A Course File

On

OPERATING SYSTEMS(Linux) LAB

(II- B. Tech. – II– Semester)

Submitted to

DEPARTMENT OF COMPUTER SCIENCE& ENGINEERING   (DS)

By

Mr.V.DILIP REDDY

Assistant Professor, Dept. of CSE

Mrs. SRI BALA

Assistant Professor, Dept. of CSE (DS)

Ms.S.SWETHA

Assistant Professor, Dept. of CSE (DS)

CMR INSTITUTE OF TECHNOLOGY

Kandlakoya(V), Medchal Road, Hyderabad – 501 401
Ph. No. 08418-222042, 22106 Fax No. 08418-222106

(2021-22)

CONTENTS

1. OPERATING SYSTEMS (Linux) LAB

B.Tech-IV-Sem   **L   T   P   C**

Subject Code:20-CS-PC-227   **-   -   3   1.5**

Course Outcomes: Upon completion of the course, the students will be able to

1. illustrate Linux shell environment
2. create process using APIs
3. interpret various CPU scheduling algorithms and file allocation methods.
4. experiment with page replacement and memory management
5. distinguish deadlock avoidance and deadlock prevention

LIST OF EXPERIMENTS

Week 1:
Study of Linux general purpose utilities (File handling, Process utilities, Disk utilities, Networking, Filters)

Week 2:
a) Write a shell script to find factorial of a given integer.
b) Write a Shell Script to wish 'Good Morning' and 'Good Evening' depending on thesystem time.

Week 3:
Implement Linux cat command using File API s.

Week 4:
Implement the Linux commands (a) cp (b)mv usingLinux system calls

Week 5:
Write a C program to create a child process and allow the parent to display 'parent' and the child to display 'child' on the screen.

Week 6:
Write a C program in which a parent writes a message to a pipe and the child reads the message.

Week 7:
Write C programs to simulate the following CPU scheduling algorithms
a) FCFS   b) Priority

Week 8:
Write C programs to simulate the following CPU scheduling algorithms
a)SJF   b) RR

Week 9:
Write C programs to simulate the following file allocation strategies:
a) Sequential   b) Linked c) Indexed

Week 10:
Write C programs to simulate the following memory management techniques
a) Paging   b) Segmentation

Week 11:
Write a C program to simulate bankersalgorithm for deadlock detectionand avoidance

Weeks 12:
Write C programs to simulate the following page replacement techniques:
a) FIFO   b) LRU   c) Optimal

Micro-Projects: Student must submit a report on one of the following Micro–Projects beforeCommencement of second internal examination.

1. Producer-consumer problem using semaphoreDining-
2. Philosopher problem using semaphore
3. Multithreading using pthread library
4. DAG (Directed Acyclic Graph) file organization technique
5. Virtual Memory Simulation.
6. Multi-level queue CPU scheduling algorithm.
7. Process/thread synchronization.
8. A slower file system mechanism
9. Demand paging technique of memory management
10. Threaded Matrix Multiply

References:

1. Operating Systems (Linux) Lab Manual, Department of CSE, CMRIT, Hyd.

CMR INSTITUTE OF TECHNOLOGY

VISION: To create world class technocrats for societal needs

MISSION: Impart global quality technical education for a better future by providing appropriate learning environment through continuous improvement and customization

QUALITY POLICY: Strive for global excellence in academics and research to the satisfaction of students and stakeholders

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Vision: To be a model for academic excellence and research in the field of computer science and engineering that prepares competent professionals with innovative skills, moral values and societal concern.

Mission: Impart quality education through state-of-art curriculum, conducive learning environment and research with scope for continuous improvement leading to overall professional success.

I. PROGRAMME EDUCATIONAL OBJECTIVES (PEO's)

PEO1: Graduate will be capable of practicing principles of computer science & engineering, mathematics and scientific investigation to solve the problems that are appropriate to the discipline.

PEO2: Graduate will be an efficient software engineerin diverse fields and will be a successful professional and/or pursue higher education and research.

PEO3: Graduate exhibits professional ethics, communication skills, teamwork and adapts to changing environments of engineering and technologyby engaging in lifelong learning.

II. PROGRAMME OUTCOMES (PO's)

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

2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

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

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

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

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the

engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **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, make effective presentations, and give and receive clear instructions.

11. **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 in multidisciplinary environments.

12. **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 of technological change.

13. **PSO1:**Design and Develop computer based systems across various domains related to Algorithms, Software Development, Networking, Security, Blockchain Technology, Multimedia, Full Stack Web Development, Cloud Computing, Artificial Intelligence, Machine Learning,Data Science, Cyber Security and IoT.

14. **PSO2:** Apply technical and research based skills learnt through professional society activities, internships and projects to provide solutions to real world problems in environment and society.

3. Course Outcomes

Course Outcome	Course Outcome Statements
CO - 1	illustrate Linux shell environment
CO – 2	create process using APIs
CO – 3	interpret various CPU scheduling algorithms and file allocation methods.
CO – 4	experiment with page replacement and memory management
CO – 5	distinguish deadlock avoidance and deadlock prevention

4. COURSE MAPPING WITH PO'S(No correlation: 0; Low: 1; Medium: 2; High: 3)

Course Title	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PO13	PO14
OPERATING SYSTEMS(LINUX) Lab	-	-	3	-	3	-	-	-	-	-	-	-	-	3

5. MAPPING OF COURSE OUTCOMES WITH PEO'S, PO'S(No correlation: 0; Low: 1; Medium: 2; High: 3)

Course Outcomes	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PO13	PO14
CO – 1	-	-	3	-	3	-	-	-	-	-	-	-	-	3
CO – 2	-	-	3	-	3	-	-	-	-	-	-	-	-	3
CO – 3	-	-	3	-	3	-	-	-	-	-	-	-	-	3
CO – 4	-	-	3	-	3	-	-	-	-	-	-	-	-	3
CO – 5	-	-	3	-	3	-	-	-	-	-	-	-	-	3

5. Direct Course Assessment

(As mentioned in following table of 10 parameters, of which consider only the parameters required for this course)

No	Description	Targeted Performance	Actual Performance	Remarks	Course Attainment
1	Internal Marks(25)	90% of Students(204 Students) should Secure 60% of Internal Marks i.e., 15 Marks	NA	All Course Outcomes in general attained & Marks Awarded or Attainment Level is 3(Strong & High)	NA
2	External Marks(50)	80% of Students(182 Students) should Secure 70% of External Marks i.e., 35 Marks	NA	All the Course Outcomes in general attained & Marks Awarded or Attainment Level is 3(Strong & High)	NA
3	Clearing of Subject(75)	A minimum of 95% of Students(216 Students) should clear this course in first attempt	NA	All Course Outcomes in general attained & Marks Awarded or Attainment Level is 3(Strong & High)	NA
4	Getting First Class	90% of Students(204 Students) should Secure I Class Marks i.e., 45 Marks in my course	NA	All Course Outcomes in general attained & Marks Awarded or Attainment Level is 3(Strong & High)	NA
5	Distinction	80% of Students (182 Students) should secure First Class With Distinction i.e., 53 Marks in my course	NA	All the Course Outcomes in general attained & Marks Awarded or Attainment Level is 1.5M(Low)	NA
6	Outstanding Performance	60% of Students (136 Students) should secure 80% and above Marks.i.e., 60 Marks in my course	NA	All the Course Outcomes in general attained & Marks Awarded or Attainment Level is 1.5M(Low)	NA

6. Indirect Course Assessment

(As mentioned-strong (3), moderate (2), weak (1) & no comment (0))

Mission Statement of CSE
● **Impart fundamentals through state of art technologies for research and career in Computer Science & Engineering.**
● **Create value-based, socially committed professionals for anticipating and satisfyingfast changing societal requirements.**
● **Foster continuous self learning abilities through regular interaction with various stake holders for holistic development.**

Correlation of Mission Elements with Mission Statement of CSE Department related to the Course (only Ticking given by faculty)

No	Mission Elements	Strong	Moderate	Weak	No Comment
M-1	Impart Fundamentals	√			
M-2	State Of Art Technologies		√		
M-3	Research & Career Development	√			
M-4	Value based Socially Committed Professional	√			
M-5	Anticipating & Satisfying Industry Trends	√			
M-6	Changing Societal Requirements		√		
M-7	Foster Continuous Learning	√			
M-8	Self Learning Abilities	√			
M-9	Interaction with stakeholders	√			
M-10	Holistic Development	√			

CMR Institute of Technology

Indirect Course Assessment through Student Satisfaction Survey

(Note for *:Parameters used for course teaching like
a: Classroom teaching b: Simulations c:labs d: Mini_Projects
e: Major Projects f: Conferences g: professional activities
h: Technical Clubs i: Guest Lectures j: Workshopsk: Technical Festsl:Tutorials
m:NPTLs n:Digital Library o: Industrial Visits
p: software Tools q: Internship/training r:Technical Seminars
s: NSS t: NSS u: sports etc.
Further assume other parameters if any)

No	Question Based on PEO/PO/PSO/CO	Parameters (a /b /c…/)*	Strong (3)	Moderate (2)	Weak (1)	No comment (0)
1	Did the course impart fundamentals through interactive learning and contribute to core competence?	a,b,c,h,l,m,r				
2	Did the course provide the required knowledge to foster continuous learning?	a,b,c,h,l,m,r				
3	Whether the syllabus content anticipates & satisfies the industry and societal needs?	a,b,c,h,l,m,r				
4	Whether the course focuses on value based education to be a socially committed professional?	a,b,c,h,l,m,r				
5	Rate the role of the facilitator in mentoring and promoting the self learning abilities to excel academically and professionally?	a,b,c,h,l,m,r				
6	Rate the methodology adopted and techniques used in teaching learning processes?	a,b,c,h,l,m,r				
7	Rate the course in applying sciences & engineering fundamentals in providing research based conclusions with the help of modern tools?	a,b,c,h,l,m,r				
8	Did the course have any scope to design, develop and test a system or component?	a,b,c,h,l,m,r				
9	Rate the scope of this course in addressing cultural, legal, health, environment and safety issues?	a,b,c,h,l,m,r				
10	Scope of applying management fundamentals to demonstrate effective technical project presentations & report writing?	a,b,c,h,l,m,r				
	Total					
	Average					
	Grand Average					

7. Overall Course Assessment

(80% Direct+20% Indirect)

No	Assessment Type	Weightage	Attainment Level
1	Direct-Assignment, Quiz, Subjective, University Exams, Results, Bench Marks		
2	Indirect-Surveys-Questionnaire		
	Overall		

Course Attainment level:

Operating Systems Course=

8. Lesson/Course Plan

Subject : OPERATING SYSTEMS (Linux) LAB   **Branch : CSE**

Year   : II-B.Tech   **Semester : II**

Text Books
   1. Operating Systems (Linux) Lab Manual, Department of CSE, CMRIT, Hyd.

Week No.	Name of the Program	No. of Lab sessions	Text Books	Mode of Assessment
1	Study of Linux general purpose utilities (File handling, Process utilities, Disk utilities, Networking, Filters)	1	T1	Viva&Execution
2	a) Write a shell script to find factorial of a given integer. b) Write a Shell Script to wish 'Good Morning' and 'Good Evening' depending on the system time.	1	T1	Viva&Execution
3	Implement Linux cat command using File API s.	1	T1	Viva&Execution
4	Implement the Linux commands (a) cp (b) mv using Linux system calls	1	T1	Viva&Execution
5	Write a C program to create a child process and allow the parent to display 'parent' and the child to display 'child' on the screen.	1	T1	Viva&Execution
6	Write a C program in which a parent writes a message to a pipe and the child reads the message.	1	T1	Viva&Execution
7	Write C programs to simulate the following CPU scheduling algorithms a) FCFS b) Priority	1	T1	Viva&Execution
8	Write C programs to simulate the following CPU scheduling algorithms a)SJF b) RR	1	T1	Viva&Execution
9	Write C programs to simulate the following file allocation strategies a) Sequential b) Linked c) Indexed	1	T1	Viva&Execution
10	Write C programs to simulate the following memory management techniques a) Paging b) Segmentation	1	T1	Viva&Execution
11	Write a C program to simulate bankersalgorithm for deadlock detectionand avoidance.	1	T1	Viva&Execution
12	Write C programs to simulate the following page replacement techniques: a) FIFO b) LRU c) Optimal	1	T1	Viva&Execution
	Total no of Lab sessions required to complete Syllabus	**12**		

9. Algorithms and Programs

Week-1:

NAME OF THE EXPERIMENT: Implement Linux general purpose utilities

AIM: Study of Linux general purpose utilities(file handling, process utilities, disk utilities, networking and filters).

