ANJUMAN
COLLEGE OF ENGINEERING & TECHNOLOGY
(MANAGED BY : ANJUMAN HAMI-E-ISLAM, NAGPUR)

Operating System Lab Manual

Roll No:____
Name:_____
Sem:_____Section_____

Computer Science
& Engineering
Department

CERTIFICATE

Certified that this file is submitted by

Shri/Ku._____

Roll No._____a student of _____ year of the course _____

_____ as a part of PRACTICAL/ORAL as prescribed by the Rashtrasant Tukadoji Maharaj Nagpur University for the subject_____ in the laboratory of _____during the academic year _____ and that I have instructed him/her for the said work, from time to time and I found him/her to be satisfactory progressive.

And that I have accessed the said work and I am satisfied that the same is up to that standard envisaged for the course.

Date:-

Signature & Name
of Subject Teacher

Signature & Name
of HOD

Anjuman College of Engineering and Technology

Vision

- To be a centre of excellence for developing quality technocrats with moral and social ethics, to face the global challenges for the sustainable development of society.

Mission

- To create conducive academic culture for learning and identifying career goals.
- To provide quality technical education, research opportunities and imbibe entrepreneurship skills contributing to the socio-economic growth of the Nation.
- To inculcate values and skills, that will empower our students towards development through technology.

Vision and Mission of the Department

Vision:

- To achieve excellent standards of quality education in the field of computer science and engineering, aiming towards development of ethically strong technical experts contributing to the profession in the global society.

Mission:

- To create outcome based education environment for learning and identifying career goals.
- Provide latest tools in a learning ambience to enhance innovations, problem solving skills, leadership qualities team spirit and ethical responsibilities.
- Inculcating awareness through innovative activities in the emerging areas of technology.

Program Educational Objectives (PEOs)

- The graduates will have a strong foundation in mathematical, scientific and engineering fundamentals necessary to formulate, solve and analyze engineering problem in their career.
- Graduates will be able to create and design computer support systems and impart knowledge and skills to analyze, design, test and implement various software applications.
- Graduates will work productively as computer science engineers towards betterment of society exhibiting ethical qualities.

Program Specific Outcomes (PSOs)

- Foundation of mathematical concepts: To use mathematical methodologies and techniques for computing and solving problem using suitable mathematical analysis, data structures, database and algorithms as per the requirement.
- Foundation of Computer System: The capability and ability to interpret and understand the fundamental concepts and methodology of computer systems and programming. Students can understand the functionality of hardware and software aspects of computer systems, networks and security.
- Foundations of Software development: The ability to grasp the software development lifecycle and methodologies of software system and project development.

PROGRAM: CSE	DEGREE: B.E
COURSE: Operating System	SEMESTER: IV CREDITS: 2
COURSE CODE: BECSE4S3T	COURSE TYPE: REGULAR
COURSE AREA/DOMAIN: C- Programming	CONTACT HOURS: 2 hours/Week.
CORRESPONDING LAB COURSE CODE : BECSE4S3TP	LAB COURSE NAME : Operating System Lab

COURSE PRE-REQUISITES:

C.CODE	COURSE NAME	DESCRIPTION	SEM
BECSE302T	C- Programming	Basic Concepts of C- Programming	III

LAB COURSE OBJECTIVES:

- This course explains the debate for each object oriented design principle.
- Draw a high level class diagram in UML for each pattern.
- Classify how the different components of the pattern collaborate with each other.
- List the consequences of each pattern to the overall software quality of a system.

COURSE OUTCOMES: Operating System

After completion of this course the students will be able –

SNO	DESCRIPTION	BLOOM'S TAXONOMY LEVEL
CO.1	**Compare** different type of operating system.	2
CO.2	**Make use of** various file allocation techniques.	1
CO.3	**Apply** scheduling algorithm and solve scheduling problems.	3
CO.4	**Develop** the program on memory management and page replacement algorithm.	6
CO.5	**Discover** the program using vi editor, the concurrency conditions and critical section problem.	4
CO.6	**Explain** the concept of deadlock and methods for its avoidance.	5

Lab Instructions:

- Make entry in the Log Book as soon as you enter the Laboratory.
- All the students should sit according to their Roll Numbers.
- All the students are supposed to enter the terminal number in the Log Book.
- Do not change the terminal on which you are working.
- Strictly observe the instructions given by the Faculty / Lab. Instructor.
- Take permission before entering in the lab and keep your belongings in the racks.
- NO FOOD, DRINK, IN ANY FORM is allowed in the lab.
- TURN OFF CELL PHONES! If you need to use it, please keep it in bags.
- Avoid all horseplay in the laboratory. Do not misbehave in the computer laboratory. Work quietly.
- Save often and keep your files organized.
- Don't change settings and surf safely.
- Do not reboot, turn off, or move any workstation or PC.
- Do not load any software on any lab computer (without prior permission of Faculty and Technical Support Personnel). Only Lab Operators and Technical Support Personnel are authorized to carry out these tasks.
- Do not reconfigure the cabling/equipment without prior permission.
- Do not play games on systems.
- Turn off the machine once you are done using it.
- Violation of the above rules and etiquette guidelines will result in disciplinary action.

Continuous Assessment Practical

Exp No	NAME OF EXPERIMENT	Date	Sign	Remark
1	Study and explain the types of operating systems (their types with structure, functionality, dependencies, application software with their differences).			
2	Installation of any one of the operating system (UBUNTU, CENT-OS).			
3	Write a C-program to present the Output of different file operation.			
4	Write a C-program to implement any file allocation technique (Linked, Indexed or Contiguous)			
5	Write a C-program to Present the Output of following CPU Scheduling algorithm. • FCFS • SJF • Priority • Round Robin			
6	Write a C-program to Present the Output of following Page Replacement Algorithm. • FIFO • LRU • OPTIMAL			
7	Write a C-program to implement memory management algorithm for memory management. • First fit • Best fit • Worst fit			
8	Write a C-program to present the Output for Producer– Consumer problem concept.			
9	Simulate Bankers algorithm for Deadlock Avoidance			
10	Write a C-program to implement Disk scheduling algorithms.			

CONTENTS

EXPERIMENT NO – 1

Aim: Study and explain the types of operating systems (their types with structure, functionality, dependencies, application software with their differences).

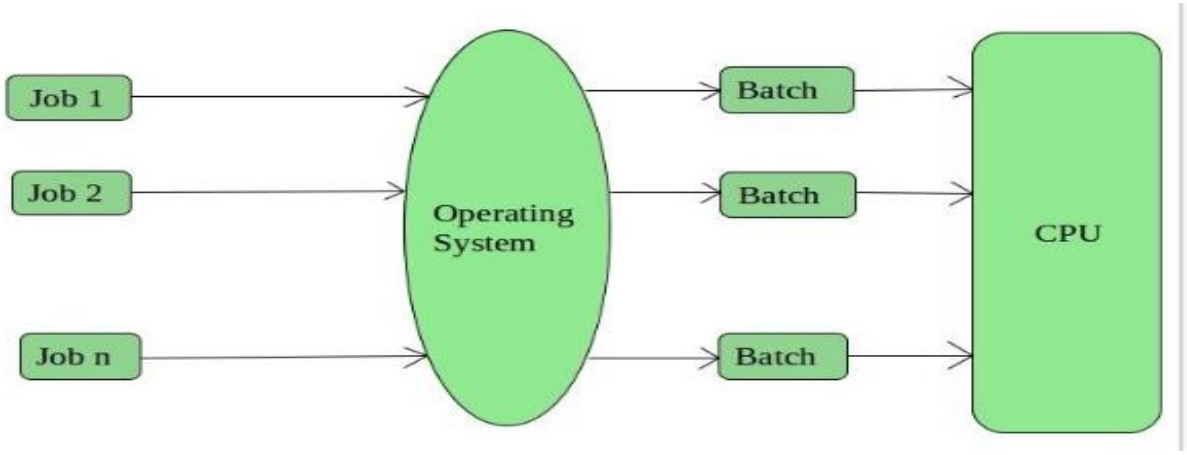
Theory:

An Operating System performs all the basic tasks like managing file, process, and memory. Thus operating system acts as manager of all the resources, i.e. **resource manager**. Thus operating system becomes an interface between user and machine.

Types of Operating Systems: Some of the widely used operating systems are as follows-

1. **Batch Operating System**

This type of operating system does not interact with the computer directly. There is an operator which takes similar jobs having same requirement and group them into batches. It is the responsibility of operator to sort the jobs with similar needs.



Advantages of Batch Operating System:

- It is very difficult to guess or know the time required by any job to complete Processors of the batch systems knows how long the job would be when it is in queue
- Multiple users can share the batch systems
- The idle time batch system is very less
- It is easy to manage large work repeatedly in batch systems

Disadvantages of Batch Operating System:

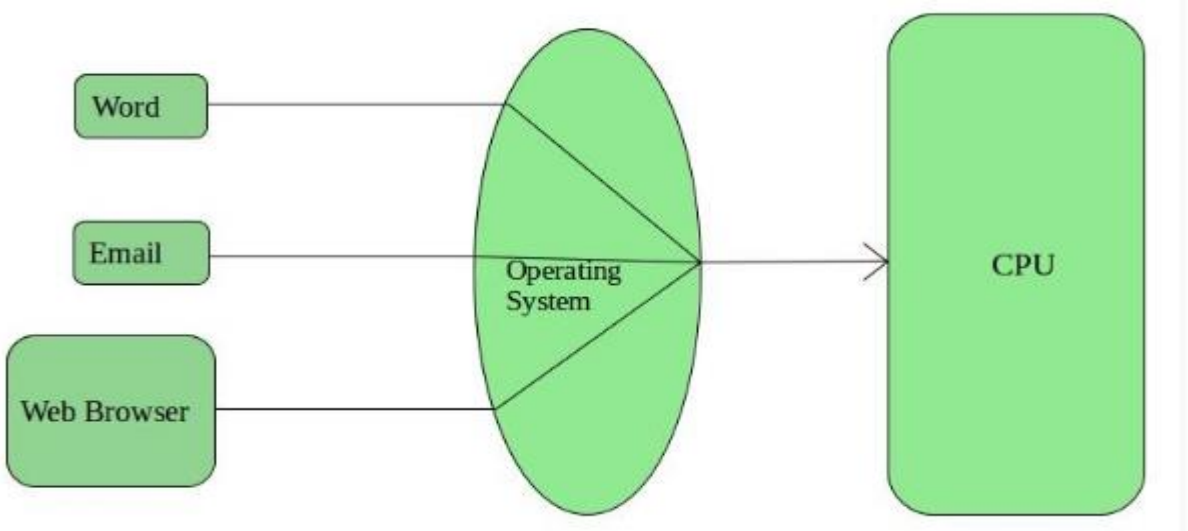
- The computer operators should be well known with batch systems
- Batch systems are hard to debug
- It is sometime costly
- The other jobs will have to wait for an unknown time if any job fails

Examples of Batch based Operating System:

- Payroll System, Bank Statements etc.

