Jawaharlal Nehru Engineering College

Laboratory Manual

Operating System LAB

For

FY MCA Students

12 January 2019 – Rev 00 – MCA – ISO 9001-2015 Tech Document

Author JNEC, Aurangabad

FORWARD

It is my great pleasure to present this laboratory manual for First year MCA students for the subject of Operating System keeping in view the coverage required for the concepts of Linux Operating System related to Installation of Linux operating system, basic concept of Linux, commands & utilities, editors, Shell script, installation of applications, backup, etc.

As a student, many of you may be wondering with some of the questions in your mind regarding the subject and exactly what has been tried is to answer through this manual.

As you may be aware that MGM has already been awarded with ISO 9001-2015 certification and it is our endure to technically equip our students taking the advantage of the procedural aspects of ISO 9001-2015 Certification.

Faculty members are also advised that covering these aspects in initial stage itself, will greatly relived them in future as much of the load will be taken care by the enthusiasm energies of the students once they are conceptually clear.

Dr. H. H. Shinde
Principal

LABORATORY MANNUAL CONTENTS

This manual is intended for the First year students of MCA branch in the subject of Operating System LAB. This manual typically contains practical/Lab Sessions related Operating System LAB covering various aspects related the subject to enhanced understanding.

Although, as per the syllabus, I have made the efforts to cover various aspects of Operating System LAB.

Students are advised to thoroughly go through this manual rather than only topics mentioned in the syllabus as practical aspects are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

Good Luck for your Enjoyable Laboratory Sessions

Nitin R. Yadav
MCA Department

DO's and Don'ts in Laboratory:

1. Do not handle any equipment before reading the instructions/Instruction manuals

2. Read carefully the power ratings of the equipment before it is switched on whether
   Ratings 230 V/50 Hz or 115V/60 Hz. For Indian equipments, the power ratings are Normally 230V/50Hz. If You have equipment with 115/60 Hz ratings, do not insert Power plug, as our normal supply is 230V/50 Hz, which will damage the equipment.

3. Observe type of sockets of equipment power to avoid mechanical damage

4. Do not forcefully place connectors to avoid the damage

5. Strictly observe the instructions given by the teacher/Lab Instructor

Instruction for Laboratory Teachers::

1. Submission related to whatever lab work has been completed should be done during the next lab session. The immediate arrangements for printouts related to submission on the day of practical assignments.

2. Students should be taught for taking the printouts under the observation of lab teacher.

3. The promptness of submission should be encouraged by way of marking and evaluation patterns that will benefit the sincere students.

Assignments List:-

1. Introduction to LINUX Operating System.
2. Installation of LINUX Operating System (Mint Flavor).
3. Study of general purpose utilities commands.
4. Study of user & session management commands.
5. Study of file system navigation commands, text processing tools, communication commands.
6. Study of VI editor.
7. Execute C & C++ programs in Linux.
8. Study of Shell Script.
9. Back up using TAR command.
10. Study of grep & awk commands.

Assignment – 1

(Practical Time: 2 Hours)

Aim - : Introduction to LINUX Operating System.

- Introduction to Linux Operating System.
- Fundamental Architecture of Linux.
- Feathers of Linux Operating System.
- Linux File Systems.
- Difference between Linux & Windows

Assignment – 2

Aim - : Installation of LINUX Operating System (Mint Flavor).

Installation Steps

Step 1: Create a live USB or disk

Go to Linux Mint website and download ISO file. This ISO file is the disk image that you can burn to a USB or DVD.

Download Linux Mint

Once you have downloaded the Linux Mint ISO, you need a tool to write the image to a disk. I recommend using a free tool called Universal USB Installer in Windows:

Download Universal USB Installer

It's an executable exe file. Just double click on it to run the software and browse it to the ISO. Make sure that you have your USB key plugged in.

