OPERATING SYSTEM LABORATORY MANUAL

Lab Manual for II Year B.E (CSE&IT) Studentsas per the ANNA UNIVERSITY Syllabus

Lab Manual
Observation Note Book
Short Procedures
Viva-Voce Questions

Dr. R. Vasanthi M.E.,Ph.D
HoD/CSE
Narasu's Sarathy Institute of Technology,
Poosaripatty, Salem-636305.

Mrs. K. Manjuparkavi M.E
Assistant Professor/CSE
Narasu's Sarathy Institute of Technology,
Poosaripatty, Salem-636305.

Mrs. S. Sasikala M.E
Assistant Professor/CSE
Narasu's Sarathy Institute of Technology,
Poosaripatty, Salem-636305.

Mrs. Bharani M.Tech
Assistant Professor/AI&DS
Narasu's Sarathy Institute of Technology,
Poosaripatty, Salem-636305.

PREFACE

The indispensable intention for writing the manual for **OPERATING SYSTEM** is to impart inclusive guidance for the students to learn clear description of the concepts that underlie operating systems. The inspiration at the rear is to spotlight on the practical phase of programming. To implement this objective it has been included couple of examples regarding each topic.

The manual provides complete programming guide for the students from beginning to advanced level. Humbly stating, the book is informative, educative and illuminative one.

We do hope that the contents and concepts of book will be highly useful and helpful to the readers.

ACKNOWLEDGEMENT

The world is a better place thanks to people who want to develop and lead others. What make it even better are people who share the gift of their time to mentor future leaders. Thank you to everyone who strives to grow and help others grow.

To all the individuals we have had the opportunity to lead, be led by, or watch their leadership from afar, we want to say thank you for being the inspiration and foundation for the leadership

Having an idea and turning it into a book is as hard as it sounds. The experience is both internally challenging and rewarding. We especially want to thank the individuals that helped make this happen.

We wish to express our sincere gratitude to our respected Chairman **Mr.B.Nitish Harihar,** Pro-Chairman **Mrs.Aishwarya Nitish** and Vice-Chairman & Secretary, **Mr.G.Prabakaran** for giving us the opportunity to explore our innovative thoughts.

We highly appreciate and express our thanks to our Principal. **Dr. Munusami Viswanathan** for giving us the opportunity to explore our innovative thoughts.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs):

Graduates can

- Apply their technical competence in computer science to solve real world problems, with technical and people leadership.
- Conduct cutting edge research and develop solutions on problems of social relevance.
- Work in a business environment, exhibiting team skills, work ethics, adaptability and life long learning.

PROGRAM OUTCOMES POs:

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assesssocietal,health,safety,legalandculturalissuesandtheconsequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OBJECTIVES (PSOs)

The Students will be able to

- Exhibit design and programming skills to build and automate business solutions using cutting edge technologies.
- Strong theoretical foundation leading to excellence and excitement towards research, to provide elegant solutions to complex problems.
- Ability to work effectively with various engineering fields as a team to design, build and develop system applications.

Mapping of POs/PSO s to PEO s

Contribution 1:Reasonable 2:Significant 3:Strong

	POs	**1.Graduateswill pursue higher education and research, or have a successful career in Industries associated with Computer Science and Engineering ,or as entrepreneurs.**	**2.Graduates will Have the ability And attitude to Adapt to emerging technological changes.**
1.	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.	3	1
2.	**Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiatedconclusionsusingfirstprinciplesofmathematics,natural sciences ,and Engineering Sciences.	3	1

3.	**Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.	3	2
4.	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.	3	2
5.	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.	2	3
6.	**The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.	2	2
7.	**Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.	2	1
8.	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.	3	1
9.	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.	3	2

10.	**Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.	3	2
11.	**Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.	2	2
12.	**Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.	1	3

S.No	**PSOs**		
1.	Exhibit design and programming skills to build and automate business solutions using cutting edge technologies	3	1
2.	Strong theoretical foundation leading to excellence and excitement towards research , to provide elegant solutions to complex problems	3	1
3.	Ability to work effectively with various engineering fields as a team to design ,build and develop system applications	1	3

GENERAL INSTRUCTIONS

These are the instructions for the students attending the laboratory:

√ Before entering the lab the student should carry the following things(MANDATORY)

- Identity card issued by the college.
- Class notes
- Lab observation book
- Lab Manual
- Lab Record

√ Student must sign in and sign out in the register provided when attending the lab session without fail.

√ Come to the laboratory in time. Students, who are late more than 15 min., will not be allowed to attend the lab.

√ Students need to maintain 100% attendance in lab, if not a strict action will be taken.

√ All students must follow a Dress Code while in the laboratory

√ Foods, drinks are NOT allowed.

√ All bags must be left at the indicated place.

√ Refer to the lab staff if you need any help in using the lab.

√ Respect the laboratory and its other users.

√ Workspace must be kept clean and tidy after experiment is completed.

√ Read the Manual carefully before coming to the laboratory and be sure about what you are supposed to do.

√ Do the experiments as per the instructions given in the manual.

√ Copy all the programs to observation which are taught in class before attending the lab session. Lab records need to be submitted on or before the date of submission.

CS3461	OPERATING SYSTEMS LABORATORY	L T P C
		0 0 3 1.5

COURSE OBJECTIVES:

- To install windows operating systems.
- To understand the basics of Unix command and shell programming.
- To implement various CPU scheduling algorithms.
- To implement Deadlock Avoidance and Deadlock Detection Algorithms
- To implement Page Replacement Algorithms
- To implement various memory allocation methods.
- To be familiar with File Organization and File Allocation Strategies.

LIST OF EXPERIMENTS:

1. Installation of windows operating system
2. Illustrate UNIX commands and Shell Programming
3. Process Management using System Calls: Fork, Exit, Getpid, Wait, Close
4. Write C programs to implement the various CPU Scheduling Algorithms
5. Illustrate the inter process communication strategy
6. Implement mutual exclusion by Semaphore
7. Write C programs to avoid Deadlock using Banker's Algorithm
8. Write a C program to Implement Deadlock Detection Algorithm
9. Write C program to implement Threading
10. Implement the paging Technique using C program

11. Write C programs to implement the following Memory Allocation Methods

    a. First Fit b.Worst Fit c.Best Fit

12. Write C programs to implement the various Page Replacement Algorithms

13. Write C programs to Implement the various File Organization Techniques

14. Implement the following File Allocation Strategies using C programs

    a. Sequential b.Indexed c.Linked

15. Write C programs for the implementation of various disk scheduling algorithms.

16. Install any guest operating system like Linux using VMware.

COURSE OUTCOMES:

At the end of this course, the students will be able to:

CO1: Define and implement UNIX Commands.

CO2: Compare the performance of various CPU Scheduling Algorithms.

CO3: Compare and contrast various Memory Allocation Methods.

CO4: Define File Organization and File Allocation Strategies.

CO5: Implement various Disk Scheduling Algorithms.

Ex.No:01	**INSTALLATION OF WINDOWS OPERATING SYSTEM**
Date:	

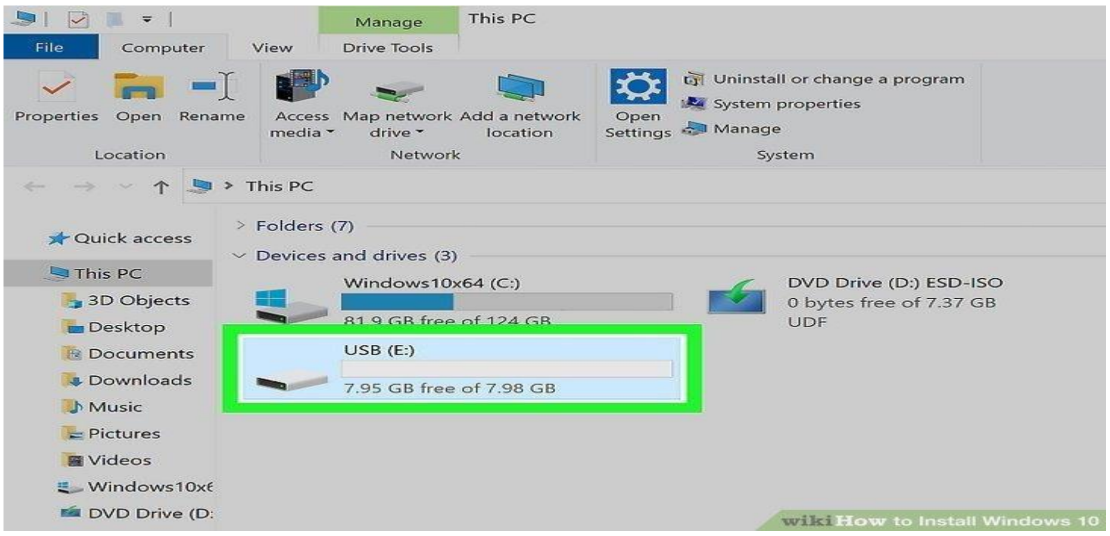
AIM:

To write a step by step process of windows operating system installation.

Installation processes:

Step1: Creating an installation Disc or Drive

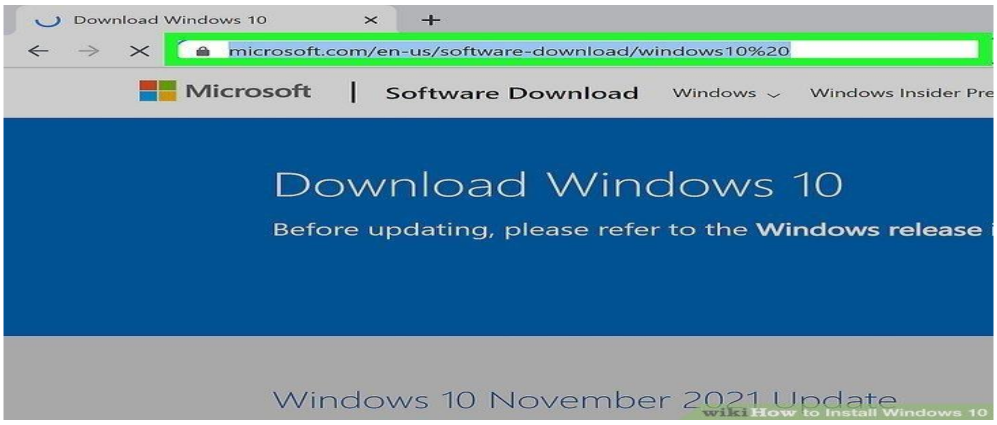
Connect a blank USB flash drive or insert a blank writable DVD. You can install Windows 10 by creating a bootable USB flash drive or DVD that contains the Windows 10 installation files. You'll need a USB flash drive that's at least 8GB, or any blank DVD to get started.



Step-2: Make sure you have the product key.

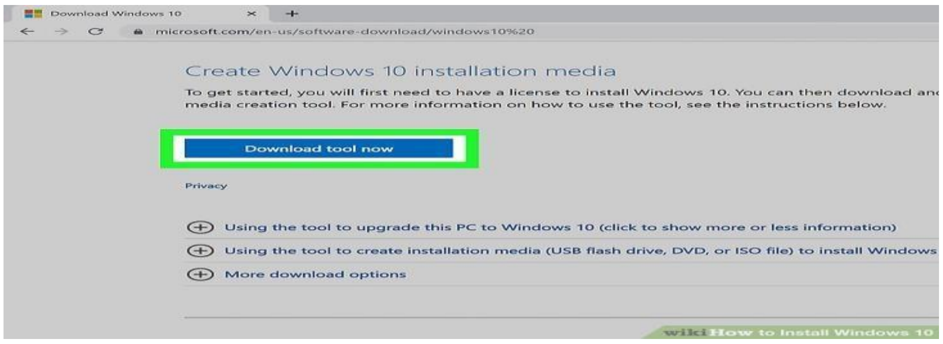
If you bought Windows 10 through Microsoft using your Microsoft account, your product key is already linked to your account. If you bought Windows 10 from another retailer, you'll have a 25-character product key that you'll need to have handy to activate Windows

- If you don't have a product key or you're installing Windows 10 on a new hard drive, make sure you've linked your Windows 10 digital license to your Microsoft account before you start the installation.
- Head to **Settings** > **Update & Security** > **Activation** from the current installation—if the activation status says Windows is activated with a digital license,
- click **Add an account** and follow the on-screen instructions to link your Microsoft account.
- If you're upgrading from an earlier version and your PC qualifies for a free upgrade, you won't need a product key.


Download Windows 10
microsoft.com/en-us/software-download/windows10%20
Microsoft | Software Download Windows Windows Insider Pre
Download Windows 10
Before updating, please refer to the Windows release
Windows 10 November 2021 Update
wiki How to Install Windows 10

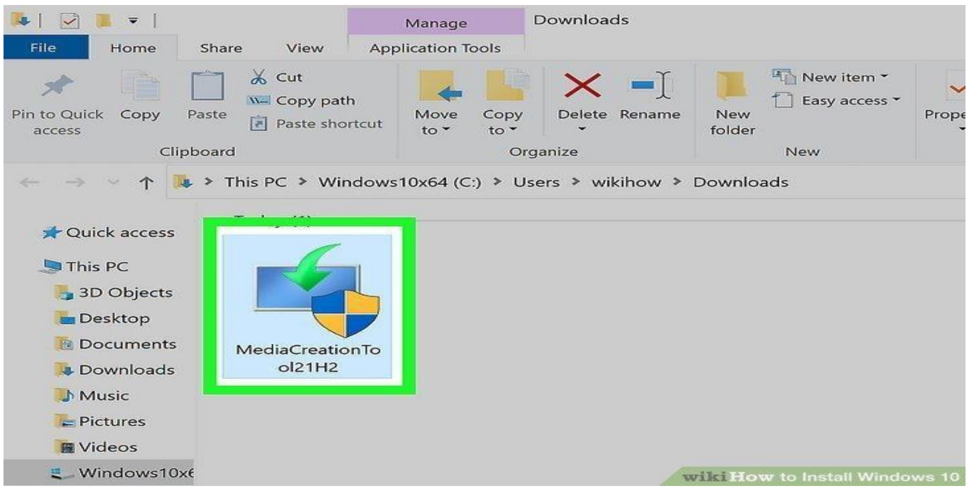
Step:3

Go to https://www.microsoft.com/en-us/software-download/windows10%20**.** This is the official download site for Windows 10.


Download Windows 10
microsoft.com/en-us/software-download/windows10%20
Create Windows 10 installation media
To get started, you will first need to have a license to install Windows 10. You can then download and media creation tool. For more information on how to use the tool, see the instructions below.
Download tool now
Privacy
Using the tool to upgrade this PC to Windows 10 (click to show more or less information)
Using the tool to create installation media (USB flash drive, DVD, or ISO file) to install Windows
More download options
wiki How to Install Windows 10

Step:4

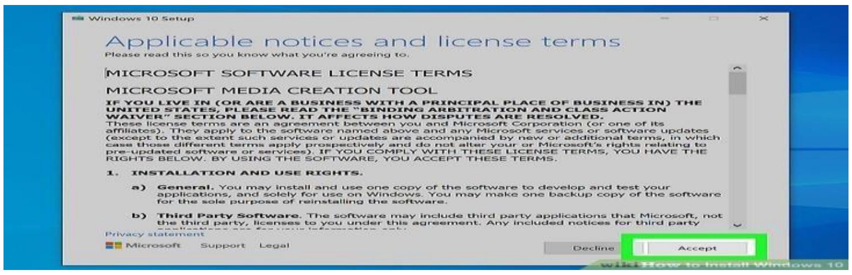
Click Download tool now. This is a blue button in the middle of the page. This downloads the Media Creation Tool, which you'll use to create your installation media (or start your upgrade).


Manage
Downloads
File Home Share View Application Tools
Pin to Quick access Copy Paste Cut Copy path Paste shortcut
Clipboard
Move to Copy to Delete Rename
Organize
New folder New item Easy access
New
Prope
This PC > Windows10x64 (C:) > Users > wikihow > Downloads
Quick access
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows10x6
MediaCreationTool21H2
wiki How to Install Windows 10

Step:5

Double-click the downloaded file. Its name begins with "MediaCreationTool" and ends with ".exe." You'll find it in your default download folder, which is usually called Downloads.

- Click **Yes** when prompted to allow the installer to run.



Applicable notices and license terms

Please read this so you know what you're agreeing to.

MICROSOFT SOFTWARE LICENSE TERMS

MICROSOFT MEDIA CREATION TOOL

IF YOU LIVE IN (OR ARE A BUSINESS WITH A PRINCIPAL PLACE OF BUSINESS IN) THE UNITED STATES, PLEASE READ THE "BINDING ARBITRATION AND CLASS ACTION WAIVER" SECTION BELOW. IT AFFECTS HOW DISPUTES ARE RESOLVED.
These license terms are an agreement between you and Microsoft Corporation (or one of its affiliates). They apply to the software named above and any Microsoft services or software updates (except to the extent such services or updates are accompanied by new or additional terms, in which case those different terms apply prospectively and do not alter your or Microsoft's rights relating to pre-updated software or services). IF YOU COMPLY WITH THESE LICENSE TERMS, YOU HAVE THE RIGHTS BELOW. BY USING THE SOFTWARE, YOU ACCEPT THESE TERMS.

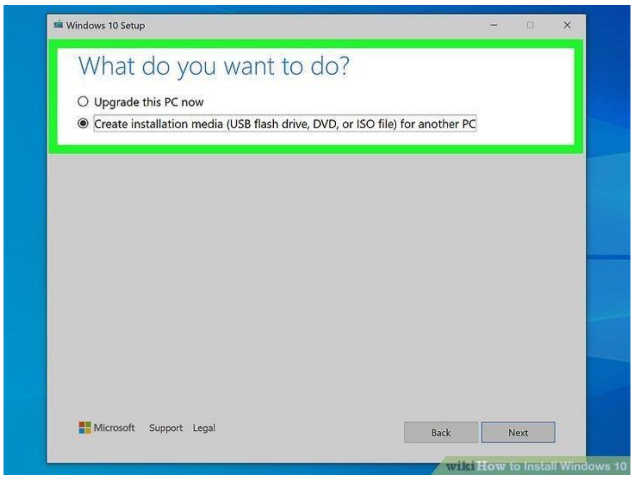
1. INSTALLATION AND USE RIGHTS.

a) General. You may install and use one copy of the software to develop and test your applications, and solely for use on Windows. You may make one backup copy of the software for the sole purpose of reinstalling the software.

b) Third Party Software. The software may include third party applications that Microsoft, not the third party, licenses to you under this agreement. Any included notices for third party

Step:6

Click Accept to accept the license. It's in the bottom-right corner of the window.

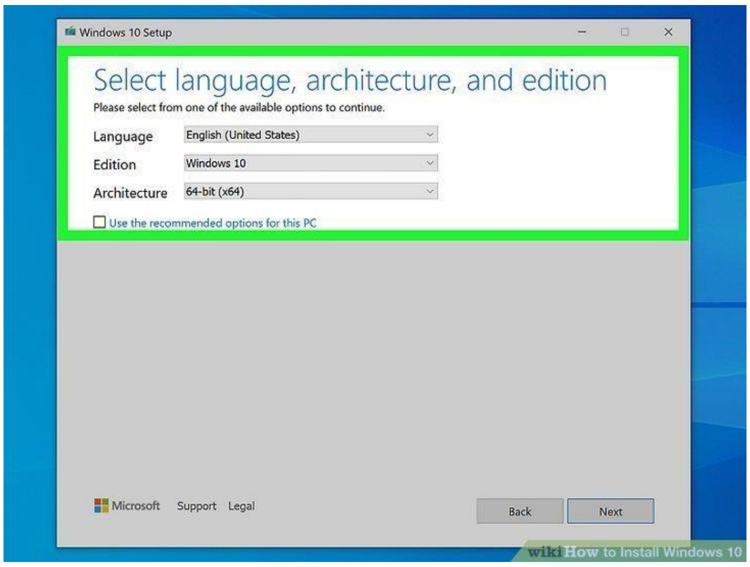


Windows 10 Setup
What do you want to do?
Upgrade this PC now
Create installation media (USB flash drive, DVD, or ISO file) for another PC
Microsoft
Support
Legal
Back
Next
wikiHow to Install Windows 10

Step-7:

Select "Create installation media" and click OK. This option lets you create a Windows installation disc or drive that will work on any compatible PC, not just the one you're using now.

- If you're updating your PC from an earlier version of Windows, select **Upgrade this PC now** instead, and then follow the on-screen instructions to install Windows 10. You're done!


Windows 10 Setup

Select language, architecture, and edition

Please select from one of the available options to continue.

Language: English (United States)

Edition: Windows 10

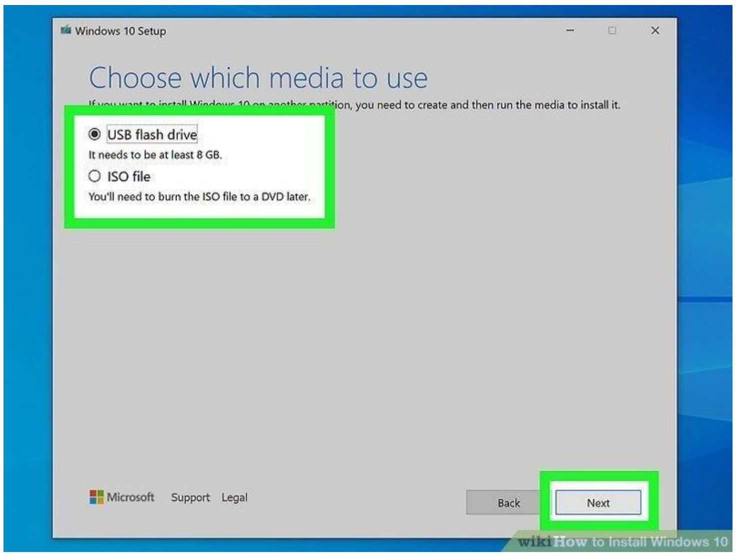
Architecture: 64-bit (x64)

Use the recommended options for this PC

Step-8:

Select your preferences and click Next. If you're installing Windows on the current PC, you can keep the default options. If you need to install on a different PC, make sure you choose the language and edition for which you have a license, and select the architecture (64-bit or 32-bit) that matches the PC you're going to install on.

- If you're not sure about the architecture, you can choose **Both** from the menu.


Windows 10 Setup

Choose which media to use

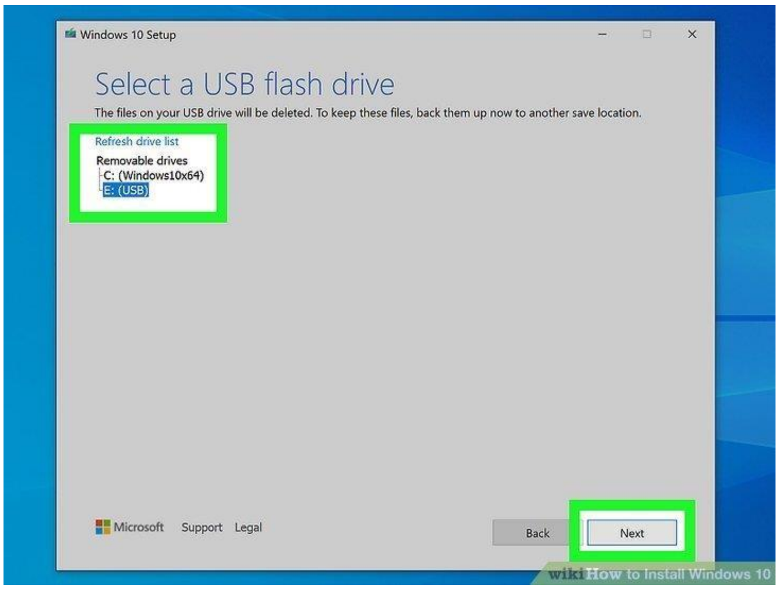
If you want to install Windows 10 on another partition, you need to create and then run the media to install it.

(●) USB flash drive
It needs to be at least 8 GB.

( ) ISO file
You'll need to burn the ISO file to a DVD later.

Step-09:

Choose an installation type and click Next. An ISO file is a type of file that can be burned to a DVD, so choose that option if you plan to create a DVD. Otherwise, choose the USB flash drive option.


Windows 10 Setup

Select a USB flash drive

The files on your USB drive will be deleted. To keep these files, back them up now to another save location.

