CSI6203 Scripting Languages

Module 7

Stream Editor (SED) and Text Manipulation

Contents

- Introducing the stream editor and simple automated text editing
- Text parsing and dealing with data files
- Text substitution
- Transforming text

Reference link: https://www.cyberciti.biz/faq/how-to-use-sed-to-find-and-replace-text-in-files-in-linux-unix-shell/

Learning Objectives

After finishing this module, you should be able to:

- Understand and execute scripts that make use of sed to perform simple search and update operations
- Write scripts that can parse text in a meaningful way

Text formatting and parsing

Text parsing and formatting

- Many scripts involve manipulating text in some way.
- Formatting
  - Arranging information into meaningful text
- Parsing
  - Reading useful information out of text

Text parsing and formatting

- Grep can be used to help with parsing

```
GREP(1)                   BSD General Commands Manual                  GREP(1)

NAME
     grep, egrep, fgrep, zgrep, zegrep, zfgrep -- file pattern searcher

SYNOPSIS
     grep [-abcdDEFGHhIiJLlmnOopqRSsUVvwxZ] [-A num] [-B num] [-C[num]] [-e pattern]
          [-f file] [--binary-files=value] [--color[=when]] [--colour[=when]]
          [--context[=num]] [--label] [--line-buffered] [--null] [pattern] [file ...]

DESCRIPTION
     The grep utility searches any given input files, selecting lines that match one or
     more patterns.  By default, a pattern matches an input line if the regular expression
     (RE) in the pattern matches the input line without its trailing newline.  An empty
     expression matches every line.  Each input line that matches at least one of the pat-
     terns is written to the standard output.

     grep is used for simple patterns and basic regular expressions (BREs); egrep can han-
     dle extended regular expressions (EREs).  See re_format(7) for more information on
     regular expressions.  fgrep is quicker than both grep and egrep, but can only handle
:
```

Output

```
#!/bin/bash
#get info about networking from the ifconfig command
net_info="$(ifconfig)"

#parse out the ip address lines
addresses="$(echo "$net_info" | grep inet)"

#format output
echo -e "IP addresses on this computer are:\n$addresses"
```

```
IP addresses on this computer are:
        inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
        inet6 fe80::821:70ff:dada:497%en0 prefixlen 64 secured scopeid 0x9
        inet 10.200.36.96 netmask 0xfffff000 broadcast 10.200.47.255
        inet6 fe80::98eb:cff:feed:feaa%awdl0 prefixlen 64 scopeid 0xb
        inet6 fe80::8cc3:97cb:7752:56b4%utun0 prefixlen 64 scopeid 0x11
        inet6 fe80::aede:48ff:fe00:1122%en6 prefixlen 64 scopeid 0x8
```

Text parsing and formatting

- Some commands have the ability to parse and format built into them. - eg Dates

```
#!/bin/bash

# Link: https://www.cyberciti.biz/faq/unix-linux-getting-current-date-in-bash-ksh-shell-script/

TEST_DATE1=$(date +'%A %d %T')
TEST_DATE2=$(date +'%B %d %Y')

#echo "**** Testing dates ****"

### mm/dd/yyyy ###
date +'%m/%d/%Y'

## Time in 12 hr format ###
date +'%r'

## Test Date 1 and 2
echo "Date1 ${TEST_DATE1}"
echo "Date2 ${TEST_DATE2}"

## backup dir format ##
backup_dir=$(date +'%m/%d/%Y')
echo "Backup dir for today: ~/CSI6203/CSI6203/portfolio/week7/${backup_dir}"
```

Date Example Output

```
MHD81805703:samples leisaa$ bash datesexample.sh
*****Testing Dates###
09/10/2019
12:42:37 pm
Date1 Tuesday 10 12:42:37
Date2 September 10 2019
backup directory for today is ~/09/10/2019
MHD81805703:samples leisaa$
```

Text parsing and formatting

- Check the man pages for formatting codes before trying to parse text manually (`man date`) or date --help in shell

```
DATE(1)

NAME
       date - print or set the system date and time

SYNOPSIS
       date [OPTION]... [+FORMAT]
       date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]

FORMAT controls the output.  Interpreted sequences are:

       %%     a literal %

       %a     locale's abbreviated weekday name (e.g., Sun)

       %A     locale's full weekday name (e.g., Sunday)……
```

sed Basics

sed

- The Stream Editor (sed) can be used to simplify the parsing and formatting process

- sed works by searching through text line by line making changes based on given commands

```
sed 'command' [Input file]
```

Displaying text with sed

- The ‘p’ command can be used with sed to print specific contents of files

```
sed -n 'p' /etc/passwd
```

- This doesn’t seem very useful as it is essentially recreating the `cat` command
- Adding a pattern to the command allows it to perform searches like `grep`:

```
sed -n '/rob/ p' /etc/passwd
```

sed on specific lines

- sed can also be restricted to only process specific lines by adding the line numbers to the command:

```
sed -n '1,3 p' /etc/passwd
```

(only show the first to third lines of the passwd file)

```
student@csi6203:~/CSI6203/CSI6203/portfolio/week7$ sed -n '1,3 p' /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```

sed Substitutions

The substitute command (1)

- One of the most common features of sed is to act as a “find and replace” operation.
- The command to replace a found pattern with another is “s” for “substitute”

```
sed -n 's/spaghetti/gnocchi/g' foods.txt
```

(searches foods.txt for a line containing the word “spaghetti” and replace with “gnocchi”

The substitute command (2)

- As per last slide
- Substitutes the text spaghetti with gnocchi and copies the modified content into new_foods.txt file.

