**grep tasks with solutions**

**grep: ( grep,egrep,fgrep)**

1. Write a grep command that selects the lines from the file1 that have exactly three characters

Ans: $ grep ^…$ filename

1. Write a grep command that selects the lines from the file1 that have at least three characters.

A: $ grep … filename

1. Write a grep command that selects the lines from the file1 that have three or fewer characters

A: $ grep -v …. filename

1. Write a grep command that count the number blank lines in the file1

A: $ grep -c ^$ filename

1. Write a grep command that count the number nonblank lines in the file1

A: $ grep -cv ^$ filename

1. Write a grep command that selects the lines from the file1 that have the string UNIX.

A: $ grep “UNIX” filename

1. Write a grep command that selects the lines from the file1 that have only the string UNIX.

A: $ grep ^UNIX$ filename

1. Write a grep command that copy the file to the monitor, but delete the blank lines.

A: $ grep -v ^$ filename

1. Write a grep command that selects the lines from the file1 that have at least two digits without any other characters in between

A: $ grep [0-9][0-9] filename

1. Write a grep command that selects the lines from the file1 that do not start with A to G

A: $ grep -v ^[A-G] filename

**sed: tasks with solutions**

1. Write a sed command that print numbers of lines beginning with “O”

Ans: $ sed -ne ‘/^O/ =’ filename

1. Write a sed command that delete digits in the given input file.

A: $ sed ‘s/[0-9]\*//g’ filename

1. Write a sed command that delete lines that contain both **BEGIN** and **END**

A: $ sed ‘/begin.\*end/d ’ filename

1. Write a sed command that delete lines that contain **BEGIN** but not **END**

A: $ sed -ne ‘/begin/p’ filename | sed ‘/end/!d’

1. Write a sed command that deletes the first character in each line in a file

A: $ sed ‘s/^.//g ’ filename

1. Write a sed command that deletes the last character in each line in a file

A: $ sed ‘s/.$//g ‘ filename

1. Write a sed command to delete character before last character in each line in a file

A: $ sed ‘s,\(.\*\)\(.\)\(.\),\1\3,g ’ filename

1. Write a sed command that swaps the first and second character in each line in the file

A: $ sed ‘s,\(.\)\(.\)\(.\*\),\2\1\3,g ‘ filename

1. Write a sed command that swaps the first and second word in each line in the file

A: $ sed ‘s,\([^ ]\*\) \*\([^ ]\*\),\2 \1 ,g ‘ filename

1. Write a sed command that deletes the first character in each line in a file.

A: $ sed 's/^./ / ' filename

1. Write a sed command that deletes the character before the last character in each line in a file.

A: $ sed 's/.$/ /' filename

1. Write a sed command that swaps the first and second words in a file.

A: $ sed ‘s/ \([^ ]\*\) \*\([^ ]\*\)/ \2 \1 /g’ filename

M: Write a sed command to delete all blank lines from a file:

A: $ sed ‘/$/d’ filename

**awk: tasks with solutions**

1. This program prints the maximum number of fields on any input line.

awk ’{ if (NF > max) max = NF } END { print max }’

1. This program prints every line longer than 80 characters. The sole rule has a relational expression as its pattern, and has no action (so the default action, printing the record, is used).

awk ’length($0) > 80’

1. This program prints every line that has at least one field. This is an easy way to delete blank lines from a file (or rather, to create a new file similar to the old file but from which the blank lines have been deleted).

awk ’NF > 0’

1. This program also prints every line that has at least one field. Here we allow the rule to match every line, then decide in the action whether to print.

awk ’{ if (NF > 0) print }’

1. This program prints 7 random numbers from 0 to 100, inclusive.

awk ’BEGIN { for (i = 1; i <= 7; i++) print int(101 \* rand()) }’

1. This program prints the total number of bytes used by files.

ls -l files | awk ’{ x += $4 } ; END { print "total bytes: " x }’

1. This program prints the maximum line length of file. The input is piped through the expand program to change tabs into spaces, so the widths compared are actually the right-margin columns.

expand file | awk ’{ if (x < length()) x = length() } END { print "maximum line length is " x }’

1. This program prints a sorted list of the login names of all users.

awk ’BEGIN { FS = ":" } { print $1 | "sort" }’ /etc/passwd

1. This programs counts lines in a file.

awk ’{ nlines++ } END { print nlines }’

1. This program also counts lines in a file, but lets awk do the work.

awk ’END { print NR }’

1. This program adds line numbers to all its input files, similar to ‘cat -n’.

awk ’{ print NR, $0 }’

**Example AWK Problems**

1. **Aim**: pipe your /etc/passwd file to awk , and print out the home directory of each user.