2. Time-Sharing Operating Systems

Each task has given some time to execute, so that all the tasks work smoothly. Each user gets time of CPU as they use single system. These systems are also known as Multitasking Systems. The task can be from single user or from different users also. The time that each task gets to execute is called quantum. After this time interval is over OS switches over to next task.



Advantages of Time-Sharing OS:

- Each task gets an equal opportunity.
- Less chances of duplication of software.
- CPU idle time can be reduced.

Disadvantages of Time-Sharing OS:

- Reliability problem.
- One must have to take care of security and integrity of user programs and data.
- Data communication problem.

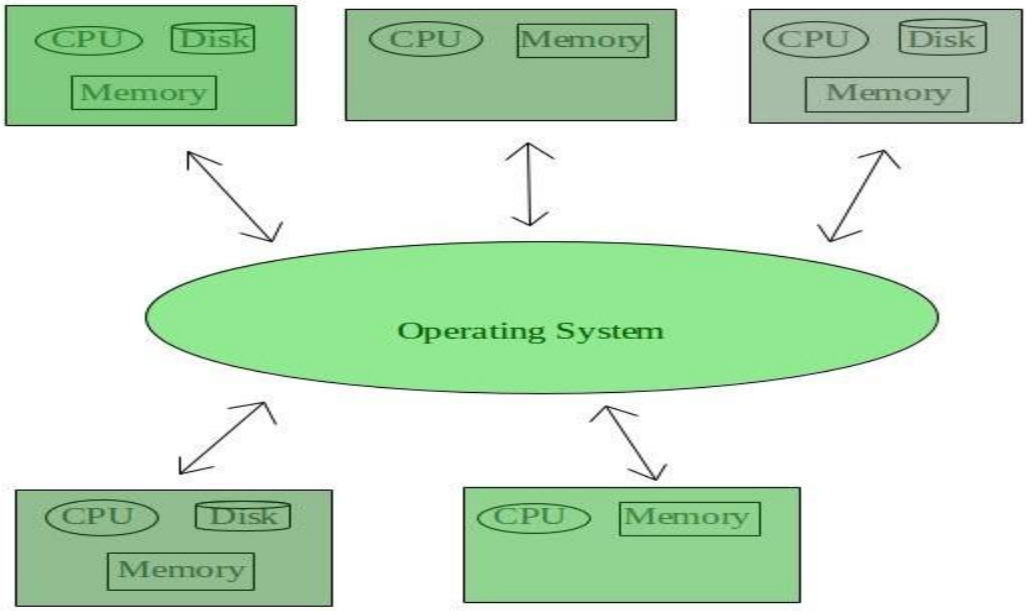
Examples of Time-Sharing OSs are:

- Multics, Unix etc.

3. Distributed Operating System

These types of operating system is a recent advancement in the world of computer technology and are being widely accepted all-over the world and, that too, with a great pace. Various autonomous interconnected computers communicate each other using a shared communication network. Independent systems possess their own memory unit and CPU. These are referred as **loosely coupled systems** or distributed systems. These systems processors differ in sizes and functions. The major benefit of working with these types of operating system is that it is always possible that one user can access the files or software which are not actually present on his system but on some other

system connected within this network i.e., remote access is enabled within the devices connected in that network.



Advantages of Distributed Operating System:

- Failure of one will not affect the other network communication, as all systems are independent from each other
- Electronic mail increases the data exchange speed
- Since resources are being shared, computation is highly fast and durable
- Load on host computer reduces
- These systems are easily scalable as many systems can be easily added to the network
- Delay in data processing reduces

Disadvantages of Distributed Operating System:

- Failure of the main network will stop the entire communication
- To establish distributed systems the language which are used are not well defined yet
- These types of systems are not readily available as they are very expensive. Not only that the underlying software is highly complex and not understood well yet

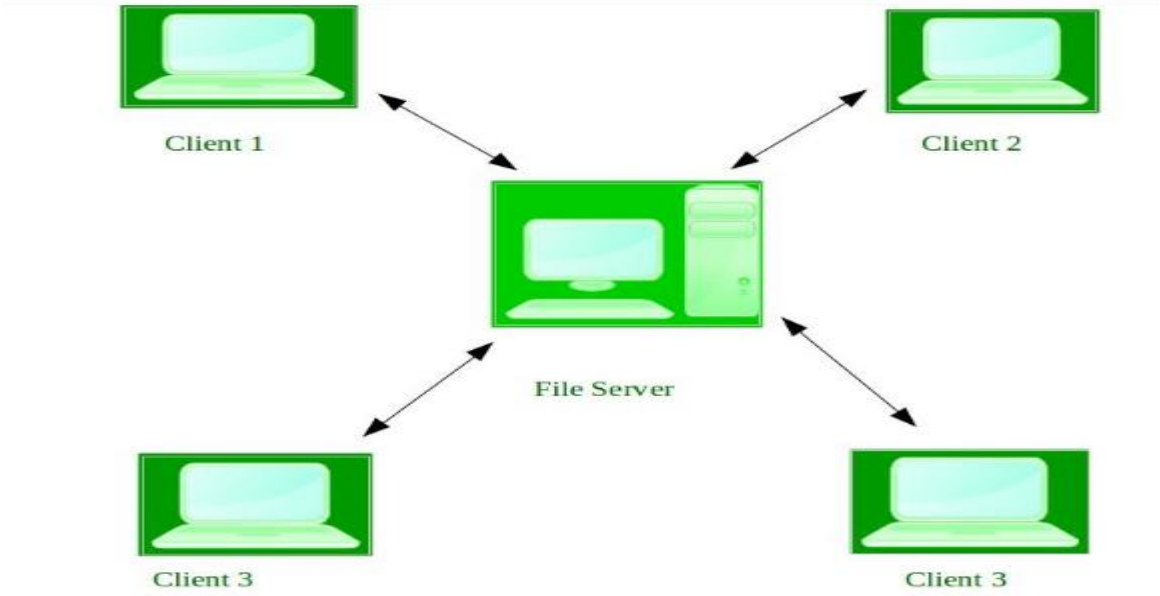
Examples of Distributed Operating System are-

- LOCUS etc.

4. Network Operating System

These systems run on a server and provide the capability to manage data, users, groups, security, applications, and other networking functions. These types of operating systems allows shared access of files, printers, security, applications, and other networking functions over a small private network. One more important aspect

of Network Operating Systems is that all the users are well aware of the underlying configuration, of all other users within the network, their individual connections etc. and that's why these computers are popularly known as **tightly coupled systems**.



Advantages of Network Operating System:

- Highly stable centralized servers
- Security concerns are handled through servers
- New technologies and hardware up-gradation are easily integrated to the system
- Server access are possible remotely from different locations and types of systems

Disadvantages of Network Operating System:

- Servers are costly
- User has to depend on central location for most operations
- Maintenance and updates are required regularly

Examples of Network Operating System are:

- Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD etc.

5. Real-Time Operating System

These types of OSs serve the real-time systems. The time interval required to process and respond to inputs is very small. This time interval is called **response time**.

Real-time systems are used when there are times requirements are very strict like missile systems, air traffic control systems, robots etc.

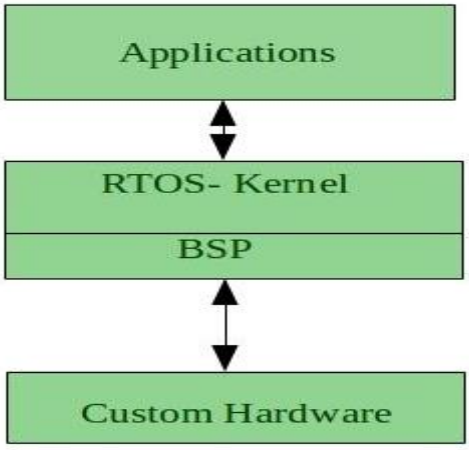
Two types of Real-Time Operating System which are as follows:

- **Hard Real-Time Systems:**

These OSs are meant for the applications where time constraints are very strict and even the shortest possible delay is not acceptable. These systems are built for saving life like automatic parachutes or air bags which are required to be readily available in case of any accident. Virtual memory is almost never found in these systems.

- **Soft Real-Time Systems:**

These OSs are for applications where for time-constraint is less strict.



Advantages of RTOS:

- **Maximum Consumption:** Maximum utilization of devices and system, thus more output from all the resources
- **Task Shifting:** Time assigned for shifting tasks in these systems are very less. For example in older systems it takes about 10 micro seconds in shifting one task to another and in latest systems it takes 3 micro seconds.
- **Focus on Application:** Focus on running applications and less importance to applications which are in queue.
- **Real time operating system in embedded system:** Since size of programs is small, RTOS can also be used in embedded systems like in transport and others.
- **Error Free:** These types of systems are error free.
- **Memory Allocation:** Memory allocation is best managed in these types of systems.

Disadvantages of RTOS:

- **Limited Tasks:** Very few tasks run at the same time and their concentration is very less on few applications to avoid errors.

- **Use heavy system resources:** Sometimes the system resources are not so good and they are expensive as well.
- **Complex Algorithms:** The algorithms are very complex and difficult for the designer to write on.
- **Device driver and interrupt signals:** It needs specific device drivers and interrupts signals to response earliest to interrupts.
- **Thread Priority:** It is not good to set thread priority as these systems are very less pron to switching tasks.

Examples of Real-Time Operating Systems are:

- Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

Sr. No.	Multiprocessing	Multiprogramming
1	Multiprocessing refers to processing of multiple processes at same time by multiple CPUs.	Multiprogramming keeps several programs in main memory at the same time and execute them concurrently utilizing single CPU.
2	It utilizes multiple CPUs.	It utilizes single CPU.
3	It permits parallel processing.	Context switching takes place.
4	Less time taken to process the jobs.	More Time taken to process the jobs.
5	It facilitates much efficient utilization of devices of the computer system.	Less efficient than multiprocessing.
6	Usually more expensive.	Such systems are less expensive.

Sr. No.	Multiprocessing	Multiprogramming
1	Multiprocessing refers to processing of multiple processes at same time by multiple CPUs.	Multiprogramming keeps several programs in main memory at the same time and execute them concurrently utilizing single CPU.
2	It utilizes multiple CPUs.	It utilizes single CPU.
3	It permits parallel processing.	Context switching takes place.
4	Less time taken to process the jobs.	More Time taken to process the jobs.
5	It facilitates much efficient utilization of devices of the computer system.	Less efficient than multiprocessing.
6	Usually more expensive.	Such systems are less expensive.