```
student@csi6203:~/CSI6203/CSI6203/portfolio/week7$ cat foods.txt
spaghetti
corba
grah
rice
noodles
chicken soup
pizza
student@csi6203:~/CSI6203/CSI6203/portfolio/week7$ sed 's/spaghetti/gnocchi/g' foods.txt > new_foods.txt
student@csi6203:~/CSI6203/CSI6203/portfolio/week7$ cat new_foods.txt
gnocchi
corba
grah
rice
noodles
chicken soup
pizza
```

The substitute command

- Substitutions are very powerful but by default will not edit the file. Instead, the edited file contents will be printed to stdout
- This allows sed to be used in scripts without editing files
- The “-i” flag can be used to save the edited data back to the file

```
sed -i 's/big/small/' input.txt
```

(replace “big” for “small” in input.txt and save the changes)

The substitute command

- By default, substitutions are limited to one per line.
- Adding “g” to a substitution will apply the change globally and replace every occurrence on the line

```
sed 's/big/small/g' input.txt
```

(replace every occurrence of “big” for “small” in input.txt)

The substitute command

- Adding a number to a substitution will apply the change that number of times for each line

```
sed 's/big/small/2' input.txt
```

(Replace only the first two occurrences an each line of “big” for “small” in input.txt)

sed Insertion and Deletion

Inserting data

- sed can also be used to add extra lines to the text using the “i” and “a” commands

```
sed '5i\toast' input.txt
```

(insert a line with the word “toast” before the 5th line)

```
sed '5a\toast' input.txt
```

(add a line with the word “toast” after the 5th line)

- Be careful of “\” vs “/”

Inserting data

- Inserting can also be done matching a pattern:

```
sed '/coffee/ i\tea' input.txt
```

(insert a line with the word “tea”
before the first line with the word “coffee”)

Deleting data

- sed can also be used to delete lines from the text using the “d” command

```
sed '5d' input.txt
```

(delete the 5th line)

```
sed ‘3,5d' input.txt
```

(delete everything from the 3rd to the 5th line)

Deleting data

- Deleting can also be done matching a pattern:

```
sed '/coffee/ d' input.txt
```

(delete the first line with the word “coffee”)

sed Change and Transform

Change

- The “Change” command is similar to the “substitute” command but will replace an entire line

- This can be useful for overwriting lines with new content without having to carefully match them

Change

- Changing syntax is similar to deleting or inserting:

```
sed '2c\New second line' input.txt
```

(change the entire 2nd line to the new text)

```
sed '/coffee/ c\tea is good' input.txt
```

(change the first line that has the word “coffee” in it to instead read “tea is good”)

Transform

- The “Transform” command (“y”) is similar to the “substitute” command but will replace any character listed throughout the line

- This can be useful for modifying specific characters or changing the case of a set of characters

Change

- Transform syntax is similar to substituting:

```
sed 'y/abc/ABC/' input.txt
```

(change any letter a, b or c to be uppercase)

```
Input:
all cats are cool
Output:
All CAts Are Cool
```

Multiple commands and actions

Multiple commands

- Sed commands can be chained together to produce complex text parsing
- This can be done by separating them with semicolons and using the –e option

```
sed -e 's/spaghetti/gnocchi/g; s/big/small/2' foods.txt
```

(Replace all “spaghetti” for “gnocchi” and the first 2 “big” to “small” on each line)

Multiple commands

- Curly braces can also be used to apply multiple commands to a single match.

```
sed '/favorite/ {
s/spaghetti/gnocchi/g
s/big/small/2
}' food.txt
```

(Replace all “spaghetti” for “gnocchi” and the first 2 “big” to “small” on each line that contains the word “favorite”)

Putting it all together

- Instead of input coming from a file, sed input can be piped from other commands or scripts using the '|' command.

- Lets use sed to list all the ipv4 addresses from the ipconfig command like we did with grep.

Putting it all together

```
#!/bin/bash
#get info about networking from the ifconfig command
net_info="$(ifconfig)"

#parse out the ip address lines using sed
addresses=$(echo "$net_info" | sed -n '/inet / {
s/inet/IP Address:/
s/netmask/\n\t\tSubnet Mask:/
s/broadcast/\n\t\tBroadcast Address:/
p
}')

#format output
echo -e "IP addresses on this computer
are:\n$addresses"
```

Putting it all together

- Output:

```
IP addresses on this computer are:
        IP Address: 192.168.184.1
                Subnet Mask: 255.255.255.0
                Broadcast Address: 192.168.184.255
        IP Address: 192.168.141.1
                Subnet Mask: 255.255.255.0
                Broadcast Address: 192.168.141.255
        IP Address: 127.0.0.1
                Subnet Mask: 255.0.0.0
        IP Address: 10.1.1.242
                Subnet Mask: 255.255.255.0
                Broadcast Address: 10.1.1.255
```

Summary

- Terms to review and know include:
  - Sed
  - Substitute
  - Change
  - Transform
  - Format
  - Parse

References and Further Reading

- Ebrahim, M. and Mallet, A. (2018) Mastering Linux Based Scripting (2nd Ed) Chapter 8, pp 141-160
- http://www.gnu.org/software/sed/manual/sed.html
- http://linux.die.net/man/1/sed