Refresh drive list
Removable drives
C: (Windows10x64)
E: (USB)

Step-10:

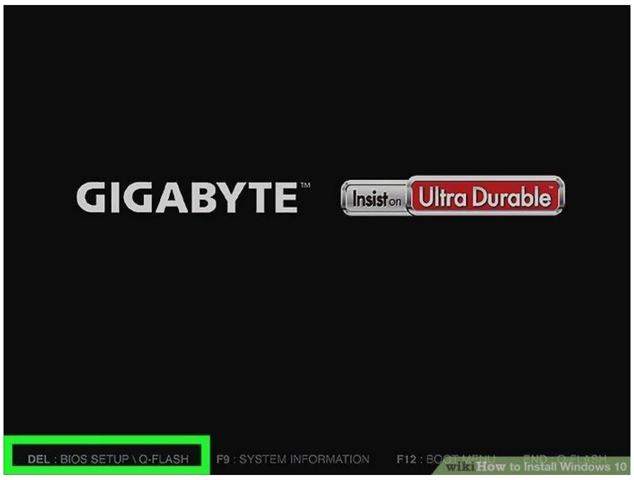
Create your installation media. The steps are a little different depending on what you're doing:

- **Flash drive:** Select your flash drive from the list, click **Next**, and wait for the installation files to install. When the process is complete, click **Finish**.
- **DVD/ISO:** Click **Save** to save the ISO file to your computer—it may take a while because the file is large and has to be downloaded. Once downloaded, you'll see a progress screen that monitors the download. When the download is complete, click **Open DVD burner** on the "Burn the ISO file to a DVD" screen, select your DVD burner, and then click **Burn** to create your DVD.

Step-11:

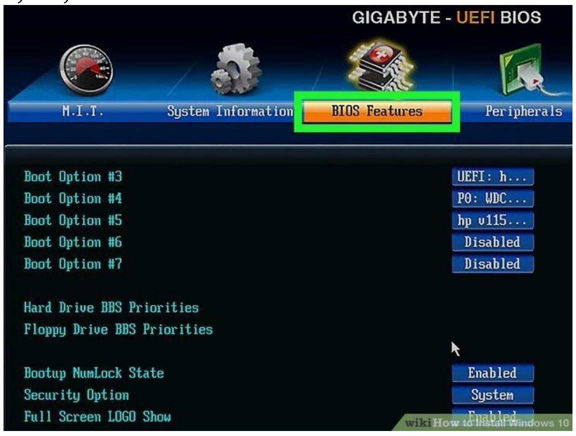
Booting from Windows 10 Installation Media

- **Connect your Windows 10 installation media.** If you created a flash drive, connect it to the PC on which you want to install Windows 10. If you made a DVD, insert it into the drive now.



Boot the PC into the BIOS. If your PC is not already set up to boot from your flash or optical drive, rebooting from your installation media won't work. You'll need to make a quick change in your BIOS to change the boot order. There are a few ways to get in:

- **Windows 8.1 or 10:** From Windows, open **Settings**, select **Update & Recovery** or **Update & Security**, and go to **Recovery** > **Restart now** > **Troubleshoot** > **Advanced Options** > **UEFI Firmware Settings** > **Restart**.
- **Any PC:** Reboot the PC and immediately start pressing (over and over again) the keyboard key required by your PC to enter "Setup," or the BIOS. The key varies by computer, but here are some of the most common keys:
    - Acer and Asus: F2 or Del
    - Dell: F2 or F12
    - HP: ESC or F10
    - Lenovo: F1, F2, or Fn + F2
    - Lenovo ThinkPads: Enter + F1.
    - MSI: DEL
    - Microsoft Surface Tablets: Press and hold the volume-up button.
    - Samsung and Toshiba: F2
    - Sony: F1, F2, or F3

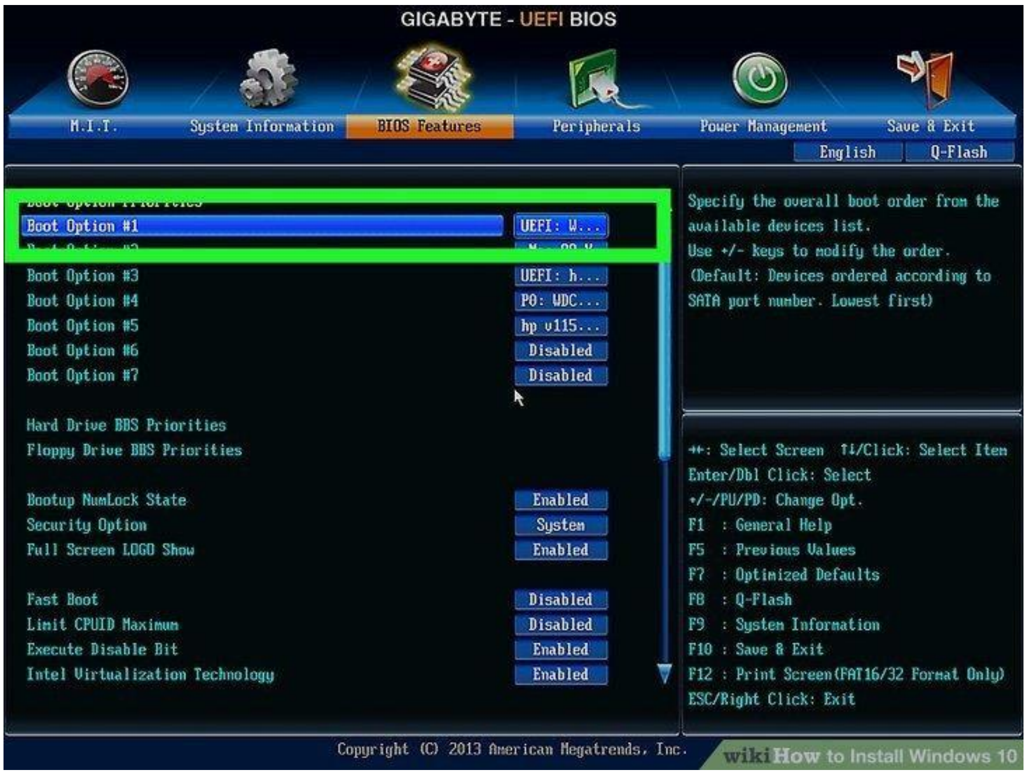


```
Boot Option #3                    UEFI: h...
Boot Option #4                    P0: WDC...
Boot Option #5                    hp v115...
Boot Option #6                    Disabled
Boot Option #7                    Disabled

Hard Drive BBS Priorities
Floppy Drive BBS Priorities

Bootup NumLock State              Enabled
Security Option                   System
Full Screen LOGO Show             Enabled
```

Step-12:

Go to the Boot tab. You'll use the arrow keys to select it.

The **Boot** tab may instead say **Boot Options** or **Boot Order**, depending on your computer's manufacturer.



```
Boot Option #1                              UEFI: W...
Boot Option #3                              UEFI: h...
Boot Option #4                              P0: WDC...
Boot Option #5                              hp v115...
Boot Option #6                              Disabled
Boot Option #7                              Disabled

Hard Drive BBS Priorities
Floppy Drive BBS Priorities

Bootup NumLock State                        Enabled
Security Option                             System
Full Screen LOGO Show                       Enabled

Fast Boot                                   Disabled
Limit CPUID Maximum                         Disabled
Execute Disable Bit                         Enabled
Intel Virtualization Technology             Enabled
```

```
Specify the overall boot order from the
available devices list.
Use +/- keys to modify the order.
(Default: Devices ordered according to
SATA port number. Lowest first)

→←: Select Screen  ↑↓/Click: Select Item
Enter/Dbl Click: Select
+/-/PU/PD: Change Opt.
F1  : General Help
F5  : Previous Values
F7  : Optimized Defaults
F8  : Q-Flash
F9  : System Information
F10 : Save & Exit
F12 : Print Screen(FAT16/32 Format Only)
ESC/Right Click: Exit
```

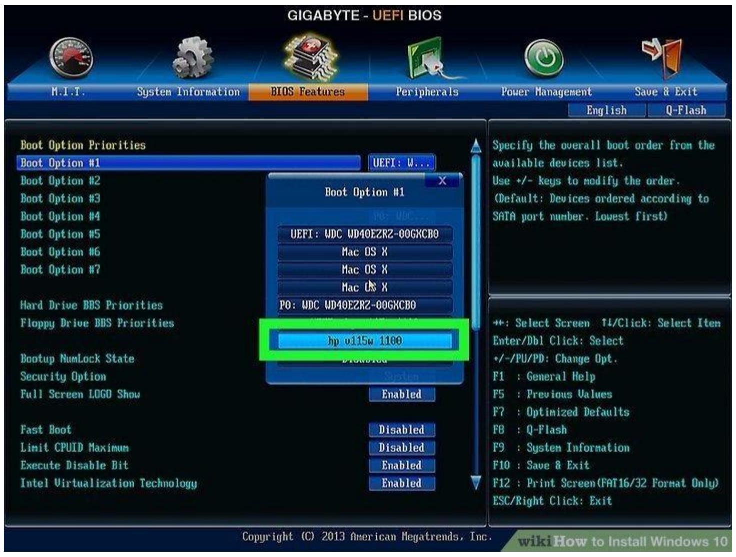
Copyright (C) 2013 American Megatrends, Inc.

wikiHow to Install Windows 10

Step-13:

Select a device from which to boot. You have a couple of options here:

- For a **USB flash drive**, select the **Removable Devices** option.
- For a **disc installation**, select the **CD-ROM Drive** or **Optical Drive** option



```
Boot Option Priorities
Boot Option #1                              UEFI: W...
Boot Option #2
Boot Option #3
Boot Option #4
Boot Option #5
Boot Option #6
Boot Option #7

Hard Drive BBS Priorities
Floppy Drive BBS Priorities

Bootup NumLock State
Security Option
Full Screen LOGO Show                       Enabled

Fast Boot                                   Disabled
Limit CPUID Maximum                         Disabled
Execute Disable Bit                         Enabled
Intel Virtualization Technology             Enabled
```

```
Boot Option #1
  UEFI: WDC WD40EZRZ-00GXCB0
  Mac OS X
  Mac OS X
  Mac OS X
  P0: WDC WD40EZRZ-00GXCB0
  hp v115w 1100
```

```
Specify the overall boot order from the
available devices list.
Use +/- keys to modify the order.
(Default: Devices ordered according to
SATA port number. Lowest first)

→←: Select Screen  ↑↓/Click: Select Item
Enter/Dbl Click: Select
+/-/PU/PD: Change Opt.
F1  : General Help
F5  : Previous Values
F7  : Optimized Defaults
F8  : Q-Flash
F9  : System Information
F10 : Save & Exit
F12 : Print Screen(FAT16/32 Format Only)
ESC/Right Click: Exit
```

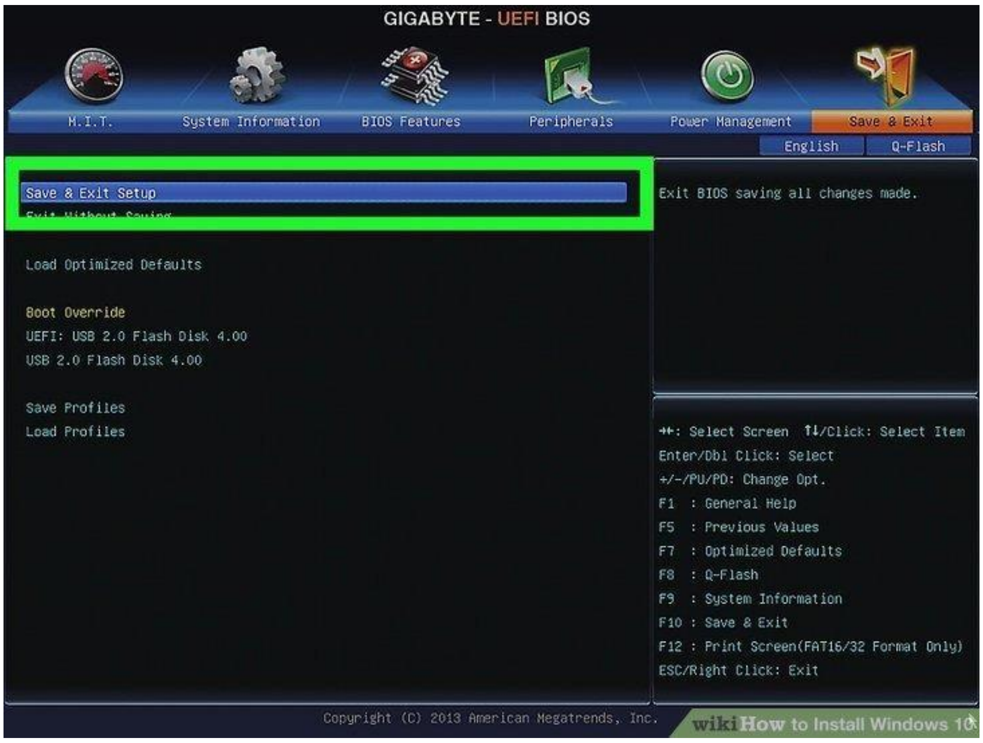
Copyright (C) 2013 American Megatrends, Inc.

wikiHow to Install Windows 10

Step-14:

Press the + key until your boot option is first. Once either **Removable Devices** or **CD-ROM Drive** is at the top of the list, your computer will select your choice as its default boot option.

On some computers, you'll instead press one of the function keys (e.g., **F5** or the arrow keys to navigate an option up to the top of the menu. The key will be listed on the right side of the screen.



Save & Exit Setup
Exit Without Saving

Load Optimized Defaults

Boot Override
UEFI: USB 2.0 Flash Disk 4.00
USB 2.0 Flash Disk 4.00

Save Profiles
Load Profiles

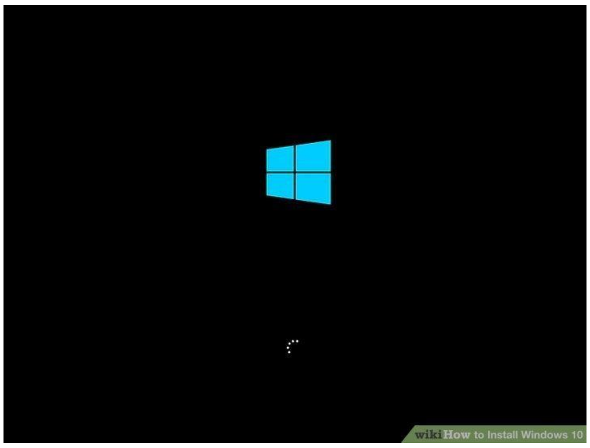
Exit BIOS saving all changes made.

```
→←: Select Screen  ↑↓/Click: Select Item
Enter/Dbl Click: Select
+/-/PU/PD: Change Opt.
F1  : General Help
F5  : Previous Values
F7  : Optimized Defaults
F8  : Q-Flash
F9  : System Information
F10 : Save & Exit
F12 : Print Screen(FAT16/32 Format Only)
ESC/Right Click: Exit
```

Copyright (C) 2013 American Megatrends, Inc.

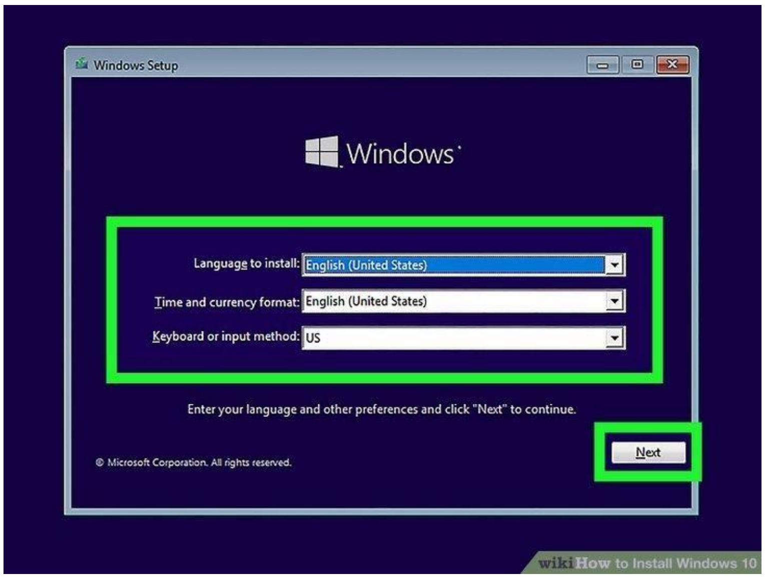
Step-15:

Save your settings. You should see a key prompt (e.g., **F10** at the bottom of the screen that correlates to "Save and Exit". Pressing it will save your settings and restart your computer.



Step-16:

Wait for your computer to restart. Once your computer finishes restarting, you'll see a window here with your geographical data. You're now ready to begin setting up your Windows 10 installation.



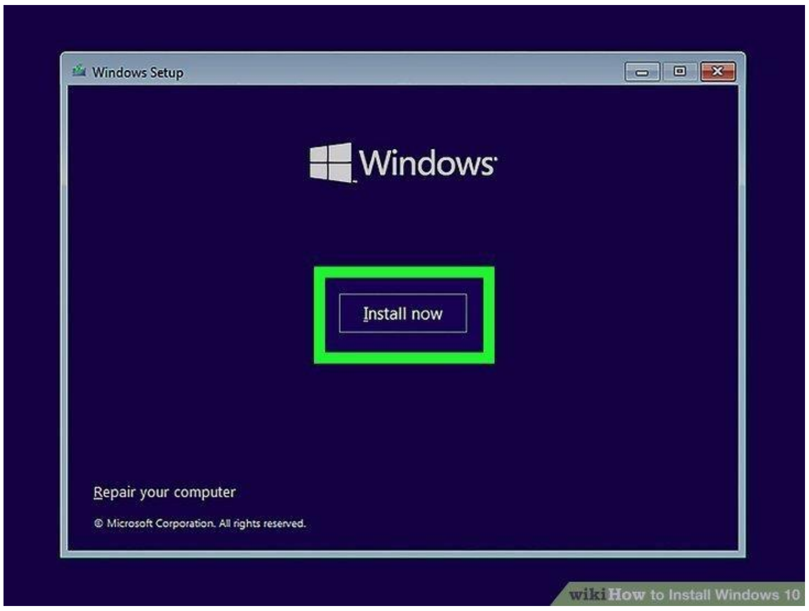
Language to install: English (United States)
Time and currency format: English (United States)
Keyboard or input method: US

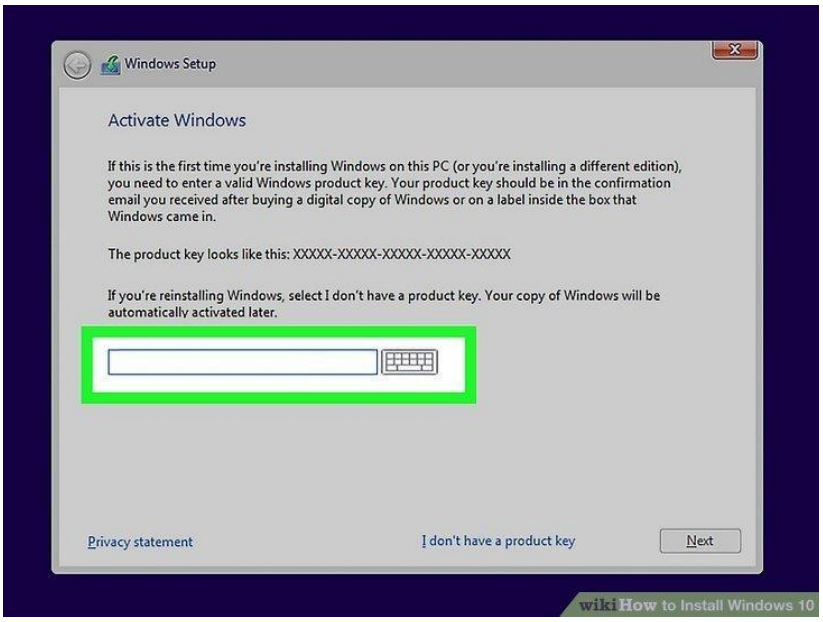
Enter your language and other preferences and click "Next" to continue.

© Microsoft Corporation. All rights reserved.

Step-17:

Click Next when prompted. You can also change the options on this page (e.g., the setup language) before continuing if need be


Windows Setup
Windows
Install now
Repair your computer
© Microsoft Corporation. All rights reserved.
wikiHow to Install Windows 10

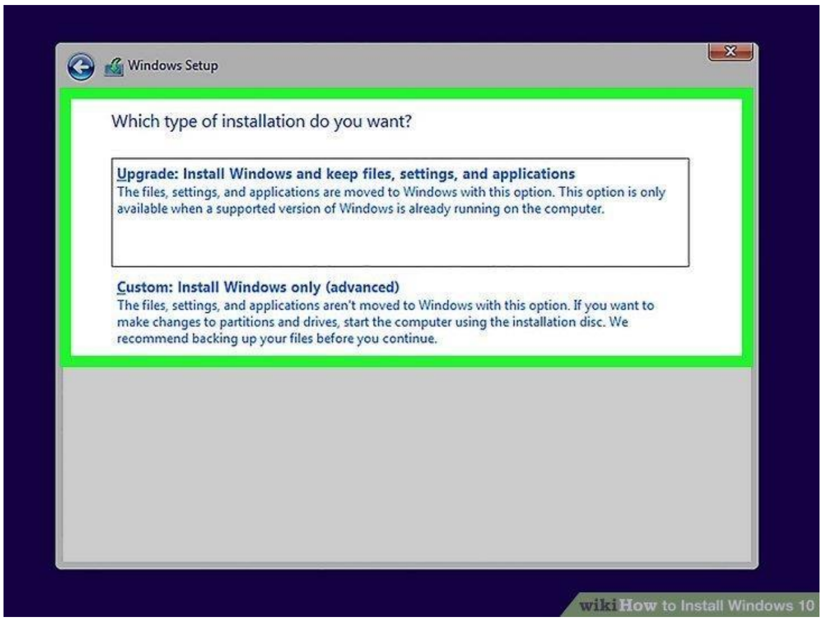

Windows Setup

Activate Windows

If this is the first time you're installing Windows on this PC (or you're installing a different edition), you need to enter a valid Windows product key. Your product key should be in the confirmation email you received after buying a digital copy of Windows or on a label inside the box that Windows came in.

The product key looks like this: XXXXX-XXXXX-XXXXX-XXXXX-XXXXX

If you're reinstalling Windows, select I don't have a product key. Your copy of Windows will be automatically activated later.


Windows Setup

Which type of installation do you want?

Upgrade: Install Windows and keep files, settings, and applications
The files, settings, and applications are moved to Windows with this option. This option is only available when a supported version of Windows is already running on the computer.

Custom: Install Windows only (advanced)
The files, settings, and applications aren't moved to Windows with this option. If you want to make changes to partitions and drives, start the computer using the installation disc. We recommend backing up your files before you continue.

Step-18:

Follow the on-screen instructions to install Windows 10. You'll be asked to perform a few tasks, such as connecting to Wi-Fi and choosing some preferences. Once the installation is complete, you'll have a fresh new installation of Windows 10.

- If you're upgrading from an earlier version of Windows, you'll be asked if you want to upgrade the current operating system or do a custom install. If you choose **Upgrade**, you'll preserve existing apps and files.

Conclusion:

Thus the windows operating system was installed and configured successfully.

Ex.No:02	IMPLEMENTATION OF UNIX COMMAND & SHELL PROGRAMMING
Date:	

AIM:

Acquire operating system skills through UNIX Commands.

UNIX SHELL:

A UNIX shell is a command-line interpreter that provides a user interface for the UNIX operating system. Users direct the operation of the computer by entering commands as text for a command line interpreter to execute or by creating text scripts of one or more such commands

GENERAL COMMANDS:

date	Used to display the current system date andtime.
date +%D	Displays date only
date +%T	Displays time only
date +%Y	Displays the year part of date
date +%H	Displays the hour part of time
cal	Calendar of the current month
cal year	Displays calendar for all months of the specified year
cal month year	Displays calendar for the specified month of the year
who	Login details of all users such as their IP, Terminal No, Username
who am i	Used to display the login details of the user
tty	Used to display the terminal name (or) terminaltype
uname	Displays the Operating System
uname –r	Shows version number of the OS (kernel)
uname –n	Displays domain name of the server
echo "txt"	Displays the given text on the screen
echo $HOME	Displays the user's home directory
bc	Basic calculator. Press Ctrl+d to quit
lp -	file Allows the user to spool a job along with others in a print queue
history	To display the commands used by the user since logon.
Exit	Exit from a process. If shell is the only process then logs out
Clear	Clear the screen

DIRECTORY COMMANDS:

Pwd	Path of the present working directory
Mkdir dir	A directory is created in the name under the current directory
Mkdir dir1 dir2	A number of sub-directories can be created under one stroke
Cd subdir	Change Directory
Cd	To switch to the home directory.
cd ..	To move back to the parent directory
rmdir subdir	Removes an empty sub-directory.

