SECTION 22

Introduction to SED (String editor) command:

Sed command stands for stream editor.

Sed command perform lot’s of function on file like

* Viewing file content
* Searching
* Find and replace
* Insertion or deletion

Sed also supports regular expression which allows it perform complex pattern matching.

**Advantage of sed over vi/vim editors**:

* Edit files without opening it.

***Syntax:***

Sed [options] commands [file -to-work-with sed]

How sed works?

Read a line by input instream

Execute sed commands on a line

Display result on output stream

Sed: Viewing file content and deleting file content based on line (**original file won’t change**)

sed ' ' filename ===>sed without any options and commands

sed ' p' filename ====>it prints each line.

sed -n 'p' filename ===> to suppress the default output

sed -n '3p' filename ===>it prints 3rd line

sed -n '6p' filename ===>it prints 6th line

sed -n '$p' filename ===>it prints last line

sed -n '3,10p' filename ===>it prints 3rd line to 10th line

sed -n '3,$p' filename ===>it prints 3rd line to last line

sed -n '3,+5p' filename ===>it prints 3rd line and adding next 5 more lines

sed -n '1~3p' filename ===>it prints 1,4,7,10,………lines etc

sed '9,11d' filename ===>deleting from 9th line to 11th line. (don’t use -n.we should use -n only when print used )

sed '9,11!d' filename ==>otherthan this range will delete all lines

sed '9,+5d' filename ===>it deletes 9th line and deleting next 5 more lines

**original file will change**

sed -i '29,$d' one.txt ====>it deletes from 29th line to last line .

sed -i.back ‘3,$d’ one.txt ======> it will take backup as well as deletes from 3rd line to last line

**one.txt.back** ===>To check the back up file .

**Searching file content :**

sed -n '/pavan/p' filename ===>it prints the line which contain pavan word.

sed -n -e '/pavan/p' -e'/kumar/p' filename ===>for multiple words.

***To search the word which is defined in a variable (Must use double quotes)***

A=”pavan”

Sed -n “/$A/p” file .txt

**Deleting the lines which are not having the particular word**

sed '/pavan/!d' one.txt ===>it will delete the lines which not having pavan word.

sed -i '/pavan/!d' one.txt ===>it will delete the lines which not having pavan word

from original file.

sed -i.back '/pavan/!d' one.txt ===>it takes backup file as well as it will delete the lines

which not having pavan word from original file.

**one.txt.back** ===>To check the back up file .

**Find and Replace with sed command :**

sed ‘s/old/new/’ one.txt ===>the first occurred old word will be replacing with new word

S=substitutute

sed ‘s/old/new/2’ one.txt ===>replaces second occurred old word with new word.

sed ‘s/old/new/g’ one.txt ===>replace all old words with new words on each and every

line.

g=globally.

sed -i.backup ‘s/old/new/g’ one.txt ===>replace all old words with new words on each and

every line in original file and it tooks backup of original file.

g=globally.

sed ‘/searchword/s/old/new/’ one.txt ==> replaces the old word with new word .if that line

consists of search word.

sed ‘/searchword/s/old/new/g’ one.txt ==> replaces the old word with new word globally.

if that line consists of search word.

**Insertion and deletion with sed command :**

i= insert a new line before a particular line number.

a=insert a new line after a particular line number.

d= to delete the line

Ex: Based on line number

sed ‘1i string1 string2’ filename ===>it will insert a new line before 1st line number with

info.

sed ‘2a string1 string2’ filename ===>it will insert a new line after 2nd line number with

info.

sed -i ‘1i string1 string2’ filename ===>it will insert a new line before 1st line number with

some info in original file.

sed -i ‘2a string1 string2’ filename ===>it will insert a new line after 2nd line number

with some info in original file.

sed -i ‘$a string1 string2’ filename ===>it will insert a new line at last line number

with some info in original file.

sed -i ‘3d’ filename ===>it will delete the 3rd line from file.

Ex: Based on search word

sed -i ‘/pavan/i string1 string2 ’ filename ==>it will search with pavan word .then it will insert

a new line before pavan word line with some

info in original file.

sed -i ‘/pritham/a string1 string2 ’ filename ==>it will search with pritham word .then it will

insert a new line after pritham word line with

some info in original file

sed -i ‘/anil/d’ filename ===>it will delete the line which consists of anil word.

**Regex: Introduction to sed command with regex :**

What is regex ?

Regex is the shortcut for regular expression.

If any expression uses a pattern then that expression is called regex.

What is pattern ?

Pattern is a string which represents more than one word

EX:

sed -n ‘/p[uo]t/p’ filename ===>it prints put/pot lines.

