Degree Engineering

# A Laboratory Manual for

**Operating Systems**

**(3140702)**

**[ B.E. (Computer Engineering) : Semester - 4 ]**

| Enrolment No |  |
| --- | --- |
| Name |  |
| Branch |  |
| Academic Term |  |
| Institute Name |  |



**Directorate of Technical Education, Gandhinagar, Gujarat**

# L.D.College of Engineering, Ahmedabad Department of Computer Engineering

**CERTIFICATE**

*This is to certify that Mr./Ms.*

*Enrollment No. of B.E. Semester -* ***IV*** *from* ***Computer Engineering Department*** *of this Institute (GTU Code: 028) has satisfactorily completed the Practical / Assignment work for the subject* ***Operating System (3140702)*** *for the academic year 2023-2024.*

**Place:**

**Date:**

**Signature of Course Faculty Head of the Department**

# Preface

Main motto of any laboratory/practical/field work is for enhancing required skills as well as creating ability amongst students to solve real time problem by developing relevant competencies in psychomotor domain. By keeping in view, GTU has designed competency focused outcome-based curriculum for engineering degree programs where sufficient weightage is given to practical work. It shows importance of enhancement of skills amongst the students and it pays attention to utilize every second of time allotted for practical amongst students, instructors and faculty members to achieve relevant outcomes by performing the experiments rather than having merely study type experiments. It is must for effective implementation of competency focused outcome-based curriculum that every practical is keenly designed to serve as a tool to develop and enhance relevant competency required by the various industry among every student. These psychomotor skills are very difficult to develop through traditional chalk and board content delivery method in the classroom. Accordingly, this lab manual is designed to focus on the industry defined relevant outcomes, rather than old practice of conducting practical to prove concept and theory.

By using this lab manual students can go through the relevant theory and procedure in advance before the actual performance which creates an interest and students can have basic idea prior to performance. This in turn enhances pre-determined outcomes amongst students. Each experiment in this manual begins with competency, industry relevant skills, course outcomes as well as practical outcomes (objectives). The students will also achieve safety and necessary precautions to be taken while performing practical.

This manual also provides guidelines to faculty members to facilitate student centric lab activities through each experiment by arranging and managing necessary resources in order that the students follow the procedures with required safety and necessary precautions to achieve the outcomes. It also gives an idea that how students will be assessed by providing rubrics.

Operating System is one of the core courses in Computer Engineering discipline. It includes basic working and application of Operating System. Process and Thread management with different Process scheduling Algorithms like FCFS, SJF, RR, Priority, etc. Concurrency control mechanisms of processes. To understand Inter Process Communication: Race Conditions, Critical Section, And Mutual Exclusion concepts are needed. Deadlock and its solutions for uninterrupted execution of processes. Memory management in Operating system. Input output and Disk scheduling algorithms. Virtualization concepts, Development of basic code of operating system using Shell Scripts.

Utmost care has been taken while preparing this lab manual however always there is chances of improvement. Therefore, we welcome constructive suggestions for improvement and removal of errors if any.

**DTE’s Vision**

* To provide globally competitive technical education
* Remove geographical imbalances and inconsistencies
* Develop student friendly resources with a special focus on girls’ education and support to weaker sections
* Develop programs relevant to industry and create a vibrant pool of technical professionals

**Institute’s Vision**

* + To contribute for sustainable development of nation through achieving excellence in technical education and research while facilitating transformation of students into responsible citizens and competent professionals.

**Institute’s Mission**

* To impart affordable and quality education in order to meet the needs of industries and achieve excellence in teaching-learning process.
* To create a conducive research ambience that drives innovation and nurtures scholars and outstanding professionals.
* To collaborate with other academic & research institutes as well as industries in order to strengthen education and multidisciplinary research.
* To promote equitable and harmonious growth of students, academicians, staff, society and industries, thereby becoming a center of excellence in technical education.
* To practice and encourage high standards of professional ethics, transparency and accountability.

**Department’s Vision**

* To achieve academic excellence in Computer Engineering by providing value based education.

**Department’s Mission**

* To produce graduates according to the needs of industry, government, society and scientific community. To develop partnership with industries, research and development organizations and government sectors for continuous improvement of faculties and students.
* To motivate students for participating in reputed conferences, workshops, seminars and technical events to make them technocrats and entrepreneurs.
* To enhance the ability of students to address the real life issues by applying technical expertise, human values and professional ethics.
* To inculcate habit of using free and open source software, latest technology and soft skills so that they become competent professionals.
* To encourage faculty members to upgrade their skills and qualification through training and higher studies at reputed universities.

# Programme Outcomes (POs)

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

**Program Specific Outcomes (PSOs)**

* + Graduates will be able to explore and propose effective solutions to the problems in the area of Computer Engineering as per the needs of society and industry.
  + Graduates will be able to apply standard practice and strategies to develop quality software products using modern techniques, programming skills, tools & an open ended programming environment and work in a team.
  + Graduates will manifest the skills of continuous learning in the fast changing field of Computer Engineering.

**Program Educational Objectives (PEOs)**

* + Provide computing solutions of complex problems as per business and societal needs.
  + Procure requisite skills to pursue entrepreneurship, research and development, higher studies and imbibe high degree of professionalism in the fields of computing.
  + Embrace life-long learning and remain continuously employable.
  + Work and excel in a highly competence supportive, multicultural and professional environment which is abiding to the legal and ethical responsibilities.

# Practical – Course Outcome matrix

| **Course Outcomes (COs)** | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CO\_3140702.1** | | Analyze the structure of OS and basic architectural components involved  in OS design | | | | | |
| **CO\_3140702.2** | | Compare and contrast various CPU scheduling algorithms. | | | | | |
| **CO\_3140702.3** | | Evaluate the requirements for the process synchronization and co-  ordination in contemporary operating system. | | | | | |
| **CO\_3140702.4** | | Analyze various algorithms for memory management, I/O management  and security aspects of operating system. | | | | | |
| **CO\_3140702.5** | | Write shell scripts in Unix/Linux O.S and write simple programs using  kernel system calls. Also understand virtualization concept. | | | | | |
|  | | | | | | | |
| **Sr.**  **No.** | **Practical Outcome/Title of experiment** | | **C O**  **1** | **C O**  **2** | **C O**  **3** | **C O**  **4** | **C O**  **5** |
| 1 | Study of Linux/Windows system Architecture ,  Installation and MS DOS Commands. | | **√** |  |  |  | **√** |
| 2 | Study and execute Basic and directory manipulation  commands of LINUX/UNIX. | |  |  |  |  | **√** |
| 3 | Study and execute Basic File manipulation  commands. | |  |  |  |  | **√** |
| 4 | Study and Execute Advance Filter Commands. | |  |  |  |  | **√** |
| 5 | Write a shell script program using Loop/ control  structure. | |  |  |  |  | **√** |
| 6 | Loop/ control structure using shell script(Using  while loop) | |  |  |  |  | **√** |
| 7 | Command execution via Shell script. | |  |  |  |  | **√** |
| 8 | Process Scheduling Algorithm and Comparison. | |  | **√** |  |  |  |
| 9 | Process creation and Thread Scheduling | |  |  | **√** |  |  |
| 10 | Page replacement and Disk Scheduling algorithm | |  |  |  | **√** |  |

## Industry Relevant Skills

Operating system specialists are professionals who design, install, configure, maintain, and troubleshoot various operating systems, one need to have a solid foundation of technical skills and knowledge, as well as soft skills to work on Operating system.

## Guidelines for Faculty members

1. Teacher should provide the guideline with demonstration of practical to the students with all features.
2. Teacher shall explain basic concepts/theory related to the experiment to the students before starting of each practical
3. Involve all the students in performance of each experiment.
4. Teacher is expected to share the skills and competencies to be developed in the students and ensure that the respective skills and competencies are developed in the students after the completion of the experimentation.
5. Teachers should give opportunity to students for hands-on experience after the demonstration.
6. Teacher may provide additional knowledge and skills to the students even though not covered in the manual but are expected from the students by concerned industry.
7. Give practical assignment and assess the performance of students based on task assigned to check whether it is as per the instructions or not.
8. Teacher is expected to refer complete curriculum of the course and follow the guidelines for implementation.

## Instructions for Students

1. Students are expected to carefully listen to all the theory classes delivered by the faculty members and understand the COs, content of the course, teaching and examination scheme, skill set to be developed etc.
2. Students will have to perform experiments on computer system on which UNIX/Linux is installed to execute programs of Operating System.
3. Students should develop programs and execute all the programs using UNIX/Linux OS. Students have to show output of each program in their practical file.
4. Students are instructed to submit practical list as per given sample list shown on next page.
5. Student should develop a habit of submitting the experimentation work as per the schedule and she/he should be well prepared for the same.

## Common Safety Instructions

Students are expected to

1. switch on the PC carefully (not to use wet hands)
2. shutdown the PC properly at the end of your Lab
3. carefully handle the peripherals (Mouse, Keyboard, Network cable etc)
4. use Laptop in lab after getting permission from Teacher
5. carefully handle all lab resources

# Index

**(Progressive Assessment Sheet)**

| **Sr.**  **No.** | **Objective(s) of Experiment** | **Pg. No.** | **Date**  **of performance** | **Date of ubmission** | **Assessment Marks** | **Sign. of Teacher with date** | **Remarks** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1. | Study of Linux/Windows system Architecture , Installation and MS DOS Commands. | | | | | | |
|  | * 1. Study of LINUX/UNIX Architecture AND Installation of Ubuntu using Virtual Box.   2. Give the advantages of UNIX over Windows.   3. Execute the DOS Commands |  |  |  |  |  |  |
| 2. | Study and execute Basic and directory manipulation commands of LINUX/UNIX. | | | | | | |
|  | * 1. Study of Unix Shell and Environment Variables.   2. man, cal, date, echo, bc, who, uname   3. Using commands : pwd, mkdir, cd, rmdir   ,ls generate given tree. |  |  |  |  |  |  |
| 3. | Study and execute Basic File manipulation commands. | | | | | | |
|  | Explore following commands: 1. cat 2.wc 3. cp 4. mv 5.rm 6. File 7. cmp 8. comm 9.  diff 10. chmod 11. sort |  |  |  |  |  |  |
| 4. | Study and Execute Advance Filter Commands. | | | | | | |
|  | Explore following commands: 1.head 2. tail  3. paste 4. cut(-f) 5. cut(-c) 6. grep |  |  |  |  |  |  |
| 5. | Write a shell script program using Loop/ control structure. | | | | | | |
|  | * 1. Write a shell script to find factorial of given number n.   2. Write a shell script which will generate first n Fibonacci numbers like: 1, 1, 2, 3,   5, 13, …" |  |  |  |  |  |  |
| 6. | Write a shell script program using Loop/ control structure.(Using while loop) | | | | | | |
|  | * 1. Write a shell script to read n numbers as   command arguments and sort them in descending order. |  |  |  |  |  |  |

|  | 6.2 Write a shell script to generate mark sheet of a student. Take 3 subjects, calculate and display total marks, percentage and Class obtained by the  student." |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7. | Command execution via Shell script. | | | | | | |
|  | * 1. Write a shell script to display all executable files, directories and zero sized files from current directory   2. Write a menu driven shell script which will print the following menu and execute the given task.   MENU   * + - Display calendar of current month     - Display today’s date and time     - Display usernames those are currently logged in the system     - Display your name at given x, y position     - Display your terminal number   Exit |  |  |  |  |  |  |
| 8. | Process Scheduling Algorithm and Comparison. | | | | | | |
|  | Write a C Program to Implement Following CPU Scheduling algorithms.   * FCFS * Round Robin |  |  |  |  |  |  |
| 9. | Process creation and Thread Scheduling | | | | | | |
|  | * 1. Implement Producer consumer problem using thread using C/JAVA programming Language.   2. Create new thread using fork() system   call using C programming Language. |  |  |  |  |  |  |
| 10. | Page replacement and Disk Scheduling algorithm | | | | | | |
|  | * 1. Implement FIFO Page replacement Algorithm using C/Java.   2. Implement C-SCAN Disk Scheduling   Algorithm using C/Java. |  |  |  |  |  |  |
| **Total** | | | | |  |  |  |

# Experiment No – 1

**AIM : S**tudy of Linux/Windows system Architecture , Installation and MS DOS Commands.

* 1. Study of LINUX/UNIX Architecture AND Installation of Ubuntu using Virtual Box.
  2. Give the advantages of UNIX over Windows.
  3. Execute the DOS Commands

**Date: *16/5/2023***

**Competency and Practical Skills:** Logic building and programming

**Relevant CO:** CO1, CO5

**Objectives:**

* + 1. To analyze various Operating Systems structure
    2. To use different Commands.
    3. To differentiate working of types of OS. **Equipment/Instruments:** Computer System with Winows/Linux **Safety and necessary Precautions:**

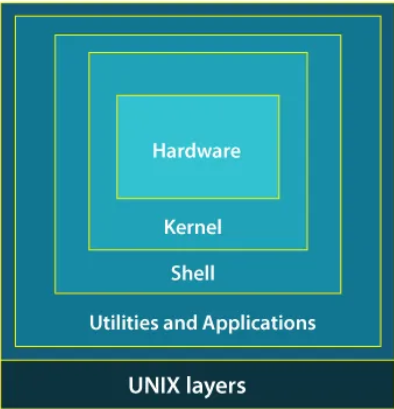
✔ Operate computer system carefully and responsibly.