BASIS FOR COMPARISON	TIME SHARING OPERATING SYSTEM	REAL-TIME OPERATING SYSTEM
Basic	Emphasis on providing a quick response to a request.	It focuses on accomplishing a computational task before its specified deadline.
Computer resources	Shared between the user.	No sharing takes place and events are external to the system.
Process deals with	More than one application simultaneously.	Single application at a time.
Modification of the program	The programs can be modified and written by the users.	No modification is possible.
Response	The response is generated within the second, but there is no compulsion.	User must get the response within the defined time constraint.

CONCLUSION: Thus we, have studied and explain the various types of operating systems successfully.

Viva Voce Question

1. **Explain the main purpose of an operating system?**

2. **What is kernel?**

3. **What are Real-time systems?**

4. **Why operating system called as recourse manager?**

Signature of Subject Teacher

EXPERIMENT NO – 2

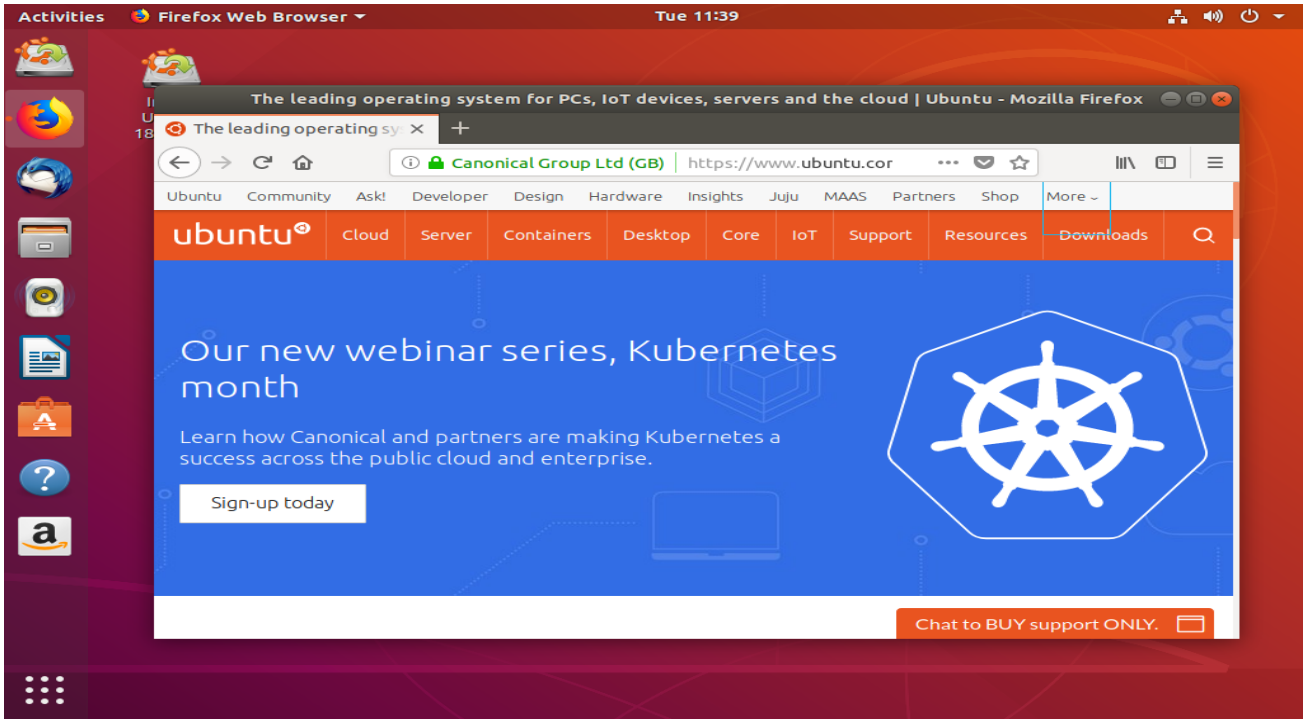
Aim: Installation of any one of the operating system (UBUNTU, CENT-OS)

Theory:

Ubuntu is one of the most popular and easy-to-use versions of Linux available, and you can download and install it absolutely free. All you need is a CD burner and an internet connection, and you can have Ubuntu up and running in just a few minutes. Follow this guide to learn how to install Ubuntu.

1. **Overview**

The Ubuntu desktop is easy to use, easy to install and includes everything you need to run your organisation, school, home or enterprise. It's also open source, secure, accessible and free to download.



In this tutorial, we're going to install Ubuntu desktop onto your computer, using either your computer's DVD drive or a USB flash drive.

2. Requirements

You'll need to consider the following before starting the installation:

- Connect your laptop to a power source.
- Ensure you have at least 25GB of free storage space, or 5GB for a minimal installation.

- Have access to either a DVD or a USB flash drive containing the version of Ubuntu you want to install.
- Make sure you have a recent backup of your data. While it's unlikely that anything will go wrong, you can never be too prepared.

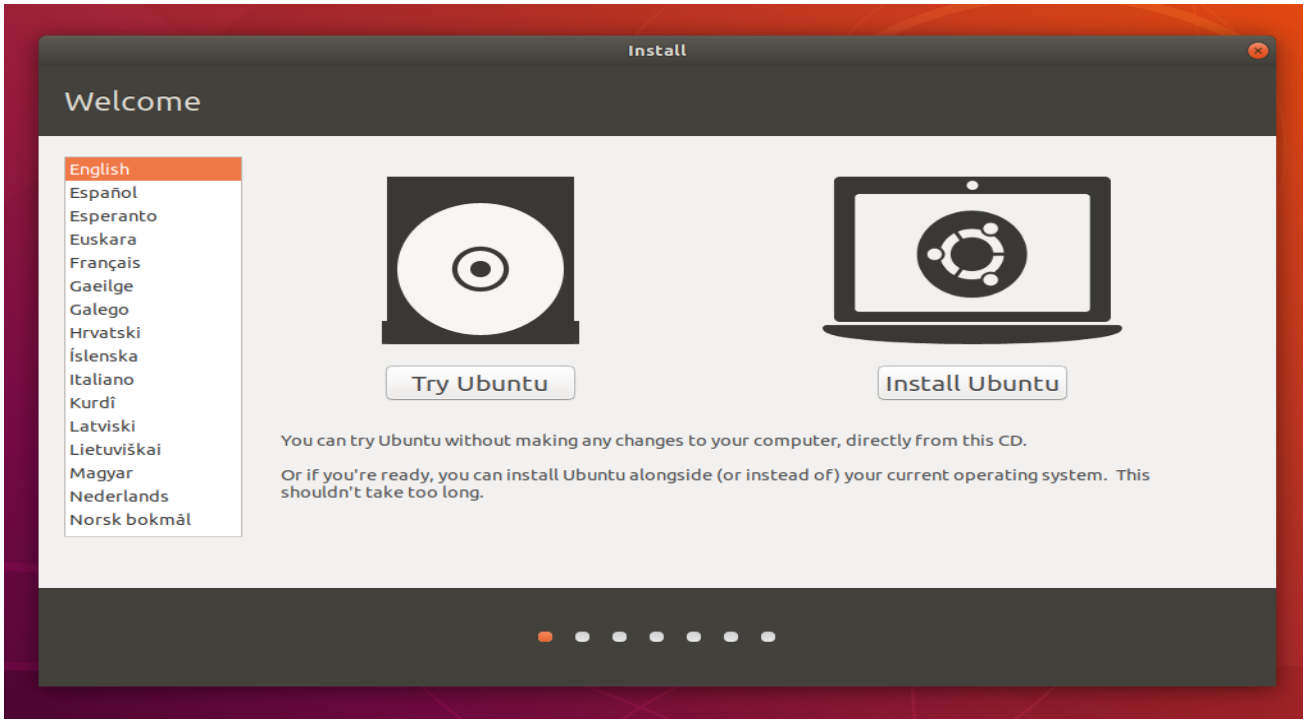
See Installation/System Requirements for more specific details on hardware requirements. We also have tutorials that explain how to create an Ubuntu DVD or USB flash drive.

3. Boot from DVD

It's easy to install Ubuntu from a DVD. Here's what you need to do:

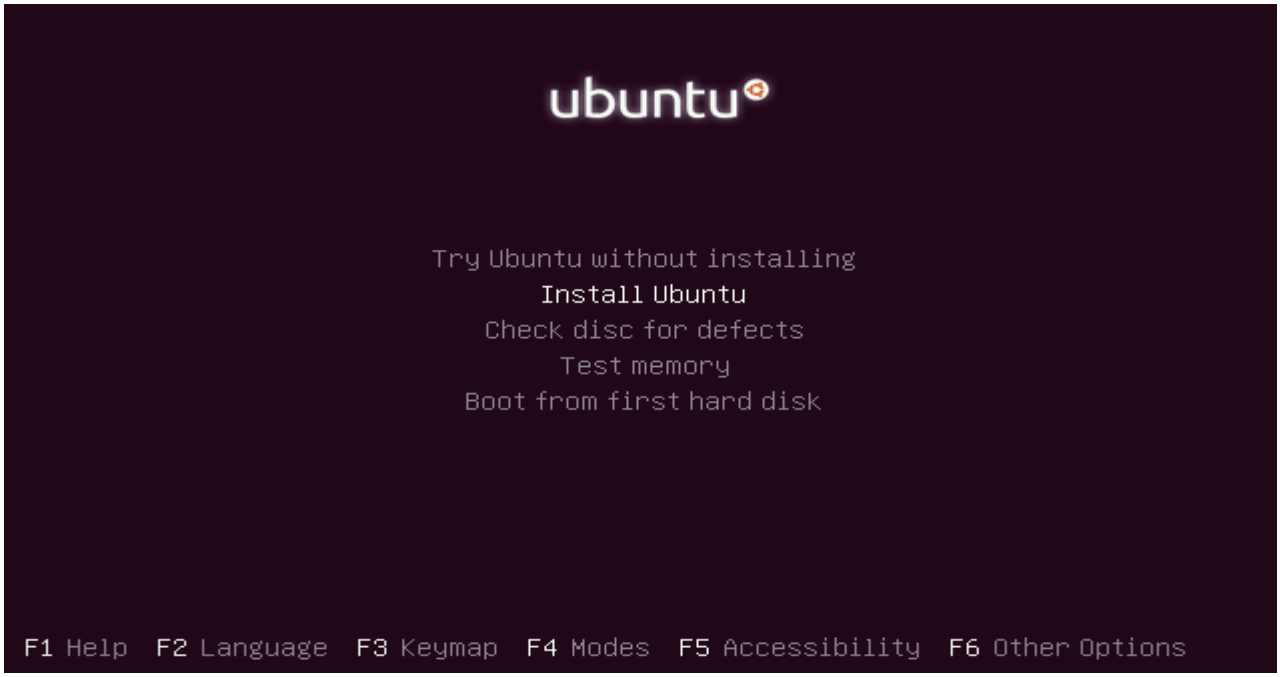
1. Put the Ubuntu DVD into your optical/DVD drive.
2. Restart your computer.

