

Federal University Dutse



CCS 206 - Introduction to Linux for Security and Forensics

Lecture Note 3
By
Muhammad S. Ali



The Shell

- When we speak of the command line, we are really referring to the shell.
- The shell is a program that takes keyboard commands and passes them to the operating system to carry out a task
- Almost all Linux distributions supply a shell program from the GNU Project called bash.
- The name bash is an acronym for Bourne Again Shell, a reference to the fact that bash is an enhanced replacement for sh, the original Unix shell program written by Steve Bourne.



The Terminal Window

- There are many ways to launch an interactive terminal depending on the Linux distro you use.
- You can press Ctrl + Alt + T to open a terminal. If that does not work, you can search for a terminal in your Linux Box.
- In your interactive terminal, there should be a dollar sign, \$ and a blinking cursor.
- The symbols at the end of the prompt may be a \$ symbol or a # symbol.
- \$ or a dollar sign – means you are logged in as a regular user
- # or a hashtag/pound symbol – means you are logged



in as a user with elevated privileges.

- The user that is noted in the # environment is also known as a root user, super user or administrator.
- Open a terminal and type the following command:

```
$ pwd
```

- Try typing gibberish at the shell and observe the output.
- Since the command make no sense, the shell tell us so and gives us another change to enter again.



Simple commands

- Now that we have learned to type, let's try some few more commands.
- The first is date which displays the current date and time:

\$ date

- A related command is cal which displays the calendar of the current month:

\$ cal



- To see the current amount of free space on your disk drives, enter *df*:

\$ df

- To display the amount of free memory enter *free*

\$ free

- To end the terminal session type *exit*:

\$ exit



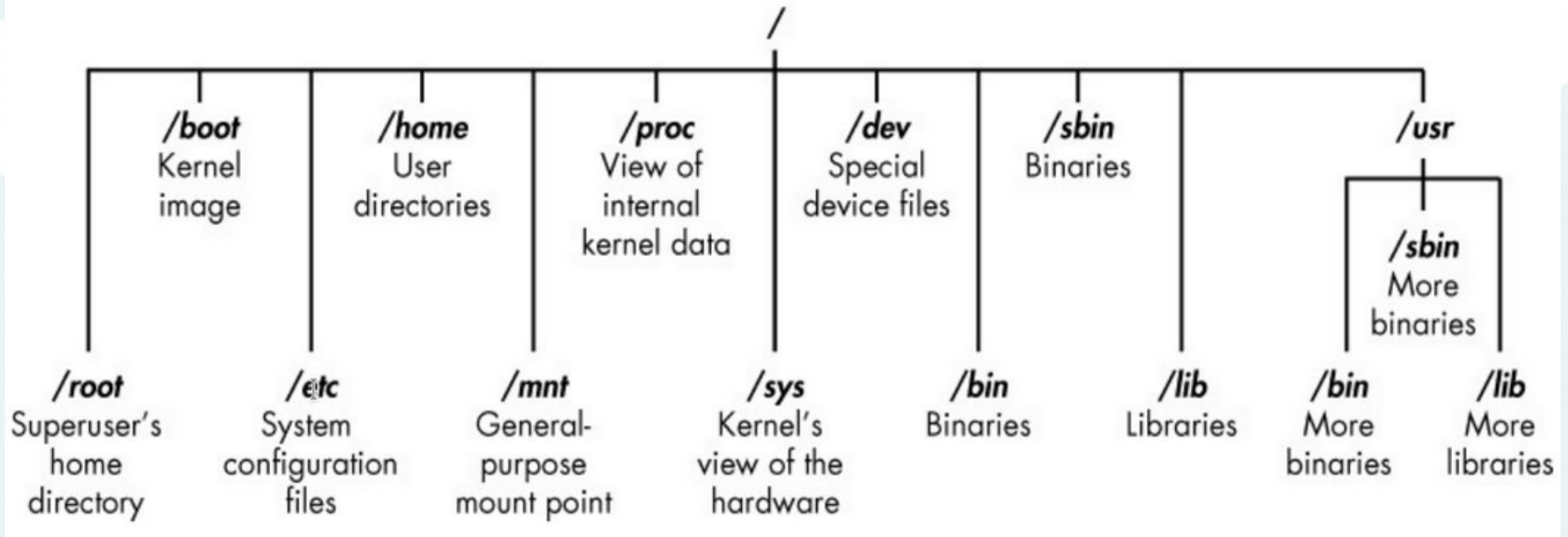
Understanding the Filesystem

- Linux organizes its files in what is called a hierarchical directory structure, it use a Filesystem Hierarchy Standard (FHS) which defines the content and directory of all Linux distributions
- This means that they are organized in a tree-like pattern of directories which may contain files and other directories
- The first directory in the filesystem is called the **root** directory
- The root directory contains files and sub-directories, which contain more files and sub-directories, and so on
- Linux always have a single filesystem tree, regardless of how many drives or storage devices are attached to the computer