Here p[uo]t = put/pot (represents two words)

**Regex-Part-1: Regex with special charecters**

**\s Matches for space.**

**\ Escape charecter**

**\t Matches for tab**

**. Matches any character, excluding newline.**

**\* Matches a sequence of zero or more instances of matches for the**

**preceding regular expression**

**\+ As \*, but matches one or more**

**\? As \*, but only matches zero or one**

Note: \s is the subset of \t . i.e space is subset of tab.

sed -n ‘/\s/p’ one.txt ===>it prints the lines which having spaces.

sed -n ‘/\\s/p’ one.txt ===>it prints the lines which having \s character.

sed -n ‘/\t/p’ one.txt ====>it prints the lines which having tab space

sed -n ‘/p.t/p’ one.txt ===>it prints the lines which the word starts with p and ends with t

i.e 3 word letters .pit,put,pot,pkt …..etc

sed -n ‘/\sp.t/p’ one.txt ===>it prints the lines which having space and after which the word

starts with p and ends with t i.e 3 word letters .pit,put,pot,pkt…etc

sed -n ‘/\sp.t\s/p’ one.txt ===>it prints the lines which having space before and after which

the word starts with p and ends with t i.e 3 word letters

.pit,put,pot,pkt…etc

sed -n ‘/\./p’ one.txt ===>it prints the lines which having only . character

sed -n ‘/This\*/p’ one.txt ====>it prints This word s is optional

This\* ===>Thi This Thisss Thissssss

sed -n ‘/This\+/p’ one.txt ====>it prints This word s atleast one time or more

This\+ ===>This Thiss Thisss Thissssss

sed -n ‘/This\?\s/p’ one.txt ====>it prints This word s atleast one time or zero time

This\? ===>Thi This

**Regex-Part-2: Regex with special charecters (^ and $)**

sed -n '/^pavan/p' filename ===>it prints the lines which contain pavan word at very first

in a line

sed -n '/pavan$/p' filename ===>it prints the lines which contain pavan word at ending

in a line

sed -n '/^$/p' filename ===>it prints empty lines i.e start with nothing and end with

nothing.

sed '/^$/d' filename ===>it will deletes the lines which are having empty .

sed -i '/^$/d' filename ===>it will deletes the lines which are having empty in original file

sed -n '/\^/p' filename ==> To print the lines which are having charet symbol from the file.

sed -n '/\$/p' filename ==> To print the lines which are having dollar symbol from the file.

**Regex-Part-3: Regex with special charecters [] {} and ()**

[ ] ==> Matches atleast any single character in list.

sed -n ‘/p[uoy]t/p’ filename ===>it prints put/pot/pyt lines. ***It won’t consider order.***

sed -n ‘/p[a-o]t/p’ filename =>The character may be a/b/c/d/e/f/g/h/i/j/k/l/m/n/o ***sequence***

sed -n ‘/p[a-cm-ox-z]t/p’ filename

The character may be a/b/c/m/n/o/x/y/z

{ } ==> Matches for required number of repetations

* \{i\}
* \{I,j\}
* \{I,\}

sed -n ‘/This\{3\}/p’ filename =>if the s repeated 3 times then it print the line.

(3 or morethan 3)

Ex: Thisss orThissss or Thisssssssss

sed -n ‘/This\{3\}\b/p’ filename =>if the s repeated 3 times then it print the line. (exactly)

Ex: Thisss

sed -n ‘/This\{3,4\}\b/p’ filename =>if the s repeated 3 or 4 times then it print the line. (exactly)

Ex: Thisss or Thissss

sed -n ‘/This\{3,\}\b/p’ filename =>if the s repeated 3 or 4 times then it print the line. (exactly)

Ex: Thisss or Thissss

( ) ==>This will search for zero or more whole sequence

put\+ ====> look for only put string

\(put\)\+ ====> look for only put or putput or putputput……

asdf entire word atleast two times i.e same pattern should repeate

sed -n ‘/asdf\{2\}/p’ filename =>if the f repeated 3 times then it print the line.

