MANNUAL & ASSIGNMENT

FOR

OPERATING SYSTEM

2nd YEAR IT and IoT

Session – I

Part -1

Now you type the following commands

Command	Description
ls	This command will display all file & directory names. Like dir command in MS-DOS
ls a*	This command will display all the file name which starting with a.
ls –l	An information listing of files is obtain when ls is used with the –l(long) option (discuss details in **session –1 part – 7**)
ls –l a*	This command will display all the file name which starting with a with long listing.
ls –d	Forces listing of a directory.
ls –a	Show all files including hidden file.

You can get help from man command.
Syntax : ls man

Part –2

cat – Command

Through the cat command you can create, append new data and display the contain of a file. This command work like **copy con & type** command in MS-DOS.

Syntax :

i. $*cat > file_name* (it is better that you should type your name) for create a blank file. After press ENTER you see a blank screen. You type here. Press **ctrl+d** after ending the writing. ii. $*cat file_name* - To display the contain of a file.

iii. $*cat file_name1 file_name2* – In this case it will display the **contain** of two files one after another.

iv. $ *cat>> file_name* - In this case you can append data in a old file. Press **ctrl+d** for save and exist.

Part –3

mkdir – This command helps to create the directory.
Syntax – $mkdir dir_name

cd - This command helps to change directory.
Syntax –

a) $cd dir_name
b) $cd ..

pwd – This command helps to display the current path.
Syntax: $ pwd

rmdir – This command helps to remove directory.
Syntax: $rmdir old_dirname

Part –4

more – more is **not** a primary command it works **on a primary command** like ls, cat etc.

Example :

a) $ls -l| more – In that case the list will display page by page. b) $cat file_name – In that case the contain of a file will display page by page.

Part –5

cp – This command is create duplicate file/files form one location to another.
Syntax: $cp [source_dir_name]/<source_file_name>

[/destination_dir_name/][destination_file_name]

Example :

a) You will copy a file (assume the name ram.c) in same directory, you may assume your location is /root/imt3rd
$ cp ram.c shyam.c – in that case the ram.c file is copied in shyam.c name

b) You will copy a file (assume the name ram.c) in other directory (assume the name of directory is rahul) , let the location of the file is /root/imt3rd
$ cp ram.c /root/imt3rd/rahul – in that case the ram.c file is copied in rahul directory in ram.c name.

Part –6

mv – This command helps to rename a file/directory. Through the command you move a file in one location to other.
Syntax: $mv [source_dir_name]/<source_file_name>
[/destination_dir_name/][destination_file_name]

Example :

c) You will move a file (assume the name shyam.c) in same directory, you may assume your location is /root/imt3rd **$ mv shyam.c jojo.c** – in that case the shyam.c file is renamed in jojo.c name

d) You will move a file (assume the name ram.c) in a other directory (assume the name of directory is rahul) , let the location of the file is /root/imt3rd
$ mv jojo.c /root/imt3rd/rahul – in that case the jojo.c file is moved in rahul directory in jojo.c name

Part –7

When you give the command ls –l you see eight columns and several rows.

In first column character 1 show either **–** or **d** . – means it is file and **d** means it is directory.

After that you find 9 character set. It is either **r** or **w** or **x** or **-** . This is the permission set. It has three parts for all three categories. The first part belongs to ***u*ser/owner** .The second part belongs to ***g*roup** and third for ***o*ther**. All the parts have three characters.

r – means **R**ead permission
w – means **W**rite permission

x – means e**X**ecute permission
- - means no permission.

You can change the mode of permission by **chmod** command.
Syntax : chmod <category><operation><permission><filename/s>
Example :
ls –l fact.c - It will display –rwxr-xr-- …… like that. That means **u**ser/owner has permission to read, write, executable. **g**roup has read and executable **no** write permission. **o**ther has **only** read permission.
You want to give the permission to all read, write, execute. The command is
$chmod u=rwx,g=rwx,o=rwx fact.c

You may use octal number. The read permission assign 4, write permission assign 2 and executable permission 1. If you give all permission of a particular category you write (4+2+1) 7. $chmod 777 fact.c – Through the command you can give all the permission to all the category.