- Storage devices are attached (or more correctly, mounted) at various points on the tree according to the whims of the system administrator, the person (or persons) responsible for the maintenance of the system.
- The Linux filesystem structure is somewhat different from that of Windows.
- Linux doesn't have a physical drive (such as the C: drive) at the base of the filesystem but uses a logical filesystem instead. At the very top of the filesystem structure is */*, which is often referred to as the root of the filesystem, as if it were an upside-down tree (see Figure below). Keep in mind that this is different from the root user. These terms may seem confusing at first, but they will become easier to differentiate once you get used to Linux.





The Linux Filesystem



- The root (**/**) of the filesystem is at the top of the tree, and the following are the most important subdirectories to know:
- **/root** – The home directory of the all-powerful root user
- **/etc** – Generally contains the Linux configuration files – files that control when and how programs startup
- **/home** – The user's directory
- **/media** – Where CDs and USB devices are usually attached or mounted to the filesystem
- **/bin** – where application binaries (the equivalent of executables in MS Windows) reside
- **/lib** – Where you will find libraries (shared programs that are similar to windows DLLs)



- **/dev** – where all device files live, such as external USB or a webcam.
- **/var** – Short for variable. It is where all programs store runtime information such as user tracking, system logging, caches, and other files that system programs manage and create
- **/proc** – Contains information about your system such as CPU and your Linux system kernel. It is a virtual system
- **/boot** – Stores the files and folders your system needs to start
- Even though various Linux distributions have minor differences, the filesystem layouts are very similar
- The best way to understand the Linux directory structure is to follow some of the above suggestions and familiarize yourself with how it works.



Navigation

- The first thing we need to learn (besides just typing) is how to navigate the filesystem on our Linux machine. We introduce the following commands in this lesson:
- **pwd** - Print name of current working directory.
- **cd** - Change directory.
- **ls** - List directory contents.



- If you don't know which directory you are working in, you can use the ***pwd*** (print working directory) to check
- You can type ***ls*** to show the contents of your current working directory by default or any other directory you specified.
- You can type ***ls -l*** to see a long list of all the contents in a directory. Information displayed may be in different colors. The general rule of thumb is that, *blue* is a *folder*, *white* is a *file* and *green* is a *program* or a *binary*.
- The ***cd*** helps you to navigate the filesystem
- Use the ***clear*** to clear the contents on the screen, you can also hold ***ctrl + l*** (Not supported by all Linux distributions)



Some helpful shortcuts

Shortcut	Result
<code>cd</code>	Changes the working directory to your home directory.
<code>cd -</code>	Changes the working directory to the previous working directory.
<code>cd ~username</code>	Changes the working directory to the home directory of <i>username</i> . For example, <code>cd ~bob</code> changes the directory to the home directory of user <i>bob</i> .

```
ccs@localhost:~> whoami
ccs
ccs@localhost:~> pwd
/home/ccs
ccs@localhost:~> cd ~salsafh
ccs@localhost:/home/salsafh> pwd
/home/salsafh
ccs@localhost:/home/salsafh> whoami
ccs
ccs@localhost:/home/salsafh>
```



Exploring the Filesystem

- Now that we know how to move around the filesystem, it is time for a guided tour of our Linux system.
- **ls** – list directory contents
- **file** – determine file type.
- **less** – view file contents



- **ls** – is probably the most widely used command. With it, we can see directory contents, determine a variety of important files and directory attributes. As we have seen, we simply enter `ls` to see a list of files and sub-directories in the current working directory.
- However, we can specify a directory to `ls` to see its contents:

```
user@localhost ~]$ ls usr/bin
```

- We can specify multiple directories, for instance, we can list both the user's home directory (using `~`) and `/usr`

```
user@localhost ~]$ ls ~ usr/bin
```



Options and Arguments

- Commands are often followed by one or more *options* that modify their behavior and, further, by one or more *arguments*, the items upon which the command acts. So most commands look something like this:

command -options arguments

- Most commands use options consisting of a single character preceded by a dash, such as `-l`. But many commands, including those from the **GNU** project, also support long options, consisting of a word preceded by two dashes.