Step 2: Make a new partition for Linux Mint

This is where you have to be cautious. If you have multiple partitions (not the recovery ones), you can either use one of them or create a new partition from an existing partition. Your existing data will be safe if you have enough free space. Typically, you install Linux in under 10 Gb, however, if disk space if not a concern, I advise using 30-40Gb at least. This way you can have more space at your disposal for downloading and keeping various files.

In Windows 10, go to start menu and type ‘partition’. This will bring up Disk Management utility. Now carefully select the disk in which you’ll make some free space by shrinking the volume.

Step 3: Boot in to live USB

Plug the live USB or disk into the computer and restart the computer. While booting the computer press F10 or F12 function key (defers from computer to computer) to go to the boot menu. Now, choose the option to boot from USB or Removable Media.

Important Note: If your computer came with Windows 8 or Windows 8.1 and you upgraded your system to Windows 10, you may have to disable secure boot. Most modern system with Windows 10 should not need this step, especially with Linux Mint or Ubuntu.

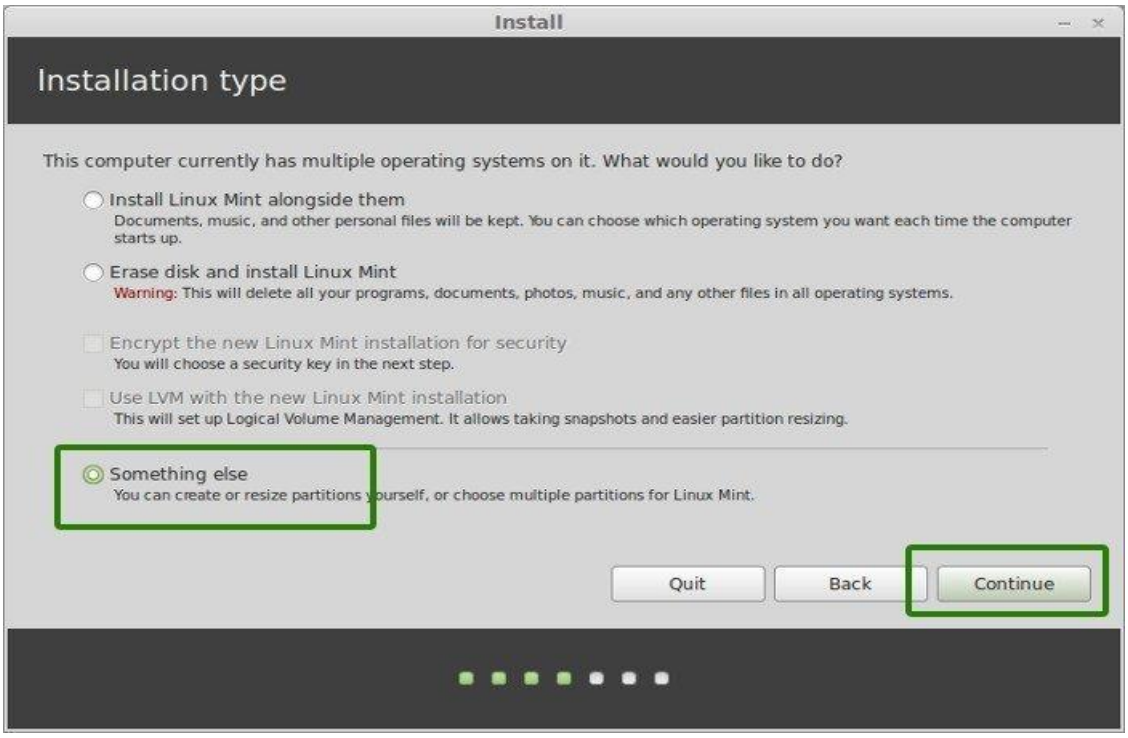
Step 4: Start the installation

It takes some time to boot from the live USB or disk. Have some patience. Once it boots in to live disk, you’ll be provided to Try Linux Mint or Install Linux Mint. Even if you choose to try it, you can find the install option on the desktop.

In next few screens, you’ll be asked to choose the language of the operating system. It will then do some checks on available space, battery and Internet connection.

