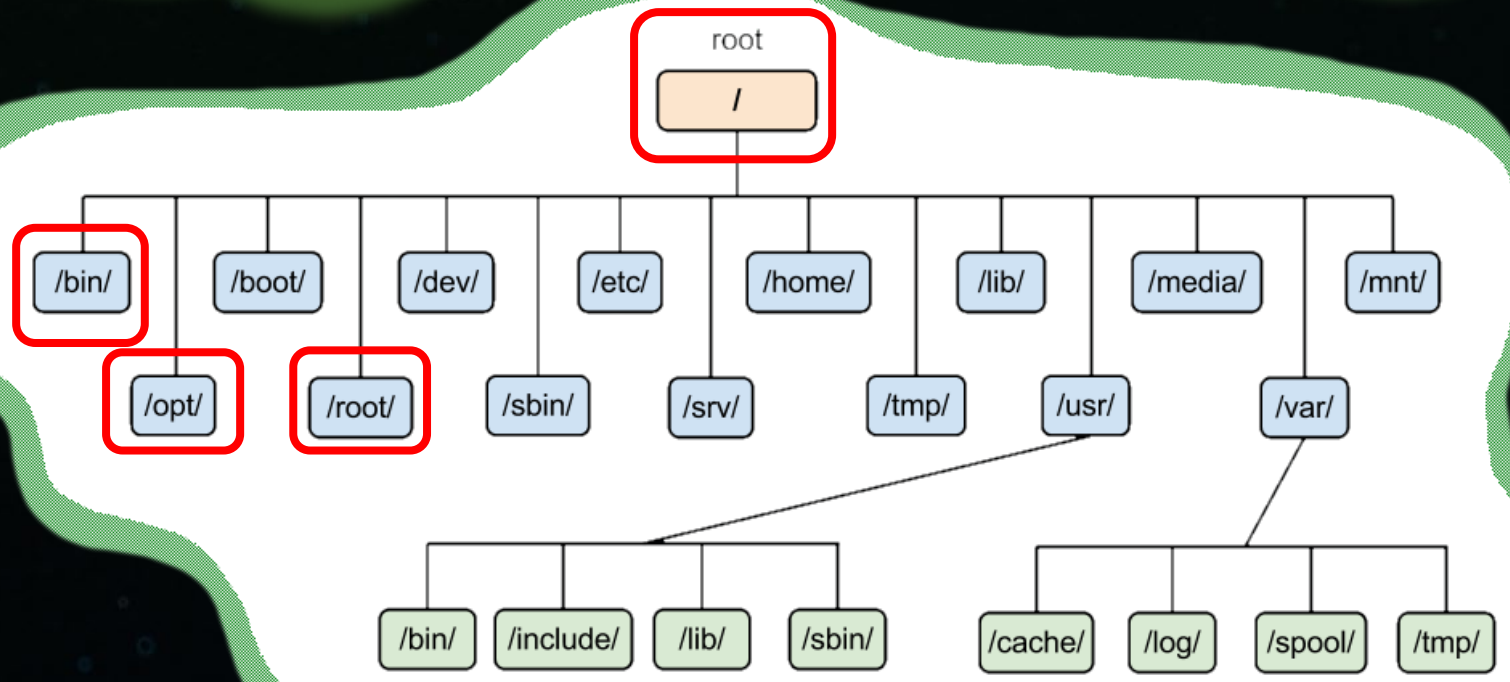
ubuntu

CORE PRINCIPALS

- Everything is a file – all configuration data is stored in text files
- Programs are small and single-purpose
- Programs can chain together to perform more complex tasks
- Designed to work with the shell or terminal, as opposed to the Graphical User Interface

/root/ – The home directory for the root user.

Root – the top-level directory. All files required to boot the OS live here.

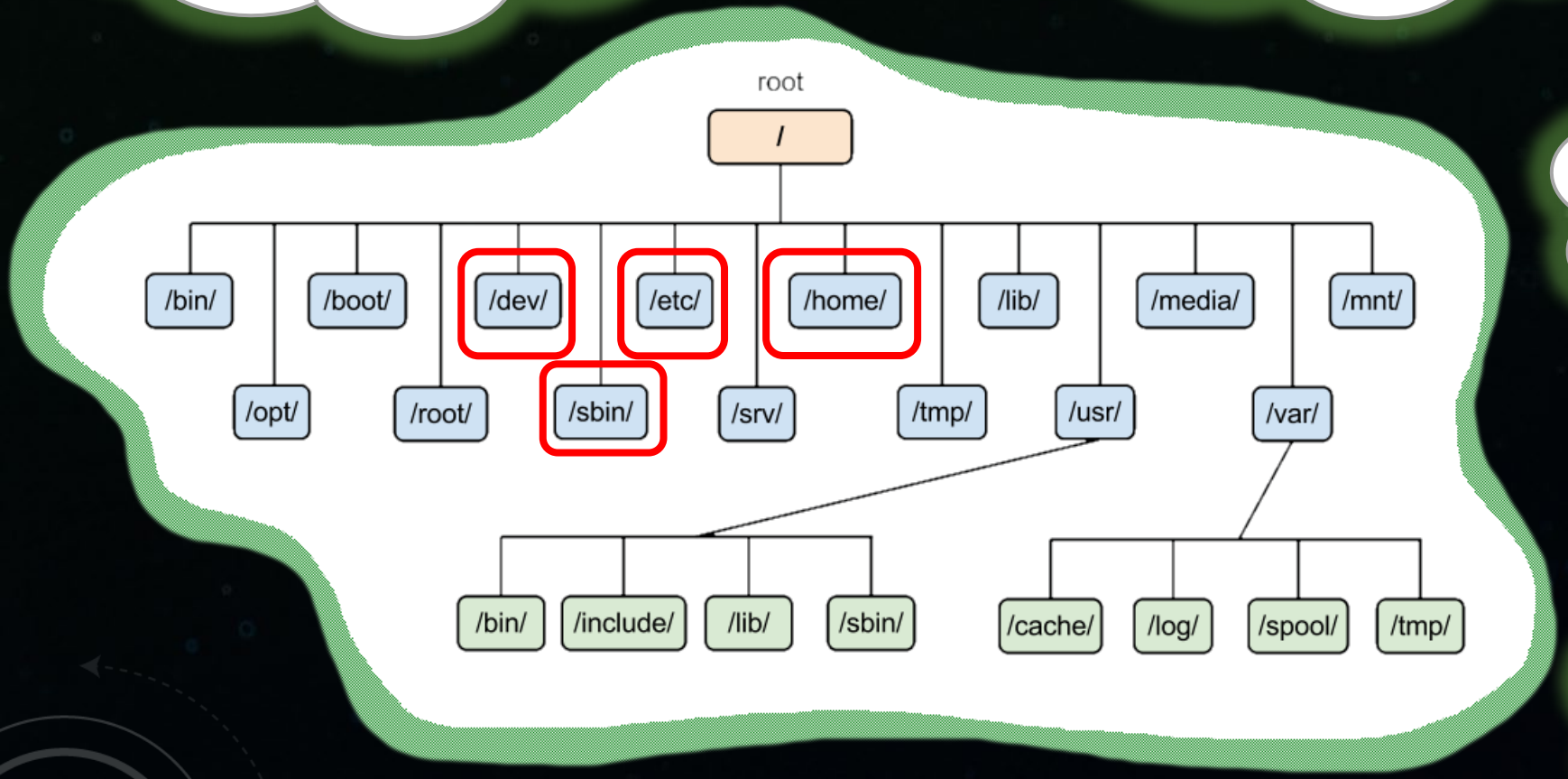


/bin – essential command binaries (or programs) are here. These binaries are essential to Linux's functionality. Example – bash, chmod

/opt – optional files, including third party tools and software go here

/home/ – contains a home folder for each user.

