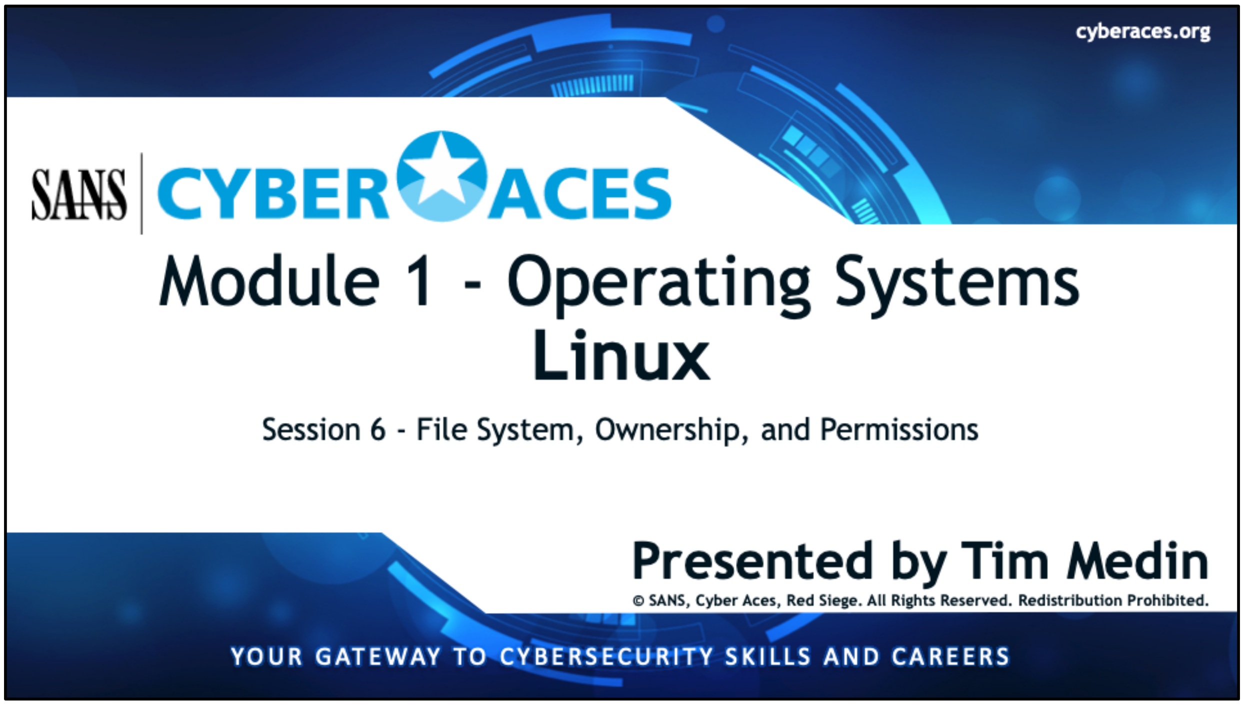

cyberaces.org

SANS | CYBER ACES

# Module 1 - Operating Systems Linux

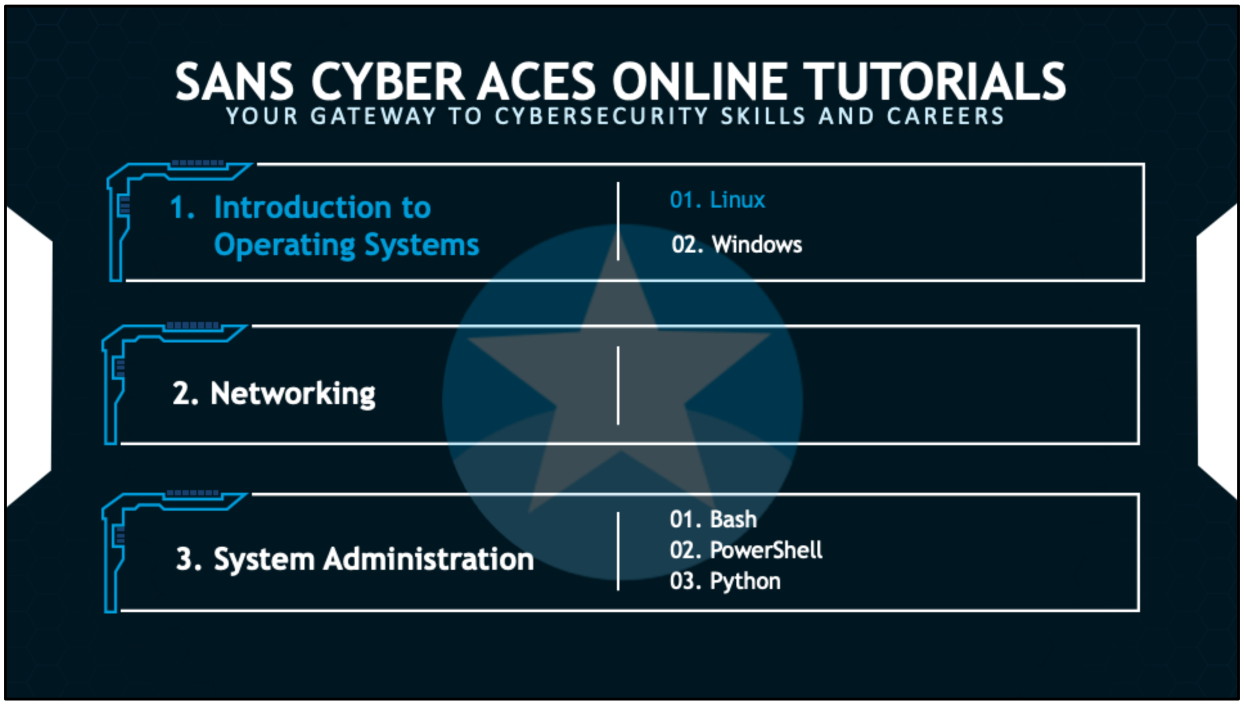
Session 6 - File System, Ownership, and Permissions

Presented by Tim Medin

YOUR GATEWAY TO CYBERSECURITY SKILLS AND CAREERS

Welcome to Cyber Aces Online Module 1. In this session we will examine the Linux file system, file ownership, and file permissions.

Content in this session has been developed by Tom Hessman, Tim Medin, Mark Baggett, Doug Burks, Michael Coppola, Russell Eubanks, Ed Skoudis, and Red Siege.



SANS CYBER ACES ONLINE TUTORIALS

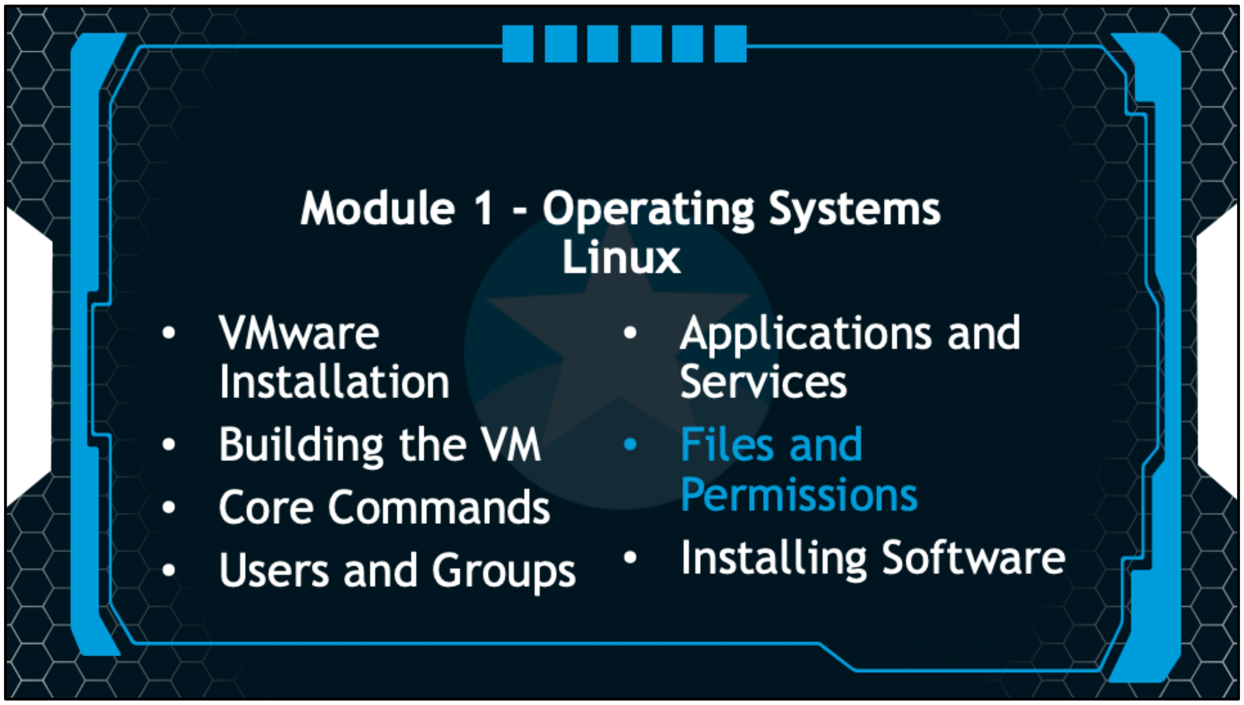
YOUR GATEWAY TO CYBERSECURITY SKILLS AND CAREERS

| | |
|---|---|
| 1. Introduction to Operating Systems | 01. Linux<br>02. Windows |
| 2. Networking | |
| 3. System Administration | 01. Bash<br>02. PowerShell<br>03. Python |

This training material was originally developed to help students, teachers, and mentors prepare for the Cyber Aces Online Competition. This module focuses on the basics of what an operating systems is as well as the two predominant OS's, Windows and Linux. This session is part of Module 1, Introduction to Operating Systems. This module is split into two sections, Linux and Windows. In this session, we will continue our examination of Linux.

The three modules of Cyber Aces Online are Operating Systems, Networking, and System Administration.

