VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be University)
-Estd. u/s 3 of UGC Act 1956

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

OPERATING SYSTEMS (19CS211)

R19 Regulation Lab Manual

IIIB. TECH ISEM

Academic Year 2021-2022

PREFACE

Pre-requisite: C/Data Structures

About the course / Lab:

Operating systems are the fundamental part of every computing device to run any type of software. The increasing use of computing devices in all areas of life (leisure, work), lead to a variety of operating systems. Yet all operating systems share common principles. These principles are important for computer science students in their understanding of programming languages and software built on top of operating systems. The Operating System Laboratory, OS Lab is a course that will teach students about principles of operating systems using a constructivist approach and problem-oriented learning.

Importance:

It is really important to have an operating system as it is a program that manages a computer's hardware because it acts as an intermediary between the user of the computer and the computer hardware. It controls the hardware and manages its use between several application programs for the numerous users. This course will examine different pertinent subtle elements regarding the operating systems for example, such that those viewpoints to working systems, those capacities about operating systems, those parts about working systems, sorts of operating system, characteristics from claiming operating system, illustrations about registering units which utilizes operating system, illustrations of registering gadgets which doesn't use working framework.

Those principle point from claiming client to see a boost and to fill a minimized exertion of the client. Practically systems are designed to be worked by a solitary client. Starting with those system's side of the point for view, working system is a system that is included with those fittings. Operating system will be an allocator which allocates memory resources in the middle of various processes and it also keeps improper usage, deficiency and handles deadlock conditions. The capacities from claiming operating system is that it manages computer's resources, it gives client interface, it runs provision for client's camwood utilizes various projects during the same long drag known as multiprogramming. Another name for multiprogramming may be multitasking. Operating systems additionally helps support utility programs and it also controls computer hardware. There are vast number of components of operating system and three components that will be discussed about. Those three components are process management, memory management and client interface. A basic function of process management is that it allocates resources that will empowers forms to offer and trade

data. Process management protects the resources to every process starting with other methods and more empowers synchronization around processes.

They are constant operating system, multi-user operating system, multitasking. Constant operating systems needs assistance outlined to situations which are embedded, for example, purchaser devices, automobiles What's more mechanical technology. A multi-user operating system permits various clients to utilize same workstation towards those which are same in the long run besides actually diverse times. A multiprocessor backs what's more used, more amazing over particular case processor. Multitasking permits a significant number product forms which runs in the same way as the long drag inasmuch as multithreading permits distinctive parts of a programming system will run all the while. The offers from claiming working framework may be that it is operational in the least times, dependable, furthermore it also enhances time sharing. Samples for registering gadgets which utilizes operating system would be computers, portable phones, 3d televisions, video games, automated teller machine (ATM) and more ticket wending machine. Samples of registering units which doesn't utilize operating systems need aid velocity machines, calculators, washing machines, micro ovens and digital

VFSTR Vision and Mission

Vision

ToevolveintoacenterofexcellenceinScience&Technologythroughcreative and innovative practices in teaching - learning, towards promoting academic achievement and research excellence to produce internationally accepted, competitive and world class professionals who are psychologically strong & emotionally balanced imbued with social consciousness & ethicalvalues.

Mission

To provide high quality academic programes, training activities, research facilities and opportunities supported by continuous industry - institute interaction aimed at promoting employability, entrepreneurship, leadership and research aptitude among students and contribute to the economic and technological development of the region, state and nation.

Department Vision and Mission

Vision

ToevolveasacenterofhighreputeinComputerScience&Engineeringandcreate computer software professionals trained on problem solving skills imbued with ethics to serve the ever evolving and emerging requirements of IT Industry and society atlarge.

Mission

- Imparting quality education through well designed curriculum, innovative teaching and learning methodologies integrated with professional skill development activities to meet the challenges in the career.
- Nurture research and consultancy activities amongst students and faculty by providing State-of-art facilities and Industry-InstituteInteraction.
- Developing capacity to learn new technologies and apply to solve social and industrial problems to become anentrepreneur.

Programme Educational Objectives (PEOs)

PEO1: Pursue successful professional careerin IT and IT-enabled industries.

PEO2: Pursue lifelong learning in generating innovative engineering solutions using research and complex problem-solving skills.

PEO3: Demonstrate professionalism, ethics, inter-personal skills and continuous learning to develop leadershipqualities.

Programme Specific Outcomes (PSOs)

PSO1: Application Development Skills: Design and development of web applications using various technologies such as HTML, JSP, PHP, ASP and ASP.NET to caterthe needs of the society

PSO2: Enrich Research Skills: Offer solutions which impact geo-socio-economic and environmental scenario by using Machine Learning, Artificial Intelligence and IoT.

Programme Outcomes (POs)

Program Outcomes (POs), are attributes acquired by the student at the time of graduation. The POs given in below, ensure that the POs are aligned to the Graduate Attributes (GAs) specified by National Board of Accreditation (NBA). These attributes are measured at the time ofGraduation.

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of
complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using firstprinciples of mathematics, natural sciences, and engineeringsciences.

PO3:Design/developmentofsolutions: Designsolutionsforcomplexengineering

problemsanddesignsystemcomponentsorprocessesthatmeetthespecifiedneeds

with appropriate consideration for the public health and safety, and the cultural,societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledgeand research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modellingto complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and theconsequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate theknowledge of, and need for sustainabledevelopment.

PO8: Ethics: Apply ethical principles and commit to professional ethics andresponsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as amember or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation,makeeffective presentations, and give and receive clearinstructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and inmultidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context oftechnological change.

LAB EVALUATION PROCEDURE

Internal Laboratory Examination:

The internal laboratory examination shall be conducted around the middle of the semester. The examination is to be conducted, by a team of two examiners, one who conducts the laboratory sessions and the other appointed by the HoD. The scheme of evaluation is given below.