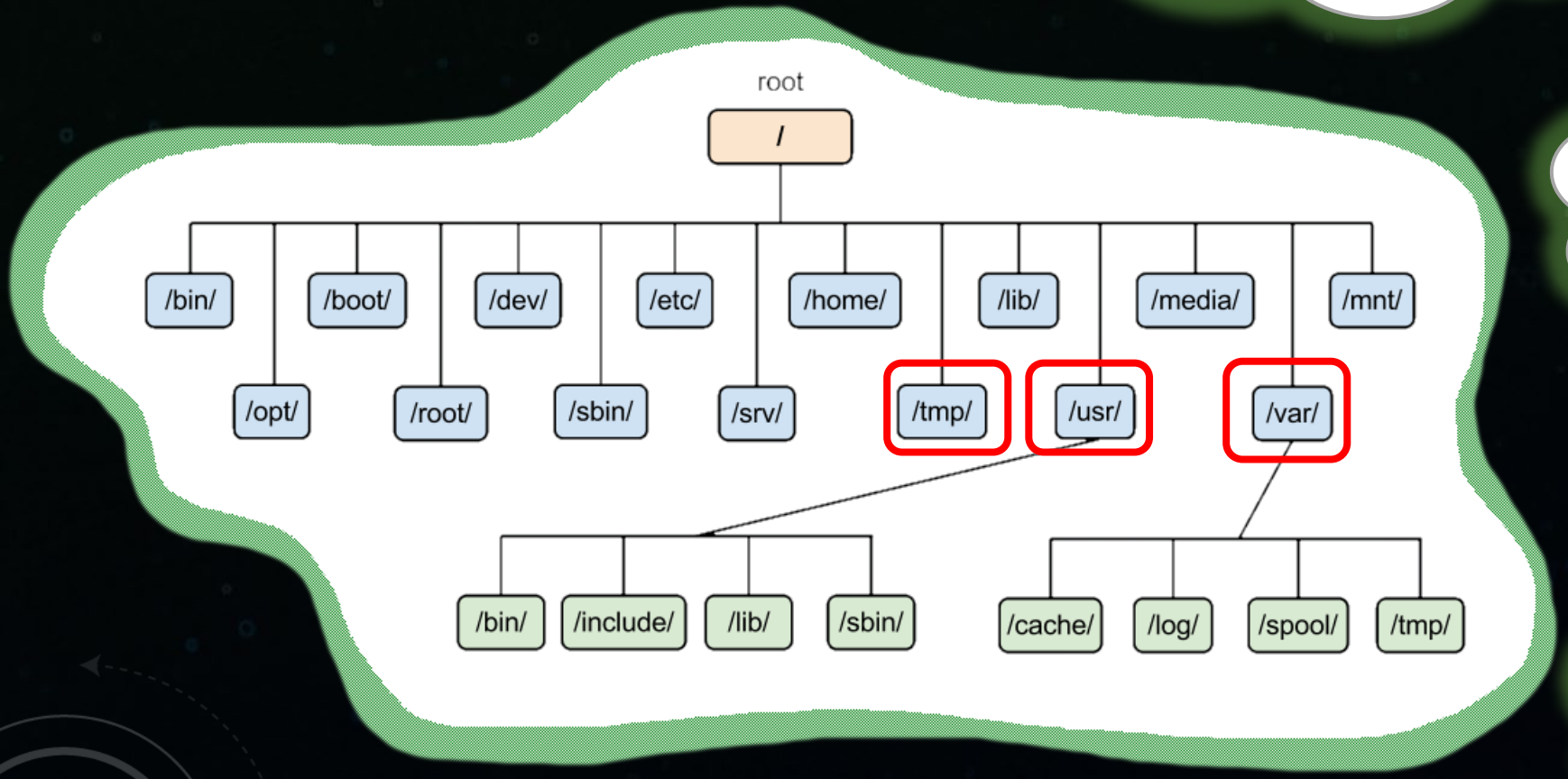
/dev/ – Devices (represented as files) are found here, for example SATA drives.



/sbin/ – System binaries. Essential binaries mainly intended to be run by the root user for system administration

/etc/ – System-wide configurations files live here.

/tmp/ – Stores temporary application files. Generally cleared upon a system restart.



/usr/ – Non-essential applications and files used by users.

/var/ – Many log files are found here.

