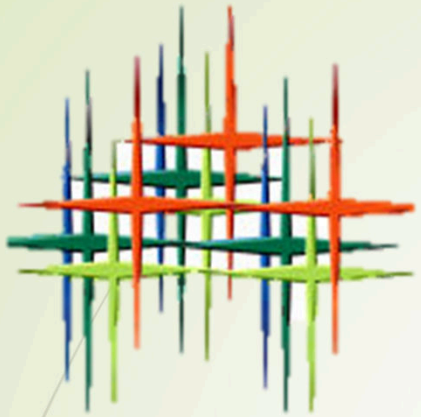

ABC IT Education

WE'LL TAKE YOU FROM ZERO TO HERO IN A SNAP

Linux Systems Administration

Linux Man Pages and Accessing Directories

Man Pages

- We have just seen some of the basic commands in Linux.
- To get additional help on a command “man pages” will be the way to get it.
- We’ll look at how to move around man pages, what the '$path' environment variable is for, what the 'which' command does, how to get command help in Linux, and how to search for man pages.
- To find help on any command in Linux we use the “man” command as follows $ man [options] [command].
- When you use the man command you also have to know how to navigate through the screens and here below is how to navigate through the man pages.
- Enter (Return) - moves man pages down one line,

- Space bar - moves down an entire screen or 1 page at a time
- Lowercase 'g‘ – takes you to the top of the page
- Uppercase 'G‘ - takes you to the bottom of the man pages
- ‘q’ – Quit the current man pages
- **Environment Variables**
- In Linux, environment variables are used to store valuable information, and to see the available environment variables simply type the “env” command like $ env
- Most environment variables are uppercase letters and are stored as a name-value pair eg USER=cwmonie

- To see the contents of an environment variable, we simply use the echo command as follows 'echo $<name of variable>' eg echo $USER
- There are many environment variables, but for now the variable we will focus on is the PATH variable which controls the command search i.e. where Linux looks for commands you type in the CLI (command line interface)
- The variable is a list of directories separated by colons, and when you enter a command Linux will look for that command in the directories listed in the PATH variable
- Type the command echo $PATH, and the system will list all the directories in the command search. These directories are searched from first to last.

- When you type a command and it is found in the any directory in the PATH variable, it is executed, otherwise you get an error 'command not found'. Remember the command is searched for from the first directory in the PATH to the last.
- **'which' Command**
- 'which' command tells you the location or the full path of any command that you are executing.
- To see how it works, type **$ which ls** at the console and the system will tell you the location of the ls command it is executing.
- **Note**: The same command can be in more than one place, but which one gets executed depends on your PATH variable

- Let's try another example with the 'tac' command.
- Look carefully and you will notice that 'tac' is similar to a command we saw earlier 'cat' but spelled backwards, and 'tac' displays the contents of a file in reverse order.
- **Another way of getting help in Linux**
- Apart from the man pages, many of the Linux commands will show you how to use them at the command line, either by using the '-h', '—h', or '—help' flag, but note that if a command does not offer any of the flags them you must use 'man' to get help about that command.
- Sometimes we are not sure which command to use in a given situation. When this happens, we can search the man pages with $ man –k <search term>
- As an example search for calendar $ man –k calendar

Accessing Directories

- We can access directories using the 'cd' command now let's stress on some shorthand ways to access directories and well as executing commands that are not in the search path i.e. are not in the directories in the PATH environment variable.
- We will also take a second look on how to create and remove directories.
- By the way what are directories really? Directories are simply containers for files and other directories and usually provide a tree-like structure of the Linux file system.
- In Linux, directories can be accessed by their name, a full path (absolute paths) or by a couple of short cuts referred to as relative paths

- As we saw in an earlier lesson, a single dot (.) represents the directory where you are i.e. the current directory and so in Linux systems, a single period is often referred to as 'dir'.
- The single dot is also referred to as ‘dot’ or ‘period’ depending on how it is used.
- Two dots, next to each other, represent the parent directory of your current directory. As an example if you are in a directory /home/cwmonie, the current directory (.) is cwmonie and the parent directory (..) is home
- Another short cut to remember is the minus (-). If used with the ‘cd’ command as follows $ cd -, it will take you to your previous directory.