Component	Marks		
	Internal Examiner	External Laboratory Examiner	Total
Objective & Procedure write up including outcomes	5	5	10
Experimentation and data collection	5	5	10
Computation of results	5	5	10
Analysis of results and interpretation	5	5	10
Viva voce	0	10	10
Total Marks	20	30	50

End – semester Laboratory Evaluation:

End semester examination for each practical course is conducted jointly by both internal and external examiners. The examiners are appointed by Dean, Evaluation from thepanelofexaminerssuggestedbytherespectiveHeadsoftheDepartment.Thescheme ofevaluationmayvarydependingonthenatureoflaboratory,whichshallbesharedwith the student by the laboratory in-charge and also stamped on the answer scripts. The general scheme of evaluation is given in Tablebelow.

Component	Marks		
	Internal Examiner	External Laboratory Examiner	Total
Objective & Procedure write up including outcomes	5	5	10
Experimentation and data collection	5	5	10
Computation of results	5	5	10
Analysis of results and interpretation	5	5	10
Viva voce	0	10	10
Total Marks	20	30	50

Continuous Assessment:

S.No	Component	Max.Marks	Marks Secured
1	Preparedness	2	
2	Viva-voce	2	
3	Experiment	3	
4	Analysis & Record	3	
Total		10	

Details of marks per test to be entered into portal by faculty:

S.No	**Program**	**Regulation**	**Type of subject**	**Marks per test**
1	B.Tech	R19	T+L	Theory: weekly test -10 marks@test, Mid exam – 30 Marks @test. LAB: CLA 10 marks @experiment, internal lab – 50 Marks, external lab 50 Marks.

COURSE DESCRIPTION AND OBJECTIVES

This course aims at concepts and principles of Operating Systems, its overall responsibility in acting as an interface between the system's hardware components and the user. Further, it also helps students to understand the different scheduling policies, process synchronization mechanisms, deadlock handling mechanisms and memory management techniques.

COURSE OUTCOMES

Upon completion of the course, the student will be able to achieve the following outcomes:

CO1 Understand, classify the basic concepts of operating system and Real Time Operating System (RTOS).

CO2 Apply the concepts of process scheduling algorithms and process synchronization techniques to derive the efficiency of resource utilization.

CO3 Analyze the requirements for attempting operating systems principles.

CO4 Design the various memory management schemes for a given scenario.

CO5 Simulate the operating systems principles using simulation tools and programming.

LAB OBJECTIVE

Upon successful completion of this Lab the student will be able to:

1. Demonstrate how to use the following Bourne Shell commands: cat, grep, ls, more, ps, chmod, finger, ftp, etc.
2. Use the following Bourne Shell constructs: test, if then, if then else, if then elif, for, while, until, and case.
3. Learn tracing mechanisms (for debugging), user variables, BourneShell variables, read-only variables, positional parameters, reading input to a BourneShell script, command substitution, comments, and exporting variables. In addition, test on numeric values, test on file type, and test on character strings are covered.
4. Copy, move, and delete files and directories
5. Write moderately complex Shell scripts.
6. Make a Shell script executable.
7. Create a ".profile" script to customize the user environment.
8. Use advanced features of File Transfer Protocol (FTP)
9. Compile source code into object and executable modules.
10. Execute programs written in c under UNIX environment

Guidelines to Students

How to Run Shell Scripts?

There are two ways you can execute your shell scripts. Once you have created a script file:

Method 1

Pass the file as an argument to the shell that you want to interpret your script.

Step 1 : create the script using vi, ex or ed

For example, the script file show has the following lines

```
echo Here is the date and time
date
```

Step 2 : To run the script, pass the filename as an argument to the sh (shell)

```
$ sh show
Here is the date and time
Sat jun 03 13:40:15 PST 2006
```

Method 2:

Make your script executable using the chmod command.

When we create a file, by default it is created with read and write permission turned on and execute permission turned off. A file can be made executable using chmod.

Step 1 : create the script using vi, ex or ed

For example, the script file show has the following lines

```
echo Here is the date and time
date
```

Step 2 : Make the file executable

```
$ chmodu+xscript_file
$ chmodu+x show
```

Step 3 : To run the script, just type the filename

```
$ show
  Here is the date and time
  Sat jun 03 13:40:15 PST 2006
```

How to run C programs

Step 1 : Use an editor, such as vi, ex, or ed to write the program. The name of the file containing the program should end in .c.

For example, the file show.c contains the following lines :

```
main()
{
        printf(" welcome to GNEC ");
}
```

Step 2 : Submit the file to CC (the C Compiler)

```
$ cc show.c
```

If the program is okay, the compiled version is placed in a file called a.out

Step 3 : To run the program, type a.out

```
$ a.out
  Welcome to GNEC
```

LISTOFEXPERIMENTS **TOTAL HOURS: 60**

1. Use vi editor to create a file with some text and save the file.
2. Add and Delete content to the file created above.
3. Write programs that use the following processing utilities.
    a. wc, od, cmp, comm, diff, head, tail, cut, paste, sort, grep, uniq
    b. Disk backup utilities
    c. Du, df, tar, cpio, ps, who
4. Write a shell script to generate a multiplication table.
5. Write a shell script that copies multiple files to a directory.
6. Write a shell script which counts the number of lines and words present in a given file.
7. Write a shell script, which displays the list of all files in the given directory.
8. Write a shell script (of small calculator) that adds, subtracts, multiplies and divides the given two integers.
9. Write a shell script to reverse the rows and columns of a matrix.
10. Write a C program that counts the number of blanks in a text file.
    i) Using standard I/O ii) Using system calls.
11. Write a C program that illustrates how to execute two commands concurrently with a command pipe.
12. Write a C program that illustrates file locking using semaphores.
13. Write a C program that implements a producer-consumer system with two processes.(using semaphores)
14. Write a C program that illustrates inter process communication using shared memory system calls.
15. Write a C program that illustrates the following:
    i) Creating a message queue.
    ii) Writing to a message queue.
    iii) Reading from a message queue.