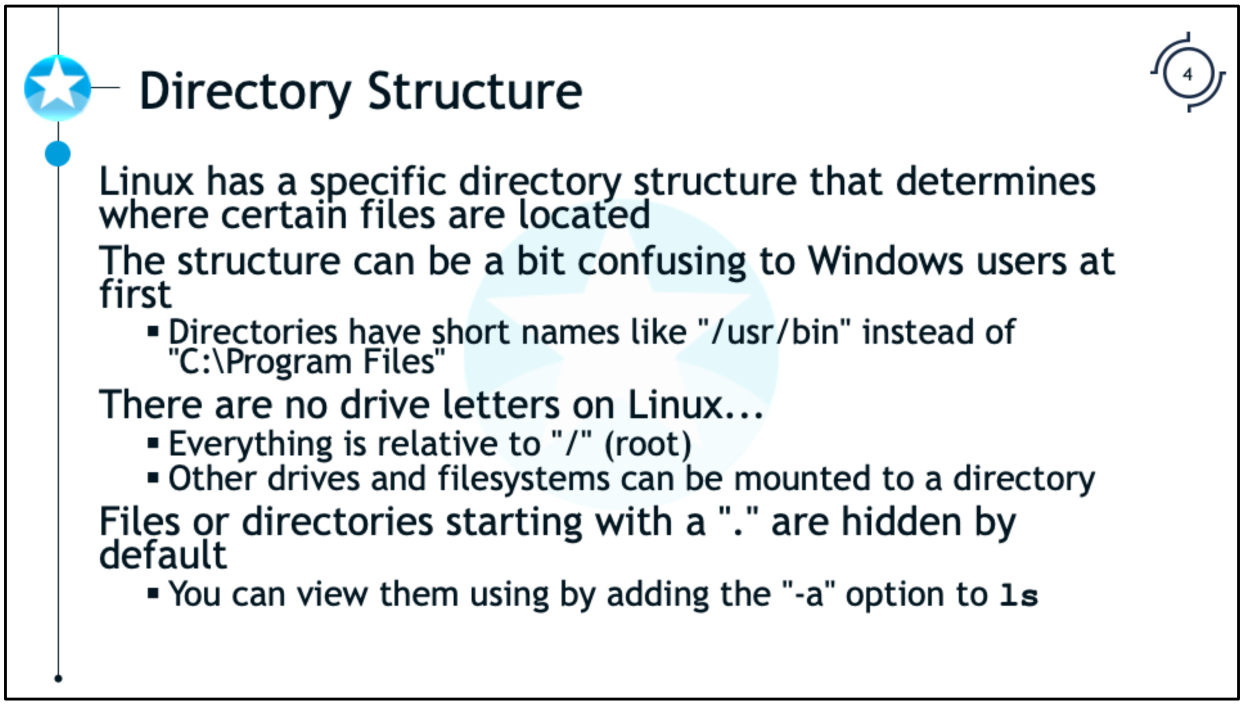
For more information about the Cyber Aces program, please visit the Cyber Aces website at https://CyberAces.org/.



## Module 1 - Operating Systems Linux

- VMware Installation
- Building the VM
- Core Commands
- Users and Groups
- Applications and Services
- Files and Permissions
- Installing Software

In this session we will discuss the Linux file system and permissions.

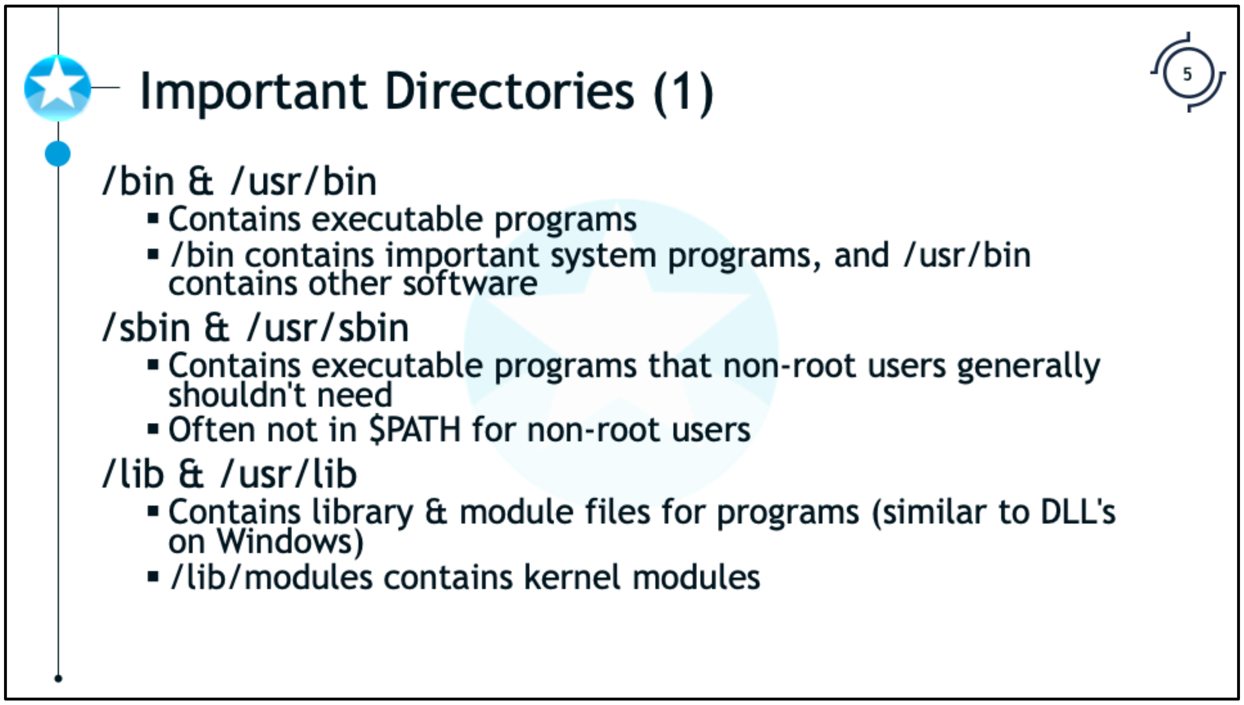


## Directory Structure

- Linux has a specific directory structure that determines where certain files are located
- The structure can be a bit confusing to Windows users at first
  - Directories have short names like "/usr/bin" instead of "C:\Program Files"
- There are no drive letters on Linux...
  - Everything is relative to "/" (root)
  - Other drives and filesystems can be mounted to a directory
- Files or directories starting with a "." are hidden by default
  - You can view them using by adding the "-a" option to **`ls`**

Linux, like Windows, has specific directories for users to store their files and other directories for the Operating System to store its files. There are directories for the Operating System to store executable programs and separate directories for it to store libraries (Linux libraries are similar to Windows DLLs). Unlike Windows, every aspect of the Linux OS is represented as a file. The operating system's memory, input/output devices, and network adapters are all represented as files in the file system.

Files or directories starting with a dot (".") are hidden by default, similar to the "hidden" file attribute on Windows.  You can view them using the "-a" option to **`ls`**.



## Important Directories (1)

/bin & /usr/bin
- Contains executable programs
- /bin contains important system programs, and /usr/bin contains other software

/sbin & /usr/sbin
- Contains executable programs that non-root users generally shouldn't need
- Often not in $PATH for non-root users

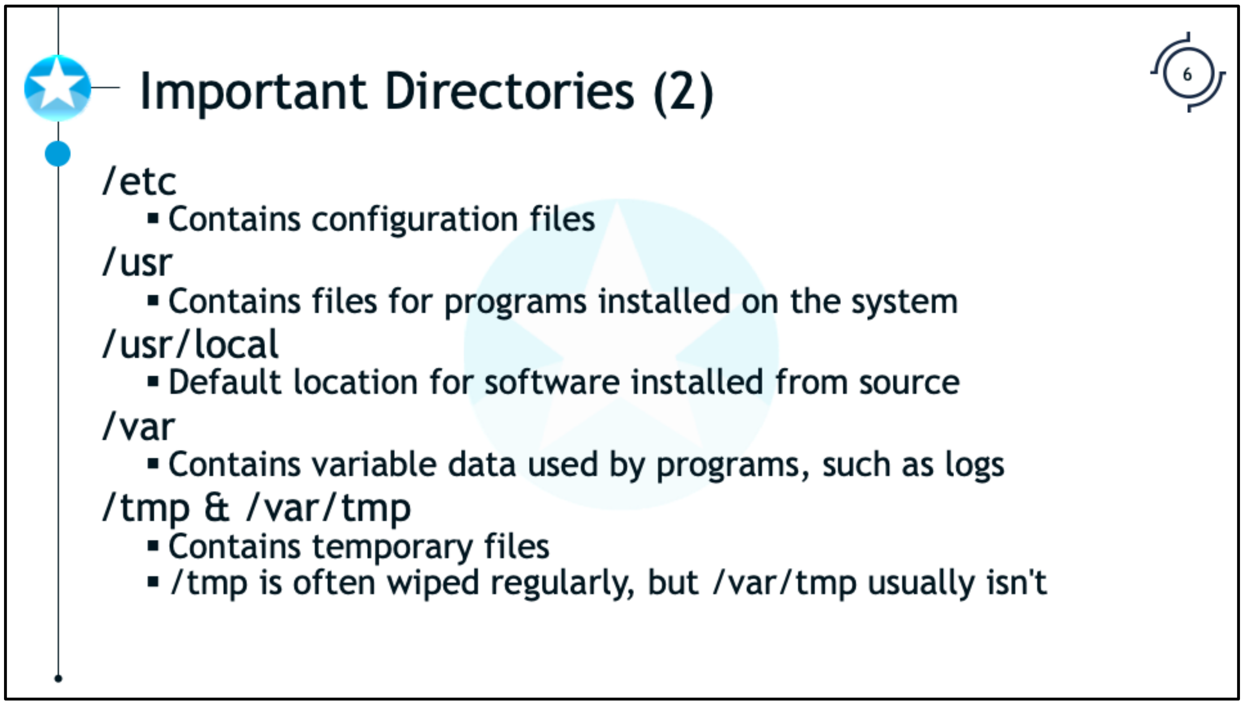
/lib & /usr/lib
- Contains library & module files for programs (similar to DLL's on Windows)
- /lib/modules contains kernel modules

Here are some of the most important directory locations on a Linux system:

**/bin & /usr/bin**: These contain executable programs. /bin contains important system programs (such as cp, ls, and mv), while /usr/bin contains other software installed on the system.

**/sbin & /usr/sbin**: These contain executable programs that non-root users generally shouldn't need, such as "ifconfig" for network interface configuration. Depending on the distribution, these directories are sometimes not in a regular user's $PATH.

**/lib & /usr/lib**: These contain library and module files for programs, which are similar to DLL files on Windows. /lib/modules contains kernel modules.



