# COMPUTER NETWORKS AND OPERATING SYSTEMS [R20A0567]

# LAB MANUAL

**B.TECH II YEAR – II SEM (R20)**

**(2021-22)**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (DATA SCIENCE,CYBER SECURITY,INTERNET OF THINGS)**

**MALLA REDDY COLLEGE OF ENGINEERING &TECHNOLOGY**
**(Autonomous Institution – UGC, Govt. of India)**

Recognized under 2(f) and 12 (B) of UGC ACT 1956
(Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2015 Certified)
Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, India

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (DATA SCIENCE,CYBER SECURITY,INTERNET OF THINGS)

### VISION

- To improve the quality of technical education that provides efficient softwareengineers with an attitude to adapt challenging IT needs of local, national and international arena, through teaching and interaction with alumni and industry.

### MISSION

- Department intends to meet the contemporary challenges in the field of IT and is playing a vital role in shaping the education of the 21st century by providing unique educational and research opportunities.

## PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

**PEO1 – ANALYTICAL SKILLS**

To facilitate the graduates with the ability to visualize, gather information, articulate, analyze, solve complex problems, and make decisions. These are essential to address the challenges of complex and computation intensive problems increasing their productivity.

**PEO2 – TECHNICAL SKILLS**

To facilitate the graduates with the technical skills that prepare them for immediate employment and pursue certification providing a deeper understanding of the technology in advanced areas of computer science and related fields, thus encouraging to pursue higher education and research based on their interest.

**PEO3 – SOFT SKILLS**

To facilitate the graduates with the soft skills that include fulfilling the mission, setting goals, showing self-confidence by communicating effectively, having a positive attitude, get involved in team- work, being a leader, managing their career and their life.

**PEO4 – PROFESSIONAL ETHICS**

To facilitate the graduates with the knowledge of professional and ethical responsibilities by paying attention to grooming, being conservative with style, following dress codes, safety codes, and adapting themselves to technological advancements.

## PROGRAM SPECIFIC OUTCOMES (PSOs)

After the completion of the course, B. Tech Information Technology, the graduates willhave the following Program Specific Outcomes:

1. **Fundamentals and critical knowledge of the Computer System:-** Able to Understand the working principles of the computer System and its components , Apply the knowledge to build, asses, and analyze the software and hardware aspects of it .

2. **The comprehensive and Applicative knowledge of Software Development:** Comprehensive skills of Programming Languages, Software process models, methodologies, and able to plan, develop, test, analyze, and manage the software and hardware intensive systems in heterogeneous platforms individually or working in teams.

3. **Applications of Computing Domain & Research:** Able to use the professional, managerial, interdisciplinary skill set, and domain specific tools in development processes, identify the research gaps, and provide innovative solutions to them.

## PROGRAMOUTCOMES (POs)

**Engineering Graduates should possess the following:**

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 multi-disciplinary 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.

## GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to starting time), thosewho come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab withthe synopsis / program / experiment details.
3. Student should enter into the laboratory with:

a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm,Procedure, Program, Expected Output, etc.,) filled in for the lab session.
b. Laboratory Record updated up to the last session experiments and other utensils (if any)needed in the lab.
c. Proper Dress code and Identity card.

4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer systemallotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observationnote book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which shouldbe utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out ; if anybody found loitering outside the lab / class without permission during working hours willbe treated seriously and punished appropriately.
10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

**Head of the Department** **Principal**

**II Year B. Tech CSE (CS & IOT) - I Sem** **L T/P/D C**

**- -/3/- 1.5**

## (R20A0567)COMPUTER NETWORKS AND OPERATING SYSTEM LAB

**OBJECTIVES:**

1. To provide an understanding of the design aspects of operating system concepts through simulation.
2. Introduce basic Linux commands, system call interface for process management, inter-process communication and I/O in Unix.
3. Student will learn various process and CPU scheduling Algorithms through simulation programs.
4. Analyze the different layers in networks.
5. To be familiar with contemporary issues in networking technologies, also to know that how the routing algorithms worked out in network layer.

**OPERATING SYSTEM**

**WEEK 1:**

Practice File handling utilities, Process utilities, Disk utilities, Networking commands, Filters, Text, Processing utilities and Backup utilities.

**WEEK 2:**

Simulate the following CPU scheduling algorithms. a)FCFS b) SJF c) Round Robin d) Priority

**WEEK 3:**

Simulate all page replacement algorithms a) FIFO b) LRU c) LFU

**WEEK 4:**

Write a program that illustrate communication between two process using unnamed pipes

**WEEK 5:**

Write a program that illustrates communication between two process using named pipes or FIFO.

**WEEK 6:**

Write a C program that receives a message from message queue and display them.

**COMPUTER NETWORKS**

**WEEK 7:**

Implement the data link layer framing methods such as character, character stuffing and bit stuffing.

**WEEK 8:**

Implement on a data set of characters the three CRC polynomials – CRC 12, CRC 16, CRC CCIP.

**WEEK 9:**

Implement Dijkstra’s algorithm to compute the shortest path through a graph.

**WEEK 10**

Take an example subnet graph with weights indicating delay between nodes. Now Obtain Routing table at each node using distance vector routing algorithm.

**WEEK 11:**

Take an example subnet of hosts. Obtain broadcast tree for it.

**REFERENCE BOOKS:**

1. Operating Systems – Internals and Design Principles, William Stallings, Fifth Edition–2005, Pearson Education/PHI
2. Operating System - A Design Approach-Crowley, TMH.
3. Modern Operating Systems, Andrew S Tanenbaum, 2nd edition, Pearson/PHI

4. UNIX Programming Environment, Kernighan and Pike, PHI/Pearson Education
5. UNIX Internals: The New Frontiers, U. Vahalia, Pearson Education.
6. Computer Networks - Andrew S Tanenbaum, 4th Edition, Pearson Education.
7. Data Communications and Networking - Behrouz A. Forouzan, Fifth Edition TMH, 2013

**COURSE OUTCOMES**:

1. Introduce basic Linux commands, system call interface for process management, inter process communication and I/O in Unix.
2. Develop various process and CPU scheduling Algorithms through simulation programs.
3. Student will have exposure to System calls and simulate them.
4. To apply knowledge of different techniques of error detection and correction to detect and solve error bit during data transmission.
5. Understand and building the skills of routing mechanisms.

**MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY**
**( UGC-Autonomous Institution , Govt. of India )**
(Permanently Affiliated to JNTUH, Approved by AICTE-Accredited by NBA & NAAC- A-Grade; ISO 9001:2008 Certified) Maisammaguda, Dhulapally Post, Via Hakimpet, Secunderabad – 500100

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**
**(DATA SCIENCE,CYBER SECURITY,INTERNET OF THINGS)**
**(R20A0567) COMPUTER NETWORKS AND OPERATING SYSTEM LAB MANUAL**

**TABLE OF CONTENTS**

## WEEK 1

**AIM :** Practice File handling utilities, Process utilities, Disk utilities, Networking commands, Filters, Text processing utilities and Backup utilities.

**FILE HANDLING UTILITIES**
**Cat Command:** cat linux command concatenates files and print it on the standard output.

***To Create a new file:***
cat > file1.txt
This command creates a new file file1.txt. After typing into the file press control+d(^d) simultaneously to end the file.

***To Append data into the file:*** To append data into the same file use append operator >> to write into thefile, else the file will be overwritten (i.e., all of its contents will be erased).
cat >> file1.txt

***To display a file:*** This command displays the data in the file.cat file1.txt

***To concatenate several files and display:***
cat file1.txt file2.txt

The above cat command will concatenate the two files (file1.txt and file2.txt) and it will display the output in the screen. Some times the output may not fit the monitor screen. In such situation you canprint those files in a new file or display the file using less command.

cat file1.txt file2.txt | less

***To concatenate several files and to transfer the output to anotherfile.***
cat file1.txt file2.txt > file3.txt

In the above example the output is redirected to new file file3.txt.

**rm COMMAND:**
rm linux command is used to remove/delete the file from the directory.

***To Remove / Delete a file:*** Here rm command will remove/delete the file file1.txt.rm file1.txt

***To delete a directory tree:***
rm -ir tmp

This rm command recursively removes the contents of all subdirectories of the tmp directory, prompting you regarding the removal of each file, and then removes the tmpdirectory itself.

***To remove more files at once:*** rm command removes file1.txt and file2.txt files at the same time. rm file1.txt file2.txt

**cd COMMAND:** cd command is used to change the directory.

**cd linux-command**
This command will take you to the sub-directory(linux-command) from its parent directory.
**Ex:**

**cd ..**
This will change to the parent-directory from the current working directory/sub-directory.

**cd ~**
This command will move to the user's home directory which is "/home/username".

**cp COMMAND:**
cp command copy files from one location to another. If the destination is an existing file, then the file is overwritten; if the destination is an existing directory, the file is copied into the directory (the directory is not overwritten).

