## **Bash Command Practice 3**

We're really cooking now. Let's practice some more advanced file commands. But first, we need something to work with.

Make a new folder with mkdir and move into it with cd. Call it whatever you want, I'll just call mine "practice".

We're going to download a real text file using the "wget" command. This grabs a file from the internet and saves it to your current directory. Run this:

wget https://www.gutenberg.org/cache/epub/1661/pg1661.txt

This file contains the full text of *The Adventures of Sherlock Holmes*. Once it finishes downloading, run Is and you should see pg1661.txt.

We grabbed this longer file so we can practice some new commands. For example, "head" shows the first 10 lines of a text file:

head pg1661.txt

And "tail" shows the last 10 lines:

tail pg1661.txt

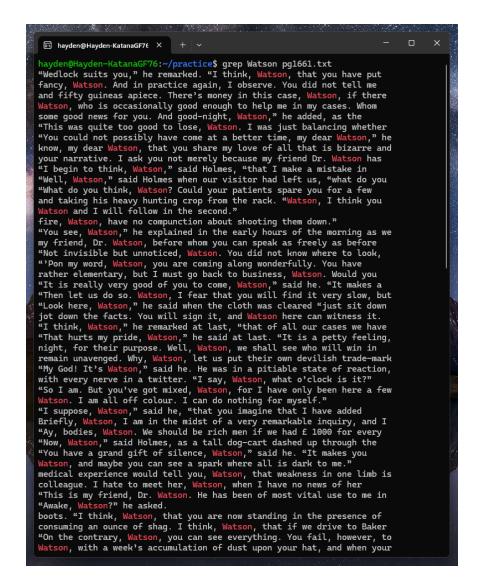
The file is so long that "cat" won't show us the entire thing; it will flood the terminal. But we can scroll through the file one line at a time with "less".

less pg1661.txt

Use the arrow keys to move up and down. Press "q" to quit when you're done. Now let's search the file using "grep". For example, if you want to find every line that mentions Watson:

grep Watson pg1661.txt

You should see this:



That's a lot of Watsons. If we use the "-c" flag we can count exactly how many Watsons there are.

grep -c Watson pg1661.txt

Looks like there are 81. Pretty cool!

Let's make a backup of the file by copying it with "cp".

cp pg1661.txt sherlock.txt

This makes a copy of pg1661.txt called sherlock.txt. Run Is to check that it worked (this should be a habit by now).

Now onto the really cool stuff.

This vertical bar I is called the "pipe" symbol. (It looks like a capital "i" or lowercase "L" but it's actually the symbol above the Enter key.) It takes the output of one command and sends it as input to another command.

Let's try it out. We'll use "grep" on sherlock.txt to find lines with "Holmes", then pipe that output into the "head" command.

## grep Holmes sherlock.txt | head

This takes the output of "grep Holmes sherlock.txt" and feeds it to the "head" command. You should see something like this:

```
hayden@Hayden-KatanaGF76:~/practice$ grep Holmes sherlock.txt | head The Project Gutenberg eBook of The Adventures of Sherlock Holmes Title: The Adventures of Sherlock Holmes
The Adventures of Sherlock Holmes
To Sherlock Holmes she is always _the_ woman. I have seldom heard him I had seen little of Holmes lately. My marriage had drifted us away while Holmes, who loathed every form of society with his whole Bohemian keen desire to see Holmes again, and to know how he was employing his "My dear Holmes," said I, "this is too much. You would certainly have "Peculiar—that is the very word," said Holmes. "It is not an English "What do you make of that?" asked Holmes.
```

Similarly, the greater-than symbol > saves the output of a command to a file. A single ">" will overwrite an existing file (or create a new one), and a double ">>" will append to the end of an existing file.

```
grep Holmes sherlock.txt > holmes.txt
```

This creates a new file "holmes.txt" that only contains the lines with the word "Holmes". Check it with head holmes.txt to see what that looks like.

```
grep the sherlock.txt > holmes.txt
```

This overwrites "holmes.txt", so it now contains only lines with the word "the". Check it again and you'll see that it's been overwritten.

```
echo "And they all lived happily ever after." >> holmes.txt
```

This adds that line to the end of holmes.txt. Run tail holmes.txt to see if it worked.

Hopefully you can see the potential in these tools, they're very powerful!

Chaining commands like this is one reason developers and IT people love the command line.