PRACTICAL FILE
OF
"UNIX AND LINUX PROGRAMMING"

Submitted To:
Er. Alisha Gupta
Assistant Professor
CSE DEPTT.

Submitted By:
Shrey Jain
1221302
CSE 3nd Year

Department of Computer Science and Engineering

Seth Jai Parkash Mukand Lal Institute of Engineering & Technology,Radaur-135001(Yamuna Nagar)

(Affiliated To Kurukshetra University Kurukshetra Haryana India)

INDEX

S.NO.	PRACTICALS	REMARKS
1	Familiarize with Unix/Linux logging/logout and simple commands.	
2	Familiarize with 'vi' editor and Linux GUIs.	
3	Using bash shell, develop simple shell programs.	
4	Using bash shell, develop a program to display Fibonacci series.	
5	Using bash shell, develop a program to find average of list of numbers.	
6	Using bash shell, develop a program to merge data of two files.	
7	Compile and debug various C programs using different options.	
8	Learning of installation and upgradation of Linux operating system.	
9	Study of UNIX shell.	
10	Develop advanced shell programs using AWK and GREP.	

Practical No. 1

AIM : - Familiarize with Unix/Linux Logging/logout and simple commands

SIMPLE COMMANDS

date: -date command prints system date and time

Syntax:-

date [OPTION]... [+FORMAT]

or:

date [OPTION] [MMDDhhmm[[CC]YY] [. ss]]

where format can be
$D Date -MM/DD/YY
$H Hour -00 to 23
$I Hour -00 to 11
$M Minute -00 to 59
$s Second -00 to 59
$T Time -HH:MM: SS
$y Year's last two digits
$w Day of the week (0 for sunday,1 for monday and so on)
$r Time in AM/PM

To display system date and time.

$ date
Tue Mar 12 11:30:31 IST 2024

To display system time in AM/PM format.

$ date +%r
09:25:10 AM

To display day of week.

$ date +%w
2

To display date in MM/DD/YY format.

$ date +%D
03/12/24

To display Hours.

$ date +%H
09

To display Minutes.

$ date +%M
30

To display Seconds.

$ date +%S
44

who:-This command is used to display who is currently logged on the system.
syntax:-who

Checking who is currently logged on.
$ who
Cse tty7 2024-03-12 11:30 (:0)

The first column of output represents the user names. The second column represents the corresponding terminal names and remaining represents at which the users are logged on. Linux is a multi-user operating system. So multiple can logged on at the same time.

who am i:-Tells who you are.
syntax:-who am i

Checking who you are.
$ who am i

cal:-This command will display calendar for specified month and year.
syntax:-cal [month] <year>

Display calendar for year 2024
$ cal 2024
It will display calendar for year 2024.

Display calendar for first month of year 2024.
$ cal 1 2024
January 2024
Mo Tu We Th Fr Sa Su
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31

history:-To display to see list of remembered commands.
syntax:-history [option]

To see history of previously executed commands
$ history
1 history
2 man clear
3 ln shi1.txt shi2.txt
4 ls -l shi1.txt
5 chmod 777 shi1.txt

clear:-To clear the screen
Syntax:-clear

Example:-To clear the screen
$ clear

expr:-This command is used to perform arithmetic operations(operators + for addition,- for subtraction ,* for multiplication , % for remainder ,/ for division is used) on integers.
Example:-To multiply 5 with 2
$ x=3
$ y=4
$ expr $x * $Y
$ 12

Directory oriented commands:-
A) mkdir:- This command ids used to create a new directory.
syntax :-
mkdir <directory-name>

Creating a new directory
$ mkdir linux
This command will create a new subdirectory with name linux of current directory.
Creating more than one directories in single move.
$ mkdir x y z
This will create three directories having names x,y,z respectively.

B) rmdir :- This command is used to remove a directory. Directory should be empty before deletion.
syntax:-
rmdir <directory-name>
Example :- Removing a directory
$ rmdir pan
This will remove directory having name pan.