***Copy two files:***
cp file1.txt file2.txt
The above cp command copies the content of file1.txt to file2.txt

**ls COMMAND:**
ls command lists the files and directories under current working directory. Display root directorycontents:

**ls /**
lists the contents of root directory.

***Display hidden files and directories:***
ls -a
lists all entries including hidden files and directories.

***Display inode information:***
ls –i

**ln COMMAND:**
ln command is used to create link to a file (or) directory. It helps to provide soft link for desired files.

***Inode will be different for source and destination.***
ln -s file1.txt file2.txt

Creates a symbolic link to 'file1.txt' with the name of 'file2.txt'. Here inode for 'file1.txt' and 'file2.txt'will be different.

**mkdir command:** Use this command to create one or more new directories.

Include one or more instances of the "**<DIRECTORY**" variable (separating each with a whitespace), and set each to the complete path to the new directory to be created.

mkdir OPTION <DIRECTORY>

rmdir command:

mv command:

diff command:

comm command:

wc command**:**

**PROCESS UTILITIES:**

**ps Command:**
ps command is used to report the process status. ps is the short name for Process Status.

**1.** **ps:** List the current running processes.
***Output:***
```
PID TTY TIME CMD
2540 pts/1 00:00:00 bash
```

**2.** **ps –f :** Displays full information about currently running processes.
***Output:***
```
UID             PID   PPID C STIME TTY TIME       CMD

nirmala         2540 2536  0 15:31     pts/1 00:00:00 bash
```

**3.** **kill COMMAND:** kill command is used to kill the background process.

***Step by Step process:***
- Open a process music player or any file.xmms

**press ctrl+z to stop the process.**
- To know group id or job id of the background task.jobs -l

It will list the background jobs with its job id as,
- xmms 3956
- kmail 3467

To kill a job or process.
- **kill 3956**

kill command kills or terminates the background process xmms.

**Disk utilities:**
**du (abbreviated from disk usage)** is a standard Unix program used to estimate file spaceusage—space used under a particular directory or files on a file system.

**$du kt.txt pt.txt** /* the first column displayed the file's disk usage */
```
8          kt.txt
4          pt.txt
```

**Using -h option:** As mentioned above, -h option is used to produce the output in humanreadable format.

**$du -h kt.txt pt.txt**
```
8.0K          kt.txt4.0K                                         pt.txt
```
/*now the output is in human readable format i.e in Kilobytes */

Using -a option
**$du -a kartik**
```
8          kartik/kt.txt                          4          kartik/thakral.png
4          kartik/pt.txt                          4          kartik/thakral
4                                                 24          kartik
           kartik/pranjal.
png
```

/*so with -a option used all the files (under directory kartik) disk usage info is displayed alongwith the thakral sub-directory */

**df command :** Report file system disk space usage

**$df kt.txt**

```
Filesystem               1K-blocks      Used Available Use% Mounted on
/dev/the2                 1957124      1512   1955612   1%     /snap/core
```

/* the df only showed the disk usage details of the file system that contains file kt.txt */

***//using df without any filename //***
**$df**
/* in this case df displayed the disk usage details of all mounted file systems */

**Using -h :** This is used to make df command display the output in human-readable format.

//using -h with df//
**$df -h kt.txt**

```
Filesystem               1K-blocks      Used Available Use% Mounted on
/dev/the2                    1.9G      1.5M         1.9G            1%  /snap/core
```

/*this output is easily understandable by the user and all cause of -h option */

**NETWORKING COMMANDS**

**ping**
The ping command sends an echo request to a host available on the network. Using thiscommand, you can check if your remote host is responding well or not.
**Syntax: $ping hostname or ip-address**

The above command starts printing a response after every second. To come out of thecommand, you can terminate it by pressing CNTRL + C keys.

***$ping google.com***

```
PING google.com (74.125.67.100) 56(84) bytes of data.
64 bytes from 74.125.67.100: icmp_seq=1 ttl=54 time=39.4 ms
```

**ftp:** ftp stands for File Transfer Protocol. This utility helps you upload and download your filefrom one computer to another computer.
**Syntax $ftp hostname or ip-address**

***$ftp amrood.com***

```
Connected to amrood.com.
220 amrood.com FTP server (Ver 4.9 Thu Sep 2 20:35:07 CDT 2009) Name
(amrood.com:amrood): amrood
331 Password required for amrood.Password:
230 User amrood logged in.ftp> dir
200 PORT command successful.
….
ftp> quit
221 Goodbye.
```

**telnet:**
Telnet is a utility that allows a computer user at one site to make a connection, login and then conduct work on a computer at another site. Once you login using Telnet, you can perform all the activities on your remotely connected machine.

```
C:>telnet amrood.comTrying...
Connected to amrood.com.Escape character is '^]'. login: amrood
amrood's Password:
```

```
****************************************************** WELCOME TO AMROOD.COM
*
******************************************************
$ logoutLINUX  PROGRAMMING LAB021-2022

Connection closed.C:>
```

**Finger**:
The finger command displays information about users on a given host. The host can be eitherlocal or remote.
***Check all the logged-in users on the local machine −***

```
$ finger
Login                    Name   Tty      Idle    Login  Time     Office
amrood                   pts/0           Jun 25   08:03           (62.61.164.115)
```

***Check all the logged-in users on the remote machine –***

```
$ finger @avtar.com
Login Name Tty Idle Login Time Office amrood pts/0 Jun 25 08:03 (62.61.164.115)
```

***Get the information about a specific user available on the remote machine −***

```
$ finger amrood@avtar.com
```

**Ifconfig:** Ifconfig is used to configure the network interfaces.

**FILTERS**
**more COMMAND:**
more command is used to display text in the terminal screen. It allows only backwardmovement.

1. ***more -c index.txt***
   Clears the screen before printing the file .
2. ***more -3 index.txt***
   Prints first three lines of the given file. Press Enter to display the file line by line.

**head COMMAND:**
head command is used to display the first ten lines of a file, and also specifies how many linesto display.

1. **head index.php**
   This command prints the first 10 lines of 'index.php'.
2. **head -5 index.php**
   The head command displays the first 5 lines of 'index.php'.
3. **head -c 5 index.php**
   The above command displays the first 5 characters of 'index.php'.

**tail COMMAND:**
tail command is used to display the last or bottom part of the file. By default it displays last 10 lines of a file.

1. **tail index.php**
   It displays the last 10 lines of 'index.php'.
2. **tail -2 index.php**
   It displays the last 2 lines of 'index.php'.
3. **tail -n 5 index.php**

   It displays the last 5 lines of 'index.php'.
4. **tail -c 5 index.php**
   It displays the last 5 characters of 'index.php'.

**cut COMMAND:**
cut command is used to cut out selected fields of each line of a file. The cut command uses delimiters to determine where to split fields.

**cut -c1-3 text.txt**
***Output:***
Thi
Cut the first three letters from the above line.

**paste COMMAND:**
paste command is used to paste the content from one file to another file. It is also used to set column format for each line.

**paste test.txt>test1.txt**
Paste the content from 'test.txt' file to 'test1.txt' file.

**sort COMMAND:**
sort command is used to sort the lines in a text file.

1. ***sort test.txt***
   Sorts the 'test.txt'file and prints result in the screen.
2. ***sort -r test.txt***
   Sorts the 'test.txt' file in reverse order and prints result in the screen.

**uniq**
Report or filter out repeated lines in a file.

uniq myfile1.txt > myfile2.txt - Removes duplicate lines in the first file1.txt and outputs theresults to the second file.

**TEXT PROCESSING UTILITIES**
**echo:** display a line of text or echo command prints the given input string to standard output. eg.
echo I love India
echo $HOME

**wc:** print the number of newlines, words, and bytes in fileseg. wc file1.txt

**nl:** which lets you number lines in files.
**eg. $ nl file11 hi**
**join-** Join command is used for merging the lines of different sorted files based on the presence of common field into a single line. The second line will be appended at the end ofthe first line and cursor is placed at the end of line after joining.

**Grep <u>(Global Regular Expression Searching for a pattern), fgrep and egrep</u>**
$ grep –sales director‖ emp1 emp2

$fgrep _good bad great' userfile
$egrep _good | bad | great' userfile

**cat, head, tail, sort, uniq, cut, paste and etc.**

**BACKUP UTILITIES**
Linux backup and restore can be done using backup commands tar, cpio, dump and restore.

**<u>Backup Restore using tar command</u>**

**tar: tape archive** is used for single or multiple files backup and restore on/from a tape or file.
**$tar cvf /dev/rmt/0 ***

Options: c -> create ; v -> Verbose ; f->file or archive device ; * -> all files and directories .

**$tar cvf /home/backup ***
Create a tar called backup in home directory, from all file and directories s in the currentdirectory.