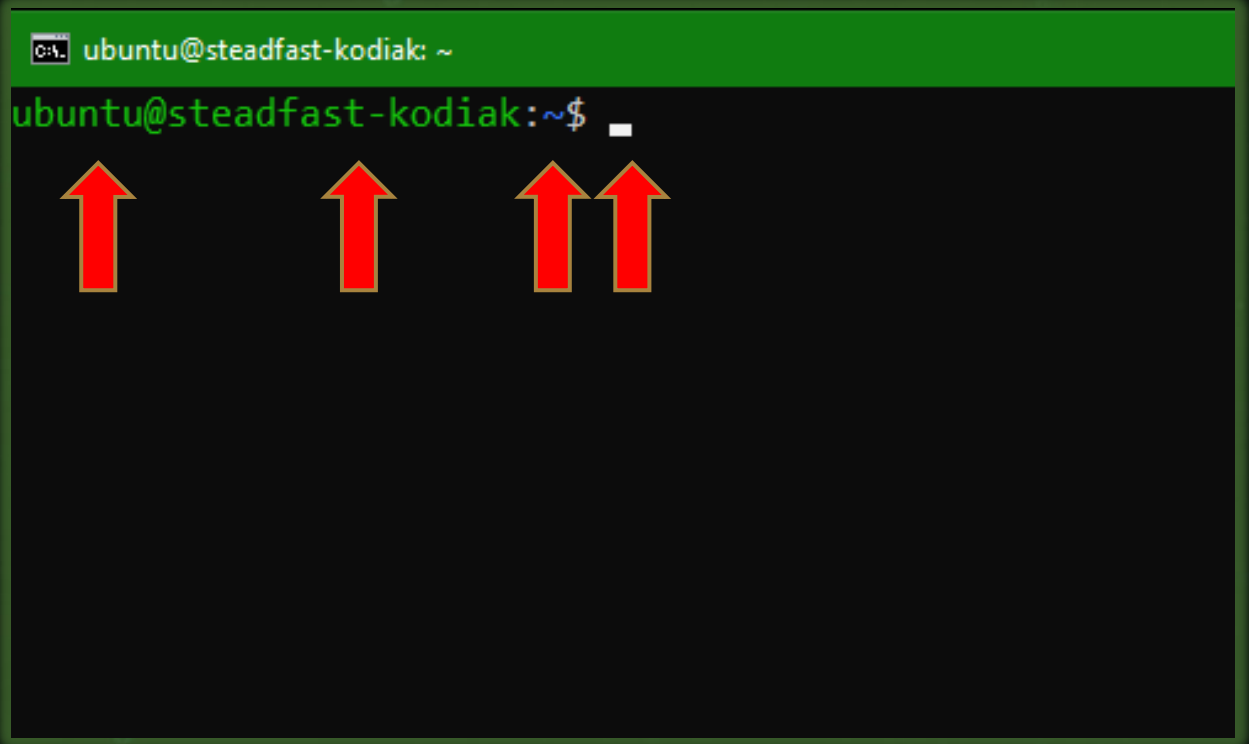
SIGNIFICANT FILES

- /dev/null – a ‘pseudo-device’ that doesn’t correspond to hardware. It produces no output, and discards all input sent to it, except to report the write operation succeeded.
- /etc/bashrc – stores configuration data relevant to the shell. For example, the prompt can be customized or command aliases can be set.
- /etc/hosts – translates hostnames or domain names to an IP address.
- /etc/passwd – A text file containing a list of the system’s accounts.
- /etc/shadow – Stores hashed passwords for a system’s accounts.

THE SHELL (OR COMMAND LINE)

- Text-based input/output (I/O) interface
- Most commonly used shell is the Bourne-Again Shell – BASH
- Others include Tcsh/Csh, Ksh, Zsh, Fish
- Everything that can be done through a Graphical User Interface (GUI) can be done through the shell
- Programs and processes can be accessed more quickly
- Scripts can be used to automate tasks
- Can be used to access remote systems through network protocols like SSH

SHELL CONTINUED…



Username

Hostname

Current working directory

Prompt

USING COMMANDS

- Introducing LS – lists the contents of a directory (similar to dir in Windows)
- Standard Input (stdin), Standard Output (stdout), Standard Error (stderr)
- Streams have a file descriptor, represented as a number:
  - stdin – 0
  - stdout – 1
  - stderr – 2

```
ubuntu@sociable-glider:~$ ls
important_stuff  linux_fun
ubuntu@sociable-glider:~$ ld

Command 'ld' not found, but can be installed with:

sudo apt install binutils

ubuntu@sociable-glider:~$
```

THE MAN PAGE

Select ubuntu@steadfast-kodiak: ~

```
ubuntu@steadfast-kodiak:~$ man ls
```

Ok that’s kind of cool but…
What else can we do?

ubuntu@steadfast-kodiak: ~

```
LS(1)                                User Commands                                LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about the FILEs (the current directory by default).  Sort entries alphabetically if none of
       -cftuvSUX nor --sort is specified.

       Mandatory arguments to long options are mandatory for short options too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

       -b, --escape
              print C-style escapes for nongraphic characters
```

OPTIONS EXERCISE…

- What if I want to…

Show all files, including hidden files starting with ‘.’ ?

Show the files in a long list

Place the most recently modified files at the top of the list

OPTIONS

```
-a, --all
       do not ignore entries starting with .

-l     use a long listing format

-t     sort by modification time, newest first
```

```
ubuntu@steadfast-kodiak: ~
ubuntu@steadfast-kodiak:~$ ls -alt
total 40
drwxr-xr-x 6 ubuntu ubuntu 4096 Aug 18 12:08 .
drwxrwxr-x 2 ubuntu ubuntu 4096 Aug 18 12:08 important_stuff
drwxrwxr-x 2 ubuntu ubuntu 4096 Aug 18 12:07 linux_fun
-rw------- 1 ubuntu ubuntu   51 Aug 18 11:45 .bash_history
drwx------ 2 ubuntu ubuntu 4096 Aug 18 11:11 .cache
drwx------ 2 ubuntu ubuntu 4096 Aug 18 11:11 .ssh
drwxr-xr-x 3 root   root   4096 Aug 18 11:11 ..
-rw-r--r-- 1 ubuntu ubuntu  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3771 Feb 25  2020 .bashrc
-rw-r--r-- 1 ubuntu ubuntu  807 Feb 25  2020 .profile
ubuntu@steadfast-kodiak:~$
```

- Hidden files and directories start with ‘.’
- . represents the current working directory
- .. represents the directory above the current directory

CLEAR AND BASH HISTORY

There's too much stuff on my screen now omg ☹

Select ubuntu@steadfast-kodiak: ~

```
ubuntu@steadfast-kodiak:~$ ls -alt
total 40
drwxr-xr-x 6 ubuntu ubuntu 4096 Aug 18 12:08 .
drwxrwxr-x 2 ubuntu ubuntu 4096 Aug 18 12:08 important_stuff
drwxrwxr-x 2 ubuntu ubuntu 4096 Aug 18 12:07 linux_fun
-rw------- 1 ubuntu ubuntu   51 Aug 18 11:45 .bash_history
drwx------ 2 ubuntu ubuntu 4096 Aug 18 11:11 .cache
drwx------ 2 ubuntu ubuntu 4096 Aug 18 11:11 .ssh
drwxr-xr-x 3 root   root   4096 Aug 18 11:11 ..
-rw-r--r-- 1 ubuntu ubuntu  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3771 Feb 25  2020 .bashrc
-rw-r--r-- 1 ubuntu ubuntu  807 Feb 25  2020 .profile
ubuntu@steadfast-kodiak:~$ clear
```

ubuntu@steadfast-kodiak: ~

```
ubuntu@steadfast-kodiak:~$
```

CLEAR AND BASH HISTORY

- Linux stores your command history in a file (ie .bash_history, .zsh_history)
- You can view this file in the command line or open it in your text editor of choice
- Alternately, you can press the up arrow in the prompt to review the last commands you entered
- Useful when re-running commands (or if you can't remember what you just did)
- Bash history can also be reviewed with the 'history' command