- Let's see how that works.
- If you are in a directory /home/cwmonie, then I do a **$ cd /tmp** that puts me in the /tmp directory, now in the /tmp directory you do a **$ cd -**, it will take me back to /home/cwmonie
- When we write directories, they begin with / (slash) and are separated by / (slash also called forward slash).
- Apart from directories beginning with a slash (/), you can also end directories with a /, example if you enter the command $ cd /home/cwmonie/ it does the same thing as the command $ cd /home/cwmonie
- We have seen directory short cuts before but let's look at them again.

- cd . : does nothing (stay in current directory)
- cd .. : takes you to the parent of your current directory
- cd - : takes you to your previous directory
- cd /home => cd /home/
- **OLDPWD environment variable**
- OLDPWD is another environment variable which holds your previous directory.
- If you are in /home/cwmonie and you $ cd /tmp, this takes you to the tmp directory and when you do the echo command on OLDPWD as in $ echo $OLDPWD, it will display /home/cwmonie. Note that is where you were before going to the /tmp directory

- Earlier we talked about how the ‘PATH’ environment variable determines your command search path, but if you know the exact location (full path or absolute path) of the command you can specify it and this will allow you to run commands that are not in your search path
- If you want to execute a command in the /bin directory, you can simply type the full path as in $ /bin/<command>
- To execute a command in the directory where you are in, just type ‘$ ./command‘, if in the parent directory just type ‘ $ ../command‘ – remember .. means parent directory
- If you do a 'which' on the ‘cat’ command it tells you ‘cat’ is in the /bin directory so running ‘cat is the same as running ‘bin/cat‘ or ‘/bin/cat’ i.e. you get the same result
- To run a different ‘cat’ in a directory not in the PATH you must specify its full path.

- Try to run a different cat that resides in '/home/jason/my-cat‘, it would not run because '/home/jason/my-cat' is not in the command search path
- To run the command, change directory to '/home/jason/my-cat‘, and execute the command thus $ ./cat, where ‘dot’ is the current directory where the command is in. Also, 'slash' is the directory separator.
- Reminder:
- If you want to execute something in your current directory, just remember - './the_command_name'.

- **‘mkdir’ Refresher**
- To create a directory, you use the 'mkdir' command as follows $mkdir [-p] <directory_name>
- Note '-p' is in brackets meaning that it's optional and mkdir requires that you give it a directory name to create.
- So $ mkdir newdir, will create a new directory, named 'newdir'.
- If we want to create a directory structure that is more than just one directory deep, we need to specify '-p‘ so that it will create the parent directories. **–p means parents**
- For example we could use, $ mkdir -p 1/2/3.
- That would create 3 directories with one command.

- 1, that's one directory,
- 1/2 - that is your second depth, your second directory,
- 1/2/3 - that is your third depth, your third directory.
- **‘rm’ and ‘rmdir’ Refresher**
- To remove a directory, use the 'rmdir' command, but the 'rmdir' command only removes directories that are empty, that have no file contents.
- To remove a directory and all the sub directories and files, you need to use 'rm –rf directory_name'.
- And that recursively deletes everything in that directory and below that directory.

- **Very Important Note**
- When you're working at the command line in Linux, there is no undo (undelete), and there is no trash can that you can go and retrieve a file that you've deleted. When you delete something, it's gone.
- Thus, when you recursively delete something, all of those files and directories that you recursively deleted, are gone and you can't get them back, **so be extremely careful when you're using 'rm -rf'.**

- **Listing Files and Understanding the Output**
- In this section, we'll talk about the components of the long listing format of the 'ls' command i.e. ls -l.
- We'll touch on the hidden files and directories, list files by type and sort them by time.
- We're going to talk about handling spaces in file names and directory names and also talk about what symbolic links and what they are.
- Here below is a sample of the 'ls –l' command

```
drwxrwxr-x@    3 bryce  staff    96B Apr  9 18:35 Creative Cloud Files
drwx------+    7 bryce  staff   224B Aug 21  2017 Pictures
-rw-r--r--     1 bryce  staff     0B Jul 25  2017 first_day.html
```

