portfolio_4

2023-02-16

The Command Line

In this portfolio we will cover working with the command line. We are going to skip quickly over the super basics and then start listing some useful command and what they do. After describing the basics we will then dive into the more advanced. This portfolio serves as a good reference point for when working with the command line.

The Basics

ls can be used to list current contents of the directory, there are lots of variations of this command that you can use to list in different ways, a good explanation of all these commands is found here "https://www.freecodecamp.org/news/the-linux-ls-command-how-to-list-files-in-a-directory-with-options/".

cp can be used to copy files like so: *cp* .

rm can be used to remove a file like so: *rm* .

Note we need to use the **-r** flag with the above commands if we want to remove or copy a folder with all its contents, it stands for recursive.

mv can be used to move a file like so: *mv* .

cat can be used to print the contents of a file to the terminal. Similarly **head** can be used to print out the start of a file and **tail** for the end of a file, we can then use the **-n** flag followed by a number to change the number of files printed. We can use **tac** to print out each line from bottom to top, **rev** to print out the file in reverse, **uniq** prints out the file without lines that are repeated, **sort** to sort a file alphabetically (by default, can use flags to sort by other things).

We can use ___*___ called wildcard to pattern match, eg. _cat name*_ will print out all files (in current directory) that start with name and _cat *.txt_ will print all files (in current directory) that end with the .txt extension.

less can be used to open a "file viewer", to exit type q.

The following are a bunch of text editors that can be used: **nano**, **micro**, **emacs**, **jove**, **vim**. Let's look at how we can use nano. If we type *nano* then it will open the file in nano, if the file doesn't exist it will create the file and open it. Here are some of the basic commands in nano: *ctrl-o* is save, *ctrl-w* is search, *ctrl-x* is used to exit nano.

man can be used to launch the command line manual where we can check for command line commands.

grep is used to search inside files like so: *grep "" .* We can use flags to do thing such as count the number of times the string appears (*-c*), give the line numbers (*-n*), etc.

find lets us search for files like so: *find "".* Note that **.** is used to represent our current location and ~ is your home directory. Also note that we can use flag **-iname** if we want find to be case insensitive. We can also use **fzf** which stands for fuzzy find and we can use it similarly to find but we can find similar matches without having to use wildcards.

We can use > to write the output of a command to a file like so: *head >* , this will write the "head" of the file you name to a new file. We can use » to append to a file instead.

We can use multiple commands together by using the pipe operator |, eg. *cat | sort* which would feed the print out of the file you choose to the sort function before printing it to the terminal.

More Advanced

Now we will move onto looking at some more "advanced" things we can do with the command line. This will include working with remote computers and shell scripts.

To work remotely we use something called **ssh** (stands for secure shell) which is a program for opening a command line session on a remote computer. It opens a shell to communicate with a remote machine on the same network, using an encrypted secure connection. You can log in using a username and host name (eg. *ssh @*) or an IP address (eg. *ssh @*), there also exist other methods of logging in. After doing this you will be prompt to enter a password. You can see who else is logged in using **w** and leave ssh using the command **exit**. We can use **scp** to "secure copy" files to and from another computer like so: *scp @:* to copy file to other computer and *scp @: .* to copy from another computer to yours.

We can also create scripts where we can write lines containing commands and then run the script to execute all the commands contained in the script. To do this we create a file with the ".sh" extension and we place a sequence of commands (one per line), we can then run the file using **bash** (a popular shell program) like so: *bash .sh*. A command that is useful when creating such scripts is **echo** which works like print in most programming languages: **echo ""**.

Into action

Let's now put some of what we've been talking about into action, I have created a text file (named "odyssey.txt") containing the odyssey by homer (translated to english) and we will use it to put into practice some shell commands. We will use r markdown code chunks to write shell scripts and execute them using bash. Let's start by printing out the beginning of the odyssey:

```
head odyssey.txt
```

```
## PREFACE TO FIRST EDITION
##
## This translation is intended to supplement a work entitled "The Authoress of the Odyssey", which I pu
##
## I shall not here argue the two main points dealt with in the work just mentioned; I have nothing eitl
##
## (1) that the "Odyssey" was written entirely at, and drawn entirely from, the place now called Trapani
##
## (2) That the poem was entirely written by a very young woman, who lived at the place now called Trapa
```