Create a home.awk file. I am creating using vi editor.

$ vi home.awk

{

if(match ($0, /^.\*home/) > 0)

{

split( $0, user)

split(user[1],homedir, ":")

print homedir[1]

}

}

Compile the program

$ cat /etc/passwd | awk -f home.awk

Administrator

Guest

sys1

cse

cse123

it

cse501

cse502

it1201

it1202

1. **Write an awk command to print the lines and line number in the given input file.**

**SOLUTION:**

**STEP1:**  **create input file**

$ cat > fileinput

welcome to

ECU

hello

**STEP2: create awk script**

$ cat > cmds.awk

{print NR, $0 }

**STEP3:** **execute awk program**

$ awk -f cmds.awk fileinput

1 welcome to

2 ECU

3 hello

$

**2.**W**rite an awk command to print first field and second field only if third field value is >=50 in the given input file. (input field separator is “:” and output field separator is “,”)**

**SOLUTION:**

**STEP1: create input data file**

$cat > file1

sachin:10:100

rahul:11:95

rohit:12:89

**STEP2: execute awk program**

$awk -F':' '$3>=50 {print $1”,”$2}' file1

sachin,10

rahul,11

rohit,12

1. **Consider the marks.txt is a file that contains one record per line( comma separate fields) of the student data in the form of studentid, student name, Telugu marks, English marks, Maths Marks, Science marks, Social Marks. Write an awk script to generate result for every students in the form of studentid, studentname, Total Marks and result. Result is PASS if marks is >=30 in TELUGU and English, and if marks>=40 in other subjects. Result is fail otherwise.**

**SOLUTION:**

**STEP1: create marks.txt file**

$cat > marks.txt

1001,name1,99,69,85,56,75

1002,name2,89,69,65,56,55

1003,name3,50,50,50,55,55

1004,name4,69,29,85,56,75

1005,name5,99,69,85,56,11

^d

**STEP2: create marks.awk script file**

$cat > marks.awk

{

total=$3+$4+$5+$6+$7

if($3>=30 && $4>=30 && $5>=40 && $6>=40 && $7>=40)

print $1,$2,total, "Pass";

else

print $1,$2,total, "fail";

}

**STEP3: execute awk program**

$awk -F “,” -f marks.awk marks.txt

1. **Write an awk program to print the fields 1 and 4 of a file that is passed as command line argument. The file contains lines of information that is separated by “,” as delimeter. The awk program must print at the end the average of all 4th field data.**

**SOLUTION:**

**STEP1: create data file**

$cat > data

12,13,14,15,one

22,23,24,25,two

34,23,45,23,three

44,55,66,77,four

^d

**STEP2: Execute awk program**

$awk -F',' '{print $1,$2,$3,$4,($1+$2+$3+$4)/4}' data

1. Write an awk program to demonstrate user defined functions and system command.

**SOLUTION:**

**STEP1: create data file**

$cat > data

12,13,14,15,one

22,23,24,25,two

34,23,45,23,three

44,55,66,77,four

^d

**STEP2: Create user.awk script file**

$cat >user.awk

{

if($3>0)

display($3)

}

function display(name)

{

print name

}

^d

**STEP3: execute awk program**

$awk -F',' -f user.awk data

1. **Write an awk script to count the number of lines in a file that do not contain vowels.**

**SOLUTION:**

**STEP1: Create a file called input**

$cat > input

this is one

213

BCDEFG

This is last line

^d

**STEP2: Create vowels.awk script file**

$cat vowels.awk

BEGIN{count=0}

!/[aeiou]/ {count++;print}

END{print "Number of lines="count}

^d

**STEP3: execute awk program**

$awk -f vowels.awk input

1. **Write an awk script to find the number of characters, words and lines in a file.**

**SOLUTION:**

**STEP1: Create a file called file7**

$cat > file7

This is a file

YEs NO

1234

^d

**STEP2: Create awk script file**

$cat > lines.awk

BEGIN{words=0;characters=0}

{

character+=length($0);

words+=NF;

}

END{print "lines=",NR," words=",words," Characters=",character}

^d

**STEP3: execute awk program**

$awk -f lines.awk file7

**EXAMPLE SHELL SCRIPTS**

1. **Aim:** Develop an interactive grep script that asks for a word and a

file name and then tells how many lines contain that word.