## Important Directories (2)

/etc
- Contains configuration files

/usr
- Contains files for programs installed on the system

/usr/local
- Default location for software installed from source

/var
- Contains variable data used by programs, such as logs

/tmp & /var/tmp
- Contains temporary files
- /tmp is often wiped regularly, but /var/tmp usually isn't

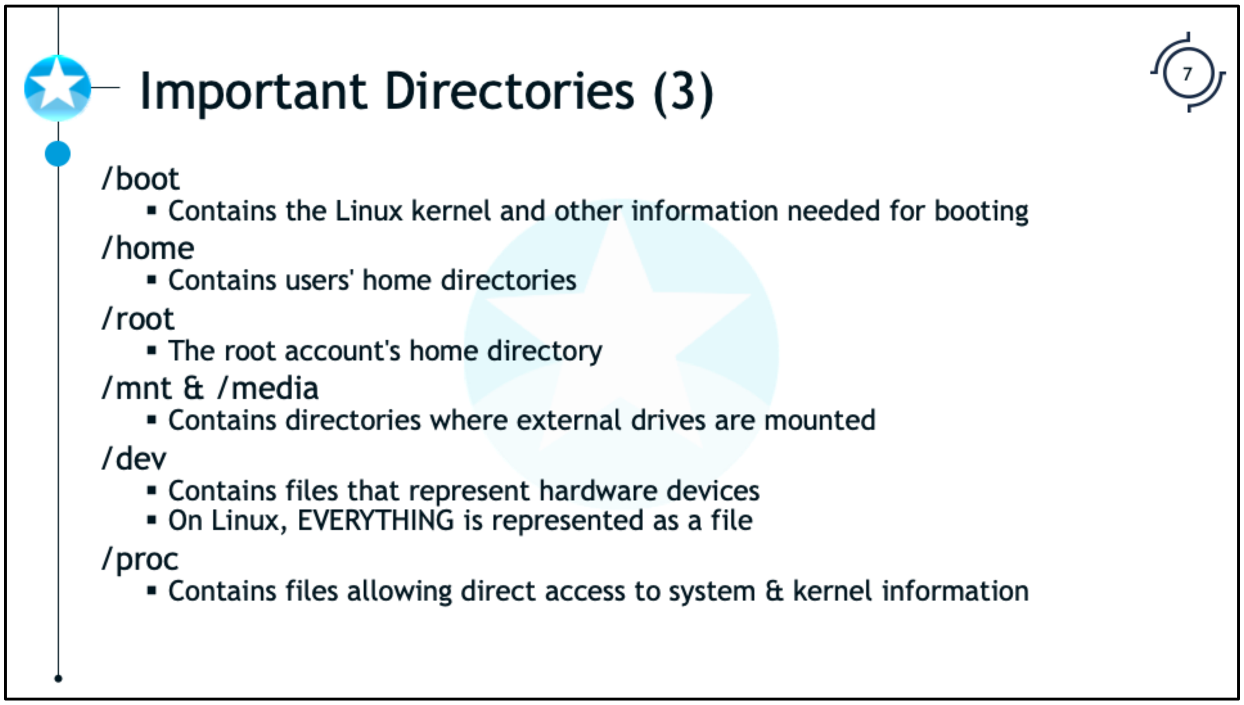
**/etc**: Contains configuration files for the system and for other software installed on the system. For example, /etc/resolv.conf contains the DNS servers the system will use to resolve names, and /etc/hosts contains local name to IP address mappings.

**/usr**: Contains files for programs installed on the system. This has a set of subdirectories that matches the rest of the system, such as bin, sbin, and lib.

**/usr/local**: The default location for software installed from source. Like /usr, this also contains a set of subdirectories including bin, sbin, and lib.

**/var**: Contains variable data used by programs, such as logs. The default web root is often /var/www/.

**/tmp** & **/var/tmp**: These contain temporary files used by programs or the system. /tmp is usually cleaned out automatically either during reboot or on a regular basis, while /var/tmp usually isn't.


## Important Directories (3)

/boot
- Contains the Linux kernel and other information needed for booting

/home
- Contains users' home directories

/root
- The root account's home directory

/mnt & /media
- Contains directories where external drives are mounted

/dev
- Contains files that represent hardware devices
- On Linux, EVERYTHING is represented as a file

/proc
- Contains files allowing direct access to system & kernel information

**/boot**: Contains the Linux kernel (the core of the operating system) and other information needed for booting.

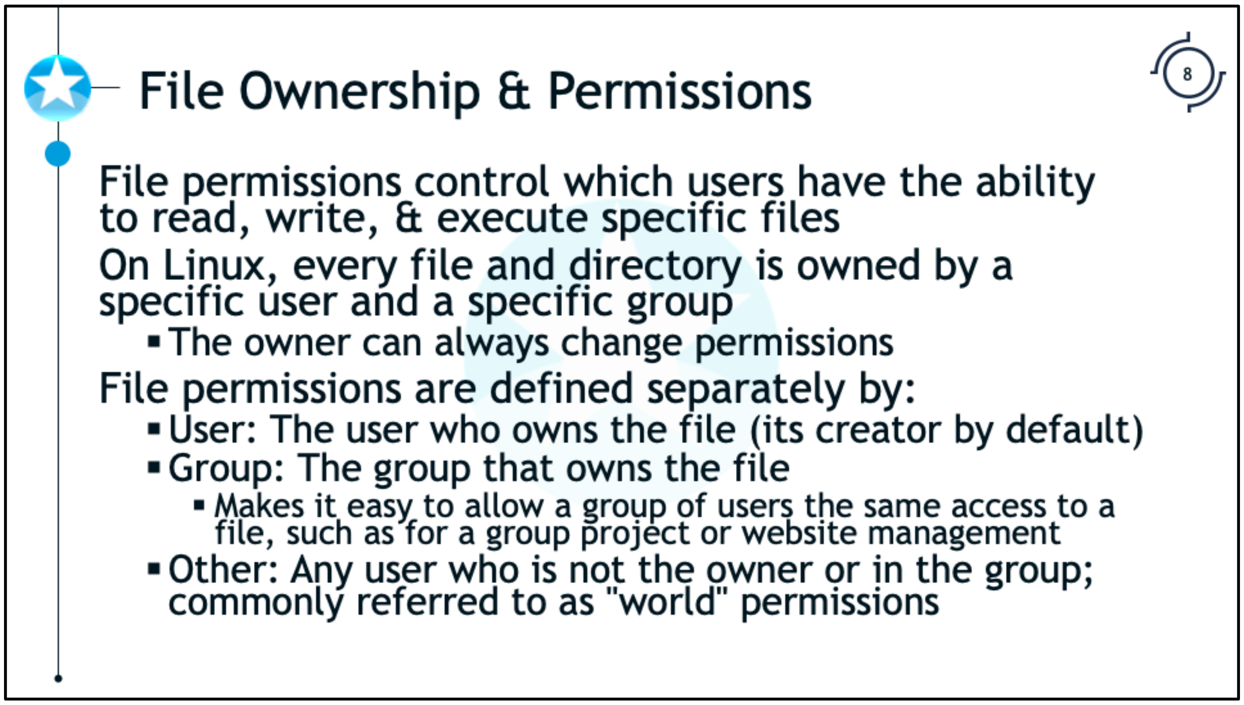
**/home**: Contains users' home directories (containing their personal files, settings, etc.).

**/root**: The root account's home directory.

**/mnt** & **/media**: These contain directories where external & network drives are mounted. /mnt generally contains generic mount points like /mnt/floppy and /mnt/cdrom, while /media is usually handled by a daemon that creates new directories on the fly with the name of the device (such as /media/SANS_Network_Tools).

**/dev**: Contains files that represent hardware devices (on Linux, ALL devices are represented by a file).

**/proc**: Contains virtual files allowing direct access to system & kernel information. For example, there are numbered subdirectories that correspond to each running process on the system. The numbers correspond to the process ID.



## File Ownership & Permissions

File permissions control which users have the ability to read, write, & execute specific files

On Linux, every file and directory is owned by a specific user and a specific group

- The owner can always change permissions

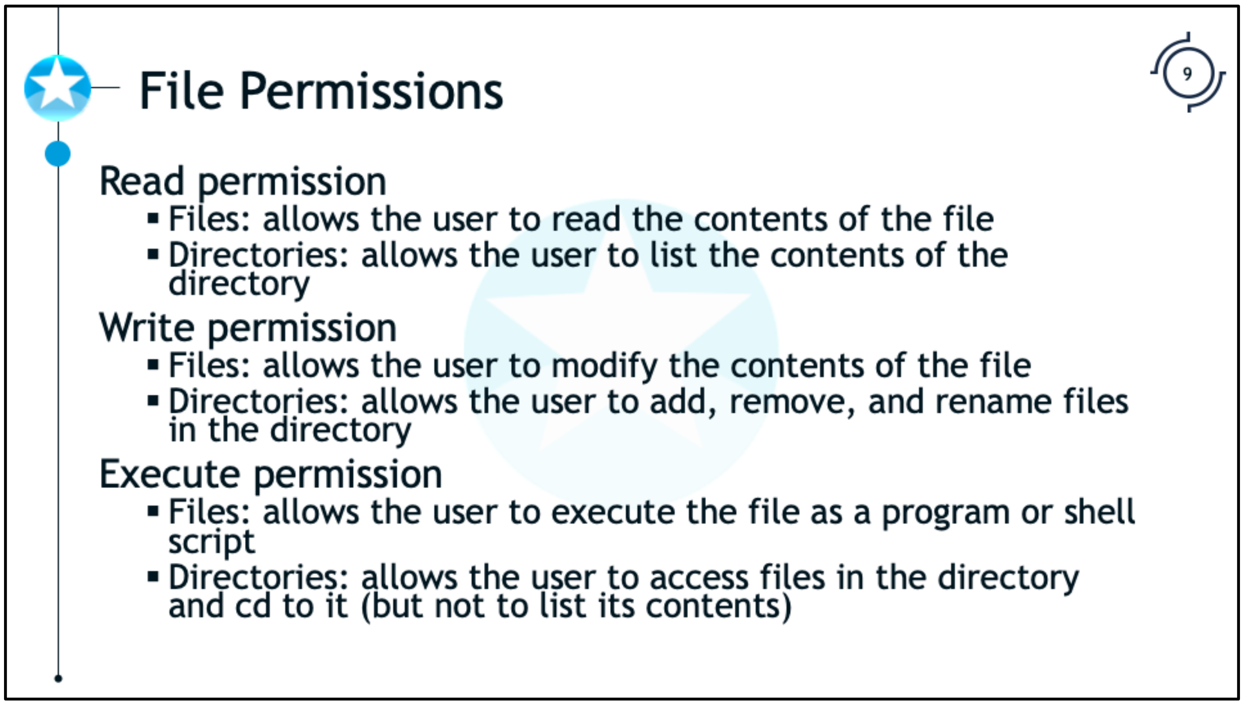
File permissions are defined separately by:

- User: The user who owns the file (its creator by default)
- Group: The group that owns the file
  - Makes it easy to allow a group of users the same access to a file, such as for a group project or website management
- Other: Any user who is not the owner or in the group; commonly referred to as "world" permissions

File permissions are used to control who can read, write and/or execute files on the hard drive. Ownership is used to specify who is considered the owner of a file. Permissions and ownership work together to control file system privileges.