Let's now print out the end in reverse:

```
tail odyssey.txt | rev
```

```
##
## ".ruolav fo rettam eht ni rehtona eno htiw gniyv era nosdnarg dna nos yM .ti ta eciojer deedni od I
##
## ".ti lruh dna raeps ruoy esiop neht ;rehtaf reh evoJ ot dna ,lesmad deye-eulb eht ot yarp-dlrow eht r
##
## ".dehsdoolb rehtruf tuohtiw ecno ta rettam eht elttes dna ,raw lufdaerd siht esaec" ,deirc ehs ",acal
##
## ".uoy htiw yrgna eb lliw evoJ ro ,efirts lufraw siht pots ,setreaL fo nos elbon ,sessylU" ,sessylU o
##
## .seitrap gnidnetnoc owt eht neewteb ecaep fo tnanevoc a edam yltneserp dna ,rotneM fo eciov dna mrof
```

Wow! Let's now count the number of times "blue" is said:

```
grep "blue" odyssey.txt -c
```

```
## 7
```

It's said 7 times! which is interesting because in Greek times they didn't have a word for blue, for example if we directly translate the color attested to a stormy dark blue see in greek writing we would find it translates to "wine-dark" in English. So here the translator has clearly opted to use the word blue instead of the other descriptors the ancient greeks would have used so the text makes more sense. Now let's create a file containing all the lines where the word blue is used and call it "blue", then print out this newly created file:

```
grep "blue" odyssey.txt > blue.txt
cat blue.txt
```

```
## With these words he led the way and the others followed after. When they had brought the things as he
## "My poor good man," said she, "why is Neptune so furiously angry with you? He is giving you a great d
## Then Minerva left Scheria and went away over the sea. She went to Marathon[59] and to the spacious st
## "The first I saw was Tyro. She was daughter of Salmoneus and wife of Cretheus the son of Aeolus.[94]
## "'When your crew have taken you past these Sirens, I cannot give you coherent directions[100] as to w
## The ship bounded forward on her way as a four in hand chariot flies over the course when the horses i
## On this Minerva came close up to him and said, "Son of Arceisius--best friend I have in the world-pra
```

Now we have read this file we are done with it, let's now delete it:

```
ls
echo ""
rm blue.txt
ls
```

```
## blue.txt
## odyssey.txt
## portfolio_4.pdf
## portfolio_4.Rmd
##
## odyssey.txt
## portfolio_4.pdf
## portfolio_4.Rmd
```

We also used ls in our script to print out the files in the directory before and after deleting "blue.txt" to check that its been removed. We also used echo to print nothing to the console to separate the directory contents for before and after deletion. Looking at the output we see that "blue.txt" has in fact dissapeared! Let's now print the newline, word and byte counts for the odyssey using wc:

```
wc odyssey.txt
```

```
##   2191 120178 632404 odyssey.txt
```

We didn't explain what **wc** was earlier, let's use the help of man to do this for us (ie. let's look it up in the manual):

```
man wc
```

```
## WC(1)                          User Commands                          WC(1)
##
## NAME
##        wc - print newline, word, and byte counts for each file
##
## SYNOPSIS
##        wc [OPTION]... [FILE]...
##        wc [OPTION]... --files0-from=F
##
```

```
## DESCRIPTION
##        Print newline, word, and byte counts for each FILE, and a total line if
##        more than one FILE is specified.  A word is a non-zero-length  sequence
##        of characters delimited by white space.
##
##        With no FILE, or when FILE is -, read standard input.
##
##        The  options  below may be used to select which counts are printed, al-
##        ways in the following order: newline, word,  character,  byte,  maximum
##        line length.
##
##        -c, --bytes
##               print the byte counts
##
##        -m, --chars
##               print the character counts
##
##        -l, --lines
##               print the newline counts
##
##        --files0-from=F
##               read  input  from the files specified by NUL-terminated names in
##               file F; If F is - then read names from standard input
##
##        -L, --max-line-length
##               print the maximum display width
##
##        -w, --words
##               print the word counts
##
##        --help display this help and exit
##
##        --version
##               output version information and exit
##
## AUTHOR
##        Written by Paul Rubin and David MacKenzie.
##
## REPORTING BUGS
##        GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
##        Report wc translation bugs to <https://translationproject.org/team/>
##
## COPYRIGHT
##        Copyright © 2018 Free Software Foundation, Inc.   License  GPLv3+:  GNU
##        GPL version 3 or later <https://gnu.org/licenses/gpl.html>.
##        This  is  free  software:  you  are free to change and redistribute it.
##        There is NO WARRANTY, to the extent permitted by law.
##
## SEE ALSO
##        Full documentation at: <https://www.gnu.org/software/coreutils/wc>
##        or available locally via: info '(coreutils) wc invocation'
##
## GNU coreutils 8.30             September 2019                          WC(1)
```

That wraps up this portfolio!

In the next portfolio we will be covering High Performance Computing (HPC) where we will use everything we've been talking about here and in particular ssh to perform work on a high performance computer.