ALGORITHM:

Step 1: Start

Step 2: open Terminal

Step 3:give the proper command and check with the different type of the options

Step 4:observe the result of the each options

Step 5: stop

SOURCE CODE

File handling utilities:

Mkdir :make directories

Usage: mkdir [OPTION] DIRECTORY...

eg. mkdir laxmi

ls : list directory contents

Usage: ls [OPTION]... [FILE]...

eg. ls, ls l,

ls laxmi

cd : changes directories

Usage: cd [DIRECTORY]

eg. cd laxmi

pwd: print name of current working directory

Usage: pwd

vim : Vi Improved, a programmers text editor

Usage: vim [OPTION] [file]...

eg. vim file1.txt

cp :copy files and directories

Usage: cp [OPTION]... SOURCE DEST

eg. cp sample.txt sample_copy.txt

cp :sample_copy.txt target_dir

mv : move (rename) files

Usage: mv [OPTION]... SOURCE DEST

eg. mv source.txt target_dir

mv old.txt new.txt

rm: remove files or directories

Usage: rm [OPTION]... FILE...

eg. rm file1.txt , rm rf

some_dir

find:search for files in a directory hierarchy

Usage: find [OPTION] [path] [pattern]

eg. find file1.txt, find name

file1.txt

history:prints recently used commands

Usage: history

Security filespermissions:

3 types of file permissions – read, write, execute

• 10 bit format from 'ls l'

command

1   2 3 4   5 6 7   8 9 10

file type   owner   group   others

eg. drwxrw-r—means owner has all three permissions,

group has read and write, others have only read permission

• read permission – 4, write – 2, execute -1

eg.rwxrw-r-- = 764

673 = rw-rwx-wx

chmod:change file access permissions

Usage: chmod [OPTION] [MODE] [FILE]

eg. chmod 744 calculate.sh

chown: change file owner and group

Usage: chown [OPTION]... OWNER[:[GROUP]] FILE...

eg. chown remo myfile.txt

su : change user ID or become superuser

Usage: su [OPTION] [LOGIN]

eg. su remo, su

passwd : update a user's authentication tokens(s)

Usage: passwd [OPTION]

eg. passwd

who: show who is logged on

Usage: who [OPTION]

eg. who , who b , who q

Process utilities:

ps : report a snapshot of the current processes

Usage: ps [OPTION]

eg. ps, ps el

kill :to kill a process(using signal mechanism)

Usage: kill [OPTION] pid

eg. kill 9

2275

Tar: to archive a file

Usage: tar [OPTION] DEST SOURCE

eg. tar cvf

/home/archive.tar /home/original

tar xvf

/home/archive.tar

Zip:package and compress (archive) files

Usage: zip [OPTION] DEST SOURSE

eg. zip original.zip original

unzip: list, test and extract compressed files in a ZIP archive

Usage: unzip filename

eg. unzip original.zip

Disk utilities:

du (abbreviated from *disk usage*) is a standard Unixprogram used to estimate file space usage—space used under a particular directory or files on a file system**.**

du takes a single argument, specifying a pathname for du to work; if it is not specified, the current directory is used. The SUS mandates for du the following options:

-a, display an entry for each file (and not directory) contained in the current directory

-H, calculate disk usage for link references specified on the command line

-k, show sizes as multiples of 1024 bytes, not 512-byte

-L, calculate disk usage for link references anywhere

-s, report only the sum of the usage in the current directory, not for each file

-x, only traverse files and directories on the device on which the pathname argument is specified.

Other Unix and Unix-like operating systems may add extra options. For example, BSD and GNU du specify a −h option, displaying disk usage in a format easier to read by the user, adding units with the appropriate SI prefix'

$ du-sk*
152304 directoryOne
1856548 directoryTwo

Sum of directories in human-readable format (Byte, Kilobyte, Megabyte, Gigabyte, Terabyte and Petabyte):

```
$ du-sh*
149M directoryOne
1.8G directoryTwo
```

disk usage of all subdirectories and files including hidden files within the current directory (sorted by filesize) :

```
$ du-sk .[!.]**|sort-n
```

disk usage of all subdirectories and files including hidden files within the current directory (sorted by reverse filesize) :

```
$ du-sk .[!.]**|sort–nr
```

The weight of directories:

```
$ du-d1-c-h
```
df command : Report file system disk space usage

Df command examples - to check free disk space

Typedf -h or df -k to list free disk space:
```
$ df -h
```
OR
```
$ df -k
```

Output:

```
Filesystem        Size   Used  Avail  Use% Mounted on
/dev/sdb1         20G    9.2G  9.6G   49% /
varrun            393M   144k  393M   1% /var/run
varlock           393M   0     393M   0% /var/lock
procbususb        393M   123k  393M   1% /proc/bus/usb
udev              393M   123k  393M   1% /dev
devshm            393M   0     393M   0% /dev/shm
lrm               393M   35M   359M   9% /lib/modules/2.6.20-15-generic/volatile
/dev/sdb5         29G    5.4G  22G    20% /media/docs
/dev/sdb3         30G    5.9G  23G    21% /media/isomp3s
/dev/sda1         8.5G   4.3G  4.3G   51% /media/xp1
/dev/sda2         12G    6.5G  5.2G   56% /media/xp2
/dev/sdc1         40G    3.1G  35G    9% /media/backup
```

du command examples

du shows how much space one ore more files or directories is using.
```
$ du -sh
103M
```

-s option summarize the space a directory is using and -h option provides "Human-readable" output.

Networking commands:

These are most useful commands in my list while working on Linux server , this enables you to quickly troubleshoot connection issues e.g. whether other system is connected or not , whether other host is responding or not and while working for FIX connectivity for advanced trading system this tools saves quite a lot of time .

• finding host/domain name and IP address - **hostname**
• test network connection – **ping**
• getting network configuration – **ifconfig**
• Network connections, routing tables, interface statistics – **netstat**
• query DNS lookup name – **nslookup**
• communicate with other hostname – **telnet**
• outing steps that packets take to get to network host – **traceroute**
• view user information – **finger**
• checking status of destination host - **telnet**

Example of Networking commands in Unix

let's see some example of various networking command in Unix and Linux. Some of them are quite basic e.g. ping and telnet and some are more powerful e.g. nslookup and netstat. When you used these commands in combination of find and grep you can get anything you are looking for e.g. hostname, connection end points, connection status etc.

hostname

hostnamewith no options displays the machines host name
hostname –ddisplays the domain name the machine belongs to
hostname –fdisplays the fully qualified host and domain name
hostname –idisplays the IP address for the current machine

ping
It sends packets of information to the user-defined source. If the packets are received, the destination device sends packets back. Ping can be used for two purposes

1. To ensure that a network connection can be established.
2. Timing information as to the speed of the connection.

If you **do ping www.yahoo.com** it will display its IP address. Use ctrl+C to stop the test.

ifconfig
View network configuration, it displays the current network adapter configuration. It is handy to determine if you are getting transmit (TX) or receive (RX) errors.

netstat
Most useful and very versatile for finding connection to and from the host. You can find out all the multicast groups (network) subscribed by this host by issuing **"netstat -g"**

netstat -nap | grep portwill display process id of application which is using that port

netstat -a or netstat –allwill display all connections including TCP and UDP

netstat --tcp or netstat –twill display only TCP connection

netstat --udp or netstat –uwill display only UDP connection

netstat -gwill display all multicast network subscribed by this host.

nslookup

If you know the IP address it will display hostname. To find all the IP addresses for a given domain name, the command nslookup is used. You must have a connection to the internet for this utility to be useful.

E.g. **nslookup blogger.com**

You can also use nslookup to convert hostname to IP Address and from IP Address from hostname.

traceroute

A handy utility to view the number of hops and response time to get to a remote system or web site is traceroute. Again you need an internet connection to make use of this tool.

finger

View user information, displays a user's login name, real name, terminal name and write status. this is pretty old unix command and rarely used now days.

telnet

Connects destination host via telnet protocol, if telnet connection establish on any port means connectivity between two hosts is working fine.

telnet hostname port   will telnet hostname with the port specified. Normally it is used to see whether host is alive and network connection is fine or not.

10 Most important linux networking commands

Linux is most powerful operating system which often needs to use commands to explore it effectively.Some of the commands are restricted to normal user groups as they are powerful and has more functionality involved in it.Here we summarized most interesting and useful networking commands which every linux user are supposed to   be   familiar   with   it.

1.Arp manipulates the kernel's ARP cache in various ways. The primary options are clearing an address mapping entry and manually setting up one. For debugging purposes, the arp program also allows a complete dump of the ARP cache.ARP displays the IP address assigned to particular ETH card and mac address

```
[fasil@smashtech]# arp
Address              HWtype  HWaddress        Flags Mask        Iface
59.36.13.1           ether     C               eth0
```

2.Ifconfig is used to configure the network interfaces. Normally we use this command to check the IP address assigned to the system.It is used at boot time to set up interfaces as necessary. After that, it is usually only needed when debugging or when system tuning is needed.

[fasil@smashtech~]#/sbin/ifconfig
eth0   UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
         RX packets:126341 errors:0 dropped:0 overruns:0 frame:0
         TX packets:44441 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000

3. Netstat prints information about the networking subsystem. The type of information which is usually printed by netstat are Print network connections, routing tables, interface statistics, masquerade connections, and multicast.

[fasil@smashtech ~]# netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address   Foreign Address   State
tcp   0   0   .230.87:https   ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags   Type   State   I-Node Path
unix 10   [ ]   DGRAM   4970   /dev/log
unix 2   [ ]   DGRAM   6625   @/var/run/hal/hotplug_socket
unix 2   [ ]   DGRAM   2952   @udevd
unix 2   [ ]   DGRAM   100564
unix 3   [ ]   STREAM   CONNECTED   62438   /tmp/.X11-unix/X0
unix 3   [ ]   STREAM   CONNECTED   62437
unix 3   [ ]   STREAM   CONNECTED   10271   @/tmp/fam-root-
unix 3   [ ]   STREAM   CONNECTED   10270
unix 3   [ ]   STREAM   CONNECTED   9276
unix 3   [ ]   STREAM   CONNECTED   9275

4.ping command is used to check the connectivity of a system to a network.Whenever there is problem in network connectivity we use ping to ensure the system is connected to network.

[root@smashtech ~]# ping google.com
PING google.com (74.125.45.100) 56(84) bytes of data.
64 bytes from yx-in-f100.google.com (74.125.45.100): icmp_seq=0 ttl=241 time=295 ms
64 bytes from yx-in-f100.google.com (74.125.45.100): icmp_seq=1 ttl=241 time=277 ms
64 bytes from yx-in-f100.google.com (74.125.45.100): icmp_seq=2 ttl=241 time=277 ms

--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 6332ms
rtt min/avg/max/mdev = 277.041/283.387/295.903/8.860 ms, pipe 2

5.Nslookup is a program to query Internet domain name servers. Nslookup has two modes: interactive and non-interactive. Interactive mode allows the user to query name servers for information about various hosts and domains or to print a list of hosts in a domain. Non-interactive mode is used to print just the name and

requested information for a host or domain.