On Linux, every file and directory is owned by a specific user and a specific group. The owner is always able to change the permissions, even if the current permissions don't allow the owner to access the file.  This allows file permissions to be defined separately by the user who owns the file, the group that owns the file, and all other users on the system (commonly referred to as "world").

Users can change the group that owns any file they own (using "chgrp"), but only root can change the user that owns a file (using "chown").



## File Permissions

Read permission
- Files: allows the user to read the contents of the file
- Directories: allows the user to list the contents of the directory

Write permission
- Files: allows the user to modify the contents of the file
- Directories: allows the user to add, remove, and rename files in the directory

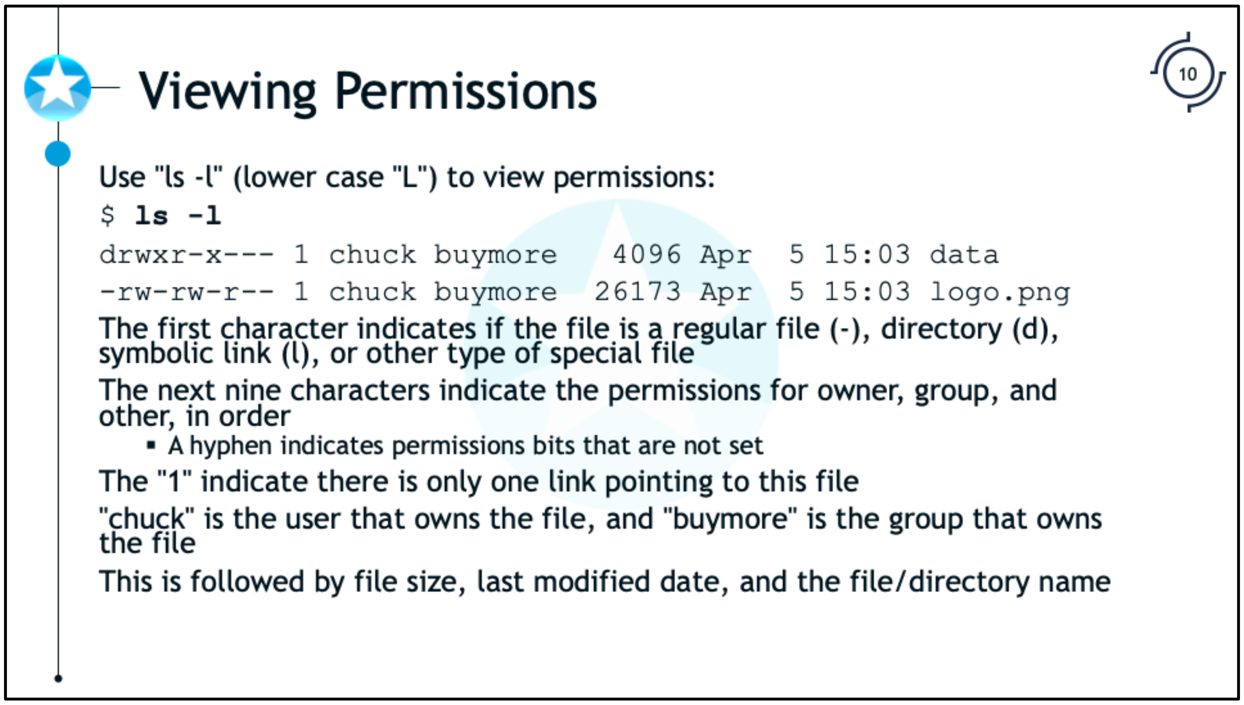
Execute permission
- Files: allows the user to execute the file as a program or shell script
- Directories: allows the user to access files in the directory and cd to it (but not to list its contents)

Linux has three types of permissions on a file: read, write, and execute.

Read permission allows the contents of files to be read, and the contents of directories to be listed.

Write permission allows the contents of a file to be modified, and allows users to add, remove, and rename files in a directory (note that even with write permission on a file, you cannot delete it without write permission on the directory!).

Execute permissions allows a file to be executed as a program or shell script, and allows users to enter and access files in a directory (but not list its contents).



## Viewing Permissions

Use "ls -l" (lower case "L") to view permissions:

```
$ ls -l
drwxr-x--- 1 chuck buymore   4096 Apr  5 15:03 data
-rw-rw-r-- 1 chuck buymore  26173 Apr  5 15:03 logo.png
```

The first character indicates if the file is a regular file (-), directory (d), symbolic link (l), or other type of special file

The next nine characters indicate the permissions for owner, group, and other, in order

- A hyphen indicates permissions bits that are not set

The "1" indicate there is only one link pointing to this file

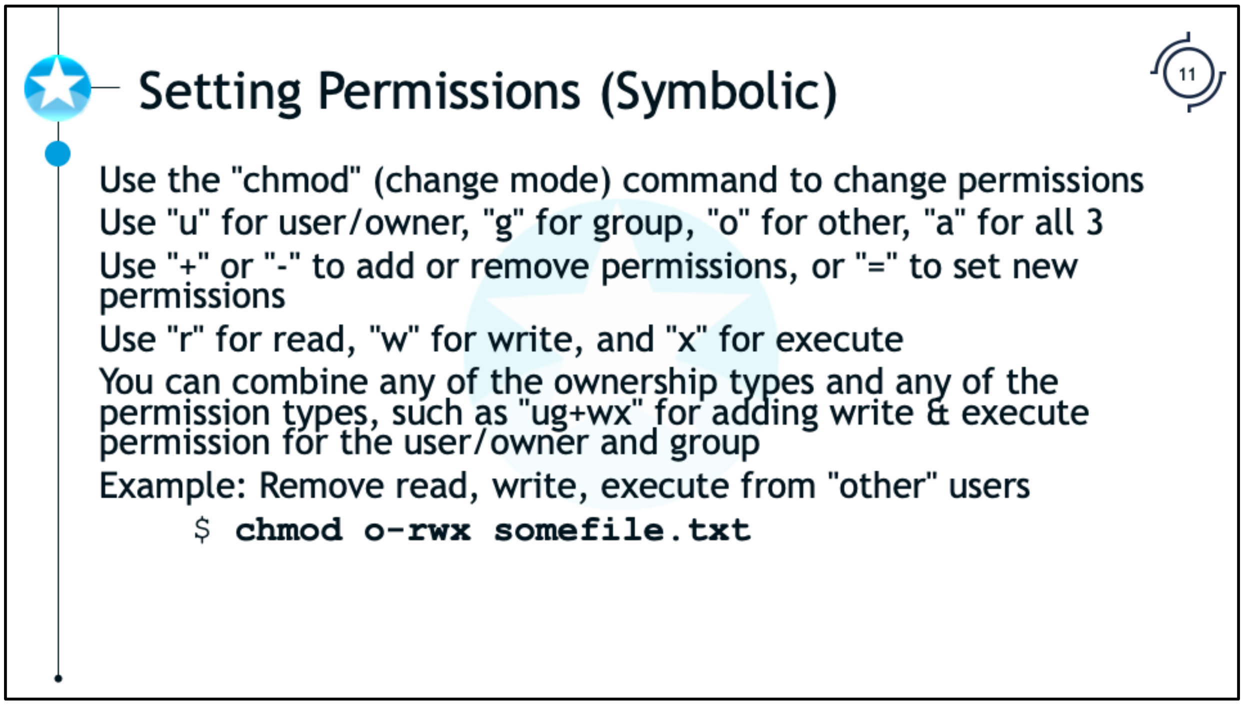
"chuck" is the user that owns the file, and "buymore" is the group that owns the file

This is followed by file size, last modified date, and the file/directory name

You can view permissions on files and directories using the long listing "-l" (lowercase L) option to "ls". This will display a listing of the current or specified directory, one item per line.

The first character indicates if the file is a regular file (-), directory (d), symbolic link (lower-case L), or other type of special file. The next nine characters indicate the permissions for the file, in a particular order (rwx). The first 3 indicate read (r), write (w), and execute (x) for the user/owner, the next three for the group, and the last three for all other users. A hyphen indicates that the given permission is not set on that file or directory for the particular permissions group. The "1" (numeral one) after the permission bits indicates that there is only one hard link to the file on the filesystem. "chuck" is the user that owns the file, and "buymore" is the group that owns the file. This is followed by the file size (usually 4096 for directories), the last modified date, and the file name.

In the example above, "data" is a directory owned by the user "chuck", and the group "buymore". The user "chuck" has read, write, and execute permissions, the group "buymore" has read and execute permissions, and all other users have no access at all. On the file "logo.png", "chuck" has read and write permissions (but not execute), the "buymore" group has read and write permissions, and all other users have read permission only.



## Setting Permissions (Symbolic)

Use the "chmod" (change mode) command to change permissions
Use "u" for user/owner, "g" for group, "o" for other, "a" for all 3
Use "+" or "-" to add or remove permissions, or "=" to set new permissions
Use "r" for read, "w" for write, and "x" for execute
You can combine any of the ownership types and any of the permission types, such as "ug+wx" for adding write & execute permission for the user/owner and group
Example: Remove read, write, execute from "other" users

```
    $ chmod o-rwx somefile.txt
```

On Linux, the "chmod" command (short for "change mode") is used to change permissions on files and directories. There are two ways to specify the desired permissions: symbolic mode and octal/numeric mode. Symbolic mode is easiest when making changes to existing permissions (rather than setting all permission bits at once). Symbolic mode allows you to change permissions for one permission group (user, group, or other) at a time.