List of Experiments with CO-PO Mapping

Exp. No.	Week No.	Program Names	CO	PO	Page No
1	**1**	Use vi editor to create a file with some text and save the file.	5	5	1
2	**2**	Add and Delete content to the file created above.	5	5	2
3	**3**	Write programs that use the following processing utilities. a) wc, od, cmp, comm, diff, head, tail, cut, paste, sort, grep, uniq b) Disk backup utilities c) Du, df, tar, cpio, ps, who	5	5	3
4	**4**	Write a shell script to generate a multiplication table.	5	5	17
5	**5**	Write a shell script that copies multiple files to a directory.	5	5	18
6	**6**	Write a shell script which counts the number of lines and words present in a given file.	5	5	19
7	**7**	Write a shell script, which displays the list of all files in the given directory.	5	5	20
8	**8**	Write a shell script (of small calculator) that adds, subtracts, multiplies and divides the given two integers.	5	5	21
9	**9**	Write a shell script to reverse the rows and columns of a matrix.	5	5	22
10	**10**	Write a C program that counts the number of blanks in a text file. i) Using standard I/O ii) Using system calls.	5	5	24
11	**11**	Write a C program that illustrates how to execute two commands concurrently with a command pipe.	5	5	26
12	**12**	Write a C program that illustrates file locking using semaphores.	4,5	3,5	27
13	**13**	Write a C program that implements a producer-consumer system with two processes.(using semaphores)	4,5	3,5	29
14	**14**	Write a C program that illustrates inter process communication using shared memory system calls.	4,5	3,5	31
15	**15**	Write a C program that illustrates the following: a. Creating a message queue. b. Writing to a message queue. c. Reading from a message queue.	4,5	3,5	33

Exercise 1

Use vi editor to create a file with some text and save the file.

Add commands:

In insert mode user can insert text into the file. Vi editor contains several commands to change the mode to text. Some of them are: i, I, a, A, o and O.

command	Explanation
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

save work and Exit commands:

The user can save the work to prevent losing it in case of system failure. There are six save and exit commands. All these commands except zz command requires the command line. That is first press :to switch to command-line mode and then type these commands.

Commands	Explanation
:w	Saves the current file with out quitting vi.
:w filename	Saves the current file under the name file
zz	Saves the current file and ends vi
:wq or :x	Saves the current file and ends vi
:q	Ends vi (if no changes were made).
:q!	Ends vi without saving changes in the file.

Creating a file :
Vi example.txt
Hi this is vu
Save file :
:wq

Exercise 2

Add and Delete content to the file created above.

Vi example.txt
Hi this vu Guntur
Vadlamudi

Deletion commands:

Following are few commands used to delete characters, words, lines etc;

Command	Explanation
x	delete single character; 5x deletes 5 characters
dw	delete word; 5dw deletes 5 words
dd	delete line; 5dd deletes 5 lines
cw	delete word, leaves you in insert mode (i.e. change word)
cc	change line -- delete line and start insert mode
s	change character -- delete character and start insert mode
D	delete from cursor to end of line
C	change from cursor to end of line -- delete and start insert mode

Exercise 3

Write programs that use the following processing utilities.

a) wc, od, cmp, comm, diff, head, tail, cut, paste, sort, grep, uniq
b) Disk backup utilities
c) Du, df, tar, cpio, ps, who

Cat:(catenate)

- Create a file
- Display the contents of the file
- Concatenating a file

Creating a file:-

It is used to create a new file

Syntax:-

cat>filename

Eg:

```
[csec@localhost~]$ cat>resume
        hi this is csec
```

Display the contents of a file:-

Here cat command is used to display the contents of a file

Syntax:-

cat filename

Eg:

```
[csec@localhost~]$ cat resume
        hi this is csec
```

concatenating a file:-

It is used to append the contents of one file with other file

Syntax:-

cat filelist

Eg:

```
csec@localhost~]$ cat>f9
        hi this is csec
[csec@localhost~]$ cat>f8
        guntur
[csec@localhost~]$ cat f8 f9
        guntur
        hi this is csec
```

nl:-(numbered lines):

It is used to display the contents along with line nimbers

Syntax:

nl options filename

Eg:

```
[csec@localhost~]$ cat>resume
        hi
        this
        is
        csec
[csec@localhost~]$nl resume
        1       hi
        2       this
        3       is
        4       csec
```

Wc:-

it is used to count the no.oflines,words,characters in a given file

syntax:- wc filename

eg:-

```
[csec@localhost~]$ cat>f45
hi
this
is vu
[csec@localhost~]$wc f45
3  4 15 f45
```

options:-

-l: count the no. of lines

-c: count the no. of characters

-w: count the no. of words

eg:-

```
[csec@localhost~]$wc -l f45
3 f45
[csec@localhost~]$wc -w f45
4 f45
[csec@localhost~]$wc -c f45
15 f45
```

More:-

- more command is used to display the contents of a file on screen at a time
- a screen contains only 24 lines

syntax:-

more options filelist

eg:-

```
[csec@localhost~]$ vi f34
[csec@localhost~]$ more f34
hi
this
is vu
csec
guntur
1
2 3
4
5
6
7
89
1
23
4
565
56
hello
hai
good morning
bye
good
```

options:-

- -c : clear screen before displaying
- -lines: set the no of lines in screen
- +number: start the output at indicated number

eg:-
[csec@localhost~]$ more -5 f34
hi
this
vu
csec
guntur

head:

- It is used to display the specified no:of lines from the beginning of one or more files.
- By default head command display first 10 lines of a given file.

