"Mastering the Linux File System" Cheat Sheet

1. The Linux File System

The Linux File System follows a tree-like structure starting at a base (or root) directory, indicated by the slash (/).

Locations on the file system are indicated using file paths.

Paths that start at the base directory (/) are known as absolute paths.

Paths that start from the current working directory of the shell, are known as relative paths.

For example both of these examples refer to a file called "file1.txt" in the Documents folder for a user called Sarah. The relative path assumes the shell is in Sarah's home directory.

Absolute: /home/sarah/Documents/file1.txt

Relative: Documents/file1.txt

For more information about the structure of the Linux file system, please refer to the File system cheat sheet provided in the resources section of the lecture entitled "The Structure of the Linux File System"

2. Key Commands for Navigating the File System

pwd	Print on standard output the absolute path to the shell's current working directory.
cd [<new location>]	Change the shell's current working directory to the optional <new location>. If no location is provided, return to the user's home directory.
ls [<location>]	List out the contents of the optional <location> directory. If no <location> is provided, print out the contents of the shell's current working directory.

3. Key Shortcuts when Navigating File System.

~	The current user's home directory
.	The current folder.
..	The parent directory of the current folder.

4. Wildcards and Regular Expressions

Regular expressions are patterns that can be used to match text. In Linux, they are used to allow a user to make rather generic expressions about what files they want a command to operate on.

Creating regular expressions to match filenames is known as globbing.

The regular expression patterns can be made using special building blocks known as wildcards.

Wildcards are symbols with specific meanings to the shell.

We covered the 3 most common types of wildcard:

*	Matches anything, regardless of length.
?	Matches anything, but for one place only.
[options]	Matches any of the options inside for 1 place only.

These 3 wildcards are incredibly versatile and will cover an exhaustive amount of your regular expression requirements.

For more on wildcards please check out these amazing blog posts (blog post 1 & blog post 2)

5. Creating Files and Directories

touch <file>	Creates an empty file. e.g. touch ~/Desktop/file1.txt
mkdir <directory>	Creates an empty directory. e.g. mkdir ~/newdir

6. Deleting Files and Directories

rm <file>	Remove a file e.g. rm ~/Desktop/file1.txt
rm -r <directory>	Removes a directory. e.g. rm -r ~/newdir
rm –i	Removes in an interactive manner. This is a good safety measure.
rmdir <empty directory>	Only remove empty directories e.g. rmdir ~/emptydir

7. Editing Files with the Nano Editor

Nano is a versatile terminal-based text editor. It comes with a built-in toolbar of various commands and all one needs to know in order to use these options are the meaning of 2 special symbols.

^	This is the CTRL key on your keyboard. For example, ^O is CTRL + O.
M-	This is the "meta" key on your keyboard. Depending on your keyboard layout this may be the ALT, ESC, CMD key. Try it out ☺ Assuming M- is the ALT key, then M-X is ALT + X

Here is a link to the official documentation for nano if you would like to learn more.

8. The Locate Command

The locate command searches a database on your file system for the files that match the text (or regular expression) that you provide it as a command line argument.

If results are found, the locate command will return the absolute path to all matching files.

For example:

```
locate *.txt
```

will find all files with filenames ending in .txt that are registered in the database.

The locate command is fast, but because it relies on a database it can be error prone if the database isn't kept up to date.

Below are some commands to update the database and some reassuring procedures in case one cannot access administrator privileges.

Locate -S	Print information about the database file.
sudo updatedb	Update the database. As the updatedb command is an administrator command, the sudo command is used to run updatedb as the root user (the administrator)
Locate --existing	Check whether a result actually exists before returning it.
locate –limit 5	Limit the output to only show 5 results

9. The Find Command

The find command can be used for more sophisticated search tasks than the locate command. This is made possible due to the many powerful options that the find command has.

The first thing to note is that the find command will list both files *and* directories, below the point the file tree that it is told to start at.

For example:

```
find .
```

Will list all files and directories below the current working directory (which is denoted by the .)

```
find /
```

Will list all files and directories below the base directory (/); thereby listing everything on the entire file system!

By default, the find command will list everything on the file system below its starting point, to an infinite depth.

The search depth can however be limited using the –maxdepth option.

For example

```
find / -maxdepth 4
```

Will list everything on the file system below the base directory, provided that it is within 4 levels of the base directory.

