



**RATHINAM COLLEGE OF ARTS AND
SCIENCE (AUTONOMOUS)**

**RATHINAM TECHZONE, EACHANARI,
COIMBATORE – 641 021.**

**MANUAL
FOR
19BCSCP4 – LINUX &
SHELL PROGRAMMING
LAB
Semester II**

Prepared by,

Dr.S.KARTHIKEYAN M.Sc., Ph.D.,

Assistant Professor,

Department of Computer Applications.

SHELL PROGRAMMING PRACTICAL

EX. NO : 1

IDENTIFY THE CURRENT SHELL AND LENGTH OF THE STRING

AIM:

To Identify the Current Shell and Length of the String in Shell Script.

ALGORITHM:

Step-1:

Open the Terminal by right click on desktop and choose open terminal option or Press shortcut key Ctrl+T.

Step-2:

Identify the current shell using \$SHELL command and display the output by using echo command.

Step-3:

Assign a string to some variable and find the length of the string using len=\${#str1} command

Step-4:

Finally display the output by using the echo command.

PROGRAM:

```
echo "-----"
echo "To Identify the Current Shell and Length of the String"
echo "-----"
echo "The Current Shell is = $SHELL"
echo "-----"
str1="Welcome to Rathinam College"
echo $str1
len=${#str1}
echo "Length of the String = $len"
echo "-----"
```

RESULT:

Hence, the above program was successfully completed and verified.

EX.NO:2

COUNT BACKWARDS FOR 100 TO 0 USING FOR LOOP

AIM:

To count Backwards for 100 to 0 using for loop in Shell Script.

ALGORITHM:

Step-1:

Open the Terminal by right click on desktop and choose open terminal option or Press shortcut key Ctrl+T.

Step-2:

Initialize the count variable c=0.

Step-3:

Using the loop for ((i =100; i >= 0; --i)) then display the numbers from 100 to 0 using echo "\$i" .

Step-4:

Finally display the output by using the echo command.

PROGRAM:

```
echo "-----"
echo "To Count Backwards for 100 to 0 Using For Loop"
echo "-----"
c=0
for (( i =100; i >= 0; --i ))
do
    echo "$i"
    s=$((c++))
done
echo "-----"
echo "Count = $s"
echo "-----"
```

RESULT:

Hence, the above program was successfully completed and verified.

EX.NO:3

TO SEARCH FILE NAME USING REGULAR EXPRESSION

AIM:

To Search File Name using Regular Expression in Shell Script.

ALGORITHM:

Step-1:

Open the Terminal by right click on desktop and choose open terminal option or Press shortcut key Ctrl+T.

Step-2:

Search all jpg images by using find . -name "*.jpg" command.

Step-3:

Search files using Regular Expressions by using Regex or Regexp.

Step-4:

Finally display the output and verified.

PROGRAM:

1. To Search all jpg images:

```
find . -name "*.jpg"
```

2. To Search files using -o flag between different parameters.

```
find ./ -type f \( -iname \*.jpg -o -iname \*.png \)
```

3. To Search files using Regular Expressions:

```
1. find -regex '.*\.(jpg|png\)'
```

```
2. find . -regex '.*sh[s]?' -type f
```

4. To Search files using File Name:

```
1. find . -iname "ex3*" -print
```

```
2. find /home/administrator/rcas -name "*.txt" -print
```

RESULT:

Hence, the above program was successfully completed and verified.

EX NO: 4

SORTING UNIQUE AND DUPLICATE TEXT FILES

AIM:

To Sorting the unique and duplicate the text files in Shell Script.

ALGORITHM:

Step-1:

Open the Terminal by right click on desktop and choose open terminal option or Press shortcut key Ctrl+T.

Step-2:

Create two or more files using `$ cat > filename.txt` command.

Step-3:

Sort the files by using Sort Command.

Step-4:

Display the unique lines from sorted file by using `$ sort file1.txt file3.txt | uniq` command.

Step-5:

Remove the duplicate lines from sorted file by using `$ uniq -u filename.txt` command.

Step-6:

Finally display the output and verified using `$ cat filename.txt`

PROGRAM:

1. To create a Files using:

```
$ cat > file1.txt
```

```
C
```

```
C++
```


JAVA

VB

ED

2. To create another File using:

```
$ Cat > file2.txt
```

PYTHON

PHP

JAVA

SHELL

DBMS

3. To Sort the Unique files

```
$ sort file1.txt
```

4. To sort files in reverse order:

```
$ sort -r file1.txt
```

5. To Merge two files:

```
$ cat file1.txt file2.txt > file3.txt
```

6. To display the unique lines in files:

```
$ sort file1.txt file3.txt | uniq
```

7. Remove the duplicate lines from sorted file by using

```
$ uniq -u filename.txt
```

8. To view the content of the file:

```
$ cat file3.txt
```

RESULT:

Hence, the above program was successfully completed and verified.

Ex.No: 05

**PERFORM OPERATION USING INTERSECTION, DIFFERENCE AND SET
DIFFERENCE**

AIM:

To Perform Operation using Intersection, Difference and Set Difference.