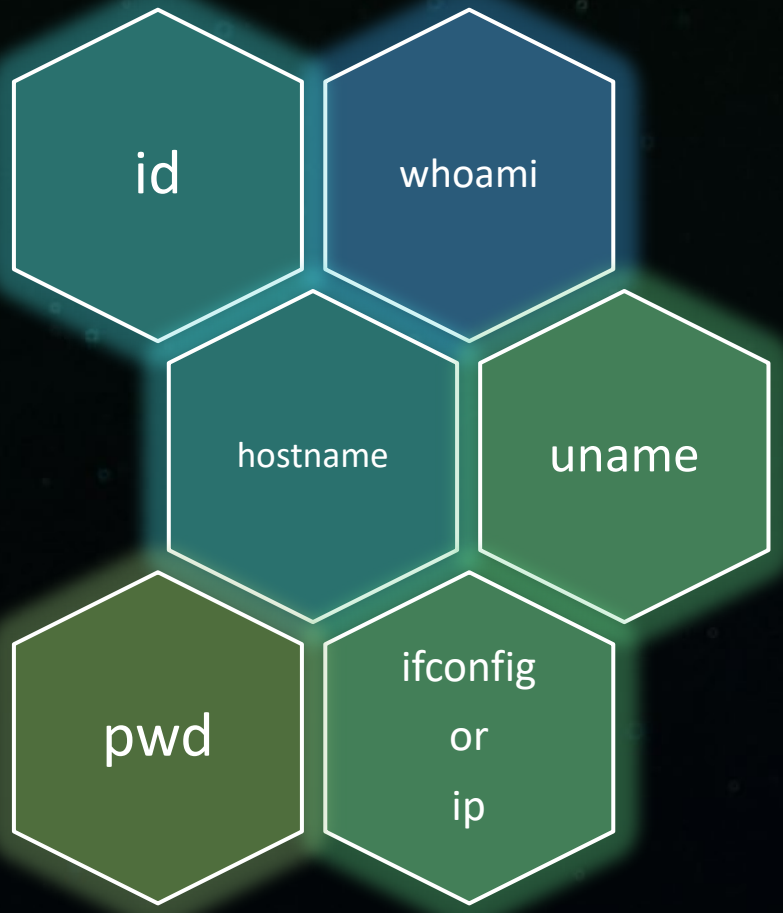
```
   49  cat sloth_fact_1
   50  echo Without sloths there would be no avocadoes >> sloth_fact_1
   51  cat sloth_fact_1
   52  history
ubuntu@spiritual-sandfish:~/linux_fun/sloth_facts$
```

APROPOS

- Can’t remember the name of the command
- Remember roughly what it does…
- The apropos command can be used to search man pages using key words or phrases as options to try and find what you’re looking for

```
ubuntu@spiritual-sandfish:~$ apropos 'working directory'
git-stash (1)        - Stash the changes in a dirty working directory away
pwd (1)              - print name of current/working directory
pwdx (1)             - report current working directory of a process
ubuntu@spiritual-sandfish:~$
```

GATHERING SYSTEM INFORMATION



id	Displays the current user's name
whoami	Show the user and group IDs (numeric values)
hostname	Shows the DNS name of the current host system
uname	Print system information
ifconfig or ip	Returns information about the network interface(s)
pwd	Prints the current working directory

GATHERING INFORMATION…

```
ubuntu@spiritual-sandfish:~$ whoami
ubuntu
```

```
ubuntu@spiritual-sandfish:~$ id
uid=1000(ubuntu) gid=1000(ubuntu) groups=1000(ubuntu),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip)
,44(video),46(plugdev),117(netdev),118(lxd)
```

```
ubuntu@spiritual-sandfish:~$ uname
Linux
```

```
ubuntu@spiritual-sandfish:~$ pwd
/home/ubuntu
```

```
ubuntu@spiritual-sandfish:~$ hostname
spiritual-sandfish
```

IFCONFIG AND IP

```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.221.132  netmask 255.255.255.0  broadcast 192.168.221.255
        inet6 fe80::20c:29ff:fe94:1aa5  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:94:1a:a5  txqueuelen 1000  (Ethernet)
        RX packets 12780  bytes 3825644 (3.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 47  bytes 6754 (6.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 24  bytes 1200 (1.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 24  bytes 1200 (1.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
ubuntu@spiritual-sandfish:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default 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 52:54:00:e1:2f:99 brd ff:ff:ff:ff:ff:ff
    inet 172.23.45.184/20 brd 172.23.47.255 scope global dynamic eth0
       valid_lft 83437sec preferred_lft 83437sec
    inet6 fe80::5054:ff:fee1:2f99/64 scope link
       valid_lft forever preferred_lft forever
```

What command lists system information?

What command shows the user/group's numeric value?

SYSTEM INFORMATION TREASURE HUNT!

1. Find the kernel release number
Flag – kernel number flag{x.x.x-xxxx-xxx}
2. Display only your group id number
Flag – four digit number flag{xxxx}

whoami

id

hostname

uname

pwd

ifconfig
or
ip

MOVING AROUND

This is boring. Let's go somewhere else.

pwd – Print Working Directory

```
ubuntu@spiritual-sandfish:~$ pwd
/home/ubuntu
```

ls – List

```
ubuntu@spiritual-sandfish:~$ ls
```

cd – Change Directory

```
ubuntu@spiritual-sandfish:~$ mkdir new_folder
ubuntu@spiritual-sandfish:~$ cd new_folder
```

mkdir – Make Directory

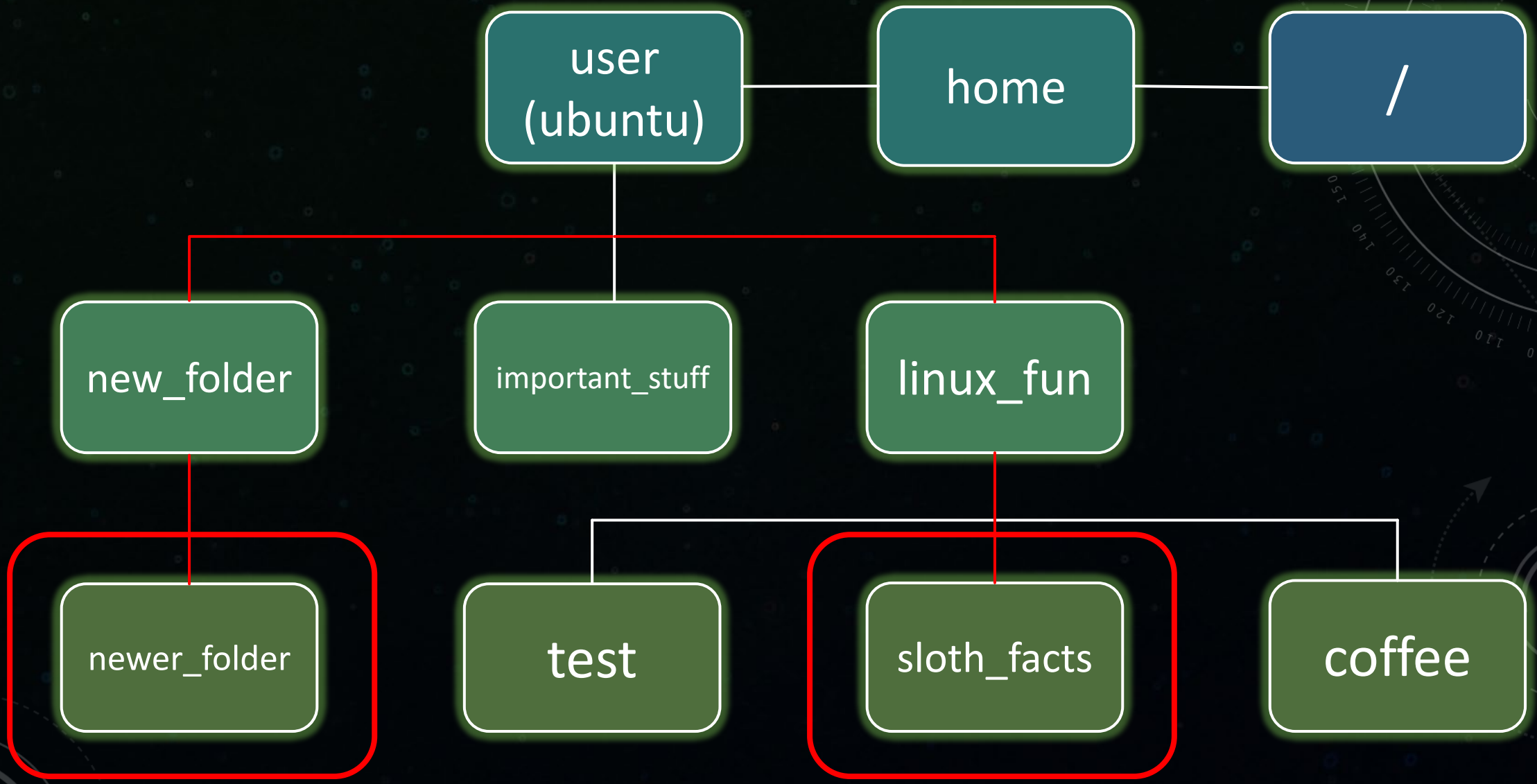
```
ubuntu@spiritual-sandfish:~/new_folder$ mkdir newer_folder
ubuntu@spiritual-sandfish:~/new_folder$ cd newer_folder
```

```
ubuntu@spiritual-sandfish:~/new_folder/newer_folder$ pwd
/home/ubuntu/new_folder/newer_folder
```