Step 5: Prepare the partition

This is the most important part of the whole installation. Where to install Linux Mint? As mentioned before, I prefer separate partitions for Windows and Linux. Windows is already installed here, we’ll prepare a new partition for Linux Mint. In the Installation Type window, choose Something Else.



Step 6: Create root, swap and home

Since you already created a new partition in Windows, it's time to install Linux Mint on it. Now, there are several ways to do it. But here, I'll show you my favorite way and that is to have a Root, a Swap and a Home.

Create a root partition first. Choose the free space available and click on +.

Here, choose the size of the root (10 GB is enough but I chose to have 20 here), choose ext4 file system, and mount point as / (i.e. root).

Now, next is to create the swap partition. Now the question is what should be the swap size for Linux Mint installation?

The answer depends upon your RAM size, your needs, available disk space and whether you would use hibernation or not. You can use the below suggestion:

RAM less than 2 GB: Swap should be double the size of RAM

RAM between 2 to 4 GB: Swap should be RAM size + 2 GB

RAM between 6 GB to 8 GB: Swap should be size of RAM

RAM more than 8 GB: Swap should be half the size of RAM or less

Don’t spend too much time thinking about swap. It is helpful for systems with less memory. For system with more than 8 GB of RAM and SSD, the less the swap, the better it is.

The next step is to create Home. Try to allocate the maximum size to Home because this is where you’ll be downloading and keeping the files.

Once you have created Root, Swap and Home partitions, click on Install Now button.

Step 7: Follow the trivial instructions

Technically, you have crossed the main hurdle if you reached this point successfully. Now you will be taken through a number of screens to select options like keyboard layout, login credentials etc. You don’t need to be a genius to figure out what to do here afterward. I have attached screenshots for reference purpose here.

Once the installation is over, you will be presented with the option to keep trying live version or to restart the system.

Assignment – 3

Aim - : Study of general purpose utilities commands.

1 cal Command –

cal ↵

2 script Command –

script <file name> ↵

3 echo Command –

echo <write message> ↵

4 printf Command –

printf <write message> ↵

5 bc Command –

bc ↵

6 who Command –

who ↵

7 whoami Command –

whoami ↵

8 tty Command –

tty ↵

9 uname Command –

uname ↵

10 clear Command –

clear ↵

11 ls Command –

ls <options> ↵

-l : list the file in long format.

-t : list in order of last modification time.

-a : list all entries, including hidden file.

-d : list the directory file instead of contents.

-u : list in order of last access file.

-I : prints the inode number of file.

Assignment – 4

Aim - : Study of user & session management commands.

1 useradd Command –

useradd <option> <user name> ↵

-u : this creates user with unique identification numbers.

-d : this sets the home directory of the user.

-s : this sets login shell of user.

-g: it used to indicate users group.

2 groupadd Command –

groupadd <options> <group name>

3 userdel Command –

userdel <user name>

4 groupdel Command –

groupdel <group name>

5 passwd Command –

passwd <option> <user name>

-d : delete the password for particular user.

-l : lock the user account.

-u : unlock the user account.

6 chown Command –

chown <file/directory name>

7 chgrp Command –

chgrp <file/directory name>

8 su (Super User) Command –

su <option> <user name>

-l : login for the new user.

-c : execute command in the new shell and then exit immediately.

9 chmod Command –

chmod <option> <filename>

-u : User -g : Group -o : Other

r : Read w : Write x : Execute

Assignment – 5

Aim - : Study of file system navigation commands, text processing tools, communication commands.

1 cat Command –

cat > <file name> ↵

------write file text---------

Ctrl+z ↵

2 file Command –

file <file name> ↵

3 wc Command –

wc <options> <file name> ↵

-l : prints number of lines along with file name.

-w : prints number of words along with file name.

-c : prints number of characters along with file name.