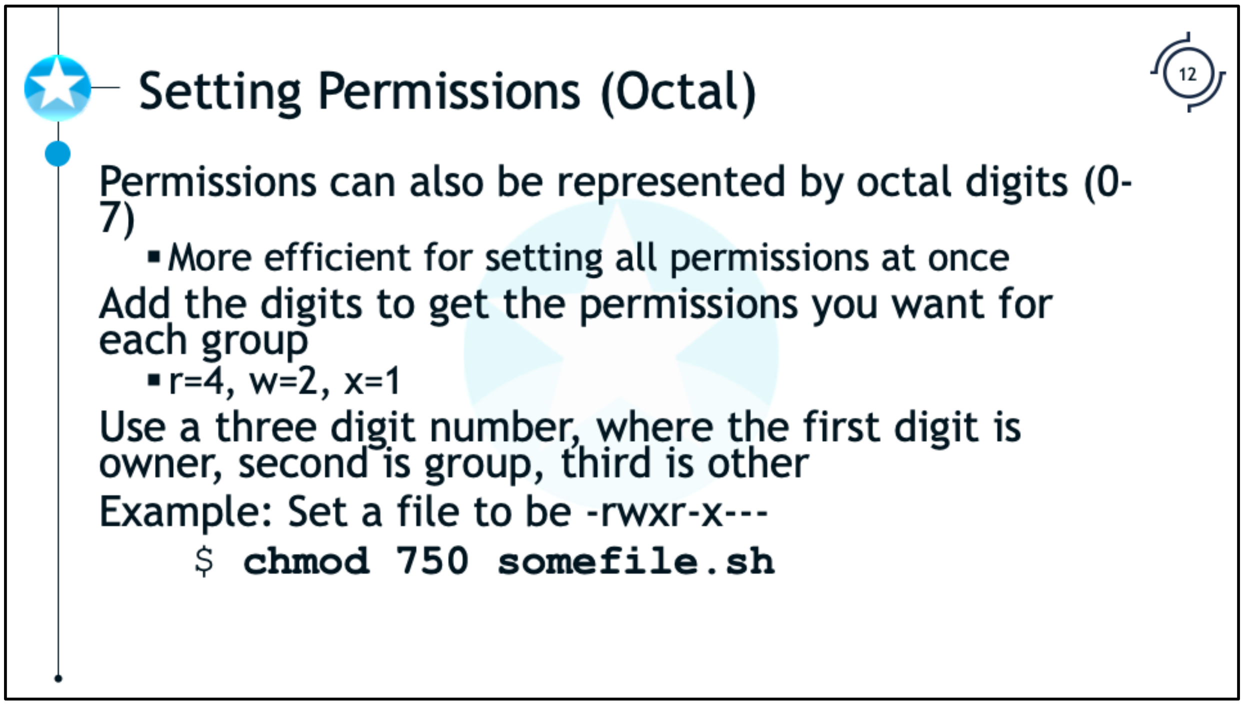
To set permissions in symbolic mode, first specify "u" for user/owner, "g" for group, "o" for other, or "a" for all three. Then, use a "+" to add permissions, a "-" to remove permissions, or "=" to assign new permissions. Then, use "r" for read, "w" for write, and "x" for execute. You can combine any of the ownership types and any of the permissions types in a single command, such as "ug+wx" for adding write & execute permission for the user/owner and group.

For example, to remove read, write, and execute permission from "other" users, run:

```
$ chmod o-rwx somefile.txt
```

To set a file to be read only (write and execute will be remove) for all users:

```
# chmod ugo=r somefile.txt
```



## Setting Permissions (Octal)

Permissions can also be represented by octal digits (0-7)

- More efficient for setting all permissions at once

Add the digits to get the permissions you want for each group

- r=4, w=2, x=1

Use a three digit number, where the first digit is owner, second is group, third is other

Example: Set a file to be -rwxr-x---

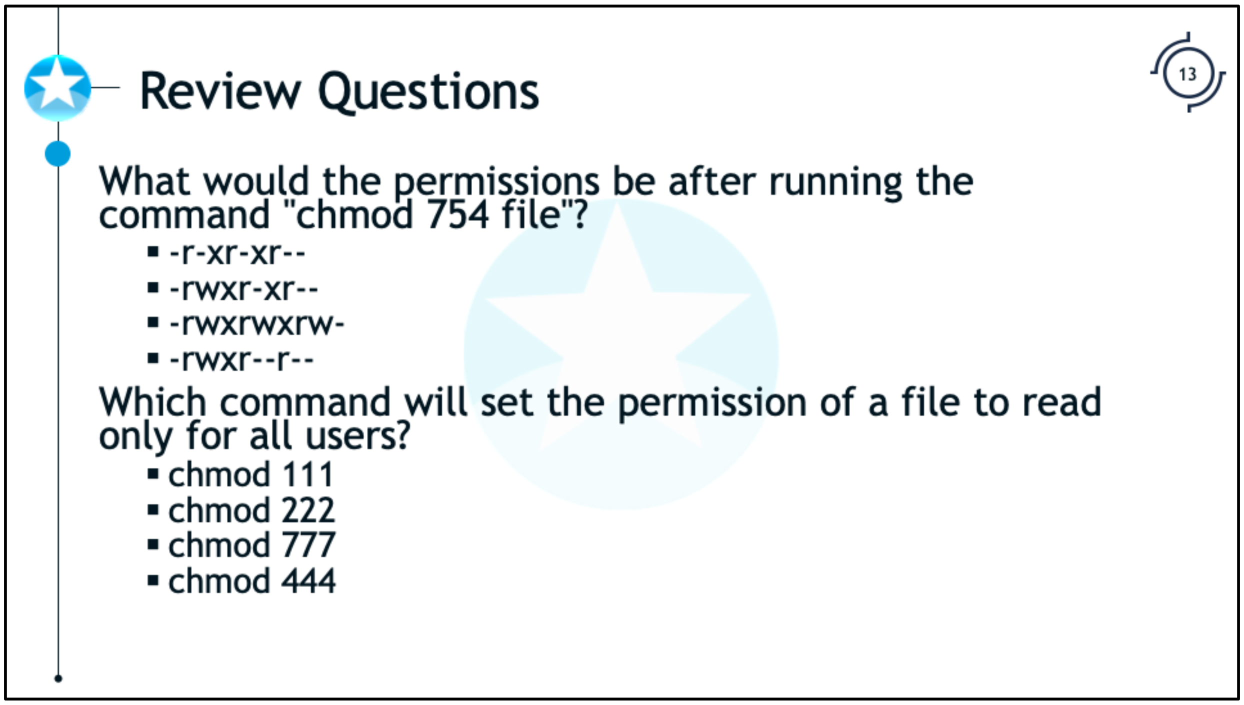
```
$ chmod 750 somefile.sh
```

Linux permissions can also be represented by three octal digits (0-7), one for each permission group.  This is a more efficient way to set all permissions bits at once.  To use octal, add the permission bits together for each group, and then put them in order as a three digit number.  Read permission is "4", write permission is "2", and execute permission is "1".  All combinations can be expressed as a combination of these digits:

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

For example, to set a file to have read, write, and execute permissions for the user/owner, read and execute permissions for the group, and no access for all other users (equivalent to symbolic -rwxr-x---), use the following command:

```
$ chmod 750 somefile.sh
```



## Review Questions

What would the permissions be after running the command "chmod 754 file"?

- -r-xr-xr--
- -rwxr-xr--
- -rwxrwxrw-
- -rwxr--r--

Which command will set the permission of a file to read only for all users?

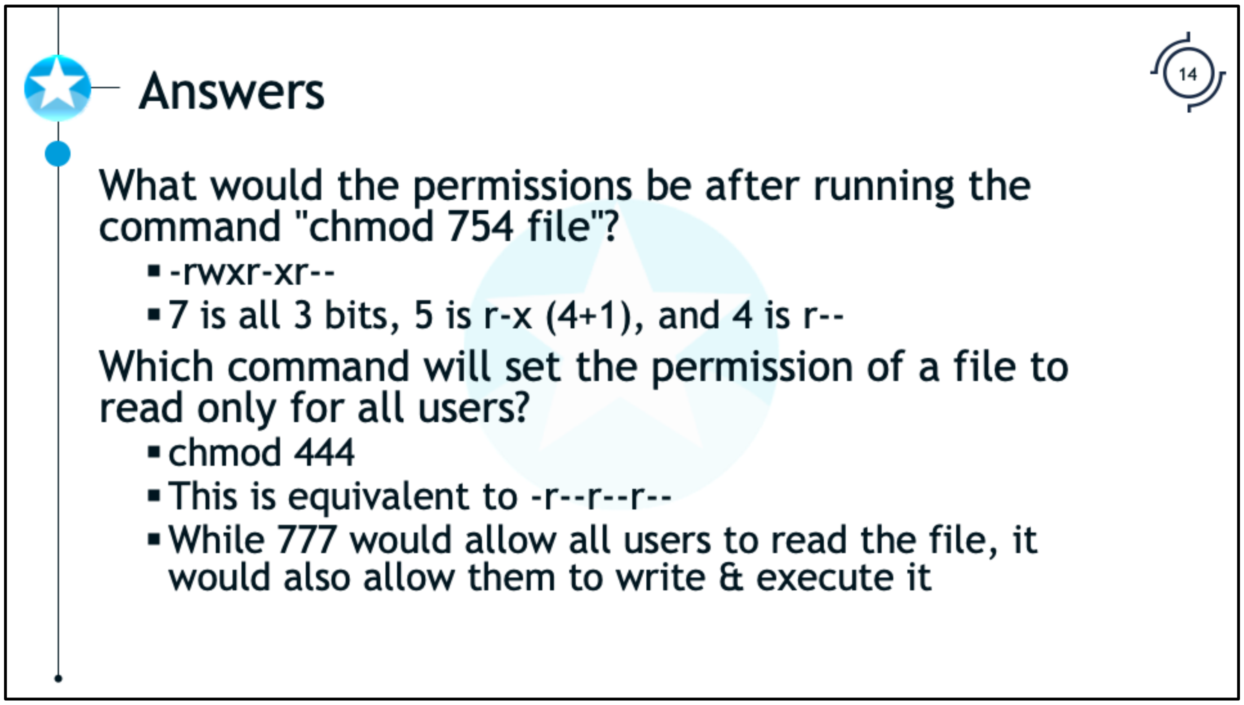
- chmod 111
- chmod 222
- chmod 777
- chmod 444

What would the permissions be after running the command "chmod 754 file"?

-r-xr-xr--
-rwxr-xr--
-rwxrwxrw-
-rwxr--r--

Which command will set the permission of a file to read only for all users?

chmod 111
chmod 222
chmod 777
chmod 444



## Answers

What would the permissions be after running the command "chmod 754 file"?

- -rwxr-xr--
- 7 is all 3 bits, 5 is r-x (4+1), and 4 is r--

Which command will set the permission of a file to read only for all users?

- chmod 444
- This is equivalent to -r--r--r--
- While 777 would allow all users to read the file, it would also allow them to write & execute it

What would the permissions be after running the command "chmod 754 file"?