CAN WE DO THIS IN ONE COMMAND?

- Yes!
- Use the –p option (for ‘parent’) with mkdir to create any parent directories
  - mkdir –p dogs/dalmatians
- ‘dogs’ is the parent directory, and will be created as well as ‘dalmatians’
- Change directories to ‘dalmatians’ using a single command

```
ubuntu@spiritual-sandfish:~$ mkdir -p dogs/dalmatians
ubuntu@spiritual-sandfish:~$ cd dogs/dalmatians/
ubuntu@spiritual-sandfish:~/dogs/dalmatians$
```


/
home
user
(ubuntu)
new_folder
important_stuff
linux_fun
newer_folder
test
sloth_facts
coffee

RELATIVE PATH

- Relative paths start from the current working directory – newer_folder.
- Use the command ‘cd ..’ to move up a directory
  - Recall ‘..’ refers to the directory immediately above the current working directory
- This can be repeated until a parent directory of sloth_facts is reached
  - Hint: you can also change multiple directories at once – cd ../..

```
ubuntu@spiritual-sandfish:~/new_folder/newer_folder$ cd ..
ubuntu@spiritual-sandfish:~/new_folder$ cd ..
ubuntu@spiritual-sandfish:~$ cd linux_fun/sloth_facts/
ubuntu@spiritual-sandfish:~/linux_fun/sloth_facts$
```

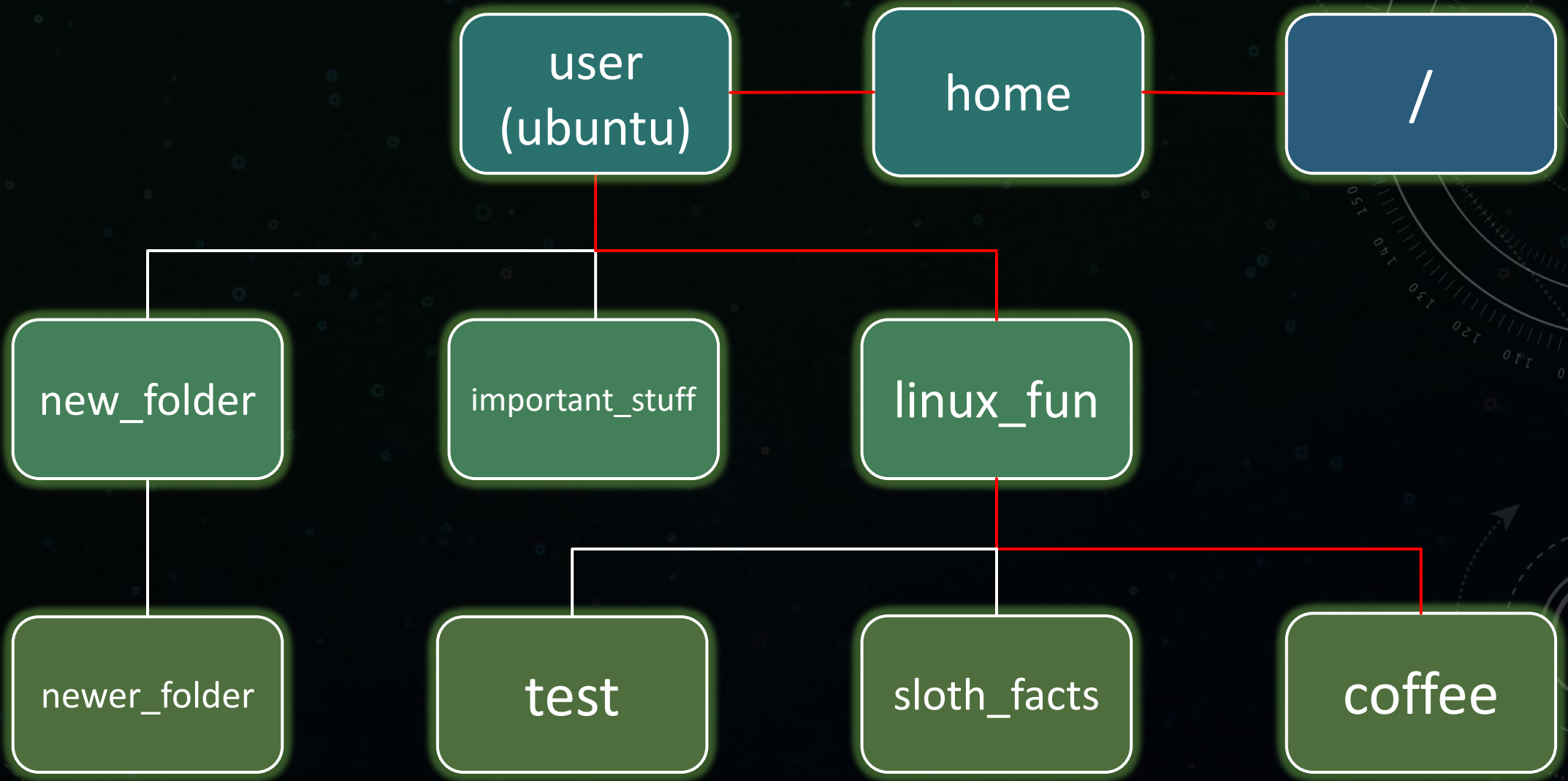
ABSOLUTE PATH

- Another, less repetitive way to access a file or folder is to use its absolute path
- Absolute Path contains a complete list of all elements (ie folders) starting at the root directory / and ending at the destination
- To move from sloth_facts to coffee using the absolute path, use the command:

cd /home/[username]/linux_fun/coffee

```
ubuntu@spiritual-sandfish:~/linux_fun$ cd /home/ubuntu/linux_fun/coffee/
ubuntu@spiritual-sandfish:~/linux_fun/coffee$
```

cd /home/[username]/linux_fun/coffee



FUN WITH FILES

touch – creates an empty file

Syntax:
touch [filename]