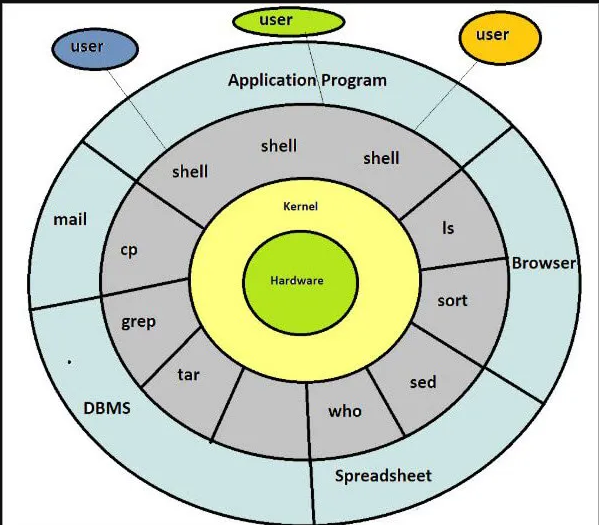
✔ Use required lab resources cautiously

**Theory:**

Computer system can be divided roughly into four components: the hardware, the operating system, the application programs, and the users).The hardware like the central processing unit (CPU), the memory, and the input/output (I/O) devices provides the basic computing resources for the system. The application programs such as word processors, spreadsheets, compilers, and Web browsers—define the ways in which these resources are used to solve users’ computing problems. The operating system controls the hardware and coordinates its use among the various application programs for the various users. In short, operating system provides the means for proper use of these resources in the operation of the computer system. It also provides an environment within which other programs can do useful work.

* 1. **Study of LINUX/UNIX Architecture AND Installation of Ubuntu using Virtual Box.**
     + **Draw LINUX/UNIX Architecture Explain significance of each component.**





**Hardware:** This is the physical layer consisting of the computer's components like CPU, RAM, hard drives, and peripherals. It provides the raw processing power and storage for the system.

**Kernel:** The kernel is the heart of the operating system. It acts as an intermediary between the hardware and the upper layers. Key functionalities of the kernel include:

* **Process Management:** Creates and manages processes (running programs), allocating CPU time and memory resources.
* **Memory Management:** Handles memory allocation and deallocation for applications.
* **File Management:** Oversees access, creation, deletion, and organization of files on the storage devices.
* **Device Management:** Controls communication with hardware devices like printers, network cards, and USB devices through device drivers.
* **Security:** Enforces access controls and system protection.

**System Libraries:** These are collections of pre-written code that provide basic building blocks for application development. They offer functionalities like file I/O, networking, memory management, and mathematical operations. This saves developers from writing repetitive code and ensures consistency across applications.

**Shell:** The shell acts as a user interface, typically a command-line interface (CLI) but can also be a graphical shell. It allows users to interact with the kernel by issuing commands. The shell interprets these commands and translates them into actions for the kernel to perform.

**Applications:** This is the topmost layer consisting of user programs that leverage the underlying layers to perform specific tasks. Examples include web browsers, office suites, games, and development tools. Applications rely on system libraries and kernel services to function.

**Significance of Layered Architecture:**

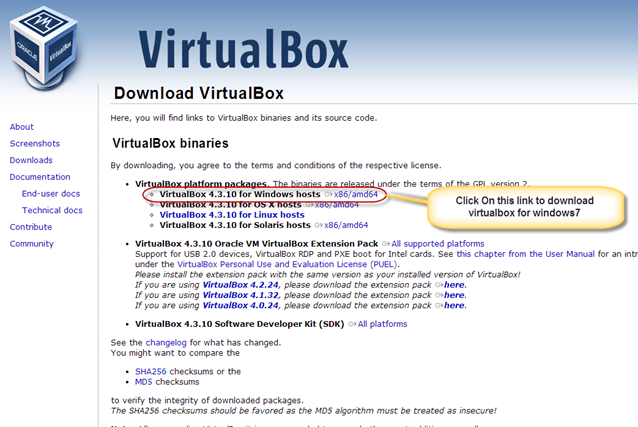
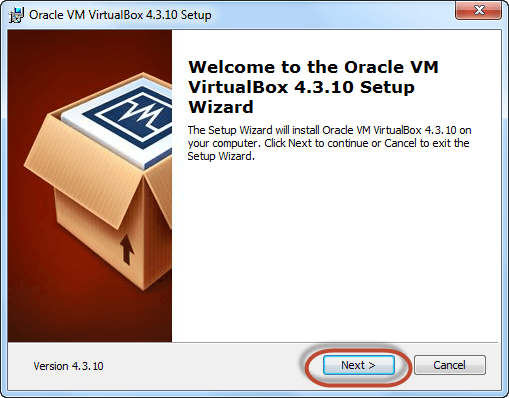
* **Modular Design:** The layered architecture promotes modularity, where changes in one layer can be made without affecting others. This simplifies development and maintenance.
* **Scalability:** The modular design allows for easier addition of new functionalities or hardware components without major architectural changes.
* **Security:** The separation of layers enhances security. The kernel, the most privileged layer, is protected from direct user interaction, reducing the attack surface.
  + - **Write a step to install Ubuntu On Virtual Box / Computer System. (Faculty Needs to Explain How VirtualBox/Virtual machine is helpful )**

**Step-1 Installing Linux using Virtual Machine**

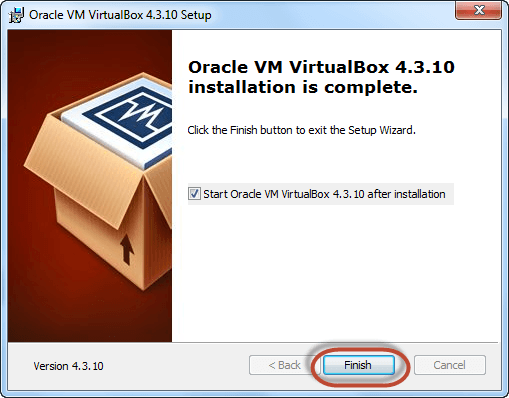
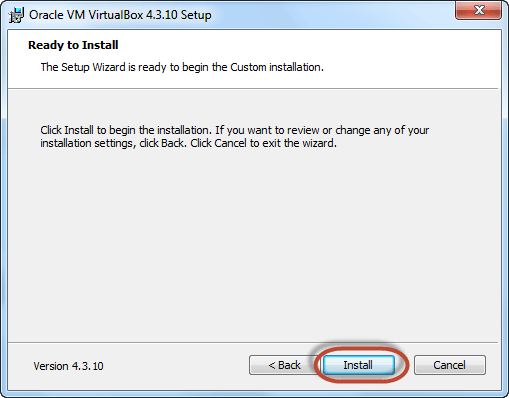
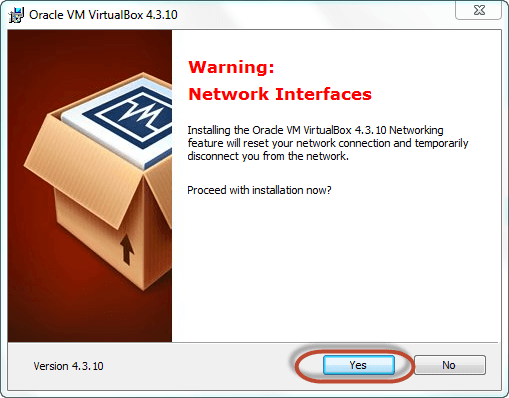
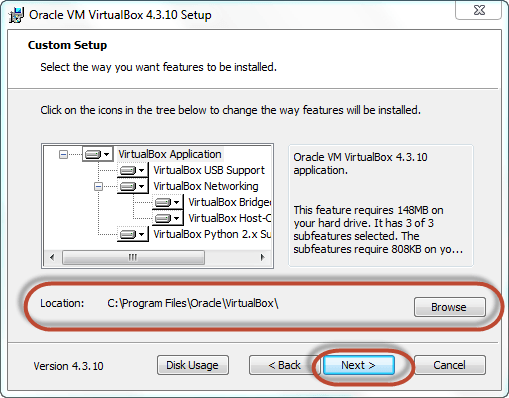
This is a popular method to install a Linux operating system. The virtual installation offers you the freedom of running Linux on an existing OS already installed on your computer. This means if you have Windows running, then you can just run Linux with a click of a button.

**Step-2 Download and Install Virtual Box**

Download Virtual box depending on your processor and OS, select the appropriate package. In our case, we have selected Windows with AMD. Once Download Complete open the source and run it and install VirtualBox on your System.Once it installed Open Virtualbox for further process.(snapshots are given below)

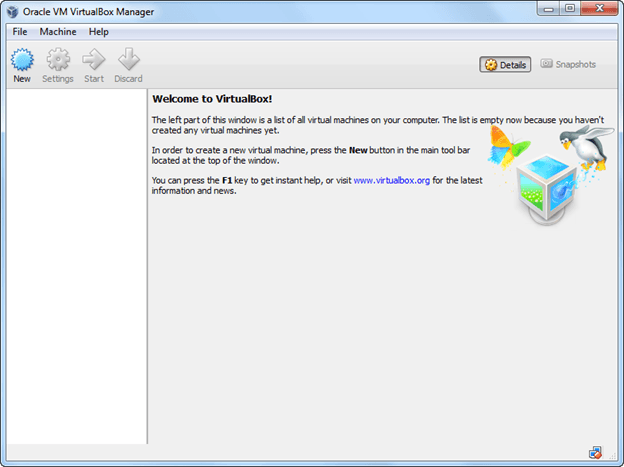
 

**Fig.1.1 Downloading Installing Virtual Box**



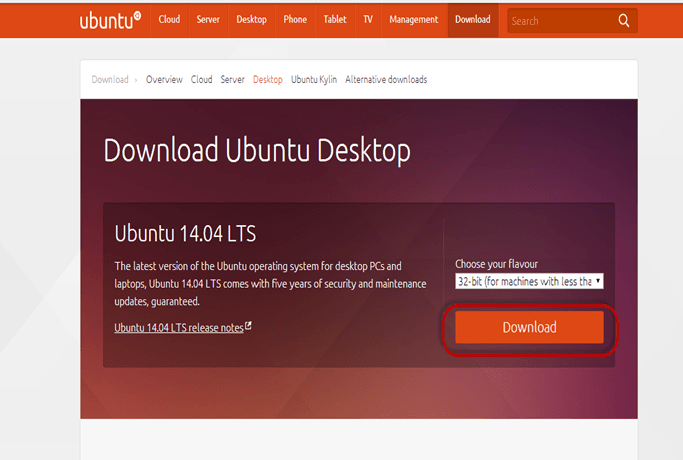
**Fig.1.2 Steps to follow while installing Virtual Box**

The virtual box dashboard looks like this-



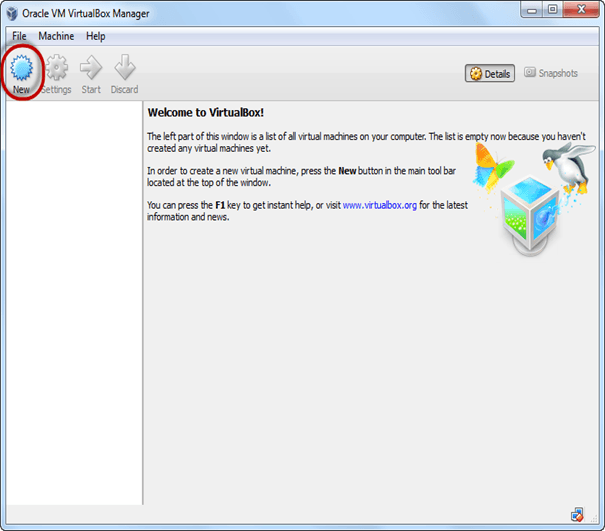
**Fig.1.3 Virtual Box dashboard after installation**

**Step-3** Download Ubuntu AND INSTALL.For download **You can select 32/64-bit versions as per your choice.**



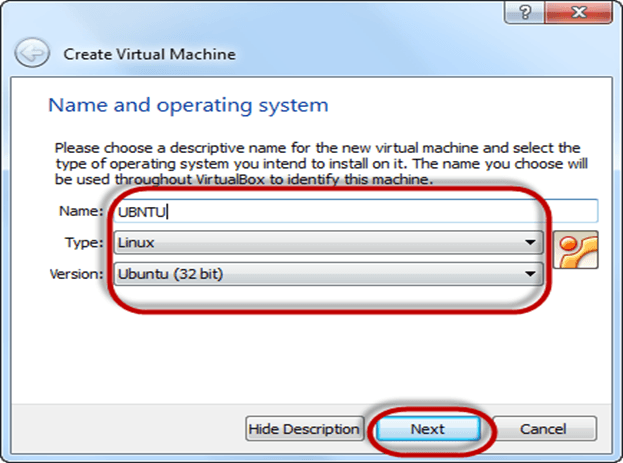
**Fig.1.4 Downloading Ubuntu**

**Step-4** Create a Machine in Virtual Box. Open Virtual box and click on new button



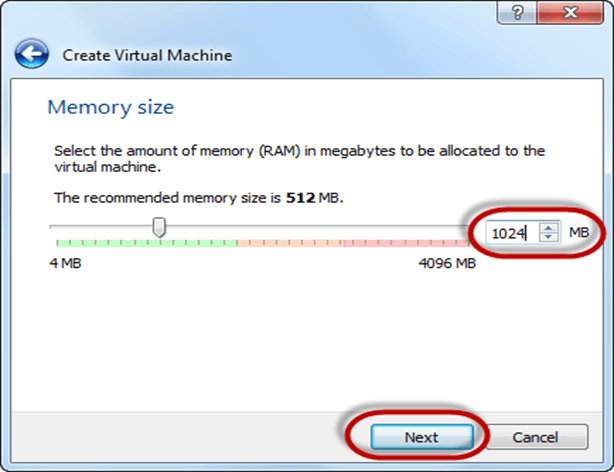
**Fig.1.5 instant starting of virtual box for ubuntu installation virtual Box**

**Step-5** In next window**,** give the name of your OS which you are installing in virtual box. And select OS like [Linux](https://www.guru99.com/unix-linux-tutorial.html) and version as Ubuntu 32 bit. And click on next



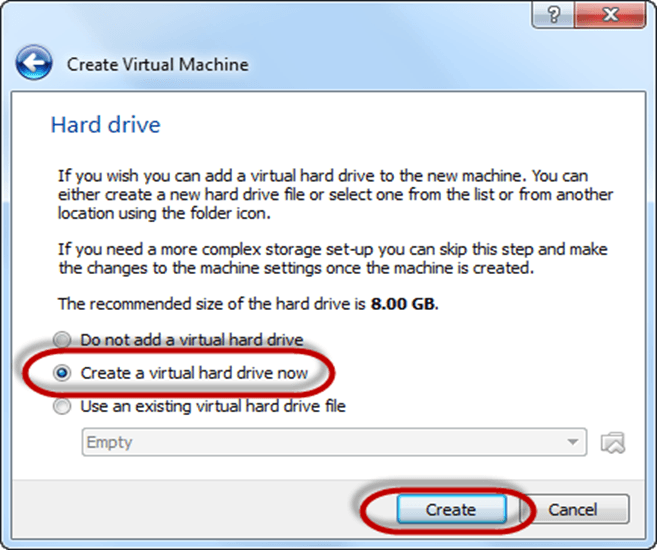
**Fig.1.6 providing basic system details for ubuntu installation**

**Step-6** Now Allocate RAM Size To your Virtual OS. It recommended keeping 1024MB (1 GB) RAM to run Ubuntu better. And click on next.