Example :- Removing subdirectories in a directory with single rmdir
-p option is used for this.
$ rmdir -p linux/teji/teji1/

C)Pwd:- This command will displays full path name for present working directory.
syntax:-pwd
Example:-To see current working directory
$ pwd
/home/cse

D) cd:- This command is used for changing current directory to specified directory.
syntax:- cd <directory-name>

Example:-To move to a specified directory
$ cd dir
~/dir$ pwd
/home/cse/dir
Example:-To move to a parent directory
$ cd ..
$ pwd
/home/cse

Example:-To move to a root directory

```
$ cd ~
$pwd
/home/cse
```

E) ls:- This command is used to list the contents of specified directory
syntax:- ls [options] <directory-name>
options :-

Example:-Lists all files and directories including hidden files.
-a(All) option is used for this purpose.

```
$ls -a
.                       .gnome2              .ssh
..                      .gnome2_private      .sversionrc
.aaa.sh.swp             .gstreamer-0.8       .swp
ani.sh~                 .gtkrc               t1.doc
.ani.sh.swp             .gtkrc-1.2-gnome2    t1.txt
.bash_history           .ICEauthority        t3.doc
.bash_logout            ishan                t4.doc
.bash_profile           linux book           .tcshrc
.bashrc                 linux slides         t.doc
c1.txt                  .mailcap             te.doc
c2.txt                  .metacity            teji
C___C___PROJECTS        .mime.types          teji1.doc
```

Example:-Lists all files and directories in reverse order
-r(reverse) option is used for this purpose.

```
$ ls -r
```

Example:-Recursively lists all files and directories as well as files in subdirectories

```
$ ls -R dir
dir:
dir1  t4.txt  teji.txt

dir/dir1:
t.txt
```

Practical No. 2

AIM : Familiarize with vi editor and Linux GUIs.

OBJECTIVES

This Practical will help the students to familiarize with vi editor & Linux GUI.

EQUIPMENTS / INSTRUMENTS / SYSTEM REQUIRED

Hardware Used: Standard PC

Software Used: , Red Hat Linux 4.0, Bash shell, vi editor

THEORY

Linux offers various types of editor like ex, sed, ed, vi etc to create and edit your files(data files,text files etc).the famous one is vi editor created by Bill Joy at the university of California at Berkley.

Starting Vi Editor:- This editor can be invoked by typing vi filename at the prompt. If you specify a filename as an argument to vi, then the vi will edit the specified file, if it exists.

$ vi <filename> [Enter]

A status line at the bottom of the screen (25th line) shows the filename, current line & character position in the edited file.

Modes: - The editor works on 3 modes as follows:-

a) Insert Mode:-

(1) The text should be entered in this mode and any key pressed is created as text.

(2) We can enter in this mode through command mode by pressing any of the

b) Command Mode:-

(1) It is the default mode when we start up vi Editor.

(2) All the commands in vi Editor should be used in this mode.

(3) We can enter into this mode from insert mode by pressing. [Esc] key and from Ex mode by pressing Enter.

c) Ex Mode:-

(1) The ex mode command can be entered at the last line of the screen of the

Mode.

(2) We can enter into this mode directly from input mode or vice-versa.

The following are some commands that are used:

Insert Command:-

(1) i :-Insert before cursor.

(2) I :-Insert at the end of current line.

(3) a :-Append after cursor.

(4) A:- Append at the end of the current line.

(5) o :-Insert a blank line below the current line.

(6) O :-Insert a blank line above the current line.

Delete Command:-

(1) x:-Delete the character at current position.

(2) <n>x(n is any no.)Delete specified no of character from current position.

(3) X:-Delete the character before the cursor.

(4) (n)X:-Delete the no. of characters before the cursor.

(5) dw:- Delete from cursor position to the end of the current word.It stops at any punctuation that appears with in the word.

(6) dW :-same as 'dw'but ignores the punctuation character.

(7) db :- Deletes from cursor position to beginning of the current word.It stops at any punctuation that appear with in the word.

(8) dB :-same as'db' but ignores punctuations.

(9) dd :-Deletes the current line.

Replace Commands:-

(1) r:-Replace single character at the cursor position.

(2) R:-Replace character until escape key is pressed from current cursor position.