**Viewing a tar backup on a tape or file**
$tar tvf /dev/rmt/0 ## view files backed up on a tape device.
$tar tvf /home/backup ## view files backed up inside the backup
**Note:** t option is used to see the table of content in a tar file.

**Extracting tar backup from the tape**
$tar xvf /home/backup ## extract / restore files in to current directory.
**Note :** x option is used to extract the files from tar file. Restoration will go to present directoryor original backup path depending on relative or absolute path names used for backup.

## Backup restore using cpio command

**Using cpio command to backup all the files in current directory to tape.**
find . -depth -print | cpio -ovcB > /dev/rmt/0

cpio expects a list of files and find command provides the list, cpio has to put these file onsome destination and a > sign redirect these files to tape. This can be a file as well .

**Viewing cpio files on a tape**

cpio -ivtB < /dev/rmt/0
## Options i -> input ; v->verbose; t-table of content; B-> set I/O block size to 5120 bytes

**Restoring a cpio backup**
cpio -ivcB < /dev/rmt/0
## Options i -> input ; v->verbose; t-table of content; B-> set I/O block size to 5120 bytes

## WEEK 2

**AIM :** To write a C program to simulate the following non-preemptive CPU scheduling algorithms to find turnaround time and waiting time for the following.
a) FCFS b) SJF c) Round Robin d) Priority

**DESCRIPTION**
Assume all the processes arrive at the same time.

***FCFS CPU SCHEDULING ALGORITHM***
For FCFS scheduling algorithm, read the number of processes/jobs in the system, their CPU burst times. The scheduling is performed on the basis of arrival time of the processes irrespective of their other parameters. Each process will be executed according to its arrival time. Calculate the waiting time and turnaround time of each of the processes accordingly.

***SJF CPU SCHEDULING ALGORITHM***
For SJF scheduling algorithm, read the number of processes/jobs in the system, their CPU burst times. Arrange all the jobs in order with respect to their burst times. There may be two jobs in queue with the same execution time, and then FCFS approach is to be performed. Each process will be executed according to the length of its burst time. Then calculate the waiting time and turnaround time of each of the processes accordingly.

***ROUND ROBIN CPU SCHEDULING ALGORITHM***
For round robin scheduling algorithm, read the number of processes/jobs in the system, their CPU burst times, and the size of the time slice. Time slices are assigned to each process in equal portions and in circular order, handling all processes execution. This allows every process to get an equal chance. Calculate the waiting time and turnaround time of each of the processes accordingly.

***PRIORITY CPU SCHEDULING ALGORITHM***
For priority scheduling algorithm, read the number of processes/jobs in the system, their CPU burst times, and the priorities. Arrange all the jobs in order with respect to their priorities. There may be two jobs in queue with the same priority, and then FCFS approach is to be performed. Each process will be executed according to its priority. Calculate the waiting time and turnaround time of each of the processes accordingly.

**PROGRAM**

***a) FCFS CPU SCHEDULING ALGORITHM***
```
#include<stdio.h>
#include<conio.h>
main()
{
int bt[20], wt[20], tat[20], i, n;
float wtavg, tatavg;
clrscr();
printf("\nEnter the number of processes -- ");
scanf("%d", &n);
for(i=0;i<n;i++)
{
printf("\nEnter Burst Time for Process %d -- ", i);
scanf("%d", &bt[i]);
}
wt[0] = wtavg = 0;
tat[0] = tatavg = bt[0];
for(i=1;i<n;i++)
{
wt[i] = wt[i-1] +bt[i-1];
tat[i] = tat[i-1] +bt[i];
wtavg = wtavg + wt[i];
tatavg = tatavg + tat[i];
}
printf("\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");
for(i=0;i<n;i++)
printf("\n\t P%d \t\t %d \t\t %d \t\t %d", i, bt[i], wt[i], tat[i]);
printf("\nAverage Waiting Time -- %f", wtavg/n);
printf("\nAverage Turnaround Time -- %f", tatavg/n);
getch();
}
```

***INPUT***

```
Enter the number of processes --              3
Enter Burst Time for Process 0 --            24
Enter Burst Time for Process 1 --             3
Enter Burst Time for Process 2 --             3
```

***OUTPUT***

```
PROCESS          BURST TIME        WAITING TIME          TURNAROUND
                                                               TIME
            P0           24                   0                  24
            P1            3                  24                  27
            P2            3                  27                  30

Average Waiting Time--          17.000000
Average Turnaround Time --              27.000000
```

***b) SJF CPU SCHEDULING ALGORITHM***

```
#include<stdio.h>
#include<conio.h>
main()
{
int p[20], bt[20], wt[20], tat[20], i, k, n, temp;
float wtavg, tatavg;
clrscr();
printf("\nEnter the number of processes -- ");
scanf("%d", &n);
for(i=0;i<n;i++)
{
p[i]=i;
printf("Enter Burst Time for Process %d -- ", i);
scanf("%d", &bt[i]);

}
for(i=0;i<n;i++)
for(k=i+1;k<n;k++)
if(bt[i]>bt[k])
{
temp=bt[i]; bt[i]=bt[k]; bt[k]=temp;
}
np=p[i];p[i]=p[k]; p[k]=temp;
wt[0]=wtavg=0;
}
   tat[0] = tatavg = bt[0];
   for(i=1;i<n;i++)
{
wt[i] = wt[i-1] +bt[i-1];
tat[i] = tat[i-1] +bt[i];
wtavg = wtavg + wt[i];
tatavg = tatavg + tat[i];
}
printf("\n\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");
for(i=0;i<n;i++)

        printf("\n\t P%d \t\t %d \t\t %d \t\t %d", p[i], bt[i], wt[i], tat[i]);printf("\nAverage Waiting Time -
- %f", wtavg/n);
printf("\nAverage Turnaround Time -- %f", tatavg/n);getch();
}
```

***INPUT***

```
   Enter the number of processes --                    4
  Enter Burst Time for Process  0 --                   6
  Enter Burst Time for Process  1 --                   8
  Enter Burst Time for Process  2 --                   7
  Enter Burst Time for Process  3 --                   3
```

***OUTPUT***

```
 PROCESS            BURST TIME        WAITING TIME        TURNAROUND
                                                                TIME
               P3              3                  0                3
               P0              6                  3                9
               P2              7                  9               16
               P1              8                 16               24
Average Waiting Time --          7.000000
Average Turnaround Time --       13.000000
```

***C)ROUND ROBIN CPU SCHEDULING ALGORITHM***

```
#include<stdio.h>
main()
{
int  i,j,n,bu[10],wa[10],tat[10],t,ct[10],max; float  awt=0,att=0,temp=0;
clrscr();
printf("Enter the no of processes -- ");scanf("%d",&n);

for(i=0;i<n;i++)
{
printf("\nEnter Burst Time for process %d -- ", i+1);
scanf("%d",&bu[i]);
ct[i]=bu[i];
}
printf("\nEnter the size of time slice -- ");scanf("%d",&t);
max=bu[0]; for(i=1;i<n;i++)
if(max<bu[i])
             max=bu[i]; for(j=0;j<(max/t)+1;j++)
for(i=0;i<n;i++)
if(bu[i]!=0)
if(bu[i]<=t)
{
tat[i]=temp+bu[i];
temp=temp+bu[i];
bu[i]=0;
}
else
{
bu[i]=bu[i]-t; temp=temp+t;

}

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

wa[i]=tat[i]-ct[i];att+=tat[i];
awt+=wa[i];
}
printf("\nThe Average Turnaround time is -- %f",att/n); printf("\nThe Average Waiting time is -- %f
",awt/n);
printf("\n\tPROCESS\t BURST TIME \t WAITING TIME\tTURNAROUND TIME\n");
for(i=0;i<n;i++)
printf("\t%d \t %d \t\t %d \t\t %d \n",i+1,ct[i],wa[i],tat[i]);
getch();
}
```

***INPUT***

```
Enter the no of processes – 3
Enter Burst Time for process 1 –        24
Enter Burst Time for process 2 --       3
Enter Burst Time for process 3 --       3

Enter the size of time slice – 3
    PROCESS          BURST TIME     AITING TIME        TURNAROUND TIME
        1                24              6                     30
        2                 3              4                      7
        3                 3              7                     10
```