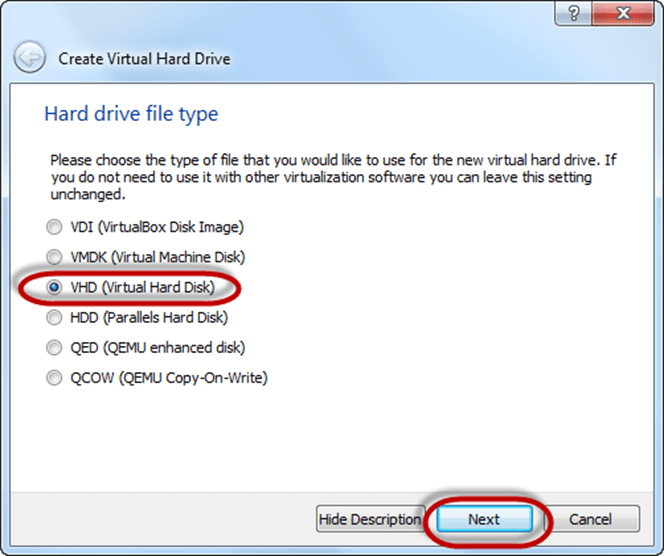
**Fig.1.7 providing memory requirement for ubuntu installation**

**Step-7** Now To run OS in virtual box we have to create virtual hard disk, click on create a virtual hard drive now and click on create button. The virtual hard disk is where the OS installation files and data/applications you create/install in this Ubuntu machine will reside



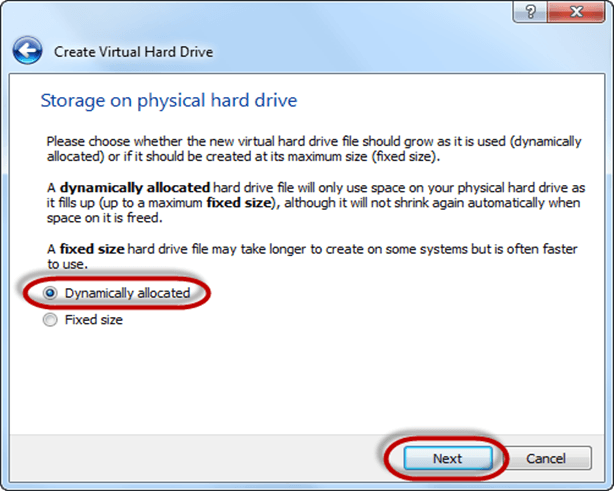
**Fig.1.8 creating virtual disk drive for ubuntu installation**

**Step-8** select VHD (virtual hard disk) option and click on next.



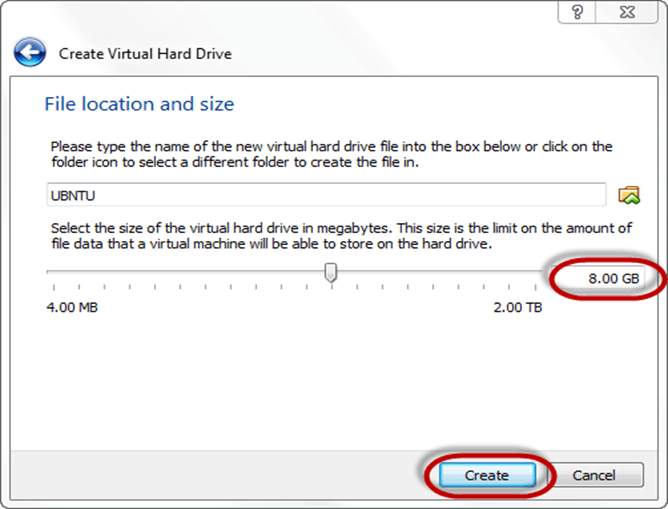
**Fig.1.9 creating virtual disk drive for ubuntu installation**

**Step-9** Click on dynamic allocated and click on next. This means that the size of the disk will increase dynamically as per requirement.



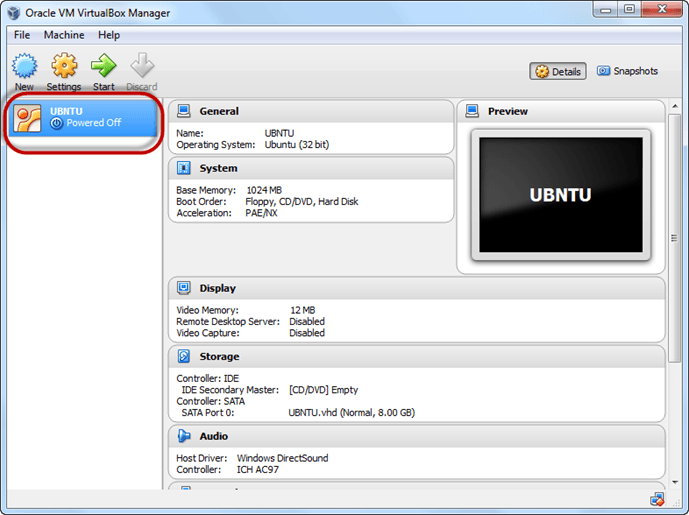
**Fig.1.10 creating virtual disk drive for ubuntu installation**

**Step-10** Allocate memory to your virtual hard drive .8GB recommended. Click on create button.



**Fig.1.11 Memory allocation for ubuntu installation**

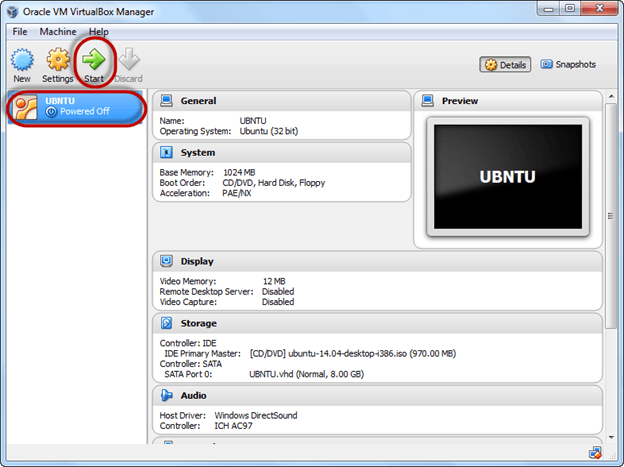
**Step-11** Now you can see the machine name in left panel



**Fig.1.12 machine configuration ready for ubuntu installation**

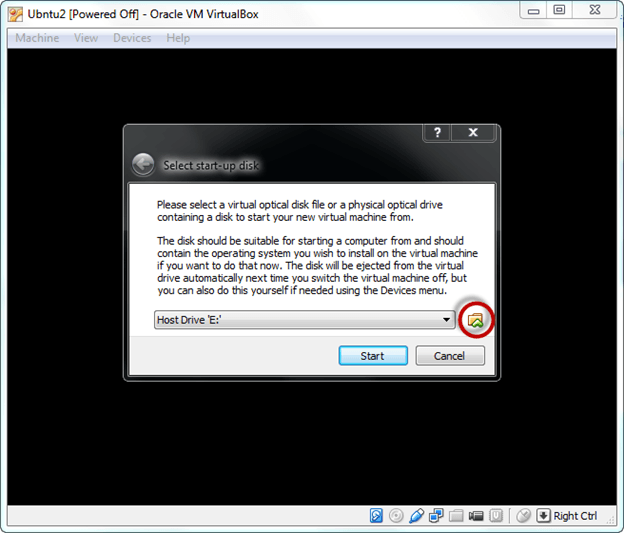
So a Machine (PC) with 8GB Hardisk, 1GB RAM is ready.

**Step 12** Select the Machine and Click on Start



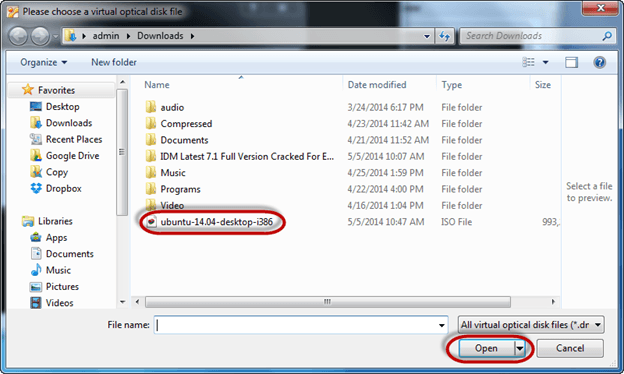
**Fig.1.13 start installing ubuntu**

**Step 13** Select the Folder Option



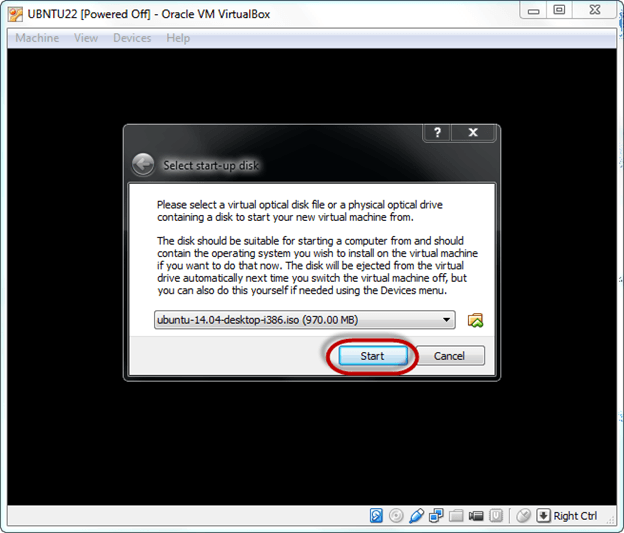
**Fig.1.14 selecting source for ubuntu installation**

**Step 14** Select the Ubuntu iso file



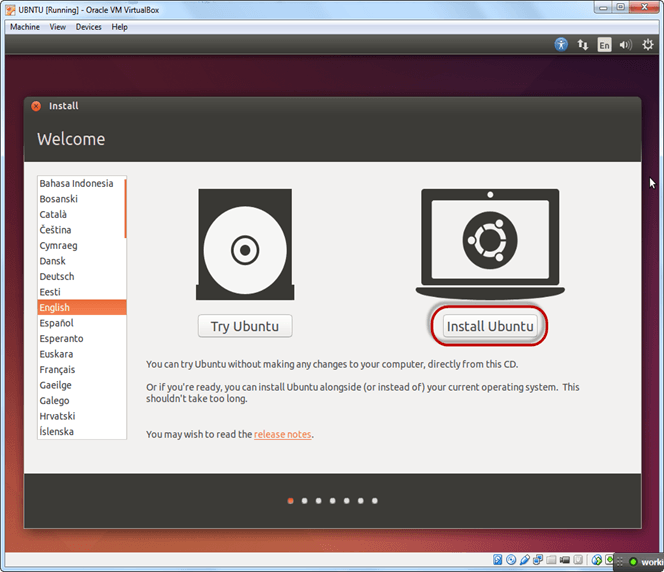
**Fig.1.15 selecting source for ubuntu installation**

