KGiSL Institute of Technology

(Approved by AICTE, New Delhi; Affiliated to Anna University, Chennai)

Recognized by UGC, Accredited by NBA (IT)

365, KGiSL Campus, Thudiyalur Road, Saravanampatti, Coimbatore – 641035.

AL3452 - OPERATING SYSTEM

NAME	: ....
REG. NO.	: ....
COURSE	: ....
SEMESTER	: ....
BATCH	: ....

KGiSL Institute of Technology

(Approved by AICTE, New Delhi; Affiliated to Anna University, Chennai)

Recognized by UGC, Accredited by NBA (IT)

365, KGiSL Campus, Thudiyalur Road, Saravanampatti, Coimbatore – 641035.

NAME	**:**
CLASS	**:**
UNIVERSITY REG NO	**:**

Certified that, this is a bonafide record of work done by ………………………………………..

Of ……………………………………………………………….. branch in **OPERATING SYSTEM LABORATORY**, during fourth semester of academic year 2022-2023.

Faculty In-charge **Head of the Department**

Submitted during Anna University Practical Examination held on at KGiSL Institute of Technology, Coimbatore – 641 035.

Internal examiner **External Examiner**

INDEX

	A	WORST-FIT			
	B	BEST-FIT			
	C	FIRST-FIT			
12		SIMULATE PAGE REPLACEMENT ALGORITHMS			
	A	FIFO			
	B	LRU			
	C	OPTIMAL			
13		IMPLEMENTATION OF FILE ORGANIZATION TECHNIQUE			
		SINGLE-LEVEL FILE ORGANIZATION TECHNIQUE			
		TWO-LEVEL FILE ORGANIZATION TECHNIQUE			
		DAG FILE ORGANIZATION TECHNIQUE			
14		IMPLEMENTATION OF FILE ALLOCAION TECHNIQUE			
		SEQUENTIAL FILE ALLOCAION TECHNIQUE			
		INDEXED FILE ALLOCATION			
		LINKED FILE ALLOCATION			
15		DISK SCHEDULING ALGORITHMS			
		FCFS			
		SCAN			
		C-SCAN			

Ex.No: 1

Date:

Operating System Installation

Windows XP Installation

It is important to understand that this guide was specifically designed for a lab environment. There are a lot of operating system vulnerabilities that are intentionally left unpatched in these installation steps. This is intentionally done to give you the best results when completing the labs and tutorials in this book. If you are interested, a great reference for building a Windows XP Professional box that is secure enough for a production environment is *Windows XP Security: Step By Step* by SANS.

To create a properly configured laptop for the Security Essentials Boot Camp, follow the detailed steps in this document—from the initial setup screen to the final login. This guide was designed for use on a system that doesn't already have a Windows platform installed on it. If your machine does not have a blank hard drive, some of the screens yousee at the beginning of the installation may be different from what you see in this chapter.If different screens appear, it is important that you always choose the option to *replace*, or *overwrite*. Do not choose to upgrade. The Windows install should also be placed in thedefault c:\windows directory.

Creating Boot Disks

If your system does not support the capability to boot off of a CD-ROM, you can use the Windows XP boot disk to boot. If you do not have a set of the four disks, you need to use a machine that already has Windows XP Professional installed on it. The following stepsshow you how to create the four boot disks:

1. Label four blank, formatted, 3.5-inch, 1.44-MB floppy disks as: **Setup Disk One**, **Setup Disk Two**, **Setup Disk Three**, and **Setup Disk Four**.
2. Insert **Setup Disk One** into the floppy disk drive of a Windows or DOS system.
3. Insert the Windows XP CD-ROM into the CD-ROM drive.
4. Click **Start**, and then click **Run**.
5. In the **Open** box, type **D:\bootdisk\makeboot a:** (where **D:** is the drive letter assigned to your CD-ROM drive), and then click **OK**.
6. Follow the screen prompts.
7. After you have completed the screen prompt requests, insert **Setup Disk One** into the floppy disk drive of the lab PC and power the PC on.

Booting from the CD-ROM

If your system supports booting off of the CD-ROM, you do not need to use the disks previously discussed. Instead, follow these steps:

1. Simply start by placing the Windows XP CD-ROM into your CD tray and power on your machine. The first non-blank screen you should see is the

one shown in the following illustration.

2. If the previous screen does not appear, reboot your machine and open up the BIOS. You need to make the system boot to the CD-ROM first. The following screen is one of several different BIOSes you could have on your system. You need to navigate to a screen that allows you to change the Boot Order. This is where you tell it to boot off of the CD-ROM.
3. Now your system should boot off of the CD-ROM. After a period of time (typically 30-45 seconds), the following screen appears. Because we are doing an initial install, you only need to press **Enter** to continue.

4. Hit **Enter** at the next screen to continue installation.
5. The Microsoft Windows XP Licensing Agreement appears next, as shown in the following screen. It is important that you read and understand this agreement before continuing with the installation. After you have read and agreed to the contents of the license, press **F8** to continue.

Defining Drive Partitions

You now need to define the drive partitions. Defining your drive partitions is used instead of FDISK. When defining your drive partitions, it is extremely important that you leave enough space for your Linux partition! Following are the steps:

1. Press **C** to create a partition for your Windows install.

2. You need a minimum of 2Gb of space for each of your operating systems. When you are prompted for the size of the partition, enter a number that is equal to 50 percent of your available hard drive space. Then, highlight the partition, which should be labeled **Unpartitioned space** (see the following illustration)**,** and press **C**.

Note: If partitions already exist they should be deleted. However you should realize that this will permanently remove any data that is currently on your system.

Now create your new partition to be at least 2 Gb. In the provided space type **2047** and press **Enter**

You should now see two partitions. Verify that the new 2047 partition is highlighted and press **Enter**.

Formatting Drive Partitions

The next step is to format your partition. For security reasons, you should format your partitions using NTFS. NTFS is a Windows partition type that allows you to assign permissions at the folder level. This level of granularity is not the same for FAT partitions. NTFS also allows for lager partition sizes compared to the 2Gb limit that comes with FAT16. The steps for formatting your partition follow:

1. Highlight the NTFS <Quick> partition option as shown in the following screen, and press **Enter**.

2. After you press **Enter**, the system formats the partition, as shown in the following screens. Depending on the size of the partition, this step can take from 5 minutes to an hour. This is a great time to refill your caffeine-laced beverage of choice. (You may need it because you have a long way to go.)

Since this will take a while you should just wait while this process continues.

When you return to your machine, you may see one of the following screens. Don't be alarmed. The system has completed the formatting process and has automatically rebooted. After this occurs, you have to answer the remaining install questions.

Customizing Your System

Now Windows presents a series of questions, which, when answered, customize your system. The following steps walk you through the process of customizing your system:

1. Typically, you only need to make changes during the next step (see the following screen) if you are located outside of the United States or if youuse a non-standard keyboard. If you are in the United States and you are using a standard QWERTY keyboard, press the **Next** button. If you are located outside of the United States, you should change your locale settings.
2. Enter your name and the organization you work for in the **Name** and **Organization** fields. For the purposes of this course, have some fun making up fictional names. Click the **Next** button when you are done.
3. In the next screen, enter the Product Key number that came with your software (find it on your CD). If you make a mistake when you enter thekey, you receive an **Invalid Key** message and the system gives you another opportunity to enter it. Once you enter in the valid key, press the**Enter** key.
4. Now enter a name in the **Computer name** field to name your computer. If you are part of a corporation's domain, you need to follow your corporation's guidelines for naming systems. For our purposes, name your machine whatever you desire. Then, type in a password in the **Administrator password** field. You also need to confirm the password, as shown in the following screen. Then, click the **Next** button.

Warning: A common mistake many administrators make at this stage is to leave the **Administrator password** field blank. It is highly advisable that you enter a password that matches your company's password policy for local passwords. You don't want to forget to change the password after you have completed the installation. Also, make sure you remember this password. You will need it to login.

Note: Depending on your configuration, you might receive the Modem Dialing Information Screen. Just cancel out of this or click Next to get to the nextscreen.

5. In the screen that appears, enter the current time, and then fill in the **Date** field and **Time Zone** field. Click **Next**.

6. After you make the previous configurations, the system installs your networking components, as shown in the following screen.

Customizing Network Settings

Now you need to set up your system so that it can be networked with other systems. Following are the steps:

1. First, you must choose the type of settings you are going to use. Note that it is rarely a good idea to use **Typical settings** when configuring an application or operating system. It is always a good idea to choose **Custom settings**, as shown in the following screen. When you use this option, you only install options you need. You won't end up installing something you aren't aware of because you chose an option that automatically does this. After you select the **Custom settings** option, click**Next**.
2. Windows no longer tries to install IPX/SPX, so there is nothing in the custom settings that you need to remove. This is a great time to setup your local IP address if you are not using a DHCP server in your environment. The assumption here is that you are not going to plug this test machine into a production environment, so it's safe to add your own IP address. Highlight **Internet Protocol (TCP/IP)** and click the **Properties** tab.
3. The following screen appears, which allows you to enter your own IP address. For the purposes of this exercise, use a non-routable IP address. Select ‘Use the following IP address’. Enter **192.168.1.2** in the correct fields. Then, enter a standard 24-bit subnet mask of **255.255.255.0**. To make the entire section complete, enter a default gateway setting of **192.168.1.1**. Enter the appropriate DNS server IP addresses for your environment into the fields shown in the following paragraph. You can leave your DNS sever fields blank for this system. Click OK. Click Next.
4. As previously stated, you are not joining a network or a domain, so just enter a name of your choice and leave the first **No** option enabled (see the illustration that follows). After you have the information entered, click **Next**.
5. Windows completes the networking portion of the installation and moveson to its final tasks. This step takes a long time, so take the opportunity tograb another caffeine-laced beverage.
6. If you get the following screen, shout for joy. Congratulations, you have successfully installed Windows XP. Click **Finish**, and then remove the Windows CD-ROM before the system reboots so that you don’t accidentally start the install process again. If you accidentally leave the CD-ROM in, and the install process starts again, simply remove the CD-ROM and hard-boot the machine (restart it).
9. After the next screen comes up, click the **OK** button.
10. As shown in the next screen, you now need to log in using the Administrator account and the password you entered earlier during the install. After you have entered the appropriate credentials, click **OK**.

Note: Depending on your version of the software you might get several screens about connecting to the Internet and registering Microsoft before you get the login screen.

Note: At this stage, the base installation of Windows XP is installed. The instructions that follow show you how to upgrade to Service Pack 2, which is recommended for this class.

Installing Service Pack 2

If you do not have SP2 on a CD, you need to get Internet access setup, so that you can patch this box to SP2. SP2 provides several functional patches, which is why you want to upgrade to it.

The first thing you need to do is verify that your NIC (Network Interface Card) is working and that you have connectivity. Ensure that your system has the NIC plugged into a switch or hub that is connected to the Internet. If you have a DHCP (Dynamic Host Configuration Protocol) server on your network, you should be able to automatically pull an IP address. Otherwise, you need to statically assign the appropriate IP address for your system.

Note: You will need to change the address scheme you entered earlier if you need to connect to the Internet to download SP2.

Following are the steps for upgrading to SP2:

1. Left click on the Start button located in the lower left portion of your screen. Then highlight and left click **Control Panel**.
2. Now click on **Network and Internet Connections.** The following screen should appear. Left click on **Network Connections**.
3. The following screen should appear. If you do not see a Local Area Connection, you do not have a NIC (Network Interface Card) installed or properly working. If this is the case you will need to check with your NIC vendor's documentation on getting your particular card installed in Windows XP. Most modern NICs are fully compatible with Windows XP.
4. Right click on **Local Area Connection** and then left click on **Properties** in the menu that appears. This screen shows you the different configuration items that this particular interface uses. To exit this, click on **OK**.
5. After the Hardware Manager is properly setup, you need to validate that the IP address we initially configured is on your system. Click on **Start, Run,** and thentype **cmd**. Type **ipconfig**. If you see an IP address next to the NIC, you can proceed.

If you do not see an IP address, or you see the address with **169.254.30.x**, you didn't pull an IP address from your DHCP server or the IP configuration step we preformed earlier was not successful. You will need to manually add an IP address by repeating the steps described during the installation of Windows XP. Ifyou need to repeat these steps to add an appropriate IP address for your network, do so now.

6. Next, you need to verify connectivity to the Internet. To make sure your local IP stack is functioning correctly, you can PING the loopback adapter. To do this, open another command window by selecting **Start, Run**, and then type **cmd**. Then, type **ping 127.0.0.1**, as shown in the following screen.
7. As you can see, there is connectivity to the local IP stack. This shows that the TCP/IP stack is functioning correctly. To verify Internet connectivity and that the DNS settings are working correctly, ping a web site. The IP address used in the following screen is not valid. You need to ping a valid IP address. For example, pinging **www.sans.org** should work.

   Note: If you are not on a network that is connected to the Internet this step will not work. Also, if you are properly connected to an Internet accessible network and you used the IP address we supplied, and it does not match the network information of your network, this step will not work. If the later is the case, please change your IP address to match the information that is appropriate for your environment.

8. If you get an **Unable to resolve name** message, you need to validate that you have entered your DNS servers into the TCP/IP properties of your NIC correctly.
9. To get your browser functioning, double-click the **Internet Explorer** icon on your desktop and follow the wizard's instructions. In the first window click on **Cancel** since we will not be using a modem.
10. In the next window choose the appropriate option for your home environment. If you are part of a LAN (Local Area Network), choose the first option (**Connect to the Internet)**. Then, click **Next**.
11. In the next screen, choose **Set up my connection manually**. Click **Next**.
12. Do not change any options for the next screen. Leave it alone, and simply click **Next**.
13. You have now setup Internet Explorer for web surfing, so click **Finish.** You can now go to Microsoft’s web site to download and install Service Pack 2.
14. The last thing we need to do prior to installing anything on our system, including the Service Pack, is to create a folder that we will be storing all of the installation executables throughout this book.

    Click on **Start** then **My Computer**. Double click on **C**. Now right click anywhere in the window and left click **New** then **Folder**. Name the new folder **tools** as shown below
15. To install Service pack 2, go to the following URL:

http://www.microsoft.com/windowsxp/sp2/default.mspx

The Microsoft web site is shown in the following screen.

16. Ensure that **English** is the Selected Language and click **Go.**

17. You can now select either **Express** or **Network Installations**. Both options work, but you should choose the **Network Installation** option. Next, you should download Service Pack 2 into the **tools** directory you created earlier or a different directory that you create.

18. After the download is completed, double-click the SP2 executable and follow the given prompts for installation.

19. Read the License Agreement carefully. When you understand and agree with it, click on the button next to **I Agree** then click **Next.**

20. Because you do not need to uninstall this Service Pack, you can check the **Do Not Archive Files…** option.

21. Now you get to wait and watch. It is important to note that this process can take an extremely long time, even if it seems your system has locked up, it most likely has not. After the Service Pack is completely installed, reboot your machine and you are ready to move on to the next section.

You have now successfully completed the installation of Windows XP.

Linux Installation

Now you are going to install Red Hat 9.0. The first step is to insert the Red Hat 9.0 Disc 1 into your CD-ROM drive. Next, power on the system. The system boots off of the CD-ROM and begins the Red Hat installation program. Follow these steps to complete the installation of Red Hat:

1. When the **Red Hat Installation** screen appears (the first screen) type **linux text** at the **boot:** prompt and press **Enter,** as shown in the followingscreen.
2. Press the **Tab** key until **Skip** is highlighted, and then press **Enter**.
3. The "Welcome' screen appears. Press **Enter**.
4. The 'Language Selection' screen appears. Ensure that the language is setto **English**. Press the **Tab** key until **OK** is highlighted, and then press **Enter**.
5. The 'Keyboard Selection' screen appears. Accept the default keyboard **us.** Press the **Tab** key until **OK** is highlighted and press **Enter**.
6. The 'Mouse Selection' screen appears. Press the **Tab** key until the box next to **Emulate 3 Buttons** is selected and press **Space Bar** to place an asterisk in the brackets **[*].** Next, press the **Tab** key until **OK** is highlighted and then press **Enter**.
7. The 'Installation Type' screen appears. Use the arrow keys to highlight **Custom**. Press the **Tab** key until **OK** is highlighted and then press **Enter**.
8. The 'Disk Partitioning Setup' screen appears. Press the **Tab** key until **Disk Druid** is highlighted, and then press **Enter**.

   Note that the values used to partition the hard drive may need to be altered basedon the memory and hard drive size of the system that you are using.

9. The 'Partitioning' screen appears. Press the **Tab** key until **New** is highlighted, and the press **Enter**.
10. The 'Add Partition' screen appears. In Mount Point: type **/.** Press the **Tab** key until the cursor is in the **Size (MB)**: field. Enter **5800**. Press the **Tab** key until **OK** is highlighted, and then press **Enter**.
11. The 'Partitioning' screen reappears. With the arrow and **Tab** keys, highlight **Free Space**, as shown in the following screen. Afterwards, pressthe **Tab** key until **New** is highlighted, and then press **Enter**.
12. The 'Add Partition' screen appears. Press the **Tab** key once to select the **File System type**: field. Using the arrow keys, highlight **swap.** Press the **Tab** key until the **Size (MB)**: field is selected. Enter **256**. Press the **Tab** key until **OK** is highlighted, and then press **Enter**. These fields and selections are shown in the following screen.
13. The 'Partitioning' screen reappears. Press the **Tab** key until **OK** is highlighted. Press **Enter**.
14. The 'Boot Loader Configuration' screen appears. Press the **Tab** key until

OK is highlighted, and then press **Enter**.