(3) s :-Replace single character at cursor position with no of characters.

(4) S:- Replace the entire line.

Cursor Movement Command:-

(1) h :-Moves cursor to the left.

(2) l :-Moves cursor to the right

(3) k :-Moves cursor to the up.

(4) j :-Moves cursor to the down

(5) w :-Forwards to the first letter of next word but stops at any punctuations that appears with the word.

(6) b :-Backword to the first letter of previous word that stops at any punctuation that appears with the word.

(7) e :-Moves forward to the end of the current word but stops at any punctuation that appears with the word.

(8) W :-same as w but ignores punctuations.

(9) E :-same as e but ignores punctuations.

(10) B:- same as b but ignores punctuations.

(11) [Enter]:- Forward to the beginning of the next line.

Redo Commands:-

(period):-Repeats the most recent editing operation performed.

Undo Commands:-

u :-undo's the most editing operation performed.

Ex Mode Commands:-

Some of the Ex mode commands are given below.

These commands should be used in Ex mode prefixed by (:)colon.

(1) :w :-Saves without quiting.

(2) :w<filename> :-Saves the content into a file specified in the filename.

(3) :mnw<filename> :-saves the lines m to n into the specified file name.

(4) :.w<filename> ;-Saves the current line into specified file.

(5) :$w<filename> :-Saves the last line of text into the specified file.

(6) :wq :-Saves file and quit from vi editor.

(7) :q! :-Quit without saving

Practical No. 3

AIM :- Using bash shell develop simple shell programs

1)TO FIND THE FACTORIAL OF A NUMBER.

```
echo "Enter the number"
read num
i=2
fact=1
while [ $i -le $num ]
do
fact=`expr $fact*$i|bc`
i=`expr $i+1|bc`
done
echo "The factorial of the number is:"
echo $fact
```

Output:

```
$ sh fact.sh
Enter the number
8
The factorial of the number is:
40320
```

2)TO FIND LARGEST AMONG THREE NUMBERS.

```
echo "enter three numbers"
read a
read b
read c
if test $a -gt $b -a $a -gt $c
then
echo "First number is largest";
elif test $a -lt $b -a $b -gt $c
then
echo "Second number is largest";
else
echo "Third number is largest";
fi
```

Output:

```
$ sh large.sh
enter three numbers
12
18
09
Second number is largest
```

Practical No. 4

AIM :- Shell program to display Fibonacci series

TO DISPLAY FIBONACCI SERIES.

```
echo "Enter the number of terms in the fibonacci series"
read n
c=0
a=-1
b=1
echo "The Series is :"
while [ $n -gt 0 ]
do
c=`expr $a+$b|bc`
a=$b
b=$c

echo $c
n=`expr $n-1|bc`
done
```

Output:

```
$ sh fibno.sh
Enter the number of terms in the fibonacci series
10
The Series is :
0
1
1
2
3
5
8
13
21
34
```

Practical No. 5

AIM :- Shell program to find average of list of numbers

TO FIND AVERAGE OF LIST OF NUMBERS.

```
echo "Enter the range of numbers"
read n
i=1
sum=0.0
avg=0.0

echo "enter the numbers"
while [ $i -le $n ]
do
read a
sum=`expr $sum+$a|bc`
i=`expr $i+1|bc`
done
avg=`expr $sum/$n|bc`
echo "the average of numbers is "
echo $avg
```

Output:

```
$ sh average.sh
Enter the range of numbers
5
enter the numbers
10
100
120
200
the average of numbers is
86
```

Practical No. 6

AIM :- Shell program to merge two files

PROGRAM TO MERGE TWO FILE.

```
echo "enter first file name="
read fname1
echo "enter second file name="
read fname2
echo "enter data in first file"
cat > $fname1
echo "enter data in second file"
cat > $fname2
echo "enter target file name"
read tname
cat >> $fname1 $fname2
mv $fname1 $tname
echo "Data After merging is"
cat $tname
```

Output:

```
$ sh merge.sh
enter first file name=
Shrey
enter second file name=
Jain
enter data in first file
Hello,
enter data in second file
how are you?
enter target file name
ShreyJain
Data After merging is
Hello,
how are you?
```

Practical No. 7

AIM:- Compile and debug various C programs.
There are plenty of c and c++ compiler under unix based operating system, but the most famous one should be GNU gcc compiler. A well known open source kernel Linux is compiled by gcc as well.