-rwxr-xr--

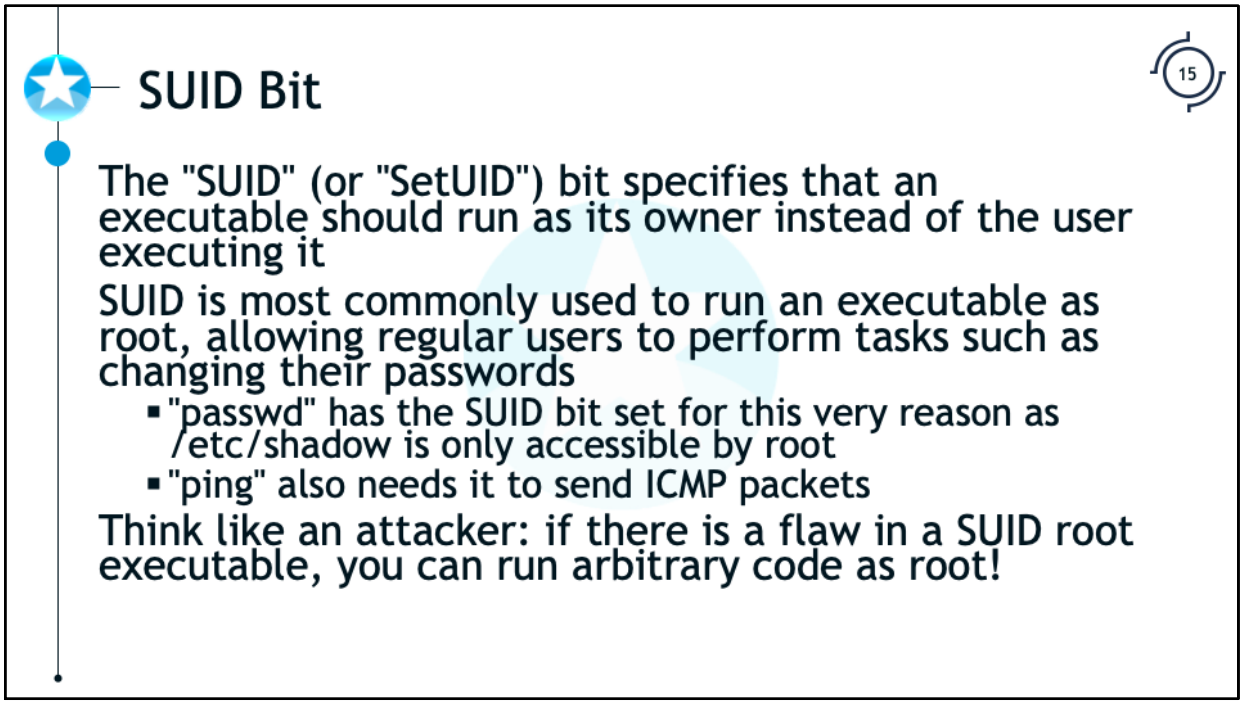
7 is all 3 bits, 5 is r-x (4+1), and 4 is r--

Which command will set the permission of a file to read only for all users?

chmod 444

This is equivalent to -r--r--r--

While 777 would allow all users to read the file, it would also allow them to write & execute it



## SUID Bit

The "SUID" (or "SetUID") bit specifies that an executable should run as its owner instead of the user executing it

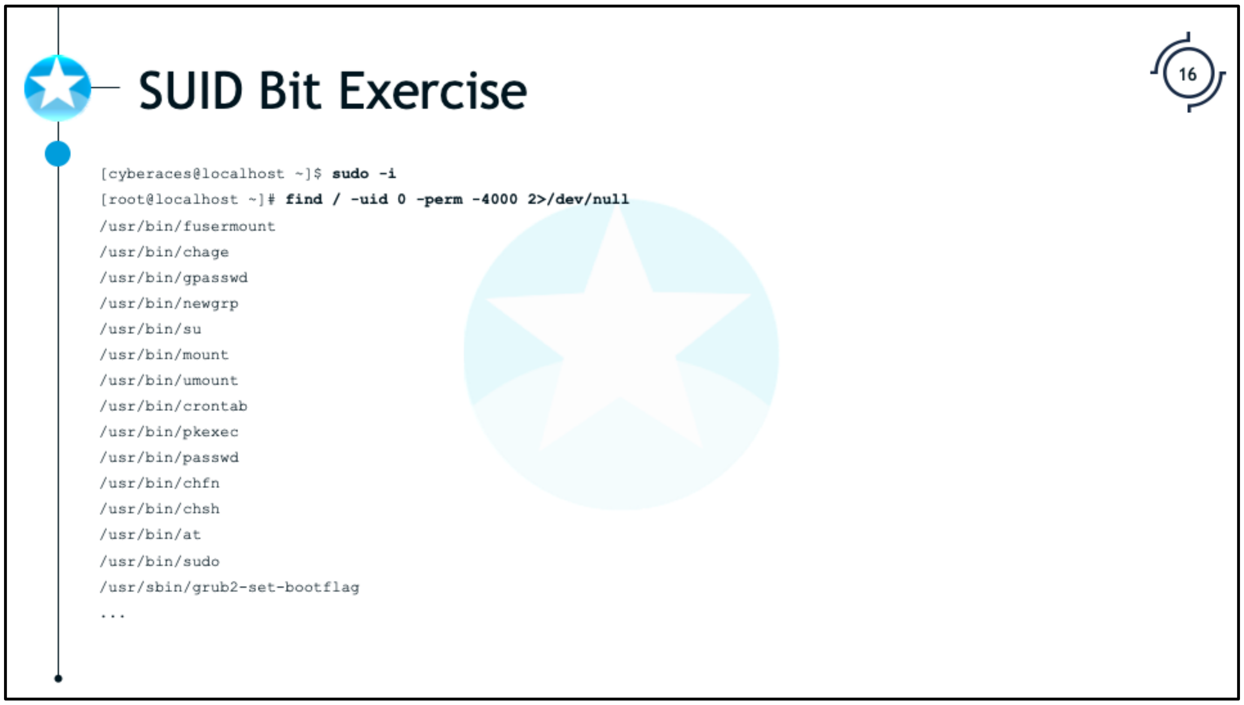
SUID is most commonly used to run an executable as root, allowing regular users to perform tasks such as changing their passwords

- "passwd" has the SUID bit set for this very reason as /etc/shadow is only accessible by root
- "ping" also needs it to send ICMP packets

Think like an attacker: if there is a flaw in a SUID root executable, you can run arbitrary code as root!

Some programs need to execute with the privileges of certain users. For example, the "passwd" program used to change your password has to be able to write your hashed password to the /etc/shadow file (which can only be written to by root). Therefore, the "passwd" program has the SUID bit set so that it will run as root, regardless of what account is actually executing it and the privileges they have.

Put your attacker hat on and think about how this could be abused. If a program is SUID root and it contains a flaw that can be exploited, then we could get full root access to the box! Because of this, programs that are SUID root have to very carefully written and audited to make sure they are free of any vulnerabilities.



## SUID Bit Exercise

```
[cyberaces@localhost ~]$ sudo -i
[root@localhost ~]# find / -uid 0 -perm -4000 2>/dev/null
/usr/bin/fusermount
/usr/bin/chage
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/su
/usr/bin/mount
/usr/bin/umount
/usr/bin/crontab
/usr/bin/pkexec
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/at
/usr/bin/sudo
/usr/sbin/grub2-set-bootflag
...
```

An attacker who has gained root access on your box may create a backdoor on your system that will enable him or her to easily recapture root access on your box. The simplest form of this backdoor is a shell with the SUID bit set. Search your computer to find all of the programs that are set to run with root privileges.

1. In your CentOS VM, open a terminal window and become root.

   ```
   $ sudo -i
   ```

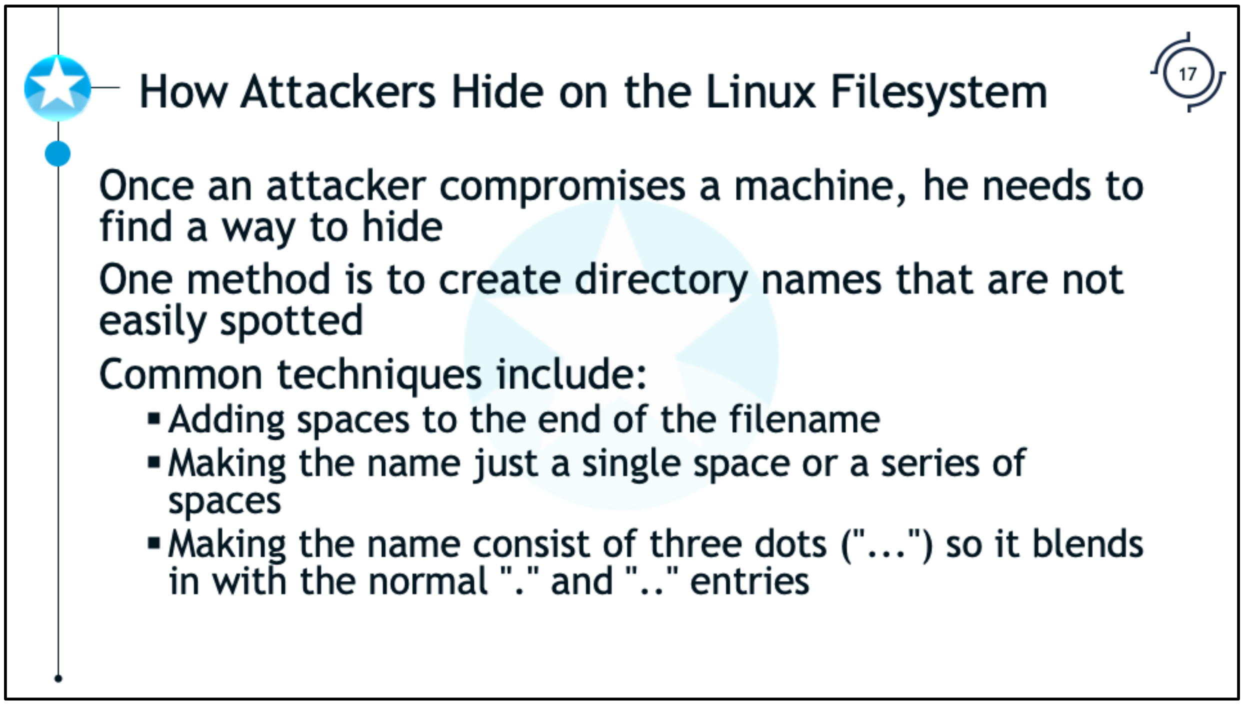
2. Search for all files that are SUID root using the following command from the SANS Linux Intrusion Discovery Cheat Sheet:

   ```
   # find / -uid 0 -perm -4000 2>/dev/null
   ```

   This command will search starting at the root of the file system (/), look for files owned by root (UID 0), and with the special SUID bit set (4000). We will throw away any error messages by sending STDERR (file descriptor 2) to /dev/null.