15. The ‘Boot Loader Configuration’ screen appears. Press the **Tab** key until **OK** is highlighted, and then press **Enter**.
16. The ‘Boot Loader Configuration’ screen appears. Press the **Tab** key until **OK** is highlighted, and then press **Enter**.
17. The ‘Boot Loader Configuration’ screen appears. Use the **Tab** and arrow keys to highlight **DOS** and then press the **Tab** key until **Edit** is highlighted, and then press **Enter**.
18. The ‘Edit Boot Label’ screen appears. Change the **Boot Label** field to **Windows XP.** Press the **Tab** key until **OK** is highlighted and press **Enter.**
19. The ‘Boot Loader Configuration’ screen appears. Press the **Tab** key until **OK** is highlighted and press **Enter**.
20. he ‘Boot Loader Configuration’ screen appears. Press the **Tab** key until **OK** is highlighted and press **Enter**.
21. The ‘Network Configuration for eth0’ screen appears. Press the **Spacebar** to remove the ***** (asterisk) in the following **[ ] Use bootp/dhcp** option. Press the **Tab** key to select the **IP address** field.

    Enter the following parameters:

• IP address:	192.168.1.50
• Netmask:	255.255.255.0
• Default gateway (IP):	192.168.1.2
• Primary nameserver:	192.168.1.4

    After you enter the parameters, press the **Tab** key until **OK** is highlighted, and then press **Enter**.
22. The ‘Hostname Configuration’ screen appears. Enter **linux-lab** in the Hostname field. Press the **Tab** key until **OK** is highlighted, and then press **Enter**.
23. The ‘Firewall Configuration’ screen appears. Press the **Tab** key until **( ) No Firewall** is selected. Press the **Spacebar** to insert an asterisk (*), as shown in the screen. Then, press the **Tab** key until **OK** is highlighted and press **Enter**.
24. The ‘Language Support’ screen appears. Press the **Tab** key until **OK** is highlighted, and then press **Enter**.
25. The ‘Time Zone Selection’ screen appears. Press the **Tab** key until **OK** is highlighted, and then press **Enter**. (If you are in a different time zone, use the **Tab** and **arrow** keys to select the appropriate time zone.)
26. The ‘Root Password’ screen appears. In the **Password:** field, type a strong password to use for the root account. Confirm the password bytyping it in the **Password (confirm)**: field. Press **Tab** until **OK** is highlighted and press **Enter**.
27. The ‘Authentication Configuration’ screen appears. Press the **Tab** keyuntil **OK** is

highlighted, and then press **Enter**.

28. The ‘Package Group Selection’ screen appears. Press **End** to highlight **Everything** and then press **Space Bar** to select it. (an asterisk identifiesthe option as selected) Press the **Tab** key until **OK** is highlighted and press **Enter**.
29. The ‘Installation to begin’ screen appears. Press **Enter**.
30. The ‘Formatting’ screen appears. The **Formatting / file system…** Message appears. Proceed to the next step.
31. The ‘Copying File’ screen appears. The **Transferring install image to hard drive…** message appears. Proceed to the next step.
32. The ‘Package Installation’ screen appears. Red Hat now starts installing the packages. Proceed to the next step.
33. The ‘Change CDROM’ screen appears. When prompted, insert the **RedHat Disc 2** and press **Enter**.
34. The ‘Change CDROM’ screen appears again. When prompted, insert **Red Hat Disc 3** and press **Enter**.
35. The ‘Post Install’ screen appears. After all of packages have been installed, Red Hat performs the post-install configuration, as shown in the following screen. Proceed to the next step.
36. The ‘Boot Diskette’ screen appears. Press **Enter** to create a boot disk.
37. The ‘Insert a floppy disk’ screen appears. Insert a blank diskette into yourfloppy drive. Press **TAB** to highlight **Make boot disk** and press **Enter**.
38. The ‘Video Card Configuration’ screen appears. Use the **Tab** key and **Enter** key to select the appropriate video card settings for your system. Press the **Tab** key until **OK** is highlighted, and then press **Enter**.
39. The ‘Monitor Configuration’ screen appears. Again, use the **Tab** and **Enter** keys to select the appropriate monitor settings for your system.Press the **Tab** key until **OK** is highlighted, and then press **Enter**.
40. The ‘X Customization’ screen appears. Press the **Tab** key until **( ) Text** isselected, and then press the **Spacebar**. Press the **Tab** key until **OK** is highlighted, and then press **Enter**.
41. The ‘Complete’ screen appears. Congratulations, you have just installed Red Hat Linux. After removing the boot disk created earlier in the installation, press **Enter** to reboot the system. The CD-ROM will eject during the reboot process.
42. As the system is rebooting you will be presented with the choice of booting into Red Hat or Windows XP. Use the arrow keys to select the OS that you want to boot into and press **Enter** to boot the choice. Note that the GRUB boot loader is only presented for a few seconds before thedefault OS is booted so you have to be paying attention.

VMWare Installation

This section will walk you through the process of installing VMWare and configuring itto work with Knoppix.

1) Install VMWare (These instructions are tested with VMWare 5.0)
    a. Double click on *.exe
    b. Follow instructions until VMWare install completes
    c. Enter the appropriate software license
    d. Reboot if asked

2) Download the Knoppix ISO image file and burn to CD
   http://www.knoppix-std.org/
3) Pre-requisites to running VMWare: (it can operate with less but for optimal performance, the following are recommended)
    a. Pentium 4, 2GHz and above
    b. At least 1Gig of RAM

4) Start VMWare (see screen below), Click on the circled icon *"New Virtual Machine"*
5) Click on *"Next"* (There are many tunable custom settings but for now just use the DEFAULT 'typical' settings)
6) Select *'Linux'* radio button, Select the *'Other Linux 2.6.x kernel'* in the version pull down, and click on *"Next"*
7) Name the Virtual Machine and select a location for the virtual machine. Select a location with a lot of free space. The recommended drive's free space should be at least 10 GigaBytes. When click on the *'browse'* button to select a location, you can also create directories if one does not exist. Finally select *'next'*
8) Use the default *'Bridged networking'* and click on *'next'*
9) Use the default disk size of *'4.0 G'* and click on *'finish'*

10) Knoppix Virtual Machine is now ready for use. Before power on the virtual machine one final configuration is required

11) Before power on the virtual machine can be powered on, one final configuration is required to make sure that the virtual machine catches the CDROM. Normally the 'auto detect' should work but to be sure that the virtual machine binds with the appropriate drive (especially with multi-drive machines), we are going to force a binding to a specified drive.

After clicking on the above circled icon the following screen will popped up. Click on the CDROM Device, then select the appropriate drive and click on OK

12) Now the Virtual machine is ready to be powered on to configure the BIOS to boot the CDROM. Click on the green play icon. Note that as soon as the Virtual PC BOOTS the key F2 is needed to be pressed right away to enter BIOS configuration.

13) Press F2 key at the following screen: (Enter BIOS configuration to boot from CD)

14) Note the mouse will not work now, so **all inputs are from the keyboard** (just like when the PC boots). NOTE: to release the VM control of the keyboard, press **ALT-CTRL** at the same time.
    - Press the right arrow key until the boot tab is highlighted
    - Press the ↓ arrow key until the CD-ROM drive is highlighted
    - Press the + button until the CD-ROM drive is at the top of the list (as shown)

15) Press the right arrow key to highlight the EXIT tab
    Select the item *'Exit Saving Changes'* (Make sure the Knoppix CD is inserted before hitting Enter)

    You should see a series of Knoppix boot screen and finally the X-Windows screen.

Ex.No:2a

BASICS OF UNIX COMMANDS
PART - A

Basic Commands:

1. **$ who**

NAME : who

SYNTAX

```
who [OPTION]... [ FILE | ARG1 ARG2 ]
```

DESCRIPTION -show who is logged on.

OPTIONS

-a, shows all details of users who are logged on
-H, --heading line of column headings.
-q, --count login names and number of users logged
on -u, --users list users logged in

OUTPUT

a. Print all the users who are
logged on [mech @labi21 ~]**$ who**

```
mech   :0 Mar 5 21:16
mech    pts/1      Mar 5 21:17 (:0.0)
```

b. Print the users who are logged on along with the column
headings [mech @labi21 ~]**$ who -uH**

```
NAME  LINE          TIME        IDLE       PID COMMENT
mech  :0          Mar 5 21:16    ?         2979
mech  pts/1         Mar 5 21:17  .          3175 (:0.0)
```

c. Print the total number
of users. [mech @labi21 p]**$**
who -q mech mech
  # users=2

2. **$who am i**

[mech @labi21 ~]**$ who am i**

```
mech  pts/1       Mar 5 21:17 (:0.0)
```

3. **$clear**

NAME

clear - clear the terminal screen

SYNTA
X

clear

DESCRIPTION

clear clears your screen if this is possible. It looks in the environment for the terminal type and then in the terminfo database to figure out how to clear the screen.

4. **$man**

NAME

man - format and display the on-line manual pages

SYNTAX

man [-acdfFhkKtwW] [--path] [-m system] [-p string]
[-C config_file] [-M pathlist] [-P pager] [-S
section_list] [section] name ...

DESCRIPTION

man formats and displays the on-line manual pages. If you specify section, man only looks in that section of the manual. name is normally the name of the manual, which is typically the name of a command, function, file.

OUTPUT

```
[mech @labi21 ~]$ man who
[mech @labi21 ~]$ man date
```

5. **$date**

NAME

date - print or set the system date and time

SYNTAX

date [OPTION]... [+FORMAT]
date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]

DESCRIPTION

Display the current time in the given FORMAT, or set the system date

OPTIONS

%a	locale’s abbreviated weekday name (Sun..Sat)
%B	locale’s full month name, variable length (January..December)
%d	day of month (01..31)
%m	month (01..12)
%p	locale’s upper case AM or PM indicator (blank in many locales)
%y	last two digits of year (00..99)
%Y	year (1970...)
%Z	time zone (e.g., EDT), or nothing if no time zone is determinable

OUTPUT

a. Display today's date

[mech @labi21 ~]**$ date**
Fri Mar 5 21:57:00 IST
2010

b. Display formatted date

[mech l@labi21 ~]**$ date '+ DATE:%d=%m-%y %n TIME : %H:%M:%S'**

DATE:05=03-10
TIME : 21:57:45

c. [mech @labi21 ~]**$ date '+%A, %B %d %Y'** Friday, March 05 2010

d. Display time in HH:MM:SS AM/PM [mech @labi21 ~]**$ date +%X**

e. Display the time zone
[mech @labi21 ~]**$ date +%Z**
IST

7. **$uname**

NAME

uname - print system information

SYNTAX

uname [OPTION]...

DESCRIPTION

Print certain system information.

OPTIONS

-a, --all
print all information, in the following order:
-p, --processor
print the processor type.
-i, --hardware-platform
print the hardware platform
-o, --operating-system
print the operating system

OUTPUT

a. Print all the system information [mech @labi21 ~]**$ uname -a**

```
Linux labi21 2.6.9-5.EL #1 Wed Jan 5 19:22:18 EST 2005 i686 i686
i386 GNU/Linux
```

b. Print name of OS and processor type.

[mech @labi21 ~]**$ uname -o -p**

```
i686 GNU/Linux
```

7. **$tty**

NAME

tty - print the file name of the terminal connected to standard input

SYNTAX

```
tty [OPTION]...
```

DESCRIPTION

Print the file name of the terminal connected to standard input.

OUTPUT:

[mech l@labi21 ~]**$ tty**

```
/dev/pts/1
```

8. **$id**

NAME

id - print real and effective UIDs and GIDs

SYNTAX

```
id [OPTION]... [USERNAME]
```

DESCRIPTION

Print information for USERNAME, or the current user.

OUTPUT

```
[mech @labi21 ~]$ id
uid=505(mech) gid=505(mech) groups=505(mech)
context=user_u:system_r:unconfined_t
```

9. **$cal**

NAME

    cal - displays a calendar

SYNTAX

    cal [-smjy13] [[month] year]

DESCRIPTION

    Cal displays a simple calendar. If arguments are not specified, the current month

is
    displayed.

OPTIONS

    -y   Display a calendar for the current year.

OUTPUT

a. Print calendar for current month [mech @labi21 ~]**$ cal**

```
    March 2010
Su Mo Tu We Th Fr Sa
   1 2 3 4 5 6
 7  8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

b. Calendar for month and year of birthday [mech @labi21 ~]$ **cal 7 1986**

```
     July 1986
Su Mo Tu We Th Fr Sa
          1 2 3 4 5
 6 7   8 9 10 1112
13 14 15 16 17 18 19
```

```
20 21 22 23 24 25 26
27 28 29 30 31
```

c. Calendar for entire year [mech @labi21 ~]**$ cal -y**

10. **$echo**

NAME
  echo - display a line of text

SYNTAX

  echo [OPTION]... [STRING]...

DESCRIPTION

  Echo the STRING(s) to standard output.

\\	backslash
\a	alert (BEL)
\b	backspace
\c	suppress trailing newline
\f	form feed
\n	new line
\r	carriage return
\t	horizontal tab
\v	vertical tab

OUTPUT

a. Print your name
[mech @labi21 ~]**$ echo statement**
statement

b. Print name of your institution [mech @labi21 ~]**$ echo SRIT** SRIT

c. $HOME $PATH
[mech @labi21 ~]**$ echo $HOME**
/home/civil
[mech @labi21 ~]**$ echo $PATH**

/usr/kerberos/bin:/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/home/ mech /bin

11. **$exit**

exit [n]

Cause the shell to exit with a status of n. If n is omitted, the exit status is that of the last command executed. A trap on EXIT is executed before the shell terminates.

<u>RESULT:</u>

Thus the program was executed successfully

Ex. No: 2b

PART - B

Working with files and directories.

1. **$pwd**

NAME
pwd - print name of current/working directory

SYNTAX

pwd [OPTION]

DESCRIPTION
Print the full filename of the current working directory.

OUTPUT

[mech @labi21 ~]**$ pwd**
/home/mech

2. $cd

cd [-L|-P] [dir]
Change the current directory to dir. The variable HOME is the default dir.

a. **change to home directory** [mech @labi21 ~]**$ cd ~** [mech @labi21 ~]$ pwd /home/ mech

b. **Change to root directory** [mech l@labi21 ~]**$ cd /** [mech @labi21 /]$ pwd
/

ci. Change to home folder using absolute path [mech @labi21 /]**$ cd /home**
[mech @labi21 home]$
pwd /home

d. **Go up one directory level** [mech @labi21 home]**$ cd ..**
[mech @labi21 /]$ pwd

/

e. **change to home folder using relative path** [mech @labi21 /]$ home/ eee
[mech @labi21 ~]$
[mech @labi21 ~]$
pwd /home/mech

f. go up two levels

[mech @labi21 ~]**$ cd ../..**
[mech @labi21 /]$ pwd

3. **$mkdir**

NAME

mkdir - make directories

SYNTAX

mkdir [OPTION] DIRECTORY...

DESCRIPTION

Create the DIRECTORY(ies), if they do not already exist.

OPTIONS

-m, --mode=MODE

set permission mode (as in chmod), not rwxrwxrwx - umask

-p, --parents

no error if existing, make parent directories as needed

OUTPUT

a. Create folder with your name [mech @labi21 /]$
cd ~ [mech @labi21 ~]**$**
mkdir abc [mech @labi21
~]$ ls -d p*

pattern.c pattern.sh pp.c pp.sh pp.sh~ pr practise.c abc

b. Create multiple levels of directory

[mech @labi21 ~]**$ mkdir -p dir1/dir2/dir3**

```
[mech @labi21 ~]$ ls dir1
dir2
[mech @labi21 ~]$ ls dir1/dir2

dir3
```

4. $rmdir

NAME

rmdir - remove empty directories

SYNTAX

rmdir [OPTION]... DIRECTORY...

DESCRIPTION

Remove the DIRECTORY(ies), if they are empty.

OPTIONS

-p, --parents

remove DIRECTORY, then try to remove each directory component of that path name.
E.g., ‘rmdir -p a/b/c’

OUTPUT

a. remove dir3

```
[mech @labi21 ~]$ rmdir dir1/dir2/dir3
[mech @labi21 ~]$ ls dir1/dir2/dir3
ls: dir1/dir2/dir3: No such file or directory
```

b. Remove dir1 and dir 2

```
[mech @labi21 ~]$ rmdir -p dir1/dir2
[mech @labi21 ~]$ ls dir1
ls: dir1: No such file or directory
```

5. $rm

NAME

rm - remove files or directories

SYNTAX

    rm [OPTION]... FILE...

DESCRIPTION

This manual page documents the GNU version of rm. rm removes each specified file. By default, it does not remove directories.

OPTIONS

    -f, --force
        ignore nonexistent files, never prompt

    -i, --interactive

        prompt before any removal

OUTPUT

a. create files f1, f2, f3
[mech @labi21 dir1]**$ touch f1 f2 f3**
[mech @labi21 dir1]$ ls

f1 f2 f3

b. Remove file f1

[mech @labi21 dir1]**$ rm f1**
[mech l@labi21 dir1]$ ls
f2 f3
c. remove file f2 with prompt [mech @labi21 dir1]**$ rm -i f2**
rm: remove regular empty file `f2'? y [mech @labi21 dir1]$ ls

f3
di. remove file f3 forcibly
[mech @labi21 dir1]**$ rm -f f3** [mech @labi21 dir1]$ ls
[mech l@labi21 dir1]$
6. $touch

NAME

    touch - change file timestamps, also used to create empty files

SYNTAX

touch [OPTION]... FILE...

DESCRIPTION

Update the access and modification times of each FILE to the current time.

OUTPUT

a. navigate to folder with your name [mech @labi21 p]**$ cd ~**

[mech @labi21 ~]**$ cd p** [mech @labi21 p]$

b. create 3 empty files s1, s2, s3 [mech @labi21 p]**$ touch s2 s3 s4**

[mech @labi21 p]$ ls s2 s3 s4

7. **$cat**

NAME

cat - concatenate files and print on the standard output

SYNTAX

cat [OPTION] [FILE]...

DESCRIPTION

Concatenate FILE(s), or standard input, to standard output.

OUTPUT

a. create a file and add some text [mech @labi21 p]$ **cat > s1**

Hi there!
[mech l@labi21 p]$

b. Display contents of file [mech l@labi21 p]**$ cat s1** Hi there!

ci. append additional content

[mech @labi21 p]**$ cat >> s1**
Greetings
[mech @labi21 p]$ cat s1
Hi there!
Greetings

ci. to concatenate two files
[mech @labi21 p]**$ cat > s2**

GoodBye...
[mech @labi21 p]**$ cat s1 s2**
Hi there!
Greetings

GoodBye...

$cp

NAME
  cp - copy files and directories

SYNTAX