The simplest way to compile and get to run the compiled binary, do as below:
gcc hello.c
or if your source code is written in c++, **g++ hello.cc**

By default, gcc will create an output binaries, a.out.

To execute the binary, ./a.out

Compile and Debug program i.e. multiplication of matrix

```
#include<stdio.h>
void main()
{
int a[3][3],b[3][3],c[3][3],m,n,i,j,p;
printf("\n Enter First 3*3 Matrix Elements\n");
for(i=0;i<3;i++)
for(j=0;j<3;j++)
scanf("%d",&a[i][j]);
printf("\n Enter Second 3*3 Matrix Elements \n");
for(i=0;i<3;i++)
for(j=0;j<3;j++)
scanf("%d",&b[i][j]);
for(m=0;m<3;m++)
{
for(n=0;n<3;n++)
{
c[m][n]=0;
for(p=0;p<3;p++)
c[m][n]=c[m][n]+(a[a][p]*b[p][n]);
printf("\n Multiplication of Matrices:\n”);
printf(“\n Third Matrix Elements: \n”);
for(m=0;m<3;m++)
{
for(n=0;n<3;n++)
{
printf(“%d”,c[m][n]);
printf(“\n”);
}
}
```

To execute:

$gcc matrixmultiplication.c

$ ./a.out

Input

Enter First 3*3 Matrix Elements

1 2 3
1 2 3
1 2 3

Enter Second 3*3 Matrix Elements

1 1 1
2 1 2
3 2 1

Output
Multiplication of Matrices

Third Matrix Elements

14 9 8
14 9 8
14 9 8

Practical No. 8

AIM :- Learning of installation and upgradation of Linux operating system
TOOLS/LIABILITIES:- Bash shell, vi editor
S/W USED:- RedHat Linux 4.0
H/W USED:- Standard PC
PRECONDITION/INPUT:- Bootable RedHat CD
POSTCONDITION :- install RedHat linux

Installing Linux on a personal computer may not be as difficult as you think. This document explains how to install Linux on a PC, starting at the beginning: choosing a distribution

The first step for setting up Linux on a PC is the most time consuming, it is simply to use a run from CD version of all distributions that you are interested in to pick the version you want to install. Once you pick the distributions you want to check out, go to the distribution's Web sites and download the live *CD/move* ISO image for it. Then, using the method for your CD burning software, burn the ISO image to CDROM. With the CDROM in the drive, reboot the computer and you are now running the Linux version.

Once you have selected which distribution(s) you will install, you can go back to that distribution's Web site and download the install version of the distribution, which is most commonly made up of three CDROM ISO images. After these are downloaded, burn the ISO images to CDROM.

Delete the images from your hard drive after burning them to CDs.

You will then have to decide, before going any further, if you wish to multi-boot and keep Windows on the system. If you are going to keep Windows, you will have to run a defragmenting utility on the drive before going any further.

Completely cleaning up your hard drive, removing all temp files and defragmenting it before backing up the data is always a good idea, and backing up your system is recommended before installing a new operating system. Because Microsoft's NTFS is not 100 percent supported, this is even more important if you are running any NT-based version of Windows.

If you have chosen to use a "Boxed set" of Linux, there is a set of "DOS" tools on one of the CDs, which will run in Windows and allow you to resize the partition for Windows. If you are using a downloaded version, then you can use the Linux tools during installation, but you may have to completely delete the Windows partition. Getting a trial version of Partition magic will allow you to resize the Windows partition without losing the data.

The partition table structure for a multi boot system looks complex, at first, but is actually very simple:

- The first primary partition is your Windows partition.