Assignment for Session 1

1. Display all the files with long listing whose first two character is **ca.**
2. Create a file the name of the file is first_<roll_no> and write Your name, address, phone no. and save it.
3. Display the contain of the file .
4. Make a directory name as<roll_no>
5. Copy the file to the directory.
6. Output of Assig. 1 store in a file name as1<roll_no>. 7. Move the as1<roll_no> file to as<roll_no> directory. 8. Change the permission owner only read, write. Execute but group and other only read the file.

Session – II

Part – 2.1

wc - Through this command you can count the word, line and character of a file.
Syntex : wc [option] < filename

Option : -l = To count the line of a file.
-c = To count the character of a file.
-w = To count the character of a file.
Example : wc myfile will show the 3 number the first is line, second is

word and third is word.

who – Through the command you can see who are current logged in and also you know the terminal number.

tee- tee uses standard input and standard output, which means that it can be placed anywhere in a pipeline. The additional feature it processes is that it breaks up the input into two components; one component is saved in a file which is used as an argument to command and there is simple connected to the standard output.
Example:
who | tee curLo
In the above example the value of who command will store in the file **curLo** and parallels this command will display the output in the screen.
Part – 2.2
cmp – This command compare files byte by byte.
Example :
cmp file1 file2

Part – 2.3

echo – Through the command you can display the message. *date* – Through date command you can display the current date.

$echo The date today is :date
The date today is
Sun Aug 01 10:10:23 EST 2004
$_
command substitution offers the facility of doing this in a single statement. Use the expression `date` as an argument to echo:

$echo The date today is `date`
The date today is Sun Aug 01 10:11:55 EST 2004
$_

$**echo There are `ls | wc –l` files in the current directory**
There are 12 files in the current directory.

$**echo** 'Enter your name :\c'
Enter your name :
read - Through the command you can input through key board.
Syntax : read <vername>
$echo 'Your name is\c $vername'

Part – 2.4

Variable – You can assign variable like any language. The variables are assign with = operator.
Example :

a. $x=50
   $echo $x
   50
   $_
b. $msg = 'Hello Everybody'
   $echo $msg
c. $msg = Hello/Everybody
   $echo $msg

You can add variables

$x=ram
$y=kumar
$echo xy

You can use following purpose
$pa='/home/imt3rd/kamal'
$cd $pa
$pwd

Part – 2.5

Vi Editor – It is an editor. You can write any thing through vi editor. When you want to write/append some thing you press ESC+a or ESC+:+wq

The **vi** editor have three mode

③ Input mode – Where any key depressed is entered as text ③ Command Mode- Where keys are used as command to act on text
③ ex Mode – Where ex commands can be entered in the last line to act on text.

Some input commands:

Command Action

i Insert text to the left of cursor
I Inserts text at beginning of line
a Appends text to right of cursor
A Appends text to end of line
o Opens line below
O Opens line above
R Replace text form cursor to right
s Replace single character under cursor with any number of character
S Replace enter line

Some delete commands:
dd Delete entire line
x Delete a single character.

Some cursor movement commands:
Command Action
h or backspace Moves cursor left
j Moves cursor down
k Moves cursor up
l or spacebar Moves cursor right
^ Moves cursor beginning of first word of the line $ Moves cursor to end of line
b Moves cursor back to beginning of word. e Moves cursor forward to end of end of word. w Moves cursor forward to beginning of word
Saving and quitting
:w – Save
:x – Save & Exit
:wq- Save & Exit
:q – Quits editing mode when no changes made in file
:q! – Quits editing mode but after abandoning changes.

Yanking text

The **y** operator yanks (or copies) text. The principles of its use are exactly the same as with the **d** and **c** operators. You can yank a word, a group of words, line segments or even entire lines with this operator. Moreover, the **p** and **P** commands act in the same way for putting the copied text at its destination.

For instance, to yank five lines of text, move the cursor to the first of these and press **5yy** or **5y**

Next, move the cursor to the new location and press
p or **P**

to place the copied text below or above the current line, respectively. When you young words, **p** and **P** have the same significance as with the **d** operator ; **p** puts text on the right of cursor location, while **P** puts it on the left.