- Many commands allow multiple short options to be strung together. In the following example, the `ls` command is given two options, the `/` option to produce the long format output, and the `t` option to sort the result by the file's modification time:

```
[user@localhost ~]$ ls -lt
```

```
[user@localhost ~]$ ls -lt -reverse
```

reverse the order of the sort

- The `ls` command has a large number of possible options. Use command: `[user@localhost ~]$ man ls` to see all the possible options.



A Longer Look at Long Format

As we saw before, the `-l` option causes `ls` to display its results in long format. This format contains a great deal of useful information. Take a look at the following output:

```
salsafh@localhost:~> ls -l
```

```
-rwxr-xr-x 1 salsafh users 285 Dec 28 12:14 bash_scripting.sh
drwxr-xr-x 1 salsafh users  0 Oct 31 09:07 bin
drwxr-xr-x 1 salsafh users 550 Apr 17 13:55 Codes
drwxr-xr-x 1 salsafh users  86 Jan  8 22:36 config
-rw-r--r-- 1 salsafh users 3084649 Dec 20 14:01 datetime
-rw-r--r-- 1 salsafh users  0 Jan  8 22:37 db.sqlite3
drwxr-xr-x 1 salsafh users 1252 Apr 17 04:44 Desktop
drwxr-xr-x 1 salsafh users 1188 Apr 13 06:28 Documents
```



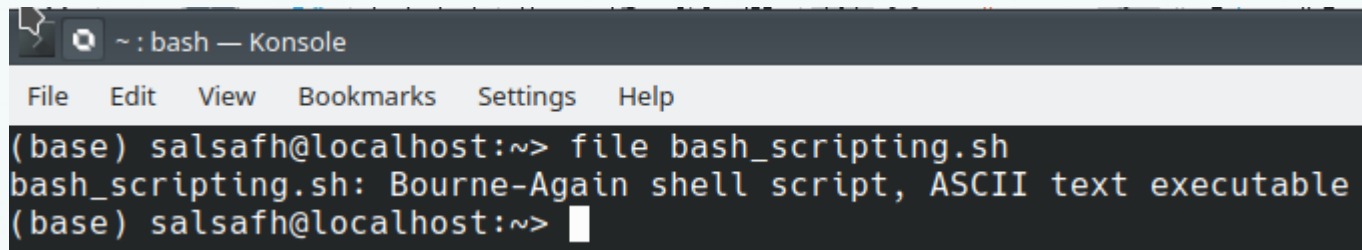
- Let us look at the different fields from one of the files and examine their meanings in the table below:

Field	Meaning
-rwxr-xr-x	Access rights to the file. The first character indicates the type of file. Among the different types, a leading dash means a regular file, while a d indicates a directory. The next three characters are the access rights for the file's owner, the next three are for members of the file's group, and the final three are for everyone else.
1	File's number of hard links.
salsafh	The user name of the file's owner.
users	The name of the group that owns the file.
285	Size of the file in bytes.
Dec 28 12:14	Date and time of the file's last modification.
bash_scripting.sh	Name of the file

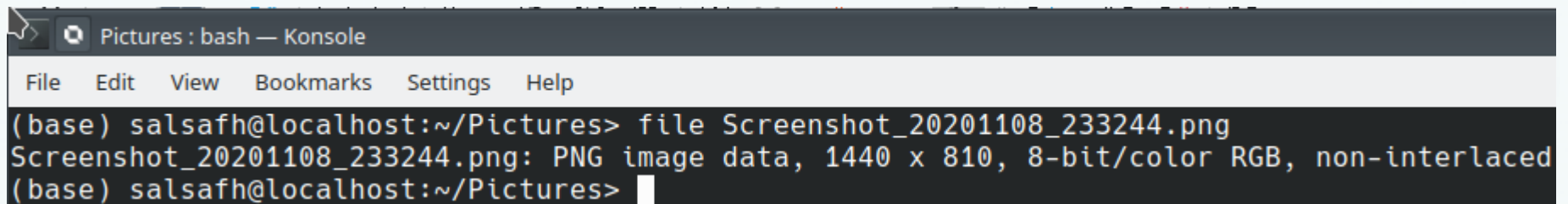
Determining a File's Type with file

- We use the file command to know what files contain. This can be achieved using:

`file filename`



```
~ : bash — Konsole
File Edit View Bookmarks Settings Help
(base) salsafh@localhost:~> file bash_scripting.sh
bash_scripting.sh: Bourne-Again shell script, ASCII text executable
(base) salsafh@localhost:~>
```



```
Pictures : bash — Konsole
File Edit View Bookmarks Settings Help
(base) salsafh@localhost:~/Pictures> file Screenshot_20201108_233244.png
Screenshot_20201108_233244.png: PNG image data, 1440 x 810, 8-bit/color RGB, non-interlaced
(base) salsafh@localhost:~/Pictures>
```



Viewing File Contents

- There are several commands to display the contents of a file to the terminal. The following are some of the commands that can be used to view the contents of a file:
- **more** – file perusal filter for crt viewing. Is a filter for paging through text one screenful at a time.
- **less** – opposite of more. Unlike more, it allows backward movement in the file as well as forward movement.
- **cat** – concatenate files and print on the standard output.
- **nl** – number lines of files



less Commands

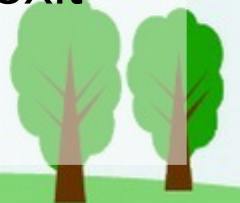
- less is the most widely used command for viewing file contents.

Below table list the most common keyboard commands used with less.

- The less command is used as follows:

`less filename`

- Once started, the less program allows you to scroll forward and backward through a text file. For example, you can use less to examine the file that defines all the system's user accounts.
- If the file is longer than a page, we can scroll up and down. To exit less, press the Q key on the keyboard.



less Keyboard commands

Command	Action
PAGE UP or b	Scroll back one page.
PAGE DOWN or Spacebar	Scroll forward one page.
Up Arrow	Scroll up one line.
Down Arrow	Scroll down one line.
G	Move to the end of the text file.
1G or g	Move to the beginning of the text file.
/characters	Search forward to the next occurrence of <i>characters</i> .
n	Search for the next occurrence of the previous search.
h	Display help screen.
q	Quit less.



Class Activity

1. **cd** into a given directory.
2. List the directory contents with **ls -l**.
3. If you see an interesting file, determine its contents with **file**.
4. If it looks as if it might be *text*, try viewing it with **less**.



The Symbolic Links

- A symbolic link, also known as a symlink or soft link, is a special type of file that points to another file or directory.
- There are **two** types of links in Linux systems.
- **Hard link** – you can think a hard link as additional name for an existing file. Hard links are associating two or more names with the same inode. You can create one or more hard links for a single file. Hard links cannot be created for directories and files on a different filesystem or partition.
- **Soft link** – A soft link is something like a shortcut in Windows. It is an indirect pointer to a file or directory. Unlike a hard link, a symbolic link can point to a file or a directory on a different filesystem or partition.



How to create links

- In – is a command-line utility for creating links between files. By default, the In command creates hard links. To create a symbolic link, use the -s (--symbolic) option.
- The In command syntax for creating symbolic links is as follows:

```
ln -s [OPTIONS] FILE LINK
```

- If both the FILE and LINK are given, In will create a link from the file specified as the first argument (FILE) to the file specified as the second argument (LINK).
- If only one file is given as an argument or the second argument is a dot (.), In will create a link to the file in the pwd. The name of the symlink will be the same as the name of the file it points to.



Creating symlink to a file

- To create a symbolic link to a file, open a terminal and type:

```
ln -s source_file symbolic_link
```

- Replace `source_file` with the name of the existing file for which you want create the symbolic link and `symbolic_link` with the name of the symbolic link
- The `symbolic_link` parameter is optional. If you do not specify the symbolic link, the `ln` command will create a new link in your `pwd`.
- The following example creates a link named `my_link.txt` to a file name `my_file.txt`:

```
$ ln -s my_file.txt my_link.txt
```



Creating symlink to a directory

- The command for creating a symbolic link to a directory is the same as when creating symbolic link to a file. Specify the directory name as the first parameter and symlink as the second parameter.
- The following example create a symbolic link from */mnt/my_drive/movies* directory to the *~/my_movies* directory:

```
ln -s /mnt/my_drive/movies ~/my_movies
```

- You can use `ls -l` to verify that the directory was created.



Overwriting Symlinks

- If you try to create a symbolic link that already exists, the ***ln*** command will display an error message. You can use the **-f** (**--force**) switch to force the creation.

```
$ ln -sf my_file.txt my_link.txt
```

- To delete a symlink, use either the ***unlink*** or ***rm*** commands.

```
unlink symlink_to_remove
```

```
rm symlink_to_remove
```

- Whichever command you use, do not append a **/** trailing slash at the end of its name because Linux will assume it is a directory.
- If you delete or move the source of a symlink to a different location, the symbolic link file will be left broken (dangling) and should be removed.

