JAGAN NATH GUPTA INSTITUTE OF ENGINEERING & TECHNOLOGY

LAB MANUAL

4CS4-24: Linux Shell Programming Lab

Week1

Session-1
a)Log into the system
Sol : Login
b)Use vi editor to create a file called myfile.txt which contains some text.
Sol : Vi mytable
c)correct typing errors during creation.
Sol: Practice vi editor commands
d)Save the file
Sol: :wq + Enter
e)logout of the system
Sol: logout

Note… Make Use of following commands:

To Get Into and Out Of vi

To Start `vi`

To use `vi` on a file, type in `vi filename`. If the file named `filename` exists, then the first page (or screen) of the file will be displayed; if the file does not exist, then an empty file and screen are created into which you may enter text.

*	**`vi filename`**	*edit `filename` starting at line 1*
	`vi -r filename`	*recover `filename` that was being edited when system crashed*

To Exit `vi`

Usually the new or modified file is saved when you leave `vi`. However, it is also possible to quit `vi` without saving the file.
Note: The cursor moves to bottom of screen whenever a colon (`:`) is typed. This type of command is completed by hitting the `<Return>` (or `<Enter>`) key.

*	**:x<Return>**	*quit `vi`, writing out modified file to file named in original invocation*
	:wq<Return>	*quit `vi`, writing out modified file to file named in original invocation*
	:q<Return>	*quit (or exit) `vi`*
*	**:q!<Return>**	*quit `vi` even though latest changes have not been saved for this `vi` call*

Session-2
a)Log into the system
b)open the file created in session 1
c)Add some text
d)Change some text
e)Delete some text
f)Save the Changes
Sol : Practice the commands in Vi editor
g)Logout of the system

Note… Make Use of following commands

Inserting or Adding Text

The following commands allow you to insert and add text. Each of these commands puts the `vi` editor into insert mode; thus, the `<Esc>` key must be pressed to terminate the entry of text and to put the `vi` editor back into command mode.

*	**i**	*insert text before cursor, until `<Esc>` hit*
	I	*insert text at beginning of current line, until `<Esc>` hit*
*	**a**	*append text after cursor, until `<Esc>` hit*
	A	*append text to end of current line, until `<Esc>` hit*
*	**o**	*open and put text in a new line below current line, until `<Esc>` hit*
*	**O**	*open and put text in a new line above current line, until `<Esc>` hit*

Changing Text

The following commands allow you to modify text.

*		*replace single character under cursor (no `<Esc>` needed)*

	R	*replace characters, starting with current cursor position, until* `<Esc>` *hit*
	cw	*change the current word with new text,* *starting with the character under cursor, until* `<Esc>` *hit*
	cNw	*change* `N` *words beginning with character under cursor, until* `<Esc>` *hit;* *e.g.,* `c5w` *changes 5 words*
	C	*change (replace) the characters in the current line, until* `<Esc>` *hit*
	cc	*change (replace) the entire current line, stopping when* `<Esc>` *is hit*
	Ncc *or* **cNc**	*change (replace) the next N lines, starting with the current line, stopping when* `<Esc>` *is hit*

Deleting Text

The following commands allow you to delete text.

*	**x**	*delete single character under cursor*
	Nx	*delete N characters, starting with character under cursor*
	dw	*delete the single word beginning with character under cursor*
	dNw	*delete* `N` *words beginning with character under cursor;* *e.g.,* `d5w` *deletes 5 words*
	D	*delete the remainder of the line, starting with current cursor position*
*	**Dd**	*delete entire current line*
	Ndd *or* **dNd**	*delete* `N` *lines, beginning with the current line;* *e.g.,* `5dd` *deletes 5 lines*

Week2

a)Log into the system
b)Use the cat command to create a file containing the following data. Call it mytable use tabs to separate the fields.

```
1425        Ravi        15.65
4320        Ramu        26.27
6830        Sita        36.15
1450        Raju        21.86
```