FILE COMMANDS:

cat > filename	To create a file with some contents. At the end press Ctrl+d.(> indicates redirecting output to a file.)
cat filename	Displays the file contents.
cat >> filename	Used to append contents to a file
cp src des	Copy files to given location. If already exists, it will be overwritten
cp –i src des	Warns the user prior to overwriting the destination file
cp –r src des	Copies the entire directory, all its sub-directories and files.
mv old new	To rename an existing file or directory. –i option can also be used
mv f1 f2 f3 dir	To move a group of files to a directory.
rm *	To delete all the files in the directory.
rm –r *	Deletes all files and sub-directories
rm –f *	To forcibly remove even write-protectedfiles
ls	Lists all files and subdirectories (blue colored) in sorted manner.
ls name	To check whether a file or directory exists.
ls name*	Short-hand notation to list out filenames of a specificpattern.
ls –a	Lists all files including hidden files (files beginning with .)
–l	Long listing showing file access rights (read/write/execute-rwx for user/group/others-ugo).
cmp file1 file2	Used to compare two files. Displays nothing if files are empty
wc	Produces the counts of lines (l), words(w), and characters(c).
chmod	Perform file Changes permission for the specified file.
who | wc –l	To terminate the unix session execute the command exit or logout
head	Print the first N number of lines.
tail	Print the last N number of lines.
new file	Splits screen into multiple windows and opens thefile.
finger	Gathers and displays the information about theusers.
grep	Used to search and print specified patterns froma file.
uniq	Fetches one copy of the redundant records writing them tothe standardoutput.
diff	Difference between the two files

CONCLUSION:

Thus the basics of UNIX commands are studied by executing the commands

Ex.No:03(a)	**IMPLEMENTATION OF UNIX SYSTEM CALLS** (getpid, waitpid)
Date:	

AIM:

To write programs using the following system calls of UNIX operating system fork, getpid, waitpid..

ALGORITHM:

Step 1: Start the program.

Step 2: Create a new process (parent process) using fork() system call.

Step 3: Create a identical copy of parent process (child process) using fork () system call

Step 4: Process id is created for both child and parent process using getpid() system call

Step 5: The Child process is made to execute.

Step 6: After the exiting of the child process parent process will getexit.

Step 7: Stop the program.

PROGRAM:

```
#include<stdio.h>

#include<sys/types.h>

#include<sys/wait.h>

#include<unistd.h>

void parent();

void child();

main()

{

int r;
r=fork();
if(r<0)

{
printf("for
k is
empty");

 }

 if(r>0)

parent(r);

else

child(r);

 }

void child()

{

int i;

printf("\nchild starts executing");for(i=0;i<5;i++)

{

        printf("\n child %d is executing %d time",getpid(),i); sleep(i);

}

printf("child %d exit \n",getpid());

}

void parent(child_pid)

{

  printf("\n Parent %d waiting for child %d to start the executing", getpid(),child_pid);printf("\n");

  waitpid(child_pid,NULL,0); printf("Parent %d exit
                    \n",getpid());

}
```

OUTPUT:

```
[com39@localhost ~]$ cd os

 [com39@localhost os]$ gcc ex1a.c

 [com39@localhost os]$ ./a.out fork is empty

Parent 1769 waiting for child 1770 to start the executing fork is empty

child starts executing

child 1770 is executing 0 time

child 1770 is executing 1 time

child 1770 is executing 2 time

child 1770 is executing 3 time

child 1770 is executing 4 time

child 1770 exit

Parent 1769 exit [com39@localhost os]
```

CONCLUSION:

Thus the C program to simulate UNIX system calls fork, getpid, wait has been executed successfully

Ex.No:03(b)	IMPLEMENTATION OF UNIX SYSTEM CALLS (opendir (), readdir(),closedir(), exit())
Date:	

AIM:

To write programs using the following system calls of UNIX operating system opendir(),readdir(), closedir() and exit().

ALGORITHM:

Step 1: Start the program.

Step 2: Create a directory entry in the directory structure and a directory variable.

Step 3: Get the name of the directory to be opened.

Step 4: Using the system call opendir(),open the given directory and store it inthe directory variable.

Step 5: Read the contents of the directory using readdir()system call and print the list of files in the directory.

Step 6: Close the directory after the read and write process.

Step 7: Stop the program.

PROGRAM:

```
#include<stdio.h>

#include<dirent.h>

#include<stdlib.h>

struct dirent *dptr;

int main(int argc,char *argv[])

{

char buff[256]; DIR *dirp;

printf("\n \n enter directory name");

scanf("%s",buff); if((dirp=opendir(buff))==NULL)

{

printf("error");

exit(0);

}

while(dptr=readdir(dirp))

{

printf("%s\n",dptr->d_name);

}

closedir(dirp);

}
```

OUTPUT:

```
[com39@localhost ~]$ cd os

[com39@localhost os]$gcc ex1b.c

[com39@localhost os]$ ./a.out

ENTER A DIRECTORY NAME: oslabpgmsex5b.c

ex5b.c
ex1b.c
..
ex1a.c
ex4a.c
ex2.c
ex3a.c
ex1d.c
ex4b.c
.
ex3b.c
ex1c.c
ex5a.c
[com39@localhost os]$
```

CONCLUSION:

Thus the C program to simulate UNIX system calls opendir, readdir, closedir and exit has beenexecuted successfully.

Ex.No:4a	**CPU SCHEDULING ALGORITHMS –** **FIRST COME FIRST SERVER (FCFS)**
Date:	

AIM

Acquire operating system skills through CPU scheduling algorithms.

ALGORITHM:

Step 1: Start the program.

Step 2: Declare the variables.

Step 3: Get the number of process and their burst time of each process.

Step 4: Calculate the waiting time and turnaround time of each of the processes

Step 5: Calculate average waiting and average turnaround time of all the process.

Step 6: Display waiting time, turnaround time, average waiting time and turnaround time of the processes

Step 7: Stop the program.

PROGRAM:

```
#include<stdio.h>

#include<conio.h>

main()

{

int bt[20], wt[20], tat[20], i, n;

float wtavg, tatavg;

clrscr();

printf("\nEnter the number of processes -- ");

scanf("%d", &n);

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

{

printf("\nEnter Burst Time for Process %d -- ", i);

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

}
wt[0] = wtavg = 0; tat[0] = tatavg = bt[0];
for(i=1;i<n;i++)
{ wt[i] = wt[i-1] +bt[i-1];
tat[i] = tat[i-1] +bt[i];
wtavg = wtavg + wt[i]; tatavg = tatavg + tat[i];
}
printf("\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");
for(i=0;i<n;i++)
printf("\n\t P%d \t\t %d \t\t %d \t\t %d", i, bt[i], wt[i], tat[i]);
printf("\nAverage Waiting Time -- %f", wtavg/n);
printf("\nAverage Turnaround Time -- %f", tatavg/n);
getch();
}
```

INPUT

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

OUTPUT

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

   Average Waiting Time-- 17.000000

   Average Turnaround Time -- 27.000000
```

CONCLUSION:

Thus the C program using FCFS scheduling has been executed successfully.

Ex.No:4b	
Date:	**CPU SCHEDULING ALGORITHMS - SHORTEST JOB FIRST (SJF)**

AIM

Acquire operating system skills through CPU scheduling algorithms.

ALGORITHM:

Step 1: Start the program.
Step 2: Declare the variables.
Step 3: Get the number of process and their burst time of each process.
Step 4: Calculate the waiting time and turnaround time of each of the processes
Step 5: Calculate average waiting and average turnaround time of all the process.
Step 6: Display waiting time, turnaround time, average waiting time and turnaround time of the processes
Step 7: Stop the program.

PROGRAM:

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

INPUT

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

OUTPUT

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

CONCLUSION:

Thus the C program using SJF scheduling has been executed successfully.

Ex.No:4c	**CPU SCHEDULING ALGORITHMS - ROUND ROBIN SCHEDULING ALGORITHM**
Date:	

AIM:

Acquire programming skills for application development in real-world problem solving.

ALGORITHM:

Step 1: Start the program.

Step 2: Declare the variables.

Step 3: Get the number of process and their burst time of each process.

Step 4: Get the quantum time.

Step 5: Schedule the CPU to the first process, after that quantum time is over schedule the second process. This will continue until all processes in the system complete their turn.

Step 6: Draw the chart of the processes as Round Robin manner.

Step 7: Calculate waiting time and turnaround time of each process.

Step 8: Calculate average waiting and average turnaround time of all the process.

Step 9: Display waiting time, turnaround time, average waiting time and turnaround time of the processes.

Step 10: Stop the program.

PROGRAM:

```
#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:

```
[com39@localhost os]$ gcc ex5b.c
[com39@localhost os]$ ./a.out
Enter the number of processes : 3
Enter the process time for process 1 : 4
Enter the process time for process 2 : 6
Enter the process time for process 3 : 9
Enter the time slice : 2
0 P1 2
2 P2 4
4 P3 6
6 P1 8
8 P2 10
10 P3 12
12 P2 14
14 P3 16
16 P3 18
18 P3 19
Average waiting time = 7.333333 msec
Average turnaround time = 13.666667
msec [com39@localhost os]$
```

CONCLUSION:

Thus the C program using Round Robin scheduling has been executed successfully.

Ex.No:5	INTER PROCESS COMMUNICATION
Date:	

AIM:

Learn about Shared Memory and Inter Process Communication

ALGORITHM:

Step 1: Start the program.
Step 2: Create a child process using fork().
Step 3: Now close the read end of the parent using close().
Step 4: Write the data in the pipe using write().
Step 5: Now close the read end of the child using close().
Step 6: Read the data in the pipe using read().
Step 7: Display the string.
Step 8: Stop the program.

PROGRAM:

```
#include<stdio.h>
int main()
{
int fd[2],child; char a[10];
printf("Enter the string to enter into the pipe:\t");
 scanf("%s", a);
pipe(fd); child=fork();
if(!child)
{
close(fd[0]);
write(fd[1],a,5);
wait(0);
}
else
{
close(fd[1]);
read(fd[0],a,5);
printf("The string received from the pipe is %s",a);
}
return 0;
}
```

OUTPUT:

```
[com39@localhost os]$ gcc ex6.c [com39@localhost os]$ ./a.out
Enter the string to enter into the pipe: welcome! The string received from the pipe
is welcome! [com39@localhost os]$
```

CONCLUSION:

Thus the C program for interprocess communication using pipes was executed and the output was verified.

Ex.No:6	SEMAPHORES
Date:	

AIM:

To write a C program to implement semaphores.

ALGORITHM:

Step 1: Start the program

Step 2: Get the number of jobs from the user

Step3: When job1 is processing, job 2 is also starts processing

Step 4: When job 1 enters critical section, next job starts processing

Step 5: When job1 comes out of the critical section, the other job enters the critical section

Step 6: The above 3 steps are performed for various programs

Step 7: End the program.

PROGRAM:

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

{

int i,a=1,h=2,n;

printf("\n Enter the no of jobs"); scanf("%d",&n);

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

{

if(a==1)

{

printf("processing %d    ! \n", i+1);

a++;

}

if(h>1)

{

if(i+2<=n)

{

printf("\n processing %d              ! \n",i+2);

}

printf("\n Process %d Enters Critical section", i+1); printf("\n Process %d Leaves Critical section",
i+1);

} h+1;

}

}
```

OUTPUT:

```
Enter the no of jobs 2

processing 1.     !

processing 2.     !

Process 1 Enters Critical section

Process 1 Leaves Critical section

Process 2 Enters Critical section

Process 2 Leaves Critical section
```

CONCLUSION:

Thus the program to implement semaphores in c was executed and the output was verified.

Ex.No:7	**BANKERS ALGORITHM FOR DEAD LOCK AVOIDANCE**
Date:	

AIM:

To implement banker's algorithm for Dead Lock Avoidance.

ALGORITHM:

Step 1: Start the program.

Step 2: Get the values of resources and processes.

Step 3: Get the avail value.

Step 4: After allocation find the need value.

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

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

Step 7: Else system is not in safety state.

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

Step 9: Or not if we allow the request.

Step 10: Stop the program.

Step 11: End

PROGRAM:

```
#include<  stdio.h>

#include< conio.h>

void main()

{

int allocated[15][15],max[15][15],need[15][15],avail[15],tres[15],work[15],fla g[15];

int pno,rno,i,j,prc,count,t,total;

count=0 clrscr();

printf("\n Enter number of process:");

scanf("%d",&pno);

 printf("\n Enter number of resources:");

scanf("%d",&rno);

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

{

flag[i]=0;

}

printf("\n Enter total numbers of each resources:"); for(i=1;i<= rno;i++) scanf("%d",&tres[i]);

printf("\n Enter Max resources for each process:"); for(i=1;i<= pno;i++)

{

printf("\n for process %d:",i); for(j=1;j<= rno;j++) scanf("%d",&max[i][j

]);

}

printf("\n Enter allocated resources for each process:"); for(i=1;i<= pno;i++)

{

printf("\n for process %d:",i); for(j=1;j<= rno;j++) scanf("%d",&allocate d[i][j]);

}

printf("\n available resources:\n"); for(j=1;j<= rno;j++)

{
```

```
avail[j]=0; total=0; for(i=1;i<= pno;i++)
{
total+=allocated[i][j];
}
avail[j]=tres[j]-total; work[j]=avail[ j];
printf(" %d \t",work[j]);
}
do
{
for(i=1;i<= pno;i++)
{
for(j=1;j<= rno;j++)
{
need[i][j]=max[i][j]-allocated[i][j];
}
}
printf("\n Allocated matrix          Max      need");
for(i=1;i<= pno;i++)
{
printf("\n");
for(j=1;j<= rno;j++)
{
printf("%4d",allocated[i][j]);
}
printf("|");
for(j=1;j<= rno;j++)
{
printf("%4d",max[i][j]);
}
}
printf("|");
for(j=1;j<= rno;j++)
{
printf("%4d",max[i][j]);
}
printf("|");
for(j=1;j<= rno;j++)
{
printf("%4d",need[i][j]);
}
}
prc=0;
for(i=1;i<= pno;i++)
{
if(flag[i]==0)
{
prc=i;
for(j=1;j<= rno;j++)
{
if(work[j]< need[i][j])
{
Prc=0; break;
}
}
}
if(prc!=0) break;
```

```
}
if(prc!=0)
{
printf("\n Process %d completed",i);
count++; printf("\n Available matrix:");
for(j=1;j<= rno;j++)
{
work[j]+=allocate d[prc][j]; allocated[prc][j]= 0; max[prc][j]=0; flag[prc]=1;
printf(" %d",work[j]);
}
}
}while(count!=pno&&prc
!=0);
if(count==pno)
 printf("\nThe system is in a safe state!!");
else
printf("\nThe system is in an unsafe state!!");
getch();
}


printf("|"); for(j=1;j<= rno;j++)
{
printf("%4d",need[i][j]);
}
}
prc=0;
for(i=1;i<= pno;i++)
{
```

```
if(flag[i]==0)
{
prc=i;
for(j=1;j<= rno;j++)
{
if(work[j]< need[i][j])
{
Prc=0; break;
}
}
}
if(prc!=0) break;
}
if(prc!=0)
{
printf("\n Process %d completed",i); count++; printf("\n Available matrix:"); for(j=1;j<= rno;j++)
{
work[j]+=allocate d[prc][j]; allocated[prc][j]= 0; max[prc][j]=0; flag[prc]=1;
printf(" %d",work[j]);
}
}
}while(count!=pno&&prc
!=0); if(count==pno) printf("\nThe system is in a safe state!!"); else
printf("\nThe system is in an unsafe state!!"); getch();
}
```

OUTPUT:

```
Enter number of process: 5

Enter number of resources: 3

Enter total numbers of each resource: 10 5 7

 Enter Max resources for each process: for process 1:7 5 3

for process 2:3 2 2

for process 3:9 0 2

for process 4:2 2 2

for process 5:4 3 3

Enter allocated resources for each process: for process 1:0 1 0 for process 2:3 0 2

for process 3:3 0 2

for process 4:2 1 1

for process 5:0 0 2

available resources: 2 Allocated matrix   Max     3

need    0

0       1       0|      7       5       3|      7       4       3

3       0       2|      3       2       2|      0       2       0

3       0       2|      9       0       2|      6       0       0

2       1       1|      2       2       2|      0       1       1

0       0       2|      4       3       3|      4       3       1

Process 2 completed

Available matrix: 5 3 2 Allocated matrix Max     need

0       1       0|      7       5       3|      7       4       3

0       0       0|      0       0       0|      0       0       0

3       0       2|      9       0       2|      6       0       0

2       1       1|      2       2       2|      0       1       1

0       0       2|      4       3       3|      4       3       1

Process 4 completed
```

```
Available matrix: 7 4 3 Allocated matrix Max     need

0       1       0|      7       5       3|      7       4       3

0       0       0|      0       0       0|      0       0       0

3       0       2|      9       0       2|      6       0       0

0       0       0|      0       0       0|      0       0       0

0       0       2|      4       3       3|      4       3       1

Process 1 completed

Available matrix: 7 5 3 Allocated matrix Max     need

0       0       0|      0       0       0|      0       0       0

0       0       0|      0       0       0|      0       0       0

3       0       2|      9       0       2|      6       0       0

0       0       0|      0       0       0|      0       0       0

0       0       2|      4       3       3|      4       3       1


Process 3 completed

Available matrix: 10 5 5 Allocated matrix Max  need

0       0       0|      0       0       0|      0       0       0

0       0       0|      0       0       0|      0       0       0

0       0       0|      0       0       0|      0       0       0

0       0       0|      0       0       0|      0       0       0

0       0       2|      4       3       3|      4       3       1


Process 5 completed

Available matrix: 10 5 7 The system is in a safe state!!
```

CONCLUSION:

Thus the program to implement banker’s algorithm for Dead Lock Avoidance was executedand the output was verified.

Ex.No:8	
Date:	**DEADLOCK DETECTION ALGORITHM**

AIM:

To implement an algorithm for Dead Lock Detection

ALGORITHM:

Step 1: Start the program.

Step 2: Get the values of resources and processes. Step 3: Get the avail value.

Step 4: After allocation find the need value. Step 5: Check whether it’s possible to allocate.

Step 6: If it is possible then the system is in safe state. Step 7: Else system is not in safety state.

Step 8: If the new request comes then check that the system is insafety. Step 9: Or not if we allow the request.

Step 10: Stop the program. Step 11: End

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 Algorithm ************\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]);
}
{
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");
}
}
}
```

PROGRAM-2

```
#include <bits/stdc++.h>
using namespace std;
int arrmax[100][100];
int alloc[100][100];
int need[100][100];
int avail[100];
int n, r;

void input()
{
  int i, j;
  cout << "Enter the no of Processes\t";
  cin >> n;
  cout << "Enter the no of resource instances\t";
  cin >> r;
  cout << "Enter the Max Matrix\n";
  for (i = 0; i < n; i++)
  {
    for (j = 0; j < r; j++)
```

```
        {
            cin >> arrmax[i][j];
        }
    }
    cout << "Enter the Allocation Matrix\n";
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < r; j++)
        {
            cin >> alloc[i][j];
        }
    }
    cout << "Enter the available Resources\n";
    for (j = 0; j < r; j++)
    {
        cin >> avail[j];
    }
}
void show()
{
    int i, j;
    cout << "Process\t Allocation\t Max\t Available\t";
    for (i = 0; i < n; i++)
    {
        cout << "\nP" << i + 1 << "\t ";
        for (j = 0; j < r; j++)
        {
            cout << alloc[i][j] << " ";
        }
        cout << "\t\t";
        for (j = 0; j < r; j++)
        {
            cout << arrmax[i][j] << " ";
        }
        cout << "\t ";
        if (i == 0)
        {
            for (j = 0; j < r; j++)
                cout << 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] = arrmax[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;
                        }
                        //cout<<"\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)
    {
        cout << "\n\nSystem is in Deadlock and the Deadlock process are\n";
        for (i = 0; i < n; i++)
        {
            cout << "P" << dead[i] << "\t";
        }
    }
    else
    {
        cout << "\nNo Deadlock Occur";
    }
```

```
}
int main()
{
    int i, j;
    cout << "********** Deadlock Detection Algorithm ************\n";
    input();
    show();
    cal();
    return 0;
}
```

OUTPUT:

```
********** Deadlock Detection Algorithm ************

Enter the no of Processes       3

Enter the no of resource instances      3

Enter the Max Matrix

3 6 8

4 3 3

3 4 4

Enter the Allocation Matrix

3 3 3

2 0 3

1 2 4

Enter the available Resources

1 2 0

Process  Allocation      Max     Available

P1          3 3 3       3 6 8     1 2 0

P2          2 0 3       4 3 3

P3          1 2 4       3 4 4


system is in Deadlock and the Deadlock process are

P0      P1      P2
```

CONCLUSION:

Thus the program to implement banker’s algorithm for Dead Lock Detection was executedand the output was verified.

Ex.No:9	
Date:	**IMPLEMENTATION OF THREADING & SYNCHRONIZATON**

AIM:

To learn about Threads and synchronization applications.

ALGORITHM:

Step 1: Start the program.

Step 2: Create number of threads**.**

Step 3: Counter is maintained for the user**.**

Step 4: Enter the logs of the jobs to start**.**

Step 5: Stop the program**.**

PROGRAM:

```
#include<st dio.h>

#include<str ing.h>

#include<pt hread.h>

#include<st dlib.h>

#include<un istd.h> pthread_t tid[2];

int counter;

void* doSomeThing(void *arg)

{

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); 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));

i++;

}

pthread_join(tid[0], NULL);

pthread_join(tid[1], NULL);

return 0;

}
```

OUTPUT:

```
Job 1 started

Job 1 finished

Job 2 started

Job 2 finished
```

CONCLUSION:

Thus the C program for threading and synchronization applications was created andexecuted successfully.

Ex.No:10	
Date:	**IMPLEMENTATION OF PAGING TECHNIQUE**

AIM:

To learn about paging techniques in memory management.

ALGORITHM

Step 1: Start the program.

Step 2: Input the pagesize and initialize memsize of page to 15.

Step 3: Compute number of pages by dividing memsize by pagesize.

Step 4: For the computed pages, give the page frames values.

Step 5: Get the logical address value. Compute frameno and offset.

Step 6: Using the page frameno and offset, Generate the physical address.

Step 7: Stop the program.

PROGRAM

```
#include<stdio.h>
 void main()
{
int memsize=15;
int pagesize,nofpage;
int p[100];
int frameno,offset;
int logadd,phyadd;
int i;
int choice=0;
printf("\nYour memsize is %d ",memsize);
printf("\nEnter page size:");
scanf("%d",&pagesize);
nofpage=memsize/pagesize;
for(i=0;i<nofpage;i++)
{
printf("\nEnter the frame of page%d:",i+1);
scanf("%d",&p[i]);
}
do
{
printf("\nEnter a logical address:");
scanf("%d",&logadd);
frameno=logadd/pagesize;
offset=logadd%pagesize;
phyadd=(p[frameno]*pagesize)+offset;
printf("\nPhysical address is:%d",phyadd);
printf("\nDo you want to continue(1/0)?:");
scanf("%d",&choice);
}while(choice==1);
}
```

Output:

```
Your memsize is 15 Enter page size: 5

Enter the frame of page1:2

 Enter the frame of page2:4

Enter the frame of page3:7

Enter a logical address: 3

Physical address is: 13

Do you want to continue (1/0)? : 1

 Enter a logical address: 1

Physical address is: 11

Do you want to continue (1/0)? : 0
```

CONCLUSION:

Thus the c program for implementing paging technique of memory management using first fit was executed and the output was verified.

Ex.No:11a	MEMORY MANAGEMENT SCHEME USING FIRST FIT
Date:	

AIM:

To learn about Memory management schemes.

ALGORITHM:

Step 1: Include the necessary header files required.

Step 2: Declare the variables needed.

Step 3: Read the number of blocks and the size of each blocks.

Step 4: Read the number of process and the size of each process.

Step 5: Check if the process size is less than or equal to block size.

Step 6: If yes, assign the corresponding block to the current process.