As soon as your computer boots you'll see the welcome window.



From here, you can select your language from a list on the left and choose between either installing Ubuntu directly, or trying the desktop first (if you like what you see, you can also install Ubuntu from this mode too).

Depending on your computer's configuration, you may instead see an alternative boot menu showing a large language selection pane. Use your mouse or cursor keys to select a language and you'll be presented with a simple menu.



Select the second option, 'Install Ubuntu', and press return to launch the desktop installer automatically. Alternatively, select the first option, 'Try Ubuntu without installing', to test Ubuntu (as before, you can also install Ubuntu from this mode too).

A few moments later, after the desktop has loaded, you'll see the welcome window. From here, you can select your language from a list on the left and choose between either installing Ubuntu directly, or trying the desktop first.

If you don't get either menu, read the booting from the DVD guide for more information.

4. Boot from USB flash drive

Most computers will boot from USB automatically. Simply insert the USB flash drive and either power on your computer or restart it. You should see the same welcome window we saw in the previous 'Install from DVD' step, prompting you to choose your language and either install or try the Ubuntu desktop.

If your computer doesn't automatically boot from USB, try holding F12 when your computer first starts. With most machines, this will allow you to select the USB device from a system-specific boot menu.

F12 is the most common key for bringing up your system's boot menu, but Escape, F2 and F10 are common alternatives. If you're unsure, look for a brief message when your system starts - this will often inform you of which key to press to bring up the boot menu.

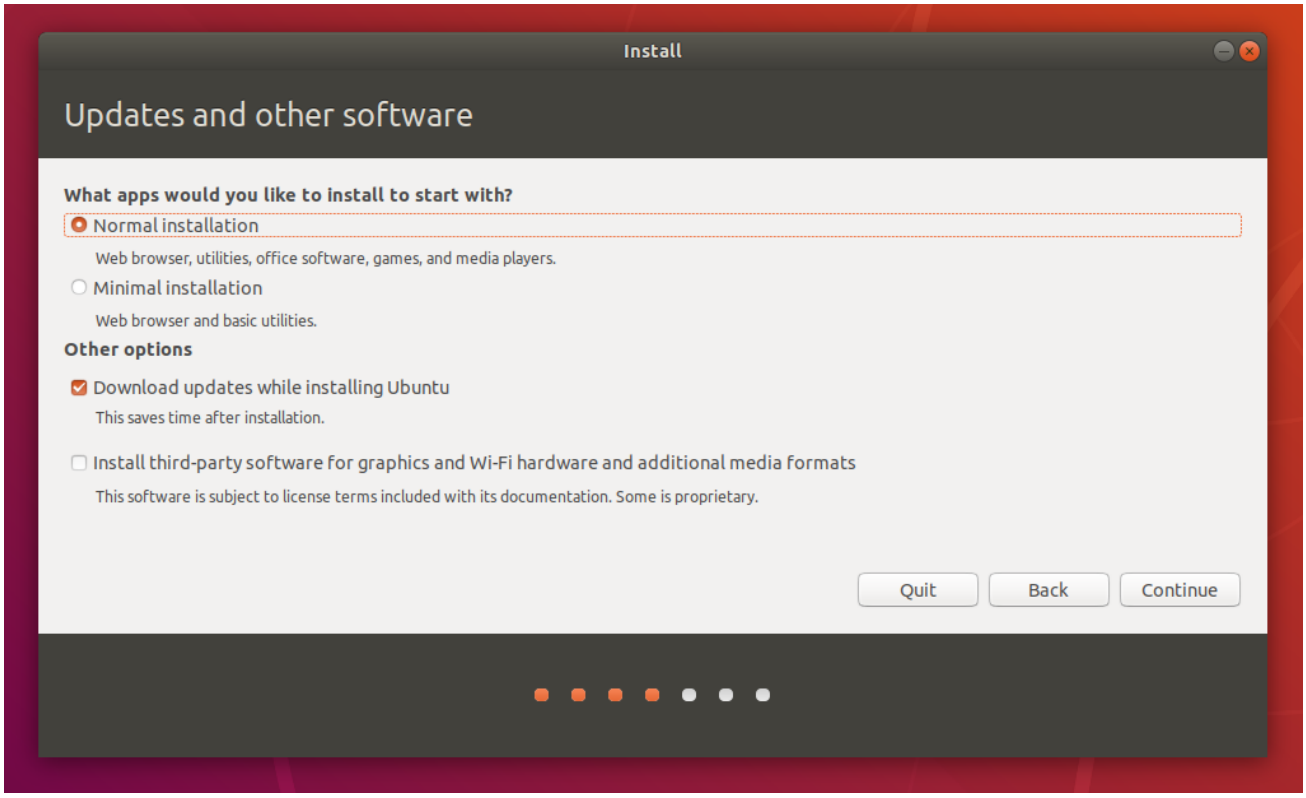
5. Prepare to install Ubuntu

You will first be asked to select your keyboard layout. If the installer doesn't guess the default layout correctly, use the 'Detect Keyboard Layout' button to run through a brief configuration procedure.

After selecting *Continue* you will be asked *What apps would you like to install to start with?* The two options are 'Normal installation' and 'Minimal installation'. The first is the equivalent to the old default bundle of utilities, applications, games and media players - a great launchpad for any Linux installation. The second takes considerably less storage space and allows you to install only what you need.

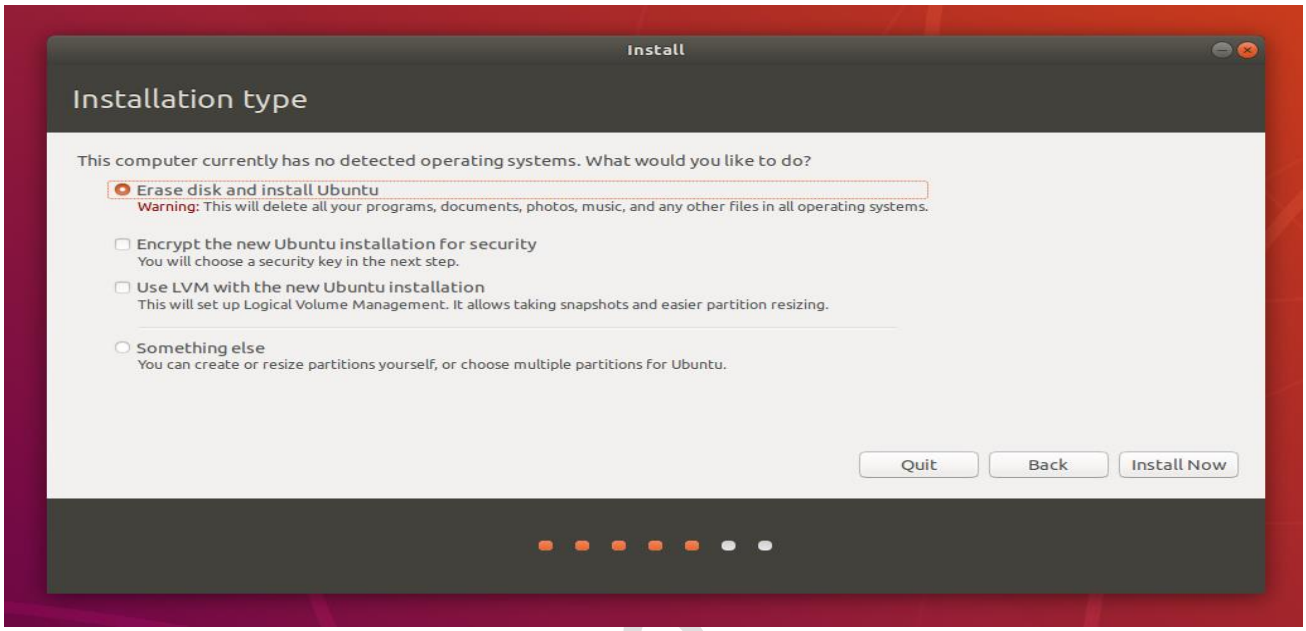
Beneath the installation-type question are two checkboxes; one to enable updates while installing and another to enable third-party software.

- We advise enabling both Download updates and Install third-party software.
- Stay connected to the internet so you can get the latest updates while you install Ubuntu.
- If you are not connected to the internet, you will be asked to select a wireless network, if available. We advise you to connect during the installation so we can ensure your machine is up to date



6. Allocate drive space

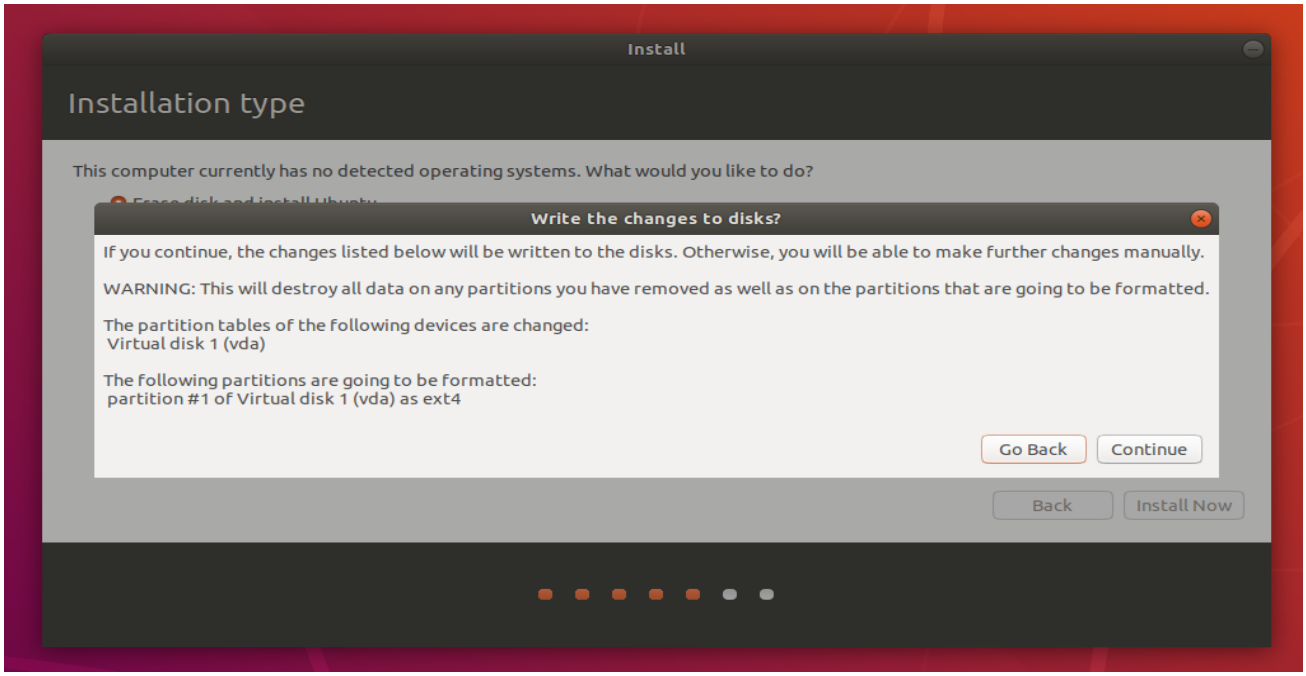
Use the checkboxes to choose whether you'd like to install Ubuntu alongside another operating system, delete your existing operating system and replace it with Ubuntu, or if you're an advanced user — choose the '**Something else**' option.



