Laboratory Manual

(Cloud Computing Lab-PEC-CSM591C)

Table of Contents

DOS & DONTS IN LABORATORY

DO's

1. Students should be punctual and regular to the laboratory.
2. Students should come to the lab in-time with proper dress code.
3. Students should maintain discipline all the time and obey the instructions.
4. Students should carry observation and record completed in all aspects.
5. Students should be at their concerned experiment table, unnecessary moment is restricted.
6. Students should follow the indent procedure to receive and deposit the components from lab technician.
7. While doing the experiments any failure/malfunction must be reported to the faculty.
8. Students should check the connections of circuit properly before switch ON the power supply.
9. Students should verify the reading with the help of the lab instructor after completion of experiment.
10. Students must endure that all switches are in the lab OFF position, all the connections are removed.
11. At the end of practical class the apparatus should be returned to the lab technician and take back the indent slip.
12. After completing your lab session SHUTDOWN the systems, TURNOFF the power switches and arrange the chairs properly.
13. Each experiment should be written in the record note book only after getting signature from the lab in charge in the observation notebook.

DON'Ts

1. Don't eat and drink in the laboratory.
2. Don't touch electric wires.
3. Don't turn ON the circuit unless it is completed.
4. Avoid making loose connections.
5. Don't leave the lab without permission.
6. Don't bring mobiles into laboratory.
7. Do not open any irrelevant sites on computer.
8. Don't use a flash drive on computers.

EXPERIMENTNo-1

Aim:

To set up a virtualization environment using software like VirtualBox or VMware Workstation and install different flavors of Linux distributions as guest operating systems on a Windows 7 or 8 host.

Learning Outcomes:

1. **Understanding Virtualization:** Gain familiarity with the concept of virtualization, its benefits, and how virtual machines work.
2. **Software Installation Skills:** Learn to install and configure VirtualBox or VMware Workstation on a Windows 7 or 8 host OS.
3. **Operating System Installation:** Gain experience in installing various Linux distributions (different flavors) as guest operating systems within the virtual environment.
4. **Configuration and Management:** Understand basic configuration settings for virtual machines, such as allocating resources (CPU, RAM, disk space) and network configurations.
5. **Troubleshooting Skills:** Develop troubleshooting skills to address common issues encountered during the installation and setup process.

Prerequisites:

1. **Virtualization Support:** Ensure that the host computer's CPU supports virtualization technology (Intel VT-x or AMD-V). Enable virtualization in the BIOS/UEFI settings if it's disabled.
2. **Sufficient Resources:** Adequate hardware resources such as CPU, RAM, and disk space to host virtual machines and different operating systems simultaneously.
3. **Virtualization Software:** Download and install either VirtualBox or VMware Workstation on the Windows 7 or 8 host operating system.
4. **ISO Images of Linux Distributions:** Obtain ISO images of various Linux distributions (e.g., Ubuntu, Fedora, CentOS, Debian, etc.) that you intend to install as guest operating systems.

Equipment Needed:

1. **Computer or Laptop:** The host system (Windows 7 or 8) where VirtualBox or VMware Workstation will be installed.
2. **Internet Connection:** Needed for downloading virtualization software and Linux ISO images.

3. **External Storage (Optional):** Additional storage to save ISO images or backups of virtual machines.

Theory :

1. **Virtualization Concept:** Understanding the basics of virtualization, the concept of host and guest operating systems, virtual machine configurations, and resource allocation.
2. **Hypervisor:** Awareness of the role of hypervisors like VirtualBox or VMware Workstation in managing virtual machines and their configurations.
3. **Guest Additions/Tools:** Familiarity with guest additions/tools (e.g., VirtualBox Guest Additions or VMware Tools) that enhance the functionality and performance of guest operating systems.
4. **Operating System Installation Process:** Knowledge of the installation process for different Linux distributions, including disk partitioning, configuring user accounts, and post-installation setup.
5. **Networking Configuration:** Basic understanding of networking concepts to configure network settings within virtual machines and enable communication between the host and guest systems.

Procedure :

1. **Virtualization Software Installation:**
   - Download and install VirtualBox or VMware Workstation on the Windows 7 or 8 host operating system.
2. **Linux Distribution Selection:**
   - Choose different flavors or distributions of Linux (e.g., Ubuntu, Fedora, CentOS, Debian, etc.) that you want to install as guest operating systems.
3. **Creating Virtual Machines:**
   - Open the virtualization software (VirtualBox or VMware Workstation).
   - Create new virtual machines for each Linux distribution you want to install.
   - Configure settings such as memory, storage, and network settings for the virtual machines.
4. **Installing Linux on Virtual Machines:**
   - Mount the ISO image of the chosen Linux distribution as a virtual CD/DVD drive for the virtual machine.
   - Start the virtual machine and follow the installation process for the selected Linux distribution.
   - Complete the installation by configuring user accounts, language settings, and other preferences as required.
5. **Post-Installation Configuration:**

- Install necessary drivers or tools within the Linux guest OS to enhance functionality (e.g., VirtualBox Guest Additions or VMware Tools).
- Configure networking settings to enable communication between the host and guest operating systems.