Syntax:

head options filelist

Options:

-n : used to specify no:of lines

EXAMPLE:
[csed@localhost~]$ cat>resume
hi
this
is
vignan
vadlamudi
guntur
ap
india
csea
cseb
csec
csed

[csed@localhost~]$ head resume
hi
this
is
vignan
vadlamudi
guntur
ap
india
csea
cseb
[csed@localhost~]$ head -4 resume
hi
this
is
vignan
[csed@localhost~]$

Tail:-

- it is used to display the specified no.of lines from the end of the file
- By default tail command display the last '10' lines

syntax:

tail options filename

eg:-
[csec@localhost~]$ cat>res
hi

this
is
csec
vignan
vadlamudi
guntur
ap
[csec@localhost~]$
[csec@localhost~]$ tail res
hi
this
is
csec
vignan
vadlamudi
guntur
ap
options:-
eg:-
[csec@localhost~]$ tail -5 res
csec
vignan
vadlamudi
guntur
ap

sort(sorting):-

* The process of arranging the data in a specific order
* It is used to display the contents of a file in specific order
*By default sort command display the content in ascending order
syntax:-
sort options filename
eg:-
[csec@localhost~]$ vi f1
[csec@localhost~]$ vi resume
[csec@localhost~]$ sort f1
this is vu
[csec@localhost~]$ vi f1
[csec@localhost~]$ sort f1
hi
is
this
vu
note:-Sort command the content based on the ASCII values
options:-
-r: used to display in descending order
-c: check whether file is sorted or not
-n: sort numeric data
-m: used to merge/compare the two sorted files
[csec@localhost~]$ sort -r f1
vu
this
is
hi
[csec@localhost~]$ sort -c f1
sort: f1:3: disorder: is
[csec@localhost~]$ sort -c resume

sort: resume:4: disorder:
[csec@localhost~]$ vi f2
[csec@localhost~]$ sort f2
100
40
50
60
[csec@localhost~]$ sort -n f2
40
50
60
100

uniq:-

- This command is used to remove duplicate lines.

syantax:-uniq options file

example:- [csed@localhost ~]$ cat>f1
hi
hi
bye
bye
bad
[csed@localhost~]$uniq f1
hi
bye
bad

Options:-

-d :used to display only duplicate lines
-u:used to display only uniq lines
-c: used to count the duplicate lines
-i: ignore case(either upper or lower the duplicate will be deleted)

ex:- [csed@localhost ~]$ cat>f1
hi
hi
bye
bye
bad
[csed@localhost~]$uniq -d f1
hi
bye
[csed@localhost~]$

[csed@localhost~]$uniq -u f1
bad
[csed@localhost~]$uniq -c f1
2 hi
2 bye
1 bad
[csed@localhost~]$uniq -i f1
hi
bye
bad
[csed@localhost~]$

Cut:

*It works on columns of data.
*It is used to extract the c'olumns of data from one or more files.

Syntax:

cut options filelist

Options:

-c : character
-f : field
-d :delemeter

EXAMPLE:

```
[csed@localhost~]$ cat>resume
hi this is vu
vignan
vadlamudi
[csed@localhost~]$ cut -c1-3 resume
hi
vig
vad
[csed@localhost~]$ cut -c1,3 resume
h
vg
vd
[csed@localhost~]$ cat>marks
rno name marks
1   ram  90
2   sri  95
3   sai  80
[csed@localhost~]$ cut -f1,3 marks
rno     marks
1        90
2        95
3        80

[csed@localhost~]$ cut -f1-3 -d ":" marks
rno:name:marks
1:ram:90
2:sri:95
3:sai:80
```

Paste:

*It is used to combine columns together.
*It is used to combine lines together.

Syntax:-

paste options file list

example:-

```
[csed@localhost~]$ cat>f1
Rno.
1
2
3
[csed@localhost~]$ cat>f2
name
```

siva
teja
ram
[csed@localhost~]$ cat>f3
marks
75
85
95
[csed@localhost~]$ paste f1 f2 f3
Rno. name marks
1 siva 75
2 teja 85
3 ram 95

options :-

-d : divider /delemeter

syntax:-

paste -d ":" file list

example:-

[csed@localhost~]$ paste -d ":" f1 f2 f3
Rno. :name :marks
1 :siva :75
2 :teja :85
3 :ram :95

Grep family: -

grep family consists of three commands: grep, fgrep, egrep

grep:-

grep command is used to search the contents of a file based on regular expression(pattern) and print the lines where the regular pattern is matched

syntax:

grep options 'reg expr' file list

Eg:

1 . [csec@localhost~]$ cat>f9
hi
this is
vu
guntur
[csec@localhost~]$grep hi f9
hi
this is

2.csec@localhost ~]$ cat>results

Rno	usp	ca	stld	ps	minor
141	90	85	89	86	85
142	90	80	85	90	85
143	90	90	86	87	85

[csec@localhost~]$grep 141 results
141 90 85 89 86 85

options:

-c:count the no.of lines matched with regular expression
-n:print the lines with line number
-v:print the lines those are not match with pattern
-i:ignore case

Eg:

[csec@localhost~]$grep -c hi f9
2
[csec@localhost~]$grep -n u f9
3:vu

4:guntur
[csec@localhost~]$grep -v i f9
vu
guntur
[csec@localhost~]$ cat>f8
hi
HI
good
[csec@localhost~]$grep hi f8
hi
[csec@localhost~]$grep -i hi f8
hi
HI

fgrep:

Syntax:
fgrep options string file list
Eg:
[csec@localhost~]$fgrep hi f9
hi
this is

egrep:

Syntax:
egrep options 'reg expr' file list
Eg:
[csec@localhost~]$egrept..s f9
this is

Comparing files:

comm,cmp,diff are three commands used to compare two files

comm:

- it is used to find the common lines between two sorted files
- it compares two sorted files line-by-line and display the put in '3' columns

syntax:
comm options file1 file2
column 1: lines uniq in file1
column 2: lines uniq in file2
column 3: lines common in both files

eg:-[csec@localhost ~]$ cat>f12
good
bad
think
[csec@localhost~]$ cat>f11
hi
good
bye
[csec@localhost~]$ sort f11>f13
[csec@localhost~]$ sort f12>f14
[csec@localhost~]$comm f13 f14
bad
bye
good
hi

think
options:-
-1: suppress column 1
-2: suppress column 2
-3:-suppress column 3

eg:-[csec@localhost ~]$ comm -1 f13 f14
bad
good
think
[csec@localhost~]$comm -2 f13 f14
bye
good
hi

cmp (compare):-

* it compares two files byte-by-byte(character- by-character)
* If the contents of two files is same then it doesn't display any thing
* If the content of two files are different, then it display byte number and line number at which first difference occur

syntax:-
cmp options file1 file2
eg:-
[csec@localhost~]$ cat>f15
good
day
[csec@localhost~]$ cat>f16
great
morning
[csec@localhost~]$cmp f15 f16
f15 f16 differ: byte 2, line 1
options:-
-l : display all the difference found in files byte-by-byte
eg:-
[csec@localhost~]$cmp -l f15 f16
2 157 162
3 157 145
4 144 141
5 40 164
7 144 155
8 141 157
9 171 162
10 12 156
cmp: EOF on f15

diff(difference):-

* diff command is used to find the difference between two files
* It shows line-by-line difference between files

syntax:-
diff options file1 file2
eg:-
[csec@localhost~]$ cat>f1
ram
raj
sri
rama
[csec@localhost~]$ cat>f2

ram
rani
sri
[csec@localhost~]$ diff f1 f2
2c2
< raj

> rani
4d3
<rama

* The difference identity such that the first file could be modified to make it match the second file

c(change):-

It indicates what lines to be replaced in file1 to make it same as file2

a(append):-

It indicates what lines need to be added to file1 to make it same as file2

*append can occur when file1 is shorter than file2

d(delete):-

It indicates what llines must be deleted from file1 to make it same as file2

eg:-[csec@localhost ~]$ cat>f1
rr
csk
mi
rcb
[csec@localhost~]$ cat>f2
rr
dc
khr
srh
pwi
[csec@localhost~]$ diff f1 f2
2,4c2,5
<csk
< mi
<rcb

> dc
>khr
>srh
>pwi

Join:-

It is used to combine lines of two files based on fileds

Syntax:-

join options f1 f2

options:-

-j:used to specify common fields

Eg:

[csec@localhost~]$ cat>f9

Rno	Name
1	R
2	S
3	P

[csec@localhost~]$ cat>f8

Rno	Marks
1	55
2	66
3	77

```
[csec@localhost~]$ join -j 1 f9 f8
        Rno Name Marks
        1    R    55
        2    S    66
        3     P    77
```

Disk Utilities :-

*It is used to know the disk space used by the files or directories.

Syntax:-

du options filename/directory name.

Example:-

```
[csed@localhost~]$mkdiranu
    [csed@localhost~]$ du anu
    4    anu
    [csed@localhost~]$ du f1
    4    f1
```

options:-

-a :- write and counts all
-s :- it display grand total
-c :- it display individual size of sub directories and grand total
-b :- it counts no characters

example:-

```
        [csed@localhost~]$du -s anu
4    anu
[csed@localhost~]$du -a anu
0    anu/f4
0    anu/f3
0    anu/f2
4    anu
[csed@localhost~]$du -c anu
4    anu
4    total
[csed@localhost~]$du -b anu
4096        anu
```

Disk free (df):-

*to know the avaliable free space for file system

*it is used to display file system available space and their used space

syntax:-

df options

example:-

```
[csed@localhost~]$df
Filesystem                    1K-blocks      Used  Available  Use% Mounted on
 /dev/mapper/VolGroup-lv_root  37414448  12529380  22984524  36% /
tmpfs                         1672308       432   1671876   1% /dev/shm
/dev/sda6                     495844     29541    440703   7% /boot
[csed@localhost~]$df -h
FilesystemSize  Used  Avail  Use% Mounted on
/dev/mapper/VolGroup-lv_root  36G   12G   22G  36% /
tmpfs                         1.6G  432K  1.6G   1% /dev/shm
/dev/sda6                      485M   29M  431M   7% /boot
```

who command:

- who command is used to display the users who are logged in to system
- who command without options and arguments display the user name,terminalnumber,date and time that each user logged into the system

Syntax:
who [option] [arguments]
Eg:
[csec@localhost~]$ who
csec tty1 2013-09-22 13:17 (:0)
csec pts/0 2013-09-22 14:33 (:0.0)
options:
-u:displysname,terminal,time IDLE and comment
-H:display header for each column
-b:time of last system boot
Eg:
[csec@localhost~]$ who -u
csec tty1 2013-09-22 13:17 old 2304 (:0)
csec pts/0 2013-09-22 14:33 . 3580 (:0.0)
[csec@localhost~]$ who -H
NAME LINE TIME COMMENT
csec tty1 2013-09-22 13:17 (:0)
csec pts/0 2013-09-22 14:33 (:0.0)
[csec@localhost~]$ who -b
system boot 2013-09-22 13:16

Tee command:

It is used to send the ouput of a command to a monitor as well as copied into many files
Syntax:
tee options filelist
Eg:
[csec@localhost~]$who|tee user.txt
csec tty1 2013-09-22 13:17 (:0)
csec pts/0 2013-09-22 13:27 (:0.0)

Back uputility:-

tar(tape archive):-
* It is used to group the many files into a single tape archive or disk archive
syntax:-
tar options tarname file list
options:-
-c: used to create new archive
-x: extract files from archive
-t: display longlist of files
eg:_
[csec@localhost~]$ tar -cvf example f11 f12 f15
f11
f12
f15
[csec@localhost~]$ tar -xvf example
f11
f12
f15
[csec@localhost~]$ tar -tvf example
-rw-rw-r-- csec/csec 13 2013-09-22 13:23 f11
-rw-rw-r-- csec/csec 15 2013-09-22 13:23 f12
-rw-rw-r-- csec/csec 10 2013-09-22 13:35 f15

cpio

cpio stands for “copy in, copy out“. It is used for processing the archive files like **.cpio* or **.tar*. This command can copy files to and from archives.
Synopsis:
Copy-out Mode: Copy files named in name-list to the archive

Syntax:
cpio -o < name-list > archive
Copy-in Mode: Extract files from the archive
Syntax:
cpio -i< archive
Copy-pass Mode: Copy files named in name-list to destination-directory
Syntax:

cpio -p destination-directory < name-list
Options:

- -i, –extract: Extract files from an archive and it runs only in copy-in mode.
- -o, –create: Create the archive and it runs only in copy-out mode.
- -p, –pass-through: Run in copy-pass mode.
- -t, –list: Print a table of contents of all the inputs present.