[fasil@smashtech ~]# nslookup google.com
Server: server ip
Address: gateway ip 3

Non-authoritative answer:
Name: google.com
Address: 209.85.171.100
Name: google.com
Address: 74.125.45.100
Name: google.com
Address: 74.125.67.100

6. dig (domain information groper) is a flexible tool for interrogating DNS name servers. It performs DNS lookups and displays the answers that are returned from the name server(s) that were queried. Most DNS administrators use dig to troubleshoot DNS problems because of its flexibility, ease of use and clarity of output. Other lookup tools tend to have less functionality than dig.

[fasil@smashtech ~]# dig google.com

; <<>> DiG 9.2.4 <<>> google.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4716
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;google.com.   IN   A

;; ANSWER SECTION:
google.com.   122   IN   A   74.125.45.100
google.com.   122   IN   A   74.125.67.100
google.com.   122   IN   A   209.85.171.100

;; AUTHORITY SECTION:
google.com.   326567 IN   NS   ns3.google.com.
google.com.   326567 IN   NS   ns4.google.com.
google.com.   326567 IN   NS   ns1.google.com.
google.com.   326567 IN   NS   ns2.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.   152216 IN   A   216.239.32.10
ns2.google.com.   152216 IN   A   216.239.34.10
ns3.google.com.   152216 IN   A   216.239.36.10
ns4.google.com.   152216 IN   A   216.239.38.10

;; Query time: 92 msec

;; SERVER: 172.29.36.1#53(172.29.36.1)
;; WHEN: Thu Mar 5 14:38:45 2009
;; MSG SIZE rcvd: 212

7.Route manipulates the IP routing tables. Its primary use is to set up static routes to specific
hosts or networks via an interface after it has been configured with the ifconfig program.When the add or del
options are used, route modifies the routing tables. Without these options, route displays the current contents
of the routing tables.

[fasil@smashtech ~]# route
Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
54.192.56.321	*	255.255.255.0	U	0	0	0	eth0
	*	255.255.0.0	U	0	0	0	eth0
default	0.0.0.0		UG	0	0	0	eth0

8.Traceroute : Internet is a large and complex aggregation of network hardware, connected together by
gateways. Tracking the route one's packets follow (or finding the miscreant gateway that's
discarding your packets) can be difficult.

Traceroute utilizes the IP protocol 'time to live' field and attempts to elicit an ICMP TIME_EXCEEDED
response from each gateway along the path to some host. The only mandatory parameter is the destination host
name or IP number. The default probe datagram length is 40 bytes, but this may be increased by specifying a
packet length (in bytes) after the destination host name.

[fasil@smashtech ~]# traceroute google.com
traceroute: Warning: google.com has multiple addresses; using 209.85.171.100
traceroute to google.com (209.85.171.100), 30 hops max, 38 byte packets
 1 * * *

9.W-displays information about the users currently on the machine, and their processes. The header shows, in
this order, the current time, how long the system has been running, how many users are currently logged on,
and the system load averages for the past 1, 5, and 15 minutes.

[fasil@smashtechl ~]# w
 15:18:22 up 4:38, 3 users, load average: 0.89, 0.34, 0.19

USER	TTY	FROM	LOGIN@	IDLE	JCPU	PCPU	WHAT
root	:0	-	10:41	?xdm?	24:53	1.35s	/usr/bin/gnome-session
root	pts/1	:0.0	10:58	1.00s	0.34s	0.00s	w
root	pts/2	:0.0	12:10	23:32	0.03s	0.03s	bash

Filters:

Filters are commands which accept data from standard input, manupulate it and write the results to standard
output

Head: command displays the top of the file, when used without any option it will display first 10 lines of the file

$ head sample1.txt

/*display first 10 lines*/

tail:command displays the end of the file. By default it will display last 10 lines of the

file

$ tail sample1.txt

/*display last 10 lines*/

tail or head with -n followed by a number will display that many number of lines from last and from first respectively

$ head -n 20 sample1.txt

/* will display first 20 lines*/

$ tail -n 15 sample1.txt

/* will display last 15 lines */

cut : cutting columns

cut command can be used to cut the columns from a file with -c option

usage: cut -c [numbers delemited by comma or range]

<file name>

$ cut -c 1,2,3-5 students.txt

1 ani

2 das

3 shu

4 sin

cut : cutting fields

 With -f option you can cut the feilds delemited by some character

$ cut -d" " -f1,4 students.txt

1 Mtech

2 Btech

3 Mtech

 -d option is used to specify the delimiter and -f option used to specify the feild number

paste : pasting side by side

paste command will paste the contents of the file side by side

$ paste cutlist1.txt cutlist2.txt

1 Mtech 1 anil H1

2 Btech 2 dasgupta H4

3 Mtech 3 shukla H7

4 Mtech 4 singhvi H12

5 Btech 5 sumit H13

sort : ordering a file

sort re-orders lines in ASCII collating sequenceswhitespaces first, then numerals, uppercase and finally lowercase

we can sort the file based on a field by using -t and –k option.

$ sort -t" " -k 2 students.txt

/* sorts the file based on the second field

using the delimiter as space*/

grep : searching for a pattern

grep scans its input for a pattern, and can display the selected pattern, the line numbers or the filename where the pattern occurs.

usage: grep options pattern filename(s)

VIVA-VOCE

1. How would you kill a process?

The **kill all** command is used to kill processes by name.

2. What is a Directory?

A directory is a file the solo job of which is to store the file names and the related information.

3. What are the different file types available?

In Linux, everything is considered as a file. In UNIX, seven standard file types are regular, directory, symbolic link, FIFO special, block special, character special, and socket.

4. What are the duties of the following commands: chmod, chown, chgrp?

The **chown** command changes the owner of a file, and the **chgrp** command changes the group

5. Explain the method of changing file access permission?

To change the file or the directory permissions, you use the **chmod (change mode) command**. There are two ways to use chmod — the symbolic mode and the absolute mode.

Week-2:

NAME OF THE EXPERIMENT: SHELL SCRIPT

AIM: a) Write a shell script to find factorial of a given integer.

ALGORITHM:

Step 1: Start

Step 2: Read any number to find factorial

Step 3: initialize fact=1 and i=1

Step 4: while i less than do

Fact=fact*i

i=i+1

Step 4:print fact

Step 5: stop

Source code :

echo enter a number

read a

i=2

fact=1

if [$a -ge 2]

then

while [$i -le $a]

do

fact=`expr $fact \* $i`

i=`expr $i + 1`

done

fi

echo factorial of $a=$fact

output:

[latha@localhost ~]$ sh fact.sh

enter a number

5

factorial of 5=120

AIM:b) write a shell script to wish 'Good Morning' and 'Good Evening' depending on the system time.

ALGORITHM:

Step 1: Start

Step 2: read system time in hours Am and Pm attributes

hour=`date +%I

ampm=`date +%p`

Step 3:if **ampm** equal to AM print 'Good Morning'

Step 4:else if hours in between 12 to 4 print 'Good Afternoon'

Step 5:else print 'Good Evening'

Step 6: Stop

Source code :

```
h=$(date +"%H")
if [ $h -gt 6 -a $h -le 12 ]
then
echo good morning
elif [ $h -gt 12 -a $h -le 16 ]
then
echo good afternoon
elif [ $h -gt 16 -a $h -le 20 ]
then
echo good evening
else
echo good night
```

fi

output:

$ good evening

VIVA-VOCE

1. **What is Shell Script?**

A Shell Script is a text file that contains one or more commands.

2. **what is used in shell script?**

Shell scripts **allow us to program commands in chains and have the system execute them as a scripted event**, just like batch files.

3. **What is the basic syntax of while loop in shell scripting?**

while[test_condition]

do

commands…

done

4. **What are the different type of variables used in a shell Script?**

In Linux shell script we can use two types of variables :

o System defined variables

o User defined variables

5. **What is difference between $$ and $!?**

$$ gives the process id of the currently executing process whereas $! Shows the process id of the process that recently went into the background.

Week-3:

NAME OF THE EXPERIMENT: Linux cat command using File API s.

AIM: Implement Linux cat command using File API s**.**

ALGORITHM:

Step 1: Start

Step 2: open Terminal

Step 3:give the cat command and check with the different type of the options

Step 4:observe the result of the each options

Step 5: Stop

SOURCE CODE:cat :concatenate files and print on the standard output

Usage: cat [OPTION] [FILE]...

eg. cat file1.txt file2.txt

cat n

file1.txt

echo :display a line of text

Usage: echo [OPTION] [string] ...

eg. echo I love India

echo $HOME

WC:print the number of newlines, words, and bytes in files

Usage: wc [OPTION]... [FILE]...

eg. wc file1.txt

wc L

file1.txt

sort:sort lines of text files

Usage: sort [OPTION]... [FILE]...

eg. sort file1.txt

sort r

file1.txt

VIVA-VOCE

1.How Copy the contents of one file to another file?

The Linux cp command is used for copying files and directories to another location. To copy a file, specify "cp" followed by the name of a file to copy. Then, state the location at which the new file should appear. The new file does not need to have the same name as the one you are copying.

2.What is Cat command to open dashed files?

The cat command is a utility command in Linux. One of its most commonly known usages is to **print the content of a file onto the standard output stream**. Other than that, the cat command also allows us to write some texts into a file.

cat < - and ./- **command**

3.What is Cat command to write in an already existing file?

 Cat >>filename

4. **How to view a single file?**

The simplest way to view text files in Linux is the cat command. It displays the complete contents in the command line without using inputs to scroll through it. Here is an example of using the cat command to view the Linux version by displaying the contents of the /proc/version file.

5. What is the Cat command to merge the contents of multiple files?

```
$ cat file1.txt file2.txt file3.txt >> merge.txt
```

Week-4:

NAME OF THE EXPERIMENT: Linux commands cp , mv using Linux system calls

AIM: Implement the Linux commands (a) cp   (b) mv using Linux system calls.

ALGORITHM:

Step 1: Start

Step 2: open Terminal

Step 3:give the cp and mv command and check with the different type of the options

Step 4:observe the result of the each options

Step 5: Stop

Program:

cp :copy files and directories

Usage: cp [OPTION]... SOURCE DEST

eg. cp sample.txt sample_copy.txt

cp :sample_copy.txt target_dir

mv : move (rename) files

Usage: mv [OPTION]... SOURCE DEST

eg. mv source.txt target_dir

mv old.txt new.txt

VIVA-VOCE

1. Which are the Linux Directory Commands?

Directory Command	Description
pwd	The pwd command stands for (print working directory). It displays the current working location or directory of the user. It displays the whole working path starting with /. It is a built-in command.

ls	The ls command is used to show the list of a folder. It will list out all the files in the directed folder.
cd	The cd command stands for (change directory). It is used to change to the directory you want to work from the present directory.
mkdir	With mkdir command you can create your own directory.
rmdir	The rmdir command is used to remove a directory from your system.

2. How to copy a file in Linux?

The Linux cp command is used for copying files and directories to another location. To copy a file, specify "cp" followed by the name of a file to copy. Then, state the location at which the new file should appear. The new file does not need to have the same name as the one you are copying**.**

.

3. How to rename a file in Linux using cp command?

If we want to rename and copy at the same time, then we use the following command.

cp program3.cpp homework6.cpp

4. How to rename a file in Linux using mv command?

The mv command is used to rename the file in the Linux system. For this, we need the current_name and new_name of the directory along with the mv command. We use the following syntax to rename the file.

mv <current_name> <new_name>

To rename a file, we just used the mv command along with current_name and new_name

5. How would you delete a directory in Linux?

1. To remove an empty directory, use either rmdir or rm -d followed by the directory name: rm -d dirname rmdir dirname.

2. To remove non-empty directories and all the files within them, use the rm command with the -r (recursive) option: rm -r dirname.

Week-5:

NAME OF THE EXPERIMENT: Creating a child process and allow the parent to display 'parent'.

AIM: Write a C program to create a child process and allow the parent to display 'parent' and the child to display 'child' on the screen.