(2 or morethan 2)

Ex: asdfff or asdffffff or asdffffffffff

sed -n ‘/\(asdf\)\{2\}/p’ filename =>grouping the asdf word i.e \(\)

(2 or morethan 2)

Ex:khdfkjjnkjasdfasdfoaieurpfh

sed -n ‘/\(asdf\)\+/p’ filename =>grouping the asdf word i.e \(\)

(1 or morethan 1)

Ex:khdfkjjnkjasdfoaieurpfh

khdfkjjnkjasdfasdfoaieurpfh

sed -n ‘/\b\(asdf\)\+/p’ filename =>grouping the asdf word i.e \(\)

(1 or morethan 1)

Ex: This is asdf

adflkdjhlnnkfk

This is a pitasdf xxxxxxxxxx it won’t print this type

Khdfkjjnkjasdfasdfoaieurpf

**One Video to understand the usage of sed cut awk and arrays an example of**:

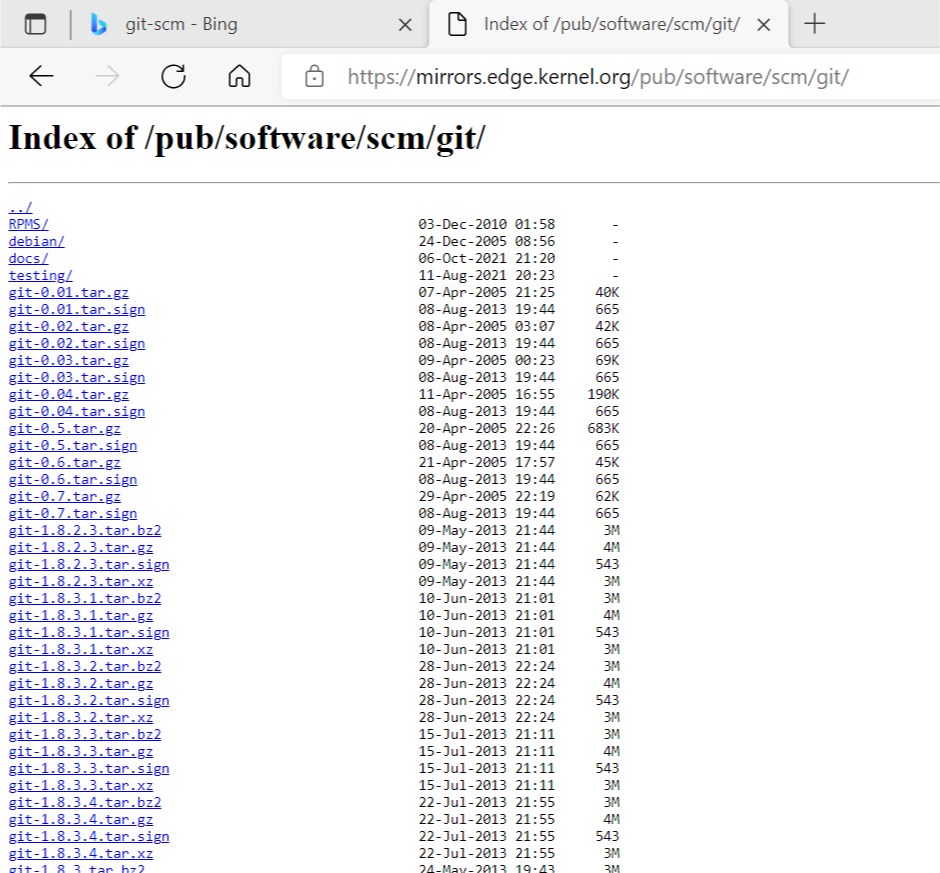
**Simple shell script to get all git versions from official git-scm website.**

Search in google with:

Git-scm ==> Downloads ==> Linux/Unix==> Download a tarball

curl [Index of /pub/software/scm/git/ (kernel.org)](https://mirrors.edge.kernel.org/pub/software/scm/git/) ===> you will get index.html page

wget [Index of /pub/software/scm/git/ (kernel.org)](https://mirrors.edge.kernel.org/pub/software/scm/git/) ===> it will download the index.html page



***get\_git\_versions.sh***

!/bin/bash

# Author : PAVAN

which wget 1>/dev/null 2>/dev/null

if [ $? -ne 0 ]

then

echo "Please install wget and retry"

exit 1

fi

if [ -e "index.html" ]

then

echo "Removing old index.html"

rm -rf index.html

fi

url="https://mirrors.edge.kernel.org/pub/software/scm/git/"

wget $url

if [ $? -ne 0 ]

then

echo "unable to download git from $url"

exit2

fi

while read A ==>it will read line one by one from file.

do

echo $A | sed -n '/git-\([0-9]\+\.\)\+tar.gz/p' |awk -F '"' '{ print $2 }' |cut -c 5- |awk -F '.tar.gz' '{ print$1 }'

#sleep 1

done < index.html

***Explanation:***

if [ -e "index.html" ] ====> if the file exist

sed -n ‘/git-\([0-9]\+\.\)\+tar.gz/p’

print the line which having git-some number That number maybe from 0-9 and \+ indicates that it may have one or morethan one time i.e 1 or 12 or 123 or 1234 etc and \. Matching with .