First we create a file with filename grep.sh

echo "Enter the pattern to be searched: "

read pattern

echo "Enter the file to be used: "

read filename

echo "Searching for $pattern from file $filename"

echo "The selected records are: "

grep "$pattern" $filename

echo "The no.of lines contains the word( $pattern ) :"

grep -c "$pattern" $filename

**Output :**

$ ./grep.sh

Enter the pattern to be searched:

computer

Enter the file to be used:

sales.dat

Searching for computer from file sales.dat

The selected records are:

10 computer 2345

10 computer 7685

The no.of lines contains the words ( computer ) :

1. **Aim**: write a shell script that takes a command –line argument and

reports on whether it is directory, a file, or something else.

First create a file named filetype.sh

echo "Enter the file name: "

read file

if [ -f $file ]

then

echo $file "---> It is a ORDINARY FILE."

elif [ -d $file ]

then

echo $file "---> It is a DIRCTORY."

else

echo $file "---> It is something else."

fi

**outputs:**

$ ./filetype.sh

Enter the file name:

sales.dat

sales.dat ---> It is a ORDINARY FILE.

$ ./filetype.sh

Enter the file name:

cse

cse---> It is a DIRCTORY.

1. **Aim**: Write a shell script that accepts one or more file name as

arguments and converts all of them to uppercase, provided they exist in the current directory.

First write the script upper.sh.

for file in \*

do

if [ -f $file ]

then

echo $file | tr '[a-z]' '[A-Z]'

fi

done

Reading input :

$ ls

afile

afile1.tar

afile.tar

aped.sed

errfile

exch.dat

exsed.sed

f1

f2

for.sh

grep.sh

**Output:**

$ ./upper.sh

AFILE

AFILE.TAR

APED.SED

ERRFILE

EXCH.DAT

EXSED.SED

F1

F2

FOR.SH

GREP.SH

1. **Aim**: Write a shell script that determines the period for which a

specified user is working on the system.

First create a file named logtime.sh.

echo "Enter the USER NAME : "

read user

last $user

**Output :**

$ ./logtime.sh

Enter the USER NAME :

cse123

cse123 tty7 :0 Fri Sep 26 13:27 still logged in

cse123 pts/1 :0.0 Thu Sep 25 15:08 - 15:45 (00:37)

cse123 tty7 :0 Thu Sep 25 14:53 - 16:32 (01:39)

cse123 tty7 :0 Thu Sep 25 14:13 - 14:25 (00:11)

cse123 tty7 :0 Tue Sep 23 13:54 - 15:30 (01:36)

cse123 pts/2 :20.0 Mon Sep 22 17:02 - 17:23 (00:21)

1. **Aim**: write a shell script that accepts a file name starting and ending line numbers as arguments and displays all the lines between the given line numbers.

First create a file named range.sh.

echo " Enter the file name :"

read file

if [ -f $file ]

then

echo "Enter the Starting line number:"

read snum

echo "Enter the Ending line number:"

read enum

if [ $snum -lt $enum ]

then

echo "The selected lines from $snum line to $enum line in $file :"

sed -n ' ' $snum','$enum' 'p' ' $file

else

echo "Enter proper starting & ending line numbers."

fi

else

echo "The file ' $file ' doesn't exists. "

fi

**Output:**

$./range.sh

Enter the file name :

sales.dat

Enter the Starting line number:

2

Enter the Ending line number:

4

The selected lines from 2 line to 4 line in sales.dat :

11 tvsets 8765

10 computer 7685

1. mouse 6785
2. **Aim**: write a shell script that deletes all lines containing a specified word in one or more files supplied as arguments to it.