```
cp [OPTION]... SOURCE DEST
cp [OPTION]... SOURCE... DIRECTORY
cp [OPTION]... --target-directory=DIRECTORY SOURCE...
```

DESCRIPTION

  Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.

OPTIONS
    -i, --interactive
        prompt before overwrite

OUTPUT

a. Coy contents from file
s1 to s3 [mech @labi21 p**]$ cp**
s1 s3

[mech @labi21 p]$ cat
s3 Hi there!
Greetings
b. copy contents with
prompt [mech @labi21 p**]$**
cp -i s3 s2 cp: overwrite
`s2'? y
[mech @labi21 p]$ cat
s2 Hi there!

Greetings

$mv

SYNTAX
    mv - move (rename) files

SYNOPSIS

    mv [OPTION]... SOURCE DEST
    mv [OPTION]... SOURCE... DIRECTORY
    mv [OPTION]... --target-directory=DIRECTORY SOURCE...

DESCRIPTION

    Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY.

a. rename s1 to file1
[mech @labi21 p**]$ mv s1 file1**
[mech @labi21p]$ ls
file1 s2 s3

b. create directory r1 and remame directory
r1 to dir1 [mech @labi21 p]$ mkdir r1

```
[mech @labi21 p]$
ls file1 r1 s2 s3

[mech @labi21 p]$ mv r1 dir1
[mech @labi21 p]$
ls dir1 file1 s2 s3
```

c. move files s2 and s3 to dir1

```
[mech @labi21 p]$ mv s2 s3 dir1
[mech @labi21 p]$ ls
dir1 file1
[mech @labi21 p]$ ls dir1

s2 s3
```

$ls

NAME
    ls - list directory contents

SYNTAX

    ls [OPTION]... [FILE]...

DESCRIPTION

    List information about the FILEs
OPTIONS
    -a, --all
        do not hide entries starting with .
    -l   use a long listing format.
    -r, --reverse
        reverse order while sorting

OUTPUT

a. create a folder named 'list' [mech @labi21 p]$ mkdir list
[mech @labi21 p]$ cd list

b. create the following files

[mech @labi21 list**]$ touch r1 r2 r3 one eight rock pepper salt malt**

c. display the files.
[mech @labi21 list**]$ ls**
eight malt one pepper r1 r2 r3 rock salt

Metacharacters

a. List the file names that begin with r [mech @labi21 list**]$ ls r***

r1 r2 r3 rock

b. list the file names that end with alt and any one beginning character.[mech @labi21 list**]$ ls ?alt**

malt salt

c. list the files that begin with a vowel [mech @labi21 list**]$ ls [aeiou]*** eight one

d. list the files that dosn't beging with a 'r' or 'm' [mech @labi21 list]**$ ls [!rm]***

eight one pepper salt

e. list the files in reverse alphabetical order. [mech @labi21 list**]$ ls -r**

salt rock r3 r2 r1 pepper one malt eight

f. Create a hidden file. Use -a to display a hidden file [mech @labi21 list]**$ touch .hidefile**

```
[mech @labi21 list]$ ls
eight malt one pepper r1 r2 r3 rock salt
[mech @labi21 list]$ ls -a

. .. eight .hidefile malt one pepper r1 r2 r3 rock salt
```

RESULT:

Thus the program was executed successfully

Ex.No: 2C

SHELL PROGRAMMING
SIMULATION OF UNIX COMMANDS “grep”

AIM: To write a C program to simulate the operation of “grep” commands in Unix.

ALGORITHM :

Step 1 : Include the necessary header files.
Step 2 : Define the buffer size.
Step 3 : Get the file name which has been created already.
Step 4 : Open the file in read mode.
Step 5 :.Read the contents of the file and store it in the buffer.
Step 6 : Print the contents stored in the buffer.
Step 7 : Close the file.

PROGRAM CODING:

```
#include<stdio.h>
#include<stdlib.h>
main()
{
FILE *fp;
char c[100],pat[10];

int l,i,j=0,count=0,len,k,flag=0;
printf("\n enter the pattern");
scanf("%s",pat); len=strlen(pat);
fp=fopen("nn.txt","r");
while(!feof(fp))
{
fscanf(fp,"%s",c);
l=strlen(c);
count++;
for(i=0;i<count;i++)
{
if(c[i]==pat[j])
{
flag=0;
for(k=1;k<i;k++)
{
if(c[i+k]!=pat[k])
flag=1;
}
if(flag==0)
printf("\n the pattern %s is present in word %d",pat,count);
}}}}
```

OUTPUT:

```
    Enter the pattern: h
The pattern 'h' is present in word 2
The pattern 'h' is present in word 4
The pattern 'h' is present in word 6
```

RESULT:

Thus the program was executed successfully

BASIC ARITHMETIC OPERATION USIG SHELL PROGRAMMING

AIM: To write a shell program to solve arithmetic operation.

ALGORITHM :

Step 1 : Include the necessary header files.
Step 2 : get the input
Step 3 : perform the arithmetic calculation.
Step 4 : print the result.
Step 5 :stop the execution.

PROGRAM CODING:

```
#!/bin/bash

echo  "enter the a vale"

read  a

echo  "enter b value"

read  b

c=`expr  $a  +  $b`

echo  "sum:"$c

c=`expr  $a  -  $b`

echo  "sub:"$c

c=`expr  $a  \*  $b`

echo  "mul:"$c

c=`expr  $a  /  $b`

echo  "div:"$c
```

OUTPUT:

```
[2mecse25@rhes3linux ~]$ sh pg2.sh
Enter the a vale 10
Enter b value 50
sum:60
sub:-40
mul:500
div:0
```

RESULT:

Thus the program was executed successfully

NUMBER CHECKING USING SHELL PROGRAM

AIM: To write a shell program to check whether the number is odd or even.

ALGORITHM :

```
Step 1 : Include the necessary header files.
Step 2 : get the input
Step 3 : perform the division by 2.
Step 4 : print the result.
Step 5 :stop the execution.
```

PROGRAM CODING:

```
#!/bin/bash

num="1 2 3 4 5 6 7 8"
for n in $num

do

q=`expr $n % 2`
if [ $q -eq 0 ] then
echo "even no"
continue

fi

echo "odd no"
done
```

OUTPUT:

```
[2mecse25@rhes3linux ~]$ sh pg2.sh
odd no

even    no
odd     no
even    no
odd     no
even    no
odd     no
even no
```

RESULT:

Thus the program was executed successfull.

MULTIPLICATION TABLE USING SHELL PROGRAM

AIM: To write a shell program to check whether the number is odd or even.

ALGORITHM :

```
Step 1 : Include the necessary header files.
Step 2 : get the input
Step 3 : perform the multiplication .
Step 4 : print the result.
Step 5 :stop the execution.
```

PROGRAM CODING:

```
#!/bin/bash

echo " which table you want" read n

for((i=1;i<10;i++)) do
echo $i "*" $n "=" `expr $i \* $n`
done
```

OUTPUT:

```
[2mecse25@rhes3linux ~]$ sh pg2.sh which
table you want 5
1 * 5 = 5

2 * 5 = 10

3 * 5 = 15

4 * 5 = 20

5 * 5 = 25

6 * 5 = 30

7 * 5 = 35

8 * 5 = 40

9 * 5 = 45
```

RESULT:

Thus the program was executed successfully

USING WHILE LOOP IN SHELL PROGRAM

AIM: To write a shell program to print the number from 1 to 10 using while loop.

ALGORITHM :

```
Step 1 : Include the necessary header files.
Step 2 : Assign values to the variables
Step 3 : Initialize the loop.
Step 4 : print the result.
Step 5 :stop the execution.
```

<u>PROGRAM CODING</u>:

```
#!/bin/bash

a=1

while [ $a -lt 11 ] do

echo "$a" a=`expr $a
+ 1` done
```

OUTPUT:

```
[2mecse25@rhes3linux ~]$ sh pg2.sh 1 2
3 4 5 6 7 8 9 10
```

<u>RESULT:\</u>

Thus the program was executed successfully

BIGGEST OF THREE NUMBERS

AIM: To write a shell program to find the greatest number.

ALGORITHM :

Step 1 : Include the necessary header files.
Step 2 : Get the inputs
Step 3 : check the conditions
Step 4 : print the result.
Step 5 :stop the execution.

PROGRAM CODING:

```
#!/bin/bash

echo "enter the first number"
read a
echo "enter the second number"
read b
echo "enter the third number"
read c
if [ $a -gt $b -a $a -gt $c ]
then
echo "$a is greater"
elif [ $b -gt $c ]
then
echo "$b is greater"
else
echo "$c is greater"
fi
```

OUTPUT:

```
[09cse@server ~]$ vi greatest.sh
[09cse@server ~]$ sh greatest.sh
enter the first number
1
enter the second number
2
enter the third number
3
3 is greater
```

RESULT:

Thus the program was executed successfully

Ex. No: 3 **SYSTEM CALLS**

i) PROGRAM USING SYSTEM CALL fork()

AIM : To write the program to create a Child Process using system call fork().

ALGORITHM :

```
Step 1 : Declare the variable pid.

Step 2 : Get the pid value using system call fork().
Step 3 : If pid value is less than zero then print as "Fork failed".

Step 4 : Else if pid value is equal to zero include the new process in the
         system‟s file using execlp system call.
Step 5 : Else if pid is greater than zero then it is the parent

         process and it waits till the child completes using the system call wait()
Step 6 : Then print "Child complete".
```

<u>SYSTEM CALLS USED</u>:

1. fork( )

Used to create new processes. The new process consistsof a copy of the address space of the original process. The value of process id for the child process is zero, whereas the value of process id for the parent is an integer value greater than zero.
Syntax : fork( )

2.execlp( )

Used after the fork() system call by one of the two processes to replace the process‟ memory space with a new program. It loads a binary file into memory destroying the memory image of the program containing the execlp system call and starts its execution.The child process overlays its address space with the UNIX command /bin/ls using the execlp system call.

Syntax : execlp( )

3. wait( )

The parent waits for the child process to complete using the wait system call. The wait system call returns the process identifier of a terminated child, so that the parent can tell which of its possibly many children has terminated.

Syntax : wait(NULL)

4. exit( )

A process terminates when it finishes executing its final statement and asks the operating system to delete it by using the exit system call. At that point, the process may return data (output) to its parent process (via the wait system call)
Syntax: exit(0)

<u>PROGRAM CODING :</u>

```
#include<stdio.h>

#include<stdlib.h>
```

```
#include<unistd.h>
void main(int argc,char *arg[])
{
    int pid;


     pid=fork();
     if(pid<0)
     {

     printf("fork  failed"); exit(1);

     }
     else if(pid==0)
     {
     execlp("whoami","ls",NULL);
     exit(0);
     }
     else
     {

     printf("\n Process id is
     -d\n",getpid()); wait(NULL);

     exit(0);
     }
     }
```

OUTPUT:

```
 [cse6@localhost Pgm]$ cc prog4a.c

 [cse6@localhost Pgm]$ ./a.out
```

RESULT:

Thus the program was executed and verified successfully

ii) PROGRAM USING SYSTEM CALLS getpid() & getppid()

AIM :

To write the program to implement the system calls getpid() and getppid().

ALGORITHM :

```
Step 1 : Declare the variables pid , parent pid , child id and grand chil id.
Step 2 : Get the child id value using system call fork().
Step 3 : If child id value is less than zero then print as “error at fork() child”.

Step 4 : If child id !=0 then using getpid() system call get the process id.
Step 5 : Print “I am parent” and print the process id.
Step 6 : Get the grand child id value using system call fork().

Step 7 : If the grand child id value is less than zero then print as “error at fork() grand
         child”.

Step 8 : If the grand child id !=0 then using getpid system call get the process id.
Step 9 : Assign the value of pid to my pid.
Step 10 : Print “I am child” and print the value of my pid.
Step 11 : Get my parent pid value using system call getppid().

Step 12 : Print “My parent‟s process id” and its value.
Step 13 : Else print “I am the grand child”.

Step 14 : Get the grand child‟s process id using getpid() and print it as “my process id”. Step
15 : Get the grand child‟s parent process id using getppid() and print it as “my parent‟s
process id
```

SYSTEM CALLS USED :

1.getpid( )

Each process is identified by its id value**.** This function is used to get the id value of a particular process.

2.getppid( )

Used to get particular process parent‟s id value.

3.perror( )

Indicate the process error

PROGRAM CODING:

```
#include<stdio.h>

#include<unistd.h>

#include<stdlib.h> int
main( )
{

int pid;

 pid=fork( );
 if(pid== 1)
```

```
   {

 perror(“fork failed”); exit(0);
 }
if(pid==0)
{
printf(“\n Child process is under execution”);

printf(“\n Process id of the child process is %d”, getpid()); printf(“\n Process id of the
parent process is %d”, getppid());
}
else
{
printf(“\n Parent process is under execution”);

printf(“\n Process id of the parent process is %d”, getpid()); printf(“\n Process id
of the child process in parent is %d”, pid()); printf(“\n Process id of the parent of
parent is %d”, getppid());
}

return(0);
}
```

OUTPUT:

```
Child process is under execution Process id of the child
process is 9314

Process id of the parent process is 9313 Parent process is under
execution
Process id of the parent process is 9313

Process id of the child process in parent is 9314 Process id of the parent
of parent is 2825
```

RESULT:

Thus the program was executed and verified successfully

iii)PROGRAM USING SYSTEM CALLS opendir( ) readdir( ) closedir()

AIM :
To write the program to implement the system calls opendir( ), readdir( ).

ALGORITHM :

```
Step 1 : Start.
Step 2 : In the main function pass the arguments.

Step 3 : Create structure as stat buff and the variables as integer.
Step 4 : Using the for loop,initialization
```

<u>SYSTEM CALLS USED</u>:

1.opendir( )
Open a directory.

2.readdir( )
Read a directory.

3.closedir()
Close a directory.

<u>PROGRAM CODING:</u>

```
#include<stdio.h>

#include<sys/types.h>
#include<sys/dir.h>
void main(int age,char *argv[])

{
DIR *dir;
struct dirent *rddir;

printf("\n Listing the directory content\n");
dir=opendir(argv[1]);
while((rddir=readdir(dir))!=NULL)
{
printf("%s\t\n",rddir->d_name);
}
closedir(dir);
}
```

<u>OUTPUT:</u>

```
RP roshi.c
first.c
pk6.c f2

abc FILE1
```

<u>RESULT:</u>
Thus the program was executed and verified successfully

iv) PROGRAM USING SYSTEM CALL exec( )

AIM : To write the program to implement the system call exec( ).

ALGORITHM :

Step 1 : Include the necessary header files.

Step 2 : Print execution of exec system call for the date Unix command.

Step 3 : Execute the execlp function using the appropriate syntax for the Unix command date.

Step 4 : The system date is displayed.

SYSTEM CALL USED :

1.execlp( )

Used after the fork() system call by one of the two processes to replace the process" memory space with a new program. It loads a binary file into memory destroying the memory image of the program containing the execlp system call and starts its execution. The child process overlays its address space with the UNIX command /bin/ls using the execlp system call.

Syntax : execlp( )

PROGRAM CODING:

```
#include<stdio.h>
#include<unistd.h>
main( )

{

printf("\n exec system call");
printf("displaying the date"); execlp(
"/bin/date", "date", 0);
}
```

OUTPUT:

```
Sat Dec 14 02 : 57 : 38  IST 2010
```

RESULT:

Thus the program was executed and verified successfully.

v) PROGRAM USING SYSTEM CALLS open( ), read() & write( )

AIM : To write the program to implement the system calls open( ),read( ) and write( ).

ALGORITHM :

```
Step 1 : Declare the structure elements.

Step 2 : Create a temporary file named temp1.
Step 3 : Open the file named "test" in a write mode.
Step 4 : Enter the strings for the file.
Step 5 : Write those strings in the file named "test".
Step 6 : Create a temporary file named temp2.
Step 7 : Open the file named "test" in a read mode.
Step 8 : Read those strings present in the file "test" and save it in temp2.