Step 7: Else print the current process is not allocated.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#define max 25
void main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=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)
{
ff[i]=j; highest=temp;
}
}
}
frag[i]=highest; bf[ff[i]]=1; highest=0;
}
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 NoFile Size        Block No        Block Size      Fragment
1       1       3       7       6
2       4       1       5       1
```

CONCLUSION:

Thus the program to implement memory management using first fit was executed and the output was verified

Ex.No:11b	
Date:	**MEMORY MANAGEMENT SCHEME USING WORST FIT**

AIM:

Learn about Memory management schemes.

ALGORITHM:

Step 1: Include the necessary header files required.

Step 2: Declare the variables needed.

Step 3: Read the number of blocks and the size of each blocks.

Step 4: Read the number of process and the size of each process.

Step 5: Check if the process size is less than or equal to block size.

Step 6: If yes, assign the corresponding block to the current process.

Step 7: Else print the current process is not allocated**.**

PROGRAM

```
#include<stdio.h>

#include<conio.h> #define max 25

void main()

{

int frag[max],b[max],f[max],i,j,nb,nf,temp; 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
```

CONCLUSION:

Thus the program to implement memory management using worst fit was executed and the output was verified.

Ex.No:11c	**MEMORY MANAGEMENT SCHEME USING BEST FIT**
Date:	

AIM:

To learn about Memory management schemes using best fit.

ALGORITHM:

Step 1: Include the necessary header files required.

Step 2: Declare the variables needed.

Step 3: Read the number of blocks and the size of each blocks.

Step 4: Read the number of process and the size of each process.

Step 5: Arrange both the process and block size in an order.

Step 6: Check if the process size is less than or equal to block size.

Step 7: If yes, assign the corresponding block to the current process.

Step 8: Else print the current process is not allocated.

PROGRAM:

```
#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
```

CONCLUSION:

Thus the program to implement memory management using best fit was executed and the output was verified.

Ex.No:12a	**IMPLEMENTATION OF PAGE REPLACEMENT ALGORITHMS (LFU)**
Date:	

AIM:

To learn about Page Replacement Algorithms.

ALGORITHM

Step 1: Start the program.

Step 2: Input the total length of reference strings, number of frames and individual reference strings from the user.

Step 3: For each reference string value, check its availability in the page frames. Step 4: If found, increment to next reference string.

Step 5: When the page is not in the frames, increment the number of page fault and remove the page whose page frequency is least in the reference strings.

Step 6: Display the number of faults.

Step 7: Stop the program**.**

PROGRAM

```
#include<stdio.h>

#include<conio.h>

void main(){

int rs[50], i, j, k, m, f, cntr[20], a[20], min, pf=0;

clrscr();

printf("\nEnter number of page references -- ");

 scanf("%d",&m);

printf("\nEnter the reference strings -- ");

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

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

  printf("\nEnter the available no. of frames -- "); scanf("%d",&f);

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

  { cntr[i]=0;

  a[i]=-1; }

  printf("\n The Page Replacement Process is-- \n");

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

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

  if(rs[i]==a[j])

  { cntr[j]++;

  break;

   }

  if(j==f)

  {

  min = 0; for(k=1;k<f;k++) if(cntr[k]<cntr[min]) min=k;

  a[min]=rs[i]; cntr[min]=1; pf++;

  }

  printf("\n"); for(j=0;j<f;j++) printf("\t%d",a[j]); if(j==f)
printf("\tPF No.-- %d",pf);

}

printf("\n\n Total number of page faults -- %d",pf);

getch();

}
```

Output

```
Enter number of page references – 10

Enter the reference string --1 2 3 4 5 2 5 2 5 1 Enter the available no. of frames – 3

The Page Replacement Process is –

1       -1   -1      PF No. 1

1       2    -1      PF No. 2

1       2    3       PF No. 3

4       2    3       PF No. 4

5       2    3       PF No. 5

5       2    3       PF No. 6

5       2    3        PF No. 7


PF No. 8

Total number of pagefaults      8
```

CONCLUSION:

Thus the program for Page replacement algorithms for LRU was executed and the output was verified.

Ex.No:12b	IMPLEMENTATION OF PAGE REPLACEMENT ALGORITHMS (FIFO)
Date:	

AIM:

To learn about Page Replacement Algorithms.

ALGORITHM

Step 1: Start the program.

Step 2: Input the total length of reference strings, number of frames and individualreference strings from the user.

Step 3: For each reference string value, check its availability in the page frames.

Step 4: If found, increment to next reference string.

Step 5: When the page is not in the frames, increment the number of page fault andremove the page that comes first in the FIFO algorithm.

Step 6: Display the number of faults.

Step 7: Stop the program.

PROGRAM

```
#include<stdio.h>

#include<conio.h>

void main()

{

int i, j, k, f, pf=0, count=0, rs[25], m[10], n; clrscr();

printf("\n Enter the length of reference string -- ");

scanf("%d",&n);

printf("\n Enter the reference string -- ");

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

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

printf("\n Enter no. of frames -- ");

 scanf("%d",&f);

for(i=0;i<f;i++) m[i]=-1;

printf("\n The Page Replacement Process is -- \n"); for(i=0;i<n;i++)

{

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

{

if(m[k]==rs[i]) break;

}

if(k==f)

{

m[count++]=rs[i];

pf++;

}

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

printf("\t%d",m[j]);

if(k==f)
printf("\tPF No. %d",pf);

printf("\n"); if(count==f) count=0;

}

printf("\n The number of Page Faults using FIFO are-- %d",pf);

getch();

}
```

OUTPUT

```
Enter the length of reference string – 20

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

The Page Replacement Process is –

7       -1   -1      PF No. 1

7        0   -1      PF No. 2

7        0    1      PF No. 3

2        0    1      PF No. 4

2        0    1

2        3    1      PF No. 5

2        3    0      PF No. 6

4        3    0      PF No. 7

4        2    0      PF No. 8

4        2    3      PF No. 9

0        2    3      PF No. 10

0        2    3

0        2    3

0        1    3      PF No. 11

0        1    2

0        1    2

7        1    2      PF No. 13

7        0    2      PF No. 14

7        0    1      PF No. 15

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

CONCLUSION:

Thus the program for Page replacement algorithms for FIFO was executed and the output was verified.

Ex.No:12c	**IMPLEMENTATION OF PAGE REPLACEMENT ALGORITHMS (LRU)**
Date:	

AIM:

To learn about Page Replacement Algorithms**.**

ALGORITHM

Step 1: Start the program.

Step 2: Input the total length of reference strings, number of frames and individual reference strings from the user.

Step 3: For each reference string value, check its availability in the page frames.

Step 4: If found, increment to next reference string.

Step 5: When the page is not in the frames, increment the number of page fault and remove the page that is used very least in the reference strings.

Step 6: Display the number of faults.

Step 7: Stop the program.

PROGRAM

```
#include<stdio.h> #include<conio.h> void main()

{

int i, j , k, min, rs[25], m[10], count[10], flag[25], n, f, pf=0, next=1; clrscr();

printf("Enter the length of reference strings -- "); scanf("%d",&n);

printf("Enter the reference strings -- "); for(i=0;i<n;i++)

{

scanf("%d",&rs[i]); flag[i]=0;

}

printf("Enter the number of frames -- "); scanf("%d",&f);

for(i=0;i<f;i++)
{
         scanf("%d",&rs[i]); flag[i]=O;

        printf("Enter the number of frames -- ");

        scanf("%d",&f); for(i=O;i<f;i++)


}
```

OUTPUT

```
Enter the length of reference string -- 20

Enter the reference string -- 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1 Enter the number of frames -- 3

The Page Replacement process is --

                    7      -1        -1                 PF No. – 1
                    7      0         -1                 PF No. – 2
                    7      0         1                  PF No. – 3
                    2      0         1                  PF No. – 4
                    2      0         1
                    2      0         3                  PF No. – 5
                    2      0         3
                    4      0         3                  PF No. – 6
                    4      0         2                  PF No. – 7
                    4      3         2                  PF No. – 8
                    0      3         2                  PF No. – 9
                    0      3         2
                    0      3         2
                    1      3         2                  PF No. – 10
                    1      3         2
                    1      0         2                  PF No. – 11
                    1      0         2
                    1      0         7                  PF No. – 12
                    1      0         7
                    1      0         7

The number of page faults using LRU are 12
```

CONCLUSION:

Thus the program for Page replacement algorithms for LRU was executed and the output was verified.

Ex.No:13a	IMPLEMENTATION OF FILE ORGANIZATION TECHNIQUES (SINGLE LEVEL FILE)
Date:	

AIM:

To learn about File Organization Techniques.

ALGORITHM

Step 1: Start the program.

Step 2: Create a structure for building a directory with dname and fname as inputs.

Step 3: Get the name of the new directory being created.

Step 4: For the generated directory, create individual files with distinct names.

Step 5: Search for the file created in the directory using its name. if found returns filefound.

Step 6: Display the total files present in the directory using display function.

Step 7: Delete a file by specifying its name using the delete function.

Step 8: Stop the program.

PROGRAM

```
#include<stdio.h>
 struct
{
char dname[10],
fname[10][10];
int fcnt;
}dir;
void main()
{
int i,ch; char f[30]; clrscr(); dir.fcnt = 0;
printf("\nEnter name of directory --");
scanf("%s", dir.dname);
while(1)
{
printf("\n\n1. Create File\t2. Delete File\t3. Search File \n4. Display Files\t5. Exit");
printf("\nEnter your choice -- ");

  switch(ch)

  {

  case 1:

  printf("\nEnter the name of the file -- "); scanf("%s",dir.fname[dir.fcnt]); dir.fcnt++;

  break; case 2:

  printf("\nEnter the name of the file -- "); scanf("%s",f);

  for(i=0;i<dir.fcnt;i++)

  {

  if(strcmp(f, dir.fname[i])==0)

  {

  printf("File %s is deleted ",f); strcpy(dir.fname[i],dir.fname[dir.fcnt-1]); break;

  }

  }

  if(i==dir.fcnt)

  printf("File %s not found",f);

  else

  dir.fcnt--; break;

  case 3:

  printf("\nEnter the name of the file -- "); scanf("%s",f);
  for(i=0;i<dir.fcnt;i++)

  {

  if(strcmp(f, dir.fname[i])==0)

  {

  printf("File %s is found ", f); break;

  }

  }

  if(i==dir.fcnt)

  printf("File %s not found",f); break;

  case 4:

  if(dir.fcnt==0) printf("\nDirectory Empty"); else

  {

  printf("\nThe Files are -- "); for(i=0;i<dir.fcnt;i++) printf("\t%s",dir.fname[i]);

  }

  break; default:

  exit(0);

  }

  }
```

OUTPUT

```
Enter name of directory –CSE

1.Create File   2. Delete File 3. Search File 4.Display Files                5. Exit  Enter your choice –
1 Enter the name of the file –A

1.Create File   2. Delete File     3. Search File 4.Display Files              5. Exit  Enter your choice –
1 Enter the name of the file –B

1.Create File   2. Delete File     3. Search File 4.Display Files              5. Exit  Enter your choice –
1 Enter the name of the file –C

1.Create File   2. Delete File     3. Search File 4.Display Files              5. Exit  Enter your choice –
4 The Files are -- A B C

1.Create File   2. Delete File     3. Search File 4.Display Files              5. Exit  Enter your choice –
3 Enter the name of the file – ABC

File ABC not found

1.Create File   2. Delete File     3. Search File 4.Display Files              5. Exit  Enter your choice –
2 Enter the name of the file – B

File B is deleted

1.Create File   2. Delete File     3. Search File 4.Display Files              5. Exit  Enter your choice –
3 Enter the name of the file – C

File C is found

1.Create File   2. Delete File     3. Search File 4.Display Files              5. Exit  Enter your choice –
5
```

CONCLUSION

Thus the program to implement file organization technique in single level

Ex.No:13b	**IMPLEMENTATION OF FILE ORGANIZATION TECHNIQUES (TWO LEVEL FILE)**
Date:	

AIM:

To learn about File Organization Techniques.

ALGORITHM:

Step 1: Start the program.

Step 2: Create a structure for building a directory with dname and fname as inputs.

Step 3: Create a new directory. Create another new directory inside it or create a file.

Step 4: For the generated directory, create individual files with distinct names.

Step 5: Search for the file created in the directory using its name. if found returns file found.

Step 6: Display the total files present in the directory using display function.

Step 7: Delete a file by specifying its name using the delete function.

Step 8: Stop the program**.**

PROGRAM

```
#include<stdio.h>

struct

{

char dname[10],fname[10][10]; int fcnt;

}dir[10]; void main()

{

int i,ch,dcnt,k; char f[30], d[30]; clrscr();

dcnt=0; while(1)

{

printf("\n\n1. Create Directory\t2. Create File\t3. Delete File");

printf("\n4. Search File\t\t5. Display\t6. Exit \t Enter your choice --        ");

scanf("%d",&ch);

switch(ch)

case 1:

printf("\nEnter name of directory --");

scanf("%s", dir[dcnt].dname); dir[dcnt].fcnt=0;

dcnt++;

printf("Directory created"); break;

case 2:

printf("\nEnter name of the directory -- ");

scanf("%s",d);

for(i=0;i<dcnt;i++) if(strcmp(d,dir[i].dname)==0)

{

printf("Enter name of the file -- ");

scanf("%s",dir[i].fname[dir[i].fcnt]);

dir[i].fcnt++;

printf("File created"); break;
```

```
}
if(i==dcnt)
printf("Directory %s not found",d); break;
case 3:
printf("\nEnter name of the directory -- "); scanf("%s",d);
for(i=0;i<dcnt;i++)
{
if(strcmp(d,dir[i].dname)==0)
{
printf("Enter name of the file -- "); scanf("%s",f); for(k=0;k<dir[i].fcnt;k++)
{
if(strcmp(f, dir[i].fname[k])==0)
{
printf("File %s is deleted ",f); dir[i].fcnt--;
strcpy(dir[i].fname[k],dir[i].fname[dir[i].fcnt]); goto jmp;
}
}
printf("File %s not found",f); goto jmp;
}
}
printf("Directory %s not found",d); jmp:
break;
case 4:
printf("\nEnter name of the directory -- "); scanf("%s",d);
for(i=0;i<dcnt;i++)
{
if(strcmp(d,dir[i].dname)==0)
{
```

```
printf("Enter the name of the file -- ");

scanf("%s",f); for(k=0;k<dir[i].fcnt;k++)

{

if(strcmp(f, dir[i].fname[k])==0)

{

printf("File %s is found ",f); goto jmp1;

}

}

printf("File %s not found",f); goto jmp1;

}

}

printf("Directory %s not found",d); jmp1:

break; case 5:

if(dcnt==0)

printf("\nNo Directory's "); else

{

printf("\nDirectory\tFiles"); for(i=0;i<dcnt;i++)

{

printf("\n%s\t\t",dir[i].dname); for(k=0;k<dir[i].fcnt;k++) printf("\t%s",dir[i].fname[k]);

}

}

break; default:

exit(0);

}

}

getch();

}
```

OUTPUT

```
1. Create Directory     2.Create File 3.Delete File
4. Search File 5.Display        6.Exit Enter Your Choice—1 Enter name of the Directory—DIR1
Directory Created.

1. Create Directory     2.Create File 3.Delete File
4. Search File 5.Display        6.Exit Enter Your Choice—1 Enter name of the Directory—DIR2
Directory Created.

1. Create Directory     2.Create File 3.Delete File
4. Search File 5.Display        6.Exit Enter Your Choice—2 Enter name of the Directory—DIR1
Enter name of the File—A1 File Created.

1. Create Directory     2.Create File 3.Delete File
4. Search File 5.Display        6.Exit Enter Your Choice—2 Enter name of the Directory—DIR1
Enter name of the File—A2 File Created.

1. Create Directory     2.Create File 3.Delete File
4. Search File 5.Display        6.Exit Enter Your Choice—2 Enter name of the Directory—DIR2
Enter name of the File—B1 File Created.

1. Create Directory     2.Create File 3.Delete File
4. Search File Directory Files  5.Display        6.Exit Enter Your Choice—5
DIR1     A1  A2
DIR2     B1

1. Create Directory     2.Create File 3.Delete File
```

```
4. Search File 5.Display        6.Exit Enter Your Choice—4 Enter name of the Directory—DIR3

Directory not found.


1. Create Directory     2.Create File 3.Delete File

4. Search File 5.Display        6.Exit Enter Your Choice—3 Enter name of the Directory—DIR1

Enter name of the File—A2 File A2 Deleted.


1. Create Directory     2.Create File 3.Delete File

4. Search File 5.Display        6.Exit  Enter Your Choice—6
```

CONCLUSION:

Thus the program to implement file organization technique in Two Level Directory was executed and the output was verified.

Ex.No:14a	**IMPLEMENTATION OF FILE ALLOCATION TECHNIQUES**
Date:	**(SEQUENTIAL FILE ALLOCATION)**

AIM:

To learn about File Allocation Techniques.

ALGORITHM:

Step 1: Start the program.

Step 2: Create a structure for building a fileTable.

Step 3: Input enter the number of files, file name, starting block and number ofblocks.

Step 4: Subsequent blocks are allocated one after the other in sequential approach.Search for the file created. if found returns file found. Else returns not found.

Step 5: Display the file allocation details.

Step 6: Stop the program.

PROGRAM:

```
 #include < stdio.h>
#include<conio.h>
void main()
{
int f[50], i, st, len, j, c, k, count = 0;
clrscr();
for(i=0;i<50;i++)
f[i]=0;
printf("Files Allocated are : \n");
x: count=0;
printf(―Enter starting block and length of
                    files: ‖);
scanf("%d%d", &st,&len);
for(k=st;k<(st+len);k++)
if(f[k]==0)
count++;
if(len==count)
{
for(j=st;j<(st+len);j++)
if(f[j]==0)
{
f[j]=1;
printf("%d\t%d\n",j,f[j]);
}
if(j!=(st+len-1))
printf(‖ The file is allocated to disk\n");
}
else
printf(‖ The file is not allocated \n");
printf("Do you want to enter more file(Yes -
                    1/No - 0)");
scanf("%d", &c);
if(c==1)
goto x;
else
exit();
getch();}
```

OUTPUT:

```
Files Allocated are :

Enter starting block and length of files: 14 3

14 1

15 1

16 1

The file is allocated to disk

Do you want to enter more file(Yes - 1/No - 0)1

Enter starting block and length of files: 14 1

The file is not allocated

Do you want to enter more file(Yes - 1/No - 0)1

Enter starting block and length of files: 14 4

The file is not allocated Do you want to enter more file(Yes - 1/No - 0)0
```

CONCLUSION:

Thus the program to implement sequential file allocation was executed and the output was verified.

Ex.No:14b	**IMPLEMENTATION OF FILE ALLOCATION TECHNIQUES**
Date:	**(INDEXED FILE ALLOCATION)**

AIM:

To learn about File Allocation Techniques.

ALGORITHM:

Step 1: Start the program.

Step 2: Create a structure for building a fileTable.

Step 3: Input enter the number of files, file name, starting block and number of blocks.

Step 4: Subsequent blocks are allocated as per the index details of the file in indexed approach. Search for the file created. if found returns file found. Else returns not found.

Step 5: Display the file allocation details.

Step 7: Stop the program.

PROGRAM

```
#include<stdio.h>

#include<conio.h>

struct fileTable

{

char name[20];

int nob, blocks[30];

}ft[30];

void main()

{

int i, j, n; char s[20]; clrscr();

printf("Enter no of files:"); scanf("%d",&n); for(i=0;i<n;i++)

{

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

scanf("%s",ft[i].name);

printf("Enter no of blocks in file %d : ",i+1); scanf("%d",&ft[i].nob);

printf("Enter the blocks of the file : "); for(j=0;j<ft[i].nob;j++) scanf("%d",&ft[i].blocks[j]);

}

printf("\nEnter the file name to be searched -- "); scanf("%s",s);

for(i=0;i<n;i++) if(strcmp(s, ft[i].name)==0) break;

if(i==n)

printf("\nFile Not Found"); else

{

printf("\nFILE NAME \t NO OF BLOCKS \t BLOCKS OCCUPIED");

printf("\n %s\t\t%d\t",ft[i].name,ft[i].nob); for(j=0;j<ft[i].nob;j++)

printf("%d, ",ft[i].blocks[j]);

}

getch();}
```

OUTPUT:

```
Enter no of files : 2 Enter file 1: A

Enter no of blocks in file 1: 4

Enter the blocks of the file 1: 12 23 9 4 Enter file 2: G

Enter no of blocks in file 2: 5

Enter the blocks of the file 2: 88 77 66 55 44 Enter the file to be searched : G

FILE NAME   NO OF BLOCKS        BLOCKS OCCUPIED G                  5
            88,77,66,55,44,
```

CONCLUSION

Thus the C program for indexed file allocation has been executed successfully.

Ex.No:14c	**IMPLEMENTATION OF FILE ALLOCATION TECHNIQUES**
Date:	**(LINKED FILE ALLOCATION)**

AIM:

To learn about File Allocation Techniques.

ALGORITHM:

Step 1: Start the program.

Step 2: Create a structure for building a fileTable.

Step 3: Input enter the number of files, file name, starting block and number of blocks.

Step 4: Subsequent blocks are allocated as per pointer value to next block of the file in linked approach. Search for the file created. if found returns file found. Else returns not found.

Step 6: Display the file allocation details.

Step 7: Stop the program.

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
struct fileTable
{
char name[20]; int nob;
struct block *sb;
}ft[30];
struct block
{
int bno;
struct block *next;
};
void main()
{
int i,j,n; char s[20];
struct block *temp; clrscr();
printf("\n Enter No. of Files:");
scanf("%d",&n); for(i=0;i<n;i++) {
printf("\n Enter File Name %d :",i+1);
 scanf("%s",ft[i].name);
printf("Enter No. of Blocks in File %d :",i+1);
 scanf("%d",&ft[i].nob);
ft[i].sb=(struct block*)malloc(sizeof(struct block)); temp=ft[i].sb;
printf("\n Enter the blocks of the File:");
scanf("%d",&temp->bno);
temp->next=NULL; for(j=1;j<ft[i].nob;j++) {

temp->next=(struct block*)malloc(sizeof(struct block)); temp=temp->next;

scanf("%d",&temp->bno);

}

temp->next=NULL;

}

printf("\n Enter the File Name to be Searched--");

 scanf("%s",s);

for(i=0;i<n;i++) if(strcmp(s,ft[i].name)==0) break;

if(i==n)

printf("\n File not Found"); else

{

printf("\n File Name \t No. of Blocks \t Blocks Occupied"); printf("\n %s \t %d
\t",ft[i].name,ft[i].nob);

temp=ft[i].sb; for(j=0;j<ft[i].nob;j++)

{

printf("%d ->",temp->bno); temp=temp->next;

}

 }

getch();

}
```

OUTPUT:

```
Enter no of files: 2 Enter file 1: A

Enter no of blocks in file1: 4

Enter the blocks of the file 1: 12 23 9 4 Enter file 2: G

Enter no of blocks in file2: 5

Enter the blocks of the file 2: 88 77 66 55 44 Enter the file to be searched: G

FILE NAME   NO OF BLOCKS          BLOCKS OCCUPIED

G           5      88-> 77-> 66-> 55-> 44->
```

CONCLUSION

Thus the C program for linked file allocation has been executed successfully.

Ex.No:15a	IMPLEMENTATION OF DISK SCHEDULING ALGORITHM (SSTF)
Date:	

AIM:

To acquire operating system skills through Disk Scheduling algorithms**.**

ALGORITHM:

Step 1: Start the program.

Step 2: Declare the variables.

Step 3: Get the number of Request.

Step 4: Initiatlize the head position

Step 5: Display the total number of head position

Step 6: Stop the programme

PROGRAM:

```
#include<stdio.h>

#include<stdlib.h>

int main()

{

    int RQ[100],i,n,TotalHeadMoment=0,initial,count=0;

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

    scanf("%d",&n);

    printf("Enter the Requests sequence\n");

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

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

    printf("Enter initial head position\n");

    scanf("%d",&initial);


    // logic for sstf disk scheduling


        /* loop will execute until all process is completed*/

    while(count!=n)

    {

        int min=1000,d,index;

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

        {

           d=abs(RQ[i]-initial);

           if(min>d)

           {

               min=d;

               index=i;

           }
```

```
            }
        TotalHeadMoment=TotalHeadMoment+min;
        initial=RQ[index];
        // 1000 is for max
        // you can use any number
        RQ[index]=1000;
        count++;
    }
        printf("Total head movement is %d",TotalHeadMoment);
    return 0;
}
```

Output:

```
Enter the number of Request

8

Enter Request Sequence

95 180 34 119 11 123 62 64

Enter initial head Position

50

Total head movement is 236
```

CONCLUSION:

Thus the program to implement file allocation strategies using c program was executed and the output was verified.

Ex.No:15b	IMPLEMENTATION OF DISK SCHEDULING ALGORITHM (Scan disk scheduling)
Date:	

AIM:

To acquire operating system skills through Disk Scheduling algorithms.

ALGORITHM:

Step 1: Start the program.

Step 2: Declare the variables.

Step 3: Get the number of Request.

Step4: Get the disk size

Step 5: Initiatlize the head position

Step 6: Display the total number of head position

Step 7: Stop the programme

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int RQ[100],i,j,n,TotalHeadMoment=0,initial,size,move;
    printf("Enter the number of Requests\n");
    scanf("%d",&n);
    printf("Enter the Requests sequence\n");
    for(i=0;i<n;i++)
     scanf("%d",&RQ[i]);
    printf("Enter initial head position\n");
    scanf("%d",&initial);
    printf("Enter total disk size\n");
    scanf("%d",&size);
    printf("Enter the head movement direction for high 1 and for low 0\n");
    scanf("%d",&move);
        // logic for Scan disk scheduling
            /*logic for sort the request array */
    for(i=0;i<n;i++)
    {
        for(j=0;j<n-i-1;j++)
        {
            if(RQ[j]>RQ[j+1])
            {
                int temp;
                temp=RQ[j];
                RQ[j]=RQ[j+1];
                RQ[j+1]=temp;
            }
        }
    }

    int index;
    for(i=0;i<n;i++)
    {
        if(initial<RQ[i])
        {
            index=i;
            break;
        }
    }
       // if movement is towards high value
    if(move==1)
    {
        for(i=index;i<n;i++)
        {
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
            initial=RQ[i];
        }
        // last movement for max size
        TotalHeadMoment=TotalHeadMoment+abs(size-RQ[i-1]-1);
        initial = size-1;
        for(i=index-1;i>=0;i--)
        {
             TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
```

```
            initial=RQ[i];
                  }
    }
    // if movement is towards low value
    else
    {
        for(i=index-1;i>=0;i--)
        {
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
            initial=RQ[i];
        }
        // last movement for min size
        TotalHeadMoment=TotalHeadMoment+abs(RQ[i+1]-0);
        initial =0;
        for(i=index;i<n;i++)
        {
             TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
             initial=RQ[i];
                  }
    }
        printf("Total head movement is %d",TotalHeadMoment);
    return 0;
}
```

OUTPUT:

```
Enter the number of Request

8

Enter the Requests Sequence

95 180 34 119 11 123 62 64

Enter initial head position

50

Enter total disk size

200

Enter the head movement direction for high 1 and for low 0

1

Total head movement is 337
```

CONCLUSION:

Thus the program to implement file allocation strategies using c program was executed and the output was verified.