Sol: cat > mytable

```
1425        Ravi        15.65
4320        Ramu        26.27
6830        Sita        36.15
1450        Raju        21.86
```

c)Use the cat command to display the file, mytable.
Sol: $cat mytable

```
  1425      Ravi        15.65
  4320      Ramu        26.27
  6830      Sita        36.15
  1450      Raju        21.86
```

d) Use the vi command to correct any errors in the file,
mytable. Sol: Verify the file with Vi editor Commannds
e) Use the sort command to sort the file mytable according to the first field. Call the sorted file my table
(same name)
Sol: $sort +1 mytable > mytable
f) Print the file mytable
Sol: cat mytable

```
1425        Ravi        15.65
1450        Raju        21.86
4320        Ramu        26.27
6830        Sita        36.15
```

g) Use the cut and paste commands to swap fields 2 and 3 of mytable. Call it my table (same name)
Sol: $cut -f1 > mytab1
$ cut –f 2 > mytab 2
$cut –f 3 > my tab3
$paste mytab3 mytab2 > mytab4
$paste mytab1 mytab4 >
mytable h)Print the new file,
mytable Sol: $ cat mytable

```
  1425     15.65     Ravi
  1450    21.86     Raju
  4320    26.27      Ramu
  6830    36.15      Sita
```

i)Logout of the system.

Note… Make Use of following commands

Cat:----

`cat`	*to display a text file or to con**cat**enate files*	
	`cat file1`	displays contents of `file1` on the screen (or window) without any screen breaks.
	`cat file1 file2`	displays contents of `file1` followed by `file2` on the screen (or window) without any screen breaks.
	`cat file1 file2 > file3`	creates `file3` containing `file1` followed by `file2`

Sort :----
The "`sort`" command sorts information piped into it. There are several options that let you sort information in a variety of ways.

```
ps -ef | sort
```

The most important options in Sort :

The following list describes the options and their arguments that may be used to control how **sort** functions.

- **-** Forces **sort** to read from the standard input. *Useful for reading from pipes and files simultaneously.*
- **-c** Verifies that the input is sorted according to the other options specified on the command line. If the input is sorted correctly then no output is provided. If the input is not sorted then **sort** informs you of the situation. The message resembles this.
    - `sort: disorder: This line not in sorted order.`
-
- `-m` Merges the sorted input. **sort** assumes the input is already sorted. **sort** normally merges input as it sorts. This option informs **sort** that the input is already sorted, thus **sort** runs much faster.
- **-o** *output* Sends the output to file *output* instead of the standard output. The *output* file may be the same name as one of the input files.
- **-u** Suppress all but one occurrence of matching keys. Normally, the entire line is the key. If field or character keys are specified, then the suppressing is done based on the keys.
- **-y** ***kmem*** Use *kmem* kilobytes of main memory to initially start the sorting. If more memory is needed, **sort** automatically requests it from the operating system.

The amount of memory allocated for the sort impacts the speed of the sort significantly. ***If no kmem is specified, sort starts with the default amount of memory (usually 32K).*** The maximum (usually 1 Megabyte) amount of memory may be allocated if needed. If 0 is specified for *kmem*, the minimum (usually 16K) amount of memory is allocated.

- **-z** *recsz* Specifies the record size used to store each line. Normally the *recsz* is set to the longest line read during the sort phase. If the -c or -m options are specified, the sort phase is not performed and thus the record size defaults to a system size. If this default size is not large enough, **sort** may abort during the merge phase. To alleviate this problem you can specify a *recsz* that will allow the merge phase to run without aborting.

Week3

Session1:

a)Login to the system

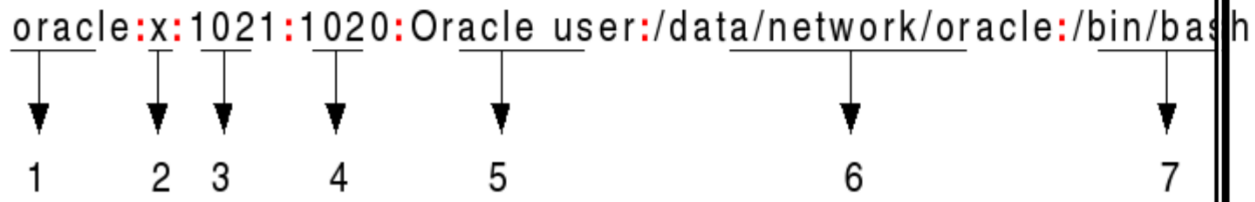
b)Use the appropriate command to determine your login shell

Sol: $echo $SHELL

sh

c)Use the /etc/passwd file to verify the result of step b.