**Step 15** Click Start



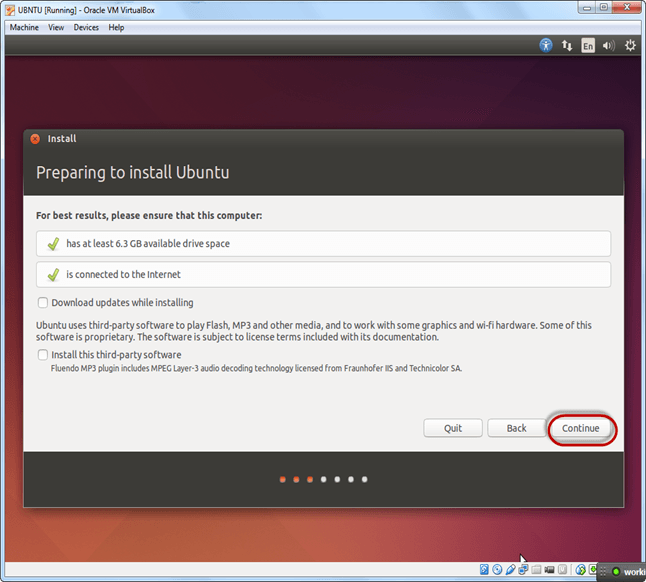
**Fig.1.16 selecting source for ubuntu installation**

**Step-16** You have an option to Run Ubuntu WITHOUT installing. In this tutorial will install Ubuntu



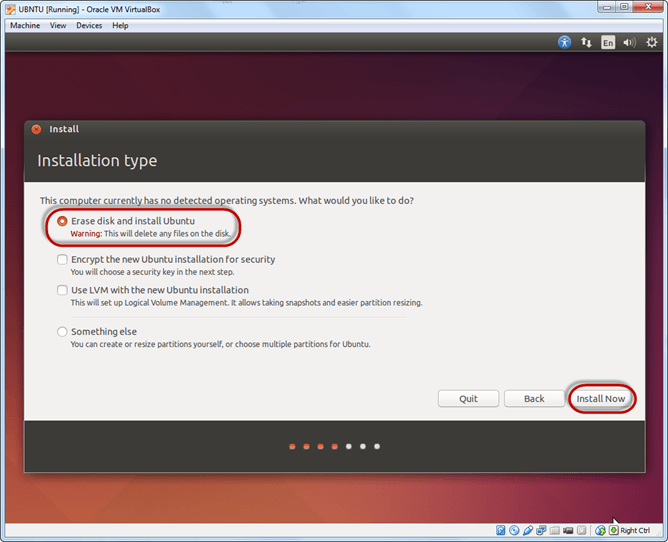
**Fig.1.17 start installation**

**Step-17** Click continue.



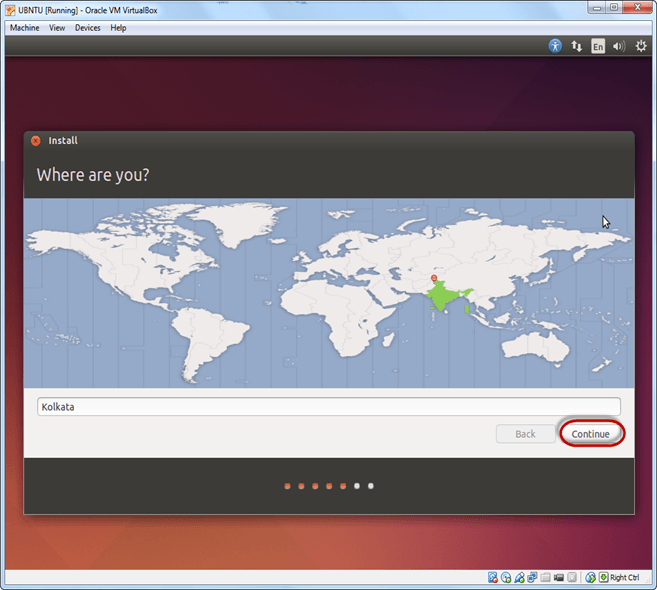
**Fig.1.18 Installation started**

**Step-18** Select option to erase the disk and install Ubuntu and click on install now. This option installs Ubuntu into our virtual hard drive which is we made earlier. It will not harm your PC or Windows installation



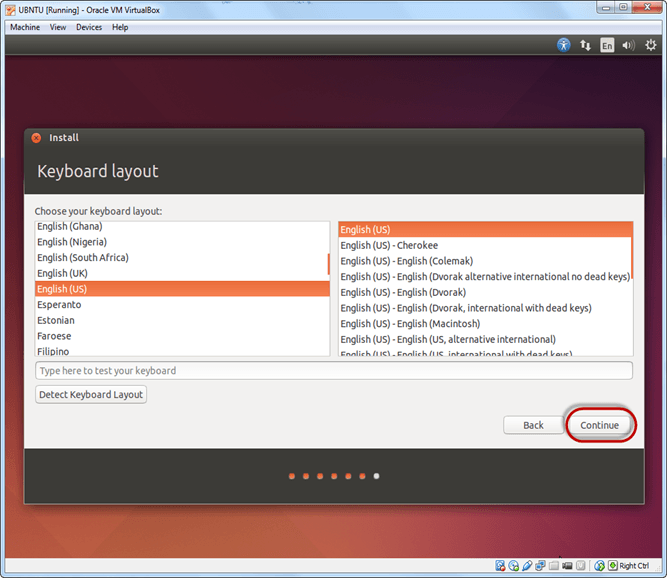
**Fig.1.19 Installation started**

**Step-19** Select your location for setting up time zone, and click on continue



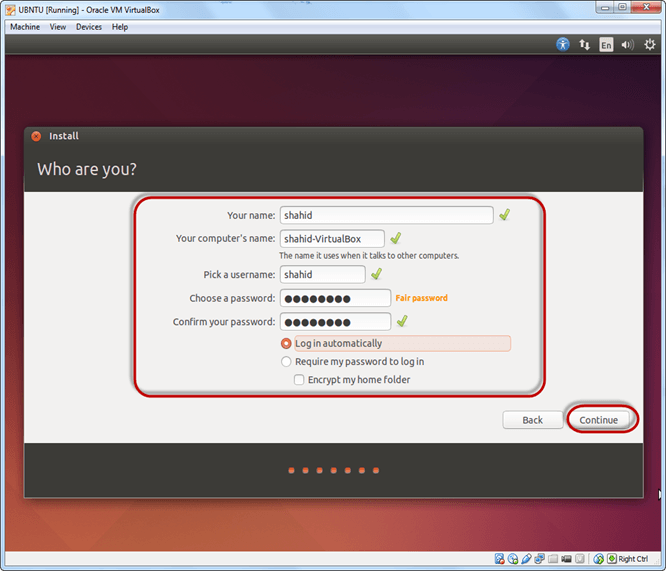
**Fig.1.20 setting up timezone**

**Step-20** Select your keyboard layout, by default English (US) is selected but if you want to change then, you can select in the list. And click on continue



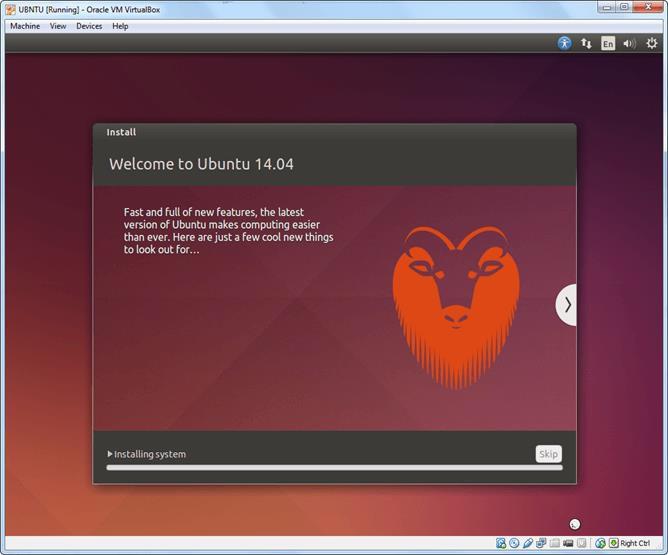
**Fig.1.21 setting up preferred Language**

**Step-21** Select your username and password for your Ubuntu admin account. This information has been needed for installing any software package into Ubuntu and also for login to your OS. Fill up your details and tick on login automatically to ignore login attempt and click on continue



**Fig.1.22 setting up User credentials**

**Step-22** Installation process starts. May take up to 30 minutes. Please wait until installation process completes. After finishing the installation, you will see Ubuntu Desktop.



**Fig.1.23 Installation in Progess**

* 1. **Give the advantage of Linux over Windows OS.**

**Open Sourc**e: Linux is open-source, meaning its source code is freely available for anyone to inspect, modify, and distribute. This fosters a large community of developers who contribute to its improvement and create custom distributions.

**Security**: Due to its open nature and focus on user permissions, Linux is generally considered more secure than Windows. Malware targeting Linux is less common, and the system architecture itself discourages unauthorized access.

**Cost**: Most Linux distributions are free to use and install, eliminating the licensing cost associated with Windows.

**Stability**: Linux systems are known for their stability and uptime. They can run for extended periods without requiring reboots, making them ideal for servers and mission-critical applications.

**Lightweight**: Linux can run efficiently on older hardware with lower resource requirements compared to Windows. This makes it suitable for breathing new life into older machines.

**Customization**: Linux offers extensive customization options. Users can personalize their desktop environment, choose from a variety of distributions tailored for specific needs, and modify the system to their liking.

**Privacy**: Many Linux users value the focus on privacy in the operating system. There's generally less data collection by default compared to Windows.

**Flexibility**: With a vast array of distributions available (like Ubuntu, Mint, Fedora), Linux caters to a wide range of users from beginners to power users and professionals with specific needs.

**Software** **Management**: Package managers in Linux provide a centralized and convenient way to install, update, and remove software.

**Community** **Support**: The Linux community is known for its collaborative and helpful nature. Numerous online forums and communities offer assistance and troubleshooting for Linux users.

* 1. **Executes Following DOS Commands.**

| 1. ATTRIB 2. CD 3. CHDIR 4. CLS 5. COPY | 1. DATE 2. DEL 3. DIR 4. ECHO 5. EXIT | 1. FC 2. FIND 3. FINDSTR 4. MKDIR 5. MOVE | 1. RENAME 2. REPLACE 3. RMDIR 4. TREE 5. SORT | 1. TITLE 2. PRINT 3. HELP 4. TIME 5. VER |
| --- | --- | --- | --- | --- |

**1. ATTRIB:**

* **Purpose:** Modifies file attributes like read-only, archive, hidden, and system.
* **Options:**
  + +R: Sets read-only attribute (file can be viewed but not modified).
  + -R: Clears read-only attribute.
  + +A: Sets archive attribute (marks file for backup).
  + -A: Clears archive attribute.
  + +H: Sets hidden attribute (hides the file).
  + -H: Clears hidden attribute.
  + +S: Sets system attribute (use with caution, marks file as critical).
  + -S: Clears system attribute.
  + /S: Applies attribute changes to subdirectories as well.

**Example:** attrib +R report.txt // Makes "report.txt" read-only

**2. CD / CHDIR:**

* **Purpose:** Changes the current directory.
* **Options:** (No options, specify the target directory path after the command)

**Example:** cd c:\Users\<username>\Documents // Navigates to "Documents" folder

**3. CLS:**

* **Purpose:** Clears the Command Prompt screen.
* **Options:** (No options)

**Example:** cls // Clears the screen

**4. COPY:**

* **Purpose:** Copies files from one location to another.
* **Options:**
  + /Y: Overwrites existing files without confirmation.
  + /Z: Copies files in compressed format (if supported for specific file types).
  + /A: Copies files with the archive attribute set.

**Example:** copy important.txt backup\ // Copies "important.txt" to "backup" folder

**5. DATE:**

* **Purpose:** Displays the current system date.
* **Options:** (No options, but custom formats are available)
  + /T: Displays both date and time.

**Example:** date /T // Shows both date and time

**6. DEL:**

* **Purpose:** Deletes files.
* **Options:**
  + /P: Prompts for confirmation before deleting each file.
  + /F: Forces deletion of read-only files (use with caution).
  + /S: Deletes files in the current directory and all subdirectories.

**Example:** del temp\*.txt // Deletes all ".txt" files starting with "temp" (prompts for confirmation)

**7. DIR:**

* **Purpose:** Lists the contents of a directory (files and folders).
* **Options:**
  + /W: Displays directory listing in wide format.
  + /A: Shows all files, including hidden ones.
  + /P: Pauses the listing after each screen fills up.
  + /O: Sorts directory listing by various criteria (e.g., /O N for sorting by name).

**Example:** dir /W /A // Shows a wide listing of all files, including hidden ones

**8. ECHO:**

* **Purpose:** Displays text on the screen.
* **Options:**
  + /ON: Enables command echoing (displays the commands you type).
  + /OFF: Disables command echoing.

**Example:** echo Today is a great day! // Prints "Today is a great day!" on the screen

**9. EXIT:**

* **Purpose:** Closes the Command Prompt window.
* **Options:** (No options)

**Example:** exit // Closes the Command Prompt window

**10. FC (File Compare):**

* **Purpose:** Compares the contents of two files and highlights differences.
* **Options:**
  + /A: Displays only lines that differ between the files.
  + /C: Displays only the byte count of the differences.
  + /L: Displays only the line numbers of the differences.
  + /T: Ignores leading tabs and spaces when comparing.

**Example:** fc file1.txt file2.txt // Compares "file1.txt" and "file2.txt"

**11. FIND (continued):**

* **Options:** (continued)
  + /V: Displays only lines that **don't** contain the specified text.
  + /C: Counts the number of lines containing the text.
  + /I: Performs a case-insensitive search.

**Example:** find /V error message.txt // Shows lines without "error" in "message.txt"

**12. FINDSTR (more advanced than FIND):**

* **Purpose:** Searches for text patterns using regular expressions.
* **Options:** Refer to help findstr for detailed options.