6. **Testing and Validation:**
   - Verify that the installed Linux distributions function properly within the virtual environment.
   - Check networking, display resolutions, and other functionalities.
7. **Documentation and Troubleshooting:**
   - Document the installation steps and configurations made for each Linux distribution.
   - Troubleshoot and resolve any issues encountered during the setup process.

Precautions:

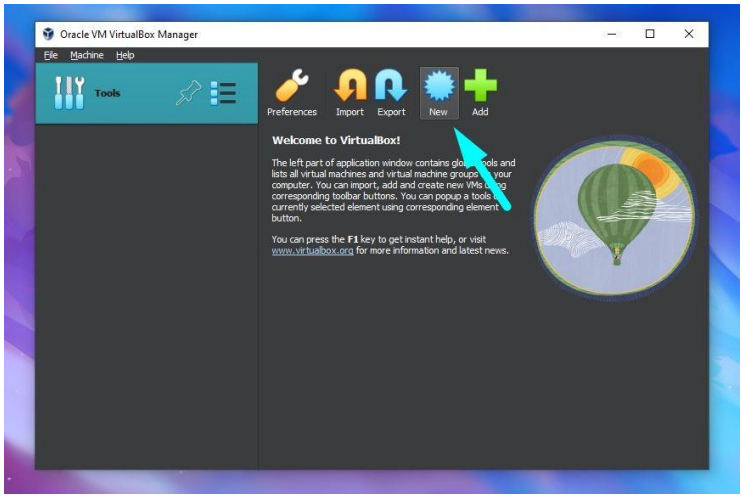
1. **Backup Data:** Before installation, back up essential data on the Windows 7 or 8 host machine to prevent potential data loss during the installation process.
2. **System Requirements Check:** Ensure that the host system meets the minimum hardware requirements to run the virtualization software and multiple virtual machines simultaneously.
3. **Enable Virtualization:** Make sure virtualization support is enabled in the BIOS/UEFI settings of the host system to enable efficient virtualization operations.
4. **Download from Official Sources:** Obtain virtualization software (VirtualBox or VMware Workstation) and Linux ISO images from official and reputable sources to prevent security risks or installation issues from unauthorized sources.
5. **Allocate Adequate Resources:** Allocate appropriate resources (CPU cores, RAM, disk space) to each virtual machine based on the requirements of the Linux distributions being installed.

Observations:

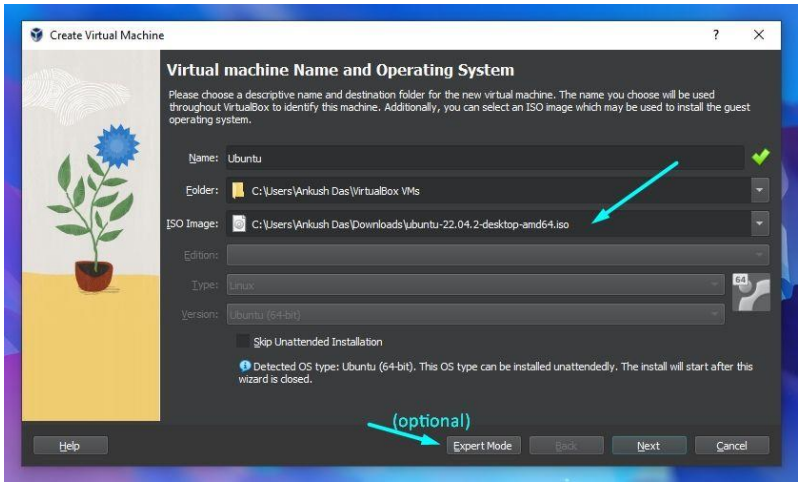
1. **Installation Process:** Observe the installation process of VirtualBox or VMware Workstation on the Windows 7 or 8 host system for any errors or warnings during the installation.
2. **Virtual Machine Creation:** During the creation of virtual machines for different Linux distributions, note any configuration settings or specific requirements needed for each distribution.
3. **Linux Installation:** Monitor the installation of each Linux distribution within its respective virtual machine, checking for any errors or anomalies during the installation process.
4. **Guest Additions/Tools:** Observe the installation and functionality of guest additions or tools provided by the virtualization software to enhance the performance and integration of the guest OS with the host system.