Operation modifiers valid in any Mode:

- -B: Changes the I/O block size to 5120 bytes.
- -c: Use the old portable (ASCII) archive format.
- -C, –io-size=NUMBER: Set the I/O block size to the given particular NUMBER of bytes.
- -D, –directory=DIR: Changes to Directory *DIR*.
- -H, –format=FORMAT: Use given arc.
- -v, –verbose: List the files processed in a particular task.
- -V, –dot: Print "." for each file processed in a particular task.
- -W, –warning=FLAG: Control warning display. Currently FLAG is one of '*none*', '*truncate*', '*all*'.

ps

Linux provides a utility called ps for viewing information related with the processes on a system which stands as abbreviation for "Process Status". ps command is used to list the currently running processes and their PIDs along with some other information depends on different options. It reads the process information from the virtual files in /proc file-system. /proc contains virtual files, this is the reason it's referred as a virtual file system.ps provides numerous options for manipulating the output according to our need.
Syntax –
ps [options]
Options
Simple process selection : Shows the processes for the current shell –
[root@rhel7 ~]#ps
PID TTY TIME CMD
12330 pts/0 00:00:00 bash
21621 pts/0 00:00:00 ps
Result contains four columns of information.
Where,
PID – the unique process ID
TTY – terminal type that the user is logged into
TIME – amount of CPU in minutes and seconds that the process has been running
CMD – name of the command that launched the process.
Note – Sometimes when we execute ps command, it shows TIME as 00:00:00. It is nothing but the total accumulated CPU utilization time for any process and 00:00:00 indicates no CPU time has been given by the kernel till now. In above example we found that, for bash no CPU time has been given. This is because bash is just a parent process for different processes which needs bash for their execution and bash itself is not utilizing any CPU time till now.

View Processes : View all the running processes use either of the following option with ps –
[root@rhel7 ~]#ps -A
[root@rhel7 ~]#ps -e
View Processes not associated with a terminal : View all processes except both session leaders and processes not associated with a terminal.
[root@rhel7 ~]#ps -a
View all the processes except session leaders :
[root@rhel7 ~]#ps -d
View all processes except those that fulfill the specified conditions (negates the selection) :
Example – If you want to see only session leader and processes not associated with a terminal. Then, run
[root@rhel7 ~]#ps -a -N
View all processes associated with this terminal :
[root@rhel7 ~]#ps -T
View all the running processes :
[root@rhel7 ~]#ps -r
View all processes owned by you : Processes i.e same EUID as ps which means runner of the ps command, root in this case –
[root@rhel7 ~]#ps–x

Exercise 4

Write a shell script to generate a multiplication table.

Objective: Students will be able to write shell script that generates the multiplication table.

Algorithm:

1. Read which table you want
2. Read the range of table you want
3. Write the multiplication logic inside while loop.
4. Print the multiplication table.

Program:

```
echo "enter which table you want"
read n
echo "enter range of the table"
read r
i=1
while [ $i -le $r ]
do
t=`expr $i \* $n`
echo "$i * $n = $t"
i=`expr $i + 1`
done
```

Output:

```
enter which table you want
5
enter range of the table
10
1 * 5 = 5
2 * 5 = 10
3 * 5 = 15
4 * 5 = 20
5 * 5 = 25
6 * 5 = 30
7 * 5 = 35
8 * 5 = 40
9 * 5 = 45
10 * 5 = 50
```

Outcome: Students learn the logic how to write a shell script to generate the multiplication table in Unix environment.

Exercise 5

Write a shell script that copies multiple files to a directory.

Objective: Students will be able to write the shell script that copies the multiple files into directory.

Algorithm:

1. Read the no of files you want to copy in to the directory
2. Read the directory name
3. Check weather directory is exists or not.
4. if exists copy the files in to directory by using cp command.

Program:

```
echo "how many files you want to copy in to the directory"
read nf
echo "enter directory name"
read dr
if [ ! -e $dr ]
then
echo "directory does not exists"
else
i=1
while [ $i -le $nf ]
do
echo "enter file name"
read fname
if [ ! -e $fname ]
then
echo "file does not exists"
else
cp $fname $dr
echo "file copied"
fi
i=`expr $i + 1`
done
fi
```

output

```
how many files you want to copy in to the directory
2
enter directory name
unix
enter file name
a.txt
file copied
enter file name
b.txt
```

Exercise 6

Write a shell script which counts the number of lines and words present in a given file.
Objective: Students will be able to know the no of lines and words present in a given file.

Algorithm:

1. read the file name.
2. you can count the no of lines in a file using wc -l command and no of words using wc -w command.

Program:

```
echo 'enter file'
read file
if [ -f $file ]
then
l=`wc -l $file`
w=`wc -w $file`
c=`wc -c $file`
echo $l
echo $w
echo $c
else
echo 'not'
fi
```

output

```
enter file name
a.txt
no of lines:1 no of words:5
```

Exercise 7

Write a shell script, which displays the list of all files in the given directory.

Algorithm:

1.read the directory name
2.check weather directory is exists or not.
3.if it exists then display all the files which are in given directory.

Program:

```
echo 'enter dir'
read dir2
if [ -d $dir2 ]
then
for iin  `ls $dir2`
do
echo $i
done
else
echo 'not'
fi
```

output

```
enter directory name
unix
files in the directory are:
.:
a.out
a.txt
b.txt
prog11.c
prog13.c
prog14.c
prog18a.c
prog2.sh
```

Exercise 8

Write a shell script (of small calculator) that adds, subtracts, multiplies and divides the given two integers.