Ex.No:15c	IMPLEMENTATION OF DISK SCHEDULING ALGORITHM (C-Scan disk scheduling)
Date:	

AIM:

To acquire operating system skills through Disk Scheduling algorithms.

ALGORITHM:

Step 1: Start the program.

Step 2: Declare the variables.

Step 3: Intially read the track head

Step4: Read the tail head

Step 5: Head moment should be in the lower value

Step 6: Display the total number of head position

Step 7: Stop the programme

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int RQ[100],i,j,n,TotalHeadMoment=0,initial,size,move;
    printf("Enter the number of Requests\n");
    scanf("%d",&n);
    printf("Enter the Requests sequence\n");
    for(i=0;i<n;i++)
     scanf("%d",&RQ[i]);
    printf("Enter initial head position\n");
    scanf("%d",&initial);
    printf("Enter total disk size\n");
    scanf("%d",&size);
    printf("Enter the head movement direction for high 1 and for low 0\n");
    scanf("%d",&move);

    // logic for C-Scan disk scheduling


        /*logic for sort the request array */
    for(i=0;i<n;i++)
    {
        for( j=0;j<n-i-1;j++)
        {
            if(RQ[j]>RQ[j+1])
            {
                int temp;
                temp=RQ[j];
                RQ[j]=RQ[j+1];
                RQ[j+1]=temp;
            }
        }
    }
    int index;
    for(i=0;i<n;i++)
    {
        if(initial<RQ[i])
        {
            index=i;
            break;
        }
    }
    // if movement is towards high value
    if(move==1)
    {
        for(i=index;i<n;i++)
        {
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
            initial=RQ[i];
        }
        // last movement for max size
        TotalHeadMoment=TotalHeadMoment+abs(size-RQ[i-1]-1);
        /*movement max to min disk */
        TotalHeadMoment=TotalHeadMoment+abs(size-1-0);
        initial=0;
```

```
        for( i=0;i<index;i++)
        {
             TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
             initial=RQ[i];
                    }
    }
    // if movement is towards low value
    else
    {
        for(i=index-1;i>=0;i--)
        {
           TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
           initial=RQ[i];
        }
        // last movement for min size
        TotalHeadMoment=TotalHeadMoment+abs(RQ[i+1]-0);
        /*movement min to max disk */
        TotalHeadMoment=TotalHeadMoment+abs(size-1-0);
        initial =size-1;
        for(i=n-1;i>=index;i--)
        {
             TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
             initial=RQ[i];
                    }
    }
        printf("Total head movement is %d",TotalHeadMoment);
    return 0;
}
```

OUTPUT:

```
Enter the number of Request
8
Enter the Requests Sequence
95 180 34 119 11 123 62 64
Enter initial head position
50
Enter total disk size
200
Enter the head movement direction for high 1 and for low 0
1
Total head movement is 382
```

CONCLUSION:

Thus the program to implement file allocation strategies using c program was executed and the output was verified.

Ex.No:15c	**IMPLEMENTATION OF DISK SCHEDULING ALGORITHM**
Date:	**(LOOK disk scheduling)**

AIM:

To acquire operating system skills through Disk Scheduling algorithms.

ALGORITHM:

Step 1: Start the program.

Step 2: Declare the variables.

Step 3: Intially read the track head

Step4: Read the tail head

Step 5: Head moment should be in the lower value

Step 6: Display the total number of head position

Step 7: Stop the programme

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int RQ[100],i,j,n,TotalHeadMoment=0,initial,size,move;
    printf("Enter the number of Requests\n");
    scanf("%d",&n);
    printf("Enter the Requests sequence\n");
    for(i=0;i<n;i++)
     scanf("%d",&RQ[i]);
    printf("Enter initial head position\n");
    scanf("%d",&initial);
    printf("Enter total disk size\n");
    scanf("%d",&size);
    printf("Enter the head movement direction for high 1 and for low 0\n");
    scanf("%d",&move);
        // logic for look disk scheduling
            /*logic for sort the request array */
    for(i=0;i<n;i++)
    {
        for(j=0;j<n-i-1;j++)
        {
            if(RQ[j]>RQ[j+1])
            {
                int temp;
                temp=RQ[j];
                RQ[j]=RQ[j+1];
                RQ[j+1]=temp;
            }
        }
    }
    int index;
    for(i=0;i<n;i++)
    {
        if(initial<RQ[i])
        {
            index=i;
            break;
        }
    }
      // if movement is towards high value
    if(move==1)
    {
        for(i=index;i<n;i++)
        {
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
            initial=RQ[i];
        }

        for(i=index-1;i>=0;i--)
        {
             TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
             initial=RQ[i];
                 }
    }
    // if movement is towards low value
    else
    {
        for(i=index-1;i>=0;i--)
        {
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
            initial=RQ[i];
        }

        for(i=index;i<n;i++)
        {
             TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
             initial=RQ[i];

        }
    }

    printf("Total head movement is %d",TotalHeadMoment);
    return 0;
}
```

OUTPUT:

```
Enter the number of Request

8

Enter the Requests Sequence

95 180 34 119 11 123 62 64

Enter initial head position

50

Enter the head movement direction for high 1 and for low 0

1

Total head movement is 299
```

CONCLUSION:

Thus the program is Executed and output is verified successfully.

Ex.No:15d	**IMPLEMENTATION OF DISK SCHEDULING ALGORITHM**
Date:	**(C-LOOK disk scheduling)**

AIM:

To acquire operating system skills through Disk Scheduling algorithms.

ALGORITHM:

Step 1: Start the program.

Step 2: Declare the variables.

Step 3: Intially read the track head

Step4: Read the tail head

Step 5: Head moment should be in the lower value

Step 6: Display the total number of head position

Step 7: Stop the programme

PROGRAM:

```
#include<stdio.h>

#include<stdlib.h>

int main()

{

    int RQ[100],i,j,n,TotalHeadMoment=0,initial,size,move;

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

    scanf("%d",&n);

    printf("Enter the Requests sequence\n");

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

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

    printf("Enter initial head position\n");

    scanf("%d",&initial);

    printf("Enter total disk size\n");

    scanf("%d",&size);

    printf("Enter the head movement direction for high 1 and for low 0\n");

    scanf("%d",&move);


    // logic for C-look disk scheduling


        /*logic for sort the request array */

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

    {

        for( j=0;j<n-i-1;j++)

        {

            if(RQ[j]>RQ[j+1])

            {

                int temp;
                temp=RQ[j];
                RQ[j]=RQ[j+1];
                RQ[j+1]=temp;
            }

        }
    }

    int index;
    for(i=0;i<n;i++)
    {
        if(initial<RQ[i])
        {
            index=i;
            break;
        }
    }

    // if movement is towards high value
    if(move==1)
    {
        for(i=index;i<n;i++)
        {
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
            initial=RQ[i];
        }

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

```
        {
             TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
             initial=RQ[i];


        }
    }
    // if movement is towards low value
    else
    {
        for(i=index-1;i>=0;i--)
        {
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
            initial=RQ[i];
        }

        for(i=n-1;i>=index;i--)
        {
             TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
             initial=RQ[i];


        }
    }

    printf("Total head movement is %d",TotalHeadMoment);
    return 0;
}
```

OUTPUT:

```
Enter the number of Request
8
Enter the Requests Sequence
95 180 34 119 11 123 62 64
Enter initial head position
50
Enter the head movement direction for high 1 and for low 0
1
Total head movement is 322
```

CONCLUSION:

Thus the programme is executed successfully and output is verified.

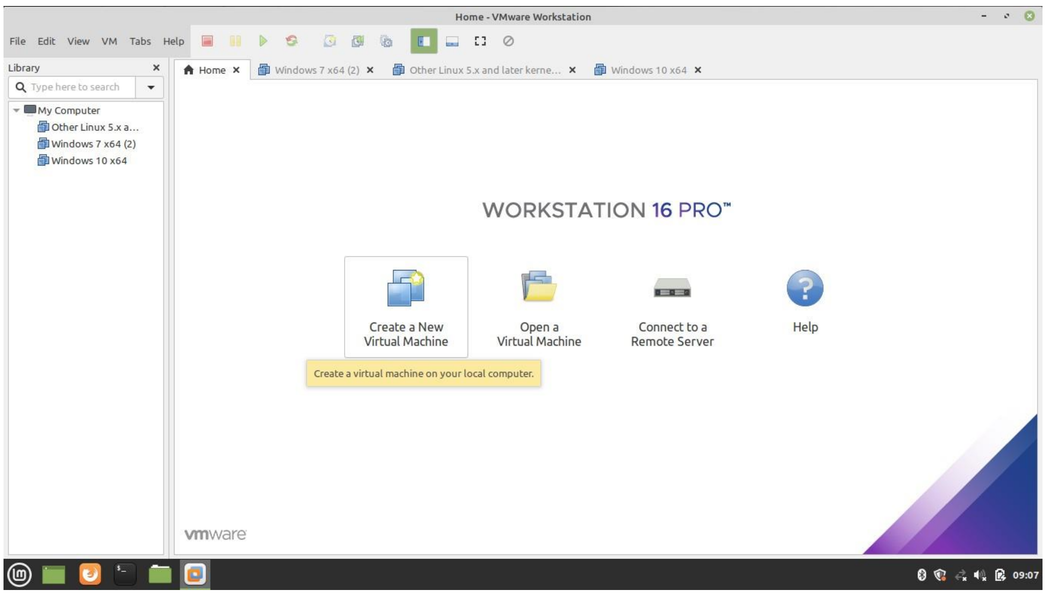
Ex.No:16	
Date:	**INSTALLATION OF UBUNTU OPERATING SYSTEM**

AIM:

To write a step by step process of Ubuntu operating system installation.

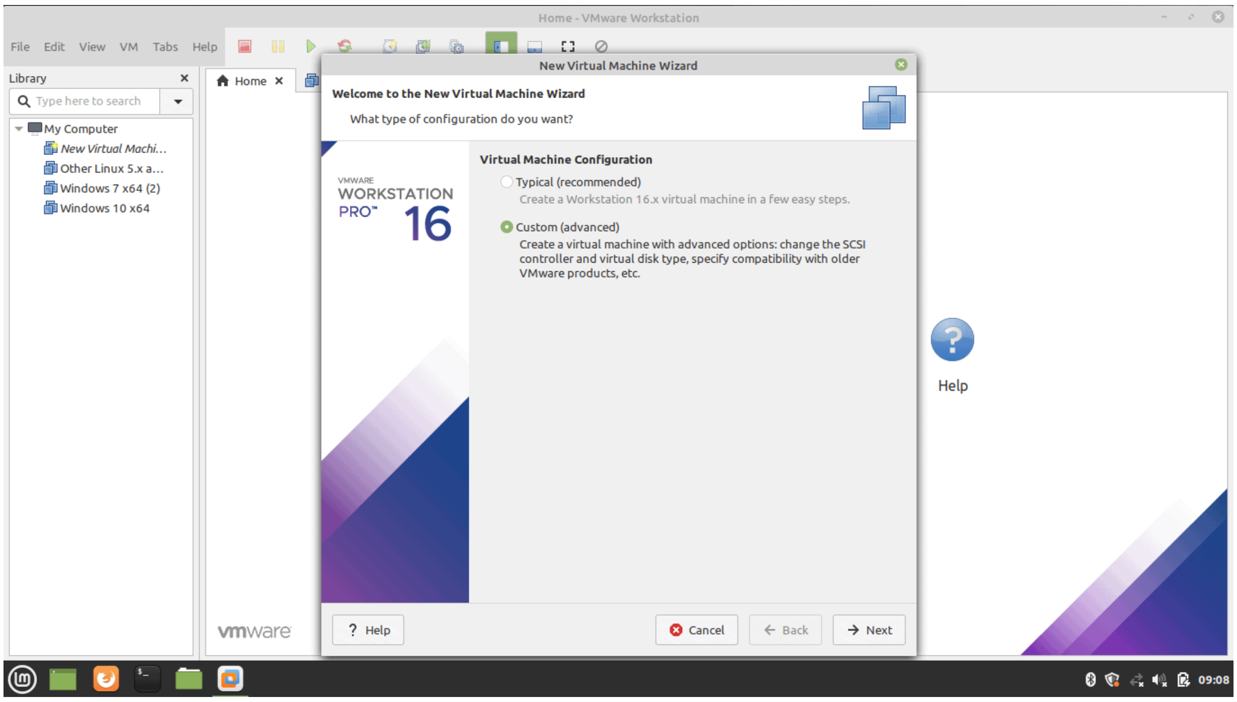
INSTALLATION PROCEDURE:

STEP 1: Download the VMWare Workstation application for your host operating system and install it on your machine.


Home - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Other Linux 5.x a...
Windows 7 x64 (2)
Windows 10 x64
Home
Windows 7 x64 (2)
Other Linux 5.x and later kerne...
Windows 10 x64
WORKSTATION 16 PRO™
Create a New Virtual Machine
Open a Virtual Machine
Connect to a Remote Server
Help
Create a virtual machine on your local computer.
vmware
09:07

STEP 2: Select Custom Configuration Wizard

You can choose either Typical or Custom Wizard. We recommend selecting Custom if you want to install with all the configurations. If you are okay with default configurations then go ahead with Typical configurations.



Welcome to the New Virtual Machine Wizard

What type of configuration do you want?

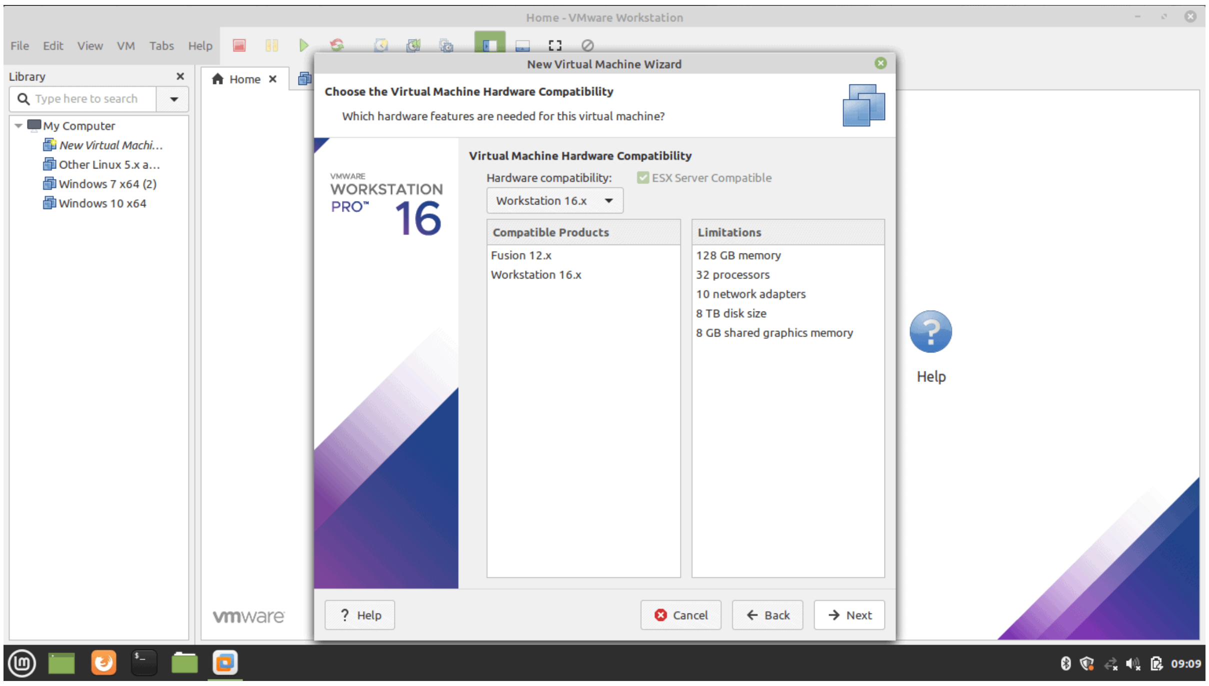
Virtual Machine Configuration

Typical (recommended) [✓ Custom (advanced)]

Typical (recommended): Create a Workstation 16.x virtual machine in a few easy steps.

Custom (advanced): Create a virtual machine with advanced options: change the SCSI controller and virtual disk type, specify compatibility with older VMware products, etc.

STEP 3: Select Virtual Machine Hardware Compatibility



Choose the Virtual Machine Hardware Compatibility

Which hardware features are needed for this virtual machine?

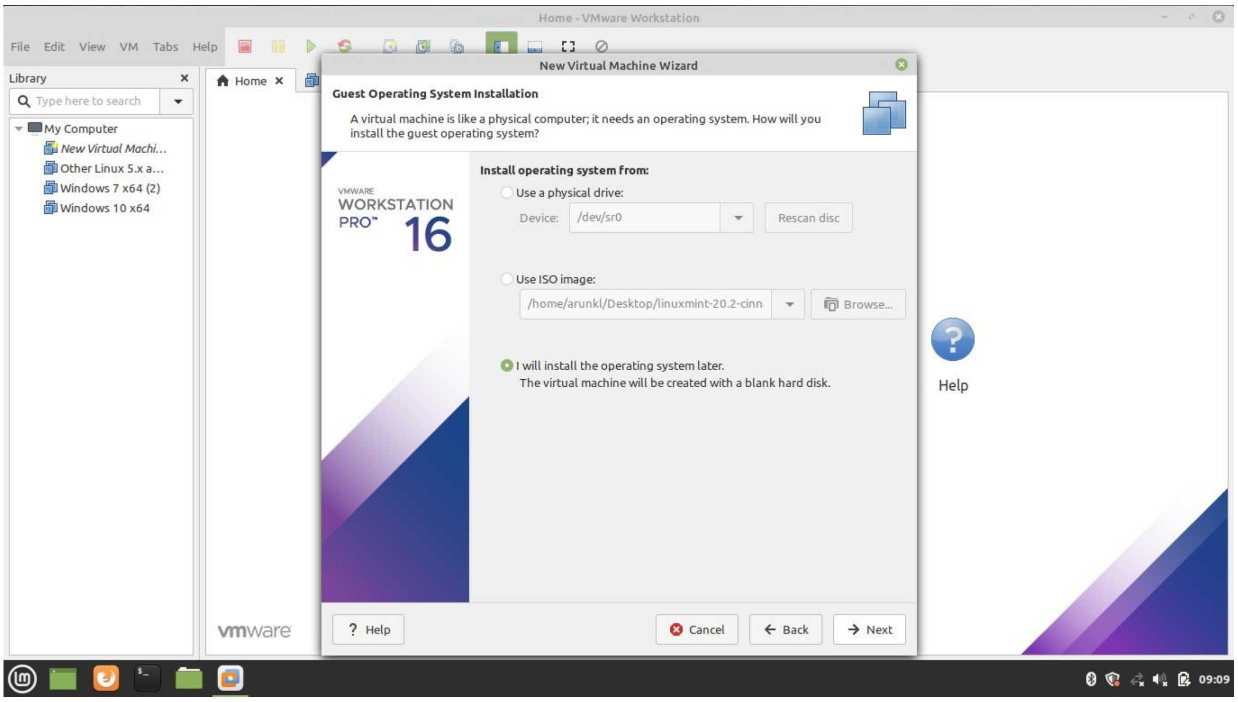
Virtual Machine Hardware Compatibility

Hardware compatibility: Workstation 16.x

ESX Server Compatible

Compatible Products	Limitations
Fusion 12.x	128 GB memory
Workstation 16.x	32 processors
	10 network adapters
	8 TB disk size
	8 GB shared graphics memory

STEP 4:- Select the Operating System Media



New Virtual Machine Wizard

Guest Operating System Installation

A virtual machine is like a physical computer; it needs an operating system. How will you install the guest operating system?

Install operating system from:

Use a physical drive:

Device: /dev/sr0 Rescan disc

Use ISO image:

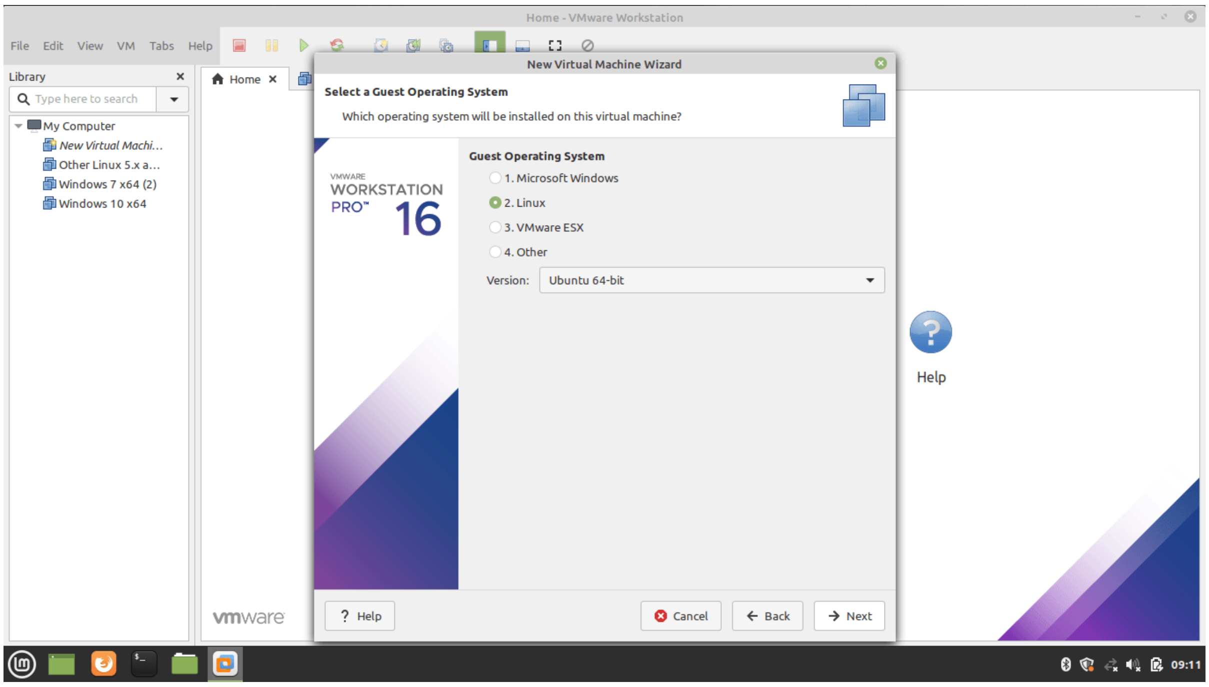
/home/arunkl/Desktop/linuxmint-20.2-cinn Browse...

[✓ I will install the operating system later.]

The virtual machine will be created with a blank hard disk.

Help Cancel Back Next

STEP 5:- Select Guest Operating System



New Virtual Machine Wizard

Select a Guest Operating System

Which operating system will be installed on this virtual machine?

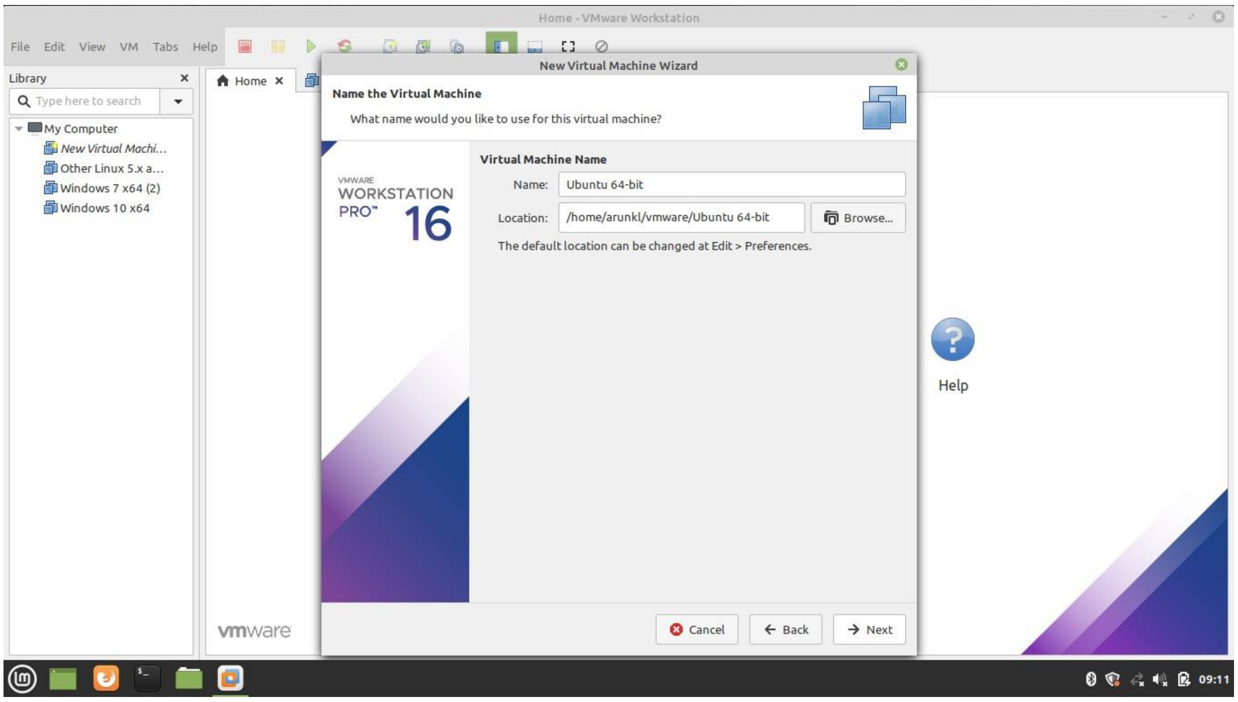
Guest Operating System

1. Microsoft Windows [✓ 2. Linux] 3. VMware ESX 4. Other

Version: Ubuntu 64-bit

Help Cancel Back Next

STEP 6:- Name the Virtual Machine Name and location



New Virtual Machine Wizard

Name the Virtual Machine

What name would you like to use for this virtual machine?

VMWARE WORKSTATION PRO™ 16

Virtual Machine Name

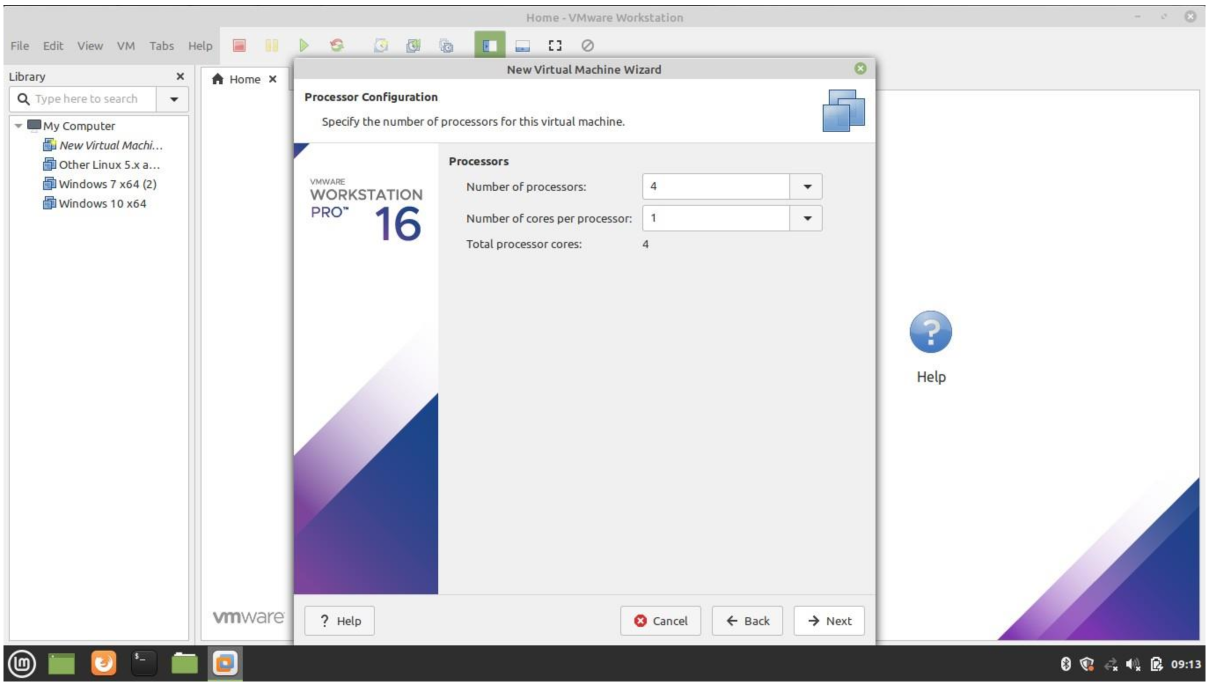
Name: Ubuntu 64-bit

Location: /home/arunkl/vmware/Ubuntu 64-bit   Browse...

The default location can be changed at Edit > Preferences.

Cancel   Back   Next

STEP 7:- Allocate the Processors



New Virtual Machine Wizard

Processor Configuration

Specify the number of processors for this virtual machine.

VMWARE WORKSTATION PRO™ 16

Processors

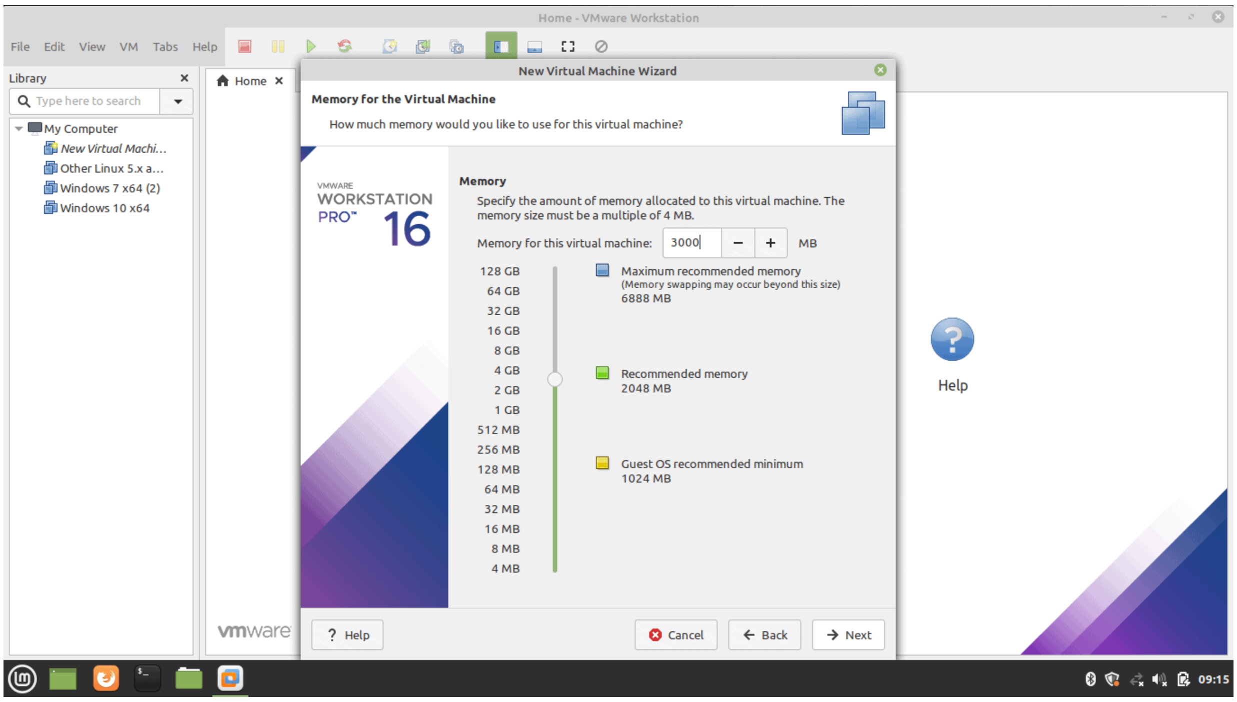
Number of processors: 4

Number of cores per processor: 1

Total processor cores: 4

Help   Cancel   Back   Next

STEP 8:- Allocate the Memory for Virtual Machine



New Virtual Machine Wizard

Memory for the Virtual Machine

How much memory would you like to use for this virtual machine?

Memory

Specify the amount of memory allocated to this virtual machine. The memory size must be a multiple of 4 MB.

Memory for this virtual machine: 3000 MB

128 GB, 64 GB, 32 GB, 16 GB, 8 GB, 4 GB, 2 GB, 1 GB, 512 MB, 256 MB, 128 MB, 64 MB, 32 MB, 16 MB, 8 MB, 4 MB

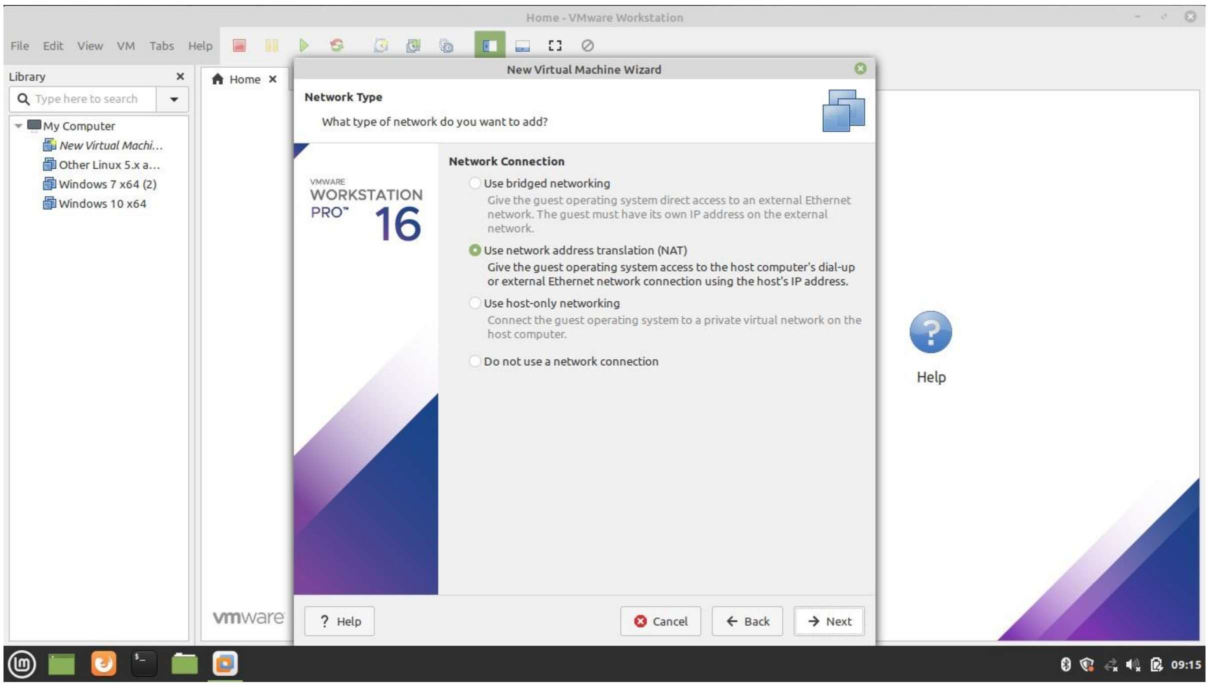
Maximum recommended memory