Make a file called sloth_fact_1 in the current directory (~/linux_fun/coffee)

```
ubuntu@spiritual-sandfish:~/linux_fun/coffee$ touch sloth_fact_1
ubuntu@spiritual-sandfish:~/linux_fun/coffee$ ls
sloth_fact_1
```

FUN WITH FILES

touch – creates an empty file

cp – Copies a file or folder

Syntax:
Copy single file in same directory:
cp src_file dest_file

Copy single file to another directory:
cp src_file dest_file dir_that_exists

Copy all files from one directory to another
cp –R src_dir dest_dir

Sloth facts are pretty rad so let's make a copy and call it sloth_fact_2

```
ubuntu@spiritual-sandfish:~/linux_fun/coffee$ cp sloth_fact_1 sloth_fact_2
ubuntu@spiritual-sandfish:~/linux_fun/coffee$ ls
sloth_fact_1  sloth_fact_2
```

FUN WITH FILES

touch – creates an empty file

cp – Copies a file or folder

mv – Moves a file or folder

Syntax:
Move single file in same directory:
mv src_file dest_file

This essentially renames the file.

Move to new directory:
mv src_file path/dest_file

Option –n (no clobber) prevents existing files from being overwritten
mv –n src_file _dest_file

Wait a sec this is not the sloth_facts folder. Move both sloth fact files to sloth_facts

```
ubuntu@spiritual-sandfish:~/linux_fun/coffee$ mv sloth_fact_1 /home/ubuntu/linux_fun/sloth_facts/sloth_fact_1
ubuntu@spiritual-sandfish:~/linux_fun/coffee$ mv sloth_fact_2 /home/ubuntu/linux_fun/sloth_facts/sloth_fact_2
ubuntu@spiritual-sandfish:~/linux_fun/coffee$ cd /home/ubuntu/linux_fun/sloth_facts/
ubuntu@spiritual-sandfish:~/linux_fun/sloth_facts$ ls
sloth_fact_1  sloth_fact_2
```

FUN WITH FILES

touch – creates an empty file

cp – Copies a file or folder

mv – Moves a file or folder

rm – Remove a file or folder

Syntax:
Remove single file in same directory:
rm file

Remove a directory:
rm –d directory

Remove a directory and all its contents
rm –r directory

Remove the directory ‘test’ and all its contents. It looked at you funny

```
ubuntu@spiritual-sandfish:~/linux_fun/sloth_facts$ rm -r /home/ubuntu/linux_fun/test
ubuntu@spiritual-sandfish:~/linux_fun/sloth_facts$ cd ..
ubuntu@spiritual-sandfish:~/linux_fun$ ls
coffee  sloth_facts
ubuntu@spiritual-sandfish:~/linux_fun$
```

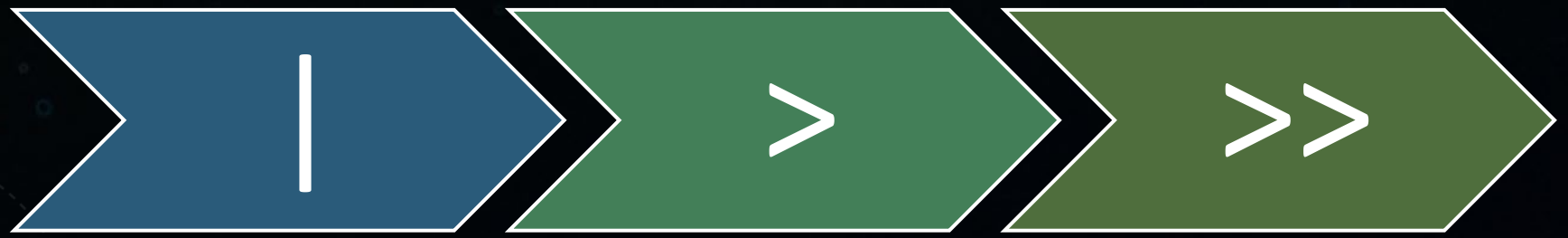
EVEN MORE FUN WITH FILES

- cat – short for ‘concatenate’, outputs the file’s contents
  - Syntax – cat [filename]
  - Change directories to sloth_facts and try it on one of the files
  - Nothing is there… yet
- echo – displays a line of text or string passed to it as an argument
  - Syntax - echo [stuff]
  - Try it out! → echo hello

```
ubuntu@spiritual-sandfish:~/linux_fun$ echo hello
hello
ubuntu@spiritual-sandfish:~/linux_fun$ echo wow that is super not impressive
wow that is super not impressive
```

REDIRECTING INPUT

- | – Pipe lets you chain two or more commands. The output (stdout) of one command will be used as the input (stdin) for the next command.
- > - An output redirector that takes the output from a command and sends it somewhere else. If output is redirected to a file, the contents of the file are overwritten.
- >> - Similar to the previous output redirector, but the redirected output will be appended to the end of a file – it will not overwrite the file contents.



SLOTH FACTS

- Using echo, put your favourite sloth fact into sloth_fact_1
- Did it work? Try reading the file using the ‘cat’ command
- Super sweet. Add on another fact without overwriting the first

echo [text] > [filename]

```
ubuntu@spiritual-sandfish:~$ cd linux_fun/sloth_facts/
ubuntu@spiritual-sandfish:~/linux_fun/sloth_facts$ echo All sloths are handsome and soft > sloth_fact_1
ubuntu@spiritual-sandfish:~/linux_fun/sloth_facts$ cat sloth_fact_1
All sloths are handsome and soft
ubuntu@spiritual-sandfish:~/linux_fun/sloth_facts$ echo Without sloths there would be no avocados >> sloth_fact_1
ubuntu@spiritual-sandfish:~/linux_fun/sloth_facts$ cat sloth_fact_1
All sloths are handsome and soft
Without sloths there would be no avocados
ubuntu@spiritual-sandfish:~/linux_fun/sloth_facts$
```

SLOTH FACTS 2

- The Nano text editor can be used in the command line to edit the file contents in a more friendly manner
- Edit sloth_fact_2 using nano:
  - nano sloth_fact_2

```
ubuntu@spiritual-sandfish:~/linux_fun/sloth_facts$ nano sloth_fact_2
```

```
ubuntu@spiritual-sandfish: ~/linux_fun/sloth_facts
  GNU nano 4.8                    sloth_fact_2                    Modified
Sloths can move faster in water than on land

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text
```

- To leave nano, press CTRL + X
- Save, Don't Save, or Cancel

```
Save modified buffer?
 Y Yes
 N No           ^C Cancel
```