3. After a few seconds, the find command will return the list of files that are SUID root. Look through the list of files and think about why they must be SUID root. You should see familiar commands like "passwd", "su", "sudo", "ping", and other commands which might not be familiar. Use man to learn more about these unfamiliar commands and why they would need to be SUID root. For example, "man chfn".

4. Close the terminal window.



## How Attackers Hide on the Linux Filesystem

Once an attacker compromises a machine, he needs to find a way to hide

One method is to create directory names that are not easily spotted

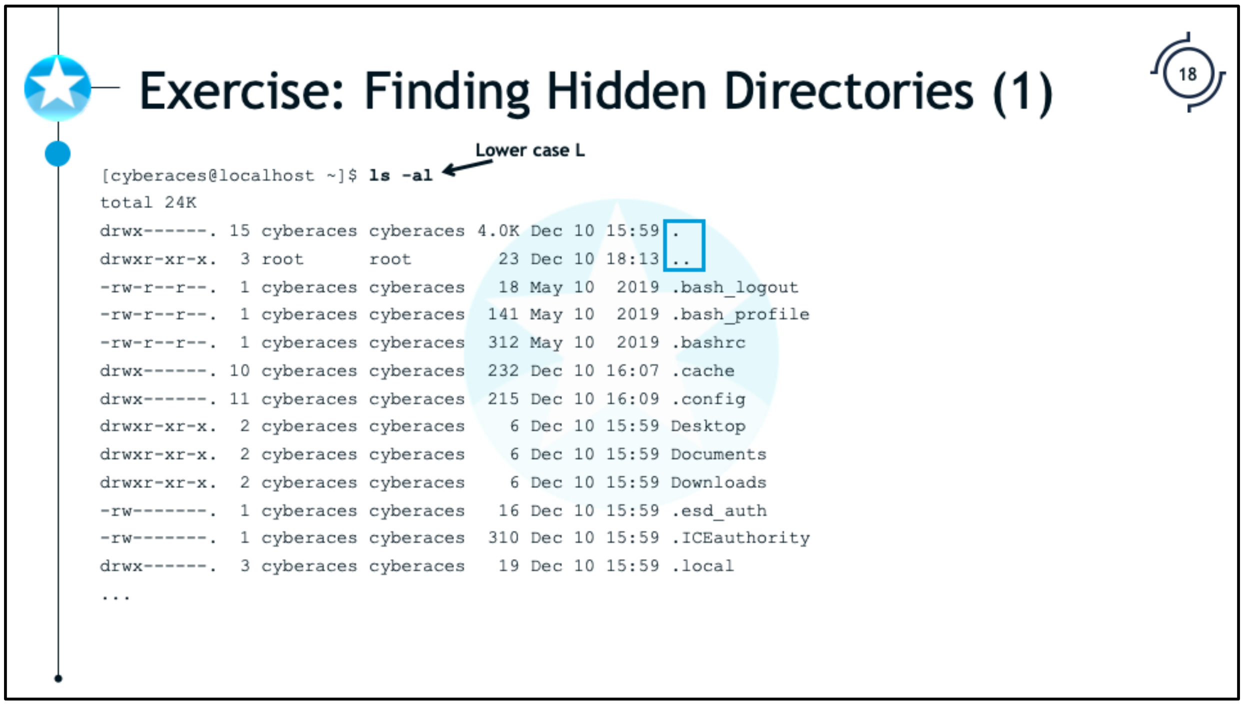
Common techniques include:

- Adding spaces to the end of the filename
- Making the name just a single space or a series of spaces
- Making the name consist of three dots ("...") so it blends in with the normal "." and ".." entries

Once an attacker has compromised a machine, one of his goals is to not be discovered. In order to do this, he needs to find ways to hide in the file system. One method of hiding is to create directory names that are not easily spotted. Common techniques used by attackers include:

- Adding spaces to the end of the filename
- Making the name just a single space or a series of spaces
- Making the name consist of three dots, so it blends in with the normal single-dot and double-dot directory entries

As a defender we need to aware of these techniques so we can find these hidden files and directories.



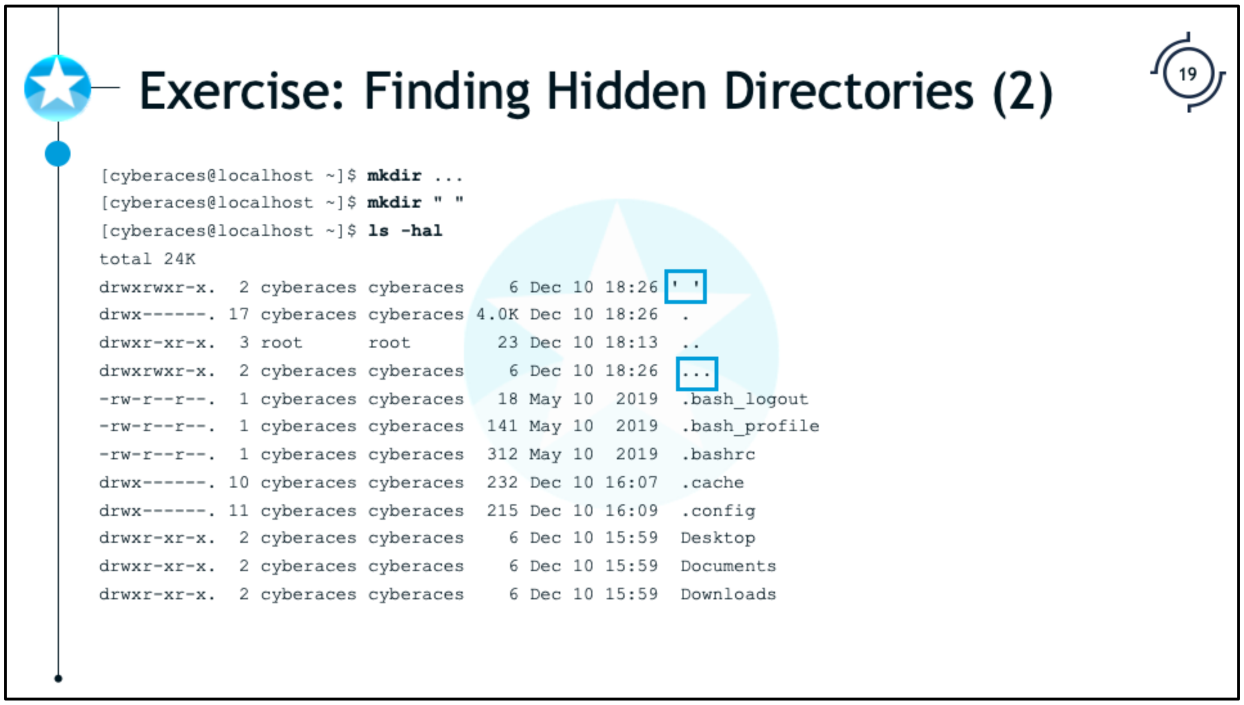
## Exercise: Finding Hidden Directories (1)

```
[cyberaces@localhost ~]$ ls -al
total 24K
drwx------. 15 cyberaces cyberaces 4.0K Dec 10 15:59 .
drwxr-xr-x.  3 root      root        23 Dec 10 18:13 ..
-rw-r--r--.  1 cyberaces cyberaces   18 May 10  2019 .bash_logout
-rw-r--r--.  1 cyberaces cyberaces  141 May 10  2019 .bash_profile
-rw-r--r--.  1 cyberaces cyberaces  312 May 10  2019 .bashrc
drwx------. 10 cyberaces cyberaces  232 Dec 10 16:07 .cache
drwx------. 11 cyberaces cyberaces  215 Dec 10 16:09 .config
drwxr-xr-x.  2 cyberaces cyberaces    6 Dec 10 15:59 Desktop
drwxr-xr-x.  2 cyberaces cyberaces    6 Dec 10 15:59 Documents
drwxr-xr-x.  2 cyberaces cyberaces    6 Dec 10 15:59 Downloads
-rw-------.  1 cyberaces cyberaces   16 Dec 10 15:59 .esd_auth
-rw-------.  1 cyberaces cyberaces  310 Dec 10 15:59 .ICEauthority
drwx------.  3 cyberaces cyberaces   19 Dec 10 15:59 .local
...
```

1. In your CentOS VM, open a terminal window. We'll do the first part of this exercise as the "centos" user, so do not become root yet. Type the following command to show all the files in the current directory (/home/liveuser/):

   ```
   $ ls -al
   ```

   Note: That is a lower case "L" at the end of the command
2. Notice that the first two entries are "." and "..". The single dot represents the current directory (/home/cyberaces/). The double dot represents the parent directory (/home/). These entries are shown in every single directory in the file system, so a sysadmin would normally see them all the time.



## Exercise: Finding Hidden Directories (2)

```
[cyberaces@localhost ~]$ mkdir ...
[cyberaces@localhost ~]$ mkdir " "
[cyberaces@localhost ~]$ ls -hal
total 24K
drwxrwxr-x.  2 cyberaces cyberaces    6 Dec 10 18:26 ' '
drwx------. 17 cyberaces cyberaces 4.0K Dec 10 18:26  .
drwxr-xr-x.  3 root      root        23 Dec 10 18:13  ..
drwxrwxr-x.  2 cyberaces cyberaces    6 Dec 10 18:26  ...
-rw-r--r--.  1 cyberaces cyberaces   18 May 10  2019  .bash_logout
-rw-r--r--.  1 cyberaces cyberaces  141 May 10  2019  .bash_profile
-rw-r--r--.  1 cyberaces cyberaces  312 May 10  2019  .bashrc
drwx------. 10 cyberaces cyberaces  232 Dec 10 16:07  .cache
drwx------. 11 cyberaces cyberaces  215 Dec 10 16:09  .config
drwxr-xr-x.  2 cyberaces cyberaces    6 Dec 10 15:59  Desktop
drwxr-xr-x.  2 cyberaces cyberaces    6 Dec 10 15:59  Documents
drwxr-xr-x.  2 cyberaces cyberaces    6 Dec 10 15:59  Downloads
```