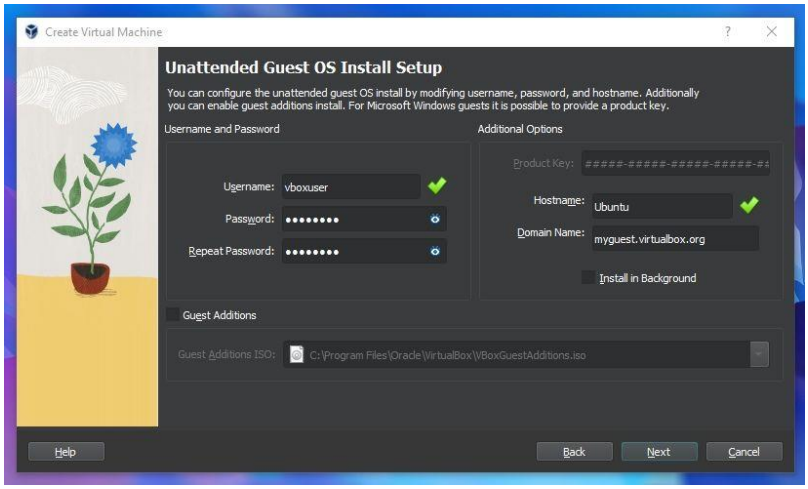
Results:

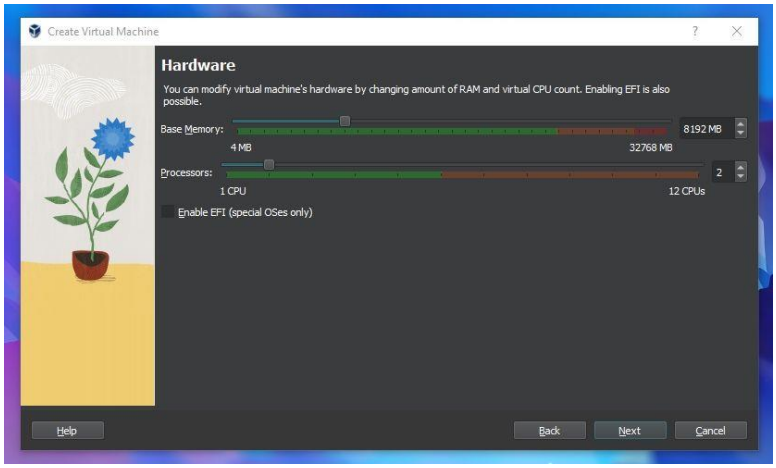
1. **Successful Installation:** The virtualization software (VirtualBox or VMware Workstation) installs smoothly on the Windows 7 or 8 host system without any errors or issues.
2. **Virtual Machine Creation:** Virtual machines are successfully created for different flavors of Linux, allocating resources as per requirements without any errors.
3. **Linux Installation Completion:** The installation of various Linux distributions within their respective virtual machines completes without encountering significant errors or failures.
4. **Functionality Testing:** After installation, the Linux distributions should function properly within the virtual environment, demonstrating essential functionalities such as networking, display, and software installation without major issues.
5. **Documentation and Troubleshooting:** Any observed issues or errors should be documented for troubleshooting purposes. If encountered, troubleshoot and resolve issues related to guest OS functionality, network connectivity, or software compatibility.

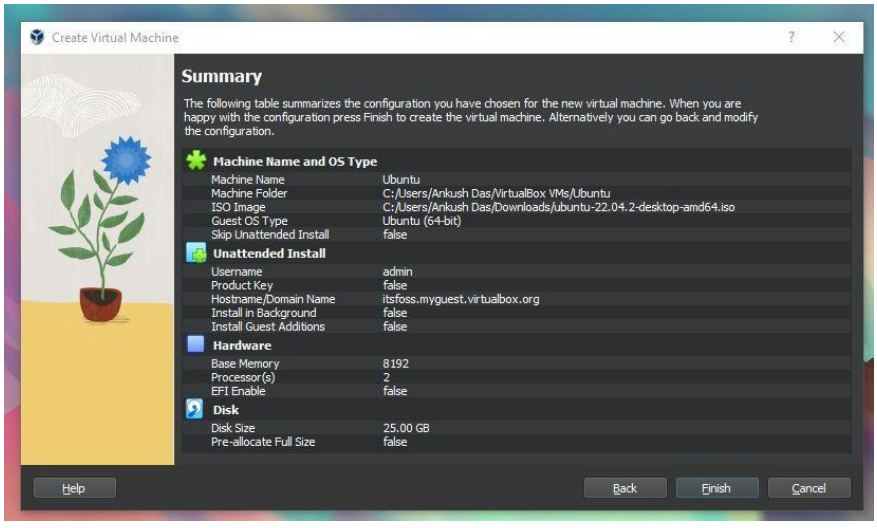


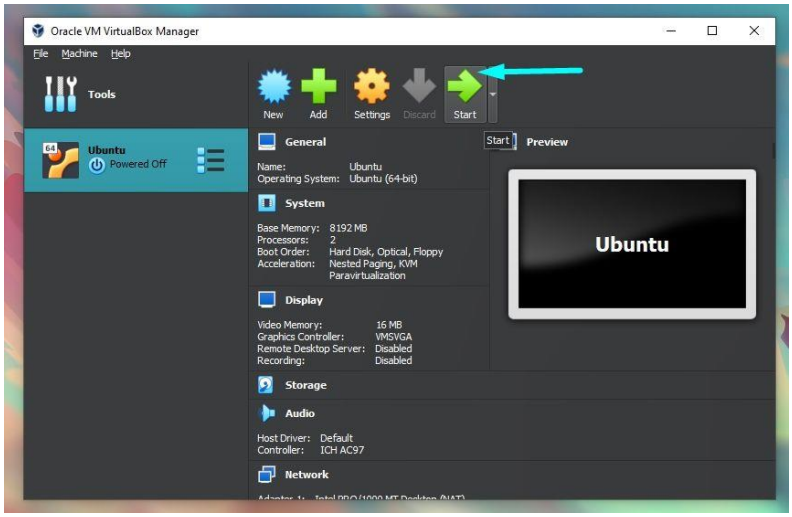
Users can ensure a successful setup of VirtualBox or VMware Workstation with different Linux flavors on Windows 7 or 8 host systems within a virtualized environment