FINDING THINGS

- Use commands to search the filesystem the user has access to
- Find – if the filename is known, use find [starting_point] –name [filename]
- A starting point '.' means current working directory.
- '/' will search the whole filesystem starting with the root directory
- Wildcards '*' can also be used, ie to find all files ending with a .txt extension using the current working directory as a starting point, use find . –name *.txt

```
ubuntu@spiritual-sandfish:~$ find . -name sloth_fact_1
./linux_fun/sloth_facts/sloth_fact_1
```

```
ubuntu@spiritual-sandfish:~$ find . -name passwd
```

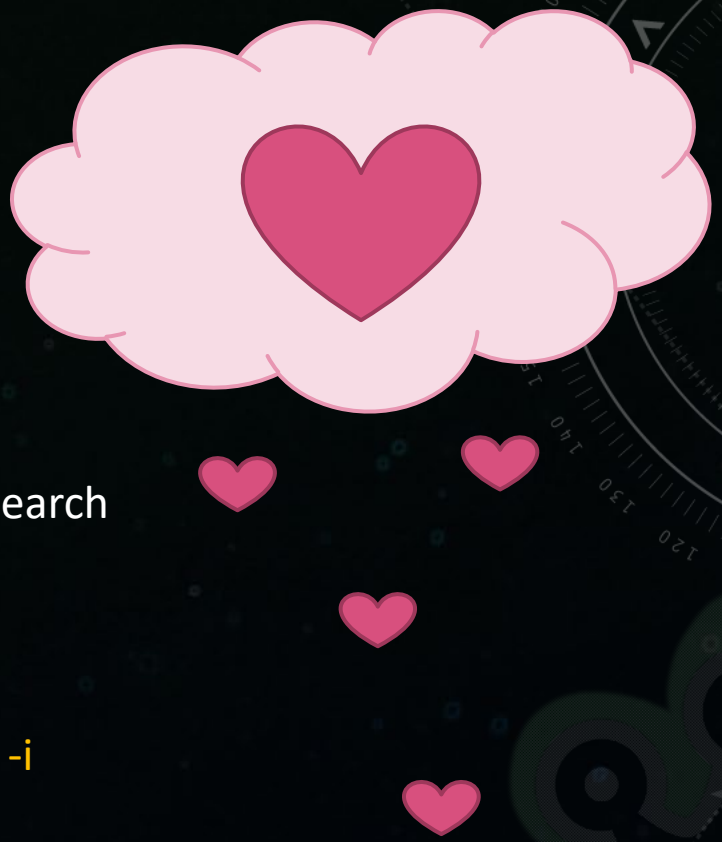
```
ubuntu@spiritual-sandfish:~$ find / -name passwd
find: '/lost+found': Permission denied
find: '/run/udisks2': Permission denied
find: '/run/user/1000/inaccessible': Permission denied
find: '/run/sudo': Permission denied
find: '/run/cryptsetup': Permission denied
find: '/run/lvm': Permission denied
find: '/run/systemd/unit-root': Permission denied
find: '/run/systemd/inaccessible': Permission denied
find: '/run/lock/lvm': Permission denied
find: '/run/initramfs': Permission denied
/usr/share/doc/passwd
/usr/share/bash-completion/completions/passwd
/usr/share/lintian/overrides/passwd
/usr/bin/passwd
```

MAKE STDERR GO AWAY

- The lines starting with ‘find’ are not considered standard output – they are standard error messages
- Recall that the redirect and pipe commands apply to standard output only, not standard error
- We can ‘filter’ error messages out by sending them to the pseudo device, /dev/null
- To redirect standard error, modify the redirect command from > to 2>
- find / -name passwd 2> /dev/null

```
ubuntu@sociable-glider:~$ find / -name passwd 2> /dev/null
/etc/pam.d/passwd
/etc/passwd
/usr/bin/passwd
/usr/share/lintian/overrides/passwd
/usr/share/bash-completion/completions/passwd
/usr/share/doc/passwd
/snap/core18/2128/etc/pam.d/passwd
/snap/core18/2128/etc/passwd
/snap/core18/2128/usr/bin/passwd
/snap/core18/2128/usr/share/bash-completion/completions/passwd
/snap/core18/2128/usr/share/doc/passwd
/snap/core18/2128/usr/share/lintian/overrides/passwd
/snap/core18/2128/var/lib/extrausers/passwd
```

GREP



- Grep – used to search the contents of a file for a specific string
- Good to use when ‘cat’ outputs way more than we could manually search
- grep [string] [filename] – will return the line containing the term
- Case insensitive – grep –i [string] [filename]
- Can be chained with other commands using pipe [command] | grep -i [string]

GREP…

- cat /etc/passwd
- cat /etc/passwd | grep [username]

```
ubuntu@sociable-glider:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
```

```
ubuntu@sociable-glider:~$ cat /etc/passwd | grep ubuntu
ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash
```

PASSWRD?

1. Username
2. Password
3. User ID
4. Group ID
5. User ID Info
6. Home Directory
7. Shell

```
ubuntu@sociable-glider:~$ cat /etc/passwd | grep ubuntu
ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash
```

1 2 3 4 5 6 7

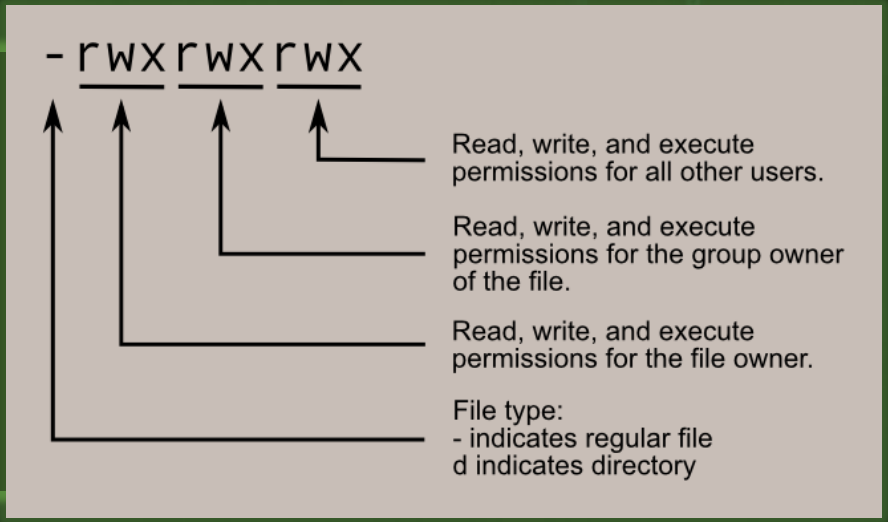
SHADOW FILE

- shadow
- Can you find it?
- find / -name shadow 2> /dev/null
- cat /etc/shadow

LET'S TALK ABOUT PERMISSIONS

- Files and folders have permissions that determine who can read, write, and execute them
- View this information with ls –al

```
ubuntu@sociable-glider:~$ ls -al
total 36
drwxr-xr-x 6 ubuntu ubuntu 4096 Aug 29 13:41 .
drwxr-xr-x 3 root   root   4096 Aug 29 11:14 ..
-rw-r--r-- 1 ubuntu ubuntu  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3771 Feb 25  2020 .bashrc
drwx------ 2 ubuntu ubuntu 4096 Aug 29 11:30 .cache
-rw-r--r-- 1 ubuntu ubuntu  807 Feb 25  2020 .profile
drwx------ 2 ubuntu ubuntu 4096 Aug 29 11:14 .ssh
drwxrwxr-x 2 ubuntu ubuntu 4096 Aug 29 13:02 important_stuff
drwxrwxr-x 5 ubuntu ubuntu 4096 Aug 29 13:03 linux_fun
```