**Example:** findstr /S /I "important.\*?" c:\ // Searches for lines starting with "important" anywhere in drive C: (case-insensitive)

**13. MKDIR:**

* **Purpose:** Creates a new directory.
* **Options:** (No options, but you can specify subdirectory levels within the path)

**Example:** mkdir Documents\Work // Creates a subfolder named "Work" within "Documents"

**14. MOVE:**

* **Purpose:** Moves a file or folder from one location to another.
* **Options:** All options available for COPY can also be used with MOVE.

**Example:** move report.txt c:\Work // Moves "report.txt" to the "Work" folder

**15. RENAME:**

* **Purpose:** Renames a file or folder.
* **Options:** (No further options beyond specifying the new filename)

**Example:** rename old\_file.txt new\_file.txt // Renames "old\_file.txt" to "new\_file.txt"

**16. REPLACE:**

* **Purpose:** Replaces specific text within a file.
* **Options:** Complex syntax for specifying the text to find, replace with, and file(s) to modify. Refer to help replace for details.

**17. RMDIR:**

* **Purpose:** Removes an empty directory.
* **Options:**
  + /S: Removes the specified directory and all its subdirectories (use with caution).
  + /Q: Suppresses confirmation prompts when deleting.

**Example:** rmdir /S /Q temp // Deletes the "temp" directory and its subdirectories (be careful!)

**18. TREE:**

* **Purpose:** Displays a hierarchical structure of directories and subdirectories.
* **Options:**
  + /F: Displays the full path for each directory.
  + /L: Limits the display to a specific number of subdirectory levels.
  + /A: Shows all files, including hidden ones.

**Example:** tree /F /A // Shows the directory tree with full paths and all files

**19. SORT:**

* **Purpose:** Sorts the contents of a text file.
* **Options:**
  + /R: Sorts in reverse order (descending).
  + /N: Sorts by line number.
  + Refer to help sort for more options and sorting criteria.

**Example:** sort /R names.txt // Sorts "names.txt" in reverse alphabetical order

**20. TITLE:**

* **Purpose:** Sets the title of the Command Prompt window.
* **Options:** (No additional options, specify the new title text after the command)

**Example:** title My Custom Title // Sets the Command Prompt title

**21. PRINT:**

* **Purpose:** Redirects the output of a command to the printer.
* **Options:**
  + > PRN: Redirects output to the default printer.
  + >> filename.txt: Appends output to a text file.

**Example:** dir > listing.txt // Saves directory listing to "listing.txt"

**22. HELP:**

* **Purpose:** Provides help information on other DOS commands.
* **Options:**
  + help: Lists all available commands.
  + help <command\_name>: Provides specific help for a command.
* **Example:** help attrib // Shows help for the ATTRIB command

**23. TIME:**

* **Purpose:** Displays the current system time.
* **Options:** (No options, but custom formats are available)
  + /T: Displays both date and time (already covered in the DATE explanation).

**Example:** This information was already covered in the explanation for the DATE command.

**24. VER:**

* **Purpose:** Displays the version of DOS you are using.
* **Options:** (No options)

**Example:** ver // Shows the DOS version information

**25. PROMPT:**

* **Purpose:** Customizes the Command Prompt appearance.
* **Options:** Allows setting various elements like the drive letter, current directory, username, and more within the prompt string.
  + $G: Displays the current username in the prompt.
  + $H: Displays the hostname in the prompt.
  + $D: Displays the current date in the prompt (format depends on regional settings).
  + $T: Displays the current time in the prompt (format depends on regional settings).
  + $P: Displays the current drive letter and path in the prompt.
  + $\_: Displays the command substitution symbol (used for advanced customization).
* **Example:** prompt $G [$P] > // Sets the prompt to show username and current directory pat

**Observations:**

I successfully completed the DOS command practice session and demonstrated a good understanding of the fundamental concepts. This experience will serve as a valuable foundation for further exploration of operating systems and file management techniques.

**Conclusion:**

Overall, practicing these DOS commands has been a valuable experience. While some may not be as relevant in today's world with graphical user interfaces, understanding these core functionalities gives me a deeper appreciation of how computers manage files and directories. As a student, this knowledge can be helpful for managing my coursework files and troubleshooting basic computer issues.

**Quiz:**

1. **Identify types of OS based on its working.**

**Batch Processing OS:** Instructions (jobs) are submitted in batches and processed sequentially, one after another. Users don't interact directly with the system while jobs are running. (e.g., early mainframe operating systems)

**Multitasking OS:** Allows multiple programs to run concurrently, giving the illusion that the system is handling them all simultaneously. Modern operating systems like Windows, macOS, and Linux are multitasking.

**Multiprogramming OS:** Similar to multitasking, but allows multiple programs to be loaded into memory at the same time. However, only one program can execute at a given time, with the CPU rapidly switching between them. (e.g., some older operating systems)

**Real-Time OS:** Designed to respond to events with minimal delay, often used in embedded systems, robotics, and industrial control applications. Timeliness is crucial, so real-time OSes prioritize tasks based on criticality.

**Single-user OS:** Only allows one user to interact with the system at a time. (e.g., MS-DOS)

**Multi-user OS:** Supports multiple users accessing the system concurrently, each with their own accounts and environments. Modern operating systems like Windows Server and Linux distributions are multi-user.

**Distributed OS:** Spans multiple interconnected computers, allowing users and applications to access resources across the network as if they were on a single system.

1. **Write use of Shell in Linux.**

The shell is the user interface for interacting with the Linux operating system. It's a command-line interpreter that accepts user input (commands) and executes them on the kernel (the core of the OS). Here are some key uses of the shell in Linux:

* **Managing files and directories:** You can create, delete, rename, copy, move, and search for files and folders using shell commands.
* **Launching programs:** You can start applications by typing their names or paths in the shell.
* **Automating tasks:** You can write scripts (series of commands) to automate repetitive tasks.
* **System administration:** Many system administration tasks, like managing users and permissions, configuring hardware, and installing software, are performed through the shell.
* **Customizing the environment:** You can personalize your shell experience by setting aliases for frequently used commands or configuring your prompt to display desired information.

1. **Give the benefits of using Virtualbox?**

Virtualbox is a free and open-source virtualization software that allows you to run multiple operating systems on a single physical computer. Here are some advantages of using Virtualbox:

* **Testing and development:** You can safely test software or experiment with different operating systems within virtual machines without affecting your host system.
* **Resource isolation:** Each virtual machine has its own allocated resources (CPU, memory, disk space), preventing conflicts between different OSes.
* **Portability:** You can easily move virtual machines between different host computers as long as they have Virtualbox installed.
* **Cost-effective:** Virtualbox is free to use, so you can create multiple virtual environments for various purposes without additional hardware costs.
* **Disaster recovery:** You can create backups of virtual machines for disaster recovery purposes, allowing you to restore a system quickly if needed.
* **Learning different operating systems:** Virtualbox is a great tool for learning and experimenting with new operating systems without affecting your primary system.

**Suggested Reference:**

1. Operating Systems: Internals & Design Principles, 9th Edition, William Stallings, Pearson Education India
2. Operating System Concepts, 9th edition Peter B. Galvin, Greg Gagne, Abraham Silberschatz, John Wiley & Sons, Inc.
3. Modern Operating Systems-By Andrew S. Tanenbaum (PHI)
4. UNIX : Concepts and Applications | 4th Edition by Sumitabha Das ,McGrawHill
5. https://ubuntu.com/tutorials/how-to-run-ubuntu-desktop-on-a-virtual-machine-using- virtualbox#1-overview

**References used by the students:**

ChatGPT , [W3Schools](https://www.w3schools.com/)

**Rubric-wise marks obtained:**

| **Rubrics** | **Understanding Related to OS/Computer System (4)** | | | **Command Execution (4)** | | | **Documentation &Timely Submission**  **(2)** | | | **Total (10)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(2)** | **Avg.**  **(1)** | **Poor**  **(0)** |  |
| **Marks** |  | | |  | | |  | | |  |

# Experiment No: 2

**AIM : Study and execute Basic and directory manipulation commands of LINUX/UNIX.**

* 1. Study of Unix Shell and Environment Variables.
  2. man, cal, date, echo, bc, who, uname
  3. Using commands : pwd, mkdir, cd, rmdir ,ls generate given tree.

**Date: *16/5/2023***

**Competency and Practical Skills:** Basic Skills to work with Computer System/ Linux Terminal

**Relevant CO:** , CO5

**Objectives:**

* + 1. To understand the importance of Shell/Envirnment Variable
    2. To Work with basic Commands.
    3. To work/Access the Directory commands.

**Equipment/Instruments:** Computer System with Linux OS.

**Safety and necessary Precautions:**

✔ Operate computer system carefully and responsibly.

✔ Use required lab resources cautiously

**Theory:**

A Unix shell is a command-line interpreter or shell that provides a command line user interface for Unix-like operating systems. The shell is both an interactive command language and a scripting language, and is used by the operating system to control the execution of the system using shell scripts.

Environment variables basically define the behavior of the environment. They can affect the running processes or programs executed in the environment. Every Linux process has an associated set of environment environment variables are typically accessed through the shell. The shell is a command-line interface that interprets and executes commands entered by the user. It provides a way to set, modify, and retrieve environment variables.just like programming language variables The scope of any variable is the region from which it can be accessed or over which it is defined. An environment variable in Linux can have global or local scope.

There are Large set of Commands supported by Linux/Unix. In this Practical we will study Basic System Command and Directory Commands.

**Basic Linux and Directory commands**

| **Command Name** | **Description** |
| --- | --- |
| man | It is used to open help manual for any linux command |
| cal | It displays the current month calendar (System). |
| date | It displays current date (System). |
| echo | It prints the string as provided by user. |
| bc | It is a command line calculator used to do basic mathematical calculations. |
| who | It prints information about users who are currently logged in. |
| uname | It displays the information about the system. |
| pwd | It displays name of present working directory. |
| mkdir | It creates new directory. |
| cd | It is used to move from one directory to another. |
| rmdir | It is used remove the directory. |
| ls | Listing the files and directory. |

For any command, help manual is available in Linux System. We can use **man** Command to open help manual for any command using following command

### 1. man

* Description: Displays the manual pages for other commands.
* Syntax: man [option] [command]
* Options:
  + -C file: Specifies the configuration file.
  + -M path: Specifies the search path for manual pages.
  + -P pager: Specifies the pager to use.
  + -B browser: Specifies the web browser to use.
  + -H browser: Specifies the HTML pager to use.
  + -S section\_list: Specifies a colon-separated list of manual sections to search.
  + -a: Displays all matching manual pages.
  + -c: Reformat the source man page.
  + -d: Print debugging information.
  + -D: Both display and print debugging information.
  + -f: Equivalent to whatis.
  + -k: Equivalent to apropos.
  + -K: Search within man page descriptions.
  + -l: Displays local files only.
  + -m system: Searches for manual pages from the specified system.
  + -p string: Specifies the sequence of preprocessors to run before formatting.
  + -w: Outputs the location of the manual page(s) that would be displayed.
  + -W: Like -w, but in addition, display the name of each file.
* Example: man ls

### 2. cal

* Description: Displays a calendar.
* Syntax: cal [options] [[month] year]
* Options:
  + -1: Display a single month (default).
  + -3: Display the previous, current, and next month.
  + -s: Use Sunday as the first day of the week.
  + -m: Use Monday as the first day of the week.
  + -y: Display a calendar for the current year.
  + -V: Display version information and exit.
* Example: cal 2023

### 3. date

* Description: Displays or sets the system date and time.
* Syntax: date [options] [+format]
* Options:
  + -d, --date=STRING: Display time described by STRING, not 'now'.
  + -f, --file=DATEFILE: Like --date once for each line of DATEFILE.
  + -I, --iso-8601[=FMT]: Output date/time in ISO 8601 format.
  + -r, --reference=FILE: Display the last modification time of FILE.
  + -R, --rfc-email: Output date/time in RFC 5322 format.
  + -u, --utc, --universal: Print or set Coordinated Universal Time (UTC).
  + -v, --version: Output version information and exit.
* Example: date +"%Y-%m-%d"

### 4. echo

* Description: Displays a line of text.
* Syntax: echo [option] [string]
* Options:
  + -n: Do not output the trailing newline.
  + -e: Enable interpretation of backslash escapes.
  + -E: Disable interpretation of backslash escapes (default).
* Example: echo "Hello, World!"