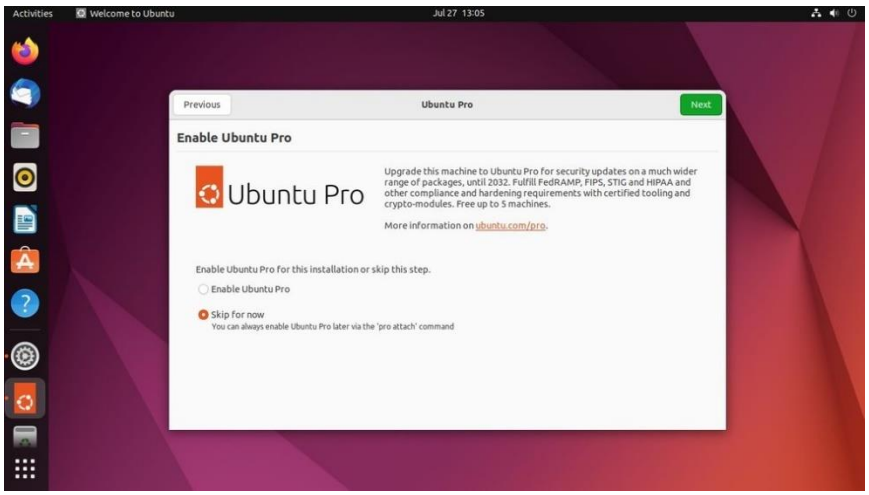



Create Virtual Machine
Unattended Guest OS Install Setup
You can configure the unattended guest OS install by modifying username, password, and hostname. Additionally you can enable guest additions install. For Microsoft Windows guests it is possible to provide a product key.
Username and Password
Additional Options
Username: vboxuser
Password:
Repeat Password:
Product Key:
Hostname: Ubuntu
Domain Name: myguest.virtualbox.org
Install in Background
Guest Additions
Guest Additions ISO: C:\Program Files\Oracle\VirtualBox\VBoxGuestAdditions.iso
Help
Back
Next
Cancel


Create Virtual Machine
Hardware
You can modify virtual machine's hardware by changing amount of RAM and virtual CPU count. Enabling EFI is also possible.
Base Memory:
8192 MB
4 MB
32768 MB
Processors:
2
1 CPU
12 CPUs
Enable EFI (special OSes only)
Help
Back
Next
Cancel


Create Virtual Machine
Summary
The following table summarizes the configuration you have chosen for the new virtual machine. When you are happy with the configuration press Finish to create the virtual machine. Alternatively you can go back and modify the configuration.
Machine Name and OS Type
Machine Name Ubuntu
Machine Folder C:/Users/Ankush Das/VirtualBox VMs/Ubuntu
ISO Image C:/Users/Ankush Das/Downloads/ubuntu-22.04.2-desktop-amd64.iso
Guest OS Type Ubuntu (64-bit)
Skip Unattended Install false
Unattended Install
Username admin
Product Key false
Hostname/Domain Name itsfoss.myguest.virtualbox.org
Install in Background false
Install Guest Additions false
Hardware
Base Memory 8192
Processor(s) 2
EFI Enable false
Disk
Disk Size 25.00 GB
Pre-allocate Full Size false
Help
Back
Finish
Cancel


Oracle VM VirtualBox Manager
File Machine Help
Tools
Ubuntu
Powered Off
New Add Settings Discard Start
General
Name: Ubuntu
Operating System: Ubuntu (64-bit)
System
Base Memory: 8192 MB
Processors: 2
Boot Order: Hard Disk, Optical, Floppy
Acceleration: Nested Paging, KVM
Paravirtualization
Display
Video Memory: 16 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled
Storage
Audio
Host Driver: Default
Controller: ICH AC97
Network
Start
Preview
Ubuntu


Activities
Welcome to Ubuntu
Jul 27 13:05
Previous
Ubuntu Pro
Next
Enable Ubuntu Pro
Ubuntu Pro
Upgrade this machine to Ubuntu Pro for security updates on a much wider range of packages, until 2032. Fulfill FedRAMP, FIPS, STIG and HIPAA and other compliance and hardening requirements with certified tooling and crypto-modules. Free up to 5 machines.
More information on ubuntu.com/pro.
Enable Ubuntu Pro for this installation or skip this step.
Enable Ubuntu Pro
Skip for now
You can always enable Ubuntu Pro later via the 'pro attach' command

EXPERIMENT No-2

Aim: To install a C/C++ compiler on a virtual machine and execute a sample program successfully.