ALGORITHM:

Step 1: Start the main function

Step 2: call the fork() function to create a child process fork function returns 2 values

Step 3:which returns 0 to child process

Step 4:which returns process id to the parent process

Step 5:stop

Source Code:

```
include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<string.h>
int global=10;
int main()
{
int local=20;
pid_t pid;
printf("before fork\n");
printf("pid=%d,global=%d,local=%d\n",getpid(),global,local);
pid=fork();
if(pid<0)
printf("failed to create child");
else if(pid==0)
{
```

```
printf("after fork\n");

global++;

local++;

}

else if(pid>0)

sleep(2);

printf("cid=%d,global=%d,local=%d\n",getpid(),global,local);

exit(0);

}
```

Output:

[latha@localhost ~]$ cc week16.c

[latha@localhost ~]$ ./a.out

before fork

pid=3005,global=10,local=20

after fork

cid=3006,global=11,local=21

cid=3005,global=10,local=20

VIVA-VOCE

1. Why are we using fork() function call?

System call **fork()** is used to create processes. It takes no arguments and returns a process ID. The purpose of **fork()** is to create a *new* process, which becomes the *child* process of the caller. After a new child process is created, *both* processes will execute the next instruction following the *fork()* system call.

2. What is meant by pid and ppid?

1. **PID stands for Process ID**, Which means Identification Number for currently running process in Memory. **2. PPID stands for Parent Process ID**, Which means Parent Process is the responsible for creating the current process(Child Process)

3.**What is a process?**

A Process is something that is currently under execution. So, an active program can be called a Process.

4. What is meant by system call?

A system call is a method for a computer program to request a service from the kernel of the operating system on which it is running.

5.Why do we create a child process?

Sometimes there is a need for a program to perform more than one function simultaneously. Since these jobs may be interrelated so two different programs to perform them cannot be created.

Week-6:

NAME OF THE EXPERIMENT: Parent writes a message to a pipe and the child reads themessage.

AIM: Write a C program in which a parent writes a message to a pipe and the child reads themessage.

ALGORITHM:

Step 1: Start

Step 2: import libraryfiles

#include <errno.h>

#include<fcntl.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/wait.h>

#include <unistd.h>

Step 3:create an array of 2 size a[0] is for reading and a[1] is for writing over a pipe

Step 4:open pipe using pipe(a)

Step 5:write a string "Hi" in pipe using write() function

Step 6:read from pipe "Hi" message using read() function

Step 7:stop

SOURCE CODE:

```
#include <stdio.h>

#include<stdlib.h>

#include <sys/wait.h> /* contains prototype for wait */

int main(void)

{

int pid;

int status;

printf("Hello World!\n");

pid = fork( );
```
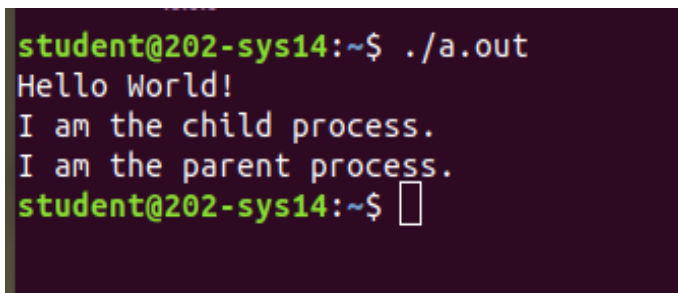
```
if(pid == -1) /* check for error in fork */

{

perror("bad fork");

exit(1);

}

if (pid == 0)

printf("I am the child process.\n");

else { wait(&status); /* parent waits for child to finish */

printf("I am the parent process.\n");

return 0;

}

}
```

OUTPUT:

```
student@202-sys14:~$ ./a.out
Hello World!
I am the child process.
I am the parent process.
student@202-sys14:~$
```

VIVA-VOCE

1. Why do we use piping?

A pipe is a form of redirection (transfer of standard output to some other destination) that is used in Linux and other Unix-like operating systems to send the output of one command/program/process to another command/program/process for further processing.

2. How does pipe work?

Pipe is used to combine two or more commands, and in this, the output of one command acts as input to another command, and this command's output may act as input to the next command and so on.

3.Different types of pipes?
   In Linux, we have two types of pipes: pipes (also known as anonymous or unnamed pipes) and FIFO's (also known as named pipes)

4. what is IPC?
Inter-process communication (IPC) is a mechanism that allows processes to communicate with each other and synchronize their actions. The communication between these processes can be seen as a method of co-operation between them.

5. What are the different system calls used in pipe?
The pipe system call finds the first two available positions in the process's open file table and allocates them for the read and write ends of the pipe.

Week-7:

NAME OF THE EXPERIMENT:CPU Scheduling Techniques FCFS , Priority

AIM: Write C Programs to simulate the following CPU scheduling algorithms:

   a)FCFS   b) Priority

a) FCFS ALGORITHM:

   Step 1: Start the process

   Step 2: Accept the number of processes in the ready Queue

   Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

   Step 4: Set the waiting of the first process as '0' and its burst time as its turnaround time

   Step 5: for each process in the Ready Q calculate

   Waiting time for process(n)= waiting time of process(n-1)+Burst time of process(n-1)

   Turn around time for Process(n)= waiting time of Process(n)+ Burst time forprocess(n)

   Step 6: Calculate

   Average waiting time = Total waiting Time / Number of process

   Average Turnaround time = Total Turnaround Time / Number of process

   Step 7: Stop the process

SOURCE CODE:

```
#include<stdio.h>

void main()

{

int pid[10],bt[10],wt[10],tat[10],n,twt=0,ttat=0,i;

float awt,atat;

printf("Enter no.of processes:");

scanf("%d",&n);

printf("\n Enter burst times:");

for(i=0;i<n;i++)

scanf("%d",&bt[i]);
```

```
wt[0]=0;

tat[0]=bt[0];

for(i=1;i<n;i++){

wt[i]=tat[i-1];

tat[i]=bt[i]+wt[i];

}

for(i=0;i<n;i++){

ttat= ttat+tat[i];

twt=twt+wt[i];

}

printf("\n PID \t BT \t WT \t TAT");

for(i=0;i<n;i++)

printf("\n %d\t%d\t%d\t%d",i+1,bt[i],wt[i],tat[i]);

awt=(float)twt/n;

atat=(float)ttat/n;

printf("\nAvg. Waiting Time=%f",awt);

printf("\nAvg. Turn around time=%f",atat);

}
```

Output:

b) PRIORITY ALGORITHM:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id, process priority and accept the CPU burst time

Step 4: Start the Ready Q according the highest priority by sorting according to highest to lowest priority.

Step 5: Set the waiting time of the first process as '0' and its turnaround time as its burst time.

Step 6: For each process in the ready queue, calculate

Waiting time for process(n)= waiting time of process (n-1)+Burst time of process(n-1)

Turn around time for Process(n)= waiting time of Process(n)+ Burst time for process(n)

Step 6: Calculate

Average waiting time = Total waiting Time / Number of process

Average Turnaround time = Total Turnaround Time / Number of process

Step 7: Stop the process

SOURCE CODE:

```
#include<stdio.h>

void main()

{

int pid[10],bt[10],pr[10],wt[10],tat[10],n,twt=0,ttat=0,i,j,t;

float awt,atat;

printf("Enter no.of processes:");

scanf("%d",&n);

printf("\n Enter burst times:");

for(i=0;i<n;i++)

scanf("%d",&bt[i]);

printf("\n Enter PID:");

for(i=0;i<n;i++)
```

```
scanf("%d",&pid[i]);

printf("\n Enter Priorities:");

for(i=0;i<n;i++)

scanf("%d",&pr[i]);

for(i=0;i<n;i++){

for(j=i+1;j<n;j++){

if(pr[i]>pr[j]){

t=pr[i];

pr[i]=pr[j];

pr[j]=t;


t=bt[i];

bt[i]=bt[j];

bt[j]=t;


t=pid[i];

pid[i]=pid[j];

pid[j]=t;

}}}

wt[0]=0;

tat[0]=bt[0];

for(i=1;i<n;i++){

wt[i]=tat[i-1];

tat[i]=bt[i]+wt[i];

}

for(i=0;i<n;i++){
```

```
ttat= ttat+tat[i];

twt=twt+wt[i];

}

printf("\n PID PRIORITY \t BT \t WT \t TAT");

for(i=0;i<n;i++)

printf("\n %d\t%d\t%d\t%d\t%d",pid[i],pr[i],bt[i],wt[i],tat[i]);

awt=(float)twt/n;

atat=(float)ttat/n;

printf("\nAvg. Waiting Time=%f",awt);

printf("\nAvg. Turn around time=%f",atat);

}
```

Output:

VIVA-VOCE

1.What is an Operating system?

Operating System (OS), program that manages a computer's resources, especially the allocation of those resources among other programs. Typical resources include the central processing unit (CPU), computer memory, file storage, input/output (I/O) devices, and network connections.

2.What is a process ?

A process is an 'active' entity, instead of a program, which is considered a 'passive' entity. A single program can create many processes when run multiple times; for example, when we open a .exe or binary file multiple times, multiple instances begin (multiple processes are created)

3.**What Are The Advantages of A Multiprocessor System?**

The advantages of the multiprocessing system are: Increased Throughput − By increasing the number of processors, more work can be completed in a unit time. Cost Saving − Parallel system shares the memory, buses, peripherals etc. Multiprocessor system thus saves money as compared to multiple single systems.

4. Explain starvation and Aging

Starvation is the problem that occurs when high priority processes keep executing and low priority processes get blocked for indefinite time. In heavily loaded computer system, a steady stream of higher-priority processes can prevent a low-priority process from ever getting the CPU.

Aging is **a technique of gradually increasing the priority of processes that wait in the system for a long time**. For example, if priority range from 127(low) to 0(high), we could increase the priority of a waiting process by 1 Every 15 minutes.

5.What are the functions of operating system?

An operating system has three main functions: (1) manage the computer's resources, such as the central processing unit, memory, disk drives, and printers, (2) establish a user interface, and (3) execute and provide services for applications software.

Week-8:

NAME OF THE EXPERIMENT:CPU Scheduling Techniques SJF, Round Robin

AIM: Write C Programs to simulate the following CPU scheduling algorithms:

a)SJF   b) Round Robin

a) SJF ALGORITHM:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Start the Ready Q according the shortest Burst time by sorting according to lowest to highest burst time.

Step 5: Set the waiting time of the first process as '0' and its turnaround time as its burst time.

Step 6: For each process in the ready queue, calculate

Waiting time for process(n)= waiting time of process (n-1)+Burst time of process(n-1)

Turn around time for Process(n)= waiting time of Process(n)+ Burst time for process(n)

Step 7: Calculate

Average waiting time = Total waiting Time / Number of process

Average Turnaround time = Total Turnaround Time / Number of process

Step 8: Stop the process

SOURCE CODE:

```
#include<stdio.h>

void main(){

int pid[10],bt[10],wt[10],tat[10],n,twt=0,ttat=0,i,j,t;

float awt,atat;

printf("Enter no.of processes:");

scanf("%d",&n);

printf("\n Enter burst times:");

for(i=0;i<n;i++)

scanf("%d",&bt[i]);
```

```
printf("\n Enter PID:");

for(i=0;i<n;i++)

scanf("%d",&pid[i]);

for(i=0;i<n;i++)

{

for(j=i+1;j<n;j++)

{

if(bt[i]>bt[j])

{

t=bt[i];

bt[i]=bt[j];

bt[j]=t;


t=pid[i];

pid[i]=pid[j];

pid[j]=t;

}}}

wt[0]=0;

tat[0]=bt[0];

for(i=1;i<n;i++)

{

wt[i]=tat[i-1];

tat[i]=bt[i]+wt[i];

}

for(i=0;i<n;i++)

{
```

ttat= ttat+tat[i];

twt=twt+wt[i];

}

printf("\n PID \t BT \t WT \t TAT");

for(i=0;i<n;i++)

printf("\n %d\t%d\t%d\t%d",pid[i],bt[i],wt[i],tat[i]);

awt=(float)twt/n;

atat=(float)ttat/n;

printf("\nAvg. Waiting Time=%f",awt);

printf("\nAvg. Turn around time=%f",atat);

}

OUTPUT :

b) ROUND ROBIN ALGORITHM:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue & time quantum or time slice

Step 3: For each process in the ready Q, assign the process id &accept the CPU burst time

Step 4: Calculate the no. of time slices for each process where No. of time slice for process(n) = burst time process(n)/time slice

Step 5: If the burst time is less than the time slice then the no. of time slices =1.