***OUTPUT***

```
The Average Turnaround time is – 15.666667
The Average Waiting time is --          5.666667
```

***d)PRIORITY CPU SCHEDULING ALGORITHM***

```
#include<stdio.h>main()
{
int p[20],bt[20],pri[20], wt[20],tat[20],i, k, n, temp; float wtavg, tatavg;
clrscr();
printf("Enter the number of processes---- ");scanf("%d",&n);

for(i=0;i<n;i++)
{
p[i] = i;
printf("Enter the Burst Time & Priority of Process %d --- ",i);scanf("%d %d",&bt[i], &pri[i]);
}
for(i=0;i<n;i++)
for(k=i+1;k<n;k++)
if(pri[i] > pri[k])
{
temp=p[i];p[i]=p[k]; p[k]=temp;

temp=bt[i]; bt[i]=bt[k]; bt[k]=temp;

temp=pri[i]; pri[i]=pri[k];pri[k]=temp;


}
wtavg = wt[0] = 0;
tatavg = tat[0] = bt[0];
for(i=1;i<n;i++)
{
wt[i] = wt[i-1] + bt[i-1];
tat[i] = tat[i-1] + bt[i];


wtavg = wtavg + wt[i]; tatavg = tatavg + tat[i];
}


printf("\nPROCESS\t\tPRIORITY\tBURST TIME\tWAITING TIME\tTURNAROUND  TIME");
for(i=0;i<n;i++)
printf("\n%d \t\t %d \t\t  %d \t\t %d \t\t %d ",p[i],pri[i],bt[i],wt[i],tat[i]);


printf("\nAverage Waiting Time is --- %f",wtavg/n); printf("\nAverage Turnaround Time is ---
%f",tatavg/n);getch();
}
```

***INPUT***

```
                  Enter the number of processes -- 5
  Enter the Burst Time & Priority of Process 0 ---  10
   Enter the Burst Time & Priority of Process 1 ---  1
   Enter the Burst Time & Priority of Process 2 ---  2
   Enter the Burst Time & Priority of Process 3 ---  1
   Enter the Burst Time & Priority of Process 4 ---  5
```

***UTPUT***

```
ROCESS      RIORITY         BURST TIME          AITING TIME     URNAROUND TIME
   1           1                1                   0                 1
   4           2                5                   1                 6
   0           3                10                  6                 16
   2           4                2                   16                18
   3           5                1                   18                19
Average Waiting Time is --- 8.200000 Average Turnaround Time is --- 12.000000
```

## WEEK 3

**AIM:** To Simulate all page replacement algorithms a) FIFO b) LRU c) OPTIMAL

### DESCRIPTION

Page replacement is basic to demand paging. It completes the separation between logical memory and physical memory. With this mechanism, an enormous virtual memory can be provided for programmers on a smaller physical memory. There are many different page-replacement algorithms. Every operating system probably has its own replacement scheme. A FIFO replacement algorithm associates with each page the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen. If the recent past is used as an approximation of the near future, then the page that has not been used for the longest period of time can be replaced. This approach is the Least Recently Used (LRU) algorithm. LRU replacement associates with each page the time of that page's last use. When a page must be replaced, LRU chooses the page that has not been used for the longest period of time. Least frequently used (LFU) page-replacement algorithm requires that the page with the smallest count be replaced. The reason for this selection is that an actively used page should have a large reference count.

### PROGRAM

### FIFO PAGE REPLACEMENT ALGORITHM

```
#include<stdio.h> #include<conio.h>main()
{
int i, j, k, f, pf=0, count=0, rs[25], m[10], n;clrscr();
printf("\n Enter the length of reference string -- ");
scanf("%d",&n);
printf("\n Enter the reference string -- ");
for(i=0;i<n;i++)
        scanf("%d",&rs[i]);
        printf("\n Enter no. of frames -- ");
        scanf("%d",&f);
for(i=0;i<f;i++)
m[i]=-1;

printf("\n The Page Replacement Process is -- \n");
for(i=0;i<n;i++)
{
for(k=0;k<f;k++)
{
                                if(m[k]==rs[i])
                                break;
}
if(k==f)
{
                                m[count++]=rs[i];
                                pf++;
}
for(j=0;j<f;j++)
printf("\t%d",m[j]);
if(k==f)
printf("\tPF No. %d",pf);
printf("\n");
if(count==f)
count=0;
}
printf("\n The number of Page Faults using FIFO are %d",pf);
getch();
}
```

***INPUT***

```
Enter the length of reference string – 20
Enter the reference string --  7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Enter no. of frames --          3
```

***OUTPUT***

```
The Page Replacement Process is –
                    -1      -1          PF No. 1
                     0      -1          PF No. 2
                     0       1          PF No. 3
                     0       1          PF No. 4
                     0       1
                     3       1          PF No. 5
                     3       0          PF No. 6
                     3       0          PF No. 7
                     2       0          PF No. 8
                     2       3          PF No. 9
                     2       3          PF No. 10
                     2       3
                     2       3
                     1       3          PF No. 11
                     1       2          PF No. 12
                     1       2
                     1       2
                     1       2          PF No. 13
                     0       2          PF No. 14
                     0       1          PF No. 15

The number of Page Faults using FIFO are 15
```

**LRU PAGE REPLACEMENT ALGORITHM**

```
#include<stdio.h>
#include<conio.h>
#define high 37
void main()
{
int fframe[10],used[10],index;
int count,n1,k,nf,np=0,page[high],tmp;
int flag=0,pf=0;
clrscr();
printf("Enter no. of frames:");
scanf("%d",&nf);
for(i=0;count<nf;count++)
fframe[count]=-1;
printf(" lru page replacement algorithm in c ");
printf("Enter pages (press -999 to exit):\n");
for(count=0;count<high;count++)
{
scanf("%d",&tmp);
if(tmp==-999) break;
page[count]=tmp;
np++;
}
for(count=0;count<np;count++)
{
flag=0;
for(n1=0;n1<nf;n1++)
{
if(fframe[n1]==page[count])
{
printf("\n\t");
flag=1;break;
}
}
  /* program for lru page replacement algorithm in c */
  if(flag==0)
  {
  for(n1=0;n1<nf;n1++) used[n1]=0;
  for(n1=0,tmp=count-1;n1<nf-1;n1++,tmp--)
  {
  for(k=0;k<nf;k++)
  {
  if(fframe[k]==page[tmp])
  used[k]=1;
  }
  }
  for(n1=0;n1<nf;n1++)
  if(used[n1]==0)
  index=n1;
  fframe[index]=page[count];
  printf("\nFault: ");
  pf++;
  }
  for(k=0;k<nf;k++)
  printf("%d\t",fframe[k]);
  } // lru algorithm in c
  printf("\nnumber of total page faults is: %d ",pf);
  getch();
  }
```

```
lru page replacement algorithm in c
Enter no. of frames Enter pages (press -999 to exit):
2 3 2 1
5 2 4 5
3 2 5 -999
-1 -1 2 Fault:
-1 -1 2 Fault:
-1 3 2 Fault:
```

```
1 3 2 Fault:
1 3 2
1 5 2 Fault:
4 5 2 Fault:
4 5 2
4 5 3 Fault:
2 5 3 Fault:
2 5 3
2 5 3
```

## OPTIMAL PAGE REPLACEMENT ALGORITHM

### DESCRIPTION

Optimal page replacement algorithm has the lowest page-fault rate of all algorithms and will never suffer from Belady's anomaly. The basic idea is to replace the page that will not be used for the longest period of time. Use of this page-replacement algorithm guarantees the lowest possible page fault rate for a fixed number of frames. Unfortunately, the optimal page-replacement algorithm is difficult to implement, because it requires future knowledge of the reference string.

### PROGRAM

```
#include<stdio.h>int n;
main()
{
int seq[30],fr[5],pos[5],find,flag,max,i,j,m,k,t,s;int count=1,pf=0,p=0;
float pfr;clrscr();
printf("Enter maximum limit of the sequence: ");scanf("%d",&max);
printf("\nEnter the sequence: ");for(i=0;i<max;i++)
scanf("%d",&seq[i]); printf("\nEnter no. of frames: ");scanf("%d",&n);
fr[0]=seq[0];pf++;
printf("%d\t",fr[0]);i=1;
while(count<n)
{
flag=1;p++;
for(j=0;j<i;j++)
{
if(seq[i]==seq[j]) flag=0;
}
if(flag!=0)
{
                                        fr[count]=seq[i]; printf("%d\t",fr[count]);count++;
                                        pf++;


} i++;
}
```

```
printf("\n"); for(i=p;i<max;i++)
{
flag=1; for(j=0;j<n;j++)
{
if(seq[i]==fr[j])
flag=0;
}
if(flag!=0)
{
for(j=0;j<n;j++)
{
m=fr[j]; for(k=i;k<max;k++)
{
if(seq[k]==m)
{
                                        pos[j]=k;break;

}
else                                    pos[j]=1;

}
}
for(k=0;k<n;k++)
{
if(pos[k]==1)
flag=0;
}
if(flag!=0)
     s=findmax(pos);if(flag==0)
{
for(k=0;k<n;k++)
{
if(pos[k]==1)
{
s=k; break;
}
}
}
fr[s]=seq[i]; for(k=0;k<n;k++)
printf("%d\t",fr[k]);
pf++; printf("\n");
}
}
pfr=(float)pf/(float)max;
printf("\nThe no. of page faults are %d",pf);printf("\nPage fault rate %f",pfr);
getch();
}
int findmax(int a[])
{
int max,i,k=0;
max=a[0];
for(i=0;i<n;i++)
{
if(max<a[i])
{
```

```
                                    max=a[i];
                                     k=i;
                      }
                      }
return k;
}
}
```