- The first extended partition is a transfer partition for enabling read/write access to files from all operating systems, and needs to be either a FAT32 partition or a FAT16 partition
- The second extended partition is the first Linux partition, and should be set up in the Linux installation process. For only Windows and one Linux version, a 500MB partition is more than enough room. It will be given the label of */boot* [500MB in size]
- You will need to create a partition with the label / [5 GB in size]
- Create a partition with the label */home* [remaining amount of space for distribution]
- Create a partition with the label */usr* [5 GB in size]
- Create a partition with the label */var* [1 or 2 GB in size]
- Create a partition with the label */swap* [double your RAM in size]

When creating the partitions you will notice that you have a number of file system type options; the oldest one being the *ext2fs*. In a graphic installer, the file system types usually have an explanation of what they mean. The current default file system is the *reiserfs*, a journaled file system. This has a small hit on speed, but a major improvement on data protection, compared to a non journaled file system.

Speed and reliability for data input/output being important, the *reiserfs* or the *ext3fs* are the best supported file systems. If speed is more important than reliability, use the *ext2fs*. This does not mean major risk of data loss, but power fluctuations may cause some data loss or corruption, if you have spotty power, and do not have an interruptible power supply, go with a journalized file system.

Follow the prompts for the distribution you are installing. Each distribution has a different installation process, so detailing them here is not a viable option. If you are using one with a console */text* interface, the space bar selects, and the right arrow will expand a category, the left arrow will collapse a category, up and down arrows will move you up and down in the listing. The enter key will finish the selection process and start the installation. A graphic interface installation program will have mouse support.

Selecting a package

Package selection is a big part of installing Linux. When first trying Linux it doesn't hurt to install everything, as this enables you to see which applications you like and will use. There are far more options than the live CD versions can include, so while you have some idea about packages from having looked at the live CD, there are more options than those shown. Generally, there are over 10 thousand packages in a three CD version of a distribution, including the libraries for software development.

You may see a message screen after you have selected, or during the selection process, that lists a number of packages. This is a list of packages required by the one you just selected, called dependencies; you can just hit the ok button when you see it. Some installers allow for this notification to be turned off, which I would recommend against, as the dependencies will help you to see the different parts of each application.

During the installation you will be presented with a series of configuration options. The four most important ones are:

- **The firewall**--you should turn it on, and at this time you can choose what traffic will be allowed incoming from the Internet, if any.
- **The Display settings**--this will configure the graphic server for the GUI, and you should use the test configuration option when doing this.
- **The mouse**--it is a good idea to specify which connection and type, as the any USB or PS2 default of many distributions is not always a reliable mouse driver.
- **The Internet connection**--if you connect through a network card, then configure it as a network connection, not a DSL connection, even if you use DSL.

If you tell Linux it's a DSL connection then Linux looks for the DSL modem as a device in the computer and you will not get online. If you use dial up connection, and your modem is classed as a Winmodem, then you will need to get the drivers for it from the linmodems Web site before installing Linux. After installing Linux, you will have to compile the drivers for the modem, install them and run the network configuration tool for the distribution you have installed.

Bootloader

When the bootloader is installing, you have the option of setting which bootloader to use, GRUB or LILO, either is a stable option. You can also set which OS will be the default OS at boot. The screens will have a default box on them, if you wish to keep Windows as default then just select default when highlighting Windows (*note, it may be DOS with some distributions*). You can, if you want, delete the failsafe and nonframebuffer boot options, keeping only Linux, Windows and floppy options. (*Note, the floppy option is not always included with newer distributions*).
If you are installing several versions of Linux, the boot partition created during the first Linux installation will need to be selected and labelled as */boot* for each one. Make the same partitions for each version, except for swap, that you only need one. Do NOT use any other pre-existing Linux partition with different distributions, as the versions of software between the distros may cause conflicts when booting into them. By using the same boot partition, the bootloader installation step will read the data in it and configure the multi-boot for each new distribution. It may be useful to label each distro by its name in the boot menu as you go, this will make the boot options clearer when you are choosing one.
When updating the Red Hat EL or SLES operating system on nodes, if the *Mode* attribute is **PreManaged**, then after the operating system upgrade, the *Mode* attribute value is changed to **Managed**. If the *Mode* attribute is **Managed**, then it is not changed. If the *Mode* attribute is **MinManaged** before the operating system upgrade, then the *Mode* attribute value remains **MinManaged**.