Sol: $cat /etc/passwd


oracle:x:1021:1020:Oracle user:/data/network/oracle:/bin/bash
1
2
3
4
5
6
7

d)Use the who command and redirect the result to a file called myfile1. Use the more command to see the contents of myfile1. Sol : $who > myfile1 | more

```
User1 pts/0 Apr 23 10:43
User2 pts/1 May 6 18:19
```

e)Use the date and who commands in sequence (in one line) such that the output of date will display on the screen and the output of who will be redirected to a file called myfile2. Use the more command to check the contents of myfile2. Sol: $ date ; who > myfile2

```
Fri Aug 9 16:47:32 IST 2008
```

Cat myfile2 :

```
User3 pts/2 Apr 25 10:43
User4 pts/3 May 8 18:19
```

Note… Make Use of following commands:

Who :---

The "who" command lets you display the users that are currently logged into your Unix computer system.

who

This is the basic who command with no command-line arguments. It shows the names of users that are currently logged in, and may also show the terminal they're logged in on, and the time they logged in.

who | more

In this example the output of the who command is piped into the more command. This is useful when there are a lot of users logged into your computer system, and part of the output of the who command scrolls off the screen. See the more command for more examples.

```
who -a
```

The -a argument lists all available output of the who command for each user.

Piping:---

To connect the output of the one command directly to the input of the other command. This is exactly what pipes do. The symbol for a pipe is the vertical bar |

For example, typing

```
% who | sort
```

will give the same result as above, but quicker and cleaner.

To find out how many users are logged on, type

```
% who | wc -l
```

Session 2:

Input File : file1.dat :
Unix is Multiuser OS
Unix was developed by Brian Kernighan and KenThomson

a)Write a sed command that deletes the first character in each line in a file.
Sol: **sed** 's/^./ /'' file1.dat
nix is Multiuser OS
nix was developed by Brian Kernighan and KenThomson

b)Write a sed command that deletes the last character in each line in a file.
Sol: sed '$s/.$//' file1.dat
Unix is Multiuser O
Unix was developed by Brian Kernighan and KenThomso

c)Write a sed command that swaps the first and second words in each line in a file.

```
      sed -e 's/\([^ ]\+\)   *\([^ ]\+\)/\2 \1/'
```

sed 's/\([a-z]*\) \([a-z]*\)/\2 \1/' (Modified & working)

(Substrings enclosed with "`\(`" and "`\)`" can be referenced with "\n" (n is a digit from 1 to 9))

Note : Make use of following Link to know more about sed

Ref : http://www.grymoire.com/Unix/Sed.html#uh-0

Week4

a)Pipe your /etc/passwd file to awk, and print out the home directory of each user.
Sol: cat /etc/passwd | awk „ { print $7}‟
b)Develop an interactive grep script that asks for a word and a file name and then tells how many lines
contain that word.
Sol:

```
echo “Enter a word”
read word
echo “Enter the filename”
read file
nol=grep -c $word $file
echo “ $nol times $word present in the $file”
```

c)Part using awk
Sol:

```
echo “Enter a word”
read word
echo “Enter the filename”
read file
nol=awk „/$word/ { print NR }‟  Infile
echo “ $nol times $word present in the $file”
```

Note… Make Use of following commands:

Grep: ---grep is one of many standard UNIX utilities. It searches files for specified words or patterns. First clear the screen, then type

```
% grep science science.txt
```

As you can see, **grep** has printed out each line containg the word **science**.

Or has it ????

Try typing

```
% grep Science science.txt
```

The **grep** command is case sensitive; it distinguishes between Science and science.

To ignore upper/lower case distinctions, use the -i option, i.e. type

```
% grep -i science science.txt
```

To search for a phrase or pattern, you must enclose it in single quotes (the apostrophe symbol). For example to search for spinning top, type

```
% grep -i 'spinning top' science.txt
```

Some of the other options of grep are:

-v display those lines that do NOT match
-n precede each matching line with the line number **-**
c print only the total count of matched lines

Try some of them and see the different results. Don't forget, you can use more than one option at a time. For example, the number of lines without the words science or Science is

```
% grep -ivc science science.txt
```