Assignment for session - 2

2.1 Count the no. of files in a directory. The result will store in a file name roll_stream and also display in the screen parallels. 2.2 Copy the 2.1 's output file in the same directory the name will d_roll_stream.

Compare the two files and also save it in a file name will roll_stream_err

2.3 Store your first_ name in variable first and also store your last name in a variable last through keyboard. Now add two variables and display it.

2.4 Create a file through vi name of the file is roll_stream.dat. Write all the steps of the following situation.

2.4.1 The contain must be name|phone|roll|stream of your friends.

2.4.2 Save the file.

2.4.3 Save and exit from vi

2.4.4 Open the file roll_stream.dat

2.4.5 Exit without saving.

2.4.6 Edit a name and save it.

2.4.7 Cut the first line and paste it to last.

Session – III

Continuation of ***Session I part 1.1***

When you type ls –l in the first line it will display *"total ...."* . It is total a number. Which indicates that a total no blocks are occupied by these files in the disk, each block consisting of 512 bytes.

Part – 3.1

Create a file name stud_roll.dat with the help of **vi** . The contain of that file is as follows.

01/S/002|Smair Das |3rd|CSE
01/V/002|Vinay Dev|3rd|IT
……
……
……

The above file you can treated like a database. The field separation is |(pipe). In unix it is default separation. This pipe we called DELEMETER. You can use any ASCII character.

pr- Through the command you can display a file with suitable header, footer and formatted text.

Syntax : pr <filename>

Example : pr stud_roll.dat

By default the length of a page is 66 lines, but page (printing page) is 72 line. You can increase the size of a page by **–l** (**L**ength) option Syntax : pr –l72 stud_roll.dat

-w option is helps to increase the size of width.

$pr –w132 stud_roll.dat

-h – To add heading through the command.

$pr –h "Student's Report " stud_roll.dat
-n option for line number.
-t option for omit them totally.
-d double space output.
-m<Filename> Merges two files.
-s char Use delimiter *char* to merge files

Part – 3.2

head Command- This command will displays the top of the file.

Syntax : head <filename>

Example : head stud_roll.dat
$head –3 stud_roll.dat

In that case it will display the first three lines.

Part – 3.3

tail Command- This command will displays the end of the file.
Through the **head** & **tail** command you can cut a file **horizontally**.

Example :

a. tail –3 stud_roll.dat- In that case it will display the last three lines
b. tail +3 stud_roll.dat – In that case it will display after first three lines.
c. tail –10c stud_roll.dat – In that case it will display the last 10 characters.

Part – 3.4

cut command - Through the command you can slice a file vertically. The command identifies both columns and fields and is a useful filter for

the programmer.

Syntax : cut <options><character or filed list> <file/s>

$cut –c10-21,26- stud_roll.dat

You can cut a file through using field separation delimiter.

Example :
$cut –d" | " –f1,2,4 stud_roll.dat

You can store the output of the above example in a file name cut_roll.dat

Part – 3.5

paste command- Through the command you can paste more than one file vertically.

Example :
$paste stud_roll.dat cut_roll.dat

In the above example both the files added vertically. The joining place of the two files is blank. If you added it with a delimiter then you use the following example.

$ paste –d" | " stud_roll.dat cut_roll.dat

You can use cut & paste command at a time.

Part – 3.6

sort command – Through the command you can sorts entire line by according to ASCII coding.

Example :
$sort cut_roll.dat

You can sort field wise by using –t option
$sort –t" | " +1 cut_roll.dat

The above example is sort according to first field. If you use +2 in replace of +1 it works on 2nd field.

The –r option for reverse sorting of the database.

The –o option for store the output of the command in a file.
Example :

$sort –o sort_roll.dat stud_roll.dat

The uniq option is eliminate the duplicate record.
$ uniq sort_roll.dat

Number sort
Create a file through vi and the contain of the file is 210,101,22 ,50,301…. One number should be one line
$sort num_roll
It should be in ASCII character pattern. In 101,210,,22,301,50 this pattern .