Step 6: Consider the ready queue is a circular Q, calculate

Waiting time for process(n) = waiting time of process(n-1)+burst time of process(n-1) + the time difference in getting the CPU from process(n-1)

Turn around time for process(n) = waiting time of process(n) + burst time of process(n) + the time difference in getting CPU from process(n).

Step 7: Calculate

Average waiting time = Total waiting Time / Number of process

Average Turnaround time = Total Turnaround Time / Number of process

Step 8: Stop the process

SOURCE CODE:

```
#include<stdio.h>

void main()

{

int ts,bt1[10],wt[10],tat[10],i,j=0,n,bt[10],ttat=0,twt=0,tot=0;

float awt,atat;

printf("Enter the number of Processes \n");

scanf("%d",&n);

printf("\n Enter the Timeslice \n");

scanf("%d",&ts);

printf("\n Enter the Burst Time for the process");

for(i=1;i<=n;i++){
```

```c
scanf("%d",&bt1[i]);

bt[i]=bt1[i];

}

while(j<n){

for(i=1;i<=n;i++){

if(bt[i]>0){

if(bt[i]>=ts){

tot+=ts;

bt[i]=bt[i]-ts;

if(bt[i]==0){

j++;

tat[i]=tot;

}}

else{

tot+=bt[i];

bt[i]=0;

j++;

tat[i]=tot;

}}}}

for(i=1;i<=n;i++){

wt[i]=tat[i]-bt1[i];

twt=twt+wt[i];

ttat=ttat+tat[i];

}

awt=(float)twt/n;

atat=(float)ttat/n;
```

```
printf("\n PID \t BT \t WT \t TAT\n");

for(i=1;i<=n;i++) {

printf("\n %d \t %d \t %d \t %d \t\n",i,bt1[i],wt[i],tat[i]);

}

printf("\n The average Waiting Time=%f",awt);

printf("\n The average Turn around Time=%f",atat);

}
```
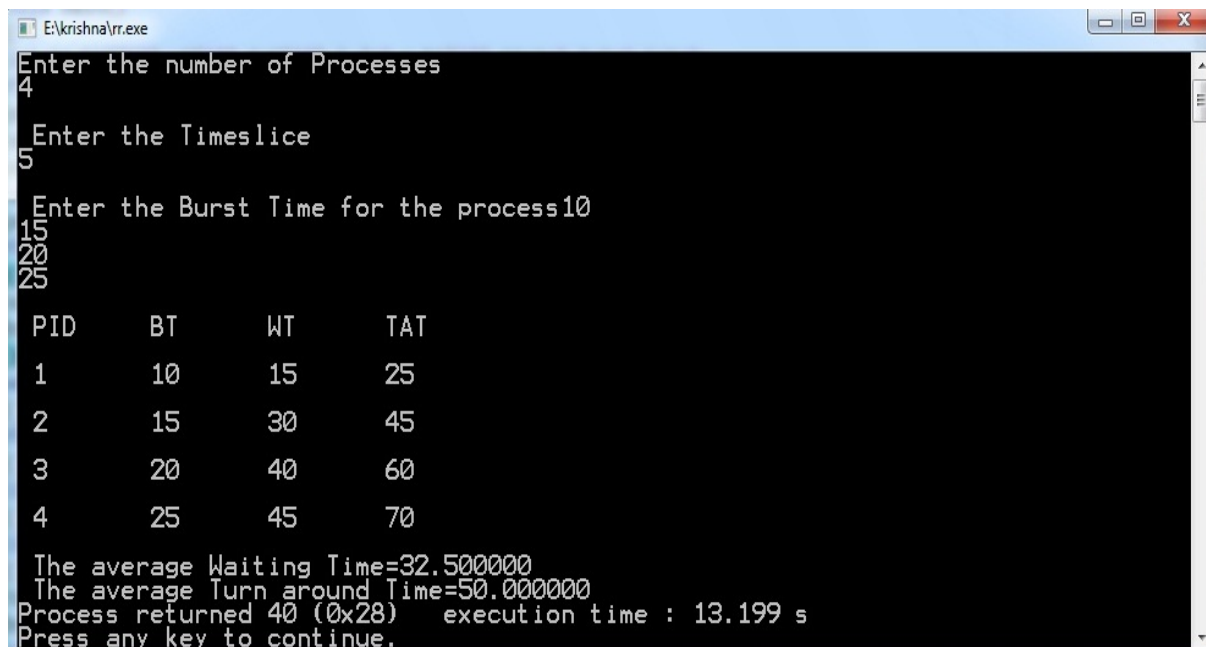
OUTPUT :

VIVA –VOCE

1. **List different CPU Scheduling algorithms.**
   The different CPU algorithms are:

   ● First Come First Serve.

   ● Shortest Job First.

   ● Shortest Remaining Time First.

   ● Round Robin Scheduling.

   ● Priority Scheduling.

   ● Multilevel Queue Scheduling.

   ● Multilevel Feedback Queue Scheduling.

2. What is FCFS Scheduling?

First Come First Serve (FCFS) is an operating system scheduling algorithm that automatically executes queued requests and processes in order of their arrival. It is the easiest and simplest CPU scheduling algorithm. In this type of algorithm, processes which requests the CPU first get the CPU allocation first.

3. What is SJF Scheduling?

Shortest Job First (SJF) is an algorithm in which the process having the smallest execution time is chosen for the next execution. This scheduling method can be preemptive or non-preemptive. It significantly reduces the average waiting time for other processes awaiting execution.

4. What is RR Scheduling?

Round Robin is **a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way**. It is simple, easy to implement, and starvation-free as all processes get fair share of CPU. One of the most commonly used technique in CPU scheduling as a core.

5. What is Priority Scheduling?

Priority Scheduling is **a method of scheduling processes that is based on priority**. In this algorithm, the scheduler selects the tasks to work as per the priority. The processes with higher priority should be carried out first, whereas jobs with equal priorities are carried out on a round-robin or FCFS basis.

Week-9:

NAME OF THE EXPERIMENT: File allocation techniques

AIM: Write C programs to simulate the following Fileallocation strategies

a)Sequential   b)Linked c)Indexed

a) **Sequential Algorithm:**

Step 1: Start the program.

Step 2: Get the number of memory partition and their sizes.

Step 3: Get the number of processes and values of block size for each process.

Step 4: First fit algorithm searches the entire   memory block until a hole which is big enough is encountered. It allocates that memory block for the requesting process.

Step 5: Best-fit algorithm searches the memory blocks for the smallest hole which can be allocated to requesting process and allocates if.

Step 6: Worst fit algorithm searches the memory blocks for the largest hole and allocates it to the process.

Step 7: Analyses all the three memory management techniques and display the best algorithm which utilizes the memory resources effectively and efficiently.

Step 8: Stop the program

SOURCE CODE:

```
#include<stdio.h>

main()

{

int n,i,j,b[20],sb[20],t[20],x,c[20][20];

printf("Enter no.of files:");

scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("Enter no. of blocks occupied by file%d",i+1);

scanf("%d",&b[i]);

printf("Enter the starting block of file%d",i+1);
```

```
scanf("%d",&sb[i]);

t[i]=sb[i];

for(j=0;j<b[i];j++)

c[i][j]=sb[i]++;

}

printf("Filename\tStart block\tlength\n");

for(i=0;i<n;i++)

printf("%d\t %d \t%d\n",i+1,t[i],b[i]);

printf("Enter file name:");

scanf("%d",&x);

printf("\nFile name is:%d",x);

printf("\nlength is:%d",b[x-1]);

printf("\nblocks occupied:");

for(i=0;i<b[x-1];i++)

printf("%4d",c[x-1][i]);

}
```

Output:

b) LinkedAlgorithm:

      Step 1: Start the program.

      Step 2: Read the number of files

      Step 3: For each file, read the file name, starting block and number of blocks and block numbers of the file.

      Step 4: Start from the starting block and link each block of the file to the next block in a linked list fashion.

      Step 5: Display the file name, starting block, size of the file & the blocks occupied by the file.

      Step 6: Stop the program

SOURCE CODE:

```c
#include<stdio.h>

struct file

{

char fname[10];

int start,size,block[10];

}f[10];

main()

{

int i,j,n;

printf("Enter no. of files:");

scanf("%d",&n);

for(i=0;i<n;i++){

printf("Enter file name:");

scanf("%s",&f[i].fname);

printf("Enter starting block:");

scanf("%d",&f[i].start);

f[i].block[0]=f[i].start;

printf("Enter no.of blocks:");

scanf("%d",&f[i].size);
```

```
printf("Enter block numbers:");

for(j=1;j<f[i].size;j++){

scanf("%d",&f[i].block[j]);

}}

printf("File\tstart\tsize\tblock\n");

for(i=0;i<n;i++){

printf("%s\t%d\t%d\t",f[i].fname,f[i].start,f[i].size);

for(j=0;j<f[i].size-1;j++)

printf("%d--->",f[i].block[j]);

printf("%d\n",f[i].block[j]);

}}
```
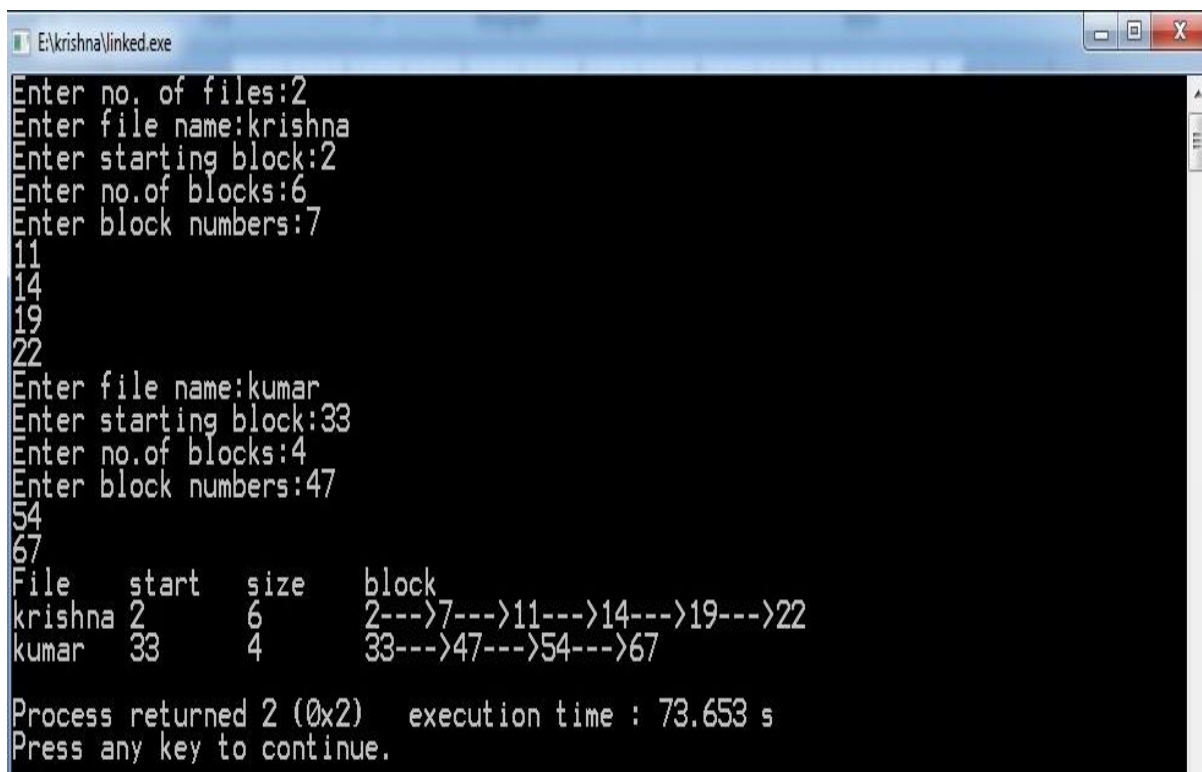
Output:

   c) IndexedAlgorithm:

      Step 1: Start the program.