(Memory swapping may occur beyond this size)
6888 MB

Recommended memory
2048 MB

Guest OS recommended minimum
1024 MB

Help | Cancel | Back | Next

STEP 9:- Choose the Network Configuration



New Virtual Machine Wizard

Network Type

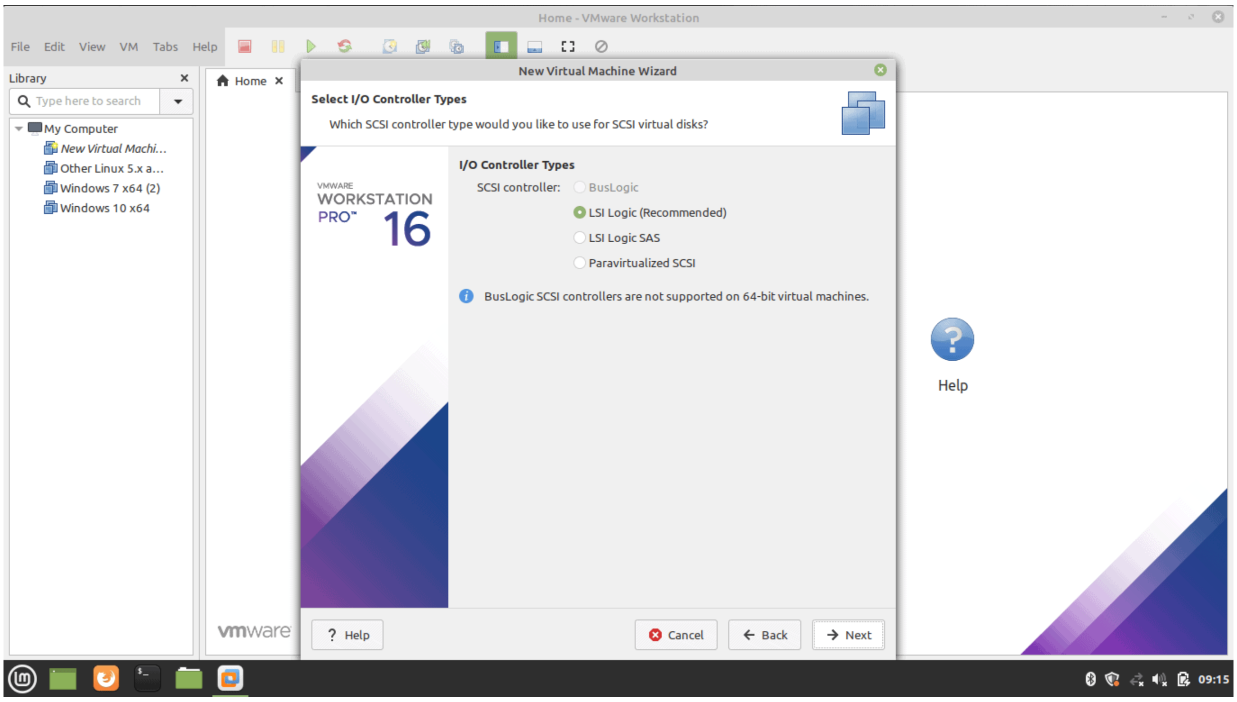
What type of network do you want to add?

Network Connection

- Use bridged networking
  Give the guest operating system direct access to an external Ethernet network. The guest must have its own IP address on the external network.
- [✓ Use network address translation (NAT)]
  Give the guest operating system access to the host computer's dial-up or external Ethernet network connection using the host's IP address.
- Use host-only networking
  Connect the guest operating system to a private virtual network on the host computer.
- Do not use a network connection

Help | Cancel | Back | Next

STEP 10:- Select the io controller type



New Virtual Machine Wizard

Select I/O Controller Types

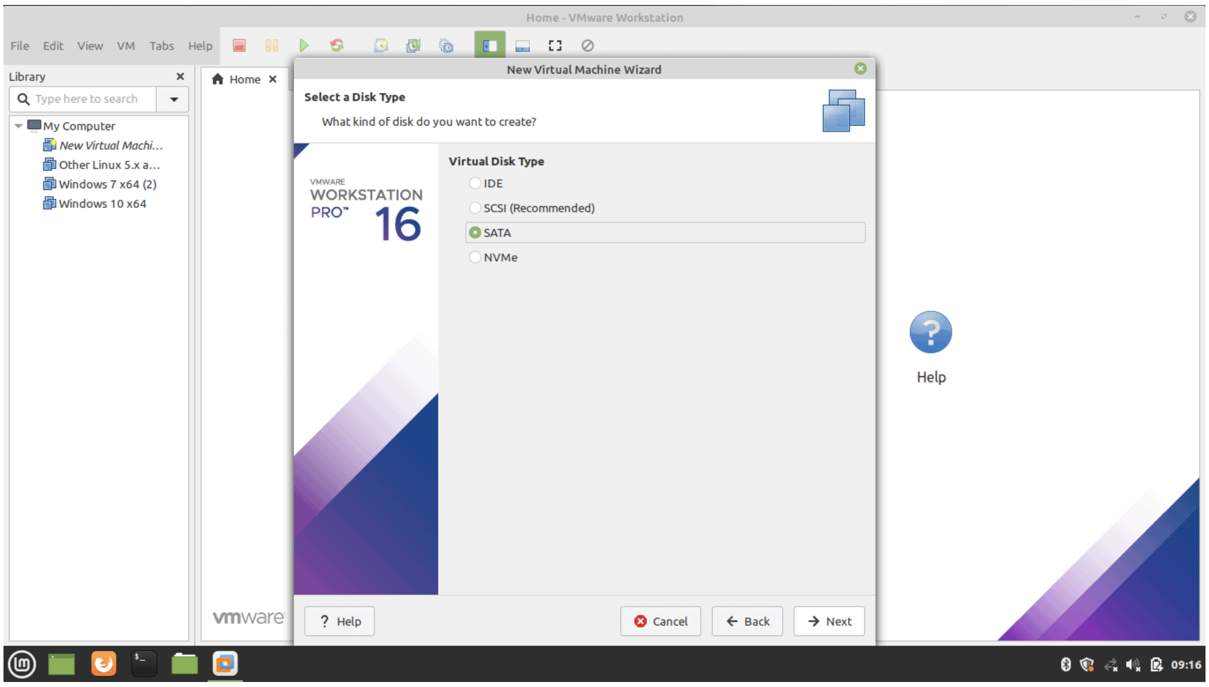
Which SCSI controller type would you like to use for SCSI virtual disks?

I/O Controller Types

SCSI controller: BusLogic [✓ LSI Logic (Recommended)] LSI Logic SAS Paravirtualized SCSI

BusLogic SCSI controllers are not supported on 64-bit virtual machines.

STEP 11:- Select Disk Type



New Virtual Machine Wizard

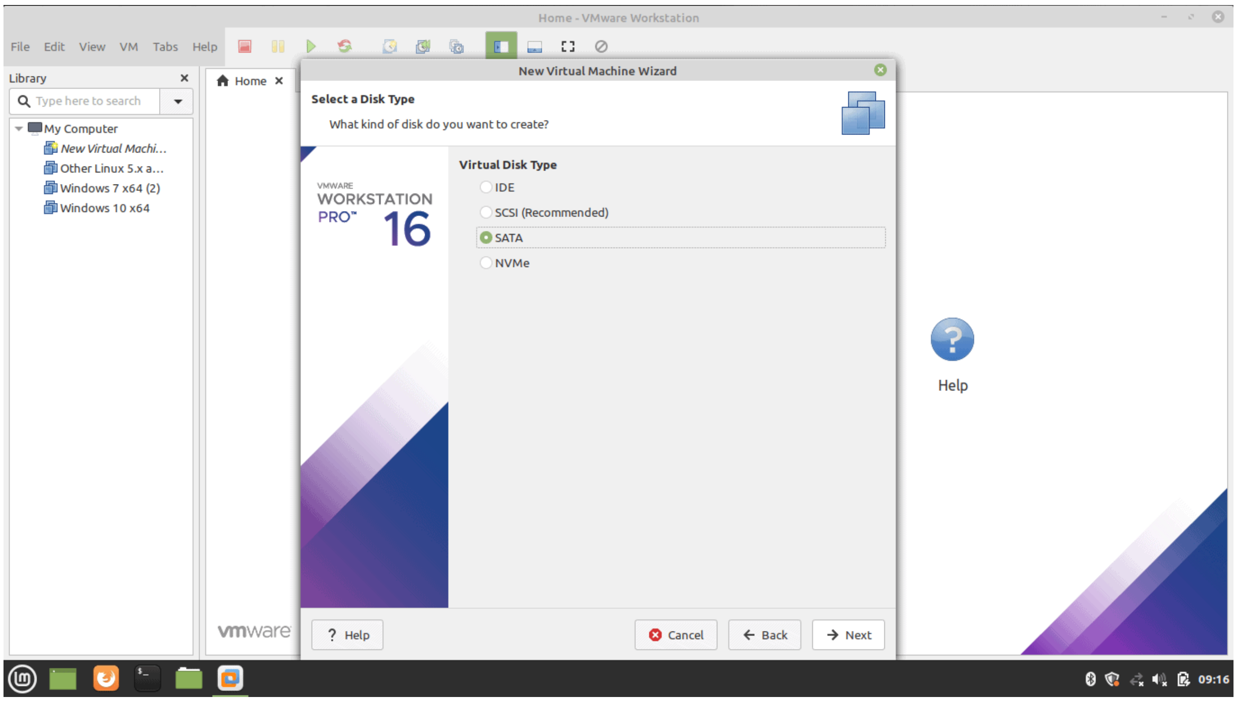
Select a Disk Type

What kind of disk do you want to create?

Virtual Disk Type

IDE SCSI (Recommended) [✓ SATA] NVMe

STEP 12:- Select Virtual Disk



New Virtual Machine Wizard

Select a Disk Type

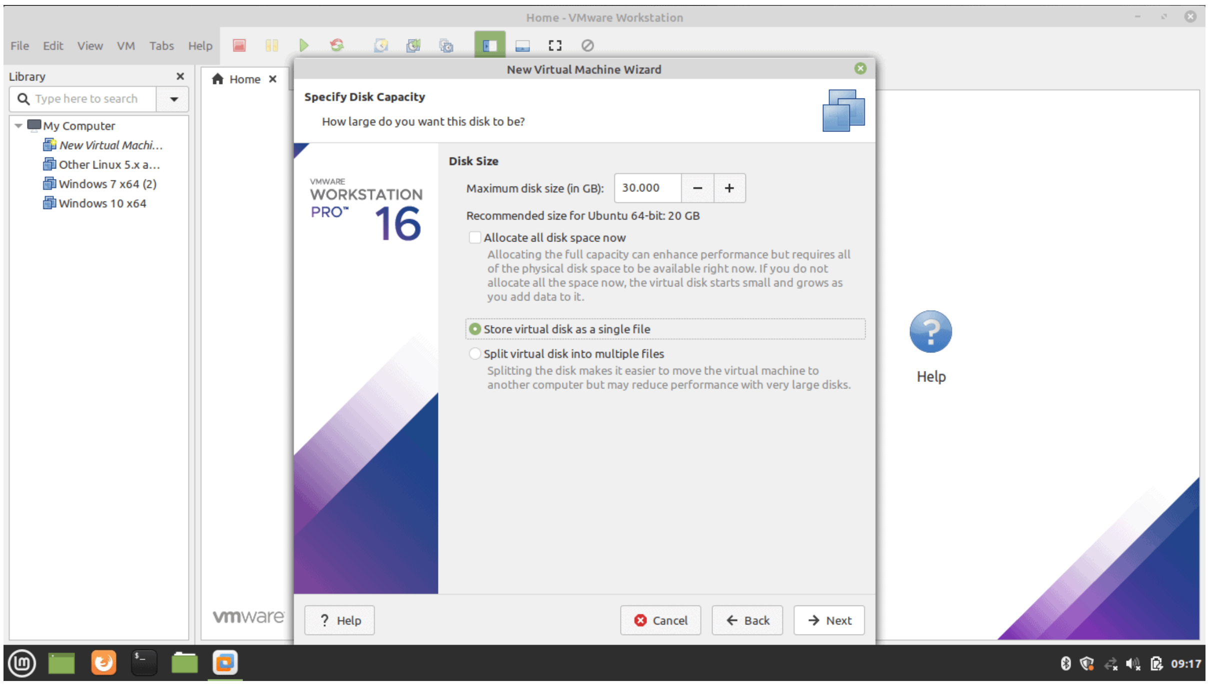
What kind of disk do you want to create?

Virtual Disk Type

- IDE
- SCSI (Recommended)
- [✓ SATA]
- NVMe

Help | Cancel | Back | Next

STEP 13:- Select Disk Capacity



New Virtual Machine Wizard

Specify Disk Capacity

How large do you want this disk to be?

Disk Size

Maximum disk size (in GB): 30.000

Recommended size for Ubuntu 64-bit: 20 GB

- Allocate all disk space now

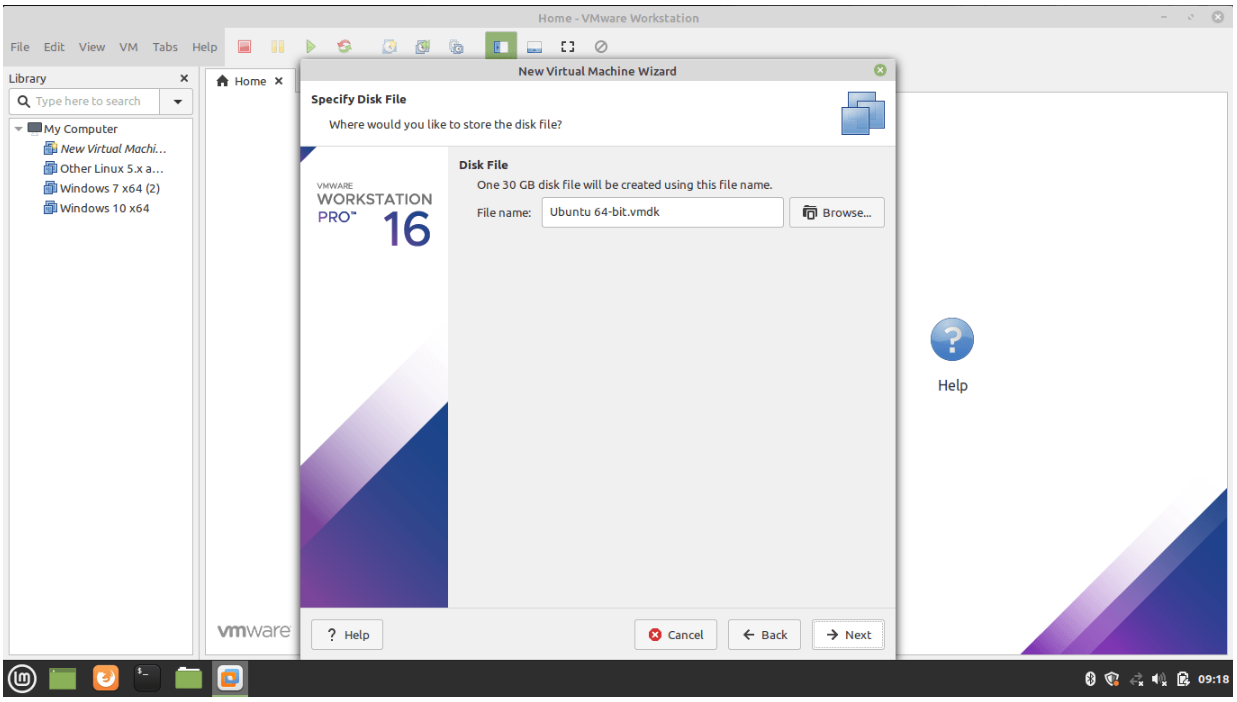
  Allocating the full capacity can enhance performance but requires all of the physical disk space to be available right now. If you do not allocate all the space now, the virtual disk starts small and grows as you add data to it.

- [✓ Store virtual disk as a single file]
- Split virtual disk into multiple files

  Splitting the disk makes it easier to move the virtual machine to another computer but may reduce performance with very large disks.

Help | Cancel | Back | Next

STEP 14:- Specify Virtual Disk File



Specify Disk File

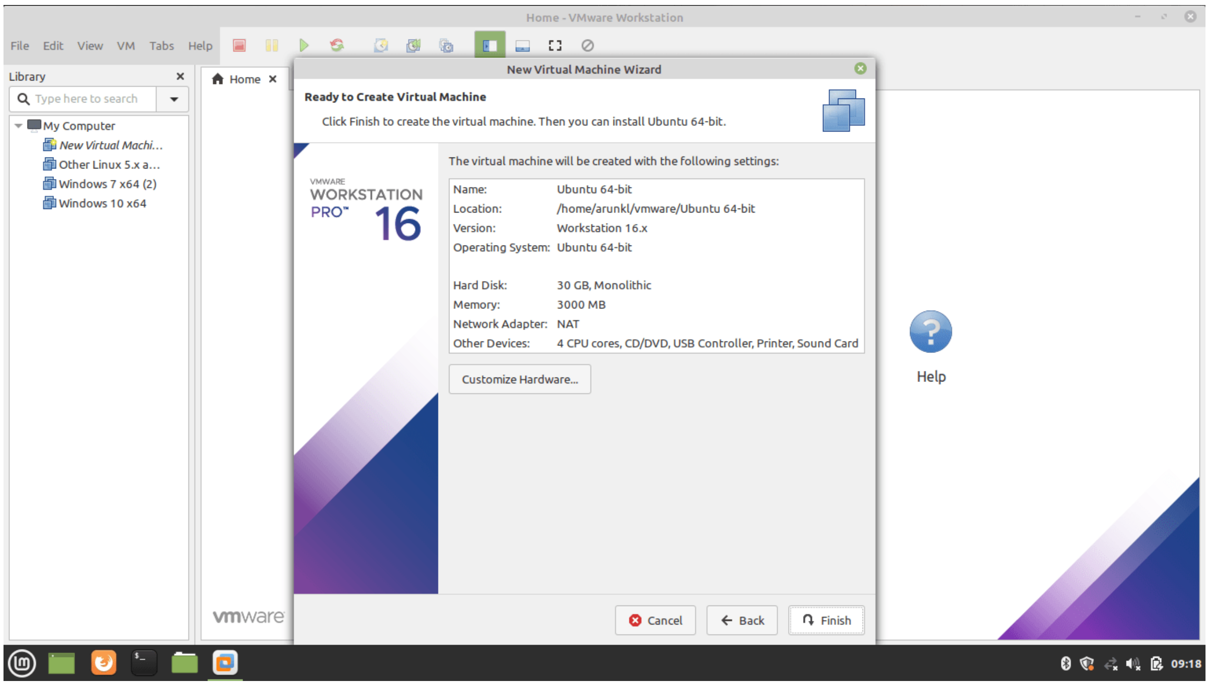
Where would you like to store the disk file?

Disk File

One 30 GB disk file will be created using this file name.

File name: Ubuntu 64-bit.vmdk

STEP 15:- Create Virtual Machine



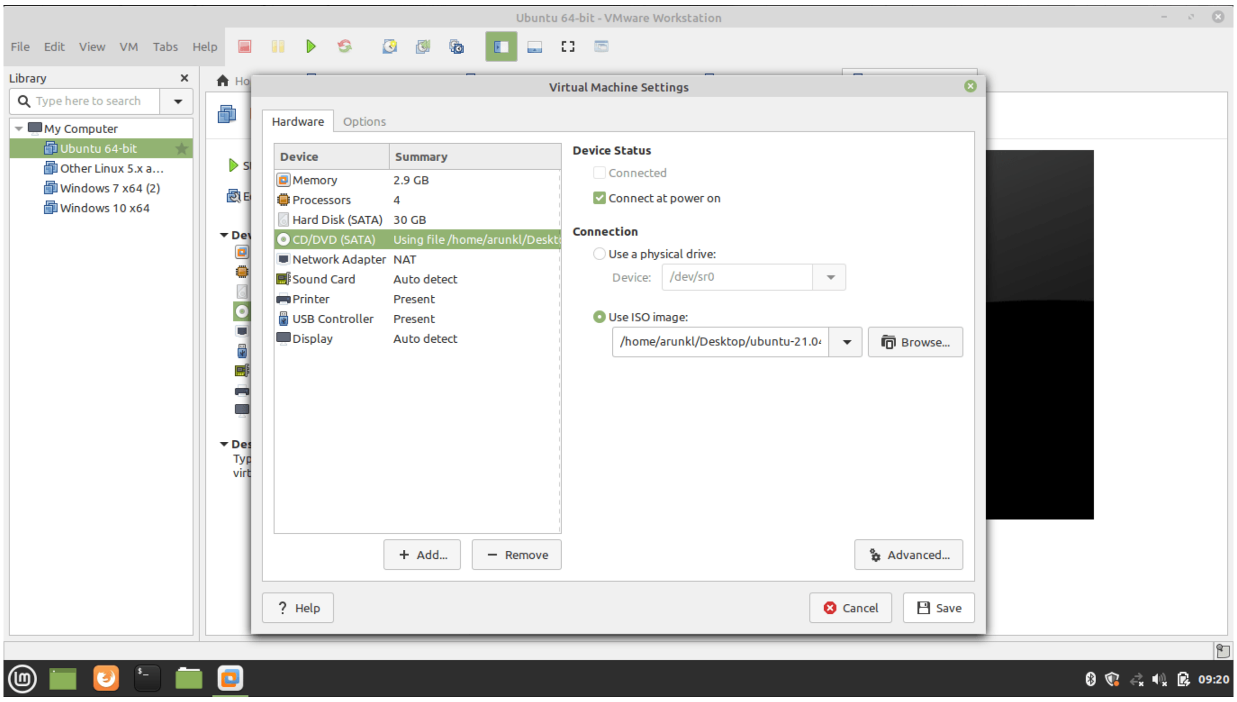
Ready to Create Virtual Machine

Click Finish to create the virtual machine. Then you can install Ubuntu 64-bit.

The virtual machine will be created with the following settings:

Name:	Ubuntu 64-bit
Location:	/home/arunkl/vmware/Ubuntu 64-bit
Version:	Workstation 16.x
Operating System:	Ubuntu 64-bit
Hard Disk:	30 GB, Monolithic
Memory:	3000 MB
Network Adapter:	NAT
Other Devices:	4 CPU cores, CD/DVD, USB Controller, Printer, Sound Card

STEP 16:- Supply Ubuntu ISO Image to Virtual Machine



Device	Summary
Memory	2.9 GB
Processors	4
Hard Disk (SATA)	30 GB
CD/DVD (SATA)	Using file /home/arunkl/Deskt
Network Adapter	NAT
Sound Card	Auto detect
Printer	Present
USB Controller	Present
Display	Auto detect

Device Status

Connected

[✓] Connect at power on

Connection

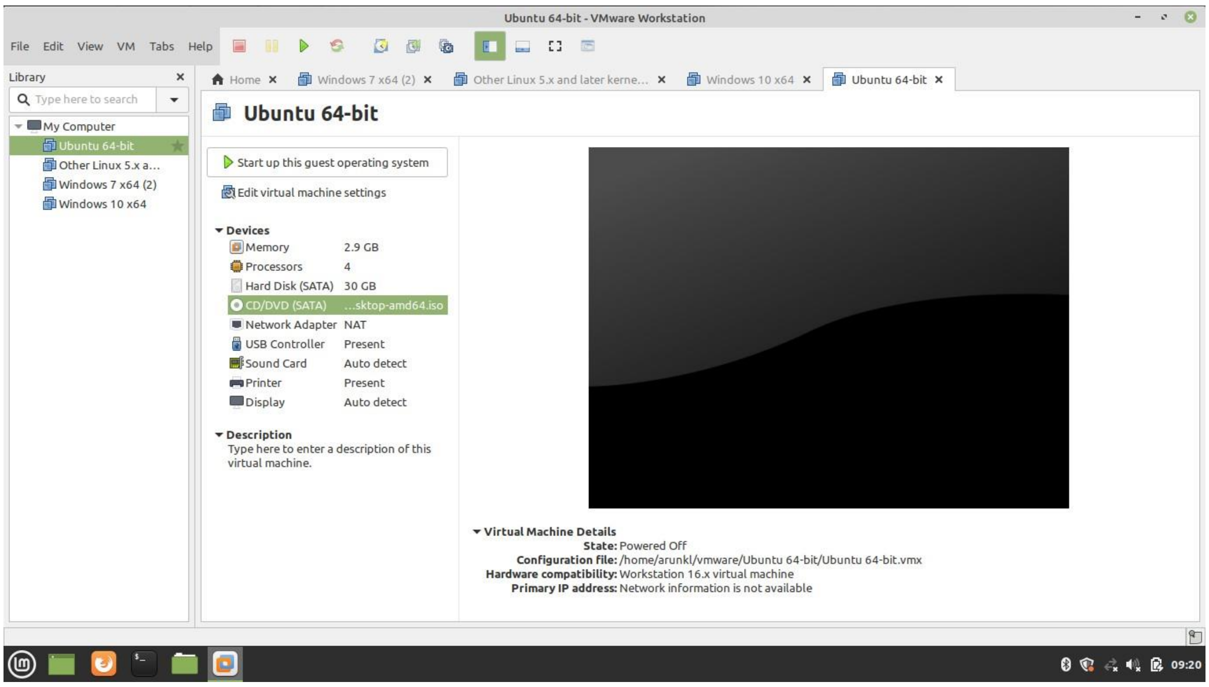
Use a physical drive:

Device: /dev/sr0

[✓] Use ISO image:

/home/arunkl/Desktop/ubuntu-21.0

STEP 17: Install Ubuntu Linux on VMWare Workstation



Devices

Memory	2.9 GB
Processors	4
Hard Disk (SATA)	30 GB
CD/DVD (SATA)	...sktop-amd64.iso
Network Adapter	NAT
USB Controller	Present
Sound Card	Auto detect
Printer	Present
Display	Auto detect

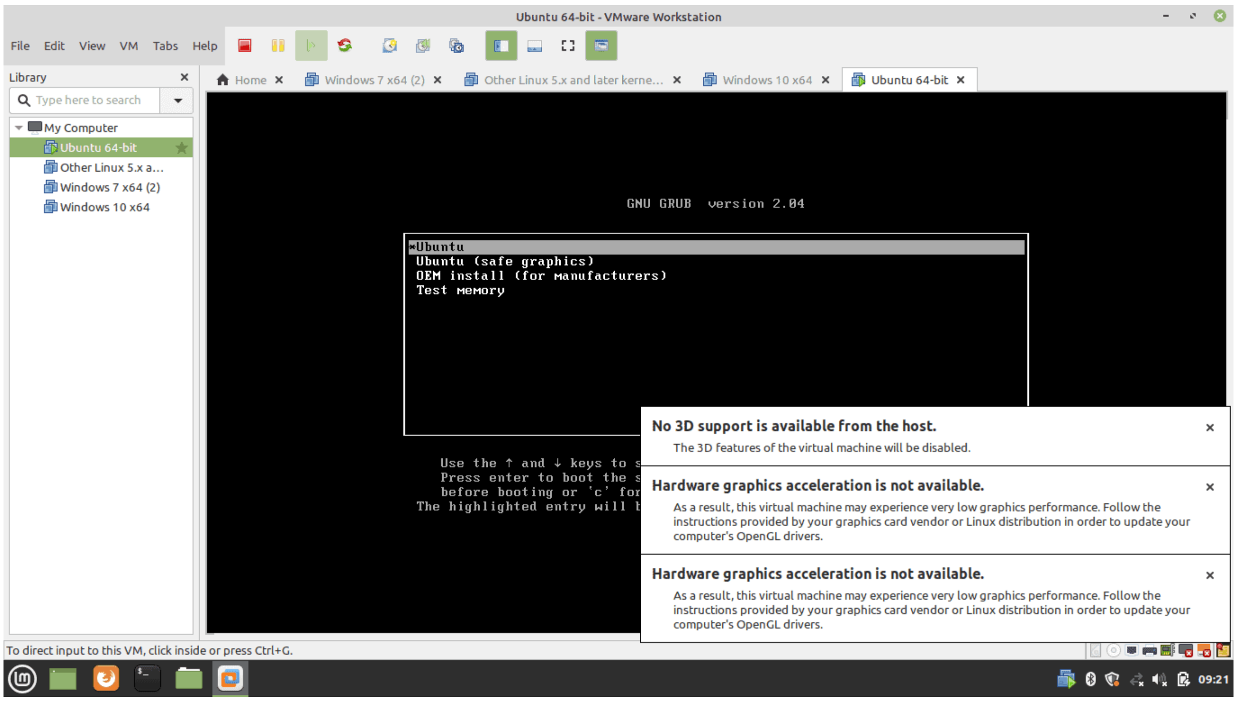
Description

Type here to enter a description of this virtual machine.

Virtual Machine Details

State: Powered Off
Configuration file: /home/arunkl/vmware/Ubuntu 64-bit/Ubuntu 64-bit.vmx
Hardware compatibility: Workstation 16.x virtual machine
Primary IP address: Network information is not available

STEP 18: Power On the Virtual Machine