If you want to sort it in numeric pattern. –n option then you sort numeric pattern.

Example :
$sort –n num_roll

Part – 3.7
nl command – This command will display a file with line number.
Example :
$nl stud_roll.dat

You can use –w(width) option to specify the width and –s option to specify the separator and –n option is for specify the leading zeros. The rz option for right justifies the gap. You can use ln for left justify and removing zeros.

Example :
$nl –w3 –s"|" stud_roll.dat
$nl –w3 –s"|" –nrz stud_roll.dat

Assignment for session - 3
3.1 Create a file with vi name mstud_roll.dat. The fields are enrlno, regno, name, phone_no. Also create file tstud_roll.dat the fields are regno, examroll, total of 2nd year 2nd sem. The number of records of the both file must be same. The delimiter of those files mast be "|". The number of records not less than 5 records. Now you display a report according to regno, examroll, name, phone_no, total and also save the file in roll_res.dat.

3.2 Display the roll_res.dat file with heading like "The result ".

3.3 Now you sort the roll_res.dat and store it s_roll_res.dat. The field must be sort on total.

3.4 Now you generate a tstud_roll.dat . The contain of the file is only top three records of s_roll_res.dat

3.5 Now you generate a estud_roll.dat . The contain of the file is only last three records of s_roll_res.dat

3.6 Now you add tstud_roll.dat and estud_roll.dat. If there is any duplicate record eliminate it and stored it in a fre_roll.dat file.

Session – IV

Part – 4.1

Pattern Matching – Through **grep** Command

Through the grep command you can match pattern from a file/s. The query feature lets you locate those lines which contain an expression. UNIX has a special feature, and the principle number of this family is the grep command. This command which referred to as the **g**lobal **r**egular **e**xpression **p**rinter, performs mainly this job.

Syntax : grep <options><pattern><file/s>

From the **Session – III** *stud_roll.dat*

$grep CSE stud_roll.dat

In the above case it will return the record who are in CSE department. You may use "CSE" but it is mandatory if in the pattern have any white space(blank, tab etc) like "Ram Das".

$grep 'IT' stud_roll.dat mstud_roll.dat

In the above case it will return the records who are in IT department from the two file(stud_roll.dat, mstud_roll.dat).

$cat stud_???.dat | grep "3rd"
$grep –h "3rd" stud_???.dat

Both the command is same. It will return the records who are in 3rd from stud_001.dat, stud_002.dat, stud_003.dat …….stud_n.dat files.

$ver1="Ram Das"
$grep "$ver1" stud_roll.dat

Part – 4.2

Options :
-c Display count of the number of occurrences -l Display list of file names only
-n Display line number along with the lines -v Display all but the lines matching the expression -i Ignore case for matching
-h Omits file names when handling multiple files

$grep –c stud_roll.dat
In the case it returns the number of occurrence.

$grep –n "3rd" stud_roll.dat

You can use also
$grep –ln "3rd" *.dat

Part – 4.3
Regular Expression:

Expression	Description
ch*	Matches zero or more occurrences of the previous character ch
[pqr]	Matches a single character p, q or r
[c1-c2]	Matches a single character within the ASCII range represented by the characters c1 and c2
[^pqr]	Matches a single character which is not a p, q or r
^ptn	Matches the pattern ptn at the beginning of a line
ptn$	Matches the pattern ptn at the end of a line

Exercise:
4.1 Find all names who's starting with s or S
4.2 Find all names which not starting with s or S.
4.3 Find all the names who's average marks is 50% to 70%

Session – V

(For two week)

Part – 5.1

The activities of the shell not restricted to command interpretation

alone; they involve much more than that. The shell also has rudimentary programming features which, coupled with the use of UNIX commands and other programs, make it an extremely useful programming language.

The shell program is a series of UNIX command which will execute one after another.

The extension of shell file is **.sh**

Through vi editor you can can create shell file.

Example 5.1;
File name fi_roll.sh – Contain is following
```
echo "\nEnter your name :\c"
read tname
echo "Listing is start"
read pause
ls -l $tname
```

Now you save the file.