Note: Make use of Following Link to know about Awk

Ref : http://www.grymoire.com/Unix/Awk.html

Week5

a)Write a shell script that takes a command –line argument and reports on whether it is directory, a file, or something else.
Sol:

```
echo " enter file"
read str
if test -f $str
then echo "file exists n it is an ordinary file"
elif test -d $str
then echo "directory file"
else
echo "not exists"
fi
if test -c $str
then echo "character device files"
fi
```

b)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. Sol:

```
# get filename
echo -n "Enter File Name : "
read fileName

# make sure file exits for reading
if [ ! -f $fileName ]
then
echo "Filename $fileName does not
exists" exit 1
fi

# convert uppercase to lowercase using tr command
tr '[A-Z]' '[a-z]' < $fileName
```

c)Write a shell script that determines the period for which a specified user is working on the system.
Sol:

```
echo “enter the login of the user”
read name
logindetails=`who|grep –w “$name” | grep “tty”
if [ $? –ne 0 ]
then
echo “$name has not logged in yet”
exit
fi
```

```
loginhours=`echo “$logindetails” | cut –c 26,27`
loginminuts=`echo “$logindetails” | cut –c 29-30`
hoursnow=‘`date | cut –c 12,13` minnow =`date |
cut –c 15,16`
hour=`expr $loginhours - $hoursnow`
min=`expr $loginminuts - $minnow`
echo “ $name is working since $hour Hrs - $min Minuts”
```

Week6

a)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. Sol:

```
If [ $#  -ne 3 ]
then
echo “chech the arguments once”
lastline=‟wc –l < $1‟
if [ $2 –lt $lastline –a $3  -le $lastline ]
then
nline=‟expr $3 -$2 + 1‟
echo “‟tail +$2 $1 | head -$nline‟”
else
echo “invalid range specification”
fi
fi
```

b) Write a shell script that deletes all lines containing a specified word in one or more files supplied as arguments to it.
Sol:

```
if [ $# -lt 1]
then
echo “ Chech the arguments once”
exit
fi
echo “Enter a
word” read word
for file in
$* do
grep –iv “$word” $file | tee 1> /dev/null
done
echo   “ lines containing given word are deleted”
```

Week7

a)Write a shell script that computes the gross salary of a employee according to the following rules:
i)If basic salary is < 1500 then HRA =10% of the basic and DA =90% of the basic. ii)If basic salary is >=1500 then HRA =Rs500 and DA=98% of the basic The basic salary is entered interactively through the key board.
Sol:

```
echo enter basic salary
read sal
a=0.1
b=0.8
echo $a
echo "hra is"
hra=`echo 0.1 \* $sal|bc`
echo da is
da=`echo 0.8\*$sal|bc`
gsal=‟expr $hra + $da + $sal‟
echo $gsal
```

b)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. Sol:

```
If [ $# -ne 2 ]
then
echo “chech the number of arguments”
count=1
result=1
if [ $2 –ge 0 ]
then
while [ $count –le $2 ]
do
result=`expr $result \* $1`
count=`expr $count + 1`
done
fi
fi
```

Week8

a)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 program ask the user for the necessary information, such as the file name, new name and so on.

b)Write shell script that takes a login name as command – line argument and reports when that person logs in

Sol:

```
#Shell script that takes loginname as command line arg and reports when that person logs
in.
if [ $# -lt 1 ]
then
echo improper usage
echo correct usage is: $0 username
exit
fi
logname=$1
while true
do
who|grep "$logname">/dev/null
if [ $? = 0 ]
then
echo $logname has logged in
echo "$logname">>sh01log.txt
date >>sh01log.txt
echo "Hi" > mesg.txt
echo "$logname" >> mesg.txt
echo "Have a Good Day" >> mesg.txt
mail "$logname" < mesg.txt
exit
else
sleep 60 fi  done
```

c)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.

Sol:

```
echo  “enter first file name”
read  file1
echo “ enter second file name”
read file2
cmp file1 file2 >   file3
if [    -z  $file1 ]          rm file2
fi
echo  “duplicate file deleted successfully”
```

Week9

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

```
ls –l | grep “^.rwx” | cut –f 9
```

b)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.

c)Write a shell script to perform the following string operations:

i)To extract a sub-string from a given string.

ii)To find the length of a given string.

Note: Make use of Following Link to know about Shell Programming

Ref : http://www.freeos.com/guides/lsst/ch02.html