### 5. bc

* Description: An arbitrary precision calculator language.
* Syntax: bc [options] [file]
* Options:
  + -h, --help: Print this help and exit.
  + -i, --interactive: Force interactive mode.
  + -l, --mathlib: Define the standard math library.
  + -q, --quiet: Do not print the normal GNU bc welcome.
  + -s, --standard: Parse with standard GNU bc behavior.
  + -w, --warn: Give warnings for extensions to POSIX bc.
  + -v, --version: Print the version number and exit.
* Example: echo "5 + 5" | bc

### 6. who

* Description: Shows who is logged on.
* Syntax: who [options]
* Options:
  + -a, --all: Same as -b -d --login -p -r -t -T -u.
  + -b, --boot: Time of last system boot.
  + -d, --dead: Print dead processes.
  + -H, --heading: Print a line of column headings.
  + --ips: Print the IP address of the remote host.
  + -l, --login: Print system login processes.
  + -p, --process: Print active processes spawned by init.
  + -q, --count: All login names and number of users logged on.
  + -r, --runlevel: Print current runlevel.
  + -s, --short: Print only name, line, and time (default).
  + -t, --time: Print last system clock change.
  + -T, --mesg: Add user's message status as +, - or ?.
  + -u, --users: List users logged in.
* Example: who -H

### 7. uname

* Description: Prints system information.
* Syntax: uname [options]
* Options:
  + -a, --all: Print all information.
  + -s, --kernel-name: Print the kernel name.
  + -n, --nodename: Print the network node hostname.
  + -r, --kernel-release: Print the kernel release.
  + -v, --kernel-version: Print the kernel version.
  + -m, --machine: Print the machine hardware name.
  + -p, --processor: Print the processor type.
  + -i, --hardware-platform: Print the hardware platform.
  + -o, --operating-system: Print the operating system.
  + --help: Display help message and exit.
  + --version: Output version information and exit.
* Example: uname -a

### 8. pwd

* Description: Prints the current working directory.
* Syntax: pwd [options]
* Options:
  + -L, --logical: Use PWD from environment, even if it contains symlinks (default).
  + -P, --physical: Avoid all symlinks.
* Example: pwd

### 9. mkdir

* Description: Creates directories.
* Syntax: mkdir [options] directory
* Options:
  + -m, --mode=MODE: Set file mode (as in chmod), not a=rwx - umask.
  + -p, --parents: No error if existing, make parent directories as needed.
  + -v, --verbose: Print a message for each created directory.
  + --help: Display help message and exit.
  + --version: Output version information and exit.
* Example: mkdir new\_directory

### 10. cd

* Description: Changes the current directory.
* Syntax: cd [directory]
* Options: This command typically does not have options. It primarily changes directories.
  + cd -: Switches to the previous directory.
* Example: cd /home/user

### 11. rmdir

* Description: Removes empty directories.
* Syntax: rmdir [options] directory
* Options:
  + -p, --parents: Remove DIRECTORY and its ancestors.
  + -v, --verbose: Output a diagnostic for every directory processed.
  + --ignore-fail-on-non-empty: Ignore each failure that is solely because a directory is non-empty.
* Example: rmdir empty\_directory

### 12. ls

* Description: Lists directory contents.
* Syntax: ls [options] [file]
* Options:
  + -a, --all: Do not ignore entries starting with .
  + -A, --almost-all: Do not list implied . and ..
  + --author: With -l, print the author of each file.
  + -b, --escape: Print C-style escapes for nongraphic characters.
  + --block-size=SIZE: Scale sizes by SIZE before printing them; e.g., '--block-size=M'; see SIZE format below.
  + -B, --ignore-backups: Do not list implied entries ending with ~.
  + -c: With -lt: sort by, and show, ctime (time of last modification of file status information) with -l: show ctime and sort by name otherwise: sort by ctime.
  + -C: List entries by columns.
  + --color[=WHEN]: Colorize the output; WHEN can be 'always' (default if omitted), 'auto', or 'never'.
  + -d, --directory: List directories themselves, not their contents.
  + -D, --dired: Generate output designed for Emacs' dired mode.
  + -f: Do not sort, enable -aU, disable -ls --color.
  + -F, --classify: Append indicator (one of \*/=>@|) to entries.
  + --file-type: Likewise, except do not append '\*'.
  + --format=WORD: Across -x, commas -m, horizontal -x, long -l, single-column -1, verbose -l, vertical -C.
  + --full-time: Like -l --time-style=full-iso.
  + -g: Like -l, but do not list owner.
  1. **Study of Unix Shell and Environment Variables.**
     1. **Display The name of the current shell of your System.**

### Method 1: Using $0 variable

The $0 variable typically contains the name of the shell script or shell.

syntax:

echo $0

### Method 2: Using $SHELL environment variable

The $SHELL environment variable holds the default shell for the current user, which might differ from the current running shell if it was changed during the session.

syntax: echo $SHELL

### Method 3: Using ps command

You can also use the ps command to get the current shell's process information.

syntax: ps -p $$ -o comm=

* + 1. **Write the different shell names available in Linux.**

1. bash (Bourne Again Shell): The most widely used shell, which is an enhanced version of sh with additional features and improvements. It is the default shell in many Linux distributions.
2. csh (C Shell): A shell with C-like syntax, created by Bill Joy. It includes features such as aliasing and job control.
3. tcsh (TENEX C Shell): An enhanced version of csh with additional features such as command-line editing and programmable word completion.
4. ksh (Korn Shell): Created by David Korn, it is backward compatible with the Bourne shell and includes features from the C shell. There are several versions, including ksh88 and ksh93.
5. zsh (Z Shell): A highly configurable and interactive shell with many features, including spell checking, improved tab completion, and the ability to customize prompts.
6. fish (Friendly Interactive Shell): Known for its user-friendliness and out-of-the-box features like syntax highlighting, autosuggestions, and easier scripting.
7. dash (Debian Almquist Shell): A modern, POSIX-compliant shell that is faster and smaller than bash, often used as the default /bin/sh in Debian-based systems.
8. ash (Almquist Shell): A lightweight sh-compatible shell often used in embedded Linux systems due to its small size.
   * 1. **Display the name of Enviornment Variable.**

**Using printenv with cut**

This command will list only the names of the environment variables:

**Syntax** : printenv | cut -d= -f1

* 1. **cal, date, echo, bc, who, uname**

2.2.1. Display current month Calendar:

cal

2.2.2. Display Calendar of given month and year:

cal <month> <year>

Example: To see the calendar for May 2024, type:

cal 5 2024

2.2.3. Display calendar of current, previous and next month:

cal -3

2.2.4. Display calendar with Monday as start day of week:

cal -w 1

2.2.5. Display calendar with Sunday as start day of week (default):

cal # No argument needed, Sunday is default start day

2.2.6. Display calendar without highlighting current date:

cal -H

2.2.7. Display current month number:

date +%m

2.2.8. Display current year in two digits:

date +%y

2.2.9. Display day of current date (short and full):

* Short: date +%d
* Full: date +%A

2.2.10. Display month of current date (short and full):

* Short: date +%b
* Full: date +%B

2.2.11. Display date in mm/dd/yyyy format:

date +%m/%d/%Y

2.2.12. Display date in yyyy-mm-dd format:

date +%Y-%m-%d

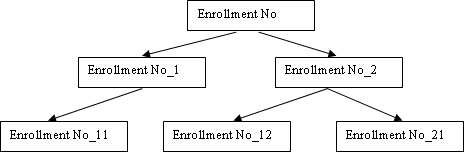
2.2.13. Display the century of current year:

You can't directly get the century using the date command. However, you can calculate it using basic math:

century=$(( (`date +%Y` / 100) ))

echo $century"th Century"

* 1. **Using pwd, mkdir, cd, rmdir,ls commands generate given tree.**



Student has to create Each directory with his/her enrollment number.

* + 1. Display the path from root directory to the last level directory of the tree.(Consider Root directory of given tree as root)

The path from the root directory to the last level directory (Enrollment No\_11, Enrollment No\_12, Enrollment No\_21) would be:

root/Enrollment No

* + 1. Remove Directory Enrollment\_12.

To remove directories using the rm command, you'll need to use the -r flag to indicate recursive deletion, meaning it will remove the directory and all its contents. However, be cautious as this action cannot be undone.

**Here's how to remove Enrollment\_12:**

rm -r Enrollment\_No/Enrollment\_No\_12

* + 1. Remove Directore Enrollment\_1.

**Here's how to remove Enrollment\_1 (assuming it's empty):**

rm -r Enrollment\_No/Enrollment\_No\_1

**Observations:**

1. Understanding the Importance of Shell/Environment Variable:
   * Shell environment variables play a crucial role in the Linux operating system as they define the behavior and environment for various processes and commands.
   * These variables provide information about the system, user preferences, and control over the execution environment.
2. Working with Basic Commands:
   * Basic commands such as cal, date, echo, etc., are fundamental tools in Linux used for various tasks like displaying calendars, managing dates, printing text, etc.
   * They provide essential functionalities that are frequently used in shell scripting, system administration, and day-to-day operations.
3. Accessing Directory Commands:
   * Directory commands like pwd, mkdir, cd, ls, etc., are vital for navigating the file system, creating, and managing directories and files.
   * These commands enable users to interact with the file system efficiently, facilitating tasks such as organizing files, accessing directories, and managing permissions.

**Conclusion:**

In conclusion, this practical exercise provided valuable insights into the significance of shell/environment variables and basic commands in Linux. By working with these tools, we gained a better understanding of how to manipulate the environment, perform common tasks, and navigate the file system effectively. The hands-on experience with directory commands enhanced our ability to manage files and directories, demonstrating the practical utility of shell commands in everyday computing tasks and system administration. Overall, this exercise has laid a solid foundation for further exploration and proficiency in Linux command-line operations.

**Quiz:**

1. **Give Significance of Shell in Linus OS.**
2. Command Execution: The shell acts as an interface between the user and the Linux kernel, allowing users to execute commands and run programs.
3. Automation and Scripting: Shells support scripting languages, enabling users to automate tasks, create complex workflows, and develop scripts for system administration, file management, and other purposes.
4. Customization: Shells provide extensive customization options through environment variables, configuration files, and shell scripts. Users can tailor their shell environment to suit their preferences and workflow.
5. Input/Output Redirection: Shells allow users to redirect input and output streams, enabling efficient handling of files, streams, and command output.
6. Job Control: Shells support job control features, such as background processing, foreground control, and process manipulation, enhancing multitasking capabilities.
7. **How to move in/out from directory using cd in single step.**
8. **Write Use of bc Command.**

The bc command is a versatile arbitrary precision calculator language. It can be used for various mathematical calculations, including arithmetic operations, mathematical functions, and scripting. Some common use cases of the bc command include:

1. Arithmetic Operations: bc can perform basic arithmetic operations such as addition, subtraction, multiplication, and division.
2. Mathematical Functions: bc supports various mathematical functions such as square root, exponentiation, logarithms, trigonometric functions, etc.
3. Scripting: bc can be used within shell scripts to perform complex calculations, process numerical data, and generate output dynamically.
4. Precision Control: bc allows users to control the precision of calculations, enabling accurate results for both integer and floating-point arithmetic.
5. Conversion: bc can be used for converting numeric values between different bases (binary, octal, hexadecimal) and performing conversions between units of measurement.

**Suggested Reference:**

1. Operating Systems: Internals & Design Principles, 9th Edition, William Stallings, Pearson Education India
2. Operating System Concepts, 9th edition Peter B. Galvin, Greg Gagne, Abraham Silberschatz, John Wiley & Sons, Inc.
3. Modern Operating Systems-By Andrew S. Tanenbaum (PHI)
4. UNIX : Concepts and Applications | 4th Edition by Sumitabha Das ,McGrawHill

**References used by the students:**

***// Write references used by you here***

**Rubric-wise marks obtained:**

| **Rubrics** | **Understanding of commands (4)** | | | **Ability to use Command for question solving**  **(4)** | | | **Documentation &Timely Submission**  **(2)** | | | **Total (10)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(2)** | **Avg.**  **(1)** | **Poor**  **(0)** |  |
| **Marks** |  | | |  | | |  | | |  |

# Experiment No: 3

AIM : Study and execute Basic File manipulation commands.

1. cat 2.wc 3. cp 4. mv 5.rm 6. File 7. cmp 8. comm 9. diff 10. chmod 11. sort

**Date: *// Write date of experiment here***

**Competency and Practical Skills:** Basic Skills to work with Computer System/ Linux Terminal

**Relevant CO:** , CO5

**Objectives:**

1. To understand the importance of Shell/Envirnment Variable
2. To work with basic File Operations.
3. To work/Access the Directory commands.

**Equipment/Instruments:** Computer System with Linux OS.

**Safety and necessary Precautions:**

✔ Operate computer system carefully and responsibly.

✔ Use required lab resources cautiously

**Theory:**

File manipulation commands are mainly used for Operations like create File, Copy File, Delete File, Rejname file, Searching No. of counts from file, finding Common values from two file, Comparing to files, finding difference between two file to make them identical. There is also File Permission command chmod to change the fillle permission.

**Basic File commands**

| **Command Name** | **Description** |
| --- | --- |
| cat | It redirects standard output to/from the file. |
| wc | It counts word, characters and lines from the file. |
| cp | Copy one file to another. |
| mv | Rename or move file from one directory to another. |
| rm | Remove the file from directory. |
| file | Displays file types and other details. |
| cmp | Compare the content of two files. |

| comm. | Find common between two files. |
| --- | --- |
| diff | Find the difference between two files and give way to make them identical. |
| chmod | It changes file permission. |
| sort | It sorts the file content. |

* 1. Create 4 Files using cat Command.
     + **Create 4 Files F1.txt ,F2.txt ,F3.txt ,F4.txt with some content.**
     + **Preferable to put paragraph in one file.**
     + **In File F2 and F3 write content in form of words in alphabetical sorted order**

**Also put some common value in both file (like student name or engineering branch names, fruits, vegetables, etc)**

* + - **In fourth File write decimal numbers separated by newline.**
  1. Copy Content of File F1 to F1\_c.
  2. Rename File F1\_c to F5.
  3. Compare File F1 and F5
  4. Move file F5 to another Directory of your Choice.
  5. Count no.of characters,words and Lines of F1 file.
  6. Find Common Value between File F2 and F3.
  7. Find the Difference Between File F2 and F3.
  8. Change the F1 file permission to read Only.
  9. Change F2 permission to Read and Write Only.
  10. Change File F4 permission to Read,Write and Execute .
  11. Perform sort command on F4.
  12. Perform numeric Sort on File F4.