/git-[0-9] ===> git-1

/git-[0-9]\+ ==>git-16 (means two digits)

/git-[0-9]\+\. ==>git-16. (matching with . and this is only for the first digit)

your version may repeate . so we need to group. The purpose of grouping concept is if any sequence is repeated to identify that we have to group your particular pattern

/git-\([0-9]\+\.\)\+

==>git-13 or git-13.2 or git-14.2.34 or git 2.45.6.4

This sequence may be one or more times i.e

***Print the o/p in the form of array and in each line it should be only three versions***

***Declaring the variable and assigning the variable in an array***

#!/bin/bash

# Author : PAVAN

which wget 1>/dev/null 2>/dev/null

if [ $? -ne 0 ]

then

echo "Please install wget and retry"

exit 1

fi

if [ -e "index.html" ]

then

echo "Removing old index.html"

rm -rf index.html

fi

url="https://mirrors.edge.kernel.org/pub/software/scm/git/"

wget $url

if [ $? -ne 0 ]

then

echo "unable to download git from $url"

exit 2

fi

**declare -a git\_vers**

while read linebyline

do

**git\_vers+**=($(echo $linebyline | sed -n '/git-\([0-9]\+\.\)\+tar.gz/p' |awk -F '"' '{ print $2 }' |cut -c 5- |awk -F '.tar.gz' '{ print $1 }'))

#sleep 1

done < index.html

echo "The all available git versions are: "

**echo "${git\_vers[\*]}"**

**Draw back :**

We will get unnecessary spaces or empty lines. 106 lines getting empty lines because we will print 3 versions in each line. So it occupies only 53 lines.

***Print the o/p in the form of array and in each line it should be only three versions***

!/bin/bash

# Author : PAVAN

which wget 1>/dev/null 2>/dev/null

if [ $? -ne 0 ]

then

echo "Please install wget and retry"

exit 1

fi

if [ -e "index.html" ]

then

echo "Removing old index.html"

rm -rf index.html

fi

url="https://mirrors.edge.kernel.org/pub/software/scm/git/"

wget $url

if [ $? -ne 0 ]

then

echo "unable to download git from $url"

exit 2

fi

declare -a git\_vers

echo "\*\*\*\*\*\*\*\*\*\*\* Please wait collecting all git versions from officieal webesite of git-scm \*\*\*\*\*\*\*\*\*\*\*\*"

while read linebyline

do

git\_vers+=($(echo $linebyline | sed -n '/git-\([0-9]\+\.\)\+tar.gz/p' |awk -F '"' '{ print $2 }' |cut -c 5- |awk -F '.tar.gz' '{ print $1 }'))

#sleep 1

done < index.html

echo "The all available git versions are: "

cnt=0

no\_vers=${#git\_vers[\*]} 🡺to get the length of the git versions

for each\_var in ${git\_vers[\*]}

do

echo -e "\t\t ${git\_vers[$cnt]} \\t ${git\_vers[$((cnt+1))]} \\t ${git\_vers[$((cnt+2))]}"

cnt=$((cnt+3))

if [ $cnt -ge $no\_vers ] 🡺This if condition is used to avoid the spaces or empty lines

then break

fi

done

**Draw back :**

It won’t arranged properly

**example:**

- 🡺to print from left to right

\* 🡺to print width size

\n 🡺 By default printf command won’t go to next line

To execute \t we need to use -e

For suppose we have 159 versions- i.e

no\_versions =159

by using for loop we are printing 3 values each time it means this will completed 159/3=53 lines.

But for loop will run 159 times ,so then we are getting empty spaces or lines because we don’t have index range for that . 159-53 =106 lines getting empty lines . to avoid that

***Print the o/p in the form of array and in each line it should be only three versions***

!/bin/bash

# Author : PAVAN

which wget 1>/dev/null 2>/dev/null

if [ $? -ne 0 ]

then

echo "Please install wget and retry"

exit 1

fi

if [ -e "index.html" ]

then

echo "Removing old index.html"

rm -rf index.html

fi

url="https://mirrors.edge.kernel.org/pub/software/scm/git/"

wget $url

if [ $? -ne 0 ]

then

echo "unable to download git from $url"

exit 2

fi

declare -a git\_vers

echo "\*\*\*\*\*\*\*\*\*\*\* Please wait collecting all git versions from officieal webesite of git-scm \*\*\*\*\*\*\*\*\*\*\*\*"

while read linebyline

do

git\_vers+=($(echo $linebyline | sed -n '/git-\([0-9]\+\.\)\+tar.gz/p' |awk -F '"' '{ print $2 }' |cut -c 5- |awk -F '.tar.gz' '{ print $1 }'))