7. Begin installation

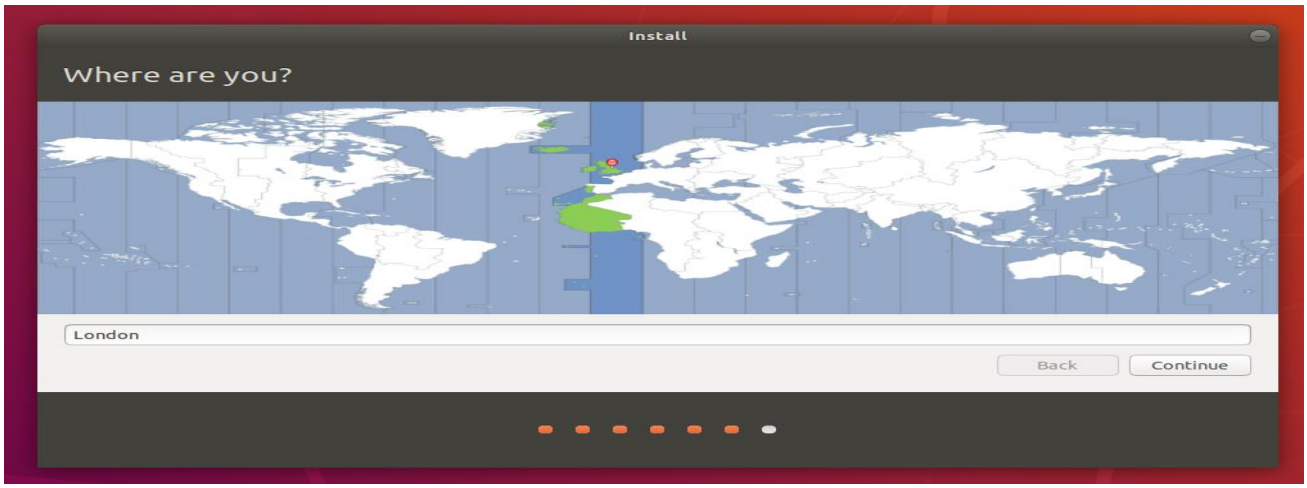
After configuring storage, click on the 'Install Now' button. A small pane will appear with an overview of the storage options you've chosen, with the chance to go back if the details are incorrect.

Click Continue to fix those changes in place and start the installation process.



8. Select your location

If you are connected to the internet, your location will be detected automatically. Check your location is correct and clicks 'Forward' to proceed. If you're unsure of your time zone, type the name of a local town or city or use the map to select your location.

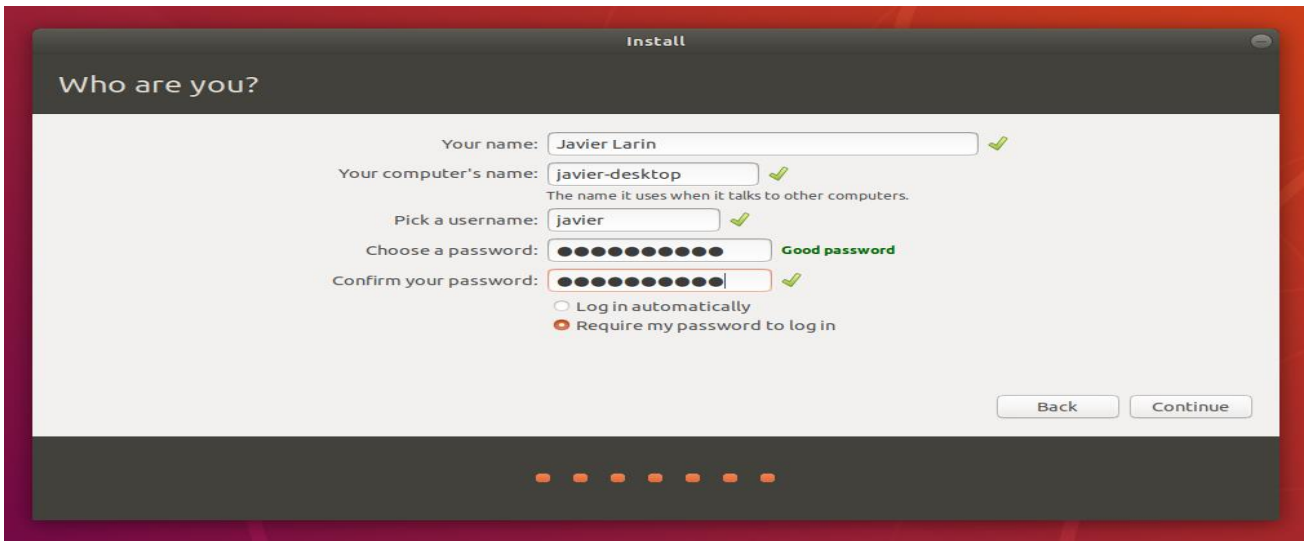


9. Login details

Enter your name and the installer will automatically suggest a computer name and username. These can easily be changed if you prefer. The computer name is how your computer will appear on the network, while your username will be your login and account name.

Next, enter a strong password. The installer will let you know if it's too weak.

You can also choose to enable automatic login and home folder encryption. If your machine is portable, we recommend keeping automatic login disabled and enabling encryption. This should stop people accessing your personal files if the machine is lost or stolen.



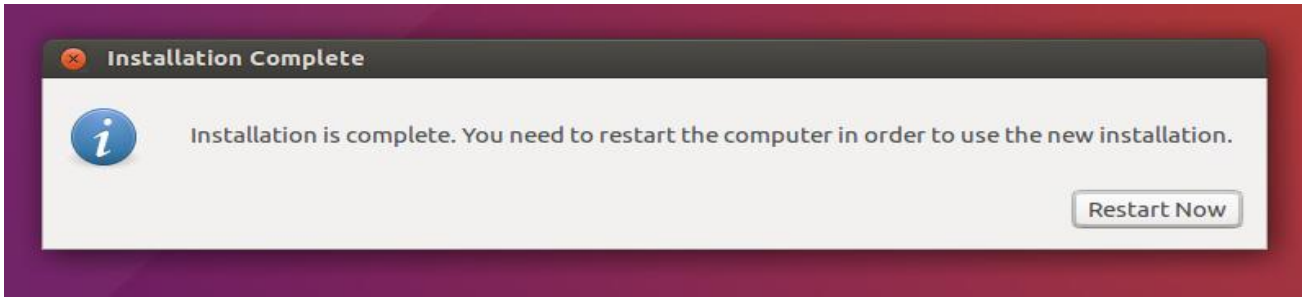
If you enable home folder encryption and you forget your password, you won't be able to retrieve any personal data stored in your home folder.

10. Background installation

The installer will now complete in the background while the installation window teaches you a little about how awesome Ubuntu is. Depending on the speed of your machine and network connection, installation should only take a few minutes.

11. Installation complete

After everything has been installed and configured, a small window will appear asking you to restart your machine. Click on Restart Now and remove either the DVD or USB flash drive when prompted. If you initiated the installation while testing the desktop, you also get the option to continue testing.



Congratulations! You have successfully installed the world's most popular Linux operating system!

It's now time to start enjoying Ubuntu!

CONCLUSION: Thus we, have studied and successfully installed the UBUNTU Operating system.

Viva Voce Question

1. **What is Linux?**

2. **What is the difference between UNIX and LINUX?**

3. **What is the advantage of open source operating system?**

4. **Why we use UBUNTU OS?**

Signature of Subject Teacher

EXPERIMENT NO – 3

Aim: Write a C-program to present the output of different file operation.

Theory:

File:

A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

File Structure

- A File Structure should be according to a required format that the operating system can understand.
- A file has a certain defined structure according to its type.
- A text file is a sequence of characters organized into lines.
- A source file is a sequence of procedures and functions.
- An object file is a sequence of bytes organized into blocks that are understandable by the machine.
- When operating system defines different file structures, it also contains the code to support these file structure. Unix, MS-DOS support minimum number of file structure.

File Type

File type refers to the ability of the operating system to distinguish different types of file such as text files source files and binary files etc. Many operating systems support many types of files. Operating system like MS-DOS and UNIX have the following types of files −

Ordinary files

- These are the files that contain user information.
- These may have text, databases or executable program.
- The user can apply various operations on such files like add, modify, delete or even remove the entire file.

Directory files

- These files contain list of file names and other information related to these files.

Special files

- These files are also known as device files.
- These files represent physical device like disks, terminals, printers, networks, tape drive etc.

These files are of two types −

- **Character special files** − data is handled character by character as in case of terminals or printers.
- **Block special files** − data is handled in blocks as in the case of disks and tapes.

Types of File Operations

Files are not made for just reading the Contents; we can also perform some other operations on the Files those are explained below as:

- Opening an existing file
- Read Operation: Meant To Read the information which is Stored into the Files.
- Write Operation: For inserting some new Contents into a File.
- Rename or Change the Name of File.
- Copy the File from one Location to another.
- Sorting or Arrange the Contents of File.
- Move or Cut the File from One Place to Another.
- Delete a File
- Execute Means to Run Means File Display Output.

Working with files

When working with files, you need to declare a pointer of type file. This declaration is needed for communication between the file and program.

FILE *fptr;

Opening a file - for creation and edit

Opening a file is performed using the library function in the **"stdio.h"** header file: fopen().

The syntax for opening a file in standard I/O is:
ptr = fopen("fileopen","mode")

For Example:

```
fopen("E:\\cprogram\\newprogram.txt","w");

fopen("E:\\cprogram\\oldprogram.bin","rb");
```

- Let's suppose the file newprogram.txt doesn't exist in the location E:\cprogram. The first function creates a new file named newprogram.txt and opens it for writing as per the mode 'w'.
  The writing mode allows you to create and edit (overwrite) the contents of the file.
- Now let's suppose the second binary file oldprogram.bin exists in the location E:\cprogram. The second function opens the existing file for reading in binary mode 'rb'.
  The reading mode only allows you to read the file, you cannot write into the file.

