Fgrep and Egrep

Written by Tyler Weiss 20 MAR 2024

Exercise 16

Task 1

Run the 'cat' command against 'password.txt'. The 'cat' command outputs the contents of a file to the console with no capability to navigate or search the file.

```
rec0nrat@demoserver1:~/RA-logs/Passwords$ cat passwords.txt
zzzzzzzzyzz
zzzzzzzzz
ZZZZZZZZZ
zzzzzzzzz1
zzzzzzzzzx
zzzzzzzzzz
ZZZZZZZZZZ
zzzzzzzzzzz
zzzzzzzzzzzz
zzzzzzzzzzzzz
zzzzzzzzzzzzzz
zzzzzzzzzzzzzzzzzzz
ZZZZZZZZZZZZZZZZZZZ
rec0nrat@demoserver1:~/RA-logs/Passwords$
```

Task 2

Examine the contents of 'passwords.txt' with the 'less' command. Using the less command allows you to navigate and search through the file. Navigation of the file can be done using the arrow keys, other shortcut keys such as 'g' or 'G', page down with space bar or search using the '/' followed by the word or search function. Using 'less' will start you at the beginning of the file. Note that there a many more search and navigation options than what was previously described. To exit 'less' type 'q'.

```
rec0nrat@demoserver1:~/RA-logs/Passwords$ less passwords.txt
```

```
#
# This is a list of 2,150,822 unique ASCII passwords in sorted order according
# to their native byte values using UNIX sort command.
#
# This list (also known as wordlist, password dictionary or password list)
# is useful for password recovery tools such as John the Ripper, hashcat and
# Aircrack-ng. To use this file, be sure to first remove these comment lines,
# i.e. the lines starting with # character.
#
# If you are looking for a better password dictionary,
# see https://dazzlepod.com/uniqpass/
#
# $DateTime: 2016/12/18 09:10:18 $
#
# Comments/Questions? Send to disclosure@dazzlepod.com
#
`
``
````
`````
``````
````````
`````````
`~
`!@##@!`
^
```

## Task 3

Run the command 'fgrep "123456" passwords.txt'. 'fgrep' or 'fixed grep', which is the same as 'grep -F', does not search using special characters in the regular expression but exact strings. The output of this command is any line that contains '123456' weather by itself or included in a string.

```
rec0nrat@demoserver1:~/RA-logs/Passwords$ fgrep "123456" passwords.txt
```

```
zzw123456
zzx123456
zzx123456789
zzxx123456
zzy123456
zzy123456789
zzz123456
zzz.123456
ZZZ123456
zzz123456789
zzzz123456
zzzzzz123456
rec0nrat@demoserver1:~/RA-logs/Passwords$
```

Exercise 17
```

Task 1

Using 'vim' add “1234567890_from_first_file” to the end of the file after the zzz not a new line. 'vim' can be navigated using the arrow keys. In order to enter "insert mode" enter 'i'. Once in "insert mode" the file can be edited. Press "escape" to exit "insert mode" and enter ':', type 'wq' and hit enter to save and exit. Use 'tail' to validate that the changes have been made to the end of the file.

```
rec0nrat@demoserver1:~/RA-logs/Passwords$ vim passwords.txt
```

```
zzzzzzzzzzzzz
zzzzzzzzzzzzzzzzzzz
ZZZZZZZZZZZZZZZZZZZZ1234567890_from_first_file
"passwords.txt" 2150838L, 20159803B
```

```
rec0nrat@demoserver1:~/RA-logs/Passwords$ tail passwords.txt
zzzzzzzzz1
zzzzzzzzzx
zzzzzzzzzz
ZZZZZZZZZZ
zzzzzzzzzzz
zzzzzzzzzzzz
zzzzzzzzzzzzz
zzzzzzzzzzzzzz
zzzzzzzzzzzzzzzzzzz
ZZZZZZZZZZZZZZZZZZZZ1234567890_from_first_file
rec0nrat@demoserver1:~/RA-logs/Passwords$
```

Task 2

Create a copy of the file 'password.txt' and name it 'passwords2.txt'. Edit the second file by changing the last line to “1234567890_from_second_file”. Use the 'cp' command to make a copy of the first file and store the copy as 'passwords2.txt'. Use 'vim' to edit and save the new file just described in task 1. Tail the file to show to validate the change.

```
rec0nrat@demoserver1:~/RA-logs/Passwords$ cp passwords.txt passwords2.txt
rec0nrat@demoserver1:~/RA-logs/Passwords$ ls
passwords2.txt  passwords.txt
rec0nrat@demoserver1:~/RA-logs/Passwords$ vim passwords2.txt
rec0nrat@demoserver1:~/RA-logs/Passwords$ tail passwords2.txt
zzzzzzzzz1
zzzzzzzzzx
zzzzzzzzzz
ZZZZZZZZZZ
zzzzzzzzzzz
zzzzzzzzzzzz
zzzzzzzzzzzzz
zzzzzzzzzzzzzz
zzzzzzzzzzzzzzzzzzzz
ZZZZZZZZZZZZZZZZZZZZ1234567890_from_second_file
rec0nrat@demoserver1:~/RA-logs/Passwords$
```

This is the view from within vim

```
zzzzzzzzzzz
zzzzzzzzzzzz
zzzzzzzzzzzzz
zzzzzzzzzzzzzzzzzzzz
ZZZZZZZZZZZZZZZZZZZZ1234567890_from_second_file
:wq
```