First create a file detline.sh

echo "Enter the word to search for all lines :"

read word

echo "the file name are $\* ."

for i in $\*

do

echo "The name of the file :" $i

grep -v $word $i

done

**Output :**

$ ./detline.sh sales.dat sales1.dat sales2.dat

Enter the word to search for all lines :

computer

the file name are sales.dat sales1.dat sales2.dat

The name of the file : sales.dat

11 tvsets 8765

12 mouse 6785

13 keyboard 2341

The name of the file : sales1.dat

11 tvsets 8765

12 mouse 6785

13 keyboard 2341

The name of the file : sales2.dat

11 tvsets 8765

12 mouse 6785

1. keyboard 2341
2. **Aim**: write a shell script that computes the gross salary of an employee according to the following rules:
3. If basic salary is <1500 then HRA=10% of the basic and DA=90% of the basic
4. If the basic salary is>=1500 then HRA=500/- and DA=98% of the basic

First create a script named gsalary.sh

The basic salary is entered interactively through the key board.

echo "enter the basic salary:"

read bsal

if [ $bsal -lt 1500 ]

then

gsal=$((bsal+((bsal/100)\*10)+(bsal/100)\*90))

echo "The gross salary : $gsal"

fi

if [ $bsal -ge 1500 ]

then

gsal=$(((bsal+500)+(bsal/100)\*98))

echo "the gross salary : $gsal"

fi

**Output :**

$ ./gsalary.sh

enter the basic salary:

1200

The gross salary : 2400

$ sh gsalary.sh

enter the basic salary:

2400

the gross salary : 5252

1. **Aim**: Write a shell script that accepts two integers as its arguments

and computers the value of first number raised to the power of the second number.

First create the following file pow.sh

echo "Enter the integer value :"

read int1

echo "Enter the power of that integer:"

read int2

pv=$int1

i=1

while [ $i -lt $int2 ]

do

pv=`expr $pv \\* $int1`

i=`expr $i + 1 `

done

echo "The value of first number to the power of the second number :"

echo "$pv"

**output:**

$ ./ pow.sh

Enter the integer value :

2

Enter the power of that integer:

5

The value of first number to the power of the second number :

32

1. **Aim**: Write an interactive file –handling shell program. Let it offer the user the choice of copying removing, renaming, or linking files. Once the user has made a choice, have the same program ask the user for the necessary information, such as the file name ,new name and so on.

echo "\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*"

echo "

1. List of files.

2. Copying files.

3. Removing files.

4. Renaming files.

5. Linking files."

Echo "enter your choice "

read ch

case "$ch" in

1 ) echo "The list of file names."

ls –l

2 ) echo "Enter the old filename."

read ofile

echo "Enter the new file name."

read nfile

cp $ofile $nfile && echo "Copied sucessfully." || echo

"Copied is not possible." ;;

3 ) echo "Enter the file name to remove."

read rfile

rm -f $rfile && echo "Successfully removed." ;;

4 ) echo "Enter the old file name."

read ofile

echo "Enter the new file name."

read nfile

mv $ofile $nfile && echo "The file $ofile name renamed

to $nfile." || echo "You cann't Rename the file. “;;

5 ) echo "Enter the original filename."

read ofile

echo "Enter the new filename to link a file."

read lfile

ln $ofile $lfile && echo "Creat the linking file

Sccessfully." || echo "You cann't Linking the file.";; \* )

echo "Invalid option."

Echo " Enter correct choice."

esac

**Output :**

$ ./8a.sh

\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*

1. List of files.

2. Copying files.

3. Removing files.

4. Renaming files.

5. Linking files.

enter your choice

2

Enter the old filename.

f1

Enter the new file name.

f0

Copied sucessfully.

1. **Aim:** Write a shell script which receives two file names as arguments. It should check whether the two file contents are same or not. If they are same then second file should be deleted.

echo "enter the first file name"

read file1

echo "enter the second file name"

read file2

cmp $file1 $file2 && rm $file2

if [ -e $file1 ]

then

if [ ! –e $file2 ]

then

echo "The two files contents are same."

echo "The second file is deleted successfully."

else

echo "The two files contents are not same."

echo "You cann't remove the file '$file2' ."

fi

else

echo "You should enter the existing file names."

fi

**Output:**

$ ./check.sh

enter the first file name

file1.dat

enter the second file name

file3.dat

The two files contents are same.

The second file is deleted successfully.