Observations:

*// Write your observation for comm., cmp, diff, sort, chmod command.*

Conclusion:

*// Write conclusion here*

Quiz:

1. Why there is need to change file permission.
2. How to Change File permission in different way using chmod command.
3. Write the use of chown command.
4. Write use of “ |” character in Linux.

**Suggested Reference:**

1. Operating Systems: Internals & Design Principles, 9th Edition, William Stallings, Pearson Education India
2. Operating System Concepts, 9th edition Peter B. Galvin, Greg Gagne, Abraham Silberschatz, John Wiley & Sons, Inc.
3. Modern Operating Systems-By Andrew S. Tanenbaum (PHI)
4. UNIX : Concepts and Applications | 4th Edition by Sumitabha Das ,McGrawHill

**References used by the students:**

***// Write references used by you here***

**Rubric-wise marks obtained:**

| **Rubrics** | **Understanding of commands (4)** | | | **Ability to use Command for question solving**  **(4)** | | | **Documentation &Timely Submission (2)** | | | **Total (10)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(2)** | **Avg.**  **(1)** | **Poor**  **(0)** |  |
| **Marks** |  | | |  | | |  | | |  |

**Experiment No: 4**

**AIM :** Study and Execute Advance Filter Commands.

1.head 2. tail 3. paste 4. cut(-f) 5. cut(-c) 6. grep

**Date: *// Write date of experiment here***

**Competency and Practical Skills:** Basic Skills to work with Computer System/ Linux Terminal

**Relevant CO:** , CO5

**Objectives:**

1. To understand the importance of File filter commands.
2. To merge, split and Search in different way from the file.

**Equipment/Instruments:** Computer System with Linux OS.

**Safety and necessary Precautions:**

✔ Operate computer system carefully and responsibly.

✔ Use required lab resources cautiously

**Theory:**

Filter commands accept input data from standard input and produce output standard output. It transforms plain-text data into a meaningful way and can be used merge with other output or file. These filters are very small programs that are designed for a specific function which can be used as building blocks.There are number of commands we already have covered I n previous practicals like sort,comm.,cat and others are cut , paste, head,tail, grep, tee ,uniq, grep. Using these commands one can search and display specif content from file also filter some specific data.

For more description about the command we can read a Linux Help manual using command :

**$man Commandname.**

| **Command Name** | **Description** |
| --- | --- |
| head | It displays first 10 lines from the input file. |
| tail | It displays last 10 lines from the file. |
| cut | It cut the file content Vertically(-f) as well as Horizontally(-c). |

| paste | It pastes content of different files and displays the output. |
| --- | --- |
| grep | Search the line with specified pattern present in the file(s). |

grep command searches the simple pattern as well pattern specified using Regular expression.To create a pattern which matches different types of string we can use different wildcard characters to create a new pattern. The basic set of wildcards in are:

**\* –** This wildcard represents all the characters. Also represent one or more occurrence of preceded character.

**+ -** represent one or more occurrence of preceded character.

**? –** This wildcard represents a single character

**[ ] –** This wildcard represents a range of characters.

To solve the question given student has to create a text file with Student details like:

**Enrollment number, Student name, Birth-date, Semester, Gender , Email Address, SPI**

**,where each field is separated by delimiter character “|” or any character as per wish.**



* 1. Display first 7 lines of the file.
  2. Display Last 4 Lines of the File.
  3. Cut the file column wise and display Student Enrollment number, Gender, Email Address.
  4. Cut the file column wise and display Student name , Branch,semester.
  5. Merge the result of Question 4.3 and 4.4 using delimiter $ and store it File named result1 using tee command.
  6. Cut the File Fieldwise (vertically) to display second, third and fourth field of the file.
  7. Display the First three student details having highest SPI in decreasing order.
  8. Change file delimiter “|” with another “\* “.
  9. Display student details studying in same branch.
  10. Display student details studying in same branch and same semester.
  11. Display students have SPI greater than 6.
  12. Display student details studying Whose surname starts will “p” and ends with “l”.
  13. Display the name of student whose surname is agarwal (surname may be in any form).
  14. Display details of all girls students.

**Observations:**

***// Write your observation here***

**Conclusion:**

***// Write conclusion here***

**Quiz:**

1. What is the use of tee and cat command.
2. By default, how many lines are displayed using the head command? which option used to display specific lines from the file?
3. Which option is used to display file content in reverse order?

**Suggested Reference:**

1. Operating Systems: Internals & Design Principles, 9th Edition, William Stallings, Pearson Education India
2. Operating System Concepts, 9th edition Peter B. Galvin, Greg Gagne, Abraham Silberschatz, John Wiley & Sons, Inc.
3. Modern Operating Systems-By Andrew S. Tanenbaum (PHI)
4. UNIX : Concepts and Applications | 4th Edition by Sumitabha Das ,McGrawHill

**References used by the students:**

***// Write references used by you here***

**Rubric-wise marks obtained:**

| **Rubrics** | **Understanding of commands (4)** | | | **Ability to use Command for question solving**  **(4)** | | | **Documentation &Timely Submission (2)** | | | **Total (10)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(2)** | **Avg.**  **(1)** | **Poor**  **(0)** |  |
| **Marks** |  | | |  | | |  | | |  |

# Experiment No: 5

**AIM : Write a shell script program using Loop/ control structure.**

* 1. **Write a shell script to find factorial of given number n**
  2. **Write a shell script which will generate first n fibonnacci numbers like: 1, 1, 2, 3, 5, 13,…**

**Date: *// Write date of experiment here***

**Competency and Practical Skills:** Logic building and programming

**Relevant CO:** CO5

**Objectives:**

a. To understand and use the loop and control structure to sove problem using shell script.

**Equipment/Instruments:** Computer System with Linux OS.

**Safety and necessary Precautions:**

✔ Operate computer system carefully and responsibly.

✔ Use required lab resources cautiously

**Theory:**

Linux/Unix shells are interactive, by means they accept commands as input from users and execute them and display the output accordingly. Normally we are executing command independently one by one. Sometimes it is require to execute same no.of commands repeatedly which is time consuming task but linux system also has solution for that called shell programming. For that we can put all commands together in single file and execute them in shell to avoid repetitive work. These files are called **Shell Scripts** or **Shell Programs**. Shell scripts are similar to the batch file in MS-DOS. The shell script file is saved with **“.sh”** extension e.g., **First\_program.sh.**

A shell script has syntax like other programming language. If you have any prior experience of programming language like Python, C/C++ etc. It would be very easy to understand shell programming. It has it’s Shell Keywords, Control flow statements, Loop statements, Shell (linux command we had used in previous experiments)and Functions.

To run the shell script program, file must be executable. To set execute permission of file we can use chmod command.e.g**. chmod 777 file.sh** command allows file.sh to execute.to run the file one can write the command **./file.sh.**

* **There are 3 types loop statements supported by shell programming:**

1. **while statement**

Syntax:

while <condition> do

<command statement 1>

<command statement 2>

.

.

<command statement n> done

1. **for statement**

The for loop operates on lists . It repeats a set of commands for every item as per list value. var is the variablename and var takes value from the list value1, value2, … value n on each iteration, respectively,

**Syntax:**

for <var> in <value 1 value 2 ... value n> do

<command statement 1>

<command statement 2>

.

.

<command statement n> done

1. **do…until statement**

The do… until loop is executed as many times as th condition/command evaluates too false. The loop terminates when the condition/command becomes true.

**Syntax:**

until <condition> do

<command statement 1>

<command statement 2>

.

.

done

To change the flow of loop statements, two commands are used they are,

1. break
2. continue
   * **There are basically 2 types of contro statement supported in shell programming:**
3. **if –else statement (different versions)**
4. **Simple if statement Syntax:**

if [ expression ] then

<command statement 1>

fi

1. **if-else statement Syntax:**

if [ expression ] then

<command statement 1> else

<command statement 2>

fi

1. if..elif..else..fi statement (Else If ladder)

**Syntax:**

if [ expression1 ] then

<command statement 1>

<command statement 2> elif [ expression2 ]

then

<command statement 3>

<command statement 4>

.

else

<command statement 5>

fi

1. if..then..else..if..then..fi..fi..(Nested if)

**Syntax:**

1. **switch statement Syntax:**

case “expression” in

Pattern 1) < command Statement 1> ;; Pattern 2) < command Statement 2> ;;

.

.

.

Pattern n) < command Statement n> ;; esac

**Example1 : “Checking whether two numbers are equal or not?”**

a=20 b=21

if [ $a -gt $b ] then

else

fi

**OUTPUT:**

#If they are equal then print this echo "a is greater than b"

#else print this

echo "a is less than b"

5.PNG

**Example 2: “shell script to Display branchcode of respective branch using case control statement”**

echo "Enter department name" read DEPARTMENT

case $DEPARTMENT in

"CE") echo " branch code:07" ;;

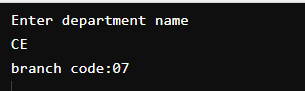
"EC") echo " branch code:11" ;; "CIVIL") echo " branch code:06" ;; "IT") echo " branch code:16" ;; "MECH") echo " branch code:19" ;;

\*) echo -n "Invalid"

;;

esac

**OUTPUT:**



* 1. **Write a shell script to find factorial of given number n.**

**Program:**

***// Write code of your program here***

**Output:**

***// Write output of your program here***

* 1. **Write a shell script which will generate first n fibonnacci numbers like: 1, 1, 2, 3, 5, 13,…**

**Program:**

***// Write code of your program here***

**Output:**

***// Write output of your program here***

**Observations:**

***// Write your observation here***

**Conclusion:**

***// Write conclusion here***

**Quiz:**

* + 1. How to read and print the value of variable in shell script?
    2. Write syntax to retrieve variable value using shell script.
    3. Write a step to run shell script.

**Suggested Reference:**

1. Operating Systems: Internals & Design Principles, 9th Edition, William Stallings, Pearson Education India
2. Operating System Concepts, 9th edition Peter B. Galvin, Greg Gagne, Abraham Silberschatz, John Wiley & Sons, Inc.
3. Modern Operating Systems-By Andrew S. Tanenbaum (PHI)
4. UNIX : Concepts and Applications | 4th Edition by Sumitabha Das ,McGrawHill

**References used by the students:**

***// Write references used by you here***

**Rubric-wise marks obtained:**

| **Rubrics** | **Understanding of Shell programming syntax (4)** | | | **Ability to implement program for given problem using Shell**  **script(4)** | | | **Documentation &Timely Submission (2)** | | | **Total (10)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(2)** | **Avg.**  **(1)** | **Poor**  **(0)** |  |
| **Marks** |  | | |  | | |  | | |  |

# Experiment No – 6

**AIM :** Loop/ control structure using shell script(Using while loop)

* 1. Write a shell script to read n numbers as command arguments and sort them in descending order.
  2. Write a shell script to generate mark sheet of a student. Take 3 subjects, calculate and display total marks, percentage and Class obtained by the student.

**Date: *// Write date of experiment here***

**Competency and Practical Skills:** Logic building and programming

**Relevant CO:** CO5

**Objectives:** explore usage of while loop in shell script

**Equipment/Instruments:** Computer System with Winows/Linux

**Safety and necessary Precautions:**

✔ Operate computer system carefully and responsibly.

✔ Use required lab resources cautiously

**Theory:**

Example:

Shows loop terminates as soon as a becomes 5 a=0

while [ $a -lt 10 ] do

echo $a

if [ $a -eq 5 ] then

break

fi

a=`expr $a + 1` done

* 1. **Write a shell script to read n numbers as command arguments and sort them in descending order.**

**Program:**

**Output:**

* 1. **Write a shell script to generate mark sheet of a student. Take 3 subjects, calculate and display total marks, percentage and Class obtained by the student.**

**Output:**

**Observations:**

***// Write your observation here***

**Conclusion:**

***// Write conclusion here***

**Suggested Reference:** [**https://www.tutorialspoint.com/**](https://www.tutorialspoint.com/)[**https://www.geeksforgeeks.org/**](https://www.geeksforgeeks.org/) **http**[**s://www.java**](http://www.javatpoint.com/)**tpoi**[**nt.com/**](http://www.javatpoint.com/)

[**https://www.tutorialspoint.com/unix/unix-loop-control.htm**](https://www.tutorialspoint.com/unix/unix-loop-control.htm)

**References used by the students:**

***// Write references used by you here***

**Rubric-wise marks obtained:**

| **Rubrics** | **Understanding of Shell programming syntax (4)** | | | **Ability to implement program for given problem using Shell**  **script(4)** | | | **Documentation &Timely Submission (2)** | | | **Total (10)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(2)** | **Avg.**  **(1)** | **Poor**  **(0)** |  |
| **Marks** |  | | |  | | |  | | |  |

# Experiment No – 7

**AIM:** Command execution via Shell script.

* 1. Write a shell script to display all executable files, directories and zero sized files from current directory.

Exit

* 1. Write a menu driven shell script which will print the following menu and execute the given task. MENU
     + Display calendar of current month
     + Display today’s date and time
     + Display usernames those are currently logged in the system
     + Display your name at given x, y position
     + Display your terminal number

**Date: *// Write date of experiment here***

**Competency and Practical Skills:** Logic building and programming

**Relevant CO:** CO5

**Objectives:** explore usage of various searching and date - time related commands.

**Equipment/Instruments:** Computer System with Winows/Linux

**Safety and necessary Precautions:**

✔ Operate computer system carefully and responsibly.

✔ Use required lab resources cautiously

* 1. **Write a shell script to display all executable files, directories and zero sized files from current directory.**

**Theory:**

find command

The **find** command in UNIX is a command line utility for walking a file hierarchy. It can be used to find files and directories and perform subsequent operations on them. It supports searching by file, folder, name, creation date, modification date, owner and permissions. By using the ‘-exec’ other UNIX commands can be executed on files or folders found.