4 cp Command –

cp <source file name> <destination file name> ↵

5 rm Command –

rm <file name> ↵

6 mv Command –

mv <file name> ↵

7 more Command –

more <file name> ↵

8 head Command –

↵

head <file name>

9 tail Command –

tail <file name>

10 gzip Command –

gzip <file name>

11 gunzip Command –

gunzip <file name>

Assignment – 6

Aim - : Study of VI editor.

VI editor is the default file editor in most of the Linux machines. It is having great capabilities to edit a file with in few key strokes.
Lets start with some general information and then move on to some good things what vi editor can do for you while editing a file.

1. Vi stands for visual.
2. Vi have its variants like vim which stands
   for Vi IMproved, VimX11for gui and winvi for MS windows.
3. Vi is the most popular editor and next most popular editor is gedit.
4. Some other editors which will do the work of editing files are neno, pico, gedit, emacs, vim, joe, nedit, ed etc.

In vi editors there are three basic modes

1. Command mode
2. Input mode
3. Ex mode or last line mode

In vi we used following commands

i – insert text to left of curser.

I – insert text at beginning of line.

a – append text to right of curser.

A – append text at the end of line.

o – opens line below.

O – opens line above.

R – replace text from curser to right.

s – replace text right character under curser with any number of characters.

S – replace entire line.

:x/:w – it is used to save and quit the editor.

:q – used to aborting editor.

:w – save file and remains in editing mode.

Navigation Commands

h - To move one character left.
j - To move one line down.
k - To move one line up.
l - To move one character right.

Assignment – 7

Aim - : Execute C & C++ programs in Linux.

To execute C program in linux operating system we use the gcc compiler. Write this program in any editor. For compilation of this program use gcc keyword. So we use following command.

gcc program name.c ↵

To execute C++ program in linux operating system we use the g++ compiler. Write this program in any editor. For compilation of this program use g++ keyword. So we use following command.

g++ program name.c++ ↵

We run both C & C++ program we use following command.

↵

./a.out

Execute any one C & C++ program in Linux.

Assignment – 8

Aim - : Study of Shell Script.

A Shell is unique multi-faceted program. It is command interpreter and a programming language rolled into one. A shell script is a group of commands in a single file. Shell provides features that enables in a single file. Shell provides features that enables it to be used as a programming language. The feature includes programming logical and conditional operators, command substitution, escape mechanism and position parameters. A set of commands to be performed repeatedly is executed using a batch file in MS-DOS shell script are similar to batch file shell script in Linux.

Creation of Shell script:

A set of commands to be performed can be entered into file by using any editor.

Execution of Shell Script:

A shell script can be executed using two methods

1 'sh filename' at # or $ prompt.

2 To grant the execution permission and type file name at # or $ prompt.

Eg. sh abc.sh

There are three types of shell variables

- User Define Variables
- Environment/Predefine Variables
- Local Variables

Programming Constructs:

1) if……then……else:

if condition

then

commands

else

commands

fi

2) for loop:

for variable in ` seq variable `

do

commands

done

3) while loop:

while commands

do

command list

done

4) case construct:

case variable in

choice 1) command;;

choice 2) command;;

…….

…….

*) command

esac

Assignment – 9

Aim - : Back up using TAR command.

Tape archive command (tar) has been is existence before the emergence of cpio command today it only not create archive on tapes but supports floppies as well.

Tar accepts directory on the command line –c key option it uses to copy file to the backup file. The verbose option (-v) shows the progress of the backup.

tar –cvf /dev/rdsk/file1.tar /home/demofile

Following commands are used with the tar command.

-c : create a new archive

-t : extract file from archive

-l : list content of archive

-r : append file at the end of tar file

-f (device) : used path name device as name of device instead of the default

-v : verbose option list file in the option

-w : conforms from the user action to be taken

-z : compress or uncompress with gzip files.

-i : compress or uncompress with bzip files.