Step 9 : Print the strings which are read.
```

SYSTEM CALLS USED :

1. **open( )**
2. **read( )**
3. **write( )**
4. **close( )**
5. **gets( )**
6. **lseek( )**

PROGRAM CODING:

```
#include<stdio.h>
#include<unistd.h>
#include<string.h>

#include<fcntl.h> main(
)
{
int fd[2];

char buf1[25]= ”just a test\n”; char
buf2[50];

fd[0]=open(“file1”, O_RDWR);
fd[1]=open(“file2”, O_RDWR);
write(fd[0], buf1, strlen(buf1));
printf(“\n Enter the text now….”);
gets(buf1);

write(fd[0], buf1, strlen(buf1));
lseek(fd[0], SEEK_SET, 0);
read(fd[0], buf2, sizeof(buf1));
write(fd[1], buf2, sizeof(buf2));
close(fd[0]);

close(fd[1]);
printf(“\n”);
return0
}
```

OUTPUT:

```
Enter the text now….progress

Cat file1 Just a test progress

Cat file2 Just a test progress
```

RESULT:

Thus the program was executed successfully

CPU SCHEDULING

A. FIRST COME FIRST SERVE SCHEDULING

AIM:

To write the program to implement CPU & scheduling algorithm for first come first serve scheduling.

ALGORITHM:

1. Start the program.
2. Get the number of processes and their burst time.
3. Initialize the waiting time for process 1 and 0.
4. Process for(i=2;i<=n;i++),wt.p[i]=p[i-1]+bt.p[i-1].
5. The waiting time of all the processes is summed then average value time is calculated.
6. The waiting time of each process and average times are displayed
7. Stop the program

PROGRAM :(FIRST COME FIRST SERVE SCHEDULING)

```
#include<stdio.h>
struct process
{
int pid;
int bt;
int wt,tt;
}p[10];
int main()
{
int i,n,totwt,tottt,avg1,avg2;   clrscr();
printf("enter the no of process \n"); scanf("%d",&n);
for(i=1;i<=n;i++)
{
p[i].pid=i;
printf("enter the burst time n"); scanf("%d",&p[i].bt);

} p[1].wt=0;
p[1].tt=p[1].bt+p[1].wt;
i=2; while(i<=n)

{

p[i].wt=p[i-1].bt+p[i-1].wt;
p[i].tt=p[i].bt+p[i].wt; i ++;
} i=1; totwt=tottt=0;

printf("\n processid \t bt\t wt\t tt\n");
while(i<=n){ printf("\n\t%d \t%d \t%d
\t%d",p[i].pid,p[i].bt,p[i].wt,p[i].tt);
totwt=p[i].wt+totwt; tottt=p[i].tt+tottt; i++;}

avg1=totwt/n; avg2=tottt/n; printf("\navg1=%d  \t avg2=%d\t",avg1,avg2);
getch();
return 0;

}
```

OUTPUT:

```
enter the no of process5

enter the burst time n2

enter the burst time n1

enter the burst time n8

enter the burst time n4

enter the burst time n5

  processid      bt      wt      tt
       1            2       0        2


       2            1       2        3
       3            8       3       11
       4            4       11      15
       5            5       15      20
avg1=6   avg2=10
```

RESULT:

Thus the FIFO process scheduling program was executed and verified successfully.

B.SHORTEST JOB FIRST SCHEDULING

AIM:

To write a program to implement cpu scheduling algorithm for shortest job first scheduling.

ALGORITHM:

1. Start the program. Get the number of processes and their burst time.
2. Initialize the waiting time for process 1 as 0.
3. The processes are stored according to their burst time.
4. The waiting time for the processes are calculated a follows: for(i=2;i<=n;i++).wt.p[i]=p[i=1]+bt.p[i-1].
5. The waiting time of all the processes summed and then the average time is calculate
6. The waiting time of each processes and average time are displayed.
7. Stop the program.

PROGRAM: (SHORTEST JOB FIRST SCHEDULING)

```
#include<stdio.h>
#include<conio.h>
struct process
{
int pid;
int bt; int wt; int tt;
}p[10],temp;
int main()
{

int i,j,n,totwt,tottt; float avg1,avg2; clrscr();
printf("\nEnter the number of process:\t");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
p[i].pid=i;
printf("\nEnter the burst time:\t");

scanf("%d",&p[i].bt);
}
for(i=1;i<n;i++){
for(j=i+1;j<=n;j++)
{
if(p[i].bt>p[j].bt)
{
  temp.pid=p[i].pid;
  p[i].pid=p[j].pid;
  p[j].pid=temp.pid
  temp.bt=p[i].bt;p[i].bt=p[j].bt;
p[j].bt=temp.bt;
}}}
p[1].wt=0; p[1].tt=p[1].bt+p[1].wt; i=2;
while(i<=n){
p[i].wt=p[i-1].bt+p[i-1].wt;
p[i].tt=p[i].bt+p[i].wt;
i++;
}
i=1;
totwt=tottt=0;
printf("\nProcess  id \tbt \twt \ttt");
while(i<=n){

printf("\n\t%d \t%d \t%d
t%d\n",p[i].pid,p[i].bt,p[i].wt,p[i].tt); totwt=p[i].wt+totwt;
tottt=p[i].tt+tottt; i++;

}
avg1=totw
t/n;
avg2=tottt/
n;

printf("\nAVG1=%f\t
AVG2=%f",avg1,avg2); getch();

return 0; }
```

OUTPUT:

```
enter the number of          3

enter the burst time: 2

enter the burst time: 4

enter the burst time: 6

processid       bt      wt    tt

1               2       0     2

2               4       2     6

3               6       6     12

AVG1=2.000000        AVG2=6.000000
```

RESULT:

Thus the SJF program was executed and verified successfully

PRIORITY SCHEDULING

AIM:

To write a C program to perform priority scheduling.

ALGORITHM:

1. Start the program.
2. Read burst time, waiting time, turn the around time and priority.
3. Initialize the waiting time for process 1 and 0.
4. Based up on the priority process are arranged
5. The waiting time of all the processes is summed and then the average waiting time
6. The waiting time of each process and average waiting time are displayed based on the priority.
7. Stop the program.

PROGRAM: (PRIORITY SCHEDULING)

```
#include<stdio.h>
#include<conio.h>
struct process
{
int pid; int bt; int wt; int tt;
int prior;
}
p[10],temp;
int main()
{
int i,j,n,totwt,tottt,arg1,arg2;
clrscr();
printf("enter the number of process");
scanf("%d",&n);
for(i=1;i<=n;i++)
{

p[i].pid=i;

printf("enter the burst time");
scanf("%d",&p[i].bt);
printf("\n enter the priority");
scanf("%d",&p[i].prior);
}
for(i=1;i<n;i++)
{
for(j=i+1;j<=n;j++)
```

```
  {
  if(p[i].prior>p[j].prior)
  {
temp.pid=p[i].pid;
p[i].pid=p[j].pid; p[j].pid=temp.pid; temp.bt=p[i].bt; p[i].bt=p[j].bt; p[j].bt=temp.bt;
temp.prior=p[i].prior;
p[i].prior=p[j].prior;
p[j].prior=temp.prior;
}
}

} p[i].wt=0; p[1].tt=p[1].bt+p[1].wt;
i=2; while(i<=n)

{

p[i].wt=p[i-1].bt+p[i-1]
.wt;
p[i].tt=p[i].bt+p[i].wt;
i++;

}
i=1;
totwt=tottt=0;

printf("\n process to \t bt \t wt \t
tt"); while(i<=n)

{

printf("\n%d\t %d\t %d\t
%d\t",p[i].pid,p[i].bt,p[i].wt,p[i].tt); totwt=p[i].wt+totwt;

tottt=p[i].tt+tottt;
i++;
}
arg1=totwt/n;
arg2=tottt/n;

printf("\n arg1=%d \t
arg2=%d\t",arg1,arg2); getch();

return 0;
}
```

OUTPUT:

```
enter the no of process:3
enter the burst time:2
enter the priority:3
enter the burst time:4
enter the priority:1
enter the burst time:6
enter the priority:2

  process    bt       pr      wt       tt
     1        4        1       4        4

     2        6        2       10       14

     3        2        3       12       22

       avg1=4              avg2=8
```

RESULT:

Thus the priority scheduling program was executed and verified successfully

D. ROUND ROBIN SCHEDULING

AIM:

To write a program to implement cpu scheduling for Round Robin Scheduling.

ALGORITHM:

1. Get the number of process and their burst time.
2. Initialize the array for Round Robin circular queue as „0‟.
3. The burst time of each process is divided and the quotients are stored on the round Robin array.
4. According to the array value the waiting time for each process and the average time are calculated as line the other scheduling.
   a. The waiting time for each process and average times are displayed.
   b. Stop the program.

PROGRAM :(ROUND ROBIN SCHEDULING)

```
#include<stdio.h>

main()

{

int
pt[10][10],a[10][10],at[10],pname[10][10],i,j,n,k=0,q,sum=0
; float avg;
printf("\n\n Enter the number of processes : ");

scanf("%d",&n);

for(i=0;i<10;i++)

{

for(j=0;j<10;j++)

{

 pt[i][j]=0;

 a[i][j]=0;

}

}
```

```
     for(i=0;i<n;i++)
     {
     j=0;
     printf("\n\n Enter the process time for process %d : ",i+1);
     scanf("%d",&pt[i][j]);
     }
     printf("\n\n Enter the time slice : ");
     scanf("%d",&q);
     printf("\n\n");
for(j=0;j<10;j++)
{
for(i=0;i<n;i++)
{
 a[2*j][i]=k;
 if((pt[i][j]<=q)&&(pt[i][j]!=0))
{
pt[i][j+1]=0;
printf(" %d P%d %d\n",k,i+1,k+pt[i][j]);
k+=pt[i][j];
a[2*j+1][i]=k;
}
else if(pt[i][j]!=0)
{
 pt[i][j+1]=pt[i][j]-q;
 printf(" %d P%d %d\n",k,i+1,(k+q));
 k+=q;

 a[2*j+1][i]=k;
}
else
{
 a[2*j][i]=0;
 a[2*j+1][i]=0;
}
}
}
for(i=0;i<n;i++)
sum+=a[0][i];
for(i=0;i<n;i++)
{
for(j=1;j<10;j++)
{
if((a[j][i]!=0)&&(a[j+1][i]!=0)&&((j+1)%2==0))
sum+=((a[j+1][i]-a[j][i]));
}
}
avg=(float)sum/n;
printf("\n\n Average waiting time = %f msec",avg);
sum=avg=0;
for(j=0;j<n;j++)
{
 i=1;
 while(a[i][j]!=0)
 i+=1;
 sum+=a[i-1][j];

}

avg=(float)sum/n;

printf("\n\n Average turnaround time = %f msec\n\n",avg);
}
```

OUTPUT:

```
Enter the no of process:3
Enter the process time for process1:24
Enter the process time for process2:3
Enter the process time for process3:3

Enter the time slice: 4

                              0 P1 4
                              4 P2 7
                              7 P3 10
                              10 P1 14
                              14 P1 18
                              18 P1 22
                              22 P1 26
                              26 P1 30

Average waiting time: 5.666666

Average turnaround time: 15.666666
```

RESULT:

Thus the Round Robin scheduling program was executed and verified successfully.

Ex.No: 5

INTER PROCESS COMMUNICATION USING SHARED MEMORY

AIM:

To write a program for interprocess communication using shared memory.

ALGORITHM:

1. Start the program
2. Create the child process using fork()
3. Create the shared memory for parent process using shmget() system call
4. Now allow the parent process to write inn shared memory using shmpet pointer which is return type of shmat()
5. Now across and attach the same shared memory to the child process
6. The data in the shared memory is read by the child process using the shnot pointer
7. Now, detach and rebase the shared memory
8. Stop the program

PROGRAM:

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/shm.h>
#include<sys/ipc.h>
int main()
{
        int child,shmid,i;
        char * shmptr;
        child=fork();
        if(!child)
        {
                shmid=shmget(2041,32,IPC_CREAT|0666);
                shmptr=shmat(shmid,0,0);
                printf("\n Parent writing\n");
                for(i=0;i<10;i++)
                {
                        shmptr[i]='a'+i;
                        putchar(shmptr[i]);
                }
                printf("\n\n %s", shmptr);
                wait(NULL);
        }
        else
        {
                shmid=shmget(2041,32,0666);
                shmptr=shmat(shmid,0,0);
                printf("\n Child is reading\n");
                for(i=0;i<10;i++)
                        putchar(shmptr[i]);
                shmdt(NULL);
                shmctl(shmid,IPC_RMID,NULL);
        }
        return 0;
```

OUTPUT:

```
[cse2@localhost ~]$ cc share.c
[cse2@localhost ~]$ ./a.out

Parent writing
        abcdefghij
Child is reading
        abcdefghij
```

Result:

Thus the inter process communication using shared memory was successfully executed.

Ex.No: 6

PRODUCER-CONSUMER PROBLEM USING SEMOPHERES

AIM:

To implement producer/consumer problem using semaphore.

ALGORITHM:

1. Declare variable for producer & consumer as pthread-t-tid produce tid consume.
2. Declare a structure to add items, semaphore variable set as struct.
3. Read number the items to be produced and consumed.
4. Declare and define semaphore function for creation and destroy.
5. Define producer function.
6. Define consumer function.
7. Call producer and consumer.
8. Stop the execution.

PROGRAM: (PRODUCER-CONSUMER PROBLEM)

```
#include<stdio.h>
void main()
{
int buffer[10], bufsize, in, out, produce, consume, choice=0;
in = 0; out = 0; bufsize = 10;
while(choice !=3)
{
printf("\n1. Produce  \t 2. Consume \t3. Exit");
printf("\nEnter  your choice:  =");
scanf("%d", &choice);
switch(choice)
{
case 1:    if((in+1)%bufsize==out)
printf("\nBuffer  is Full");
else

{
printf("\nEnter the value: "); scanf("%d", &produce); buffer[in] = produce;
in = (in+1)%bufsize;
}
break;
case 2:    if(in == out)
printf("\nBuffer  is Empty");
else
{
```

```
consume = buffer[out];
printf("\nThe consumed value is %d", consume);
out = (out+1)%bufsize;
}
break;
}  }    }
```

OUTPUT:

```
1. Produce          2. Consume          3. Exit
Enter your choice: 2
Buffer is Empty
1. Produce          2. Consume          3. Exit
Enter your choice: 1
Enter the value: 100
1. Produce          2. Consume          3. Exit
Enter your choice: 2
The consumed value is 100
1. Produce          2. Consume          3. Exit
Enter your choice: 3
```

RESULT:

Thus the producer consumer problem is implemented successfully.

Ex.No: 7

SIMULATE BANKERS ALGORITHM FOR DEADLOCK AVOIDENCE

AIM:

To write a c program to implement the Bankers algorithm.

ALGORITHM:

1. Start the program
2. Attacking Mutex condition: never grant exclusive access. But this may not be possible for several resources.
3. Attacking preemption: not something you want to do.
4. Attacking hold and wait condition: make a process hold at the most 1 resource
5. At a time. Make all the requests at the beginning. Nothing policy. If you feel, retry.
6. Attacking circular wait: Order all the resources. Make sure that the requests are issued in the
7. Correct order so that there are no cycles present in the resource graph. Resources numbered 1 ... n.
8. Resources can be requested only in increasing
9. Order. i.e. you cannot request a resource whose no is less than any you may be holding.
10. Stop the program

PROGRAM: **(SIMULATE ALGORITHM FOR DEADLOCK PREVENTION)**

```
#include<stdio.h>
#include<conio.h>
void main() {
int  k=0, output[10],d=0,t=0,ins[5],i,avail[5],allocated[10][5],need[10][5],MAX[10][5],pno,
P[10],j,rz, count=0;
clrscr();
printf("\n Enter the number of resources : ");
scanf("%d", &rz);

printf("\n enter the max instances of each resources\n");
for (i=0;i<rz;i++) {
avail[i]=0;
printf("%c= ",(i+97));
scanf("%d",&ins[i]);
}
printf("\n Enter the number of processes : ");
```

```
     scanf("%d", &pno);
     printf("\n Enter the allocation matrix \n      ");
     for (i=0;i<rz;i++)
     printf(" %c",(i+97));
printf("\n");
for (i=0;i <pno;i++) {
P[i]=i;
printf("P[%d]  ",P[i]);
for (j=0;j<rz;j++) {
scanf("%d",&allocated[i][j]);
avail[j]+=allocated[i][j];
}
}
printf("\nEnter the MAX matrix \n      ");
for (i=0;i<rz;i++) {
printf(" %c",(i+97));
avail[i]=ins[i]-avail[i];
}
printf("\n");
for (i=0;i <pno;i++) {

printf("P[%d]  ",i);
for (j=0;j<rz;j++)
scanf("%d", &MAX[i][j]);
}
printf("\n");
A: d=-1;
for (i=0;i <pno;i++) {
count=0;
t=P[i];
for (j=0;j<rz;j++) {
need[t][j] = MAX[t][j]-allocated[t][j];
if(need[t][j]<=avail[j])
count++;
}
if(count==rz) {
output[k++]=P[i];
for (j=0;j<rz;j++)
avail[j]+=allocated[t][j];

} else
P[++d]=P[i];

}

if(d!=-1) {
pno=d+1;
goto A;

}

printf("\t <"); for
(i=0;i<k;i++)
```

```
printf(" P[%d] ",output[i]);
printf(">");

getch();
}
```

OUTPUT:

```
Enter the number of resources : 3

enter the max instances of each resources
a= 10
b= 5
c= 7

Enter the number of processes : 5
Enter the allocation matrix
P[0]  a b c
      0 1 0
P[1]  2 0 0
P[2]  3 0 2
P[3]  2 1 1
P[4]  0 0 2
Enter the MAX matrix
P[0]  a b c
      7 5 3
P[1]  3 2 2
P[2]  9 0 2
P[3]  2 2 2
P[4]  4 3 3
< P[1]  P[3]  P[4]  P[0]  P[2] >
```

RESULT:

Thus the program deadlock was executed successfully.

Ex.No: 8

ALGORITHM FOR DEADLOCK DETECTION

AIM:
To write a C program to implement algorithm for deadlock detection.