Options:

* + - -exec CMD: The file being searched which meets the above criteria and returns 0 for as its exit status for successful command execution.
    - -ok CMD : It works same as -exec except the user is prompted first.
    - -inum N : Search for files with inode number ‘N’.
    - -links N : Search for files with ‘N’ links.
    - -name demo : Search for files that are specified by ‘demo’.
    - -newer file : Search for files that were modified/created after ‘file’.
    - -perm octal : Search for the file if permission is ‘octal’.
    - -print : Display the path name of the files found by using the rest of the criteria.
    - -empty : Search for empty files and directories.
    - -size +N/-N : Search for files of ‘N’ blocks; ‘N’ followed by ‘c’can be used to measure the size in characters; ‘+N’ means size > ‘N’ blocks and ‘-N’ means size < ‘N’ blocks.
    - -user name : Search for files owned by username or ID ‘name’.
    - \(expr \) : True if ‘expr’ is true; used for grouping criteria combined with OR or AND.
    - ! expr : True if ‘expr’ is false.

Example: Search for a file with a specific name.

1. **$ find ./GFG -name sample.txt**

It will search for sample.txt in GFG directory.

1. **$ find ./GFG -name \*.txt**

It will give all files which have ‘.txt’ at the end.

**Program:**

**Output:**

Exit

* 1. **Write a menu driven shell script which will print the following menu and execute the given task.**

MENU

* + - Display calendar of current month
    - Display today’s date and time
    - Display usernames those are currently logged in the system
    - Display your name at given x, y position
    - Display your terminal number

**Theory:**

**Case Statement**

A case statement in bash scripts is used when a decision has to be made against multiple choices. In other words, it is useful when an expression has the possibility to have multiple values. This methodology can be seen as a replacement for multiple if-statements in a script. Case statements have an edge over if-statements because it improves the readability of our code and they are easier to maintain. Case statements in a Bash script are quite similar to Case statements in C language.

But unlike C, the Bash Case statement stops continuing the search as soon as the match occurs. In

simple words, they don’t require any break statement that is mandatory to be used in C to stop searching for a pattern further.

The basic syntax of a case statement is given below, Syntax:

case EXPRESSION in Pattern\_Case\_1) STATEMENTS

;;

Pattern\_Case\_1) STATEMENTS

;;

Pattern\_Case\_N) STATEMENTS

;;

\*)

STATEMENTS

;;

esac

**cal command**

By default, the cal command shows the current month calendar as output.

cal command is a calendar command in Linux which is used to see the calendar of a specific month or a whole year.

Syntax:

cal [ [ month ] year]

cal 08 2000 : Shows calendar of selected month and year. cal 2018 : Shows the whole calendar of the year.

cal -3 : Shows calendar of previous, current and next month

**who command**

The who command is used to get information about currently logged in user on to system.

Syntax : $who [options] [filename]

Examples :

1. The who command displays the following information for each user currently logged in to the system if no option is provided :

Login name of the users Terminal line numbers

Login time of the users in to system Remote host name of the user

**Program:**

**Output:**

**Observations:**

***// Write your observation here***

**Conclusion:**

***// Write conclusion here***

**Suggested Reference:** [**https://www.tutorialspoint.com/**](https://www.tutorialspoint.com/)[**https://www.geeksforgeeks.org/**](https://www.geeksforgeeks.org/) **http**[**s://www.java**](http://www.javatpoint.com/)**tpoi**[**nt.com/**](http://www.javatpoint.com/)

**References used by the students:**

***// Write references used by you here***

**Rubric-wise marks obtained:**

| **Rubrics** | **Understanding of Shell programming syntax (4)** | | | **Ability to implement program for given problem using Shell**  **script(4)** | | | **Documentation &Timely Submission (2)** | | | **Total (10)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(2)** | **Avg.**  **(1)** | **Poor**  **(0)** |  |
| **Marks** |  | | |  | | |  | | |  |

# Experiment No – 8

**AIM :** Process Scheduling Algorithm and Comparison.

**Date: *// Write date of experiment here***

**Competency and Practical Skills:** Logic building and programming

**Relevant CO:** CO2

**Objectives:** Study and implement process scheduling algorithms.

**Equipment/Instruments:** Computer System with Winows/Linux

**Safety and necessary Precautions:**

✔ Operate computer systems carefully and responsibly.

✔ Use required lab resources cautiously

**Theory:**

* 1. **Write a C Program to Implement Following CPU Scheduling Algorithms.**
     + **FCFS**
     + **Round Robin**

**First Come First Serve (FCFS)** is an operating system scheduling algorithm that automatically executes queued requests and processes in order of their arrival. It is the easiest and simplest CPU scheduling algorithm. In this type of algorithm, processes which request the CPU first get the CPU allocation first. This is managed with a FIFO queue. The full form of FCFS is First Come First Serve.

**Characteristics of FCFS CPU Scheduling Algorithm**

* It supports non-preemptive and pre-emptive scheduling algorithm.
* Jobs are always executed on a first-come, first-serve basis.
* It is easy to implement and use.
* This method is poor in performance, and the general wait time is quite high.

**Round** Robin is a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way. It is basically the preemptive version of First come First Serve CPU Scheduling algorithm. Round Robin CPU Algorithm generally focuses on Time Sharing technique.

The period of time for which a process or job is allowed to run in a pre-emptive method is called time quantum.

Each process or job present in the ready queue is assigned the CPU for that time quantum, if the execution of the process is completed during that time then the process will end else the process will go back to the waiting table and wait for its next turn to complete the execution.

**Characteristics of Round Robin CPU Scheduling Algorithm**

It is simple, easy to implement, and starvation-free as all processes get fair share of CPU. One of the most commonly used technique in CPU scheduling as a core.

It is preemptive as processes are assigned CPU only for a fixed slice of time at most. The disadvantage of it is more overhead of context switching.

**Program:**

**Output:**

**Observations:**

***// Write your observation here***

**Conclusion:**

***// Write conclusion here***

**Suggested Reference:** https:/[/www.tutorialspoint.com/](http://www.tutorialspoint.com/) https:/[/www.g](http://www.geeksforgeeks.org/)e[eksforgeeks.org/](http://www.geeksforgeeks.org/) https:/[/www.javatpoint.com/](http://www.javatpoint.com/)

**References used by the students:**

***// Write references used by you here***

**Rubric-wise marks obtained:**

| **Rubrics** | **Understanding of Shell programming syntax (4)** | | | **Ability to implement program for given problem using Shell**  **script(4)** | | | **Documentation &Timely Submission (2)** | | | **Total (10)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(2)** | **Avg.**  **(1)** | **Poor**  **(0)** |  |
| **Marks** |  | | |  | | |  | | |  |

# Experiment No – 9

**AIM : Process creation and Thread Scheduling**

* 1. Implement Producer consumer problem using thread using C/JAVA programming Language.
  2. Create new thread using fork() system call using C programming Language.

**Date: *// Write date of experiment here***

**Competency and Practical Skills:** Logic building and programming

**Relevant CO:** CO3

**Objectives:** Study and implement thread management.

**Equipment/Instruments:** Computer System with Winows/Linux

**Safety and necessary Precautions:**

✔ Operate computer systems carefully and responsibly.

✔ Use required lab resources cautiously

**Theory:**

The producer-consumer problem (also known as the bounded-buffer problem) is a classic example of a multi-process synchronization problem. The problem describes two processes, the producer and the consumer, which share a common, fixed-size buffer used as a queue.

The producer’s job is to generate data, put it into the buffer, and start again. At the same time, the consumer is consuming the data (i.e. removing it from the buffer), one piece at a time.

Problem:

To make sure that the producer won’t try to add data into the buffer if it’s full and that the consumer won’t try to remove data from an empty buffer.

Solution:

The producer is to either go to sleep or discard data if the buffer is full. The next time the consumer removes an item from the buffer, it notifies the producer, who starts to fill the buffer again. In the same way, the consumer can go to sleep if it finds the buffer to be empty. The next time the producer puts data into the buffer, it wakes up the sleeping consumer.

An inadequate solution could result in a deadlock where both processes are waiting to be awakened.

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light- weight processes within a process.

Threads can be created by using two mechanisms :

* Extending the Thread class
* Implementing the Runnable Interface

The **Fork** system call is used for creating a new process in Linux, and Unix systems, which is called the child process, which runs concurrently with the process that makes the fork() call (parent process). After a new child process is created, both processes will execute the next instruction following the fork() system call.

The child process uses the same pc(program counter), same CPU registers, and same open files which use in the parent process. It takes no parameters and returns an integer value.

Below are different values returned by fork().

**Negative Value:** The creation of a child process was unsuccessful.

**Zero:** Returned to the newly created child process.

**Positive value:** Returned to parent or caller. The value contains the process ID of the newly created child process.

**Example:**

#include <stdio.h> #include <sys/types.h> #include <unistd.h> int main()

{

// make two process which run same

// program after this instruction pid\_t p = fork();

if(p<0){ perror("fork fail"); exit(1);

}

printf("Hello world!, process\_id(pid) = %d \n",getpid()); return 0;

}

* 1. **Implement Producer consumer problem using thread using C/JAVA programming Language.**

**Program:**

**Output:**

* 1. **Create new thread using fork() system call using C programming Language.**

**Program:**

**Output:**

**Observations:**

***// Write your observation here***

**Conclusion:**

***// Write conclusion here***

**Suggested Reference:** https:/[/www.tutorialspoint.com/](http://www.tutorialspoint.com/) https:/[/www.g](http://www.geeksforgeeks.org/)e[eksforgeeks.org/](http://www.geeksforgeeks.org/) https:/[/www.javatpoint.com/](http://www.javatpoint.com/)

<https://www.geeksforgeeks.org/producer-consumer-solution-using-threads-java/> https:/[/www.g](http://www.geeksforgeeks.org/thread-functions-in-c-c/)e[eksforgeeks.org/thread-functions-in-c-c/](http://www.geeksforgeeks.org/thread-functions-in-c-c/)

**References used by the students:**

***// Write references used by you here***

**Rubric-wise marks obtained:**

| **Rubrics** | **Understanding of Shell programming syntax (4)** | | | **Ability to implement program for given problem using Shell**  **script(4)** | | | **Documentation &Timely Submission (2)** | | | **Total (10)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(2)** | **Avg.**  **(1)** | **Poor**  **(0)** |  |
| **Marks** |  | | |  | | |  | | |  |

# Experiment No – 10

**AIM :** Page replacement and Disk Scheduling algorithm

* 1. Implement FIFO Page replacement Algorithm using C/Java.
  2. Implement C-SCAN Disk Scheduling Algorithm using C/Java.

**Date: *// Write date of experiment here***

**Competency and Practical Skills:** Logic building and programming

**Relevant CO:** CO4

**Objectives:** Study and implement memory management by operating system.

**Equipment/Instruments:** Computer System with Winows/Linux

**Safety and necessary Precautions:**

✔ Operate computer systems carefully and responsibly.

✔ Use required lab resources cautiously

**Theory:**

In an operating system that uses paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when a new page comes in.

**Page Fault:** A page fault happens when a running program accesses a memory page that is mapped into the virtual address space but not loaded in physical memory. Since actual physical memory is much smaller than virtual memory, page faults happen. In case of a page fault, Operating System might have to replace one of the existing pages with the newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce the number of page faults.

**First In First Out (FIFO):** This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

A Process makes the I/O requests to the operating system to access the disk. Disk Scheduling Algorithm manages those requests and decides the order of the disk access given to the requests.

Important Terms related to Disk Scheduling Algorithms

**Seek Time -** It is the time taken by the disk arm to locate the desired track.

Rotational Latency - The time taken by a desired sector of the disk to rotate itself to the position where it can access the Read/Write heads is called Rotational Latency.

Transfer Time - It is the time taken to transfer the data requested by the processes.

**Disk Access Time -** Disk Access time is the sum of the Seek Time, Rotational Latency, and Transfer Time.

**C-SCAN**

This algorithm is the same as the SCAN algorithm. The only difference between SCAN and C- SCAN is, it moves in a particular direction till the last and serves the requests in its path. Then, it returns in the opposite direction till the end and doesn't serve the request while returning. Then, again reverses the direction and serves the requests found in the path. It moves circularly.

* 1. **: Implement FIFO Page replacement Algorithm using C/Java. Program:**

**Output:**

* 1. **: Implement C-SCAN Disk Scheduling Algorithm using C/Java. Program:**

**Output:**

**Observations:**

***// Write your observation here***

**Conclusion:**

***// Write conclusion here***

**Suggested Reference:** https:/[/www.tutorialspoint.com/](http://www.tutorialspoint.com/) https:/[/www.g](http://www.geeksforgeeks.org/)e[eksforgeeks.org/](http://www.geeksforgeeks.org/) https:/[/www.javatpoint.com/](http://www.javatpoint.com/) <https://www.baeldung.com/cs/fifo-page-replacement>

**References used by the students:**

***// Write references used by you here***

**Rubric-wise marks obtained:**

| **Rubrics** | **Understanding of Shell programming syntax (4)** | | | **Ability to implement program for given problem using Shell**  **script (4)** | | | **Documentation &Timely Submission (2)** | | | **Total (10)** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(4)** | **Avg.**  **(3-2)** | **Poor**  **(1-0)** | **Good**  **(2)** | **Avg.**  **(1)** | **Poor**  **(0)** |  |
| **Marks** |  | | |  | | |  | | |  |