**INPUT**

```
Enter number of page references -- 10
Enter the reference string --          1 2 3 4 5 2 5 2 5 1 4 3
Enter the available no. of frames ---3
```

***OUTPUT***

```
The Page Replacement Process is –

        1    -1     -1       PF No. 1
        1    2      -1       PF No. 2
        1    2      3        PF No. 3
        4    2      3        PF No. 4
        5    2      3        PF No. 5
        5    2      3
        5    2      3
        5    2      1        PF No. 6
        5    2      4        PF No. 7
        5    2      3        PF No. 8

Total number of page faults --     8
```

## WEEK 4

**AIM:-** To write a C program that illustrate communication between two process using unnamed pipes

**Program file name: unnamed_pipe.c**

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<string.h>
#include<fcntl.h>
void server(int,int);
void  client(int,int);
int main()
{
int p1[2],p2[2],pid;
pipe(p1);
pipe(p2);
pid=fork();
if(pid==0)
{
close(p1[1]);
close(p2[0]);
server(p1[0],p2[1]);
return 0;
}
close(p1[0]);
close(p2[1]);
client(p1[1],p2[0]);
wait();
return 0;
}
void client(int wfd,int rfd)
{
int i,j,n;
char fname[2000];
char buff[2000];
printf("ENTER THE FILE NAME :");
scanf("%s",fname);
printf("CLIENT SENDING THE REQUEST............................PLEASE WAIT\n");
sleep(10);
write(wfd,fname,2000);
n=read(rfd,buff,2000);
buff[n]='\0';
printf("THE RESULTS OF CLIENTS ARE............................\n");
write(1,buff,n);
}

void server(int rfd,int wfd)
{
int i,j,n;
char fname[2000];char buff[2000];
n=read(rfd,fname,2000);fname[n]='\0';
int fd=open(fname,O_RDONLY);sleep(10);
if(fd<0)
write(wfd,"can't open",9);
else
n=read(fd,buff,2000);
write(wfd,buff,n);
}
```

## WEEK 5

**AIM :**a) To Write a program that illustrates communication between two process using named pipes or FIFO.

**Algorithm:**

Create two processes, one is fifoserver_two way and another one is fifoclient_twoway.

**Algorithm for fifoserver_twoway :**

```
step 1:Start
step 2: Creates a named pipe (using library function mkfifo()) with name ―fifo_twoway‖ in /tmp directory, if not created.
step 3: Opens the named pipe for read and write purposes.
step 4: Here, created FIFO with permissions of read and write for Owner. Read for Group and nopermissions for Others.
step 5: Waits infinitely for a message from the client.
step 6: If the message received from the client is not ―end‖, prints the message and reverses the string. The reversed string is sent back to the client. If the message is ―end‖, closes the fifo and ends the process.
step 7:stop.
```

**Algorithm for client :**

```
Step 1: start
Step 2: Opens the named pipe for read and write purposes.Step 3: Accepts string from the user.
Step 4: Checks, if the user enters ―end‖ or other than ―end‖. Either way, it sends a message to the server. However, if the string is ―end‖, this closesthe FIFO and also ends the process.
Step 5: If the message is sent as not ―end‖, it waits for the message (reversed string) from the client and prints the reversed string.
Step 6: Repeats infinitely until the user enters the string ―end‖. Step 7: stop
```

**<u>Programs:</u>**

**/* Filename: fifoserver_twoway.c */**

```
  #include <stdio.h>
  #include <sys/stat.h>
  #include <sys/types.h>
  #include <fcntl.h>
  #include <unistd.h>
  #include <string.h>

  #define FIFO_FILE "/tmp/fifo_twoway"
  void reverse_string(char *);
int main()
 {
int fd;

 char readbuf[80];
 char end[10];

 int to_end;
  int read_bytes;

  /* Create the FIFO if it does not exist */mkfifo(FIFO_FILE, S_IFIFO|0640);
  strcpy(end, "end");
  fd = open(FIFO_FILE, O_RDWR);
  while(1)
  {
  read_bytes = read(fd, readbuf, sizeof(readbuf));
  readbuf[read_bytes] = '\0';
 printf("FIFOSERVER:   Received   string:   \"%s\"   and   length   is   %d\n",   readbuf,
  (int)strlen(readbuf));
  to_end = strcmp(readbuf, end);

if (to_end == 0) {close(fd); break;
  }
  reverse_string(readbuf);
 printf("FIFOSERVER: Sending Reversed String: \"%s\" and length is %d\n", readbuf,
  (int)strlen(readbuf));
  write(fd, readbuf, strlen(readbuf));
  /*
  sleep - This is to make sure other process reads this, otherwise thisprocess would retrieve
  the message
  */ sleep(2);
  }
  return 0;
  }


void reverse_string(char *str)
{
    int last, limit, first;
  char temp;
  last = strlen(str) - 1;limit = last/2;
  first = 0;

while (first < last)
{
 temp = str[first];
str[first] = str[last
str[last] = temp;
first++;
  last--;
  }
return;
}
```

**OUTPUT:**

```
FIFOSERVER: Received string: "LINUX IPCs" and length is 10
FIFOSERVER: Sending Reversed String: "sCPI XUNIL" and length is 10
FIFOSERVER: Received string: "Inter Process Communication" and length is 27
FIFOSERVER: Sending Reversed String: "noitacinummoC ssecorP retnI" and length
is 27
FIFOSERVER: Received string: "end" and length is 3
```

**/* Filename: fifoclient_twoway.c */**

```
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

#define FIFO_FILE "/tmp/fifo_twoway"
int main()
{
int fd;
int end_process;int stringlen;
int read_bytes; char readbuf[80];char end_str[5];
printf("FIFO_CLIENT: Send messages, infinitely, to end enter \"end\"\n");
fd = open(FIFO_FILE, O_CREAT|O_RDWR);
strcpy(end_str, "end");

while (1)
{
printf("Enter string:  ");
fgets(readbuf, sizeof(readbuf), stdin);
stringlen = strlen(readbuf);
readbuf[stringlen - 1] = '\0';
end_process = strcmp(readbuf, end_str);


//printf("end_process is %d\n", end_process);
if (end_process != 0)
{
write(fd, readbuf, strlen(readbuf));
printf("FIFOCLIENT: Sent string: \"%s\" and string length is %d\n", readbuf,
(int)strlen(readbuf));
read_bytes = read(fd, readbuf, sizeof(readbuf)); readbuf[read_bytes] = '\0';
printf("FIFOCLIENT: Received string: \"%s\" and length is %d\n", readbuf,
(int)strlen(readbuf));
}
else
{
write(fd, readbuf, strlen(readbuf));
printf("FIFOCLIENT: Sent string: \"%s\" and string length is %d\n", readbuf,
(int)strlen(readbuf));
close(fd);
break;
}
}
return 0;
}
```

**OUTPUT:**

```
FIFO_CLIENT: Send messages, infinitely, to end enter "end"
Enter string: LINUX IPCs
FIFOCLIENT: Sent string: "LINUX IPCs" and string length is 10
FIFOCLIENT: Received string: "sCPI XUNIL" and length is 10
Enter string: Inter Process Communication
FIFOCLIENT: Sent string: "Inter Process Communication" and string length is 27
FIFOCLIENT: Received string: "noitacinummoC ssecorP retnI" and length is 27
Enter string: end
FIFOCLIENT: Sent string: "end" and string length is 3
```

## WEEK 6

**AIM** : Write a C program that receives a message from message queue and display them.

**ALGORITHM:**

```
Step 1:Start
Step 2:Declare a message queue
typedef struct msgbuf
{
long mtype;
char mtext[MSGSZ];
}
message_buf;
Mtype =0                     Retrieve the next message on the queue, regardless of its
mtype.
PositiveGet the next message with an mtype equal to the specifiedmsgtyp.
 Negative                    Retrieve the first message on the queue whose mtype
fieldisless than or equal to the absolute value of the msgtyp argument.
Usually mtype is set to1
mtext is the data this will be added to the queue.
Step 3:Get the message queue id for the "name" 1234, which was created by the server
key = 1234
Step 4 : if ((msqid = msgget(key, 0666< 0) Then print error
The msgget() function shall return the message queue identifier associated with the
argument key.
Step 5: Receive message from message queue by using msgrcv function
int msgrcv(int msqid, void *msgp, size_t msgsz, long msgtyp, int ms
gflg);
#include < sys/msg.h>
(msgrcv(msqid, &rbuf, MSGSZ, 1, 0)msqid: message queue id
&sbuf: pointer to user defined structure MSGSZ: message sizeMessage type: 1
Message flag:The msgflg argument is a bit mask constructed by ORing together zero
or more of the following flags: IPC_NOWAIT or MSG_EXCEPT or
MSG_NOERROR
Step 6:if msgrcv <0 return error
Step 7:otherwise print message sent is sbuf.mext Step 8:stop
```

**Program:**