ALGORITHM:
Step-1: Start the program.
Step-2: Declare the memory for the process.
Step-3: Read the number of process, resources, allocation matrix and available matrix.
Step-4: Compare each and every process using the banker‟s algorithm.
Step-5: If the process is in safe state then it is a not a deadlock process otherwise it is a deadlock process
Step-6: produce the result of state of process
Step-7: Stop the program

PROGRAM:
```
#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("********** Deadlock Detection Algo ************\n");
input();
show();
cal();
getch();
return 0;
}
void input()
{int i,j;
printf("Enter the no of Processes\t");
scanf("%d",&n);
printf("Enter the no of resource 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 dead[100];
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];
}}
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("\nP%d",i);
if(finish[i]==1)
{i=n;
}}}}}}
j=0;
flag=0;
for(i=0;i<n;i++)
{
if(finish[i]==0)
{dead[j]=i;
j++;
flag=1;
}}
if(flag==1)
{
printf("\n\nSystem is in Deadlock and the Deadlock process are\n");
for(i=0;i<n;i++)
{printf("P%d\t",dead[i]);
}}
 else
{
printf("\nNo Deadlock Occur"); }}
```

OUTPUT :

```
          ********** Deadlock Detection Algo ************
Enter the no of Processes   3
Enter the no of resource instances   4
Enter the Max Matrix
4 3 2 1
2 1 3 2
3 2 4 2
Enter the Allocation Matrix
1 0 0 1
1 0 1 2
1 3 0 0
Enter the available Resources
1 1 2 1
Process    Allocation    Max     Available
P1          1 0 0 1      4 3 2 1       1 1 2 1
P2          1 0 1 2      2 1 3 2
P3          1 3 0 0      3 2 4 2

System is in Deadlock and the Deadlock process are
P2   P3
```

<u>**RESULT:**</u>

Thus the program deadlock was executed successfully.

Ex.No: 9 THREADING & SYNCHRONIZATION APPLICATIONS

AIM:

To write a c program to implement Threading and Synchronization Applications.

ALGORITHM:

```
Step 1: Start the process
Step 2: Declare process thread, thread-id.
Step 3: Read the process thread and thread state.
Step 4: Check the process thread equals to thread-id by using if condition.
Step 5: Check the error state of the thread.
Step 6: Display the completed thread process.
Step 7: Stop the process
```

PROGRAM:

```
#include<stdio.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>
pthread_t tid[2];
void* doSomeThing(void *arg)
{
 unsigned long i = 0;
 pthread_t id = pthread_self();
 if(pthread_equal(id,tid[0]))
 {
 printf("\n First thread processing\n");
 }
 else
 {
 printf("\n Second thread processing\n");
 }
 for(i=0; i<(0xFFFFFFFF);i++);
 return NULL;
}
int main(void)
{
 int i = 0;

int err;
 while(i < 2)
 {
 err = pthread_create(&(tid[i]), NULL, &doSomeThing, NULL);
 if (err != 0)
 printf("\ncan't create thread :[%s]", strerror(err));
 else
 printf("\n Thread created successfully\n");
 i++;
 }
 sleep(5);
 return 0;
}
```

OUTPUT :

```
        Thread created successfully
        Thread created successfully
        First thread processing
        Second thread processing
```

<u>RESULT:</u>

Thus the program was successfully executed.

Ex.No: 10

PAGING TECHNIQUE OF MEMORY MANAGEMENT

AIM:

To write a c program to implement Paging technique for memory management.

ALGORITHM:

```
Step 1: Start the process
Step 2: Declare page number, page table, frame number and process size.
Step 3: Read the process size, total number of pages
Step 4: Read the relative address
Step 5: Calculate the physical address
Step 6: Display the address
Step 7: Stop the process
```

PROGRAM:

```
#include<stdio.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>
pthread_t tid[2];
int counter;
pthread_mutex_t lock;
void* doSomeThing(void *arg)
{
pthread_mutex_lock(&lock);
unsigned long i = 0;
counter += 1;
printf("\n Job %d started\n", counter);
for(i=0; i<(0xFFFFFFFF);i++);
printf("\n Job %d finished\n", counter);
pthread_mutex_unlock(&lock);
return NULL;
}
int main(void)
{
int i = 0;
int err;
if (pthread_mutex_init(&lock, NULL) != 0)
{ printf("\n mutex init failed\n");
return 1;
}
while(i < 2)
{
err = pthread_create(&(tid[i]), NULL, &doSomeThing, NULL);
if (err != 0)
printf("\ncan't create thread :[%s]", strerror(err));
i++;
}
pthread_join(tid[0], NULL);
pthread_join(tid[1], NULL);
pthread_mutex_destroy(&lock);
return 0;}
```

OUTPUT :

```
Job 1 started
Job 1 finished
Job 2 started
Job 2 finished
```

RESULT:

Thus the program was successfully executed.

Ex.No: 11 MEMORY ALLOCATION TECHNIQUES

AIM:

To Write a C program to simulate the following contiguous memory allocation techniques
a) Worst-fit b) Best-fit c) First-fit

DESCRIPTION

One of the simplest methods for memory allocation is to divide memory into several fixed-sized partitions. Each partition may contain exactly one process. In this multiple-partition method, when a partition is free, a process is selected from the input queue and is loaded into the free partition. When the process terminates, the partition becomes available for another process. The operating system keeps a table indicating which parts of memory are available and which are occupied. Finally, when a process arrives and needs memory, a memory section large enough for this process is provided. When it is time to load or swap a process into main memory, and if there is more than one free block of memory of sufficient size, then the operating system must decide which free block to allocate. Best-fit strategy chooses the block that is closest in size to the request. First-fit chooses the first available block that is large enough. Worst-fit chooses the largest available block.

PROGRAM

WORST-FIT

```
#include<stdio.h>
#include<conio.h>
#define max 25
void main()
{
        int
        frag[max],b[max],f[max],i,j,nb,nf,t
        emp; static int bf[max],ff[max];
        clrscr();
        printf("\n\tMemory Management Scheme - First Fit");
        printf("\nEnter the number of blocks:");
        scanf("%d",&nb);
        printf("Enter the number of files:");
        scanf("%d",&nf);
        printf("\nEnter the size of the blocks:-\n");
        for(i=1;i<=nb;i++)
                {
                printf("Block %d:",i);
                scanf("%d",&b[i]);
                }
        printf("Enter the size of the files :-\n");
        for(i=1;i<=nf;i++)
                {
                printf("File %d:",i);
                scanf("%d",&f[i]);
                }
        for(i=1;i<=nf;i++)
        {
                for(j=1;j<=nb;j++)
                {
                        if(bf[j]!=1)
                        {
                        temp=b[j]-f[i];
                        if(temp>=0)
                        {
                                ff[i]=j;
                                break;
                        }

                        }
                }
                frag[i]=temp;
                bf[ff[i]]=1;
        }
        printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
        for(i=1;i<=nf;i++)
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
        getch();
}
```

INPUT

```
Enter the number of blocks: 3
Enter the number of files: 2
Enter the size of the blocks:-
Block 1: 5
Block 2: 2
Block 3: 7
Enter the size of the files:-
File 1: 1
File 2: 4
```

OUTPUT

```
File No          File Size       Block No       Block Size      Fragment
1           1           1           5           4
2           4           3           7           3
```

BEST-FIT

```
                #include<stdio.h>
                #include<conio.h>
                #define max 25
                void main()
                {
                        int frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000;
                        static int bf[max],ff[max];
                        clrscr();
                        printf("\nEnter the number of blocks:");
                        scanf("%d",&nb);
                        printf("Enter the number of files:");
                        scanf("%d",&nf);
                        printf("\nEnter the size of the blocks:-\n");
                        for(i=1;i<=nb;i++)
                        printf("Block %d:",i);
                        scanf("%d",&b[i]);
                        printf("Enter the size of the files :-\n");
                        for(i=1;i<=nf;i++)
                        {
```

```
                printf("File %d:",i);
                scanf("%d",&f[i]);
        }
        for(i=1;i<=nf;i++)
        {
                for(j=1;j<=nb;j++)
                {
                        if(bf[j]!=1)
                        {
                                temp=b[j]-f[i];
                                if(temp>=0)
                                        if(lowest>temp)
                                        {
                                        ff[i]=j;
                                        lowest=temp;
                                        }
                        }}
                frag[i]=lowest; bf[ff[i]]=1; lowest=10000;
        }
        printf("\nFile No\tFile Size \tBlock No\tBlock
        Size\tFragment"); for(i=1;i<=nf && ff[i]!=0;i++)
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
        getch();
}
```

INPUT

```
Enter the number of blocks: 3
Enter the number of files: 2
Enter the size of the blocks:-
Block 1: 5
Block 2: 2
Block 3: 7
Enter the size of the files:-
File 1: 1
File 2: 4
```

OUTPUT

```
File No         File Size       Block No        Block Size      Fragment
1               1               2                2               1
2               4               1               5               1
```

FIRST-FIT

```
#include<stdio.h>
#include<conio.h>
#define max 25
void main()
{
        int
        frag[max],b[max],f[max],i,j,nb,nf,temp,highes
        t=0; static int bf[max],ff[max];
        clrscr();
        printf("\n\tMemory Management Scheme - Worst Fit");
        printf("\nEnter the number of blocks:");
        scanf("%d",&nb);
        printf("Enter the number of files:");
        scanf("%d",&nf);
        printf("\nEnter the size of the blocks:-\n");
        for(i=1;i<=nb;i++)
        {
                printf("Block %d:",i);
                scanf("%d",&b[i]);
        }
        printf("Enter the size of the files :-\n");
        for(i=1;i<=nf;i++)
        {
                printf("File %d:",i);
                scanf("%d",&f[i]);
        }
        for(i=1;i<=nf;i++)
        {
                for(j=1;j<=nb;j++)
                {
                        if(bf[j]!=1) //if bf[j] is not allocated
                        {
                                temp=b[j]-f[i];
                                if(temp>=0)
                                if(highest<temp)
                                {
                        }
                }
                frag[i]=highest; bf[ff[i]]=1; highest=0;
                }
                ff[i]=j; highest=temp;
        }
        printf("\nFile_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragement");
        for(i=1;i<=nf;i++)
                printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
        getch();}
```

INPUT

```
Enter the number of blocks: 3
Enter the number of files: 2
Enter the size of the blocks:-
Block 1: 5
Block 2: 2
Block 3: 7
Enter the size of the files:-
File 1: 1
File 2: 4
```

OUTPUT

```
File No     File Size     Block No     Block Size     Fragment
1           1             3            7               6
2           4             1            5               1
```

RESULT:

Thus the program was successfully executed.

Ex.No: 12 **SIMULATE PAGE REPLACEMENT ALGORITHMS FIFO**

AIM:

To Simulate FIFO page replacement algorithms.

ALGORITHM:

1. Start the program
2. Read the number of frames
3. Read the number of pages
4. Read the page numbers
5. Initialize the values in frames to -1
6. Allocate the pages in to frames in First in first out order.
7. Display the number of page faults.
8. Stop the program

PROGRAM: (SIMULATE PAGE REPLACEMENT ALGORITHMS FIFO)

```
#include<stdio.h>
#include<conio.h>
int i,j,nof,nor,flag=0,ref[50],frm[50],pf=0,victim=-1;
void main()
{
clrscr();
printf("\n \t\t\t FIFI PAGE REPLACEMENT ALGORITHM"); printf("\n Enter no.of frames. .. ");
scanf("%d",&nof);
printf("Enter number of Pages.\n");
scanf("%d",&nor);
printf("\n Enter the Page No.. "); for(i=0;i<nor;i++) scanf("%d",&ref[i]);
printf("\nThe given Pages are:"); for(i=0;i<nor;i++) printf("%4d",ref[i]); for(i=1;i<=nof;i++)
frm[i]=-1; printf("\n"); for(i=0;i<nor;i++)
{
flag=0;
printf("\n\t page no %d->\t",ref[i]);

for(j=0;j<nof;j++)
{
if(frm[j]==ref[i])
{
flag=1;
break;
}}
if(flag==0)
{
pf++; victim++; victim=victim%nof; frm[victim]=ref[i]; for(j=0;j<nof;j++) printf("%4d",frm[j]);
```

```
  }  }

 printf("\n\n\t\t  No.of pages faults...%d",pf);
 getch();
 }
```

OUTPUT:

```
Enter no.of frames. .. 3
Enter number of Pages.
20

 Enter the Page No.. 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

The given Pages are:   1   2                 3  4  2  1  5  6  2  1  2  3  7  6  3 2  1  2  3  6
         page no 1->        1   -1 -1
         page no 2->        1   2  -1
         page no 3->        1   2   3
         page no 4->        4   2   3
         page no 2->        4   1   3
         page no 1->
         page no 5->        4   1   5
         page no 6->        6   1   5
         page no 2->        6   2   5
         page no 1->        6   2   1
         page no 2->        3   2   1
         page no 3->
         page no 7->        3   7   1
         page no 6->        3   7   6
         page no 3->        2   7   6
         page no 2->
         page no 1->        2   1   6
         page no 2->        2   1   3
         page no 3->
         page no 6->        6   1   3

                 No.of pages faults…16
```

RESULT:

Thus the program FIFO page replacement was successfully executed.

SIMULATE PAGE REPLACEMENT ALGORITHMS: LRU

AIM:

To Simulate LRU page replacement algorithms

ALGORITHM:

a. Start
b. Read the number of frames
c. Read the number of pages
d. Read the page numbers
e. Initialize the values in frames to -1
f. Allocate the pages in to frames by selecting the page that has not been used for the longest period of time.
g. Display the number of page faults.
h. stop

PROGRAM: (SIMULATE PAGE REPLACEMENT ALGORITHMS: LRU)

```
#include<stdio.h>
#include<conio.h>
int i,j,nof,nor,flag=0,ref[50],frm[50],pf=0,victim=-1;
int recent[10],lrucal[50],count=0;
int lruvictim();
void main()
{
clrscr();

printf("\n\t\t\t LRU PAGE REPLACEMENT ALGORITHM");
printf("\n Enter no.of Frames. .. "); scanf("%d",&nof);

printf(" Enter no.of reference string..");

scanf("%d",&nor);
printf("\n Enter reference string..");
for(i=0;i<nor;i++)
scanf("%d",&ref[i]);
printf("\n\n\t\t  LRU PAGE REPLACEMENT ALGORITHM ");
printf("\n\t The given reference string:"); printf("\n……..............................................."); for(i=0;i<nor;i++)
```

```
    printf("%4d",ref[i]);
    for(i=1;i<=nof;i++)
    {
    frm[i]=-1;
    lrucal[i]=0;
}
for(i=0;i<10;i++)
recent[i]=0; printf("\n"); for(i=0;i<nor;i++)
{
flag=0;
printf("\n\t Reference NO %d->\t",ref[i]);
for(j=0;j<nof;j++)
{
if(frm[j]==ref[i])
{
flag=1;
break;
}
}
if(flag==0)
{

count++; if(count<=nof) victim++;
else victim=lruvictim(); pf++; frm[victim]=ref[i]; for(j=0;j<nof;j++) printf("%4d",frm[j]);
}
recent[ref[i]]=i;
}
printf("\n\n\t No.of page faults...%d",pf);
getch();
}
int lruvictim()
{
int i,j,temp1,temp2;
for(i=0;i<nof;i++)
{
temp1=frm[i];
lrucal[i]=recent[temp1];
} temp2=lrucal[0]; for(j=1;j<nof;j++)
{
if(temp2>lrucal[j])
temp2=lrucal[j];

} for(i=0;i<nof;i++) if(ref[temp2]==frm[i]) return i; return 0;

}
```

OUTPUT:

```
                    LRU PAGE REPLACEMENT ALGORITHM
        Enter no.of Frames. .. 3
        Enter no.of reference string. 20

        Enter reference string. 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 3 6 5

                 LRU PAGE REPLACEMENT ALGORITHM
        1   2 The given reference string:   3   7   6   3   2 1   3   6 5
           3   4   2   1   5   6  2  1   2
           Reference NO 1->          1   -1 -1
           Reference NO 2->          1  2   -1
           Reference NO 3->          1  2    3
           Reference NO 4->          4  2    3
           Reference NO 2->          4  2    1
           Reference NO 1->
           Reference NO 5->          5  2    1
           Reference NO 6->          5  6    1
           Reference NO 2->          5  6    2
           Reference NO 1->          1  6    2
           Reference NO 2->          1  3    2
           Reference NO 3->
           Reference NO 7->          7  3    2
           Reference NO 6->          7  3    6
           Reference NO 3->          2  3    6
           Reference NO 2->
           Reference NO 1->          2  3    1
           Reference NO 3->          6  3    1
           Reference NO 6->
           Reference NO 5->          6  3    5

           No.of page faults...16
```

RESULT:

Thus the process LRU page replacement was executed and verified successfully.

SIMULATE PAGE REPLACEMENT ALGORITHMS: OPTIMAL

AIM:

To create program for optimal page replacement algorithms.

ALGORITHM:

1. Start the program
2. Read the number of frames
3. Read the number of pages
4. Read the page numbers
5. Initialize the values in frames to -1
6. Allocate the pages in to frames by selecting the page that will not be used for the longest period of time.

7. Display the number of page faults.

8.Stop the program

PROGRAM: (SIMULATE PAGE REPLACEMENT ALGORITHMS: OPTIMAL)

```
#include<stdio.h>
#include<conio.h>
int n,page[20],f,fr[20],i;
void display()
{
for(i=0;i<f;i++)
{
printf("%d",fr[i]);
}

printf("\n");
}
void request()
{
printf("enter no.of pages:");
scanf("%d",&n);
printf("enter no.of frames:");
scanf("%d",&f);
printf("enter no.of page no.s");
for(i=0;i<n;i++)
```

```
{
scanf("%d",&page[i]);
}
for(i=0;i<n;i++)
{
fr[i]=-1;
}
}
void replace()
{
int j,flag=0,pf=0;
int max,lp[10],index,m;
for(j=0;j<f;j++)
{
fr[j]=page[j];
flag=1;

pf++;
display();
}
for(j=f;j<n;j++)
{
flag=0;
for(i=0;i<f;i++)
{
if(fr[i]==page[j])
{
flag=1;
break;
}
}
if(flag==0)
{
for(i=0;i<f;i++)
lp[i]=0;
for(i=0;i<f;i++)
{
for(m=j+1;m<n;m++)
{
if(fr[i]==page[m])
{

lp[i]=m-j;
break;
}
}
}
max=lp[0];
index=0;
for(i=0;i<f;i++)
{
if(lp[i]==0)
{
```

```
index=i;
break;
}
else
{
if(max<lp[i])
{
max=lp[i];
index=i;
}
}
}
fr[index]=page[j];
pf++;
display();
}

}
printf("page faults:%d",pf);
}
void main()
{
clrscr();
request();
replace();
getch();
}
```

OUTPUT:

```
enter no.of pages:20
enter no.of frames:3
enter no.of page no.s 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6
1-1-1
1-2-1
1-2-3
1-2-4
1-2-5
1-2-6
3-2-6
3-7-6
3-2-6
3-2-1
6-2-1
page faults:11
```

RESULT:

Thus the process optimal page replacement was executed and verified successfully.