ALGORITHM:

Step-1:

Open the Terminal by right click on desktop and choose open terminal option or Press shortcut key Ctrl+T.

Step-2:

Create two or more files using `$ cat > filename.txt` command.

Step-3:

Sort the files by using Sort Command.

Step-4:

Sorting and Comparison between two files using, `$ comm a.txt b.txt` command.

Step-5:

Find the intersection between two files using, `$ comm a.txt b.txt -1 -2`.

Step-6:

Find the set difference between two files using, `$ comm a.txt b.txt -2 -3`.

Step-7:

Finally display the output and verified.

PROGRAM:

1. Creating Files:

```
~$ cat > a.txt  
    red  
    black  
    blue  
    yellow  
    green  
    orange  
    gold  
    silver
```

```
~$ cat > b.txt  
    red  
    black  
    blue  
    yellow  
    green  
    orange  
    gold  
    pink  
    white
```

2. Displaying a Files:

```
~$ cat a.txt  
    red  
    black  
    blue  
    yellow  
    green  
    orange  
    gold
```

3. Combing two files:

```
~$ comm a.txt b.txt
      red
      black
      blue
      yellow
      green
      orange
      gold
pink
white
```

4. Sorting Two files:

```
~$ sort a.txt -o a.txt
~$ sort b.txt -o b.txt
~$ comm a.txt b.txt
      black
      blue
      gold
      green
      orange
pink
      red
white
      yellow
```

5. Intersection of Two Files:

```
~$ comm a.txt b.txt -1 -2
black
blue
gold
green
orange
red
yellow
```

6. Set Difference A-B of Two Files:

```
~$ comm a.txt b.txt -2 -3
```

7. Remove Common lines in Two Files:

```
~$ comm a.txt b.txt -3
```

```
pink  
white
```

8. Remove the Space before the content:

```
~$ comm a.txt b.txt -3 | sed 's/^\t/'
```

```
pink  
white
```

RESULT:

Hence, the above program was successfully completed and verified.

EX. NO : 6

**TO FIND AND DELETE DUPLICATE FILES
IN A DIRECTORY OF FILES**

AIM:

To find and delete duplicate files in a directory of files by using Shell
Script program.

ALGORITHM:

Step-1:

Open the Terminal by right click on desktop and choose open terminal option or Press shortcut key Ctrl+T.

Step-2:

Create a Shell Script program using nano command. For example
nano pgm6.sh

Step-3:

Type a program and save by pressing Ctrl + X and Choose Yes option for Save the program.

Step-4:

Compile the program by using chmod +x pgm6.sh and run the program by ./pgm6.sh.

Step-5:

Create a one file with some data and create a duplicate files for that created file.

Step-6:

Comparing the two files using MD5sum command and comm command.

Step-7:

Finally display the output and verified.

PROGRAM:

```
echo "-----"
echo "To find and delete duplicate Files in a directory of files"
echo "-----"

#!/bin/bash

#Filename: ex6.sh

#Description: Find and remove duplicate files and keep one sample of
each file.

echo "hello"> test ; cp test test_copy1 ; cp test test_copy2;
echo "next"> other;

# test_copy1 and test_copy2 are copy of test


ls -ls --time-style=long-iso | awk 'BEGIN {
getline; getline;
name1=$8; size=$5
}
{
name2=$8;
if (size==$5)
{
"md5sum "name1 | getline; csum1=$1;
"md5sum "name2 | getline; csum2=$1;
if ( csum1==csum2 )
{
```

```
print name1; print name2
}
};
size=$5; name1=name2;
}' | sort -u > test
cat test | xargs -I {} md5sum {} | sort | uniq -w 32 |
awk '{ print "^"$2"$" }' | sort -u > test_copy1
echo Removing..
comm test test_copy1 -2 -3 | tee /dev/stderr |
xargs rm
echo Removed duplicates files successfully.
```

RESULT:

Hence, the above program was successfully completed and verified.

EX. NO : 7

TO PERFORM SILENT OUTPUT FOR GREP

AIM:

To perform Silent output for grep by using Shell Script program.

ALGORITHM:

Step-1:

Open the Terminal by right click on desktop and choose open terminal option or Press shortcut key Ctrl+T.

Step-2:

Create a Shell Script program using nano command. For example
nano pgm7.sh

Step-3:

Type a program and save by pressing Ctrl + X and Choose Yes option for Save the program.

Step-4:

Compile the program by using chmod +x pgm7.sh and run the program by ./pgm7.sh.

Step-5:

The [\$# -ne 2] statement checks whether the total number of arguments to the script is two, otherwise it exits.

Step-6:

\$0 is used for arguments can be passed to scripts and can be accessed by script.

Step-7:

The command returns zero when it terminates after successful completion. The return status can be read from special variable \$? (run echo \$? immediately after the command execution statement to print the exit status).

Step-8:

Finally display the output and verified.