To delete file from existing tar file

tar –f file1.taer –delete f3

Assignment – 10

Aim - : Study of grep & awk command.

grep = global regular expression print

In the simplest terms, grep (global regular expression print) will search input files for a search string, and print the lines that match it. Beginning at the first line in the file, grep copies a line into a buffer, compares it against the search string, and if the comparison passes, prints the

line to the screen. Grep will repeat this process until the file runs out of lines. Notice that nowhere in this process does grep store lines, change lines, or search only a part of a line.

Example data file

Please cut & paste the following data and save to a file called 'a_file':

boot

book

booze

machine

boots

bungie

bark

aardvark

broken$tuff

robots

A Simple Example

The simplest possible example of grep is simply:

grep "boo" a_file

In this example, grep would loop through every line of the file "a_file" and print out every line that contains the word 'boo':

boot

book

booze

boots

Useful Options

This is nice, but if you were working with a large fortran file of something similar, it would probably be much more useful to you if the lines identified which line in the file they

were, what way you could track down a particular string more easily, if you needed to open the file in an editor to make some changes. This can be accomplished by adding the -n parameter:

grep -n "boo" a_file

This yields a much more useful result, which explains which lines matched the search string:

1:boot

2:book

3:booze

5:boots

Another interesting switch is -v, which will print the negative result. In other words, grep will print all of the lines that do not match the search string, rather than printing the lines that match it. In the following case, grep will print every line that does not contain the string "boo," and will display the line numbers, as in the last example

grep -vn "boo" a_file

In this particular case, it will print

4:machine

6:bungie

7:bark

8:aaradvark

9:robots

The -c option tells grep to supress the printing of matching lines, and only display the number of lines that match the query. For instance, the following will print the number 4, because there are 4 occurences of "boo" in a_file.

grep -c "boo" a_file

4

The -l option prints only the filenames of files in the query that have lines that match the search string. This is useful if you are searching through multiple files for the same string. like so:

grep -l "boo" *

An option more useful for searching through non-code files is -i, ignore case. This option will treat upper and lower case as equivalent while matching the search string. In the following example, the lines containg "boo" will be printed out, even though the search string is uppercase.

grep -i "BOO" a_file

The -x option looks for exact matches only. In other words, the following command will print nothing, because there are no lines that only contain the pattern "boo"

grep -x "boo" a_file

Finally, -A allows you to specify additional lines of context file, so you get the search string plus a number of additional lines, e.g.

grep -A2 “mach” a_file

machine

boots

bungie

Regular Expressions:

A regular expression is a compact way of describing complex patterns in text. With grep, you can use them to search for patterns. Other tools let you use regular expressions (“regexps”) to modify the text in complex ways. The normal strings we have been using so far are in fact just very simple regular expressions. You may also come across them if you use wildcards such as '*' or '?' when listing filenames etc. You may use grep to search using basic regexps such as to search the file for lines ending with the letter e:

grep "e$" a_file

This will, of course, print

booze

machine

bungie

If you want a wider range of regular expression commands then you must use 'grep -E' (also known as the egrep command). For instance, the regexp command ? will match 1 or 0 occurences of the previous character:

grep -E "boots?" a_file

This query will return

boot

boots

You can also combine multiple searches using the pipe (|) which means 'or' so can do things like:

grep -E "boot|boots" a_file

boot

boots

Special characters

What if the thing you want to search for is a special character? If you wanted to find all lines containing the dollar character '$' then you cannot do grep '$' a_file as the '$' will be interpreted as a regexp and instead you will get all the lines which have anything as an end of line, ie all lines! The solution is to 'escape' the symbol, so you would use

grep '\$' a_file

broken$tuff

You can also use the '-F' option which stands for 'fixed string' or 'fast' in that it only searches for literal strings and not regexps.

AWK command in Unix/Linux with examples

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 performs the associated actions.