Ex.No: 13

FILE ORGANIZATION TECHNIQUE
IMPLEMENTATION OF SINGLE-LEVEL FILE ORGANIZATION TECHNIQUE

AIM:

To write a C program to implement Single-Level File Organization Technique.

ALGORITHM:

Step 1: Start the program.
Step 2: Using graphics program initialize the graphics window and get the number of files.
Step 3: Get the file names.
Step 4: Display the file structure for the file name given.

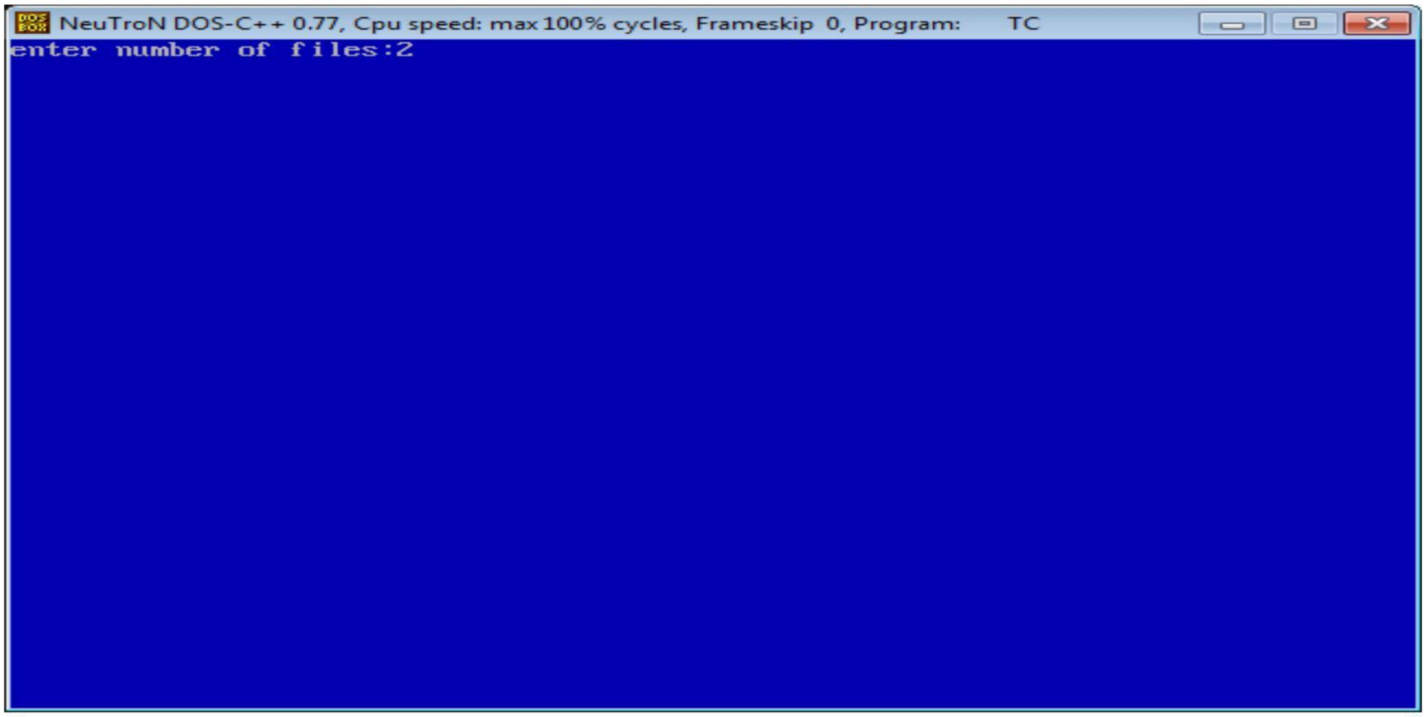
Step 5: Stop the program.

PROGRAM:

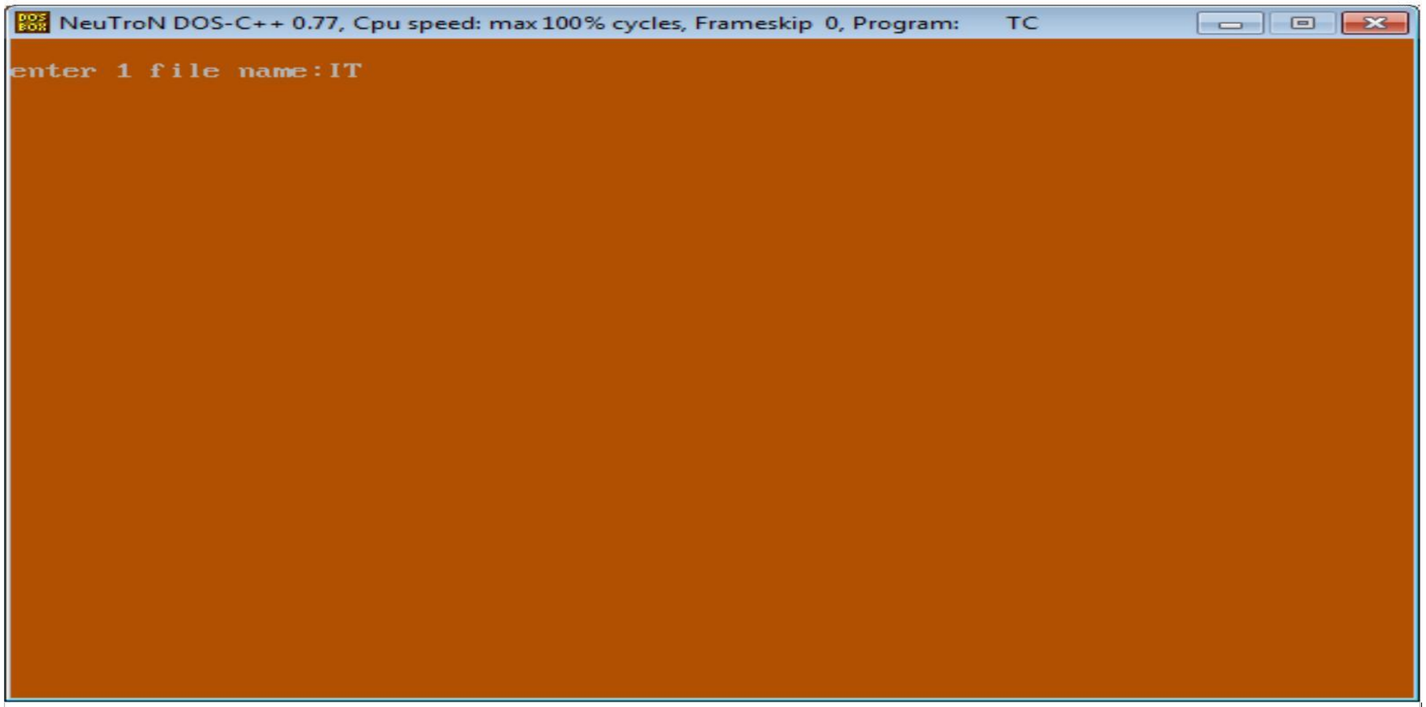
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<graphics.h>

void main()
{
int gd=DETECT,gm,count,i=0,j,mid,cir_x;
char fname[10][20];
clrscr();
initgraph(&gd,&gm,"c:\\Turboc3\\BGI");
cleardevice();
setbkcolor(BLUE);
printf("enter number of files:");
scanf("%d",&count);
if(i<count)
{
cleardevice();
setbkcolor(6);
printf("enter %d file name:",i+1);
scanf("%s",fname[i]);
setfillstyle(1,MAGENTA);
mid=640/count;

cir_x=mid/3;
bar3d(270,100,370,150,0,0);
settextstyle(2,0,4);
settextjustify(1,1);
outtextxy(320,125,"root directory");
setcolor(10);
i++;
for(j=0;j<i;j++,cir_x+=mid)
{
```

```
line(320,150,cir_x,250);
fillellipse(cir_x,250,30,30);
outtextxy(cir_x,250,fname[j]);
}
}
getch();
closegraph();
}
```

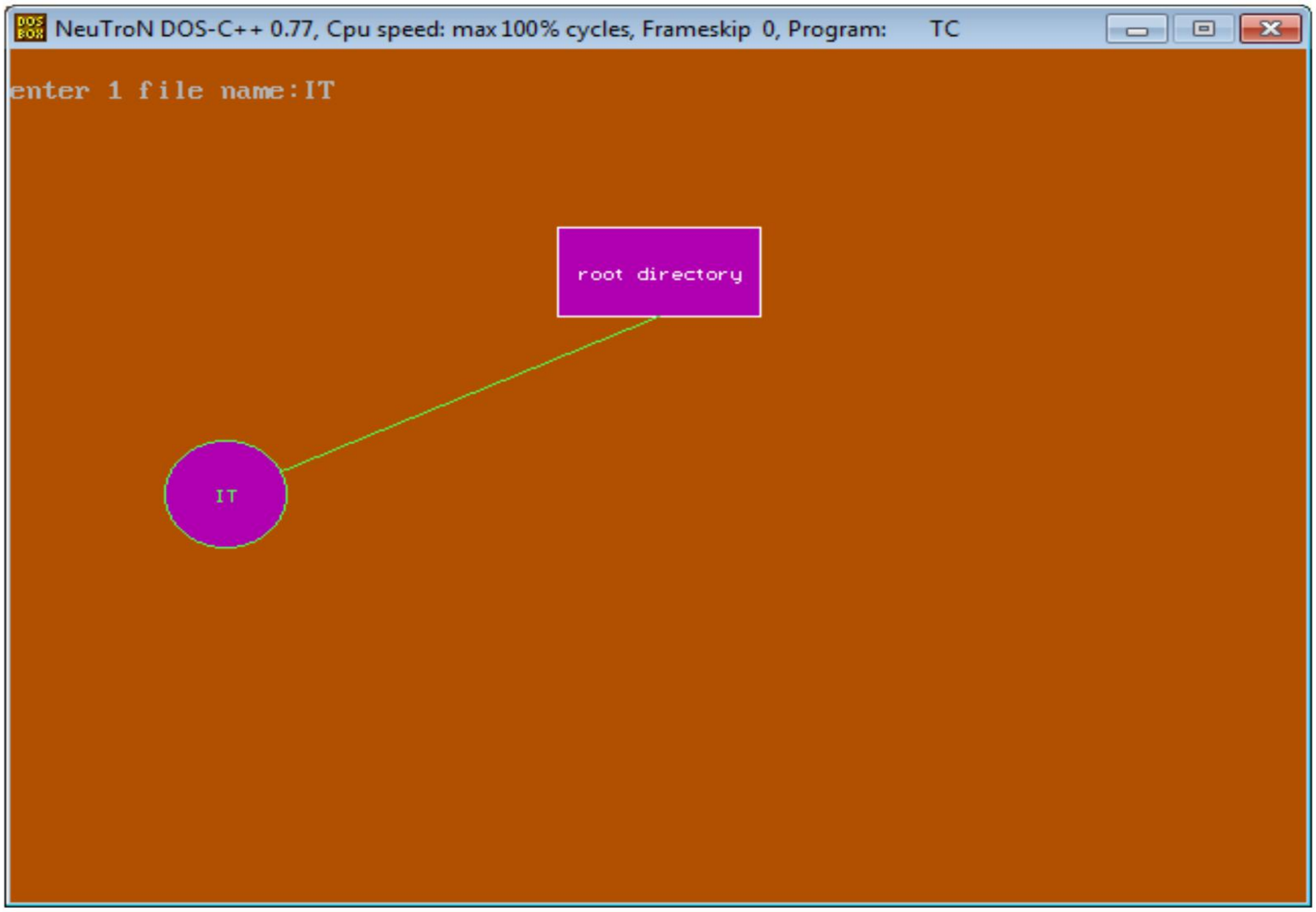
OUTPUT:


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
enter number of files:2
```


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
enter 1 file name:IT
```



```
enter 1 file name:IT
```

NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
root directory
IT

RESULT:

Thus C program to implement Single-Level File Organization Technique was written, executed and output is verified successfully.

IMPLEMENTATION OF TWO-LEVEL FILE ORGANIZATION TECHNIQUE

AIM:

To write a C program to implement Two-Level file Organization Technique.

ALGORITHM:

Step 1: Start the program.
Step 2: Create a structure to generate levels.
Step 3: Intialize the graphics window.
Step 4: Display the root directory based on which get the sub directories and files.

Step 5: Based on the files and directories create a 2 level file orgranization and display it.

Step 6: Stop the program.

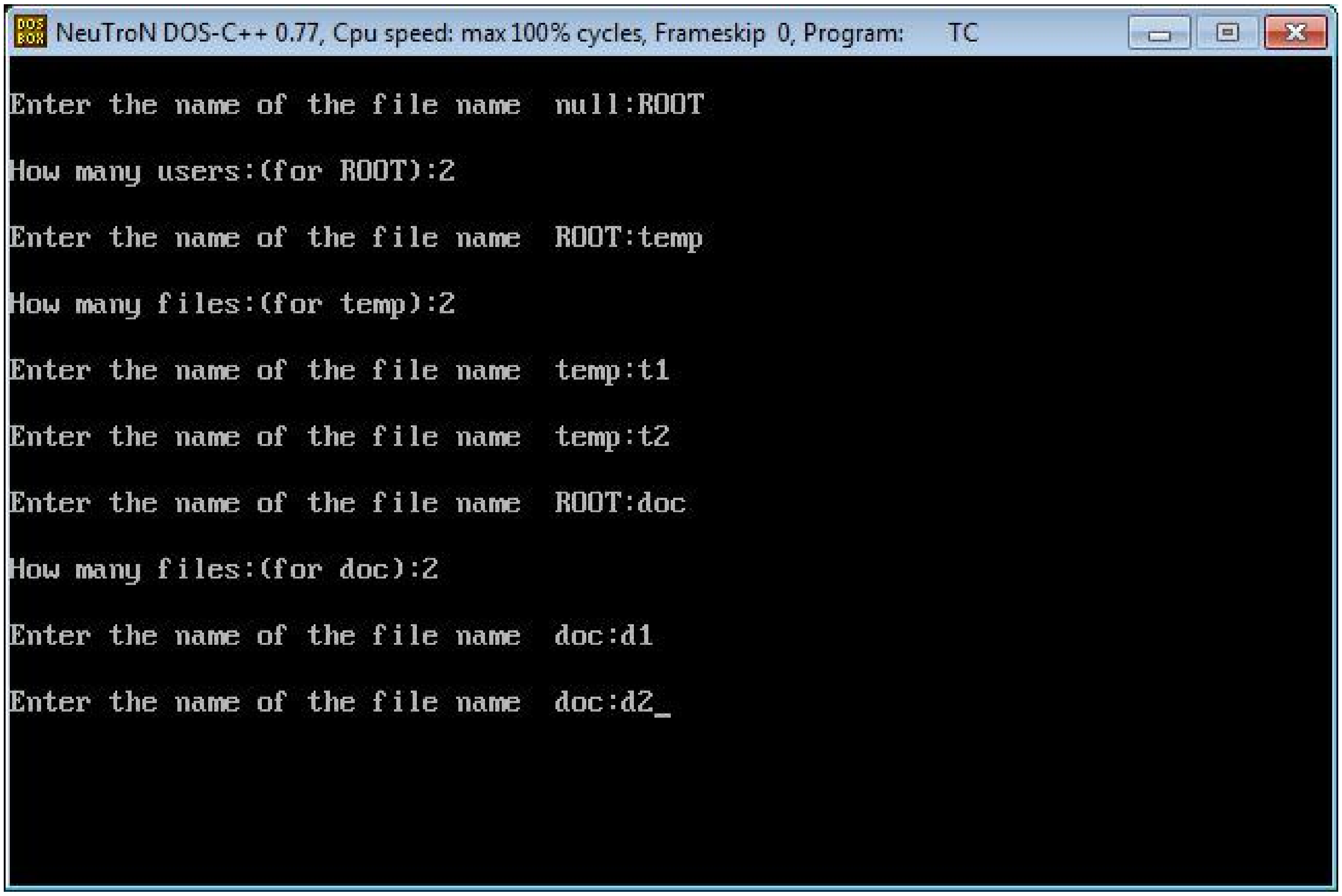
PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
struct tree_element
{
char name[20];
int x,y,ftype,lx,rx,nc,level;
struct tree_element *link[5];
};
typedef struct tree_element node;
void main()
{
int gd=DETECT,gm;
node *root;
root=NULL;

clrscr();
create(&root,0,"null",0,639,320);
clrscr();
initgraph(&gd,&gm,"c:\\Turboc3\\BGI");
display(root);

getch();
closegraph();
}
create(node **root,int lev ,char *dname,int lx,int rx,int x)
{
int i, gap;
if(*root==NULL)
{
(*root)=(node*)malloc(sizeof(node));
printf("\nEnter the name of the file name  %s:",dname);
fflush(stdin);
```

```
gets((*root)->name);
if(lev==0 || lev==1)
(*root)-> ftype=1;
else
(*root)->ftype=2;
(*root)->level=lev;
(*root)->y=50+lev*50;
(*root)->x=x;
(*root)->lx=lx  ;
(*root)->rx=rx;
for(i=0;i<5;i++)
(*root)->link[i]=NULL;
if((*root)->ftype==1)
{
if(lev==0 || lev==1)
{

if((*root)->level==0)
printf("\nHow many users:");
else
printf("\nHow many files:");
printf("(for %s):",(*root)->name);
scanf("%d",&(*root)->nc);
}
else
(*root)->nc=0;
if((*root)->nc==0)
gap=rx-lx;
else
gap=(rx-lx)/(*root)->nc;
for(i=0;i<(*root)->nc;i++)
create(&((*root)->link[i]),lev+1,(*root)->name,lx+gap*i,lx+gap*i+gap,lx+gap*i+gap/2);
}
else
(*root)->nc=0;
}
}
display(node *root)
{
int i;
settextstyle(2,0,4);

settextjustify(1,1);
setfillstyle(1,BLUE);
setcolor(14);
if(root!=NULL)
{
for(i=0;i<root->nc;i++)
{
line(root->x,root->y,root->link[i]->x,root->link[i]->y);
}
if(root->ftype==1)
bar3d(root->x-20, root->y-10,root->x+20,root->y+10,0,0);
```