Step2: Read the number of files

Step 3: Read the index block for each file.

Step 4: For each file, read the number of blocks occupied and number of blocks of the file.

Step 5: Link all the blocks of the file to the index block.

Step 6: Display the file name, index block, and the blocks occupied by the file.

Step 7: Stop the program

SOURCE CODE:
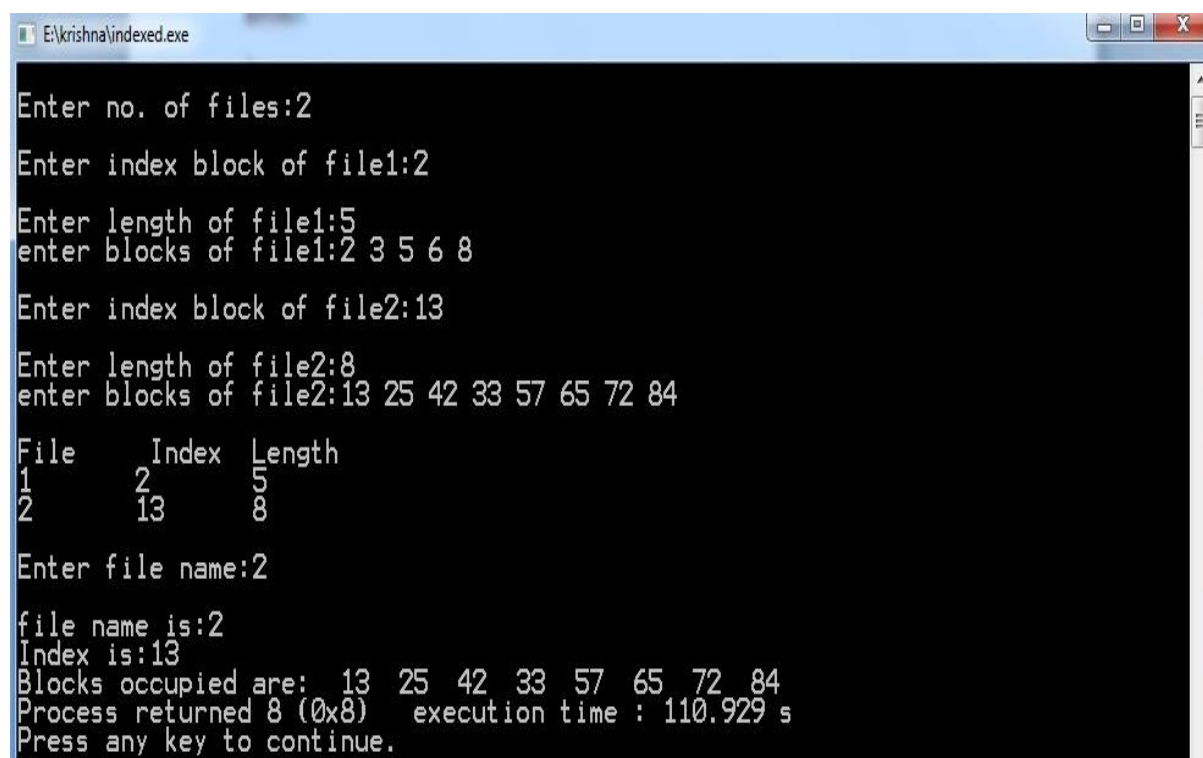
```c
#include<stdio.h>

main()

{

int n,m[20],i,j,sb[20],b[20][20],x;

printf("\nEnter no. of files:");

scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("\nEnter index block of file%d:",i+1);

scanf("%d",&sb[i]);

printf("\nEnter length of file%d:",i+1);

scanf("%d",&m[i]);

printf("enter blocks of file%d:",i+1);

for(j=0;j<m[i];j++)

scanf("%d",&b[i][j]);

}

printf("\nFile\t Index\tLength\n");

for(i=0;i<n;i++)

{

printf("%d\t%d\t%d\n",i+1,sb[i],m[i]);
```

```
}

printf("\nEnter file name:");

scanf("%d",&x);

printf("\nfile name is:%d",x);

printf("\nIndex is:%d",sb[x-1]);

printf("\nBlocks occupied are:");

for(j=0;j<m[x-1];j++)

printf("%4d",b[x-1][j]);

}
```

OUTPUT:

VIVA-VOCE

1. List File access methods.

There are three ways to access a file into a computer system: Sequential-Access, Direct Access, Index sequential Method.
- Sequential Access – It is the simplest access method. ...
- Direct Access – Another method is direct access method also known as relative access method. ...
- Index sequential method

2. Explain Sequential file allocation method.

In this allocation strategy, each file occupies a set of contiguously blocks on the disk. This strategy is best suited. For sequential files, the file allocation table consists of a single entry for each file. It shows the filenames, starting block of the file and size of the file**.**

3. Explain Linked file allocation method.

Each file is a linked list of disk blocks which need not be contiguous. The disk blocks can be scattered anywhere on the disk.
The directory entry contains a pointer to the starting and the ending file block. Each block contains a pointer to the next block occupied by the file.

4. Explain Indexed file allocation method.

Indexed allocation scheme stores all the disk pointers in one of the blocks called as indexed block. Indexed block doesn't hold the file data, but it holds the pointers to all the disk blocks allocated to that particular file. Directory entry will only contain the index block address.

5. What is file system mounting?

Mounting a file system attaches that file system to a directory (mount point) and makes it available to the system. The root (/) file system is always mounted. Any other file system can be connected or disconnected from the root (/) file system.

Week-10:

NAME OF THE EXPERIMENT: Memory management techniques

AIM: Write a C program to simulate the following memory management techniques

     a)   Paging   b) Segmentation

a) PAGING Algorithm:

     Step 1: Read all the necessary input from the keyboard.

     Step 2: Pages - Logical memory is broken into fixed - sized blocks.

     Step 3: Frames – Physical memory is broken into fixed – sized blocks.

     Step 4: Calculate the physical address using the following

     Physical address = (Frame number * Frame size) + offset

     Step 5: Display the physical address.

     Step 6: Stop the process

SOURCE CODE:

```
#include<stdio.h>

void main(){

        int i,j,temp,framearr[20],pages,pageno,frames,memsize,log,pagesize,prosize,base;

        printf("Enter the Process size: ");

        scanf("%d",&prosize);

        printf("\nEnter the main memory size: ");

        scanf("%d",&memsize);

        printf("\nEnter the page size: ");

        scanf("%d",&pagesize);

        pages=prosize/pagesize;

        printf("\nThe process is divided into %d pages",pages);

        frames = memsize/pagesize;

        printf("\n\nThe main memory is divided into %d frames\n",frames);

        for(i=0;i<frames;i++)
```

```
            framearr[i]=-1;        /* Initializing array elements with -1*/

     for(i=0;i<pages;i++){

     pos:    printf("\nEnter the frame number of page %d: ",i);

            scanf("%d",&temp);  /* storing frameno in temporary variable*/

            if(temp>=frames)  /*checking wether frameno is valid or not*/

            {

                    printf("\n\t****Invalid frame number****\n");

                    goto pos;

            }

            /* storing pageno (i.e 'i' ) in framearr at framno (i.e temp ) index */

            for(j=0;j<frames;j++)

                    if(temp==j)

                            framearr[temp]=i;

     }

     printf("\n\nFrameno\tpageno\tValidationBit\n------\t------\t----------");

     for(i=0;i<frames;i++){

            printf("\n  %d  \t  %2d  \t",i,framearr[i]);

            if(framearr[i]==-1)

                    printf("  0");

            else

                    printf("  1");

     }

     printf("\nEnter the logical address: ");

     scanf("%d",&log);

     printf("\nEnter the base address: ");

     scanf("%d",&base);
```

pageno = log/pagesize;

for(i=0;i<frames;i++)

       if(framearr[i]==pageno)

       {

           temp = i;

           break;

       }

j = log%pagesize;   /* here 'j' is displacement */

temp = base + (temp*pagesize)+j; //lhs 'temp' is physical address rhs and 'temp' is frame num

printf("\nPhysical address is : %d",temp);

}

Output:

    b) SEGMENTATION ALGORITHM

Step 1: Start the program.

Step 2: Get the number of segments.

Step 3: Get the base address and length for each segment.

Step 4: Get the logical address.

Step 5: Check whether the segment number is within the limit, if not display the error message.

Step 6: Check whether the byte reference is within the limit, if not display the error message.

Step 7: Calculate the physical memory and display it.

Step 8: Stop the program.

SOURCE CODE:

```c
#include<stdio.h>

void main(){
        int i,j,m,size,val[10][10],badd[20],limit[20],ladd;
        printf("Enter the size of the segment table:");
        scanf("%d",&size);
        for(i=0;i<size;i++){
                printf("\nEnter infor about segment %d",i+1);
                printf("\nEnter base address:");
                scanf("%d",&badd[i]);
                printf("\nEnter the limit:");
                scanf("%d",&limit[i]);
                for(j=0;j<limit[i];j++){
                        printf("\nEnter %d address values:",badd[i]+j);
                        scanf("%d",&val[i][j]);
                }}
        printf("\n\n****SEGMENT TABLE****");
        printf("\nseg.no\tbase\tlimit\n");
        for(i=0;i<size;i++)
```

```
    {
            printf("%d\t%d\t%d\n",i+1,badd[i],limit[i]);
    }
    printf("\nEnter segment no.::");
    scanf("%d",&i);
    if(i>=size)
    {
    printf("invalid");
    }
    else
    {
    printf("\nEnter the logical address:");
    scanf("%d",&ladd);
    if(ladd>=limit[i])
    printf("invalid");
    else
    {
    m=badd[i]+ladd;
    printf("\nmapped physical address is=%d",m);
    printf("\nthe value is %d ",val[i][ladd]);
    }   }}
```

OUTPUT:

```
E:\krishna\segement.exe

Enter the size of the segment table:3

Enter infor about segment 1
Enter base address:2

Enter the limit:3

Enter 2 address values:11

Enter 3 address values:12

Enter 4 address values:13

Enter infor about segment 2
Enter base address:22

Enter the limit:4

Enter 22 address values:33

Enter 23 address values:44

Enter 24 address values:55

Enter 25 address values:66

Enter infor about segment 3
Enter base address:75

Enter the limit:2

Enter 75 address values:88

Enter 76 address values:99


****SEGMENT TABLE****
seg.no   base     limit
1        2        3
2        22       4
3        75       2

Enter segment no.::1

Enter the logical address:2

mapped physical address is=24
the value is 55
Process returned 17 (0x11)   execution time : 86.104 s
Press any key to continue.
```

VIVA-VOCE

1.What is the basic function of paging?

Paging is a memory management scheme that permits the physical-address space of a process to be non contiguous. It avoids the considerable problem of having to fit varied sized memory chunks onto the backing store.

2.What is fragmentation?

Fragmentation is an unwanted problem in the operating system in which the processes are loaded and unloaded from memory, and free memory space is fragmented. Processes can't be assigned to memory blocks due to their small size, and the memory blocks stay unused.

3.What is thrashing?

Thrashing is caused by under allocation of the minimum number of pages required by a process, forcing it to continuously page fault.

4.Differentiate between logical and physical address.

Logical Address is generated by CPU while a program is running. The logical address is virtual address as it does not exist physically, therefore, it is also known as Virtual Address.

Physical Address identifies a physical location of required data in a memory. The user never directly deals with the physical address but can access by its corresponding logical address.

5.Explain internal fragmentation and external fragmentation.

Internal Fragmentation

When a process is allocated to a memory block, and if the process is smaller than the amount of memory requested, a free space is created in the given memory block. Due to this, the free space of the memory block is unused, which causes internal fragmentation.