```
//IPC_msgq_send.c

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAXSIZE             128
void die(char *s)
```

```
{
perror(s);
exit(1);
}
typedef struct msgbuf
{
long                    mtype;
char                    mtext[MAXSIZE];
};

main()
{
int msqid;
int msgflg = IPC_CREAT | 0666;
key_t key;
struct msgbuf sbuf;
size_t buflen;

key = 1234;

   if ((msqid = msgget(key, msgflg )) < 0) //Getthe message queue ID for the given key
die("msgget");

//Message Typesbuf.mtype = 1;

   printf("Enter a message to add to messagequeue : ");
scanf("%[^\n]",sbuf.mtext);
getchar();

buflen = strlen(sbuf.mtext) + 1 ;

 if (msgsnd(msqid, &sbuf, buflen,
IPC_NOWAIT) < 0)
{
      printf ("%d, %d, %s, %d\n", msqid,sbuf.mtype, sbuf.mtext, buflen);
die("msgsnd");
}

else
printf("Message Sent\n");

exit(0);
}
```

**Program:**

```
//IPC_msgq_rcv.c

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdio.h>
#include <stdlib.h>
#define MAXSIZE

void die(char *s)
{
perror(s);exit(1);
}

typedef struct msgbuf
{
long                    mtype;
char                    mtext[MAXSIZE];
} ;
main()
{
int msqid;key_t key;
struct msgbuf rcvbuffer;key = 1234;

if ((msqid = msgget(key, 0666)) < 0)die("msgget()");

//Receive an answer of message type 1.
 if (msgrcv(msqid, &rcvbuffer, MAXSIZE, 1, 0) < 0)die("msgrcv");

printf("%s\n", rcvbuffer.mtext);exit(0);
}
```

**Computer Networks**
**WEEK 7(a)**

**NAME OF THE EXPERIMENT:** Bit Stuffing.

**AIM:** Implement the data link layer framing methods such as and bit stuffing.

**ALGORITHM:**

```
    Begin
Step 1: Read frame length n
Step 2: Repeat step (3 to 4) until i<n(: Read values in to the input
frame (0's and1's) i.e.
Step 3: initialize I i=0;
Step 4: read a[i] and increment i
Step 5: Initialize i=0, j=0,count =0
Step 6: repeat step (7 to 22) until i<n
Step 7: If a[i] == 1 then
Step 8: b[j] = a[i]
Step 9: Repeat step (10 to 18) until (a[k] =1 and k<n and count <5)
Step 10: Initialize k=i+1;
Step 11: Increment j and b[j]= a[k];
Step 12: Increment count ;
Step 13: if count =5 then
Step 14: increment j,
Step 15: b[j] =0
Step 16: end if
Step 17: i=k;
Step 18: increment k
Step 19: else
Step 20: b[j] = a[i]
Step 21: end if
Step 22: increment I and j
Step 23: print the frame after bit stuffing
Step 24: repeat step (25 to 26) until i< j
Step 25: print b[i]
```

**Step 26:** increment i

**End**

**SOURCE CODE:**

```
// BIT Stuffing program
#include<stdio.h>
#include<string.h>
void main()
{
int
a[20],b[30],i,j,k,
count,n;
printf("Enter
frame length:");
scanf("%d",&n)
printf("Enter
input frame (0's
& 1's only):");
for(i=0;i<n;i++)
scanf("%d",&a[i]);i=0;
count=1;
j=0;
while(i<n)
{
if(a[i]==1)
{
b[j]=a[i];
for(k=i+1;a[k]==1 && k<n && count<5;k++)
{
j++;
b[j]=a[k];
count
if(count==5)
{
j++;
b[j]=0;
}
   i=k;
   }}
   else
   {
   b[j]=a[i];
}
i++;
j++;
   }
   printf("After stuffing
   the frame is:");
   for(i=0;i<j;i++)
   printf("%d",b[i]);
   }
```

**OUTPUT:**

```
   Enter frame length: 5
   Enter input frame (0's
   & 1's only):1
   1
   1
   1
   1
   After stuffing the frame is: 111110
```

## Week: 7(b)

**NAME OF THE EXPERIMENT:** Character Stuffing.
**AIM:** Implement the data link layer framing methods such as character, character stuffing.
**ALGORITHM:**

```
Begin
Step 1: Initialize I and j as 0
Step 2: Declare n and pos as integer and a[20],b[50],ch as character
Step 3: read the string a
Step 4: find the length of the string n, i.e n-strlen(a)
Step 5: read the position, pos
Step 6: if pos > n then
Step 7: print invalid position and read again the position, pos
Step 8: end if
Step 9: read the character, ch
Step 10: Initialize the array b , b[0…5] as ’d’, ’l’, ’e’, ’s’ ,’t’
,’x’respectively
Step 11: j=6;
Step 12: Repeat step[(13to22) until i<n
Step 13: if i==pos-1 then
Step 14: initialize b array,b[j],b[j+1]…b[j+6] as‘d’, ‘l’, ‘e’ ,’ch, ’d’, ‘l’,‘e’respectively.
Step 15: increment j by 7, i.e j=j+7
Step 16: end if
Step 17: if a[i]==’d’ and a[i+1]==’l’ and a[i+2]==’e’ then
Step 18: initialize array b, b[13…15]=’d’, ‘l’, ‘e’ respectively
Step 19: increment j by 3, i.e j=j+3
Step 20: end if
Step 21: b[j]=a[i]
Step 22: increment I and j;
Step 23: initialize b array,b[j],b[j+1]…b[j+6] as‘d’, ‘l’,‘e’ ,’e’,‘t’, ‘x’,‘\0’ respectively
Step 24: print frame after stuffing
Step 25: print b
End
```

**SOURCE CODE:**

```
//PROGRAM FOR CHARACTER STUFFING
#include<stdio.h>
#include<string.h>
#include<process.h>
void main()
    {
    int i=0,j=0,n,pos;
    char a[20],b[50],ch;
    printf("enter string\n");
    scanf("%s",&a);
    n=strlen(a);
    printf("enter position\n");
    scanf("%d",&pos);
    if(pos>n)
    {
    printf("invalid position, Enter again :");
    scanf("%d",&pos);
    }
    printf("enter the character\n");
    ch=getche();
    b[0]='d';
    b[1]='l';
    b[2]='e';
    b[3]='s';
    b[4]='t';
    b[5]='x';

    j=6;
    while(i<n)
    {
    if(i==pos-1)
    {
    b[j]='d';
    b[j+1]='l';
    b[j+2]='e';
    b[j+3]=ch;
    b[j+4]='d';
    b[j+5]='l';

    b[j+6]='e';

    j=j+7;

    }

    if(a[i]=='d' && a[i+1]=='l' && a[i+2]=='e')

    {

    b[j]='d';

    b[j+1]='l';

    b[j+2]='e';

    j=j+3;

    }

    b[j]=a[i];

    i++;

    j++;

    }

    b[j]='d';

    b[j+1]='l';

    b[j+2]='e';

    b[j+3]='e';

    b[j+4]='t';

    b[j+5]='x';

    b[j+6]='\0';

    printf("\nframe after

    stuffing:\n");

    printf("%s",b);

    }
```

**OUTPUT:**

```
enter string: MRCET
enter position:2
enter the character frame after stuffing:

dlestxMdldleTECRMdleetx
```

## Week: 8

**NAME OF THE EXPERIMENT:** Cyclic Redundancy Check.

**AIM:** Implement on a data set of characters the three CRC polynomials – CRC 12,CRC 16 and CRC CCIP.

**ALGORITHM/FLOWCHART:**

```
Begin
Step 1: Declare I,j,fr[8],dupfr[11],recfr[11],tlen,flag,gen[4],genl,frl,rem[4] as integer
Step 2: initialize frl=8 and genl=4
Step 3: initialize i=0
Step 4: Repeat step(5to7) until i<frl
Step 5:read fr[i]
Step 6: dupfr[i]=fr[i]
Step 7: increment i
Step 8: initialize i=0
Step 9: repeat step(10to11) until i<genl
Step 10: read gen[i]
Step 11: increment i
Step 12: tlen=frl+genl-1
Step 13: initialize i=frl
Step 14: Repeat step(15to16) until i<tlen
Step 15: dupfr[i]=0
Step 16: increment i
Step 17: call the function remainder(dupfr)
Step 18: initialize i=0
Step 19: repeat step(20 to 21) until j<genl
Step 20: recfr[i]=rem[j]
Step 21: increment I and j
Step 22: call the function remainder(dupfr)
Step 23: initialize flag=0 and i=0
Step 24: Repeat step(25to28) until i<4
Step 25: if rem[i]!=0 then
Step 26: increment flag
Step 27: end if
Step 28: increment i
```

**Step 29:** if flag=0 then

**Step 25:** print frame received correctly

**Step 25:** else

**Step 25:** print the received frame is wrong

**End**

**SOURCE CODE:**

**//PROGRAM FOR CYCLIC REDUNDENCY CHECK**