Learning Outcomes:

- Understand the process of setting up a virtual machine environment.
- Gain proficiency in installing software packages within a virtual environment.
- Learn how to use a command-line interface to compile and execute C++ programs.
- Gain familiarity with basic C++ programming constructs.

Prerequisites:

- **Virtual Machine Software**: Choose and install virtual machine software like VirtualBox, VMware, or Hyper-V.
- **Operating System**: Select an operating system to install within the virtual machine (e.g., Ubuntu, CentOS, Windows, etc.).
- **Basic Computer Skills**: Familiarity with navigating an operating system and using a command-line interface.

Equipment Needed:

- Computer or Laptop: Host machine to run the virtualization software.
- Internet Connection: Required for downloading the necessary software packages.
- Sufficient Hardware Resources: Adequate CPU, RAM, and disk space allocation for the virtual machine..

Theory:

- Virtual Machines (VMs): A virtual machine is an emulation of a physical computer system that runs within a host machine. It allows multiple operating systems to run simultaneously on a single physical machine.

- C++ Compiler:

- Compiler: A compiler is a software tool that translates high-level programming languages (like C++) into machine-readable code.
- C++ Compiler Options: g++ (GCC - GNU Compiler Collection) is a popular choice for C++ compilation in many Unix-like systems. It translates C++ source code files into executable binaries.

PROCEDURE:-

Steps to Install a C++ Compiler and Execute a Sample Program:

- Set up a Virtual Machine:
- Install Virtual Machine software (e.g., VirtualBox).
- Create a new virtual machine, allocate resources (CPU, RAM, storage), and install an operating system (e.g., Ubuntu).
- Installing C++ Compiler:

- Open a terminal within the virtual machine.
- Update the package repository using commands **sudo apt-get update** (for Ubuntu).
- Install the C++ compiler (g++ for GCC) using commands **sudo apt-get install g++.**
- Writing a Sample C++ Program:

- Use a text editor within the virtual machine (e.g., **nano, vim, or a gedit**)
  nano hello.cpp
- Save and Exit the Editor: Save the file (Ctrl + O in **nano,** or according to the editor's instructions) and exit (Ctrl + X in nano).
- Compile the C++ Program: In the terminal, navigate to the directory where the hello.cpp file is saved and compile the program using g++: **g++ hello.cpp -o hello**.
- Execute the compiled program (./sample) in the terminal. **./hello**

Observations:

- You should see "Hello, World!" printed in the terminal, confirming the successful execution of the sample program.
- Following these steps will enable you to install a C++ compiler in a virtual machine, write a basic C++ program, compile it using g++, and execute the compiled program within the virtual environment.

RESULT:- The successful execution of the program will result in the output:

Hello, World!

Confirmation that "Hello, World!" is displayed in the terminal confirms the successful execution of the compiled C++ program.

No errors or warnings during the installation, compilation, or execution processes indicate a successful setup.

PRECAUTIONS:-

- Allocate Sufficient Resources
- Select a Stable VM Environment:
- Update the System:
- Install from Trusted Sources
- Verify Package Authenticity
- Regularly Update and Maintain the System

.

EXPERIMENTNo-3

Aim: To execute a sample programs successfully.

Learning Outcome:
Understand the basics of C programming language, familiarize with compiling and executing C programs in a virtual machine environment.

Prerequisite:
Basic understanding of programming concepts, familiarity with the C programming language syntax, and access to a virtual machine software (e.g., VirtualBox, VMware) installed on your system.

Equipment Required:
- Computer with a virtual machine software installed
- Operating system image compatible with your virtual machine software
- Integrated development environment (IDE) or a text editor for writing C code (e.g., Code::Blocks, Visual Studio Code, Sublime Text)

Theory:
C is a powerful procedural programming language known for its efficiency and flexibility. Programs are written using a set of syntax rules to perform specific tasks.

Steps to Execute a Sample C Program:

1. **Open your preferred text editor or IDE within the virtual machine.**
2. **Write a simple C program (e.g., a "Hello, World!" program) in the editor.**
3. **Save the file with a ".c" extension (e.g., "hello.c").**
4. **Open a terminal or command prompt in the virtual machine.**
5. **Navigate to the directory where the C file is saved.**
6. **Compile the C program using a C compiler (e.g., gcc hello.c -o hello).**
7. **If compilation is successful, execute the compiled program (e.g., ./hello).**
8. **Observe the output displayed on the terminal.**

a) Write a program in C to add two numbers.

```
#include <stdio.h>
void main (){
int a, b;
printf (“Enter two numbers:”);
scanf (“%d %d”, &a, &b);
printf (“The sum is %d”, a+b);
}
```

Output:

```
root @ kali :~ /Desktop # make1
cc 1.c -01
root @ kali :~ /Desktop # .11
Enter two numbers: 4
9
The sum is 13 root @ kali :~ /Desktop
```

b) write a program in C to swap two numbers.