External Fragmentation

External fragmentation happens when a dynamic memory allocation method allocates some memory but leaves a small amount of memory unusable. The quantity of available memory is substantially reduced if there is too much external fragmentation. There is enough memory space to complete a request, but it is not contiguous. It's known as external fragmentation

Week-11:

NAME OF THE EXPERIMENT: Bankers Algorithm for Deadlock Detection and Avoidance.

AIM: Write a C program to simulate Bankers Algorithm for Detection and Avoidance.

ALGORITHMFOR DETECTION:

Step 1: Start the program.

Step 2: Get the values of resources and processes.

Step 3: Get the WORK and FINISH with length m and n respectively.

**Step 4*:*Initialize Work= Available. For i=0, 1, …., n-1, if Requesti = 0

Step 5:Then Finish[i] = true; otherwise, Finish[i]= false.

.**Step 6:**Find an index i such that both
a) Finish[i] == false
b) Requesti <= Work .If no such i exists go to step 4.

.**Step 7:**Work= Work+ Allocationi
Finish[i]= true,Go to Step 2.

.**Step 8:**If Finish[i]== false for some i, 0<=i<n.

Step 9: Then the system is in a deadlocked state.

Step10: If Finish[i]==false the process Pi is deadlocked

Step 11: Stop the program.

ALGORITHM FOR AVOIDANCE:

Step 1: Start the program.

Step 2: Get the values of resources and processes.

Step 3: Get the avail value.

Step 4: After allocation find the need value.

Step 5: Check whether it's possible to allocate.

Step 6: If it is possible then the system is in safe state.

Step 7: Else system is not in safety state.

Step 8: If the new request comes then check that the system is in safety.

Step 9: Or not if we allow the request.

Step 10: Stop the program.

SOURCE CODE:

```c
#include<stdio.h>
static int mark[20];
int i,j,np,nr;

int main()
{
int alloc[10][10],request[10][10],avail[10],r[10],w[10];

printf("\nEnter the no of process: ");
scanf("%d",&np);
printf("\nEnter the no of resources: ");
scanf("%d",&nr);
for(i=0;i<nr;i++)
{
printf("\nTotal Amount of the Resource R%d: ",i+1);
scanf("%d",&r[i]);
}

printf("\nEnter the request matrix:");

for(i=0;i<np;i++)
for(j=0;j<nr;j++)
scanf("%d",&request[i][j]);

printf("\nEnter the allocation matrix:");
for(i=0;i<np;i++)
for(j=0;j<nr;j++)
scanf("%d",&alloc[i][j]);
/*Available Resource calculation*/
for(j=0;j<nr;j++)
{
avail[j]=r[j];
for(i=0;i<np;i++)
{
avail[j]-=alloc[i][j];

}
}

//marking processes with zero allocation

for(i=0;i<np;i++)
{
int count=0;
 for(j=0;j<nr;j++)
  {
    if(alloc[i][j]==0)
      count++;
    else
      break;
  }
 if(count==nr)
 mark[i]=1;
```

```
}
// initialize W with avail

for(j=0;j<nr;j++)
   w[j]=avail[j];

//mark processes with request less than or equal to W
for(i=0;i<np;i++)
{
int canbeprocessed=0;
 if(mark[i]!=1)
{
  for(j=0;j<nr;j++)
   {
    if(request[i][j]<=w[j])
     canbeprocessed=1;
    else
      {
      canbeprocessed=0;
      break;
       }
    }
if(canbeprocessed)
{
mark[i]=1;

for(j=0;j<nr;j++)
w[j]+=alloc[i][j];
}
}
}

//checking for unmarked processes
int deadlock=0;
for(i=0;i<np;i++)
if(mark[i]!=1)
deadlock=1;


if(deadlock)
printf("\n Deadlock detected");
else
printf("\n No Deadlock possible");
}
```

OUTPUT:

CASE-1

Enter the no of process: 4

Enter the no of resources: 5

Total Amount of the Resource R1: 2

Total Amount of the Resource R2: 1

Total Amount of the Resource R3: 1

Total Amount of the Resource R4: 2

Total Amount of the Resource R5: 1

Enter the request matrix:

 0 0 1 0 1

0 0 0 0 1

1 0 1 0 1

0 0 1 0 1

Enter the allocation matrix:

1 0 0 1 1

1 0 1 0 0

1 1 0 0 0

0 0 0 0 0

 Deadlock detected

CASE-2

Enter the no of process: 3

Enter the no of resources: 3

Total Amount of the Resource R1: 2

Total Amount of the Resource R2: 1

Total Amount of the Resource R3: 2

Enter the request matrix:1 0 0

1 0 1

0 0 1

Enter the allocation matrix:

0 0 1

1 0 1

0 0 0

 No Deadlock possible

DEADLOCK AVOIDANCE

```c
#include<stdio.h>

#include<conio.h>

int max[100][100];

int alloc[100][100];

int need[100][100];

int avail[100];

int n,r;

void input();

void show();

void cal();

int main()

{

int i,j;

printf("********** Banker's Algo ***********\n");

input();

show();

cal();

getch();

return 0;

}
```

```c
void input()

{

int i,j;

printf("Enter the no of Processes\t");

scanf("%d",&n);

printf("Enter the no of resources instances\t");

scanf("%d",&r);

printf("Enter the Max Matrix\n");

for(i=0;i<n;i++)

{

for(j=0;j<r;j++)

{

scanf("%d",&max[i][j]);

}

}

printf("Enter the Allocation Matrix\n");

for(i=0;i<n;i++)

{

for(j=0;j<r;j++)

{

scanf("%d",&alloc[i][j]);

}

}

printf("Enter the available Resources\n");

for(j=0;j<r;j++)

{

scanf("%d",&avail[j]);

}
```

```
}
void show()
{
int i,j;
printf("Process\t Allocation\t Max\t Available\t");
for(i=0;i<n;i++)
{
printf("\nP%d\t ",i+1);
for(j=0;j<r;j++)
{
printf("%d ",alloc[i][j]);
}
printf("\t");
for(j=0;j<r;j++)
{
printf("%d ",max[i][j]);
}
printf("\t");
if(i==0)
{
for(j=0;j<r;j++)
printf("%d ",avail[j]);
}
}
}
void cal()
{
int finish[100],temp,need[100][100],flag=1,k,c1=0;
```

```
int safe[100];

int i,j;

for(i=0;i<n;i++)

{

finish[i]=0;

}

//find need matrix

for(i=0;i<n;i++)

{

for(j=0;j<r;j++)

{

need[i][j]=max[i][j]-alloc[i][j];

}

}

printf("\n");

while(flag)

{

flag=0;

for(i=0;i<n;i++)

{

int c=0;

for(j=0;j<r;j++)

{

if((finish[i]==0)&&(need[i][j]<=avail[j]))

{

c++;

if(c==r)

{
```

```
for(k=0;k<r;k++)

{

avail[k]+=alloc[i][j];

finish[i]=1;

flag=1;

}

printf("P%d->",i);

if(finish[i]==1)

{

i=n;

}

}

}

}

}

for(i=0;i<n;i++)

{

if(finish[i]==1)

{

c1++;

}

else

{

printf("P%d->",i);

}

}

if(c1==n)
```

```
{

printf("\n The system is in safe state");

}

else

{

printf("\n Process are in dead lock");

printf("\n System is in unsafe state");

}

}
```

OUTPUT:

********** Banker's Algo ************

Enter the no of Processes   2 3

Enter the no of resources instances   3

Enter the Max Matrix

2 0 3

2 3 4

2 3 4

Enter the Allocation Matrix

1 2 3

2 1 3

1 1 1

Enter the available Resources

0 1 2

Process	Allocation	Max	Available
P1	1 2 3	2 0 3	
P2	2 1 3	2 3 4	
P3	1 1 1	2 3 4	

P0->P1->P2->

Process are in dead lock System is in unsafe state

VIVA-VOCE

1. What is deadlock?

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

2. What are the necessary conditions for deadlock?

Conditions for Deadlock- Mutual Exclusion, Hold and Wait, No preemption, Circular wait.

3. What is resource allocation graph?

The resource allocation graph is the pictorial representation of the state of a system. As its name suggests, the resource allocation graph is the complete information about all the processes which are holding some resources or waiting for some resources.

4. Explain safe state?

A state is safe if the system can allocate resources to each process (up to its maximum) in some order and still avoid a deadlock.

5.Explain critical section?
The critical section problem is used to design a protocol followed by a group of processes, so that when one process has entered its critical section, no other process is allowed to execute in its critical section.

Week-12:

NAME OF THE EXPERIMENT: Page Replacement Techniques

 AIM: Write C programs to simulate the following Page Replacement Techniques:

a) FIFO   b) LRU   c)OPTIMAL

 a)FIFO Algorithm:

       Step1: Start

       Step2: Read the number of frames

       Step3: Read the number of pages

       Step4: Read the page numbers

       Step5: Initialize the values in frames to -1

       Step6: Allocate the pages in to frames in First in first out order.

       Step7: Display the number of page faults.

       Step8: Stop

SOURCE CODE:

```
#include<stdio.h>

void main()

{

   int i,j,n,a[50],frame[10],fno,k,avail,pagefault=0;

printf("\nEnter the number of Frames : ");

scanf("%d",&fno);

printf("\nEnter number of reference string :");
```

```
scanf("%d",&n);

printf("\n Enter the Reference string :\n");

for(i=0;i<n;i++)

scanf("%d",&a[i]);

for(i=0;i<fno;i++)

frame[i]= -1;

    j=0;

printf("\n FIFO Page Replacement Algorithm\n\n The given reference string is:\n\n");

for(i=0;i<n;i++)

    {

printf(" %d ",a[i]);

    }

printf("\n");

for(i=0;i<n;i++)

    {

printf("\nReference No %d-> ",a[i]);

avail=0;

for(k=0;k<fno;k++)

if(frame[k]==a[i])

avail=1;

if (avail==0)

    {

frame[j]=a[i];

        j = (j+1) % fno;

pagefault++;

for(k=0;k<fno;k++)
```
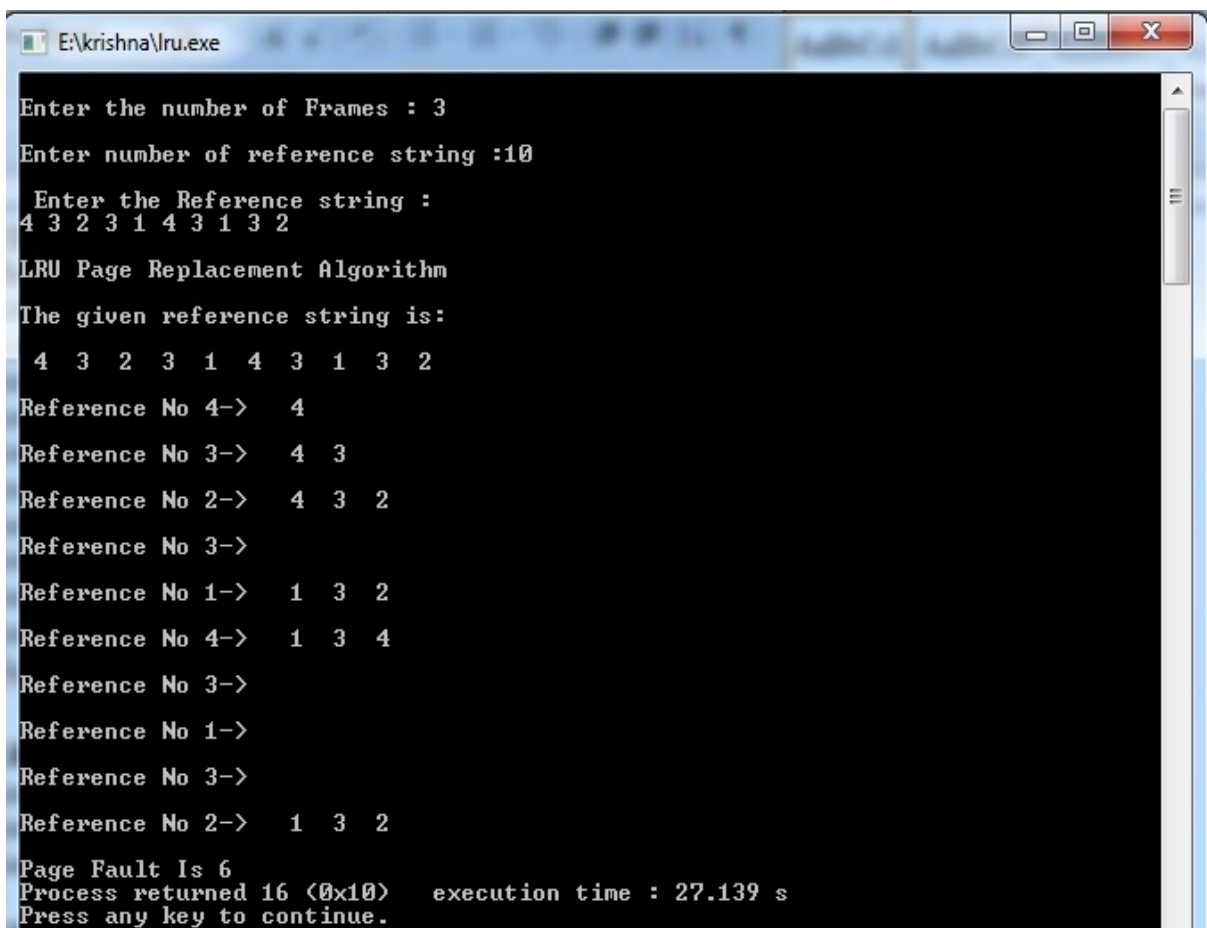
```
if(frame[k]!=-1)

printf(" %2d",frame[k]);

    }

printf("\n");

  }

printf("\nPage Fault Is %d",pagefault);

  }
```