```
GNU GRUB  version 2.04

*Ubuntu
 Ubuntu (safe graphics)
 OEM install (for manufacturers)
 Test memory

   Use the ↑ and ↓ keys to s
   Press enter to boot the s
   before booting or 'c' for
 The highlighted entry will b
```

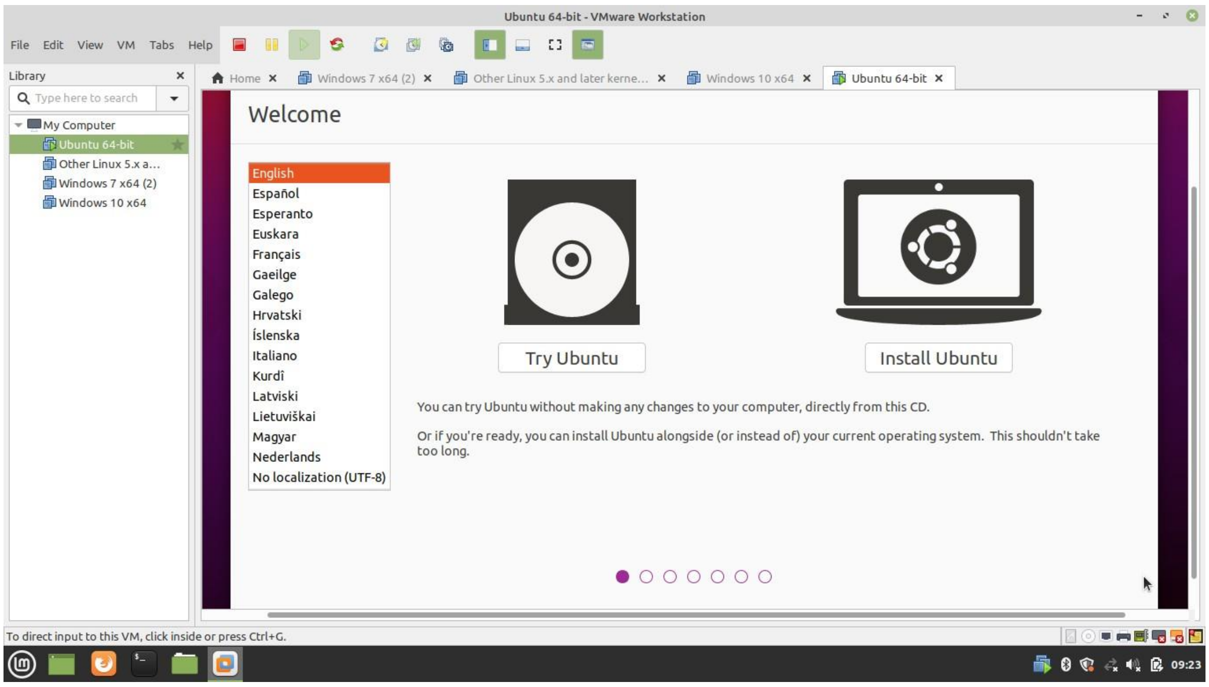
No 3D support is available from the host.
The 3D features of the virtual machine will be disabled.

Hardware graphics acceleration is not available.
As a result, this virtual machine may experience very low graphics performance. Follow the instructions provided by your graphics card vendor or Linux distribution in order to update your computer's OpenGL drivers.

Hardware graphics acceleration is not available.
As a result, this virtual machine may experience very low graphics performance. Follow the instructions provided by your graphics card vendor or Linux distribution in order to update your computer's OpenGL drivers.

To direct input to this VM, click inside or press Ctrl+G.

STEP 19:- After powering on the Virtual machine, you will be treated with a welcome screen on which you will see two options: Try Ubuntu and Install Ubuntu. Select Try Ubuntu if you want to run Ubuntu in live mode. Select Install to continue the installation process.

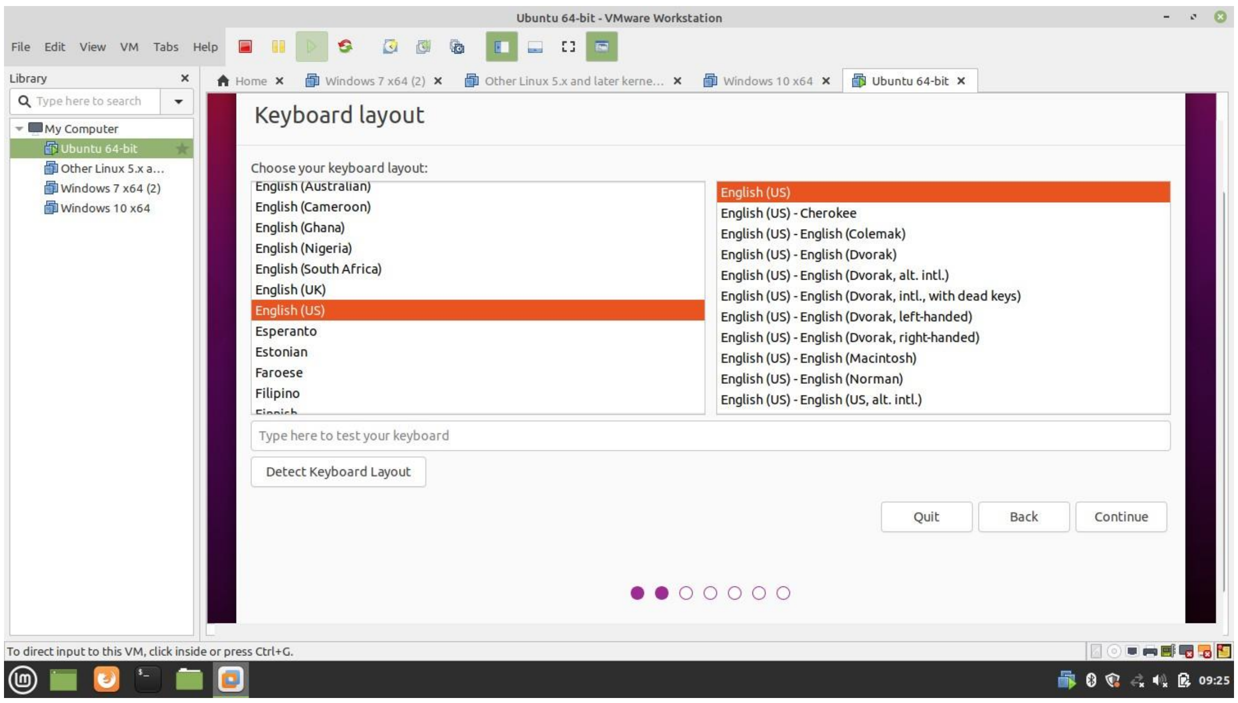


You can try Ubuntu without making any changes to your computer, directly from this CD.

Or if you're ready, you can install Ubuntu alongside (or instead of) your current operating system. This shouldn't take too long.

To direct input to this VM, click inside or press Ctrl+G.

STEP 20:-Select a keyboard Layout



Keyboard layout

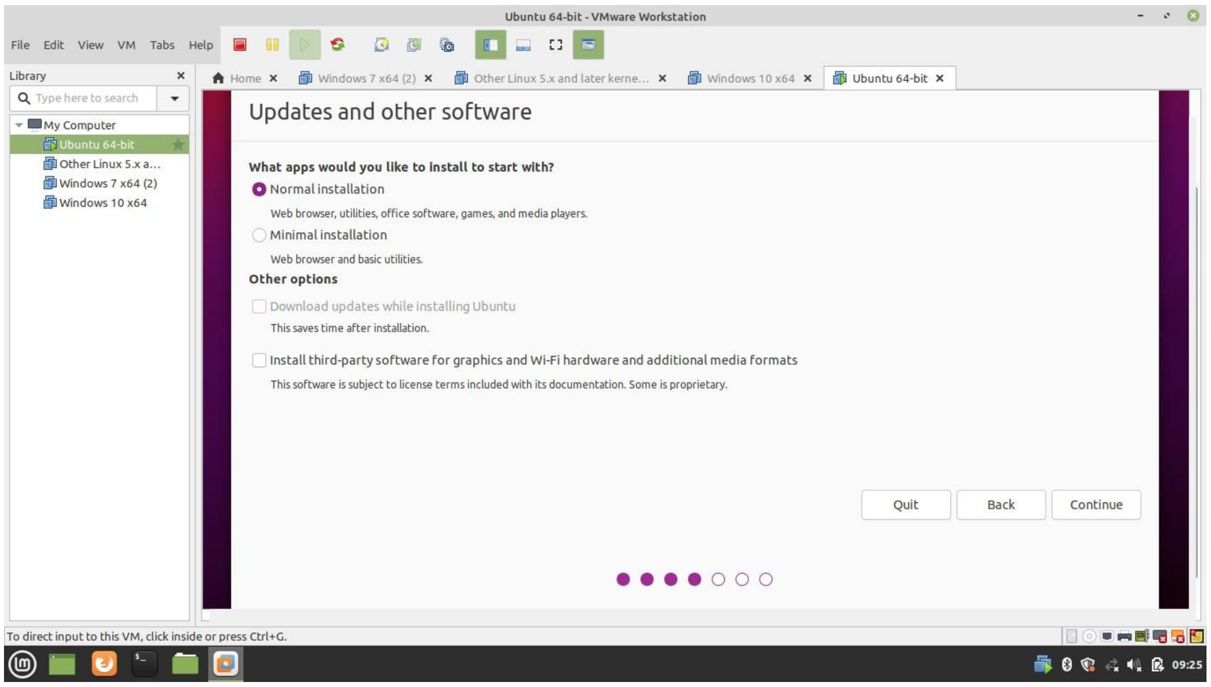
Choose your keyboard layout:

English (Australian)
English (Cameroon)
English (Ghana)
English (Nigeria)
English (South Africa)
English (UK)
English (US)
Esperanto
Estonian
Faroese
Filipino
Finnish

English (US)
English (US) - Cherokee
English (US) - English (Colemak)
English (US) - English (Dvorak)
English (US) - English (Dvorak, alt. intl.)
English (US) - English (Dvorak, intl., with dead keys)
English (US) - English (Dvorak, left-handed)
English (US) - English (Dvorak, right-handed)
English (US) - English (Macintosh)
English (US) - English (Norman)
English (US) - English (US, alt. intl.)

Type here to test your keyboard

STEP 21:- Software Update and Package Selection in Virtual Machine



Updates and other software

What apps would you like to install to start with?

[✓ Normal installation]

Web browser, utilities, office software, games, and media players.

Minimal installation

Web browser and basic utilities.

Other options

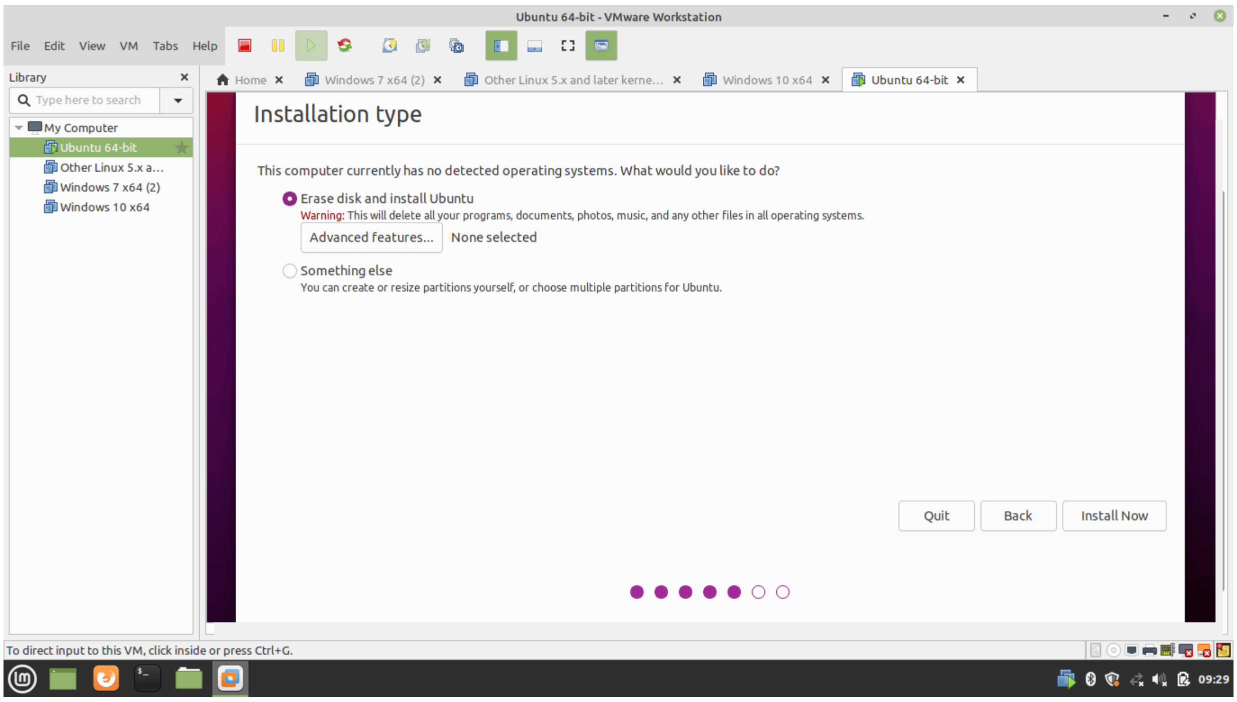
Download updates while installing Ubuntu

This saves time after installation.

Install third-party software for graphics and Wi-Fi hardware and additional media formats

This software is subject to license terms included with its documentation. Some is proprietary.

STEP 22:- Partition the Disk

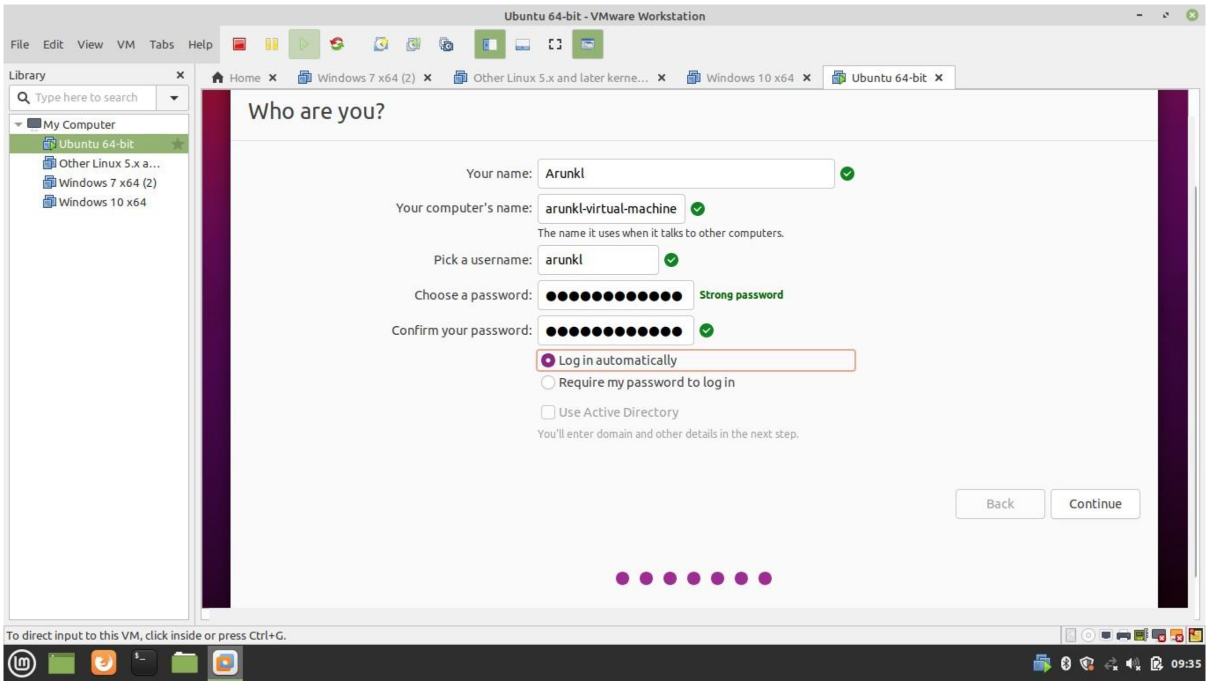

Installation type

This computer currently has no detected operating systems. What would you like to do?

[✓ Erase disk and install Ubuntu]
Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.
Advanced features... None selected

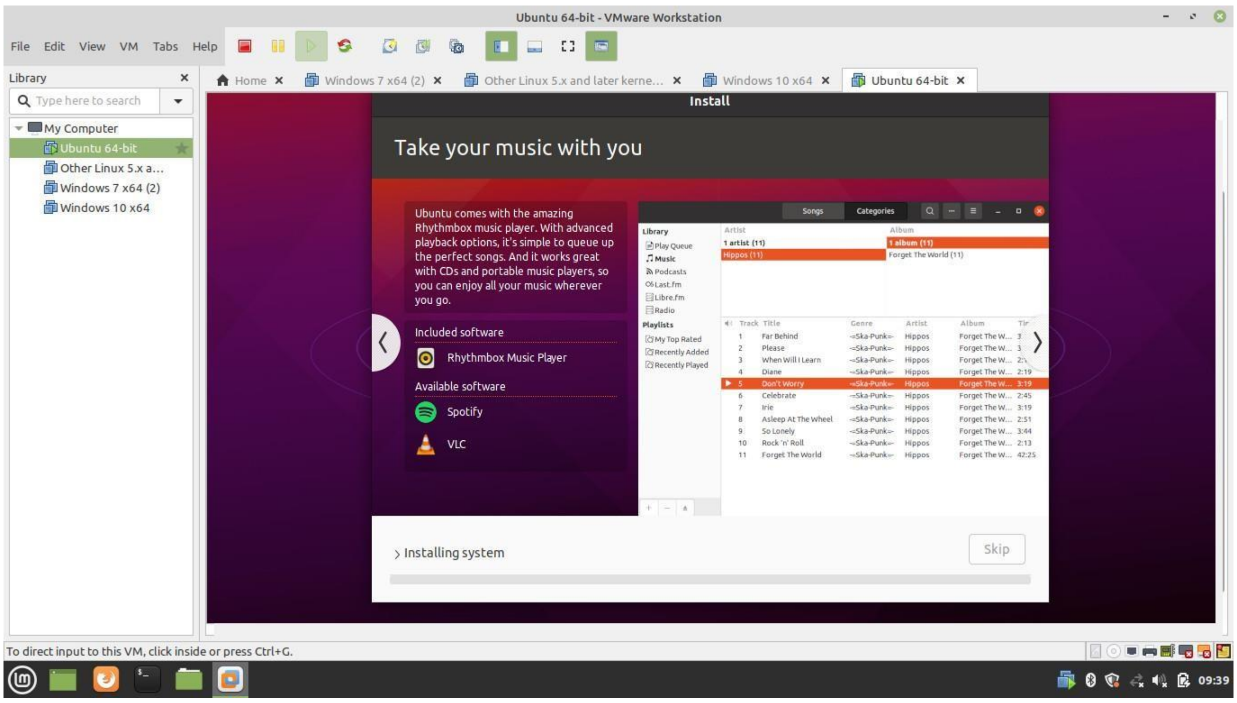
Something else
You can create or resize partitions yourself, or choose multiple partitions for Ubuntu.

STEP 23:- Create an Admin Account

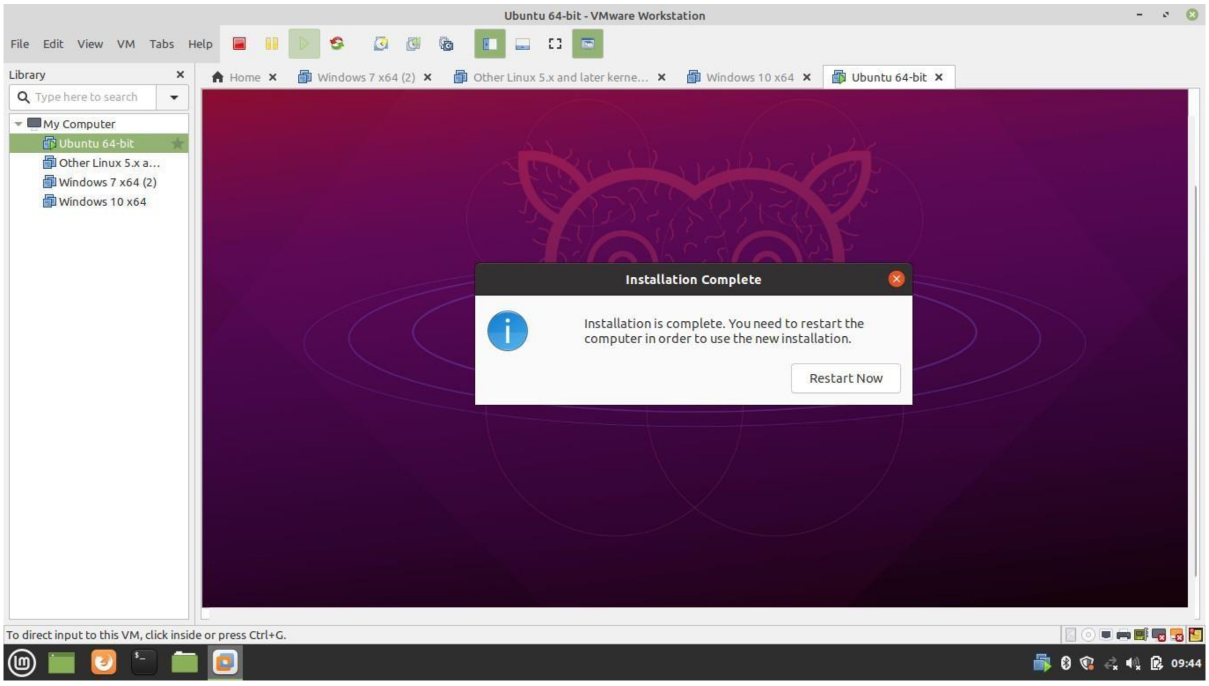

Who are you?

Your name: Arunkl
Your computer's name: arunkl-virtual-machine
The name it uses when it talks to other computers.
Pick a username: arunkl
Choose a password: ●●●●●●●●●●●● Strong password
Confirm your password: ●●●●●●●●●●●●
[✓ Log in automatically]
Require my password to log in
Use Active Directory
You'll enter domain and other details in the next step.

STEP 24: Installation of Ubuntu in Progress

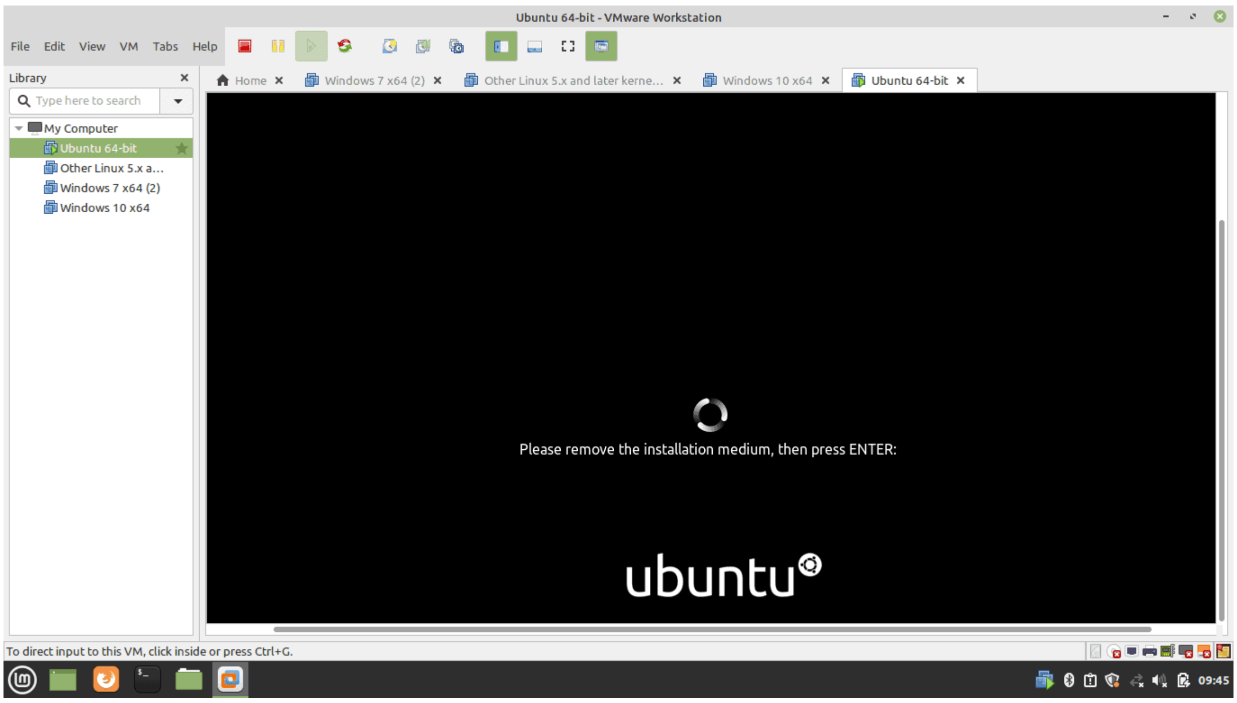

Ubuntu 64-bit - VMware Workstation
Install
Take your music with you
Ubuntu comes with the amazing Rhythmbox music player. With advanced playback options, it's simple to queue up the perfect songs. And it works great with CDs and portable music players, so you can enjoy all your music wherever you go.
Included software
Rhythmbox Music Player
Available software
Spotify
VLC
Installing system
Skip

STEP 25:Reebot virtual machine



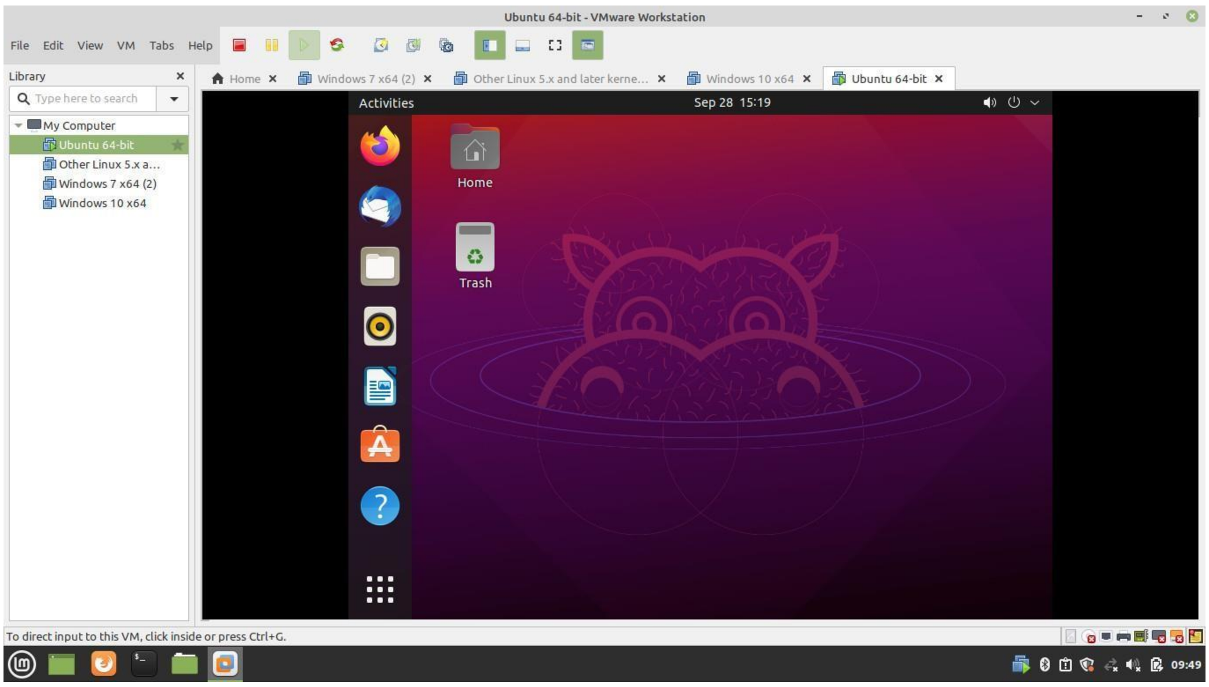
Installation is complete. You need to restart the computer in order to use the new installation.

STEP 26: Remove the installation media



Please remove the installation medium, then press ENTER:

STEP 27: Boot Ubuntu



CONCLUSION:

Thus the Ubuntu operation system is installed over the virtual machine and its correctness of working is verified.

VIVA QUESTIONS

1. Which command is used to display the top of the file?
2. Which command is used to remove a directory?
3. Which of the following commands is used to display the directory attributes rather than itscontents?
4. Which command is used to concatenate all files beginning with the string 'emp' and followed by a non-numeric character?
5. Which command is used to delete all files in the current directory and all its sub-directories?
6. What is UNIX?
7. How is UNIX different from Linux?
8. What is a kernel?
9. What is the difference between multi-user and multi-tasking?
10. What command can you use to display the first 3 lines of text from a file and how does it work?
11. Write command to list all the links from a directory?
12. How will you find which operating system your system is running on in UNIX?
13. How do you find which process is taking how much CPU?
14. How do you check how much space left in current drive?
15. What is Zombie process in UNIX? How do you find Zombie process in UNIX?
16. How do you find which processes are using a particular file?
17. What is ephemeral port in UNIX?
18. How do you find for how many days your Server is up?
19. What is scheduling?
20. Define scheduler.
21. What are the types of scheduler?
22. What are the different types of scheduling algorithms?
23. What is burst time?
24. Define average waiting time?
25. Define average turnaround time?

26. What time quantum/time slice?
27. Define SRTF.
28. What is a semaphore?
29. What are the two kinds of semaphores?
30. What is a mutex?
31. Semaphores are mostly implemented in?
32. What are the two atomic operations permissible on semaphore?
33. What are the different methods used for deadlock recovery?
34. What is Critical Section Problem?
35. What is the disadvantage of invoking the detection algorithm?
36. When is the condition, a wait-for graph deadlock detection algorithm that is used?
37. What is the condition when the wait for graph contains a cycle?
38. What is a thread pool?
39. What are multithreaded models?
40. What is a P-thread?
41. What is Process synchronization?
42. What is a critical section?
43. What is a semaphore?
44. What is a bounded buffer problem?
45. What is multi tasking?
46. What is a macro kernel?
47. What is the main goal of the Memory Management?
48. What is the difference between Swapping and Paging?
49. What is dirty bit?
50. What is a shared pool?
51. What is virtual memory?
52. What is a file?
53. What are file operations?
54. What are the types of file allocation methods?
55. What is indexed file allocation method?

56. What is the difference between sequential & Indexed method?
57. What are file accessing methods?
58. What is the main purpose of sequential file allocation?