3. Create a new directory called "..." with the following command:

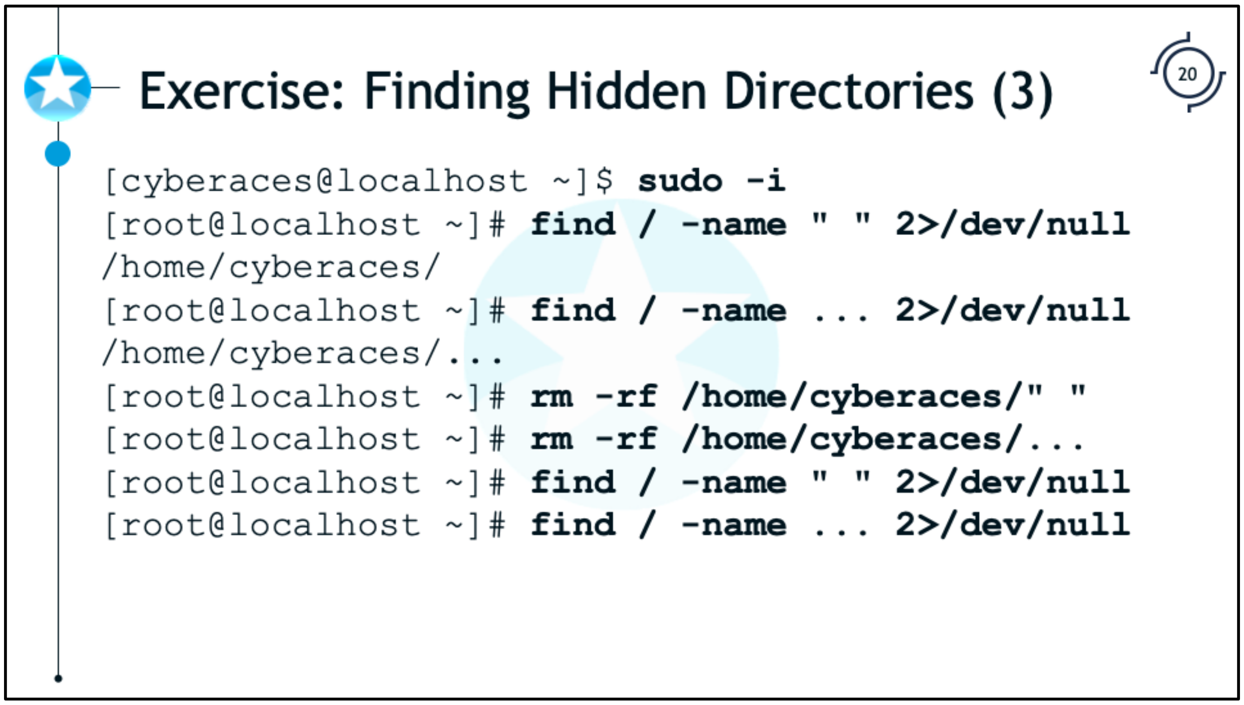
   $ **mkdir ...**

4. Create a new directory called " " (a single space) with the following command:

   $ **mkdir " "**

5. Press the Up arrow three times to retrieve the "ls -hal" command and press Enter.

6. Notice in the output how the new " " and "..." directories blend in with the normal "." and ".." entries. Think about how easy it would be for a sysadmin to overlook these stealthy directories. How would a sysadmin find them?



## Exercise: Finding Hidden Directories (3)

```
[cyberaces@localhost ~]$ sudo -i
[root@localhost ~]# find / -name " " 2>/dev/null
/home/cyberaces/
[root@localhost ~]# find / -name ... 2>/dev/null
/home/cyberaces/...
[root@localhost ~]# rm -rf /home/cyberaces/" "
[root@localhost ~]# rm -rf /home/cyberaces/...
[root@localhost ~]# find / -name " " 2>/dev/null
[root@localhost ~]# find / -name ... 2>/dev/null
```

8. Become root

   ```
   $ sudo -i
   ```

9. Use this command to look for filenames of a single space:

   ```
   # find / -name " "
   ```

10. After a few seconds, the find command should display the location of the stealthy directory (/home/cyberaces/ ). Note that there is a space after the final slash, you just can't tell that it's there.

11. Edit the previous command to look for the "..." directory. Press the Up arrow to retrieve the previous command, then press the backspace key three times and then type the following (then press enter):

   ```
   ...
   ```

12. Again, the find command should display the location of the stealthy directory (/home/cyberaces/...).

13. Remove the stealthy directories with the following commands:

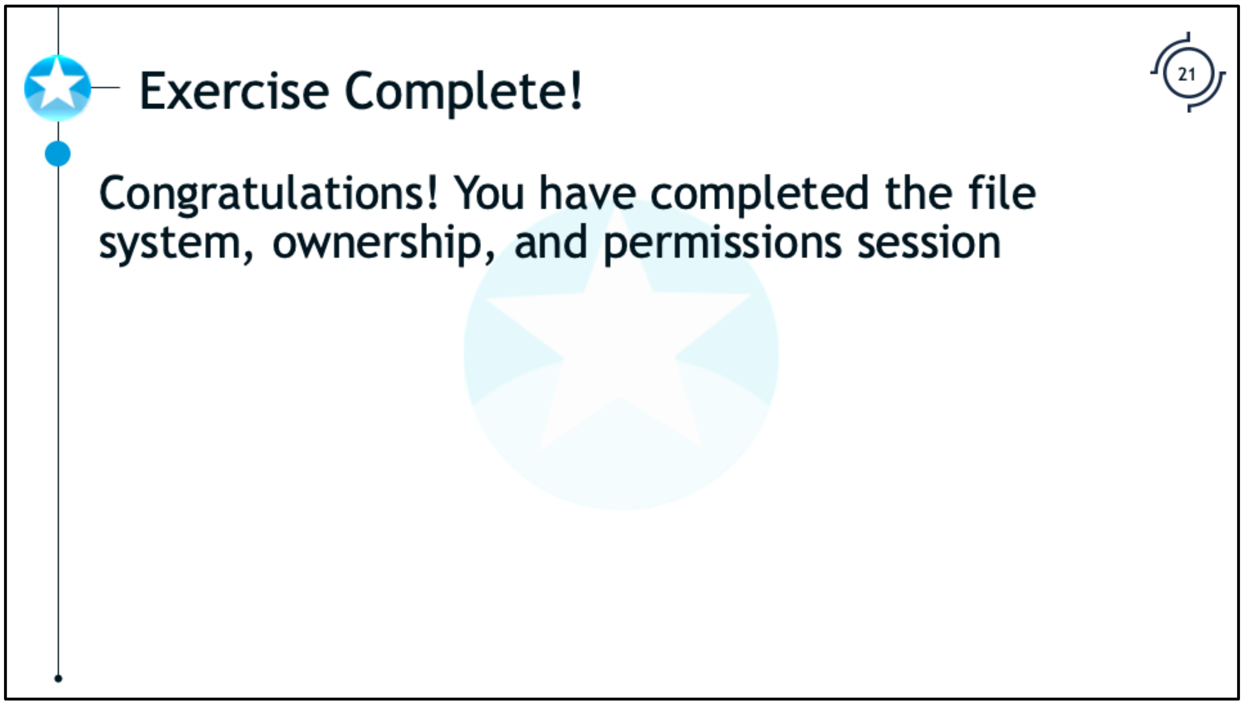
   ```
   # rm -rf /home/cyberaces/" "
   ```

   ```
   # rm -rf /home/cyberaces/...
   ```

   The "rm -rf" command recursively removes an entire directory structure even if the directories are not empty. Always be very careful when typing an "rm -rf" command, especially if you have root privileges! On a production system, one little typo could result in irreversible damage to the system.

14. Re-run the find commands above to verify that the stealthy directories have been

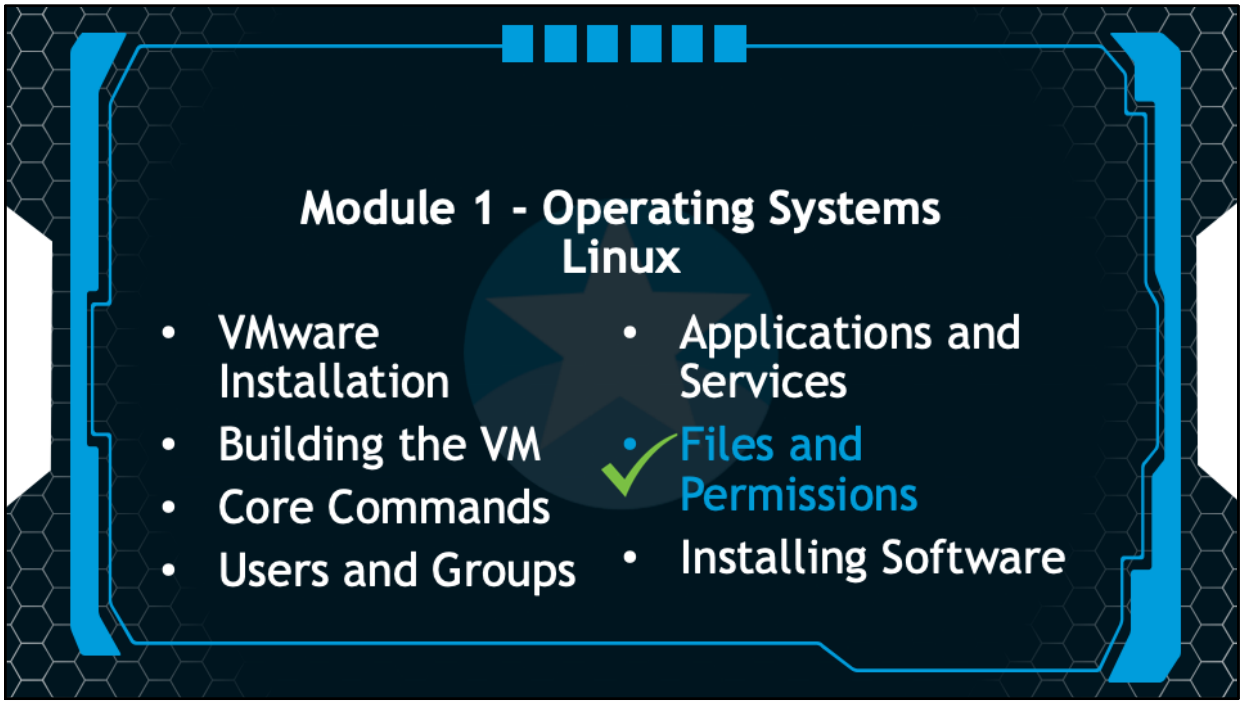
removed.



## Exercise Complete!

Congratulations! You have completed the file system, ownership, and permissions session

Congratulations! You have completed the file system, ownership, and permissions session.



## Module 1 - Operating Systems Linux

- VMware Installation
- Building the VM
- Core Commands
- Users and Groups
- Applications and Services
- ✓ Files and Permissions
- Installing Software

You have completed the session on the Linux file system and permissions. In the next session, we will discuss ways to install software in Linux.