Allgorithm:
1. Read two numbers
2. Apply Addition, subtraction, multiplication and division on those two numbers
3. Print the result.

```
if [ $# -lt3 ]; then
echo "usage:sh filename option int1 int2"
exit
fi
if [ $1 = a ]; then
i=`expr $2 + $3`
echo "sum=$i"
elif [ $1 = s ]; then
i=`expr $2 - $3`
echo "subtraction=$i"
elif [ $1 = m ]; then
i=`expr $2 \* $3`
echo "muliplication=$i"
elif [ $1 = r ]; then
i=`expr $2 % $3`
echo "remainder=$i"
elif [ $1 = c ]; then
i=`expr $2 / $3`
echo "quotent=$i"
fi
```

output

```
# sh prog5.sh a 2 3
sum=5
#  sh prog5.sh s 5 3
subtraction=2
# sh prog5.sh m 5 3
muliplication=15
# sh prog5.sh c 5 3
quotent=1
# sh prog5.sh r 5 3
remainder=2
```

Exercise 9

Write a shell script to reverse the rows and columns of a matrix.

Algorithm:

1. Read no of rows and columns.
2.do transpose for the given matrix.
3.print the transpose matrix

Program:

```
echo "enter no of rows"
read rows
echo "no of columns"
read colmns
k=`expr $rows \* $colmns`
i=0
while [ $i -lt $k ]
do
echo "enter an element"
read a[$i]
i=`expr $i + 1`
done
echo "the matrix is"
i=0
while [ $i -lt $k ]
do
echo -n ${a[$i]} " "
i=`expr $i + 1`
l=`expr $i % $colmns`
if [ $l -eq0 ]
then
echo " "
fi
done
echo "transpose matrix is"
i=0
while [ $i -lt $colmns ]
do
j=0
while [ $j -lt $rows ]
do
k=`expr $i + $j \* $colmns`
echo -n ${a[$k]} " "
j=`expr $j + 1`
done
echo " "
i=`expr $i + 1`
done
```

Output

```
enter no of rows
3
no of columns
3
enter an element
1
 enter an element
```

```
2
 enter an element
3
 enter an element
4
 enter an element
5
 enter an element
6
 enter an element
7
 enter an element
8
enter an element
9

the matrix is
1   2   3
4   5   6
7   8   9

transpose matrix is
1   4   7
2   5   8
3   6   9
```

Exercise 10

Write a C program that counts the number of blanks in a text file.
i) Using standard I/O ii) Using system calls.

i) Using standard I/O

Algorithm:
1. Read the file
2. Open the file using fopen function.
3. Count the no blanks in a file and print the result.

Program

```
#include<stdio.h>
intmain()
{
int  count=0,n;
FILE      *fptr;
char    fname[20];
printf("enter a file name");
scanf("%S",&fname);
fptr=fopen(fname,"r");
while(!feof(fptr))
{
if(ch==' ')
count++;
ch=getc(fptr);
}
printf("no of blank spaces are:%d \n",count);
close(fptr);
return(0);
}
```

Output

```
Enter file name
a.txt
no of blank spaces in the file are 4
```

ii) Using system calls.

Description:
1. Read the file
2. Open the file using open function.
3. Read the contents of file using read function.
4. Count the no blanks in a file and print the result.

Program

```
#include<stdio.h>
#include<fcntl.h>
intmain()
{
charch,fname[20];
intfd,count=0,n;
printf("enter file name");
scanf("%s",&fname);
fd=open(fname,O_RDONLY);
```

```
while(ln=read(fd,&ch,1))>0)
{
if(ch==' ')
count++;
}
printf("\n no of blankspaces:%d",count);
close(fd);
}
```

output

```
enter file name:a.txt
no of blank spaces in the file are 4
```

Exercise 11

Write a c program to provide Inter Process Communication using pipes?

Algorithm:

1. Pipe is used between child and parent process.
2. Here parent process is created child process using fork() function
3. Pipe () function is used to create pipe.
4. Parent process write the data into pipe using write function.
5. Child process read the data from pipe using read function
6. Print the data on monitor.

Program

```
#include<stdio.h>
#include<fcntl.h>
#include<unistd.h>
#include<sys/stat.h>
intmain()
{
char str[20];
pid_tpid;
intfd[2],n;
pipe(fd);
pid =fork();
if(pid>0)
{
close(fd[0]);
write(fd[1],"example pipe",12);
}
else
{
close(fd[1]);
n=read(fd[0],str,12);
write(STDOUT_FILENO,str,n);
}
}
```

Output:

Example pipe

Exercise 12

Write a C program that illustrates file locking using semaphores

Algorithm

- Define a structure for semaphore with three data members val, buffer, and array of short type
- Define procedures for locking semaphore and unlocking semaphore my_lock(int) and my_unlock(int)
- Create a semaphore
- Set a value to semaphore
- Lock a semaphore using my_lock(int) and then unlock using my_unlock(int)

Program:-

```
#include <stdio.h>
 #include <sys/file.h>
 #include <error.h>
 #include <sys/sem.h>
 #define MAXBUF 100
 #define KEY 1216
 #define SEQFILE "suhritfile"
intsemid,fd;
 void my_lock(int);
 void my_unlock(int);
 union semnum
 {
intval;
structsemid_ds *buf;
     short *array;
}arg;
intmain(])
 {
int child, i,n, pid, seqno;
       char buff[MAXBUF+1];
pid=getpid();
if((semid=semget(KEY, 1, IPC_CREAT | 0666))= = -1)
       {
perror("semget");
exit(1);
       }
arg.val=1;
       if(semctl(semid,0,SETVAL,arg)<0)
perror("semctl");
       if((fd=open(SEQFILE,2))<0)
       {
perror("open");
exit(1);
       }
pid=getpid();
       for(i=0;i<2;i++)
       {
my_lock(fd);
lseek(fd,01,0);
              if((n=read(fd,buff,MAXBUF))<0)
              {
perror("read");
exit(1);
              }
printf("pid:%d, Seq no:%d\n", pid, seqno);
```

```
seqno++;
sprintf(buff,"%d\n", seqno);
                n=strlen(buff);
lseek(fd,01,0);
                if(write(fd,buff,n)!=n)
                {
perror("write");
exit(1);
                }
sleep(1);
my_unlock(fd);
        }
 }
    void my_lock(intfd)
    {
structsembuffsbuf=(0, -1, 0);
if(semop(semid, &sbuf, 1)= =0)
printf("Locking: Resource…\n");
              else
printf("Error in Lock\n");
    }
    void my_unlock(intfd)
    {
structsembuffsbuf=(0, 1, 0);
if(semop(semid, &sbuf, 1)= =0)
printf("UnLocking: Resource…\n");
              else
printf("Error in Unlock\n");
    }
```