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


```
void  main (){
int a=10, b=20;
printf ("Before swap a=%d, b=%d", a,b);
a=a+b;
b=a-b;
a=a-b;
printf ("\n After swap a=%d, b=%d\n", a, b);
}
```

Output:

```
root  @ kali :~ /Desktop #  make2
cc   2.c   -02
root  @ kali :~ /Desktop # .12
Before swap a=10, b=20
After swap a=20, b=10
root  @ kali :~ /Desktop #
```

3. write a program in C to find the factorial of a given number.

```
#include <stdio.h>
void  main (){
int n, i ;
unsigned long fact =1;
printf ("Enter an integer:");
scanf ("%d", &n);
if (n <0)
printf("Error! Fact of negative");
else {
for(i=1; i<=n; ++i){
fact= i; }
printf ("%d\n", fact);
}
}
```

Output:

```
root  @ kali :~ /Desktop #  make2
cc   3.c   -03
root  @ kali :~ /Desktop # .13
Enter an integer: 5
5! = 120
```

Observations:

- Understanding and writing C code syntax.
- Compiling the code using a C compiler.
- Executing the compiled code within the virtual machine environment.
- Observing the output or behavior of the executed code.

Result:

Successful execution of sample C programs within the virtual machine environment, displaying expected output or behavior as defined in the code.

Precautions:

1. **Backup:** Always back up your code and important data before working in a virtual machine.
2. **Resource Allocation:** Allocate sufficient resources (CPU, memory) to the virtual machine for smooth execution.
3. **Code Review:** Review your code for syntax errors or logical mistakes before compilation.
4. **Compiler Compatibility:** Ensure the C compiler you use within the virtual machine is compatible with the code syntax.
5. **Virtual Machine Snapshots:** Consider taking snapshots of your virtual machine at key points to revert back in case of unexpected issues

EXPERIMENTNo-4

Aim:

Create a GitHub account to collaborate on coding projects, share code repositories, and contribute to open-source projects.

Learning Outcome:

Understand the process of setting up a GitHub account, navigating the GitHub platform, and utilizing basic features for code management and collaboration.

Prerequisite:

Access to the internet, a valid email address for verification, and a web browser to access the GitHub website.

Equipment Required:

- **Computer or mobile device**
- **Internet connection**
- **Web browser**

Theory:

GitHub is a web-based platform used for version control using Git. It allows users to collaborate on projects by hosting and managing Git repositories. Users can share code, track changes, and contribute to various projects.

Procedure:

Step 1: go to **https://github.com/join** in web browser
Step 2: Enter the required personal details like username, email ADDRESS, and password
Step 3: click verify to start the verification process
Step 4: After verification click on the green **create account** button.
Step 5: Then you will receive a verification code in your gmail, Put that code in verification page.
Step 6: Select your preference if you want to and then click continue or just press the skip personalization.
Step 7: Note the types of plan by github. (Free, Pro, Team, Enterprise).
Step 8: After selecting the plan, your account will be created.

6. Creating a Hello world python code in github.
Step 1: Create a new repository and select owner followed by repository name.
Step 2: Select the visibility (Public or Private) then click on create repository.
Step 3: Now click on the **Add file** and select **Create new file** .
Step 4: Enter file name followed by **.py** and write the python code.
Step 5: Now click on the **omit changes** .
Step 6: Open the repository an d click on code and copy the **HTTPS clone** link.

7. Set up google cloud console.
Step 1: Go to **console.cloud.google.com** and login with your google account.

Step 2: Create a new project an d give it a name.

8. Run Python code from Git repository.
Step 1: In the google cloud console, navigate to "**cloud shell**" from the top right.
Step 2: clone your Github repository using **git clone https: // github.com / Di-3am / Prog1.git**
Step 3: Navigate your project folder using **cd Prog1 .**
Step 4: Run python code using python **Hello.py**

Observations:

- Account creation process: providing personal details, username, and email verification.
- Navigation of the GitHub interface: exploring repositories, profiles, and settings.
- Understanding basic GitHub functionalities: repository creation, forking, cloning, and pushing code changes.

Result:
Successful creation of a GitHub account with a unique username and email verification. Access to GitHub's interface for managing repositories and contributing to projects.

Precautions:

1. Username Choice: Choose a unique and appropriate username.
2. Password Security: Use a strong and secure password for the GitHub account.
3. Email Verification: Verify the email address associated with the account for security and account recovery purposes.
4. Privacy Settings: Review and set privacy settings according to preferences.
5. Two-Factor Authentication (2FA): Consider enabling 2FA for added security.
6. Public/Private Repositories: Be cautious when creating public or private repositories and ensure they are set to the intended visibility

EXPERIMENTNo-5

Aim:

Create a Python program hosted on GitHub that calculates the factorial of a given number using recursion.

Learning Outcomes:

Understanding and implementing recursive functions in Python, utilizing GitHub for version control and code sharing, and creating a repository for a Python project.

Prerequisite:

- Basic understanding of Python programming language.
- Access to a computer with Python installed.
- A GitHub account created (as outlined in the previous GitHub account creation process).

