

Linux Administration with sed and awk

Introduction to Regular Expressions
and Filtering Text



Andrew Mallett

Author and Trainer

@theurbanpenguin | www.theurbanpenguin.com



**Master the Linux Command
Line unlocking the power of
text file manipulation using
sed and awk!**



Overview



The Course

- Using the tools: grep, sed and awk
- Mastering sed
- Working with awk
 - YAML and XML data
 - Log files

This Module

- Using grep to practice regular expressions
- Work with simple text file formats





Tools to Discover

grep:

- Search and filter on lines in a text file
- Horizontal filtering
- **g**lobal / **r**egular expression / **p**rint

sed:

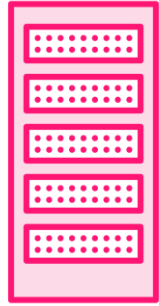
- The **s**tream **e**ditor allowing us to programmatically edit file

awk:

- Powerful reporting tool, similar to grep but way more powerful
- Alfred **A**ho, Peter **W**einberger, and Brian **K**ernighan



The Lab Environment: You'll Need



A single Linux system with root access



Tools are similar across Linux, so the distribution does not matter



We use a Raspberry Pi running Raspbian GNU/Linux 10 (buster)





Searching Using grep



```
$ grep 'root' /etc/passwd # search for string root
$ grep '^root' /etc/passwd # search for string root at start of the line
$ grep 'bash$' /etc/passwd # search for string bash at end of the line
$ grep -c 'bash$' /etc/passwd # count the lines with bash at the end of the line
$ grep -c '/tcp' /etc/services # count the TCP protocols in the /etc/services file
```

Basic Searches Using grep

We can search files with grep using a regular expression. This may be pure text or text with metacharacters, such as **^** for the start of a line and **\$** for the end of a line. We can also add options such as **-c** to just count the lines rather than print them



Demo



We will start to discover grep and regular expressions

Starting with easy-to-understand examples

- We can become used to grep and more importantly regular expressions to give you confidence in the tools
- Searching the `/etc/passwd`, which we all have, for simple text adding meta-characters for the start and end of line




```
$ grep -c ' ' /etc/services # count the lines in the file
$ grep -E '/(udp|tcp)' # print services that are either TCP or UDP requires -E
$ grep -vE '/(udp|tcp)' # reverse the search, we now see lines without /tcp or /udp
$ grep -E '/(sctp|ddp)' # having located additional protocols we can list them
```

Digging Deeper with grep

We can count lines with grep, just use a blank regular expression. Using -E we can use an enhanced regular expression which allows grouping with parenthesis. We can now search for either TCP or UDP protocols. Furthermore, we can inverse the search to show lines NOT containing tcp or udp



```
$ grep -c ' ' /etc/ssh/ssh_config # count the lines in the file  
$ grep -vE '^(#|$)' /etc/ssh/ssh_config # print uncommented lines with content
```

Remove Comments and Empty Lines

Sometimes, it is is useful to print just effective lines without comments and empty lines



Demo



Moving on from Basic Regular Expressions (BRE) we can look at Enhanced Regular Expressions (ERE) and grouping

Using Groups

- The option -E can be used to invoke the ERE allowing for groups
- The option -v allows us to invert an expression easily allowing us to drop commented and empty lines as an example
- These basic skills allow you much more control in filtering text





Search Many Files




```
$ grep 'pam_motd' /etc/pam.d/*  
$ grep -l 'pam_motd' /etc/pam.d/*  
$ man pam_motd
```

File Globbing

Using the standard globbing methods in the shell it is easy to list matches in many files. Implementing the option `-l` we can list filenames only rather than matching names



Demo



Needing to locate where a configuration is made can be useful when many files may be used for the same configuration

Grep with File Globbing

- Here we search for which files hold the relevant configuration
- Knowing which files have the configuration shows how we need to edit the files
- We need to show a different Message of the Day file for SSH connections





Processing Text Files



```
$ echo "Fred 23 Sales" >> employees.txt
$ echo "Rahinda 22 IT" >> employees.txt
while read name age dept; do
echo $name
echo $age
echo $dept
done < employees.txt
```

IFS

The default internal field separator (IFS) is white space. With a simple space separated file, we can use the read command to populate the three fields used




```
$ tr ' ' ',' < employees.txt > employees.csv
$ OLDIFS="$IFS"
$ IFS=','
$ while read name age dept; do
echo $name
echo $age
echo $dept
done < employees.csv
$ IFS="$OLDIFS"
```

CSV Files

A common data format is CSV, text files with comma separated values. Configuring the IFS variable with a comma can help. The tr command can quick change the spaces to commas so we do not need to worry too much about editing the file



```
$ OLDIFS="$IFS"
$ IFS=':'
$ while read n p u g c d s ; do
> echo -e "User: $n\n\tPass: $p\n\tUID: $u\n\tGID: $g\n\tComment: $c\n\tHOME:
$d\n\tSHELL: $s"
> done < /etc/passwd
$ IFS="$OLDIFS"
```

/etc/passwd

The /etc/passwd file is separated using the colon (:). We can read the 7 values into 7 variables, here kept short for space reasons. We can use single echo command with embedded control characters to indent entries under the username



Demo



Even without using grep we can process text files

IFS

- Becoming used to working with the IFS helps us process different data formats
 - space separated
 - comma separated
 - colon separated



Summary



The Course

- Using the tools: grep, sed and awk

This Module

- Using grep to practice regular expressions
 - ^ lines begin with
 - \$ lines end with
- Processing text files and the IFS



Up Next:

The Stream Editor

