awk, sed, grep, large data scans

I was able to use regular expressions after I installed the MinGW environment on my Windows machine. I use them regularly for command-line navigation of the file system. I store my working files on a server across the network, using a terminal window to manipulate them. I have backup copies of all my work on a separate drive, and on a separate server for safety. I can lose a drive, or even a server, and still have copies of my work. I access the files using ssh (secure shell) from a terminal window in Linux or Windows by typing **ssh Hera** (the name of my larger server). Hera asks for its password, so I enter that too. Once the handshaking is complete I have access to my files on Hera.

At the next command prompt, I type **cd /mnt/*ons/*PR*/** , which takes me to kevin@Hera:/mnt/share/ Bacona Design lessons/lessons – PRIMARY. By using those three asterisks I was able to save a lot of typing. *ons instead of 'Bacona Design lessons' saved a lot of characters. Since there are spaces in the file name I would have had to add quotation marks to type the complete path. *PR* translated to lessons – PRIMARY, saving both time and typing. In the first case, *ons, I used the asterisk to represent the first part, asking for any path ending in ons. The trick is to know when there is only one path choice which does end in ons, otherwise you will wind up at the first path which ends in ons. In the second case, *PR*, I was seeking a path which has the two characters PR in upper case in the path. I could have used *RY and achieved the same result. It is your choice.

The next good reason for the MinGW environment is its method of reissuing old commands. While at a command prompt, hit the up arrow key to examine your command history. If you want to reuse any of those commands, scan backward with the up arrow until you find it, then hit Enter; or scan to one which best fits your desire and modify it.

Home and End are good keys to remember, along with the arrow keys, the backspace key, and the Delete key. All are valuable while moving through your command line history, and remain a lazy programmer. Laziness, hubris, and impatience are good values for a programmer; at least in the mind of Larry Wall :) Use the up and down arrow keys to scroll through the commands you have used in the past. Once you find one which is close to the one you want, use the left and right arrow keys to find the characters which need to change, then use the delete or backspace key to remove the unneeded ones. Use the Home key to get to the very start; useful when you need to add sudo to the beginning of the line. Or use the End key, to get to the very end of the command string.

Remember the * to represent any number of characters, and . to represent just one unknown character.

Now we get to other useful tools in the Unix arsenal.

awk – a text processing, data extraction, programming language. Simple and effective.

sed – a stream editor which parses and transforms text. Older than, and superseded by, awk

grep – is a utility which **G**lobally searches a string with a **R**egular **E**xpression and **P**rints it

Unix commands are designed to be strung together. For instance:

```
sudo find / -name "*mp3" \
| grep "Music" \
| awk -F/ '{print $(NF-1), "\t", $NF}' \
> musicFiles.txt
```

There are four commands in that one long line, using the Unix tools I have mentioned. The '\' character is a line extension signifier. If you are typing in a terminal window this command string is entered as one long string without the '\'s. If read from a script, use the \ character to improve readability.

I will explain this script line by line. The leading sudo tells the operating system I am elevating myself to root privileges. sudo find ... tells the system to look for all the mp3 files I have on my entire network, with about 40 TB of files on six computers. This gets piped (the Unix tool |) to grep which is a regular expression tool. This uses the list from the find command, looking for those files with Music in their directory path. That gets piped into awk, which parses out the last and next to last 'columns'. Then this gets sent to the musicFiles.txt disk file. This command string is quite fast because it does not write to the display. I found all of the mp3s in the Music directory tree, grabbed the last two levels of the directory structure, and saved them to disk by typing this command string. All that work gave me a list to create my mp3 music database.

Common regular expressions

dot (.)	Match any character.
[ ]	Match a range of characters contained within the square brackets.
[^]	Match a character which is not one of those contained within the square brackets.
*	Match zero or more of the preceding item.
+	Match one or more of the preceding item.
?	Match zero or one of the preceding item.
{n}	Match exactly n of the preceding item.
{n,m}	Match between n and m of the preceding item.
{n,}	Match n or more of the preceding item.
\	Escape, or remove the special meaning of the next character.
String	A sequence of characters.

```
sed 's/regexp/replacement/g' inputFileName > outputFileName
```

s for substitute g for global

https://en.wikipedia.org/wiki/AWK

http://www.hcs.harvard.edu/~dholland/computers/awk.html

http://www.grymoire.com/Unix/Awk.html

https://www.tutorialspoint.com/awk/index.htm

https://en.wikipedia.org/wiki/Regular_expression

https://www.cheatography.com/davechild/cheat-sheets/regular-expressions/

http://www.zytrax.com/tech/web/regex.htm