```
#include<stdio.h>

int gen[4],genl,frl,rem[4];

void main()
{
int i,j,fr[8],dupfr[11],recfr[11],tlen,flag;frl=8; genl=4;
printf("enter frame:");
for(i=0;i<frl;i++)
{
scanf("%d",&fr[i]);
dupfr[i]=fr[i];
}
printf("enter generator:");
for(i=0;i<genl;i++)
scanf("%d",&gen[i]);
tlen=frl+genl-1;
for(i=frl;i<tlen;i++)
{

dupfr[i]=0;

}

remainder(dupfr);

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

recfr[i]=fr[i];

}

for(i=frl,j=1;j<genl;i++,j++)

{

recfr[i]=rem[j];

}
```

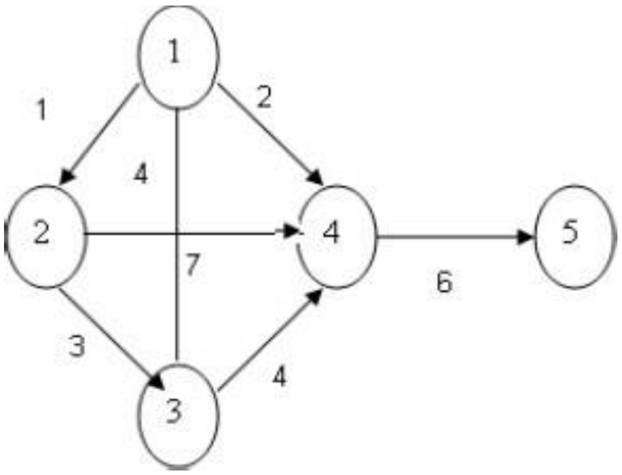
```
remainder(recfr);
flag=0;
for(i=0;i<4;i++)
{
if(rem[i]!=0)
flag++;
}
if(flag==0)
{
printf("frame received correctly");
}
else
{
printf("the received frame is wrong");
}
}
remainder(int fr[])
{
int k,k1,i,j;
 for(k=0;k<frl;k++)
{

if(fr[k]==1)
{
k1=k;
for(i=0,j=k;i<genl;i++,j++)
{
rem[i]=fr[j]^gen[i];
}
for(i=0;i<genl;i++)
{
fr[k1]=rem[i];
k1++;
}
}
}
}
```

**OUTPUT:**

```
enter frame: MRCET
enter generator: frame received correctly
```

## Week: 9

**NAME OF THE EXPERIMENT:** Shortest Path.
**AIM:** Implement Dijkstra‘s algorithm to compute the Shortest path thru a given graph



**ALGORITHM/FLOWCHART:**

**Begin**

**Step1**: Declare array path [5] [5], min, a [5][5],index, t[5];

**Step2:** Declare and initialize st=1,ed=5

**Step 3:** Declare variables i, j, stp, p, edp

**Step 4**: print “enter the cost “
**Step 5**: i=1

**Step 6**: Repeat step (7 to 11) until (i<=5)

**Step 7**: j=1

**Step 8**: repeat step (9 to 10)until (j<=5)

**Step 9**: Read a[i] [j]

**Step 10**: increment j

**Step 11**: increment i

**Step 12**: print “Enter the path”
**Step 13**: read p
**Step 14**: print “Enter possible paths”Step 15: i=1
**Step 16**: repeat step(17 to 21) until (i<=p)
**Step 17**: j=1
**Step 18**: repeat step(19 to 20) until (i<=5)
**Step 19**: read path[i][j]
**Step 20**: increment j
**Step 21**: increment i

```
Step 22: j=1
Step 23: repeat step(24 to 34) until(i<=p)
Step 24: t[i]=0
Step 25: stp=st
Step 26: j=1
Step 27: repeat step(26 to 34) until(j<=5)
Step 28: edp=path[i][j+1]
Step 29: t[i]= [ti]+a[stp][edp]
Step 30: if (edp==ed) then
Step 31: break;
Step 32: else
Step 33: stp=edp
Step 34: end if
Step 35: min=t[st]
Step 36: index=st
Step 37: repeat step( 38 to 41) until (i<=p)
Step 38: min>t[i]
Step 39: min=t[i]
Step 40: index=i
Step 41: end if
Step 42: print" minimum cost" min
Step 43: print" minimum cost pth"
Step 44: repeat step(45 to 48) until (i<=5)
Step 45: print path[index][i]
Step 46: if(path[idex][i]==ed) then
Step 47: break
Step 48: end if
End
```

**SOURCE CODE:**
**PROGRAM FOR FINDING SHORTEST PATH FOR A GIVEN GRAPH**

```
#include<stdio.h>
void main()
{
int path[5][5],i,j,min,a[5][5],p,st=1,ed=5,stp,edp,t[5],index;
printf("enter the cost matrix\n");
for(i=1;i<=5;i++)
for(j=1;j<=5;j++)
scanf("%d", &a[i][j]);
printf("enter the paths\n");
scanf("%d",&p);
printf("enter possible paths\n");
for(i=1;i<=p;i++) for(j=1;j<=5;j++)
scanf("%d",&path[i][j]); for(i=1;i<=p;i++)
{
t[i]=0;
stp=st;
for(j=1;j<=5;j++)
{
edp=path[i][j+1];
t[i]=t[i]+a[stp][edp];
if(edp==ed)
break;
else stp=edp;
}
}
min=t[st];index=st;
for(i=1;i<=p;i++)
{
if(min>t[i])
{
min=t[i];index=i;
}

}

printf("minimum cost %d",min);

printf("\n minimum cost path ");

for(i=1;i<=5;i++)
{
printf("--> %d",path[index][i]);
if(path[index][i]==ed)
break;