Equipment:

- Text editor or integrated development environment (IDE) for writing Python code
- Internet connection to access GitHub

Theory:

Recursion is a programming technique where a function calls itself to solve a problem. In this case, a Python function will recursively calculate the factorial of a number by calling itself until a base case is reached.

Procedure:

Steps for Creating Python Code to Find Factorial Using Recursion on GitHub:
Step 1: Create a new repository and select owner followed by repository name.
Step 2: Select the visibility (Public or Private) then click on **Create repository.**
Step 3: Now click on **Add file** and select **Create new file.**
Step 4: Enter file name followed by **.py** and write the python code in the editor

```
def recur_fact(n):
    if n==1:
      return n
    Else :
      return n*recur_fact (n-1)
num=int(input("Enter the number:"))
if num<0:
    print ("Factorial not exist.")
elif num==0:
    print ("0!=1")
else :
    print (num,"!=", recur_fact(num))
```

Step 5: Now click on **commit changes**.

Set up google cloud console.
Step 1: Go to **console.cloud.google.com** and log in with your google account.
Step 2: Create a new project and give it a name.

- **Run python code from Git repository**.

Step 1: In GCC, navigate to **cloud shell** from the top right.
Step 2: clone your github repository using
git clone https://github.com/Di-3am/Prog2.git
Step 3: Navigate to your project folder using
cd Prog2
Step 4: Run python code using
Python fact.py

Output : Enter the number: 8
8! = **40320**

Observations:

- Writing a Python function that uses recursion to calculate the factorial of a number.
- Creating a GitHub repository to host the Python code.
- Pushing the code to the repository on GitHub.
- Observing the commits, changes, and history within the GitHub repository.

Result:

Successful creation of a Python program that utilizes recursion to find the factorial of a given number, hosted in a GitHub repository. Accessible and shareable code via the GitHub platform**.**

Precautions:

1. Function Efficiency: Ensure the recursive function is optimized and doesn't lead to excessive memory usage for large input numbers.
2. Documentation: Add appropriate comments and documentation to the code for clarity.
3. Testing: Test the function with different inputs to verify its correctness and efficiency.
4. Commit Messages: Use descriptive and meaningful commit messages when pushing changes to the GitHub repository**.**
5. Repository Privacy: Set the repository to public or private based on your intention to share or keep the code private.

EXPERIMENTNo-6

Aim: The aim of installing Google App Engine is to set up a platform for developing, testing, and deploying scalable web applications on Google's infrastructure.

Learning Outcome: Upon completion, learners should be able to:

1. Understand the basics of cloud-based application deployment.
2. Develop and deploy applications on Google's infrastructure using App Engine.
3. Learn how to configure and manage applications using App Engine services.
4. Gain insight into managing and scaling applications in a cloud environment.

Prerequisites:

- Basic understanding of programming languages like Python, Java, PHP, or Go.
- Knowledge of web application development concepts.
- Google account for accessing Google Cloud Platform services.

Equipment Required:

- Computer with an internet connection.
- Web browser.
- Google Account.

Theory: Google App Engine is a Platform as a Service (PaaS) offered by Google Cloud Platform. It allows developers to build and host web applications on Google's infrastructure. App Engine provides auto-scaling and managed services, allowing developers to focus on writing code without worrying about infrastructure management.

Procedure:

1. **Create a Google Cloud Platform (GCP) Account:**
    - Go to the Google Cloud Platform Console (console.cloud.google.com) and sign in or create an account.
2. **Enable Billing Free and Create a Project:**
    - Enable billing Free for your GCP account and create a new project.
3. **Install Google Cloud SDK:**
    - Download and install Google Cloud SDK (software development kit) for your operating system.
4. **Initialize App Engine:**
    - Open a terminal or command prompt and use the **gcloud** command to initialize App Engine within your project.
5. **Develop and Deploy an App:**
    - Write your application code using supported languages (Python, Java, PHP, Go, etc.).
    - Configure your app.yaml or appengine-web.xml file (depending on the language) to define settings for your app.
    - Use the **gcloud app deploy** command to deploy your application to App Engine.
6. **Access the Deployed App:**
    - Once deployed, access your application using the provided URL.

Observations:

- During the deployment process, observe the logs and messages in the terminal/command prompt for any errors or warnings.
- Monitor the deployment progress on the Google Cloud Platform Console.

Result: Upon successful installation and deployment, you will have a web application hosted on Google App Engine that can be accessed via its URL.

Precautions:

- Ensure proper configuration of access controls and security settings.
- Regularly monitor resource usage and billing to avoid unexpected costs.
- Back up your application code and data to prevent data loss.
- Follow best practices for securing your application and managing dependencies.

EXPERIMENT No-7

Aim: The aim is to create and deploy a basic "Hello World" web application using Python on Google App Engine.

Learning Outcomes:

Upon completion, learners should be able to:

- Understand the basics of creating web applications using Python in Google App Engine.
- Develop simple web applications and deploy them using Google App Engine.
- Learn how to configure and structure a basic web application project.
- Gain familiarity with deploying and testing applications on the Google App Engine platform.

