Basic Linux Command

Working with Files

Table 1. Commands for navigating and working with files/directories

Command	Description
ls	List directory contents
cd	Change directory
pwd	Print working directory
mkdir	Make directory
rmdir	Remove directory (*only works if empty*)

Commands for Working with Files

Some of the most useful and basic commands for using Linux are listed in Table 2. These commands come in handy for working with files. Most are used by providing a file name as an argument.

Table 2

Command	Description
touch	Creates a file or updates the timestamp on an existing file
cat	Outputs the full contents of a file
head	Returns the first X lines of a file starting at the top
tail	Returns the first X lines of a file starting at the bottom
cp	Copies a file or directory
rm	Removes a file or directory
mv	Moves a file or folder
less	Displays contents of file while allowing easy scrolling up and down
diff	Compares two files for differences
cmp	Checks if two files are identical on a byte-by-byte level
file	Gets information on file type

Create Files or Update Timestamps with the touch Utility

Sometimes you want to create a blank file, either as a placeholder that you plan to edit later or possibly as an indicator, for example, with a lock file. The touch command allows you to quickly create a blank file or multiple blank files. Simply run the command and use the desired file name or names as the argument, for example:

```
touch notes.txt
```

Or for multiple:

```
touch file1 file2 file3
```

Another thing the touch command can be used for is updating the timestamp on a file. After running a script, you may want to update a file with touch that was otherwise unused so you can leave some trace of when the script finished. For example, you might update a log file's timestamp despite not adding any new logs so others (people or programs) can infer that the script ran and no logs were produced. This is done in

exactly the same way as creating a file except instead of providing the path to a file you want to create, you provide the path to an existing file:

```
touch log.txt
```

After executing touch on an existing file, you can use `ls -l` in the directory of that file to confirm that the timestamp has been updated. It's important to note that touch will never modify the contents of an existing file so don't worry about overwriting any existing contents with a blank file

Get File Contents with Cat

When using the command line, cat is one of the most useful commands to know. cat simply takes the contents of a file and outputs them to the command line. This allows you to either visually see contents of that file (without opening and closing a program) or to use the contents of that file as the input for some other program.
As an example of using cat, you can run the following which will output the contents of a file on your system:

```
cat /etc/passwd
```

This file lists the users on your system and some related information, but understanding the content isn't important. What's important here is that you can use cat to take the contents of any file on your system and show it as terminal output.

Get Less Content with Head or Tail

If you understand what "*cat*" does, you'll just as easily be able to understand the head and tail commands. When you use *cat*, the full contents of a file are returned. With large files, this means you may get several pages of content at once and have all your previous work and commands pushed up the screen.
If you want to get a preview of a file but don't want the whole thing, you can use head which will return the first 10 lines of a file by default.

```
head /etc/passwd
```

.
To get the first n lines, use: `head -n /etc/passwd`

To get the last n lines, use: `tail -n /etc/passwd`

Copying Files with cp

It's a very simple but very useful command which stands for copy. When using the command the first argument is the file you want to copy and the second argument is the location to copy to, for example copying `file1` in `location1` to `location2` file2 would be done with:

```
cp file1 location2
```

Running the preceding command would result in a new file called file2 which contained the same contents as file1.
In addition to copying files, cp can also be used to copy whole folders. To use cp with folders, you need to specify the -r flag, which stands for recursive (similar to using the rm command with folders). So copying a folder would be much the same as copying a file,

for example:

```
cp -r folder1 folder2
```

Removing Files with rm

We've already made use of `rm` due to the limitations of `rmdir` in the section on directories. Be aware that the `rm` command is primarily used for deleting files, and when doing so, there is no need to include the -r flag, for example:

```
rm file1
```

Moving Files with mv

Another very popular built-in command, `mv` allows you to move a file or directory to a new location. It's used very similarly to cp except you only end up with one file, for example, if we use:

```
mv file1 location2
```

Also like many of the other commands we've looked at, you can use mv with directories, but with mv there is no need to use a special flag, you can simply use

```
mv folder1 folder2
```

Be aware that mv will overwrite a file without a warning if it already exists. For example, if I moved file1 to file2 but file2 already existed, my original file2 will be lost forever. If you're worried about that happening, there is a special flag -i which will prompt you before overwriting anything.

```
mv -i file1 location2
```

Regular Use of Less

less is used for viewing file text data in a way that allows you to start at the top and slowly scroll your way down. Let's review using less normally once more before opening some other file types. First create a long file with several lines of text using the command seq, short for sequence. The seq command takes a starting number and an ending number as arguments and returns a sequence of numbers between them:

```
seq 1 999
```

```
seq 1 999 > numbers.txt
```

Get File Type

If you're coming from Windows, you may be used to the concept that the extension of a file determines the type and what program it's run with. On Linux, file extensions are often used, but this is simply for the benefit of the human reader. File extensions are not mandatory and in some cases not used.

You may find a text file or program which has a name but no extension. In this situation, you may find the file command to be useful. Given a file as input, it will return information on the file type. For example, if we run the file command on file1 created in the last section, by passing the file location as an argument like below:

```
file file1
```

Command Information with type, which, whereis, or locate

Similar to getting information about a file with file, we can get information on a command using `type`, `which`, `whereis`, or `locate`. The first command type is built into

bash itself and searches your path and gets information on the command when found, for example:

```
type ls
```

Then with which we can find the location of the executable:

```
which ls
```

Similarly, we can use whereis and find the executable location, source location, and manual page files for the command.

```
whereis ls
```

Opening Compressed Folder with Less Pipe

Compressed files and folders can be opened with less when you have less pipe installed. To demonstrate, create a folder with some files and compress them using tar (a common utility for compressing and uncompressing files):

```
cd /tmp
mkdir folder
cd folder
touch file1 file2 file3
cd ..
tar -zcvf folder.tar.gz folder
```

After running these commands, you'll have a compressed folder which contains three empty files. Next let's try opening it with less. You should get a list of folders and files including the permissions of each file.

File Structure Visualization with Tree

Besides `ls`, ranger is my most used program for viewing file structure. However, another worth a mention is tree, which will need to be installed on most distros. 'tree' is also very lightweight, instead of opening up a full program like ranger to explore the file structure, 'tree' can be used to immediately create a visualization of your file structure – for example, if I navigate into a project and run the following command

```
tree -L 2
```

Note two here signifies how many levels (or directories) deep show; to go deeper, simply increase the number.