Opening Modes in Standard I/O

File Mode	Meaning of Mode	During Inexistence of file
r	Open for reading.	If the file does not exist, fopen() returns NULL.
rb	Open for reading in binary mode.	If the file does not exist, fopen() returns NULL.
w	Open for writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
wb	Open for writing in binary mode.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
a	Open for append. i.e, Data is added to end of file.	If the file does not exist, it will be created.
ab	Open for append in binary mode. i.e, Data is added to end of file.	If the file does not exist, it will be created.

Opening Modes in Standard I/O		
File Mode	**Meaning of Mode**	**During Inexistence of file**
r+	Open for both reading and writing.	If the file does not exist, fopen() returns NULL.
rb+	Open for both reading and writing in binary mode.	If the file does not exist, fopen() returns NULL.
w+	Open for both reading and writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
wb+	Open for both reading and writing in binary mode.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
a+	Open for both reading and appending.	If the file does not exist, it will be created.
ab+	Open for both reading and appending in binary mode.	If the file does not exist, it will be created.

Closing a File

The file (both text and binary) should be closed after reading/writing.
Closing a file is performed using library function fclose().

fclose(fptr); //fptr is the file pointer associated with file to be closed.

PROGRAM:

OUTPUT:

CONCLUSION: Thus we, have implemented the C program for different file operations successfully.

Viva Voce Question

1. **Explain different types of files?**

2. **Give the various types of file extensions?**

3. **Can you create a program to merge a file with another file?**

Signature of Subject Teacher

EXPERIMENT NO – 4

Aim: Write a C-program to implement any file allocation technique (Linked, Indexed or Contiguous) (any one)

Theory:

Allocation Method:

The allocation method defines how the files are stored in the disk blocks. The direct access nature of the disks gives us the flexibility to implement the files. In many cases, different files or many files are stored on the same disk. The main problem that occurs in the operating system is that how we allocate the spaces to these files so that the utilization of disk is efficient and the quick access to the file is possible. There are mainly three methods of file allocation in the disk. Each method has its advantages and disadvantages. Mainly a system uses one method for all files within the system.

- Contiguous allocation
- Linked allocation
- Indexed allocation

The main idea behind contiguous allocation methods is to provide

- Efficient disk space utilization
- Fast access to the file blocks

Contiguous allocation

In this scheme, a file is made from the contiguous set of blocks on the disk. Linear ordering on the disk is defined by the disk addresses. In this scheme only one job is accessing the disk block b after that it accesses the block b+1 and there are no head movements. When the movement of the head is needed the head moves only from one track to another track. So the disk number that is required for accessing the contiguous allocation is minimal. Contiguous allocation method provides a good performance that's why it is used by the IBM VM/CMS operating system. For example, if a file requires n blocks and is given a block b as the starting location, then the blocks assigned to the file will be: **b, b+1, b+2,..., b+n-1**. This means that given the starting block address and the length of the file (in terms of blocks required), we can determine the blocks occupied by the file. For a contiguous allocation the directory entry the address of the starting block and Length of the allocated portion.

The file **'A'** in the following figure starts from block 19 with **length = 6 blocks**. Therefore, it occupies **19, 20, 21, 22, 23, 24** blocks.

- Each file in the disk occupies a contiguous address space on the disk.
- In this scheme, the address is assigned in the linear fashion.
- It is very easy to implement the contiguous allocation method.
- In the contiguous allocation technique, external fragmentation is a major issue.

Advantages:

1. In the contiguous allocation, sequential and direct access both is supported.
2. For the direct access, the starting address of the kth block is given and further blocks are obtained by b+K,
3. This is very fast and the number of seeks is minimal in the contiguous allocation method.

Disadvantages:

1. Contiguous allocation method suffers internal as well as external fragmentation.
2. In terms of memory utilization, this method is inefficient.
3. It is difficult to increase the file size because it depends on the availability of contiguous memory.

Example:

File	Start	Length
Count	0	2
Tr	14	3
Mail	19	6
List	28	4

ALGORITHM:

Step 1: Start the program.

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

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

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

Step 5: Best-fit algorithm searches the memory blocks for the smallest hole which can

be allocated to requesting process and allocates if.

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

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

Step 8: Stop the program.

PROGRAM:

OUTPUT:

Linked allocation

The problems of contiguous allocation are solved in the linked allocation method. In this scheme, disk blocks are arranged in the linked list form which is not contiguous. The disk block is scattered in the disk. In this scheme, the directory entry contains the pointer of the first block and pointer of the ending block. These pointers are not for the users. For example, a file of six blocks starts at block 10 and end at the block. Each pointer contains the address of the next block. When we create a new file we simply create a new entry with the linked allocation. Each directory contains the pointer to the first disk block of the file. When the pointer is nil then it defines the empty file.

Advantages:

1. In terms of the file size, this scheme is very flexible.
2. We can easily increase or decrease the file size and system does not worry about the contiguous chunks of memory.

3. This method free from external fragmentation this makes it better in terms of memory utilization.

Disadvantages:

1. In this scheme, there is large no of seeks because the file blocks are randomly distributed on disk.
2. Linked allocation is comparatively slower than contiguous allocation.
3. Random or direct access is not supported by this scheme we cannot access the blocks directly.
4. The pointer is extra overhead on the system due to the linked list.

ALGORITHM:

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

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

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

Step 4: Create a stack

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

Step 6: Stop the allocation.

PROGRAM:

OUTPUT:

Indexed Allocation

In this scheme, a special block known as the index block contains the pointer to all the blocks occupied by a file. Each file contains its index which is in the form of an array of disk block addresses. The ith entry of index block point to the ith block of the file. The address of the index block is maintained by the directory. When we create a file all pointers is set to nil. A block is obtained from the free space manager when the first ith block is written. When the index block is very small it is difficult to hold all the pointers for the large file. to deal with this issue a mechanism is available. Mechanism includes the following:

- Linked scheme
- Multilevel scheme
- Combined scheme

Advantages:

1. This scheme supports random access of the file.
2. This scheme provides fast access to the file blocks.
3. This scheme is free from the problem of external fragmentation.

Disadvantages:

1. The pointer head is relatively greater than the linked allocation of the file.
2. Indexed allocation suffers from the wasted space.
3. For the large size file, it is very difficult for single index block to hold all the pointers.
4. For very small files say files that expend only 2-3 blocks the indexed allocation would keep on the entire block for the pointers which is insufficient in terms of memory utilization.

ALGORITHM:

Step 1: Start.

Step 2: Let n be the size of the buffer

Step 3: check if there are any producer

Step 4: if yes check whether the buffer is full

Step 5: If no the producer item is stored in the buffer

Step 6: If the buffer is full the producer has to wait

Step 7: Check there is any consumer. If yes check whether the buffer is empty

Step 8: If not the consumer consumes them from the buffer

Step 9: If the buffer is empty, the consumer has to wait.

Step 10: Repeat checking for the producer and consumer till required

Step 11: Terminate the process.

PROGRAM:

OUTPUT:

CONCLUSION: Thus we, have implement the C-program for any file allocation technique (Linked, Indexed or Contiguous) successfully.

Viva Voce Question

1. **What is a file? What do you understand by file allocation techniques?**

2. **What are the different types of file access techniques?**

3. **What is contiguous memory allocation?**

4. **What is External Fragmentation?**

Signature of Subject Teacher

EXPERIMENT NO – 5

Aim: Write a C-program to Present the Output of following CPU Scheduling algorithm.(any one)

- FCFS
- SJF
- Priority
- Round Robin

Process scheduling is a task of operating system to schedules the processes of different states like ready, running, waiting. To study about process states you can refer Process Management in Operating Systems according to their priorities. This task is very useful in maintain the computer system. Process scheduling allocates the time interval of each process in which the process is to be executed by the central processing unit (CPU).

Process scheduling is very important in **multiprogramming and multitasking operating system**, where multiple processes execute simultaneously. To study about multiprogramming and multitasking operating system you can refer Introduction of O.S. and Its types Process scheduling ensures maximum utilization of central processing unit (CPU) because a process is always running at the specific instance of time. At first, the processes that are to be executed are placed in a queue called **Job queue**. The processes which are already placed in the main memory and are ready for CPU allocation, are placed in a queue called **Ready queue**. If the process is waiting for input / output, then that process is placed in the queue called **Device queue**.

An operating system uses a program scheduler to schedules the processes of computer system. The schedulers are of following types:

1. Long term scheduler
2. Mid - term scheduler
3. Short term scheduler

1) Long Term Scheduler

It selects the process that is to be placed in ready queue. The long term scheduler basically decides the priority in which processes must be placed in main memory. Processes of long term scheduler are placed in the ready state because in this state the process is ready to execute waiting for calls of execution from CPU which takes time that's why this is known as long term scheduler.

2) Mid – Term Scheduler

It places the blocked and suspended processes in the secondary memory of a computer system. The task of moving from main memory to secondary memory is called **swapping out**. The task of moving back a swapped out process from secondary

memory to main memory is known as **swapping in**. The swapping of processes is performed to ensure the best utilization of main memory.

3) Short Term Scheduler

It decides the priority in which processes is in the ready queue are allocated the central processing unit (CPU) time for their execution. The short term scheduler is also referred as central processing unit (CPU) scheduler.

An operating system uses two types of scheduling processes execution, **preemptive** and **non - preemptive**.

1. **Preemptive process:**

In preemptive scheduling policy, a low priority process has to be suspending its execution if high priority process is waiting in the same queue for its execution.

2. **Non Preemptive process:**

In non - preemptive scheduling policy, processes are executed in first come first serve basis, which means the next process is executed only when currently running process finishes its execution.

Operating system performs the task of scheduling processes based on priorities using these following algorithms:

1. **First come first serve (FCFS)**
In this scheduling algorithm the first process entered in queue is processed first.

2. **Shortest job first (SJF)**
In this scheduling algorithm the process which requires shortest CPU time to execute is processed first.

3. **Priority scheduling**
In this scheduling algorithm the priority is assigned to all the processes and the process with highest priority executed first. Priority assignment of processes is done on the basis of internal factor such as CPU and memory requirements or external factor such as user's choice. The priority scheduling algorithm supports preemptive and non - preemptive scheduling policy.

4. **Round Robin (RR) scheduling**
In this algorithm the process is allocated the CPU for the specific time period called **time slice**, which is normally of 10 to 100 milliseconds. If the process completes its execution within this time slice, then it is removed from the queue otherwise it has to wait for another time slice.

Algorithm for FCFS scheduling:

Step 1: Start the process

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

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

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

Step 5: for each process in the Ready Q calculate

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

(d) Turnaround time for Process (n) = waiting time of Process(n)+ Burst time for Process (n)

Step 6: Calculate

(e) Average waiting time = Total waiting Time / Number of process

(f) Average Turnaround time = Total Turnaround Time / Number of process

Step 7: Stop the process

Algorithm for SJF

Step 1: Start the process

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

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

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

highest burst time.