}

}
```

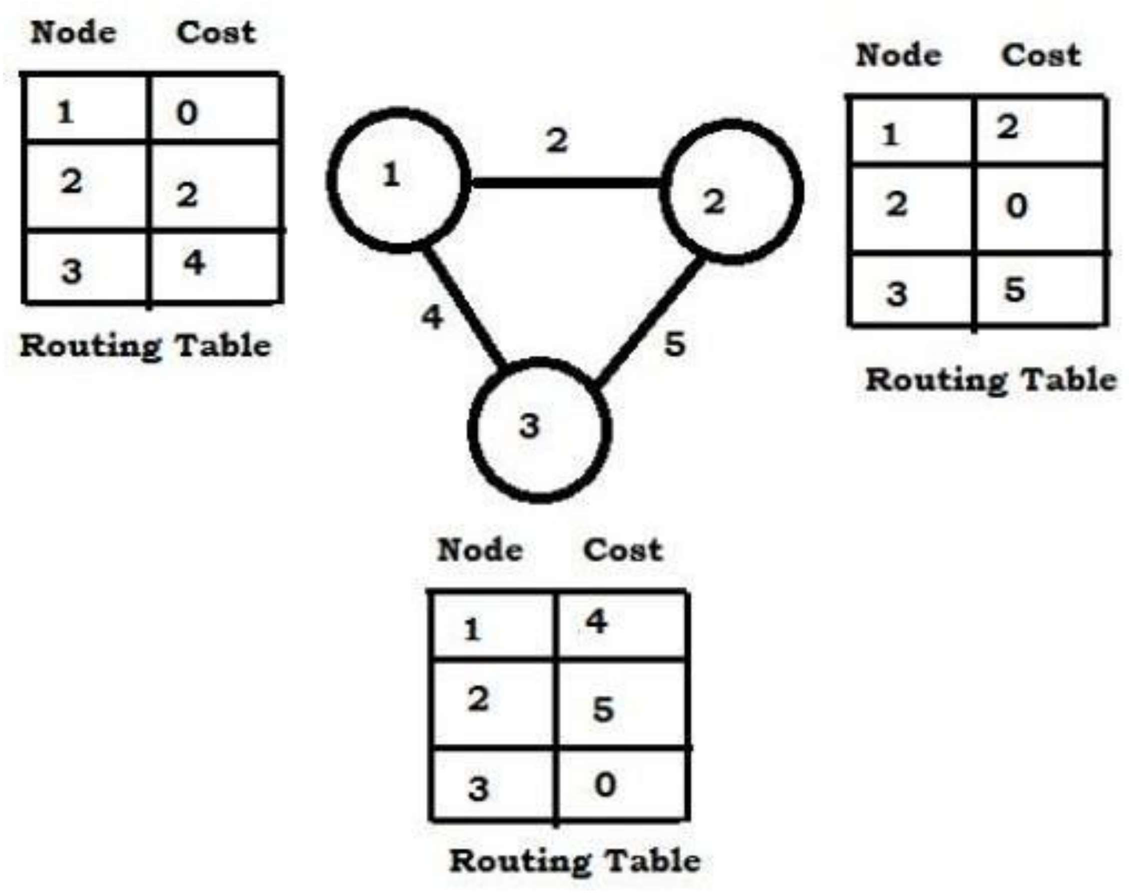
**OUTPUT:**

```
enter the cost matrix
 1 2 3 4 5
1 2 3 4 5

1 2 3 4 5

1 2 3 4 5

1 2 3 4 5

enter the paths2

enter possible paths1 2 3 4 5
1 2 3 4 5

minimum cost 14
```

## Week: 10

**NAME OF THE EXPERIMENT:** Distance Vector routing.

**AIM:** Obtain Routing table at each node using distance vector routing algorithm for agiven subnet.



| Node | Cost |
|---|---|
| 1 | 0 |
| 2 | 2 |
| 3 | 4 |

Routing Table

| Node | Cost |
|---|---|
| 1 | 2 |
| 2 | 0 |
| 3 | 5 |

Routing Table

| Node | Cost |
|---|---|
| 1 | 4 |
| 2 | 5 |
| 3 | 0 |

Routing Table

**ALGORITHM:**

**Begin**

**Step1**: **Create struct node unsigned dist[20],unsigned from[20],rt[10]**

**Step2:** initialize int dmat[20][20], n,i,j,k,count=0,

**Step3:** write "the number of nodes "

**Step4:** read the number of nodes "n"

**Step5:** write" the cost matrix :"

**Step6:** intialize i=0

**Step7:** repeat until i<n

**Step8:** increment i

**Step9:** initialize j=0

**Step10:** repeat Step(10-16)until j<n

**Step11**: increment j

**Step12**:read dmat[i][j]

**Step13**:intialize dmat[i][j]=0

**Step14**:intialize rt[i].dist[j]=dmat[i][j]

**Step15**:intialize rt[i].from[j]=j

**Step16**:end

**Step17**:start do loop Step (17-33)until

**Step18**:intilialize count =0

**Step19**:initialize i=0

**Step20**:repeat until i<n

**Step21**:increment i

**Step22**:initialize j=0

**Step23**:repeat until j<n

**Step24**:increment j

**Step25**:initialize k=0

**Step26**:repeat until k<n

**Step27**:increment k

**Step28**:if repeat Step(28-32) until rt[i].dist[j]>dmat[i][k]+rt[k].dist[j]

**Step29**:intialize rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j]

**Step30**:intialize rt[i].from[j]=k;

**Step31**:increment count

**Step32**:end if

**Step33**:end do stmt

**Step34**:while (count!=0)

**Step35**:initialize i=0

**Step36**:repeat Steps(36-44)until i<n

**Step37**:increment i

**Step38**:write ' state values for router',i+1

**Step39**:initialize j=0

```
Step40:repeat Steps ( 40-43)until j<n
Step41: increment j
Step42:write 'node %d via %d distance % ',j+1,rt[i].from[j]+1,rt[i].dist[j]
Step43:end
Step44:end
end
```

**SOURCE CODE:**

```
#include<stdio.h>
struct node
   {
   unsigned dist[20];
   unsigned from[20];
   }
   rt[10];

   int main()

   {

   int dmat[20][20];

   int n,i,j,k,count=0;
   printf("\nEnter the number of nodes : ");
   scanf("%d",&n);

   printf("Enter the cost

   matrix :\n");

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

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

   {

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

   dmat[i][i]=0;

   rt[i].dist[j]=dmat[i][j];

   rt[i].from[j]=j;
   }

   do

   {

   count=0;

   for(i=0;i<n;i++)
   for(j=0;j<n;j++)
    for(k=0;k<n;k++)
   if(rt[i].dist[j]>dmat[i][k]+rt[k].dist[j])
   {
   rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
   rt[i].from[j]=k;
   count++;
   }}
   while(count!=0);
   for(i=0;i<n;i++)
   {
   printf("\nState value for router %d is \n",i+1);for(j=0;j<n;j++)
   {
   printf("\nnode %d via %d Distance%d",j+1,rt[i].from[j]+1,rt[i].dist[j]);
   }
   }
   printf("\n");
   }
```

**OUTPUT:**

```
Enter the number of nodes: 2
Enter the cost matrix:
   1 2
   1 2
State value for router 1 is
node 1 via 1 Distance0
node 2 via 2 Distance2
State value for router 2 is
node 1 via 1 Distance1
node 2 via 2 Distance0
```

## Week: 11

**NAME OF THE EXPERIMENT:** Broadcast Tree.

**AIM:** Implement broadcast tree for a given subnet of hosts

**ALGORITHM/FLOWCHART:**

```
step 1: declare variable as int p,q,u,v,n;
step 2: Initialize min=99,mincost=0;
step 3: declare variable as int t[50][2],i,j;
step 4: declare variable as int parent[50],edge[50][50];
step 5: Begin
step 6: write "Enter the number of nodes"
step 7: read "n"
step 8: Initialize i=0
step 9: repeat step(10-12) until i<n
step10: increment i
step11: write"65+i"
step12: Initialize parent[i]=-1step13:wite "\n"
step14: Initialize i=0
step15: repeat step(15-21) until i<n
step16: increment i
step17: write"65+i"
step18: Initialize j=0
step19: repeat until j<n
step20: increment j
step21: read edge[i][j]
step22: Initialize i=0
step23: repeat step(23-43) until i<n
step24: increment i
step25: Initialize j=0
step26: repeat until j<n
step27: increment j
step28: if'edge[i]j]!=99
step29: if'min>edge[i][j] repeat step (29-32)
step30: intialize min=edge[i][j]
step31: intialize u=istep32: intialize v=j
step33: calling function p=find(u);
step34: calling function q=find(v);
step35: if'P!=q repeat steps(35-39)
step36: intialize t[i][0]=u
step37: intialize t[i][1]=v
step38: initialize mincost=mincost+edge[u][v]step39: call function sunion(p,q)
step40: else repeat steps(40-42)
step41: Intialize t[i][0]=-1;
step42: Intialize t[i][1]=-1;
step43: intialize min=99;
step44; write"Minimum cost is %d\n Minimum spanning tree is",mincost
```

```
step45: Initialize i=0
step46: repeat until i<nstep47: increment i
step48: if't[i][0]!=-1 && t[i][1]!=-1'repeat step(48-50)
step49: write "%c %c %d", 65+t[i][0], 65+t[i][1], edge[t[i][0]][t[i][1]]
step50: write"\n"
step51: end
step52: called function sunion(int l,int m) repeat step(51-52)
step53: intialize parent[l]=m
step54: called function find(int l) repeat step(53-56)
step55: if parent([l]>0)
step56: initialize l=parent
step57: return l
```

**SOURCE CODE:**
**// Write a program for Broadcast tree from subnet of host**

```
#include<stdio.h>
int p,q,u,v,n;
int min=99,mincost=0;int t[50][2],i,j;
int parent[50],edge[50][50];main()
   {
   clrscr();
   printf("\n Enter the number of nodes");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
   printf("%c\t",65+i);
   parent[i]=-1;
   }
   printf("\n");
    for(i=0;i<n;i++)
   {
   printf("%c",65+i);
   for(j=0;j<n;j++)
   scanf("%d",&edge[i][j]);
   }
   for(i=0;i<n;i++)
   {
   for(j=0;j<n;j++)
   if(edge[i][j]!=99)
   if(min>edge[i][j])
   {
   min=edge[i][j];u=i;
   v=j;
   }
   p=find(u);
   q=find(v)
   ;if(p!=q)
   {
   t[i][0]=u;t[i][1]=v;
   mincost=mincost+edge[u][v];
   sunion(p,q);
   }
   else
   {
   t[i][0]=-1;
   t[i][1]=-1;
   }
   min=99;
   }printf("Minimum cost is %d\n Minimum spanning tree is\n" ,mincost);
   for(i=0;i<n;i++)
   if(t[i][0]!=-1 && t[i][1]!=-1)
   {
   printf("%c %c %d", 65+t[i][0], 65+t[i][1],
   edge[t[i][0]][t[i][1]]);
   printf("\n");
   }
   }
   sunion(int l,int m)
   {
   parent[l]=m;
   }
   find(int l)
   {
```

```
if(parent[l]>0)
l=parent[l];
return l;
}
```

**OUTPUT:**

```
Enter the number of nodes 3
A       B       C
A1 2 3 4
B1 2 3 4
C4 5 6 7
Minimum cost is 3
Minimum spanning tree is
C A 3
```