$ .check.sh

enter the first file name

file1.dat

enter the second file name

file2.dat

file1.dat file2.dat differ: byte 1, line 1

The two files contents are not same.

You cann't remove the file 'file2.dat' .

$ ./check.sh

enter the first file name

file5.dat

enter the second file name

file3.dat

cmp: file3.dat: No such file or directory

You should enter the existing file names.

1. **Aim**: write a shell script that displays a list of all the files in the current directory to which the use has read ,write and execute permissions.

echo "The list of File Names in the curent directory."

echo "Which have Read,Write and Execute permisions. "

for file in \*

do

if [ -f $file ]

then

if [ -r $file -a -w $file -a -x $file ]

then

ls -l $file

fi

fi

done

**Output :**

$chmod 777 file1

$chmod 777 file2

$chmod 777 file3

$chmod 777 file4

$ ./perm.sh

The list of File Names in the curent directory.

Which have Read,Write and Execute permisions.

-rwxrwxr-x 1 allam allam 6722 2008-08-21 10:00 a.out

-rwxrwxrwx 1 allam allam 4 2008-08-18 16:48 file1

-rwxrwxrwx 1 allam allam 9 2008-08-18 16:48 file2

-rwxrwxr-x 1 allam allam 4 2008-08-19 14:33 file3

-rwxrwxrwx 1 allam allam 14 2008-08-18 16:48 file4

1. **Aim**: Develop an interactive script that ask for a word and a file name and then tells how many times that word occurred in the file.

echo " Enter the word to be searched"

read word

echo "Enter the filename to be used"

read flname

echo "the no. of times the word occured in the file."

grep -c $word $flname

**Output:**

$ ./wcount.sh

Enter the word to be searched

computers

Enter the filename to be used

sales.dat

the no. of times the word occured in the file.

2

1. **Aim**: Write a shell script to perform the following string operations:
   * 1. To extract a sub-string from a given string.
     2. To find the length of a given string.

echo "To find the length of the given string."

echo "Enter the string."

read string

strlen=${#string}

echo "The string length is : $strlen"

**Output :**

$ ./strlen.sh

To find the length of the given string.

Enter the string.

computer science and engineering.

The string length is : 33

1. Write a shell script that deletes all lines containing a specific word in one or more files supplied as arguments to it.

AIM: By using bash shell for writing the shell scripts, first create any file with 5-10 lines of data, then specify any particular word, if that word is present in any lines of file those lines will be deleted.

SOURCE CODE:

if [ $# -eq 0 ]  
then  
echo NO ARGUMENTS

else  
pattern=$1  
shift  
for fname in $\*  
do  
if [ -f $fname ]  
then  
echo DELETING $pattern FROM $fname

sed '/'$pattern'/d' $fname

else  
echo $fname NOT FOUND fi  
done  
fi

1. Write a Shell Script that accepts a file name, starting and ending line numbers as arguments and displays all lines between the given line numbers.

AIM: By using bash shell for writing the shell scripts, first create any file in that file read any line number as starting and ending line number and print all lines between that line numbers.

SOURCE CODE

if [ $# -eq 0 ]  
then  
echo no arguments

elif [ $# -eq 1 ]  
then  
echo one argument

elif [ $# -eq 2 ]  
then  
echo two arguments

else  
if [ ! -e $1 ]  
then  
echo file does not exist else  
sed -ne ''$2','$3' p' $1  
fi  
fi

1. Write a shell script that displays a list of files in current directory to which the user has read, write and execute permissions.

AIM: In the current directory we are able know which files have read, write and execute permissions.

SOURCE CODE:

METHOD 1:

$ls –l | grep ‘^.rwx’

METHOD 2:

echo "enter the directory name" read dir

if [ -d $dir ] then  
cd $dir  
ls > f

exec < f  
while read line  
do  
if [ -f $line ]  
then  
if [ -r $line -a -w $line -a -x $line ] then  
echo "$line has all permissions"  
else  
echo "files not having all permissions" fi  
fi  
done

fi

Method3:

echo "List of Files which have Read, Write and Execute Permissions in Current Directory" for file in \*  
do if [ -r $file -a -w $file -a -x $file ]  
then

echo $file  
fi  
done  
INPUT: sh prog3.sh