Output:

b) LRU Algorithm:

Step1: Start

Step2: Read the number of frames

Step3: Read the number of pages

Step4: Read the page numbers

Step5: Initialize the values in frames to -1

Step6: Allocate the pages in to frames by selecting the page that has not been used for the longest period of time.

Step7: Display the number of page faults.

Step8: Stop

SOURCE CODE:

```
#include<stdio.h>

void main()

{

   int i,j,l,max,n,a[50],frame[10],flag,fno,k,avail,pagefault=0,lru[10];

printf("\nEnter the number of Frames : ");

scanf("%d",&fno);

printf("\nEnter number of reference string :");

scanf("%d",&n);

printf("\n Enter the Reference string :\n");

for(i=0;i<n;i++)

scanf("%d",&a[i]);

for(i=0;i<fno;i++)

   {

frame[i]= -1;

lru[i] = 0;

   }

printf("\nLRU Page Replacement Algorithm\n\nThe given reference string is:\n\n");

for(i=0;i<n;i++)
```

```c
    {
printf(" %d ",a[i]);
    }
printf("\n");
    j=0;
for(i=0;i<n;i++)
    {
max = 0;
flag=0;
printf("\nReference No %d-> ",a[i]);
avail=0;
for(k=0;k<fno;k++)
if(frame[k]==a[i])
        {
avail=1;
lru[k]=0;
break;
        }
if(avail==1)
    {
for(k=0;k<fno;k++)
if(frame[k]!=-1)
        ++lru[k];
max = 0;
for(k=1;k<fno;k++)
if(lru[k]>lru[max])
```

```
max = k;

        j = max;

    }
if(avail==0)

    {
lru[j]=0;

frame[j]=a[i];

for(k=0;k<fno;k++)

        {
if(frame[k]!=-1)

            ++lru[k];

else

        {
            j = k;
flag = 1;

break;

        }
    }
if(flag==0){

max = 0;

for(k=1;k<fno;k++)

if(lru[k]>lru[max])

max = k;

        j = max;

    }
pagefault++;
```

```
for(k=0;k<fno;k++)

if(frame[k]!=-1)

printf(" %2d",frame[k]);

    }

printf("\n");

  }

printf("\nPage Fault Is %d",pagefault);

}
```

OUTPUT :

c) Optimal Algorithm:

> **Step1:** Start
>
> **Step2:** Read the number of frames
>
> **Step3:** Read the number of pages

Step4: Read the page numbers

Step5: Initialize the values in frames to -1

Step6: Allocate the pages in to frames by selecting the page that will not be used for the longest period of time.

Step7: Display the number of page faults.

Step8: Stop

SOURCE CODE :

```
#include<stdio.h>

int main()

{

    int i,j,l,min,flag1,n,a[50],temp,frame[10],flag,fno,k,avail,pagefault=0,opt[10];

printf("\nEnter the number of Frames : ");

scanf("%d",&fno);

printf("\nEnter number of reference string :");

scanf("%d",&n);

printf("\n Enter the Reference string :\n");

for(i=0;i<n;i++)

scanf("%d",&a[i]);

for(i=0;i<fno;i++)

    {

frame[i]= -1;

opt[i]=0;

    }

printf("\nLFU Page Replacement Algorithm\n\nThe given reference string is:\n\n");

for(i=0;i<n;i++)
```

```
printf(" %d ",a[i]);

printf("\n");

   j=0;

for(i=0;i<n;i++)

   {

flag=0;

     flag1=0;

printf("\nReference No %d-> ",a[i]);

avail=0;

for(k=0;k<fno;k++)

if(frame[k]==a[i])

         {

avail=1;

break;

         }

if(avail==0)

     {

temp = frame[j];

frame[j]=a[i];

for(k=0;k<fno;k++)

       {

if(frame[k]==-1)

          {

              j = k;

flag = 1;

break;
```

```
            }

          }

if(flag==0)

        {

for(k=0;k<fno;k++)

          {

opt[k]=0;

for(l=i;l<n;l++)

            {

if(frame[k]==a[l])

               {

                  flag1 = 1;

break;

               }

            }

if(flag1==1)

opt[k] = l-i;

else

          {

opt[k] = -1;

break;

          }

        }

min = 0;

for(k=0;k<fno;k++)

if(opt[k]<opt[min]&&opt[k]!=-1)
```

```
min = k;

else if(opt[k]==-1)

            {

min = k;

frame[j] = temp;

frame[k] = a[i];

break;

            }

        j = min;

    }

pagefault++;

for(k=0;k<fno;k++)

if(frame[k]!=-1)

printf(" %2d",frame[k]);

    }

printf("\n");

  }

printf("\nPage Fault Is %d",pagefault);

return 0;

}
```
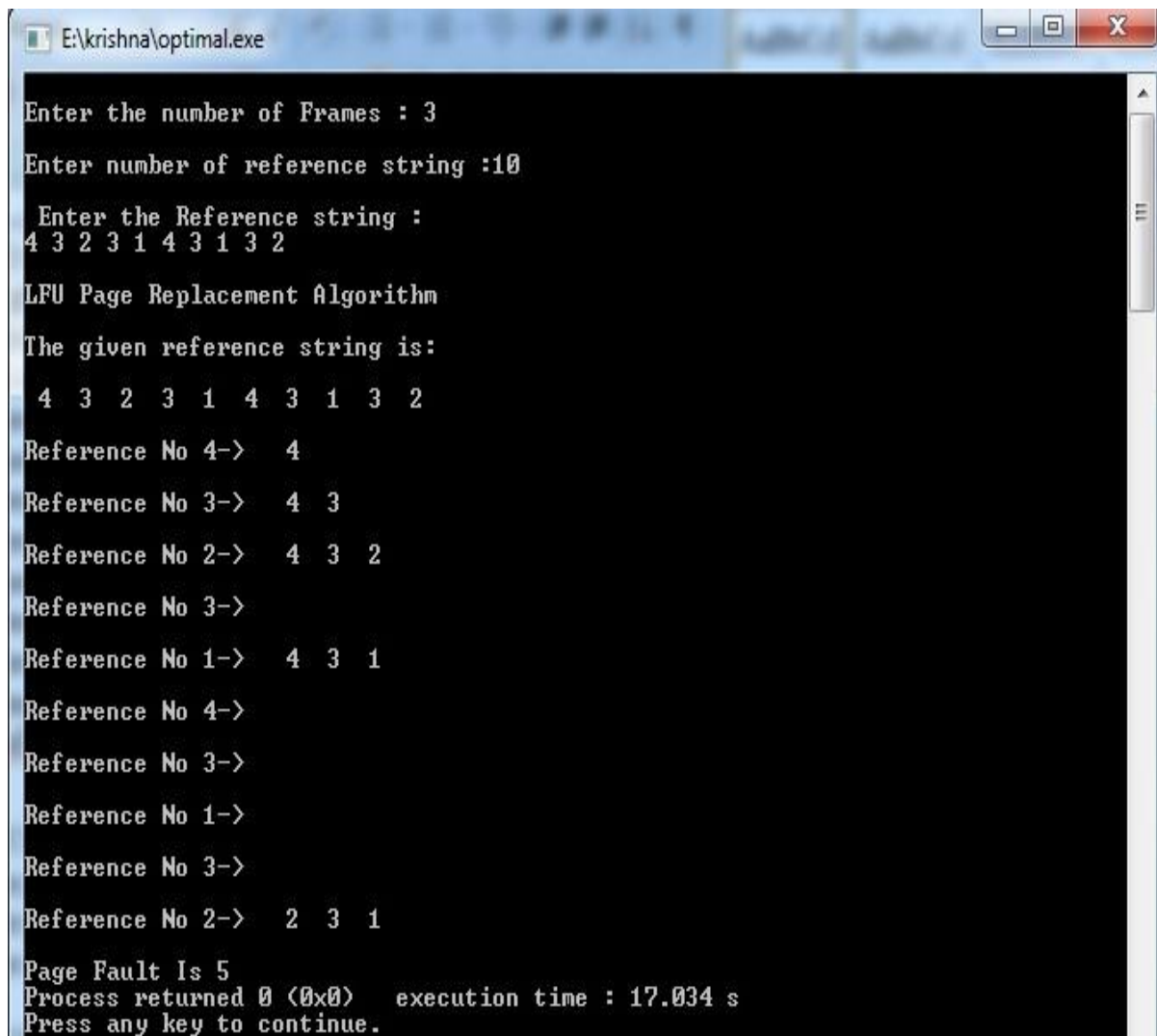
Output:

```
E:\krishna\optimal.exe

Enter the number of Frames : 3

Enter number of reference string :10

 Enter the Reference string :
4 3 2 3 1 4 3 1 3 2

LFU Page Replacement Algorithm

The given reference string is:

 4  3  2  3  1  4  3  1  3  2

Reference No 4->   4

Reference No 3->   4  3

Reference No 2->   4  3  2

Reference No 3->

Reference No 1->   4  3  1

Reference No 4->

Reference No 3->

Reference No 1->

Reference No 3->

Reference No 2->   2  3  1

Page Fault Is 5
Process returned 0 (0x0)    execution time : 17.034 s
Press any key to continue.
```

VIVA-VOCE

1.Why do we use page replacement algorithms?

Page replacement algorithms are an important part of virtual memory management and it helps the OS to decide which memory page can be moved out, making space for the currently needed page.However, the ultimate objective of all page replacement algorithms is to reduce the number of page faults

2.Which is best page replacement algorithm and why?

LRU resulted to be the best algorithm for page replacement to implement. LRU maintains a linked list of all pages in the memory, in which, the most recently used page is placed at the front, and the least recently used page is placed at the rear.

3.Explain LRU algorithm.

LRU stands for **Least Recently Used.**LRU replaces the line in the cache that has been in the cache the longest with no reference to it. It works on the idea that the more recently used blocks are more likely to be referenced again.

4. When does a page fault occur?

Page fault occurs when a requested page is mapped in virtual address space but not present in memory.

5.What are page replacement algorithms in OS?

1. First In First Out (FIFO)

2. Optimal Page replacement

3. Least Recently Used

4. When does a page fault occur?