Task 3

Run the command 'fgrep 1234567890 *passw*'. Using the 'fgrep' will search for a string literal as previously described. Every instance of '123456789*' in a string will be output along with the name of the file where the string was found preceding it. Multiple files will be searched due to 'passw' being used with wildcard '*' as the file argument.

```
rec0nrat@demoserver1:~/RA-logs/Passwords$ fgrep 123456789_ passw*
passwords2.txt:123456789_
passwords2.txt:123456789_abc
passwords2.txt:123456789_f
passwords.txt:123456789_
passwords.txt:123456789_abc
passwords.txt:123456789_f
rec0nrat@demoserver1:~/RA-logs/Passwords$
```

Exercise 18

Task 1-2

Run the command 'fgrep 1234567890_ passwords.txt > passwords3.txt'. As mentioned before 'fgrep' will search for a string literal and output lines containing that string. Since only a single file is being searched the output will only contain the strings from the file and not the file name. Redirecting the output to 'passwords3.txt' will cause no output to be printed to the console. By using 'tail' we can print the last ten lines of 'password3.txt' thus confirming it's contents.

```
rec0nrat@demoserver1:~/RA-logs/Passwords$ fgrep 1234567890_ passwords.txt > passwords3.txt
rec0nrat@demoserver1:~/RA-logs/Passwords$ tail passwords3.txt
1234567890_
ZZZZZZZZZZZZZZZZZZZZ1234567890_from_first_file
```

Task 3

Run the command 'fgrep -c 1234567890_ passw*'. The '-c' switch for 'fgrep' stand for count and using 'passw*' will search each text file starting with that string. The output will contain the file name followed by the number of matches, or count, within files.

```
rec0nrat@demoserver1:~/RA-logs/Passwords$ fgrep -c 1234567890_ passw*
passwords2.txt:2
passwords3.txt:2
passwords.txt:2
```

Task 5

Run the command 'fgrep -w 1234567890_ passw*'. The switch '-w' stands for 'word regex' will will cause 'fgrep' to match only whole words. As a side note, the information regarding different switches and options can be found by typing 'grep --help'. The output will contain the file name followed by the lines containing only the word being searched.

```
rec0nrat@demoserver1:~/RA-logs/Passwords$ fgrep -w 1234567890_ passw*
passwords2.txt:1234567890_
passwords3.txt:1234567890_
passwords.txt:1234567890_
rec0nrat@demoserver1:~/RA-logs/Passwords$
```

Task 6

Run the command 'fgrep -n 1234567890_ passw*'. The switch '-n' stands for 'line number' and will output the line number where a match is found. The output will contain the file

name, the line number of the match and the matching line itself.

```
rec0nrat@demoserver1:~/RA-logs/Passwords$ fgrep -n 1234567890_ passw*
passwords2.txt:126428:1234567890_
passwords2.txt:2150838:ZZZZZZZZZZZZZZZZZZZZ1234567890_from_second_file
passwords3.txt:1:1234567890_
passwords3.txt:2:ZZZZZZZZZZZZZZZZZZZZ1234567890_from_first_file
passwords.txt:126428:1234567890_
passwords.txt:2150838:ZZZZZZZZZZZZZZZZZZZZ1234567890_from_first_file
rec0nrat@demoserver1:~/RA-logs/Passwords$
```

Exercise 19

Task 1

Run the command 'grep -E "[0-9]{10}_from" passwords.txt'. The '-E' switch tells 'grep' to search using 'extended regex' which the use of more features and negates the need to escape certain characters. The regex in the command uses a class '[0-9]' with range of '{10}' to search for 10 numbers preceding the string literal '_from'. The resulting output will be any line that contains a regex match within it.

```
rec0nrat@demoserver1:~/RA-logs/Passwords$ grep -E "[0-9]{10}_from" passwords.txt
ZZZZZZZZZZZZZZZZZZZZ1234567890_from_first_file
rec0nrat@demoserver1:~/RA-logs/Passwords$
```

Task 2

Run the command 'grep -E "[0-9]{10}_from" passwords*.txt'. By using the wild card character '*' within the file name the 'grep' command will search all password files. Since the regex is the same as the previous task the output should be same except it will also be preceded by the filename of where the match was found.

```
rec0nrat@demoserver1:~/RA-logs/Passwords$ grep -E "[0-9]{10}_from" passwords*.txt
passwords2.txt:ZZZZZZZZZZZZZZZZZZZZ1234567890_from_second_file
passwords3.txt:ZZZZZZZZZZZZZZZZZZZZ1234567890_from_first_file
passwords.txt:ZZZZZZZZZZZZZZZZZZZZ1234567890_from_first_file
rec0nrat@demoserver1:~/RA-logs/Passwords$
```

Exercise 20

Task 1-3

Using authy.log determine why there was a service interruption on our website.
Using authy.log determine by who was this done?
Using authy.log determine from where was this done from?

The website is run by a using a service. That service can be interrupted using the 'systemctl' command followed by an action (i.e. stop, start, restart, shutdown). Knowing that the service that is interrupted was for a website we may also assume that 'apache2' may be involved. If we grep the 'authy.log' file for any of these key words or commands we can narrow our search. The grep command used to in this example is 'grep systemctl authy.log -n' which will search for any use of the command 'systemctl' and print the line number followed by any matching entries. This produces a single result.
The output show that the command 'systemctl shutdown apache2' was issued by the user 'tara' using 'sudo' permissions. It also show where the command was issued from within the file system because 'PWD', which stands for 'present working directory' is equal to '/home/tara'.