- For ‘ls’ if you just want to see what files or directories exist, just run ‘ls’ without any options, but if you want to see detailed information, then use 'ls -l'.
- If you do ‘ls –l’ and get the listing below

```
drwxrwxr-x@    3 bryce  staff    96B Apr  9 18:35 Creative Cloud Files
drwx------+    7 bryce  staff   224B Aug 21  2017 Pictures
-rw-r--r--     1 bryce  staff     0B Jul 25  2017 first_day.html
```

- The first bit of information that is displayed is a series of characters representing the files' permission.
- The second part of information is the number of links to that file or directory.
- Next is the owner of that file/directory, then group that file/directory belongs to, the size of that file/directory, the last time that file/directory was modified, then the name

- Files or directories that began with a period or a dot, are considered hidden files.
- Running ‘ls’ or ‘ls –l’ will not display hidden files by default, so in order to see hidden files, you need to use the ‘-a’ options to ls, thus you could run ‘ls –a’, 'ls -l –a’ or ‘ls –a –l’ or ‘ls –al’ or ‘ls –la’
- You can use 'ls -F' to reveal file types and 'ls -F' appends a character to a file or directory name that tells you what it is.
- If it ends in '/' or forward slash, then it's a directory.
- If it ends in a percent, that's a link.
- A star or asterisk means that it's executable.
- Symbolic links (symlinks) are pointers that look like a file or directory, but it just points to where the actual file or directory is.

- You would operate on the symbolic link or symlink for short, the same as if it were the file that it's pointing to.
- Symlinks are often used to create short cuts to long file names or of long directory paths.
- Another common use of symlinks is to indicate the current version of an application.
- Let's change into the home directory. When we perform an ls, we don't know if current is a link, if it's a directory, if it's a regular file, or an executable file.
- We can get this information by running 'ls -F'.

- If you want to list files sorted by time, use 'ls -t'.
- Use the '-r' option to reverse the order.
- A common command that I like to use is 'ls -latr'. This provides a long listing that includes all files sorted by time in reverse.
- Look at the home directory with an 'ls -l'. It sorts the files and directories by name. If you want to reverse this, we can use long listing and 'r' for reverse as follows ‘ls –rl’
- Let's do a long listing including all files sorted by time, we see 'bash_history' is at the top, it's the most recently modified file.
- If we do 'ls - latr', that will reverse this output, and we'll see the most recently modified files at the bottom.

- Remember to list the directory contents recursively, use 'ls -R'. This will display all the contents of the current directory and any subdirectories below that.
- The tree command does the similar thing to 'ls -R‘ but it's more of a visual output. You can give tree a '-d' option and that will only list directories.
- And if you use a '-C', that will colorized the output.
- Tree is on a lot of Linux distributions, it may not be installed by default, so you may or may not have access to this command.
- Like the tree command, 'ls' has a '-d' option that only list directory names and not their contents.
- By default when you run 'ls directory_name', 'ls' will provide the contents of that directory.

- You can also tell 'ls' to colorized output by using 'ls --color'.
- This is just another option to distinguish file types, like you can do 'ls -F', or 'ls --color' or even combine the two.
- You can use 'ls --color' for ls to get color output but sometimes this is done by default on someone Linux distributions.
- The color codings like in this configuration, blue are directories, which we can confirm by using '-F'.
- **Spaces in Directory Names**
- To make your life a little easier when working at the command line, do not use spaces in file and directory names, instead get in the habit of using maybe, a hyphen or a dash or maybe an underscore where you would use a space, or even something called CamelCase

- CamelCase is where you can capitalize the first letter of each word.
- Sometimes soon you are going to encounter a file name
- that has a space in it, and it may be a file that you didn't create.
- How you can deal with this is, wrapping the file name in quotes, single quotes or double quotes.
- Another thing you can do is escape the space character.
- To escape something you proceed it with a backslash.
- If you run 'ls' against a file called ‘my notes.txt’, 'ls' treats it as 2 different files, not 1 file name with a space.
- To get around this, we can use quotes, double quotes or single quotes.

- We've talked about some of the more common and useful ls options.
- We've learned about symlinks, talked about hidden files and directories and how they start with a dot.
- We also talked about a couple of strategies for dealing with spaces and file names.
- Here are the ls options we covered today.
- '-a' to list all files.
- '--color' to list files with colorized output.
- '-d' to show directory names and not directory contents.
- '-l' for the long listing format.
- '-r' for reverse order.
- '-R' to list files recursively
- '-t' to sort files by time.