There are many other options for the find command. Some of the most useful are tabulated below:

-type	Only list items of a certain type. –type f restricts the search to file and –type d restricts the search to directories.
-name "*.txt"	Search for items matching a certain name. This name may contain a regular expression and should be enclosed in double quotes as shown. In this example, the find command will return all items with names ending in .txt.
-iname	Same as –name but uppercase and lowercase do not matter.
-size	Find files based on their size. e.g –size +100k finds files over 100 KiB in size –size -5M finds files less than 5MiB in size. Other units include G for GiB and c for bytes**.

** **Note**: 1 Kibibyte (KiB) = 1024 bytes. 1 Mebibyte (MiB) = 1024 KiB. 1 Gibibyte = 1024 MiB.

A supremely useful feature of the find command is the ability to execute another command on each of the results.

For example

```
find /etc –exec cp {} ~/Desktop \;
```

will copy every item below the /etc folder on the file system to the ~/Desktop directory.

Commands are executed on each item using the –exec option.

The argument to the –exec option is the command you want to execute on each item found by the find command.

Commands should be written as they would normally, with {} used as a placeholder for the results of the find command.

Be sure to terminate the –exec option using \; (a backslash then a semicolon).

The –ok option can also be used, to prompt the user for permission before each action.

This can be tedious for a large number of files, but provides an extra layer of security of a small number of files; especially when doing destructive processes such as deletion.

An example may be:

```
find /etc –ok cp {} ~/Desktop \;
```

10. Viewing File Content

There exist commands to open files and print their contents to standard output. One such example is the cat command. Let's say we have a file called hello.txt on the Desktop.

By performing:

```
cat ~/Desktop/hello.txt
```

This will print out the contents of hello.txt to standard output where it can be viewed or piped to other commands if required.

One such command to pipe to would be the less command. The less command is known as a "pager" program and excels at allowing a user to page through large amounts of output in a more user-friendly manner than just using the terminal.

An example may be:

```
cat ~/Desktop/hello.txt | less
```

Or more simply:

```
less ~/Desktop/hello.txt
```

By pressing the q key, the less command can be terminated and control regained over the shell.

Here are some other ways to view file contents:

tac <path/to/file>	Print a file's contents to standard output, reversed vertically.
rev <path/to/file>	Print a file's content to standard output, reversed horizontally (along rows).
head –n 15 <path/to/file>	Read the first 15 lines from a file (10 by default if -n option not provided.)
tail –n 15 <path/to/file>	Read the bottom 15 lines from a file (10 by default if -n option not provided.)

11. Sorting Data

A useful ability when working with file data is to be able to sort it either alphabetically or numerically. This behaviour is handled using the sort command.

By default, the sort command sorts smallest first. So if sorting alphabetically, it will by default sort from a – z. If sorting numerically, it will put the smallest numbers first, and largest last.

Here are some common options when using the sort command:

sort -r	Reverse the default sorting order.
sort –n	Sort in a numerical manner.
sort –u	Sort data an only return unique entries.

It is also possible to sort tabular data using the sort command using one of the columns. This is possible by providing a KEYDEF as an argument to the –k option.

```
sort –k <KEYDEF>
```

KEYDEFS are made using a column number and then additional options can be added (without dashes).

As an example:

```
sort –k 5nr
```

The KEYDEF is 5nr. This will sort using column 5 of the data, and sort numerically (-n option) but in reverse (-r option).

12. Searching File Contents

The ability to search for and filter out what you want from a file or standard output makes working with the command line a much more efficient process.

The command for this is called the grep command.

The grep command will return all lines that match the particular piece of text (or regular expression) provided as a search term.

For example:

```
grep hello myfile.txt
```

Will return all lines containing the word "hello" in myfile.txt

and

```
ls /etc | grep *.conf
```

will return all lines with anything ending in ".conf" in data piped from the ls command.

Some common options when working with the grep command include:

grep -i	Search in a case insensitive manner (upper case and lowercase don't matter).
grep -v	Invert the search. i.e return all lines that DON'T contain a certain search term.
grep -c	Return the number of lines (count) that match a search term rather than the lines themselves.

13. File Archiving and Compression

File archiving and compression is a two-step process in Linux.

First one creates a tarball to hold all the files that comprise the archive, and then compresses that tarball using one of a variety of compression algorithms.

For detailed information on file archiving and compression, please refer to the File Archiving and Compression cheat sheet provided in the resources section of the lecture entitled "File Archiving and Compression – Part 2"

"Well done! You should now be well prepared for working with files using the Linux command Line. I hope that this cheat sheet is useful to you if you ever need a refresher!"

Best wishes, Ziyad.