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

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

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

process (n-1)

(b) Turnaround time for Process (n)= waiting time of Process(n)+ Burst time for process(n)

Step 6: Calculate

(a) Average waiting time = Total waiting Time / Number of process

(b) Average Turnaround time = Total Turnaround Time / Number of process

Step 7: Stop the process

Algorithm for Priority Scheduling:

Step 1: Start the process

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

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

Step 4: Sort the ready queue according to the priority number.

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

Step 6: For each process in the Ready Q calculate

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

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

Step 7: Calculate

(a) Average waiting time = Total waiting Time / Number of process

(b) Average Turnaround time = Total Turnaround Time / Number of process

Step 8: Stop the process

Algorithm for RR

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue and time quantum (or) time slice

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

Step 4: Calculate the no. of time slices for each process where

No. of time slice for process (n) = burst time process (n)/time slice

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

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

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

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

Step 7: Calculate

(a) Average waiting time = Total waiting Time / Number of process

(b) Average Turnaround time = Total Turnaround Time / Number of process

Step 8: Stop the process

PROGRAM:

- **Program for First Come First Served (FCFS) Scheduling Algorithm:**

OUTPUT:

- **Program for shortest job first (SJF) scheduling algorithm:**

OUTPUT:

- **Program for Priority Scheduling algorithm.**

OUTPUT:

- **Program for Round Robin scheduling algorithm**

OUTPUT:

CONCLUSION: Thus we, have implement the C-program to present the output for CPU Scheduling algorithm successfully.

Viva Voce Question

In general, which CPU scheduling algorithm works with highest waiting time?

Define arrival time, burst time, waiting time, turnaround time?

What is the advantage of round robin CPU scheduling algorithm?

Define a process?

Signature of Subject Teacher

EXPERIMENT NO – 6

Aim Write a C-program to Present the Output of following Page Replacement Algorithm.(any one)

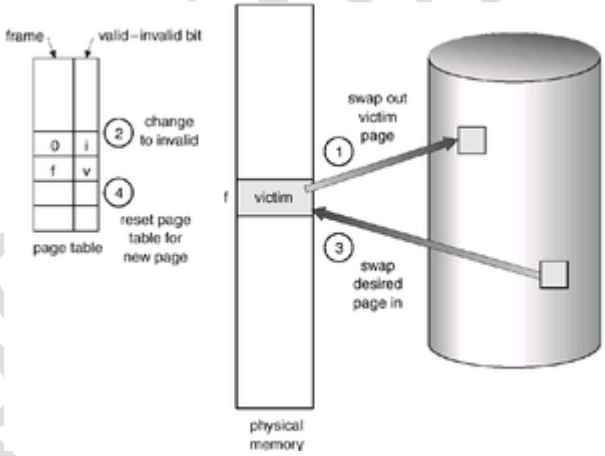
a. FIFO
b. LRU
c. OPTIMAL

Theory:

Page Replacement Algorithm:

The page replacement is a mechanism that loads a page from disc to memory when a page of memory needs to be allocated. Page replacement can be described as follows:

- Find the location of the desired page on the disk.
- Find a free frame:
- If there is a free frame, use it.
- If there is no free frame, use a page-replacement algorithm to select a victim frame.
- Write the victim page to the disk; change the page and frame tables accordingly.
- Read the desired page into the (newly) free frame; change the page and frame tables.
- Restart the user process.



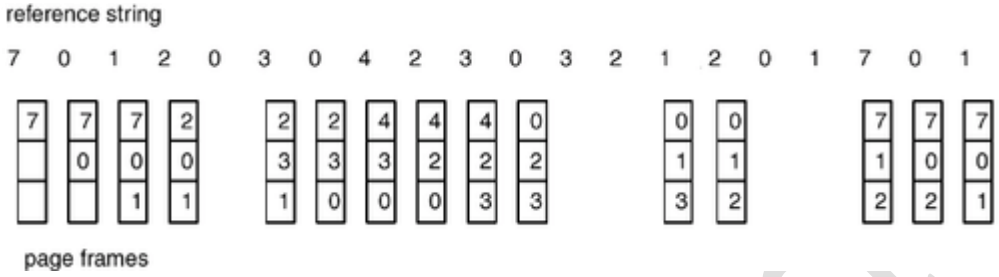
(Diagram of Page replacement)

The page replacement algorithms decide which memory pages to page out (swap out, write to disk) when a page of memory needs to be allocated. We evaluate an algorithm by running it on a particular string of memory references and computing the number of page faults. The string of memory references is called a reference string.

The different page replacement algorithms are described as follows:

First-In-First-Out (FIFO) Algorithm:

A FIFO replacement algorithm associates with each page the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen to swap out. We can create a FIFO queue to hold all pages in memory. We replace the page at the head of the queue. When a page is brought into memory, we insert it at the tail of the queue.



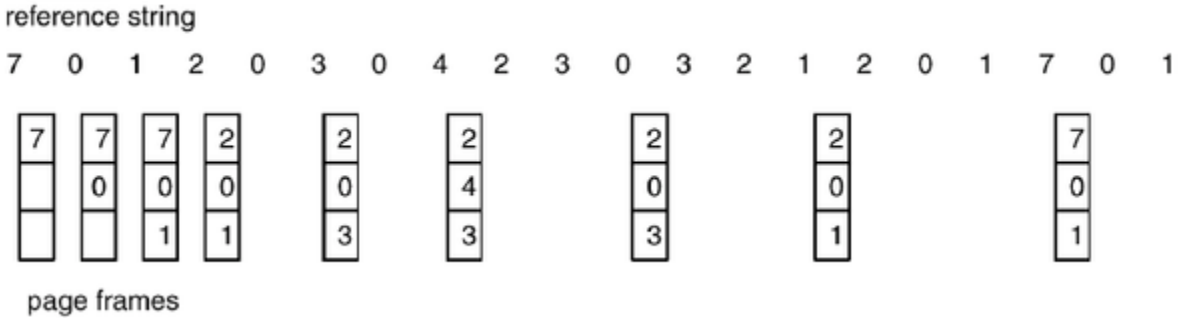
(FIFO page-replacement algorithm)

Note: For some page-replacement algorithms, the page fault rate may increase as the number of allocated frames increases. This most unexpected result is known as Belady's anomaly.

Optimal Page Replacement algorithm:

One result of the discovery of Belady's anomaly was the search for an optimal page replacement algorithm. An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms, and will never suffer from Belady's anomaly. Such an algorithm does exist and has been called OPT or MIN.

It is simply "Replace the page that will not be used for the longest period of time". Use of this page-replacement algorithm guarantees the lowest possible page fault rate for a fixed number of frames.

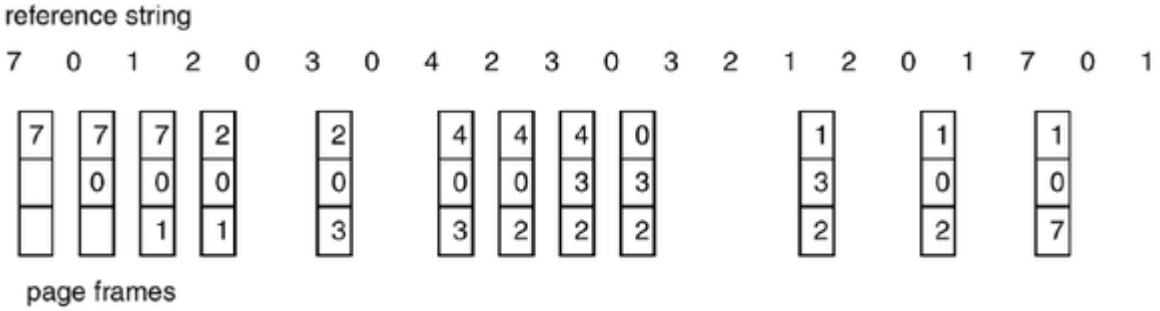


(Optimal Page Replacement Algorithm)

LRU Page Replacement Algorithm:

If we use the recent past as an approximation of the near future, then we will replace the page that has not been used for the longest period of time. This approach is the least-recently-used (LRU) algorithm.

LRU replacement associates with each page the time of that page's last use. When a page must be replaced, LRU chooses that page that has not been used for the longest period of time.



(LRU page-replacement algorithm)

FIFO (First in First out):

ALGORITHM

1. Start the process
2. Declare the size with respect to page length
3. Check the need of replacement from the page to memory
4. Check the need of replacement from old page to new page in memory
5. Forma queue to hold all pages
6. Insert the page require memory into the queue
7. Check for bad replacement and page fault
8. Get the number of processes to be inserted
9. Display the values
10. Stop the process

PROGRAM:

OUTPUT:

LRU (Least Recently Used)

ALGORITHM:

1. Start the process
2. Declare the size
3. Get the number of pages to be inserted
4. Get the value
5. Declare counter and stack
6. Select the least recently used page by counter value
7. Stack them according the selection.
8. Display the values
9. Stop the process

PROGRAM:

OUTPUT:

Optimal Page Replacement Algorithm:

Algorithm:

Here we select the page that will not be used for the longest period of time.

Step 1: Create a array

Step 2: When the page fault occurs replace page that will not be used for the longest Period of time

PROGRAM:

OUTPUT:

CONCLUSION: Thus we, have implement the C-program to Present the output for Page Replacement algorithm successfully.

Viva Voce Question

1. Which page replacement algorithm is used in Windows?

2. What is paging in operating system?

3. What is difference between segmentation and paging?

4. Which Page Replacement Algorithm is best? Why?

5. Define Page Fault?

Signature of Subject Teacher

EXPERIMENT NO. – 7

Aim: Write a C-program to implement memory management algorithm for memory management.

- First fit
- Best fit
- Worst fit

Theory:

Memory Management is one of the services provided by OS which is needed for Optimized memory usage of the available memory in a Computer System.

In addition to the responsibility of managing processes, the operating system must efficiently manage the primary memory of the computer. The part of the operating system which handles this responsibility is called the memory manager. Since every process must have some amount of primary memory in order to execute, the performance of the memory manager is crucial to the performance of the entire system.

Nutt explains: "The memory manager is responsible for allocating primary memory to processes and for assisting the programmer in loading and storing the contents of the primary memory. Managing the sharing of primary memory and minimizing memory access time are the basic goals of the memory manager."

The real challenge of efficiently managing memory is seen in the case of a system which has multiple processes running at the same time. Since primary memory can be space-multiplexed, the memory manager can allocate a portion of primary memory to each process for its own use. However, the memory manager must keep track of which processes are running in which memory locations, and it must also determine how to allocate and de allocate available memory when new processes are created and when old processes complete execution.