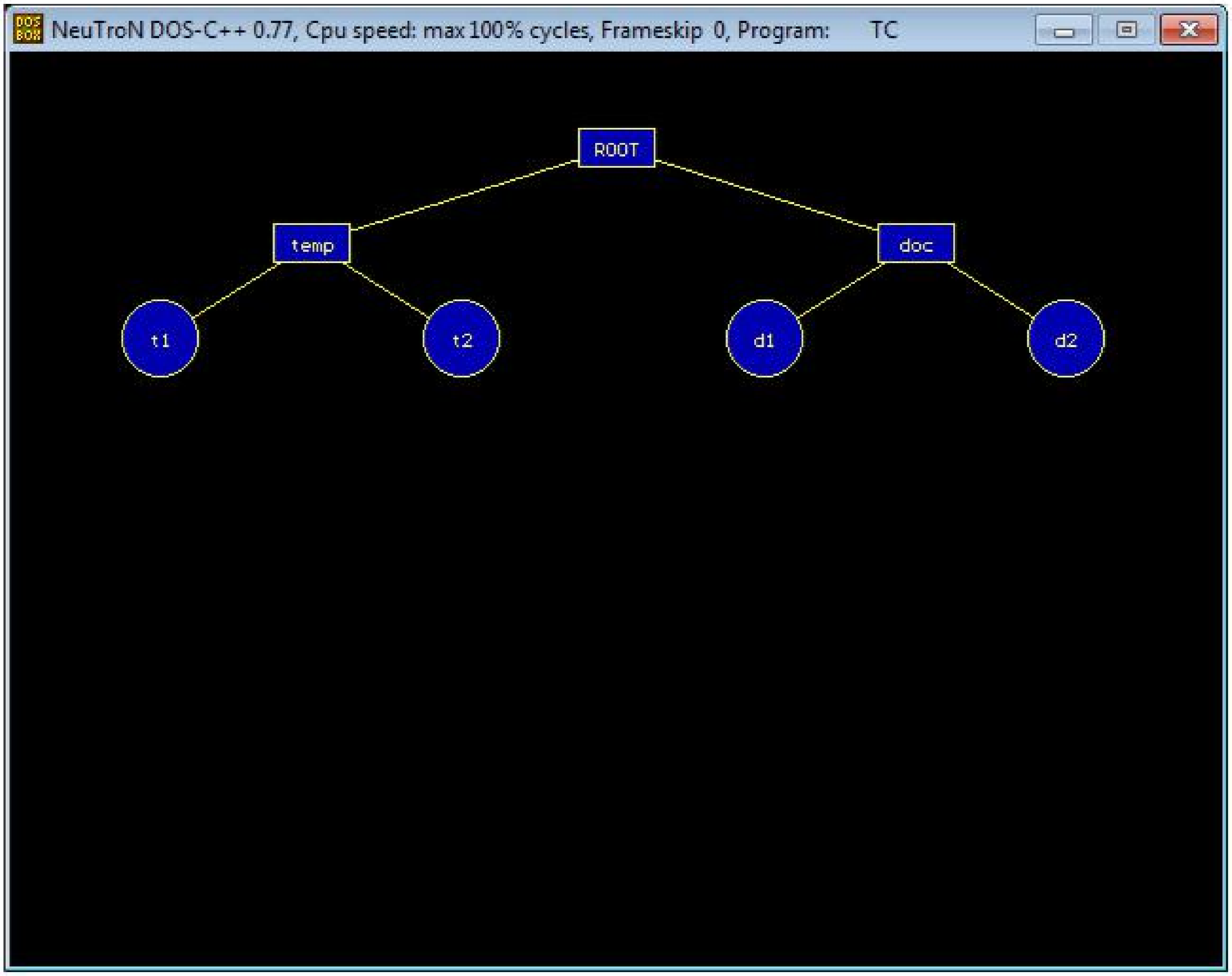
```
else
fillellipse(root->x,root->y,20,20);
outtextxy(root->x,root->y,root->name);
for(i=0;i<root->nc;i++)
{
display(root->link[i]);
}
}
}
```

OUTPUT:



NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
Enter the name of the file name  null:ROOT

How many users:(for ROOT):2

Enter the name of the file name  ROOT:temp

How many files:(for temp):2

Enter the name of the file name  temp:t1

Enter the name of the file name  temp:t2

Enter the name of the file name  ROOT:doc

How many files:(for doc):2

Enter the name of the file name  doc:d1

Enter the name of the file name  doc:d2_
```


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
ROOT
temp
doc
t1
t2
d1
d2

RESULT:

Thus C program to implement Two-Level File Organization Technique was written, executed and output is verified successfully.

IMPLEMENTATION OF DAG FILE ORGANIZATION TECHNIQUE

AIM:

To write a C program to implement DAG File Organization Technique.

ALGORITHM:

Step 1: Start the program.
Step 2: Create a structure to generate levels.
Step 3: Intialize the graphics window.
Step 4: Display the root directory based on which get the sub directories and files.
Step 5: Based on the files and directories create a DAG file orgranization.
Step 6: Get the links from the user and display the structure and the link.

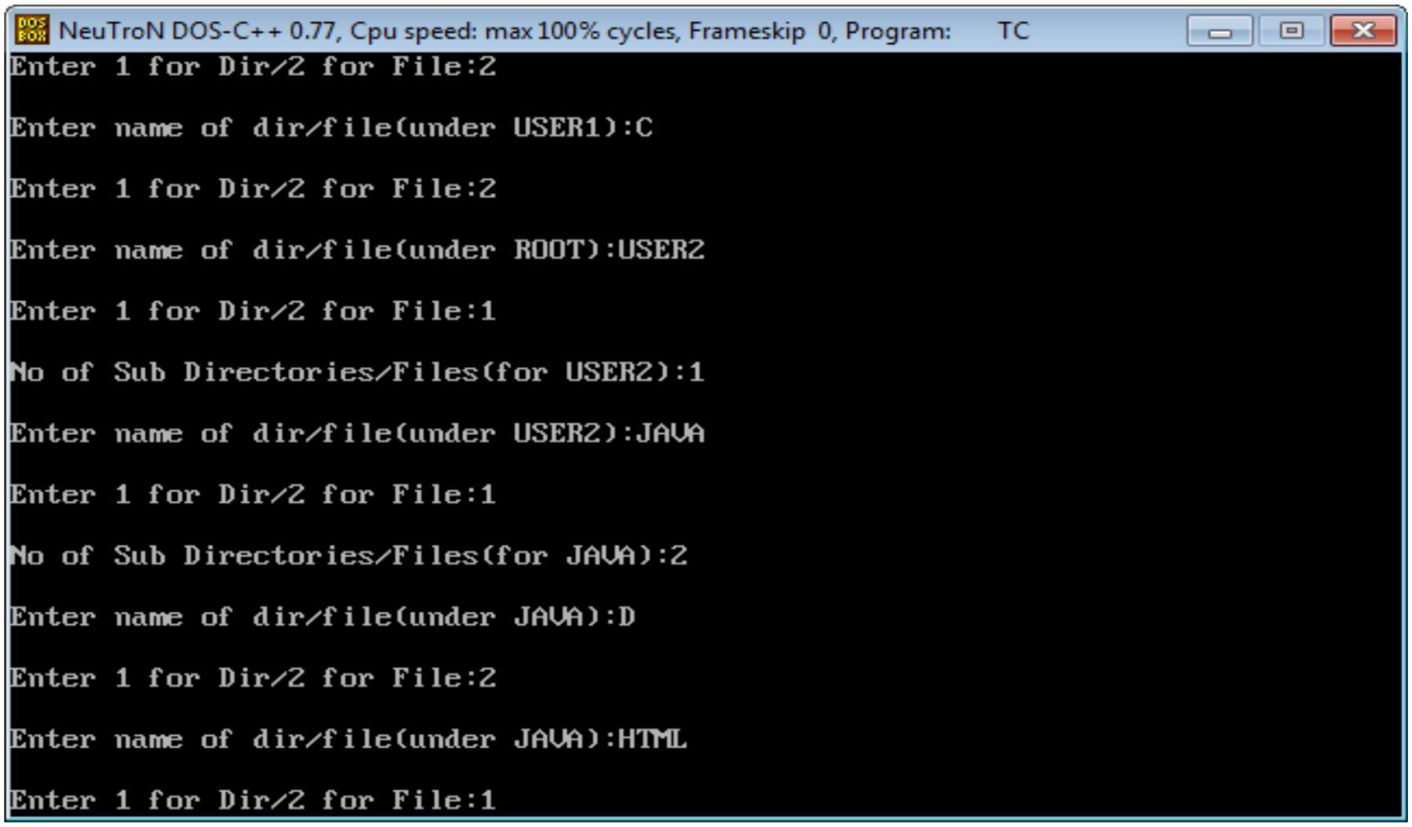
Step 6: Stop the program.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<graphics.h>
struct tree_element
{
char name[20];
int x,y,ftype,lx,rx,nc,level;
struct tree_element *link[5];
};
typedef struct tree_element node;
typedef struct
{
char from[20];
char to[20];
}link;
link L[10];
int nofl;
node *root;
void main()

{
int gd=DETECT,gm;
root=NULL;
clrscr();
create(&root,0,"root",0,639,320);
read_links();
clrscr();
initgraph(&gd,&gm,"c:\\Turboc3\\BGI");
draw_link_lines();
display(root);
getch();
closegraph();
}
create(node **root,int lev,char *dname,int lx,int rx,int x)
{
int i,gap;
if(*root==NULL)
{
(*root)=(node*)malloc(sizeof(node));
printf("\nEnter name of dir/file(under %s):",dname);
fflush(stdin);
gets((*root)->name);
printf("\nEnter 1 for Dir/2 for File:");
scanf("%d",&(*root)->ftype);
(*root)->level=lev;
(*root)->y=50+lev*50;
(*root)->x=x;

(*root)->lx=lx;
(*root)->rx=rx;
for(i=0;i<5;i++)
(*root)->link[i]=NULL;
if((*root)->ftype==1)
{
printf("\nNo of Sub Directories/Files(for %s):",(*root)->name);
scanf("%d",&(*root)->nc);
if((*root)->nc==0)
gap=rx-lx;
else
gap=(rx-lx)/(*root)->nc;
for(i=0;i<(*root)->nc;i++)
create(&((*root)->link[i]),lev+1,(*root)->name,lx+gap*i,lx+gap*i+gap,lx+gap*i+gap/2);
}
else
(*root)->nc=0;
}
}
read_links()
{
int i;
printf("\nHow many Links:");
scanf("%d",&nofl);

for(i=0;i<nofl;i++)
{
printf("\nFile/Dir:");
fflush(stdin);
gets(L[i].from);
printf("\nUser Name:");
fflush(stdin);
gets(L[i].to);
}
}
draw_link_lines()
```

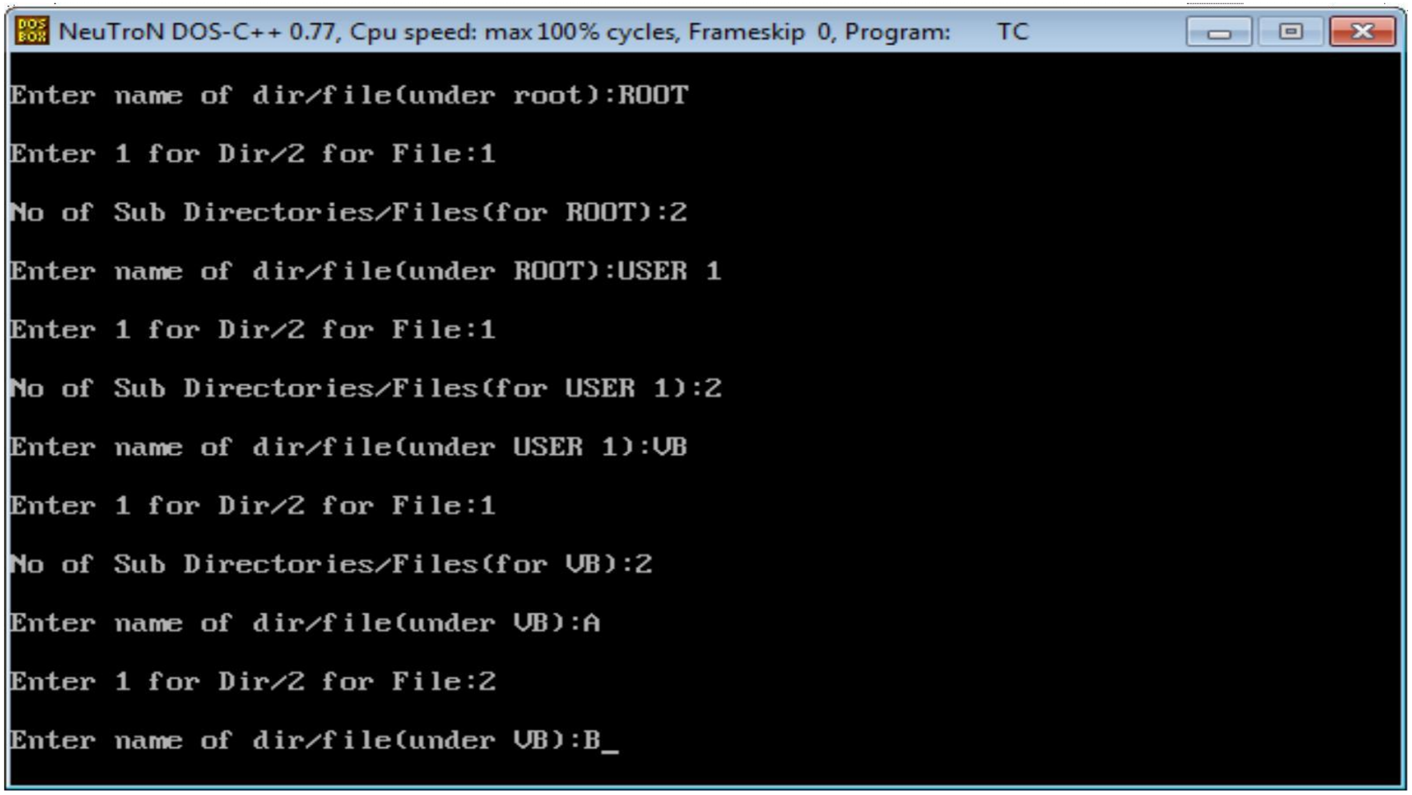
```
{
int i,x1,y1,x2,y2;
for(i=0;i<nofl;i++)
{
search(root,L[i].from,&x1,&y1);
search(root,L[i].to,&x2,&y2);
setcolor(LIGHTGREEN);
setlinestyle(3,0,1);
line(x1,y1,x2,y2);
setcolor(YELLOW);
setlinestyle(0,0,1);
}
}
search(node *root,char *s,int *x,int *y)
{
int i;

if(root!=NULL)
{
if(strcmpi(root->name,s)==0)
{
*x=root->x;
*y=root->y;
return;
}
else
{
for(i=0;i<root->nc;i++)
search(root->link[i],s,x,y);
}}}
display(node *root)
{
int i;
settextstyle(2,0,4);
settextjustify(1,1);
setfillstyle(1,BLUE);
setcolor(14);
if(root!=NULL)
{
for(i=0;i<root->nc;i++)
{

line(root->x,root->y,root->link[i]->x,root->link[i]->y);
}
if(root->ftype==1)
bar3d(root->x-20,root->y-10,root->x+20,root->y+10,0,0);
else
fillellipse(root->x,root->y,20,20);
outtextxy(root->x,root->y,root->name);
for(i=0;i<root->nc;i++)
{
display(root->link[i]);
}}}
```

OUTPUT:



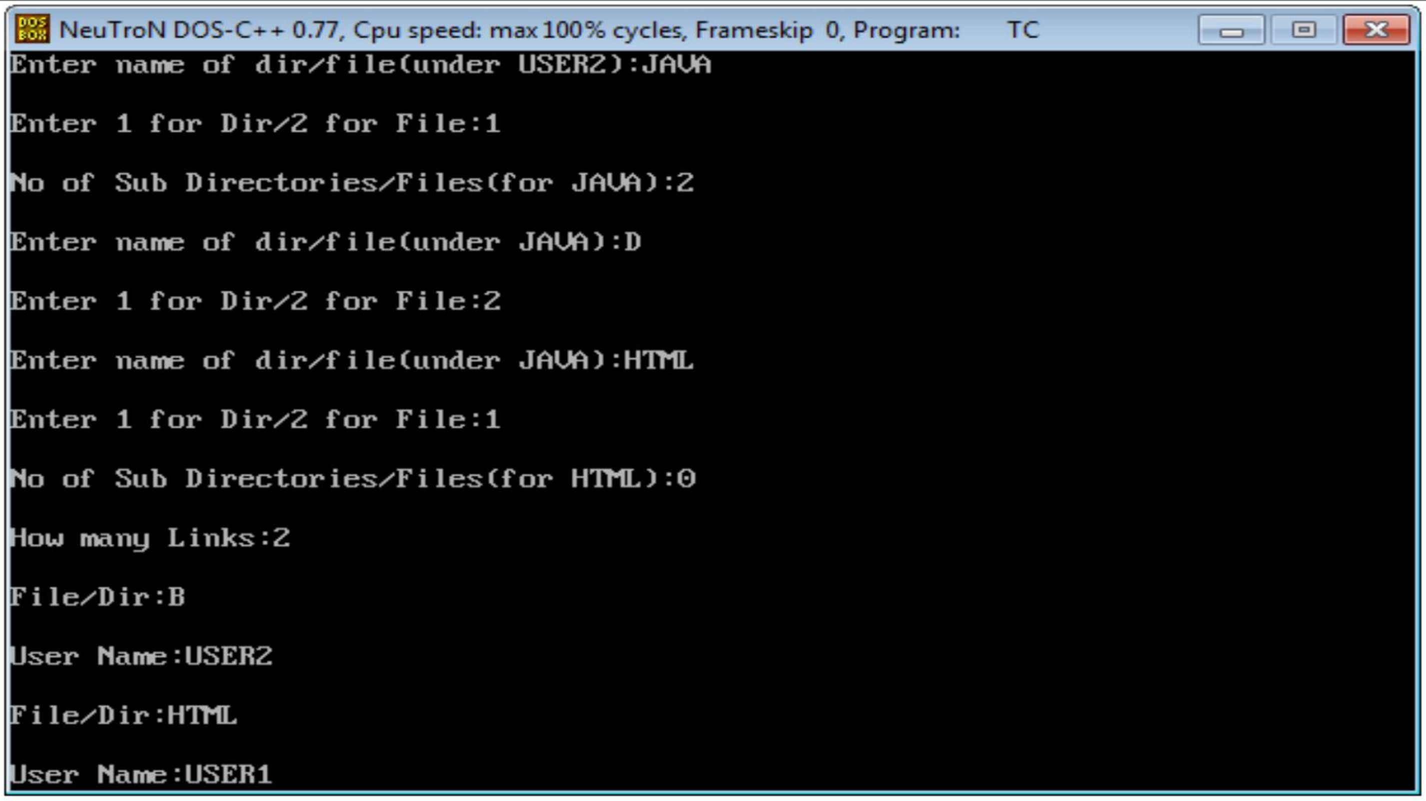
NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
Enter 1 for Dir/2 for File:2

Enter name of dir/file(under USER1):C

Enter 1 for Dir/2 for File:2

Enter name of dir/file(under ROOT):USER2

Enter 1 for Dir/2 for File:1

No of Sub Directories/Files(for USER2):1

Enter name of dir/file(under USER2):JAVA

Enter 1 for Dir/2 for File:1

No of Sub Directories/Files(for JAVA):2

Enter name of dir/file(under JAVA):D

Enter 1 for Dir/2 for File:2

Enter name of dir/file(under JAVA):HTML

Enter 1 for Dir/2 for File:1
```

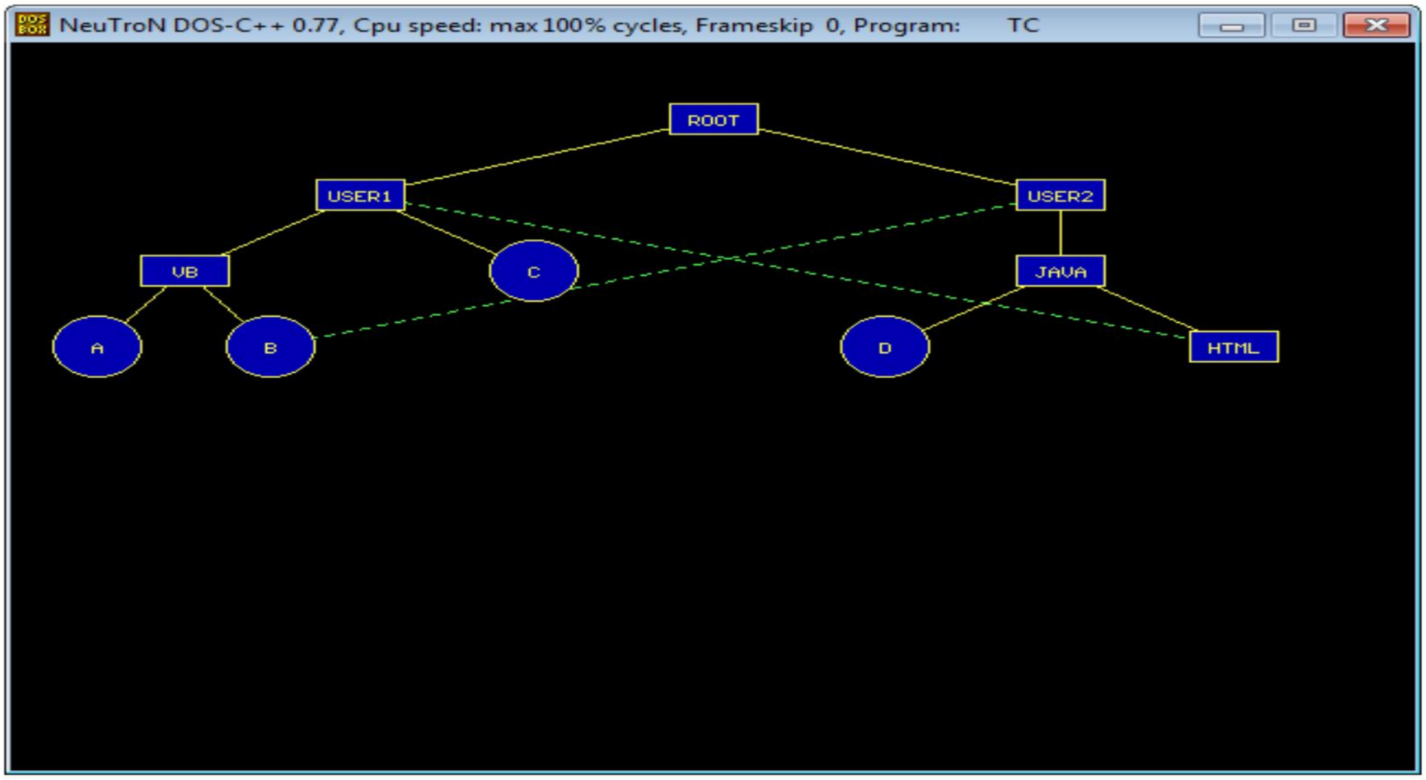


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
Enter name of dir/file(under root):ROOT

Enter 1 for Dir/2 for File:1

No of Sub Directories/Files(for ROOT):2

Enter name of dir/file(under ROOT):USER 1

Enter 1 for Dir/2 for File:1

No of Sub Directories/Files(for USER 1):2

Enter name of dir/file(under USER 1):VB

Enter 1 for Dir/2 for File:1

No of Sub Directories/Files(for VB):2

Enter name of dir/file(under VB):A

Enter 1 for Dir/2 for File:2

Enter name of dir/file(under VB):B_
```


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
Enter name of dir/file(under USER2):JAVA

Enter 1 for Dir/2 for File:1

No of Sub Directories/Files(for JAVA):2

Enter name of dir/file(under JAVA):D

Enter 1 for Dir/2 for File:2

Enter name of dir/file(under JAVA):HTML

Enter 1 for Dir/2 for File:1

No of Sub Directories/Files(for HTML):0

How many Links:2

File/Dir:B

User Name:USER2

File/Dir:HTML

User Name:USER1
```


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
ROOT
USER1
USER2
VB
C
JAVA
A
B
D
HTML

RESULT

Thus C program to implement DAG File Organization Technique was written, executed and output is verified successfully.

Ex.No: 14 **FILE ALLOCAION TECHNIQUE**

IMPLEMENTATION OF SEQUENTIAL FILE ALLOCAION TECHNIQUE

AIM:

Write a C Program to implement Sequential File Allocation method.

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 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.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
main()
{
 int n,i,j,b[20],sb[20],t[20],x,c[20][20];
 clrscr();
 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("File name is:%d",x);
 printf("length is:%d",b[x-1]);
 printf("blocks occupied:");
 for(i=0;i<b[x-1];i++)
            printf("%4d",c[x-1][i]);
 getch();
}
```

OUTPUT:

```
Enter no.of files: 2
Enter no. of blocks occupied by file1 4
Enter the starting block of file1 2
 Enter no. of blocks occupied by file2 10
Enter the starting block of file2 5
Filename        Start block     length
 1                    2               4
2                     5               10
Enter file name: rajesh
 File name is:12803 length is:0blocks occupied.
```

RESULT:

Thus the program for Sequential File Allocation method was executed and verified successfully

INDEXED FILE ALLOCAION TECHNIQUE

AIM:

Write a C Program to implement Indexed File Allocation method.

ALGORITHM:

```
Step 1:  Start.
Step 2:  Let n be the size of the buffer
Step 3:  check if there are any producer
Step 4:  if yes check whether the buffer is full
Step 5:  If no the producer item is stored in the buffer
Step 6:  If the buffer is full the producer has to wait
Step 7:  Check there is any cosumer. If yes check whether the buffer is empty
Step 8:  If no the consumer consumes them from the buffer
Step 9:  If the buffer is empty, the consumer has to wait.
Step 10: Repeat checking for the producer and consumer till required
Step 11: Terminate the process.
```

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
main()
{
 int n,m[20],i,j,sb[20],s[20],b[20][20],x;
 clrscr();
 printf("Enter no. of files:");
 scanf("%d",&n);
 for(i=0;i<n;i++)
 {       printf("Enter starting block and size of file%d:",i+1);
         scanf("%d%d",&sb[i],&s[i]);
         printf("Enter blocks occupied by 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("file name is:%d\n",x);
 i=x-1;
 printf("Index is:%d",sb[i]);
 printf("Block occupied are:");
 for(j=0;j<m[i];j++)
          printf("%3d",b[i][j]);
 getch();
}
```

OUTPUT:

```
Enter no. of files:2
 Enter starting block and size of file1: 2   5
Enter blocks occupied by file1:10
enter blocks of file1:3
2  5   4  6  7   2   6  4  7
Enter starting block and size of file2: 3   4
Enter blocks occupied by file2:5
enter     blocks     of     file2:     2          3              4          5              6
File    index  length
 1      2      10
2      3      5
Enter file name: venkat
file name is:12803
Index is:0 Block occupied are:
```

RESULT:

Thus the program for Indexed File Allocation method was executed and verified successfully.

LINKED FILE ALLOCATION

AIM:

Write a C Program to implement Linked File Allocation method.

ALGORITHM:

Step 1: Create a queue to hold all pages in memory

Step 2: When the page is required replace the page at the head of the queue

Step 3: Now the new page is inserted at the tail of the queue

Step 4: Create a stack

Step 5: When the page fault occurs replace page present at the bottom of the stack

Step 6: Stop the allocation.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
struct file
{
 char fname[10];
 int start,size,block[10];
}f[10];
main()
{
 int i,j,n;
 clrscr();
 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=1;j<=f[i].size-1;j++)
              printf("%d--->",f[i].block[j]);
        printf("%d",f[i].block[j]);
        printf("\n");
 }
 getch();
}
```

OUTPUT:

```
Enter no. of files:2
Enter file name:venkat
Enter starting block:20
Enter no.of blocks:6
Enter block numbers: 4
12
15
45
32
25
Enter file name:rajesh
Enter starting block:12
Enter no.of blocks:5
Enter block numbers:6
5
4
3
2
File    start   size    block
venkat  20      6       4--->12--->15--->45--->32--->25
rajesh  12      5       6--->5--->4--->3--->2
```

RESULT:

Thus the program for Linked File Allocation method was executed and verified successfully

Ex.No: 15 **DISK SCHEDULING ALGORITHMS**

AIM: To Write a C program to simulate disk scheduling algorithms
a) FCFS b) SCAN c) C-SCAN

DESCRIPTION

One of the responsibilities of the operating system is to use the hardware efficiently. For the disk drives, meeting this responsibility entails having fast access time and large disk bandwidth. Both the access time and the bandwidth can be improved by managing the order in which disk I/O requests are serviced which is called as disk scheduling. The simplest form of disk scheduling is, of course, the first-come, first-served (FCFS) algorithm. This algorithm is intrinsically fair, but it generally does not provide the fastest service. In the SCAN algorithm, the disk arm starts at one end, and moves towards the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk. At the other end, the direction of head movement is reversed, and servicing continues. The head continuously scans back and forth across the disk. C-SCAN is a variant of SCAN designed to provide a more uniform wait time. Like SCAN, C-SCAN moves the head from one end of the disk to the other, servicing requests along the way. When the head reaches the other end, however, it immediately returns to the beginning of the disk without servicing any requests on the return trip

PROGRAM

A) FCFS DISK SCHEDULING ALGORITHM

```
#include<stdio.h>
main()
{
        int t[20], n, I, j, tohm[20], tot=0; float avhm;
        clrscr();
        printf("enter the no.of tracks");
        scanf("%d",&n);
        printf("enter the tracks to be traversed");
        for(i=2;i<n+2;i++)
                scanf("%d",&t*i+);
        for(i=1;i<n+1;i++)
        {
                tohm[i]=t[i+1]-t[i];
                if(tohm[i]<0)
                tohm[i]=tohm[i]*(-1);
        }
        for(i=1;i<n+1;i++)
                tot+=tohm[i];
        avhm=(float)tot/n;
        printf("Tracks traversed\tDifference between tracks\n");
        for(i=1;i<n+1;i++)
                printf("%d\t\t\t%d\n",t*i+,tohm*i+);
        printf("\nAverage header movements:%f",avhm);
        getch();
}
```

INPUT

```
Enter no.of tracks:9
Enter track position:55 58 60 70 18 90 150 160 184
```

OUTPUT

```
Tracks traversed            Difference between tracks

        55                          45
        58                          3
        60                          2
        70                          10
        18                          52
        90                          72
        150                         60
        160                         10
        184                         24


Average header movements:30.888889
```

B) SCAN DISK SCHEDULING ALGORITHM

```
#include<stdio.h>
main()
{
        int t[20], d[20], h, i, j, n, temp, k, atr[20], tot, p, sum=0;
        clrscr();
        printf("enter the no of tracks to be traveresed");
        scanf("%d'",&n);
        printf("enter the position of head");
        scanf("%d",&h);
        t[0]=0;t[1]=h;
        printf("enter the tracks");
        for(i=2;i<n+2;i++)
                scanf("%d",&t[i]);
        for(i=0;i<n+2;i++)
        {
                for(j=0;j<(n+2)-i-1;j++)
                {
                        if(t[j]>t[j+1])
                        {
                                temp=t[j];
                                t[j]=t[j+1];
                                t[j+1]=temp;
                        }
                 }
        }
        for(i=0;i<n+2;i++)
        if(t[i]==h)
                j=i;k=i;
        p=0;
        while(t[j]!=0)
        {
                atr[p]=t[j]; j--;
                p++;
        }
        atr[p]=t[j];
        for(p=k+1;p<n+2;p++,k++)
                atr[p]=t[k+1];
        for(j=0;j<n+1;j++)
        {
                if(atr[j]>atr[j+1])
                        d[j]=atr[j]-atr[j+1];
                else
                        d[j]=atr[j+1]-atr[j];
                sum+=d[j];
        }
        printf("\nAverage header movements:%f",(float)sum/n);
        getch(); }
```

INPUT

```
Enter no.of tracks:9
Enter track position:55 58 60 70 18 90 150 160 184
```

OUTPUT

```
Tracks traversed                    Difference between tracks
        150                                    50
        160                                    10
        184                                    24
        90                                     94
        70                                     20
        60                                     10
        58                                     2
        55                                     3
        18                                     37

        Average header movements: 27.77
```

C) C-SCAN DISK SCHEDULING ALGORITHM

```
#include<stdio.h>
main()
{
        int t[20], d[20], h, i, j, n, temp, k, atr[20], tot, p, sum=0;
        clrscr();
        printf("enter the no of tracks to be traveresed");
        scanf("%d'",&n);
        printf("enter the position of head");
        scanf("%d",&h);
        t[0]=0;t[1]=h;
        printf("enter total tracks");
        scanf("%d",&tot);
        t[2]=tot-1;
        printf("enter the tracks");
        for(i=3;i<=n+2;i++)
                scanf("%d",&t[i]);
        for(i=0;i<=n+2;i++)
                for(j=0;j<=(n+2)-i-1;j++)
                        if(t[j]>t[j+1])
                        {
                                temp=t[j];
                                t[j]=t[j+1];
                                t[j+1]=temp
                        }
        for(i=0;i<=n+2;i++)
        if(t[i]==h);
                j=i;break;
                p=0;
                while(t[j]!=tot-1)
                {
                        atr[p]=t[j];
                        j++;
                        p++;
                }
                atr[p]=t[j];
                p++;
                i=0;
        while(p!=(n+3) && t[i]!=t[h])
        {
                atr[p]=t[i]; i++;
                p++;
        }
        for(j=0;j<n+2;j++)
        {
        if(atr[j]>atr[j+1])
        {
                d[j]=atr[j]-atr[j+1];
                else
                d[j]=atr[j+1]-atr[j];
                sum+=d[j];
            }
            printf("total header movements%d",sum);
            printf("avg is %f",(float)sum/n);
            getch();
    }
```

INPUT

```
    Enter the track position : 55 58 60 70 18 90 150 160 184
    Enter starting position : 100
```

OUTPUT

```
    Tracks traversed                Difference Between tracks
    150                                        50
    160                                        10
    184                                        24
    18                                         240
    55                                         37
    58                                         3
    60                                         2
    70                                         10
    90                                         20
    Average seek time : 35.7777779
```

RESULT:

Thus the program for Linked File Allocation method was executed and verified successfully