Exercise 13

Write a C program that implements a producer-consumer system with two processes. (using semaphores)

Algorithm

- Define number of operations as 20
- Create a semaphore using semget()
- Set value to the created semaphore with semctl()
- Create a child process using fork()
- Set and unset semaphore with respect to the producer and consumer

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#define NUM_LOOPS 20
intmain(intargc, char* argv[])
{
intsem_set_id;
   union semunsem_val;
intchild_pid;
inti;
structsembufsem_op;
intrc;
structtimespec delay;

sem_set_id = semget(IPC_PRIVATE, 1, 0600);
   if (sem_set_id == -1) {
     perror("main: semget");
     exit(1);
   }
printf("semaphore set created,
 semaphore set id '%d'.\n", sem_set_id);

sem_val.val = 0;
rc = semctl(sem_set_id, 0, SETVAL, sem_val);
child_pid = fork();
   switch (child_pid) {
     case -1:
     perror("fork");
     exit(1);
     case 0:
        for (i=0; i<NUM_LOOPS; i++) {
               sem_op.sem_num = 0;
               sem_op.sem_op = -1;
               sem_op.sem_flg = 0;
               semop(sem_set_id, &sem_op, 1);
               printf("consumer: '%d'\n", i);
               fflush(stdout);
sleep(3);
         }
         break;
```

```
    default:
        for (i=0; i<NUM_LOOPS; i++)
         {
                printf("producer: '%d'\n", i);
                fflush(stdout);
                sem_op.sem_num = 0;
                sem_op.sem_op = 1;
                sem_op.sem_flg = 0;
                semop(sem_set_id, &sem_op, 1);
                sleep(2);
                if (rand() > 3*(RAND_MAX/4))
                {
                delay.tv_sec = 0;
                delay.tv_nsec = 10;
                nanosleep(&delay, NULL);
                }
        }
        break;
   }

   return 0;
}
```

Exercise 14

Write a C program that illustrates inter process communication using shared memory system calls.

Algorithm

Process A:

- Here shared memory segment is created using shmget() function
- If shmid is less than zero it prints error.
- Attach the shared memory segment to the process
- Copy the data into shared memory segment using strcpy() function.

```
#include<stdio.h>
#include<sys/shm.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<string.h>
main()
{
intshmid,flag;
key_t key=0x1000;
char *msg;
shmid=shmget(key,10,IPC_CREAT|0666);
if(shmid<0)
{
printf("error");
}
printf("%d\n",shmid);
msg=shmat(shmid,0,0);
strcpy(msg,"example for sharedmemory");
//write(1,msg,strlen(msg));
}
```

Output

70012

Process B

- Get the shared memory segment using shmget() function
- If shmid is less than zero it prints error.
- Attach the shared memory segment to the process
- Read the data into shared memory segment using write function.

Program

```
#include<stdio.h>
#include<sys/shm.h>
#include<sys/types.h>
#include<sys/ipc.h>

main()
{
intshmid;
key_t key=0x1000;
char *msg;
shmid=shmget(key,10,IPC_CREAT|0666);
if(shmid<0)
{
printf("error");
}
```

```
printf("id is%d",shmid);
msg=shmat(shmid,0,0);
//read(shmid,msg,strlen(msg));
printf("%s",msg);
}
```

Output:
70012

Exercise 15

Write a C program that illustrates the following:
i) Creating a message queue.
ii) Writing to a message queue.
iii) Reading from a message queue.

Program to create a message queue and send a message into the queue.

Algorithm

- Here message queue is created using msgget() function
- If msqid is less than zero it prints error.
- Copy the message into into message queue using strcpy() function.

Program:

```
#include<stdio.h>
#include<sys/ipc.h>
#include<sys/msg.h>
#include<sys/types.h>
structmesg
{
long type;
char mtext[252];
} mesg;
main ()
{
intmsqid,len;
if((msqid=msgget((key_t)10,IPC_CREAT|0666))<0)
{
printf("not");
}
printf("qid is=%d",msqid);
mesg.type=6;
strcpy(mesg.mtext,"example for mq");
len=strlen(mesg.mtext);
if(msgsnd(msqid,&mesg,len,0)==-1)
printf("write error");
printf("data is placed successfully");
}
```

O/P:mesg que id is =0.
Data is placed into the queue=example of mq.

b) Read the message in the message queue written in the previous program.

Algorithm:

- Get the message queue using msgget() function
- If msqid is less than zero it prints error.
- Read the message from the message queue

Program:

```
#include<stdio.h>
#include<sys/msg.h>
#include<sys/ipc.h>
#include<sys/types.h>
structmesg
{
long type;
char mtext[255];
}mesg;
main()
{
intmsqid;
if((msqid=msgget((key_t)10,IPC_CREAT|0666))<0)
printf("error");
printf("received mq id is=%d",msqid);
if((msgrcv(msqid,&mesg,255,6,IPC_NOWAIT))<0)
printf("ERRROR");
printf("%s",mesg.mtext);
}
```

Output:

Received mesg que id is =0.
example of mq.