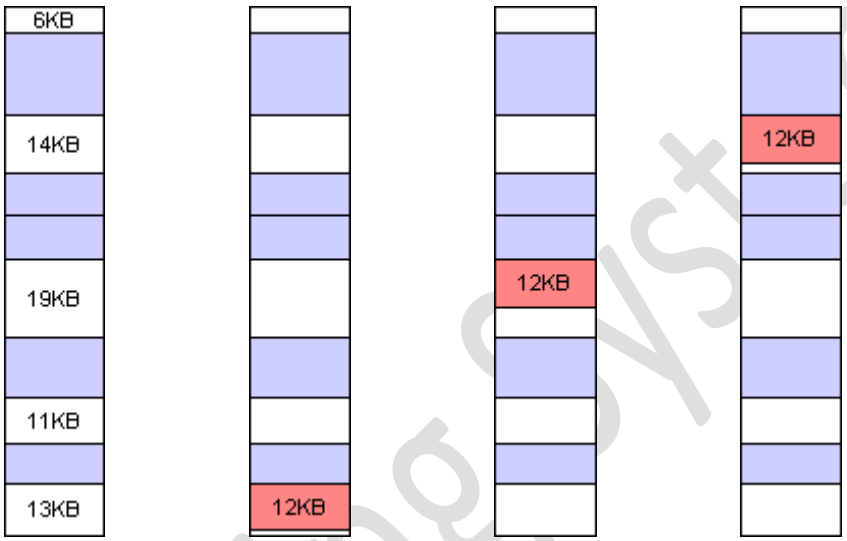
While various different strategies are used to allocate space to processes competing for memory, three of the most popular are **best fit, Worst fit, and first fit.** Each of these strategies is described below:

- **Best fit:** The allocator places a process in the smallest block of unallocated memory in which it will fit. For example, suppose a process requests 12KB of memory and the memory manager currently has a list of unallocated blocks of 6KB, 14KB, 19KB, 11KB, and 13KB blocks. The best-fit strategy will allocate 12KB of the 13KB block to the process.

- **Worst fit:** The memory manager places a process in the largest block of unallocated memory available. The idea is that this placement will create the largest hold after the allocations, thus increasing the possibility that, compared

to best fit, another process can use the remaining space. Using the same example as above, worst fit will allocate 12KB of the 19KB block to the process, leaving a 7KB block for future use.

- **First fit:** There may be many holes in the memory, so the operating system, to reduce the amount of time it spends analyzing the available spaces, begins at the start of primary memory and allocates memory from the first hole it encounters large enough to satisfy the request. Using the same example as above, first fit will allocate 12KB of the 14KB block to the process.



Notice in the diagram above that the Best fit and first fit strategies both leave a tiny segment of memory unallocated just beyond the new process. Since the amount of memory is small, it is not likely that any new processes can be loaded here. This condition of splitting primary memory into segments as the memory is allocated and de allocated is known as *fragmentation*. The Worst fit strategy attempts to reduce the problem of fragmentation by allocating the largest fragments to new processes. Thus, a larger amount of space will be left as seen in the diagram above.

First Fit Memory Management Scheme

Algorithm:

1. Get no. of Processes and no. of blocks.
2. After that get the size of each block and process requests.
3. Now allocate processes
   if(block size >= process size)
   //allocate the process
   else
   //move on to next block

4. Display the processes with the blocks that are allocated to a respective process.
5. Stop.

PROGRAM:

OUTPUT:

Best Fit Memory Management Scheme

Algorithm

1. Get no. of Processes and no. of blocks.
2. After that get the size of each block and process requests.
3. Then select the best memory block that can be allocated using the above definition.
4. Display the processes with the blocks that are allocated to a respective process.
5. Value of Fragmentation is optional to display to keep track of wasted memory.
6. Stop.

PROGRAM:

OUTPUT:

Worst Fit Memory Management Scheme

Algorithm:

1. The OS keeps a list of the available free memory spaces sorted from low to high address.
2. The OS always allocates memory from the beginning of the largest available slot.

PROGRAM:

OUTPUT:

CONCLUSION: Thus we, have implement the C-program to Present the output for Page Replacement algorithm successfully.

Viva Voce Question

1. Which of the memory allocation techniques first-fit, best-fit, worst-fit is efficient? Why?

2. What is compaction?

3. Which of the dynamic contiguous memory allocation strategies suffer with external fragmentation?

4. Differentiate between paging and segmentation memory allocation techniques?

Signature of Subject Teacher

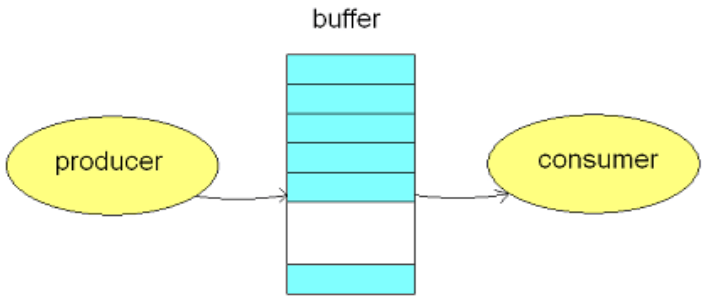
EXPERIMENT NO. – 8

Aim Write a C-program to present the output for Producer– Consumer problem concept.

THEORY:

The producer-consumer problem outlines the requirement for synchronization in systems where there are many processes that share a single resource. Let us consider two processes that share a fixed-sized buffer. Here one process produces and keeps the information in a buffer whereas the other process consumes it. Both the processes work concurrently. The data which is left unconsumed is kept in the buffer itself.

But there may be situations where the producer produce and tries to put an item into a full buffer. Similarly consumer may try to consume from an empty buffer. Thus to achieve synchronization, the producer must be blocked when the buffer is full and the consumer must be blocked when the buffer is empty. It is also called as bounded buffer problem.



ALGORITHM:

1. Start
2. Define the maximum buffer size.
3. Enter the number of producers and consumers.
4. The producer produces the job and put it in the buffer.
5. The consumer takes the job from the buffer.
6. If the buffer is full the producer goes to sleep.
7. If the buffer is empty then consumer goes to sleep.
8. Stop

PROGRAM:

OUTPUT:

CONCLUSION: Thus we, have implement the C-program to Present the output for Producer– Consumer problem successfully.

Viva Voce Question

1. What is the need for process synchronization? Define a semaphore?

2. How many semaphores are used in the producer and consumer problem?

3. Discuss the consequences of considering bounded and unbounded buffers in producer-consumer problem?

4. What is producer consumer problem in multithreading?

Signature of Subject Teacher

EXPERIMENT NO 9

Aim : Simulate Bankers algorithm for Deadlock Avoidance.

THEORY:

Banker's algorithm is a deadlock avoidance algorithm. It is named so because this algorithm is used in banking systems to determine whether a loan can be granted or not.

Consider there are n account holders in a bank and the sum of the money in all of their accounts is S. Every time a loan has to be granted by the bank, it subtracts the loan amount from the total money the bank has. Then it checks if that difference is greater than S. It is done because, only then, the bank would have enough money even if all the n account holders draw all their money at once.

Banker's algorithm works in a similar way in computers.

Whenever a new process is created, it must specify the maximum instances of each resource type that it needs, exactly.

Let us assume that there are n processes and m resource types. Some data structures that are used to implement the banker's algorithm are:

1. Available

It is an **array** of length m. It represents the number of available resources of each type. If Available[j] = k, then there are k instances available, of resource type R(j).

2. Max

It is an n x m matrix which represents the maximum number of instances of each resource that a process can request. If Max[i][j] = k, then the process P(i) can request at most k instances of resource type R(j).

3. Allocation

It is an n x m matrix which represents the number of resources of each type currently allocated to each process. If Allocation[i][j] = k, then process P(i) is currently allocated k instances of resource type R(j).

4. Need

It is an n x m matrix which indicates the remaining resource needs of each process. If Need[i][j] = k, then process P(i) may need k more instances of resource type R(j) to complete its task.

Need[i][j] = Max[i][j] - Allocation [i][j]

Resource Request Algorithm

This describes the behavior of the system when a process makes a resource request in the form of a request matrix. The steps are:

1. If number of requested instances of each resource is less than the need (which was declared previously by the process), go to step 2.
2. If number of requested instances of each resource type is less than the available resources of each type, go to step 3. If not, the process has to wait because sufficient resources are not available yet.
3. Now, assume that the resources have been allocated. Accordingly do,

   Available = Available - Requesti
   Allocation(i) = Allocation(i) + Request(i)
   Need(i) = Need(i) - Request(i)

This step is done because the system needs to assume that resources have been allocated. So there will be fewer resources available after allocation. The number of allocated instances will increase. The need of the resources by the process will reduce. That's what is represented by the above three operations.

After completing the above three steps, check if the system is in safe state by applying the safety algorithm. If it is in safe state, proceed to allocate the requested resources. Else, the process has to wait longer.

Safety Algorithm

1. Let Work and Finish be vectors of length **m** and **n**, respectively. Initially,
2. **Work = Available**
3. Finish[i] =false for i = 0, 1... n - 1.

This means, initially, no process has finished and the number of available resources is represented by the **Available** array.

4. Find an index **i** such that both
5. **Finish[i] ==false**
6. **Needi <= Work**

If there is no such i present, then proceed to step 4.

It means we need to find an unfinished process whose need can be satisfied by the available resources. If no such process exists, just go to step 4.

Perform the following:

7. **Work = Work + Allocation;**
8. **Finish[i] = true;**
9. Go to step 2.

When an unfinished process is found, then the resources are allocated and the process is marked finished. And then, the loop is repeated to check the same for all other processes.

10. If Finish[i] == true for all i, then the system is in a safe state.

That means if all processes are finished, then the system is in safe state.

ALGORITHM:

1. Start the program.
2. Get the values of resources and processes.
3. Get the avail value.
4. After allocation find the need value.
5. Check whether it's possible to allocate.
6. If it is possible then the system is in safe state.
7. Else system is not in safety state.
8. If the new request comes then check that the system is in safe state.
9. If not then we allow the request.
10. Stop the program.

PROGRAM:

OUTPUT:

CONCLUSION: Thus we, have implement the C-program for Banker's Algorithm for Deadlock Avoidance successfully.

Viva Voce Question

1. What is for Deadlock? What are the conditions to be satisfied for the deadlock to occur?

2. How can be the resource allocation graph used to identify a deadlock situation?

3. How is Banker's algorithm useful over resource allocation graph technique?

4. Differentiate between deadlock avoidance and deadlock prevention?

Signature of Subject Teacher

EXPERIMENT NO 10

Aim: Write a C-program to present the Output of following Disk Arm Scheduling algorithm. (Any Three)

- FCFS
- SSTF
- SCAN
- C-SCAN
- LOOK
- C-LOOK

THEORY:

PROGRAM:

OUTPUT:

CONCLUSION: Thus, we have written a c-program to present the output of following disk arm scheduling algorithm successfully.