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

WHAT CAN WE DO WITH AWK ?
1. AWK Operations:
(a) Scans a file line by line

(b) Splits each input line into fields
(c) Compares input line/fields to pattern
(d) Performs action(s) on matched lines

2. Useful For:
(a) Transform data files
(b) Produce formatted reports

3. Programming Constructs:
(a) Format output lines
(b) Arithmetic and string operations
(c) Conditionals and loops

Syntax:

```
awk options 'selection _criteria {action }' input-file > output-file
```

Options:

```
-f program-file : Reads the AWK program source from the file
                  program-file, instead of from the
                  first command line argument.
-F fs           : Use fs for the input field separator
```

Sample Commands

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

```
$cat > employee.txt
ajay manager account 45000
sunil clerk account 25000
varun manager sales 50000
amit manager account 47000
tarun peon sales 15000
deepak clerk sales 23000
sunil peon sales 13000
satvik director purchase 80000
```

1. Default behavior of Awk : By default Awk prints every line of data from the specified file.

```
$ awk '{print}' employee.txt
```

Output:

```
ajay manager account 45000
sunil clerk account 25000
varun manager sales 50000
amit manager account 47000
tarun peon sales 15000
deepak clerk sales 23000
sunil peon sales 13000
satvik director purchase 80000
```

In the above example, no pattern is given. So the actions are applicable to all the lines. Action print without any argument prints the whole line by default, so it prints all the lines of the file without failure.

2. Print the lines which matches with the given pattern.

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

Output:

```
ajay manager account 45000
varun manager sales 50000
amit manager account 47000
```

In the above example, the awk command prints all the line which matches with the ‘manager’.

3. Spliting a Line Into Fields : For each record i.e line, the awk command splits the record delimited by whitespace character by default and stores it in the $n variables. If the line has 4 words, it will be stored in $1, $2, $3 and $4 respectively. Also, $0 represents the whole line.

```
$ awk '{print $1,$4}' employee.txt
```

Output:

```
ajay 45000
sunil 25000
varun 50000
amit 47000
tarun 15000
deepak 23000
```

```
sunil 13000
satvik 80000
```

In the above example, $1 and $4 represents Name and Salary fields respectively.

Built In Variables In Awk

Awk's built-in variables include the field variables—$1, $2, $3, and so on ($0 is the entire line) — that break a line of text into individual words or pieces called fields.

NR: NR command keeps a current count of the number of input records. Remember that records are usually lines. Awk command performs the pattern/action statements once for each record in a file.
NF: NF command keeps a count of the number of fields within the current input record.
FS: FS command contains the field separator character which is used to divide fields on the input line. The default is "white space", meaning space and tab characters. FS can be reassigned to another character (typically in BEGIN) to change the field separator.

RS: RS command stores the current record separator character. Since, by default, an input line is the input record, the default record separator character is a newline.
OFS: OFS command stores the output field separator, which separates the fields when Awk prints them. The default is a blank space. Whenever print has several parameters separated with commas, it will print the value of OFS in between each parameter.
ORS: ORS command stores the output record separator, which separates the output lines when Awk prints them. The default is a newline character. print automatically outputs the contents of ORS at the end of whatever it is given to print.

Examples:

Use of NR built-in variables (Display Line Number)

```
$ awk '{print NR,$0}' employee.txt
```

Output:

```
1 ajay manager account 45000
2 sunil clerk account 25000
3 varun manager sales 50000
4 amit manager account 47000
5 tarun peon sales 15000
6 deepak clerk sales 23000
7 sunil peon sales 13000
```

```
8 satvik director purchase 80000
```

In the above example, the awk command with NR prints all the lines along with the line number.

Use of NF built-in variables (Display Last Field)

```
$ awk '{print $1,$NF}' employee.txt
```

Output:

```
ajay 45000

sunil 25000

varun 50000

amit 47000

tarun 15000

deepak 23000

sunil 13000

satvik 80000
```

In the above example $1 represents Name and $NF represents Salary. We can get the Salary using $NF , where $NF represents last field.

Another use of NR built-in variables (Display Line From 3 to 6)

```
$ awk 'NR==3, NR==6 {print NR,$0}' employee.txt
```

Output:

```
3 varun manager sales 50000

4 amit manager account 47000

5 tarun peon sales 15000

6 deepak clerk sales 23000
```