Working with a Vi Editor

1. Create a file using vi and enter the following text:

A network is a group of computers that can communicate with each other, share resources, and access remote hosts or other networks. Netware is a computer network operating system designed to connect, manage, and maintain a network and its services. Some of the network services are Netware Directory Services (NDS), file system, printing and security.

a. Change the word “Netware” in the second line to “Novell Netware.”

bash

vi file.txt

:%s/Netware/Novell Netware/2

b. Insert the text “(such as hard disks and printers)” after “share resources” in the first line.

Navigate to the line in vi and enter insert mode. After "share resources," type:

csharp

(such as hard disks and printers)

c. Append the following text to the file:

"Managing NDS is a fundamental administrator role because NDS provides a single point for accessing and managing most network resources."

Move to the end of the file in command mode:  
bash  
  
$

Append the text in insert mode.

Working with Shell

1. Type some text on the shell separated by space.

Perform the following:

Move cursor one word back:  
bash  
  
Ctrl + b

Move cursor one word forward:  
bash  
  
Ctrl + f

Move cursor to the first character:  
bash  
  
Ctrl + a

Move cursor to the end:  
bash  
  
Ctrl + e

Delete text from the second word to the last character:  
Use Ctrl + w repeatedly.

Delete the current line:  
bash  
  
Ctrl + u

2. In lab 4, display errorlog.txt using cat with command completion.

bash

cat errorlog.txt

3. Display the history of commands used so far.

bash

history

4. Search ls command in the history file.

bash

history | grep ls

5. Repeat the last command.

bash

!!

6. Execute 3 commands from the history file.

bash

!<command\_number>

7. What are the different shells available?

bash

cat /etc/shells

Understanding Access Permissions

1. Create an empty file “demofile” and perform the following instructions:

bash

touch demofile

a. Revoke read permission from owner and use cat:  
bash chmod u-r demofile cat demofile # Gives permission error b. Revoke write permission from owner and open using vi:  
bash chmod u-w demofile vi demofile # Cannot save changes c. Add read and write permission to owner:  
bash chmod u+rw demofile d. Revoke write and execute permissions from others and group:  
bash chmod go-wx demofile e. Add write permission to group only:  
bash chmod g+w demofile f. Assign read permission to all:  
bash chmod a+r demofile g. Revoke read permission from others:  
bash chmod o-r demofile

2. Additional tasks:

Give the execute permission for the user for a file chap1:  
bash  
  
chmod u+x chap1

Give execute permission for user, group, and others for a file add.c:  
bash  
  
chmod a+x add.c

Remove execute permission from user and give read permission to group and others for a file aa.c:  
bash  
  
chmod u-x,go+r aa.c

Give execute permission for users for a.c, kk.c, nato, and myfile using a single command:  
bash  
  
chmod u+x a.c kk.c nato myfile

3. Create a directory “demo” and copy /etc/passwd file in it.

Create the directory and copy the file:  
bash  
  
mkdir demo

cp /etc/passwd demo

Display contents of demo:  
bash  
  
ls demo

Revoke read permission from the demo directory and use ls:  
bash  
  
chmod u-r demo

ls demo # Gives permission error

Revoke write permission from the demo directory and try to copy /etc/profile into it:  
bash  
  
chmod u-w demo

cp /etc/profile demo # Permission denied

Delete passwd file from the demo directory:  
bash  
  
rm demo/passwd

Revoke execute permission from the demo directory and try cd command on it:  
bash  
  
chmod u-x demo

cd demo # Permission denied

Using Process-Related Commands

1. Find out the PID of the processes activated by you.

bash

ps -u $USER

2. Find information about all processes currently active.

bash

ps -aux

3. Start a process in the background and find its PID.

bash

sleep 300 &

jobs -l

4. Run a job in the background.

bash

sleep 100 &

5. Bring the last background job to the foreground.

bash

fg

6. Run 3 jobs in the background and bring the first job to the foreground.

bash

sleep 100 &

sleep 200 &

sleep 300 &

fg %1

7. Stop the current job.

bash

Ctrl + z

8. Start a stopped job.

bash

bg

9. Run a job.

bash

sleep 500

10. Kill the last job.

bash

kill %%

11. Kill your shell using the process ID.

bash

kill -9 $$

12. Execute an ls command by setting priority as -10 using nice.

bash

nice -n -10 ls

13. Display a date every hour using cron.

Edit the crontab file:  
bash  
  
crontab -e

Add the following line to execute date every hour:  
bash

0 \* \* \* \* date