SHADOW FILE PERMISSIONS

- ls –l /etc/shadow

```
ubuntu@sociable-glider:~$ ls -l /etc/shadow
-rw-r----- 1 root shadow 1025 Aug 29 14:07 /etc/shadow
```

1 2 3 4 5 6

1. Permissions
   - - - shadow is a file (as opposed to a directory, denoted by d)
   - rw- - The user root can read and write to this file
   - r-- - Members of the group shadow may read this file
   - --- - Other users may not read, write, or execute this file
2. Indicates the user owning the file is root,
3. Indicates the group owning the file is shadow
4. Process ID
5. Date
6. File/Directory name

SUPER USER DO

- Prefixing a command with ‘sudo’ allows you to run commands with elevated privileges
- (As the root user)
- Not all users are permitted to do this – fortunately we are on the sudoers list
- sudo cat /etc/shadow | grep ubuntu
- This will require the system’s password – P@ssw0rd!

```
ubuntu@sociable-glider:~$ sudo cat /etc/shadow | grep ubuntu
ubuntu:$6$NMCcJTDB6JbXsrMU$zI2TKuqnFNta1Z9tE639LTW5vwfl51khmE86pOaWnJThEr1VUZexnCztYymPpurnJDS4CiaZDxbe3WU/LBQw3.:18868:0:99999:7:::
```

- Username
- 6 - Hash algorithm used (SHA-512)
- Password Hash
- Information about date password was changed, expiry date, and other information

PERMISSIONS EXERCISE

- Move to the important_stuff directory
  - If you're not in the home directory, cd ~
  - cd important_stuff
- touch super_secret_file_1
- touch super_secret_file_2
- ls –l
- What are the permissions for the User? Group? Everyone else?

```
ubuntu@sociable-glider:~/important_stuff$ ls -l
total 0
-rw-rw-r-- 1 ubuntu ubuntu 0 Aug 29 14:14 super_secret_file
```

User – Read/Write

Group – Read/Write

Others - Read

- We need to lock these things down so only the user can read, write, and execute it, and your group can only read it

CHANGE OWNERSHIP

- chown: change ownership of a file or directory
- Syntax: chown [options] [user]:[group] [file/directory]
  - chown root super_secret_file_1
- Use sudo!
  - sudo !! will rerun your previous command as root

```
ubuntu@sociable-glider:~/important_stuff$ chown root super_secret_file_1
chown: changing ownership of 'super_secret_file_1': Operation not permitted
```

```
ubuntu@sociable-glider:~/important_stuff$ sudo !!
sudo chown root super_secret_file_1
```

```
ubuntu@sociable-glider:~/important_stuff$ ls -l
total 0
-rw-rw-r-- 1 root   ubuntu 0 Aug 29 15:43 super_secret_file_1
-rw-rw-r-- 1 ubuntu ubuntu 0 Aug 29 15:44 super_secret_file_2
```

CHANGE PERMISSIONS

- chmod: Change permissions for a file or directory
- Using Symbolic Notation
- Syntax: chmod [options] [who] [+/-/=] [permissions] [file/directory]
- For example, to add read and write file permissions to a user:
  - chmod u+rw [filename]
- To give a user execute permissions AND remove a group's execute permissions for a directory and all its contents recursively:
  - chmod –R u+x,g-x [directory]

Who	Symbol
• User • Group • Others • All	• u • g • o • a

Operator	Meaning
• + • - • =	• Add • Remove • Equals

CHANGE THOSE PERMISSIONS

- We want the user (root) to be able to read, write, and execute it
- Your group can only read it
- Set the permissions of super_secret_file_1 using symbolic notation

super_secret_file_1

```
-rw-rw-r-- 1 root   ubuntu 0 Aug 29 15:43 super_secret_file_1
-rw-rw-r-- 1 ubuntu ubuntu 0 Aug 29 15:44 super_secret_file_2
```

- Syntax: chmod [options] [who – u,g,o] [+/-/=] [permissions - rwx] [file/directory]

sudo

chmod

u+x,g-w,o-r

CHANGE PERMISSIONS 2

- Using Octal Notation
- Syntax: chmod [permissions – user-group-others] [file/directory]
  - Permissions represented by a single number for each type
  - None = 0, Execute = 1, Write = 2, Read =4. Everything else is simply adding these together.
- Give the User Read and Write permissions and Read only for Group and Others:
  - chmod 644 [file]

Permission	Symbol	Number
None	---	0
Execute	--x	1
Write	-w-	2
Execute + Write	-wx	3
Read	r--	4
Read + Execute	r-x	5
Read + Write	rw-	6
All	rwx	7

CHANGE THOSE PERMISSIONS

chmod

- The user (ubuntu) can read, write, and execute it
- Your group can only read it
- Set the permissions of super_secret_file_2 using octal notation

740

```
-rw-rw-r-- 1 root   ubuntu 0 Aug 29 15:43 super_secret_file_1
-rw-rw-r-- 1 ubuntu ubuntu 0 Aug 29 15:44 super_secret_file_2
```

- Syntax: chmod [permissions] [file/directory]
- None = 0, Execute = 1, Write = 2, Read = 4

super_secret_file_2

RESULTS...

```
ubuntu@sociable-glider:~/important_stuff$ sudo chmod u+x,g-w,o-r super_secret_file_1
ubuntu@sociable-glider:~/important_stuff$ ls -l
total 0
-rwxr----- 1 root   ubuntu 0 Aug 29 15:43 super_secret_file_1
-rw-rw-r-- 1 ubuntu ubuntu 0 Aug 29 15:44 super_secret_file_2
ubuntu@sociable-glider:~/important_stuff$ chmod 740 super_secret_file_2
ubuntu@sociable-glider:~/important_stuff$ ls -l
total 0
-rwxr----- 1 root   ubuntu 0 Aug 29 15:43 super_secret_file_1
-rwxr----- 1 ubuntu ubuntu 0 Aug 29 15:44 super_secret_file_2
```

```
ubuntu@sociable-glider:~/important_stuff$ ./super_secret_file_2
```

CAPTURE THE FLAG !

Good luck!

- Start from the home directory - cd ~
- Flags contain the word ‘flag’, but be aware of case sensitivity
- Flag 1 – Find the file ‘BuriedFile.txt’. Read the contents to find the flag
  - Hints: find, cat, autocomplete using tab
- Flag 2 – Find the bigFile.txt. Read the contents to find the flag, but beware – that file is GIANT
  - Hints – find, cat, grep, pipe |, case sensitivity
- Flag 3 – Find exeFile. Change it so it can be executed by the user, then run it to get the flag
  - Hints - chmod, ./

You can do it!

THANK YOU!