#sleep 1

done < index.html

echo "The all available git versions are: "

cnt=0

no\_vers=${#git\_vers[\*]} 🡺to get the length of the git versions

width=20

for each\_var in ${git\_vers[\*]}

do

printf “%-\*s %-\*s %-\*s\n” $width ${git\_vers[$cnt]} $width ${git\_vers[$((cnt+1))]} $width ${git\_vers[$((cnt+2))]}

cnt=$((cnt+3))

if [ $cnt -ge $no\_vers ] 🡺This if condition is used to avoid the spaces or empty lines

then break

fi

done

**Shell script to get all git versions using functions concept**

#Please complete function concept first and then come back here

#!/bin/bash

#Author: Narendra

#Version: 1.0

COLUMNS=$(tput cols)

prRed(){

  echo -e "\033[91m$1 \033[00m"

}

prGreen(){

  echo -e "\033[92m$1 \033[00m"

}

prYellow(){

  echo -e "\033[93m$1 \033[00m"

}

prPurple(){

  echo -e "\033[95m$1 \033[00m"

}

prCyan(){

  echo -e "\033[96m$1 \033[00m"

}

prHeader(){

for each in $(seq 1 $COLUMNS)

do

   echo -n $1

done

}

prtxtCentre(){

title=$1

printf "%\*s\n" $(((${#title}+$COLUMNS)/2)) "$title"

}

download\_git\_versions\_info(){

GIT\_VERS\_URL="https://mirrors.edge.kernel.org/pub/software/scm/git/"

GIT\_VERS\_FILE="git\_vers\_info.html"

if [ -e "${GIT\_VERS\_FILE}" ]

then

    prYellow "Found old ${GIT\_VERS\_FILE}.Deleting this old file and downloading new content.Please wait..."

    rm -rf ${GIT\_VERS\_FILE}

else

    prGreen "Downloading git vers info. Please wait..."

fi

which wget 1>/dev/null 2>&1

if [ $? -ne 0 ]

then

   prRed "Sorry unable to download , wget command is not installed on this host. Please install it and retry"

   exit 2

fi

wget ${GIT\_VERS\_URL} --output-document=${GIT\_VERS\_FILE} 1>/dev/null 2>&1

if [ $? -ne 0 ]

then

   prRed "Unable to download. Please try the below command manually and verify"

   prRed "wget ${GIT\_VERS\_URL} --output-document=${GIT\_VERS\_FILE}"

else

   prGreen "Successfully downloaded git vers info from git-scm and stored the info into a file: ${GIT\_VERS\_FILE}"

fi

}

diplay\_all\_available\_git\_versions(){

  if [ ! -e ${GIT\_VERS\_FILE} ]

  then

    prRed "Unable to find the ${GIT\_VERS\_FILE}"

  fi

  prPurple "Sorting git versions. Please wait..."

  while read line

  do

    git\_vers+=($(echo $line | sed -n '/git-\([0-9]\+\.\)\+tar.gz/p'|awk -F '"' '{ print $2 }'|cut -c 5- | awk -F '.tar.gz' '{ print $1}'))

done < ${GIT\_VERS\_FILE}

  #echo ${#git\_vers[@]}

  prHeader "="

  prtxtCentre "Displaying all available git versions"

  cnt=0

  no\_vers=${#git\_vers[\*]}

  WIDTH=14

  prHeader "="

  for each\_ver in ${git\_vers[\*]}

  do

   printf "%-\*s %-\*s %-\*s %-\*s %-\*s %-\*s %-\*s %-\*s\n" $WIDTH  ${git\_vers[$cnt]}  $WIDTH ${git\_vers[$((cnt+1))]} $WIDTH ${git\_vers[$((cnt+2))]} $WIDTH ${git\_vers[$((cnt+3))]} $WIDTH  ${git\_vers[$((cnt+4))]} $WIDTH  ${git\_vers[$((cnt+5))]}  $WIDTH  ${git\_vers[$((cnt+6))]} $WIDTH  ${git\_vers[$((cnt+7))]}

   cnt=$((cnt+8))

   if [ $cnt -ge $no\_vers ]

   then

     break

   fi

  done

  prHeader "\_"

}

main() {

  clear

  prHeader "="

  prtxtCentre "Welcome to Automate Git installation using shell script"

  prHeader "="

  prCyan "Checking all available git versions from official git-scm websites. Please wait...."

  download\_git\_versions\_info

  diplay\_all\_available\_git\_versions

}

main