PROGRAM:

```
echo "-----"
```

```
echo "to perform Silent output for grep"
```

```
echo "-----"
```

```
#!/bin/bash
```

```
#Filename: silent_grep.sh
```

```
#Desc: Testing whether a file contain a text or not
```

```
if [ $# -ne 2 ]; then
```

```
echo "Usage: $0 match_text filename"
```

```
exit 1
```

```
fi
```

```
match_text=$1
filename=$2
grep -q "$match_text" $filename
if [ $? -eq 0 ]; then
echo "The text exists in the file"
else
echo "Text does not exist in the file"
fi
```

RESULT:

Hence, the above program was successfully completed and verified.

Ex.No: 08

PRINTING LINES BEFORE AND AFTER TEXT MATCHES

AIM:

To Printing lines before and after text matches by using Shell Script Commands.

ALGORITHM:

Step-1:

Open the Terminal by right click on desktop and choose open terminal option or Press shortcut key Ctrl+T.

Step-2:

Using seq command we can display the numbers in sequence manner, for example seq 10.

Step-3:

Grep command with -A option is to print the lines after a text match.

Step-4:

Grep command with -B option is to print the lines before a text match.

Step-5:

Grep command with -C option is to print the lines before and after a text match.

Step-7:

Finally display the output and verified.

PROGRAM:

1. In order to print three lines after a match, use the -A option:

```
$ seq 10 | grep 5 -A 3
```

5

6

7

8

2. In order to print three lines before the match, use the -B option:

```
$ seq 10 | grep 5 -B 3
```

2

3

4

5

3. Print three lines after and before the match, and use the -C option as follows:

```
$ seq 10 | grep 5 -C 3
```

2

3

4

5

6

7

8

4. If there are multiple matches, then each section is delimited by a line "--":

```
$ echo -e "a\nb\nc\na\nb\nc" | grep a -A 1
```

a

b

--

a

b

RESULT:

Hence, the above program was successfully completed and verified.

Ex.No: 09

PRINTING TEXT BETWEEN LINE NUMBERS OR PATTERNS

AIM:

To printing text between line numbers or patterns by using Shell Script Commands.

ALGORITHM:

Step-1:

Open the Terminal by right click on desktop and choose open terminal option or Press shortcut key Ctrl+T.

Step-2:

To print the lines of a text in a range of line numbers, M to N , use the following syntax:

\$ awk 'NR==M, NR==N' filename

Or, it can take the stdin input as follows:

\$ cat filename | awk 'NR==M, NR==N'

Step-3:

To print the lines of a text in a section with start_pattern and end_pattern , use the following syntax:

\$ awk '/start_pattern/, /end_pattern/' filename

Step-4:

Finally display the output and verified.

PROGRAM:

1. Printing Text between Line Numbers:

```
$ seq 100 | awk 'NR==4,NR==6'
```

4

5

6

2. Printing Text between Patterns:

```
$ cat section.txt
```

line with pattern1

line with pattern2

line with pattern3

line end with pattern4

line with pattern5

```
$ awk '/pa.*3/, /end/' section.txt
```

line with pattern3

line end with pattern4

RESULT:

Hence, the above program was successfully completed and verified.

Ex.No: 10

PARSING E-MAIL ADDRESSES AND URLS FROM TEXT

AIM:

To Parsing E-Mail addresses and URLs from text by using Shell Script Commands.

ALGORITHM:

Step-1:

Open the Terminal by right click on desktop and choose open terminal option or Press shortcut key Ctrl+T.

Step-2:

Create a file with e-mail address and URLs.

Step-3:

\$ egrep -o '[A-Za-z0-9._]+@[A-Za-z0-9._]\.[a-zA-Z]{2,4}' - this command is used to print the E-mail addresses from the given input file.

Step-4:

http://[a-zA-Z0-9\-\.\.]\.[a-zA-Z]{2,4} - this command is used to print the HTTP URL lines from the given input file.

Step-5:

Finally display the output and verified.

PROGRAM:

1. \$ cat > url_email.txt

This is a line of text contains,<email> #slynux@slynux.com. </email>
and email address, blog "http://www.google.com", test@yahoo.com
dfdfdfdddfdf;cool.hacks@gmail.com

<h1>Heading</h1>

2. \$ cat url_email.txt

This is a line of text contains,<email> #slynux@slynux.com. </email>
and email address, blog "http://www.google.com", test@yahoo.com
dfdfdfdddfdf;cool.hacks@gmail.com

<h1>Heading</h1>

3. As we are using extended regular expressions (+ , for instance), we should use egrep .

```
$ egrep -o '[A-Za-z0-9._]+@[A-Za-z0-9._]\.[a-zA-Z]{2,4}'
```

slynux@slynux.com

test@yahoo.com

cool.hacks@gmail.com

url_email.txt

4. The egrep regex pattern for an HTTP URL is as follows:

```
http://[a-zA-Z0-9\-\.\.]+\.[a-zA-Z]{2,4}
```

For example:

```
$ egrep -o "http://[a-zA-Z0-9\-\.\.]+\.[a-zA-Z]{2,3}" url_email.txt
```

```
http://www.google.com
```

```
http://code.google.com
```

RESULT:

Hence, the above program was successfully completed and verified.