How to run the file ?
```
$sh fi_roll.sh
```

Exercise
5.1 Write a shell programe to display a calendar of a particular month. The value of the month should be inputted through keyboard. 5.2 Write a shell program to enter your first name and last name separately through keyboard and display it in full format. 5.3 Write a shell program to find a pattern form a file. The pattern and name of file will given through keyboard.
5.4 Write a shell program to display first n line of a file and the value of n will given through keyboard.

Part – 5.2

Like C you pass value through command line argument. The first argument referred by $0. $# is count the number of arguments. $* will display the arguments.

Example 5.2
```
echo "Program :$0
The number of arguments specified is $#
The arguments are $*"
grep "$1" $2 || echo "Pattern Not found"
echo "\nJob Over"
```

The 2nd last line of the above example will return the message "Pattern not found" if the pattern not found in the file.

Part – 5.3

The **if** conditional

```
if <condition>
then
 <statement/s>
else
 <statement/s>
fi
```

test – When you utilize **if** to evaluate expressions, the test statement is invariably used as its control command. On most UNIX system, it is an internal feature of the shell. **test** evaluates the condition placed on its right, and returns either a true or false exit status. This return value is used by **if** for taking decisions.

Relational operators used by **if**
-eq Equal to
-ne Not equal to
-gt Greater than
-ge Greater than and equal to
-lt Less than
-le Less than and equal to
Example 5.3

```
if test$# -ne 3 #3 arguments not entered
then
 echo "You have not keyed in 3 arguments"
 exit 3
else
 if grep "$1" $2>$3
 then
 echo "pattern found – job over"
 else
 echo "Pattern not found – Job over"
 fi
fi
```

Example 5.4

```
echo “Enter the value of x:”
read x
echo “Enter the value of y:”
read y
if test $x –ge $y
then
 echo “x is greater than y”
else
 echo “y is greater than x”
fi
```

Exercise

5.5 – Write a shell program two find the greatest number among the three numbers, which will inputted through command line and also check the argument must be 3. If it is not 3 then give error message.

expr- The shell has no computing feature at all; it has to rely on the **expr** command for this purpose. This command combines two functions in one – it can perform arithmetic operations on integers and also manipulate.

Example 5.5

```
echo “Enter the value of x:”
read x
echo “Enter the value of y:”
read y
a=`expr $x + $y`
echo “The sum is $a”
```

case - In replace of multiple if you can use case

```
case <expression> in
 <pattern 1> <execute commands>;;
 <pattern 2> <execute commands>;;
 <pattern 3> <execute commands>;;
easc
```

Example 5.6

```
echo “Menu”
echo “1. List”
echo “2. Long List”
echo “3. Date “

echo “Enter your option :”
read choice
```

```
echo
case "$choice" in
     1) ls ;;
     2) ls –l;;
     3) date;;
ease
```

while –

```
while <condition is true>
do
 <command>
done
```

Example 5.7

```
x=5
while [ $x –gt 3 ]
do
 echo "hi"
done
```

Exercise

5.6 Write a shell program to calculate the sum of n numbers. The value of n will given through keyboard.

5.7 Write a shell program to find out the highest temperature among the n number. The value of n and temperature will given through keyboard.

5.8 Create a menu. The contain of the menu is i. add ii. Edit iii. Delete iv. Report v. Under the report another sub menu a. Display all Records b. Display Batchwise c. Display a particular record. When you press 1 in main menu add new records in the file. The name of the file is roll_str.lst . The fields are roll, name, stream . The field separator is | 'pipe'. The roll is auto generate, unique and after entering a record it gives a prompt do u want to continue ? If the user enter press 'y' then the addition program will automatically execute.

In the case of Edit and Delete the process wants to input roll from user. In the case of edit the record will update and delete case the record will erase.

Under the report menu there is another sub menu which is discuss earlier. The first submenu is display all record. The send case is to enter the stream name which will given through keyboard and display that record and the third sub menu is - to enter the roll of a student display the status of the particular student.