If the **installnode** command fails for a **PreManaged** node, the attribute value of the node is **PreManaged** or **Installing**. If the **installnode** command fails for a **MinManaged** node, the attribute value of the node is **MinManaged** or **MinManaged-Installing**.

To upgrade the operating system and CSM on all the nodes, issue the following command:
```
installnode -a
```

The Red Hat EL nodes with the *InstallMethod* attribute set to **kickstart-upgrade** are rebooted and upgraded with the new level of the operating system, rebooted, and then CSM is updated if necessary. The Red Hat EL operating system upgrade runs asynchronously. Immediately after the operating system upgrade process is initiated (that is, when the node is rebooted), the **installnode** command exits, even though the operating system upgrade is not complete.

The SLES nodes with the *InstallMethod* attribute set to **you** are updated with the new level of the operating system, rebooted to pick up the new kernel, and then CSM is upgraded if necessary. The **installnode** command continues to run as the SLES operating system is updated. This may take a while to run. Then, **installnode** reboots the node, and exits. The rest of the upgrade process (CFM, SMS, **osupgradepostreboot** scripts, upgrading CSM) continues asynchronously.

You can use the -t flag on the installnode command to provide a timeout value in minutes. If you do not specify a value for *timeout*, the default is 60 minutes:

```
installnode -P -t timeout
```

If the operating system upgrade process does not complete within the timeout period specified, CSM considers the operating system upgrade process as failed. You can use the monitorinstall command to provide output information for the installation process.

After the operating system is updated, the following jobs run on the node:

1. The **osupgradeprereboot** customization scripts are run.
2. The node reboots to its local hard disk.
3. CSM is installed, along with the software listed in Planning for CSM for Linux.
4. The node's *Mode* attribute changes to **Managed**, or remains **MinManaged**.
5. SSH or RSH are set up on the node so that the node is accessible from the management server.
6. Any CFM files are transferred to the node.
7. SMS is run to install or update software, if it is configured.
8. The **osupgradepostreboot** customization scripts are run.

If you defined Kerberos options with the csmconfig command when you defined the management server, the **installnode** command sets up the Kerberos options for the cluster..

After an operating system upgrade the node BIOS boot order can remain:

1. diskette
2. CD-ROM
3. network
4. hard disk

Every time the node boots, it uses Dynamic Host Configuration Protocol (DHCP) to contact the management server or install server, which uses **pxelinux** to boot the node from its hard drive. Alternately, after the operating system upgrade is complete, you can change the boot order in the BIOS to the following:

1. diskette
2. CD-ROM
3. hard disk
4. network

Practical No. 9

AIM:- Study of Unix shells

A **Unix shell** is a command-line interpreter (see shell) and script host that provides a traditional user interface for the Unix operating system and for Unix-like systems. Users direct the operation of the computer by entering command input as text for a command line interpreter to execute or by creating text scripts of one or more such commands.
The most generic sense of the term *shell* means *any* program that users use to type commands. Since in the Unix operating system users can select which shell they want to use (which program should execute when they log in), many shells have been developed. It is called a "shell" because it hides the details of the underlying operating system behind the shell's interface (in contrast with the "kernel", which refers to the lowest-level, or 'inner-most' component of an operating system). Similarly, graphical user interfaces for Unix, such as GNOME, KDE, and Xfce can be called *visual shells* or *graphical shells*. By itself, the term *shell* is usually associated with the command line. In Unix, any program can be the user's shell. Users who want to use a different syntax for typing commands can specify a different program as their shell, though in practice this usually requires administrator rights

Types of shells

Bourne shell (sh)
C shell (csh)
TC shell (tcsh)
Korn shell (ksh)
Bourne Again SHell (bash)

Bourne shell (`sh`)

This is the original Unix shell written by Steve Bourne of Bell Labs. It is available on all UNIX systems. This shell does not have the interactive facilites provided by modern shells such as the C shell and Korn shell. You are advised to to use another shell which has these features.
The Bourne shell does provide an easy to use language with which you can write shell scripts.

C shell (`csh`)

This shell was written at the University of California, Berkeley. It provides a C-like language with which to write shell scripts - hence its name.

TC shell (`tcsh`)
This shell is available in the public domain. It provides all the features of the C shell together with `emacs` style editing of the command line.

Korn shell (`ksh`)

This shell was written by David Korn of Bell labs. It is now provided as the standard shell on Unix systems.
It provides all the features of the C and TC shells together with a shell programming language similar to that of the original Bourne shell.
It is the most efficient shell. Consider using this as your standard interactive shell.

Bourne Again Shell (`bash`)

This is a public domain shell written by the Free Software Foundation under their GNU initiative. Ultimately it is intended to be a full implementation of the IEEE Posix Shell and Tools specification. This shell is widely used within the academic commnity.
bash provides all the interactive features of the C shell (csh) and the Korn shell (ksh). Its programming language is compatible with the Bourne shell (sh).
If you use the Bourne shell (sh) for shell programming consider using bash as your complete shell environment

Practical No. 10

AIM :- Familiarize with grep and awk command.

grep command in Unix/Linux

The grep filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression (grep stands for global search for regular expression and print out).

Syntax:

grep [options] pattern [files]

Options Description

-c : This prints only a count of the lines that match a pattern
-h : Display the matched lines, but do not display the filenames.
-i : Ignores, case for matching
-l : Displays list of a filenames only.
-n : Display the matched lines and their line numbers.
-v : This prints out all the lines that do not matches the pattern
-w : Match whole word

Sample Commands

Consider the below file as an input

$cat > geekfile.txt

unix is great os. unix is free os. Unix linux which one you choose.

uNix is easy to learn.unix is a multiuser os.Learn unix.

1. Simple command: this command will display the line in our text file that contain the word "great".

Output:

$grep "great" geekfile.txt

unix is great os. unix is free os. Unix linux which one you choose.

2. Case insensitive search : The -i option enables to search for a string case insensitively in the given file. It matches the words like "UNIX", "Unix", "unix".

$grep -i "UNix" geekfile.txt

Output:

unix is great os.unix is free os. Unix linux which one you choose.

uNix is easy to learn.unix is a multiuser os.Learn unix.\

3. Displaying the count of number of matches : We can find the number of lines that matches the given string pattern

Output:

$grep -c "unix" geekfile.txt

4. Checking for the whole words in a file : By default, grep matches the given string/pattern even if it is found as a substring in a file. The -w option to grep makes it match only the whole words.

Output:
$ grep -w "unix" geekfile.txt
unix is great os.unix is free os. Unix linux which one you choose.

uNix is easy to learn.unix is a multiuser os.Learn unix.

5. Show line number while displaying the output using grep -n : To show the line number of file with the line matched.

$ grep -n "unix" geekfile.txt

Output:
1:unix is great os. unix is opensource. unix is free os.

2:uNix is easy to learn.unix is a multiuser os.Learn unix .

Awk command:

Awk is a scripting language used for manipulating data and generating reports. The awk command programming language requires no compiling and allows the user to use variables, numeric functions, string functions, and logical operators.

Awk is a utility that enables a programmer to write tiny but effective programs in the form of statements that define text patterns that are to be searched for in each line of a document and the action that is to be taken when a match is found within a line. Awk is mostly used for pattern scanning and processing. It searches one or more files to see if they contain lines that matches with the specified patterns and then perform the associated actions.

Awk is abbreviated from the names of the developers – Aho, Weinberger, and Kernighan.

Sample Commands

Example:
Consider the following text file as the input file for all cases below:

$cat > employee.txt

Mahir manager 45000

Riya clerk 25000

Pragya manager 50000

Amanpreet manager 47000

1. Default behavior of Awk: By default Awk prints every line of data from the specified file.
$ awk '{print}' employee.txt

Output:
Mahir manager 45000

Riya clerk 25000

Pragya manager 50000

Amanpreet manager 47000

2. Print the lines which match the given pattern.

$ awk '/manager/ {print}' employee.txt

Output:
Mahir manager 45000

Pragya manager 50000

Amanpreet manager 47000