Prerequisites:

- Basic knowledge of Python programming.
- Google Account for accessing Google Cloud Platform services.
- Install Google Cloud SDK on your computer.

Equipment Required:

- Computer with an internet connection.
- Web browser.
- Google Account.
- Text editor or Integrated Development Environment (IDE) for Python (e.g., VSCode, PyCharm).

Theory:

Google App Engine supports various programming languages, including Python. It provides a scalable and managed platform for deploying web applications without worrying about infrastructure management.

Procedure:

- Set Up Google Cloud Platform:
- Create a Google Cloud Platform (GCP) account and enable billing.
- Install Google Cloud SDK on your computer.
- Create a Basic Python Web Application:
- Create a directory for your project.
- Inside the project directory, create an app.yaml file. This file configures the app's settings.
- Write a simple Python web application code using a framework like Flask or Django.
- Deploy the Application:
- Use the Google Cloud SDK's command-line interface (gcloud) to deploy the application.
- Execute the command: gcloud app deploy in the project directory.
- Follow the prompts to deploy your application to Google App Engine.
- Test the Deployed Application:
- Once the deployment process is completed, access your web application using the provided URL.
- Verify that the "Hello World" or the intended functionality is working as expected.

Observations:

- Observe the terminal/command prompt during deployment for any errors or warnings.
- Check the logs and status on the Google Cloud Platform Console for deployment progress and potential issues.
- Result: Upon successful deployment, you'll have a basic "Hello World" web application running on Google App Engine that can be accessed via its URL.
- Precautions:
- Secure sensitive information (such as API keys, passwords) within your application.
- Regularly back up your code and data.
- Monitor resource usage and billing to avoid unexpected costs.
- Follow best practices for application development and security.

EXPERIMENT No-8

Aim:

The aim is to use the Google App Engine Launcher to manage, test, and deploy web applications on the Google App Engine platform.

Learning Outcomes:

Upon completion, learners should be able to:

- Understand the basics of using the Google App Engine Launcher.
- Launch, manage, and test web applications locally.
- Deploy web applications to Google App Engine using the GAE Launcher.
- Gain practical experience in deploying and managing applications on App Engine.

Prerequisites:

- Install Google App Engine Launcher (part of Google App Engine SDK).
- Basic knowledge of web application development.
- A web browser for testing deployed applications.
- Google Account for accessing Google Cloud Platform services.

Equipment Required:

- Computer with an internet connection.
- Web browser.
- Google Account.
- Google App Engine Launcher installed on your computer.
- Theory: Google App Engine Launcher provides a graphical user interface (GUI) that simplifies the process of managing and deploying web applications on Google App Engine. It allows users to manage projects, test applications locally, and deploy them to the App Engine platform.

Procedure:

- Install Google App Engine Launcher:
- Download and install Google App Engine SDK, which includes the Launcher.
- Create a New Project:
- Open GAE Launcher and create a new project.
- Configure project settings such as the application ID and runtime environment.
- Develop and Test Locally:
- Write your web application code (using Python, Java, PHP, Go, etc.).
- Run the application locally using the Launcher to test its functionality.
- Deploy the Application:
- Use the GAE Launcher to deploy your application to Google App Engine.
- Enter your Google Account credentials.
- Follow the prompts to deploy your application.

Observations:

- Monitor the GAE Launcher interface during the deployment process for any status messages or errors.
- Check the logs for any warnings or errors encountered during deployment.
- Result: Upon successful deployment, your web application will be live and accessible through the provided App Engine URL.
- Precautions:
- Ensure your project configurations and settings are accurate before deployment.
- Check for any dependencies or environment-specific issues that might arise during deployment.
- Keep your Google Account credentials secure and do not share them

EXPERIMENT No-9

Aim:

The aim is to simulate a cloud computing environment using CloudSim to understand and analyze the behavior and performance of cloud-based systems.

Learning Outcomes:

Upon completion, learners should be able to:

- Understand the principles and components of cloud computing.
- Gain hands-on experience in simulating cloud scenarios using CloudSim.
- Analyze and evaluate the performance of various cloud computing models.
- Learn about resource allocation, scheduling, and scalability in cloud environments.

Prerequisites:

- Basic understanding of cloud computing concepts.
- Familiarity with Java programming (as CloudSim is primarily implemented in Java).
- Knowledge of distributed systems and simulation basics would be beneficial.
- Equipment Required:
- Computer with Java Development Kit (JDK) installed.
- Internet access for downloading CloudSim libraries and documentation.
- Text editor or Integrated Development Environment (IDE) for Java development.

Theory:

CloudSim provides a comprehensive simulation framework for modeling cloud environments. It enables the creation of a simulated cloud infrastructure with various components such as data centers, hosts, virtual machines (VMs), and applications. Users can define scenarios, workload patterns, and resource allocation policies to observe how the system behaves under different conditions.

Procedure:

Step 1: Download and extract cloud sim 3.0.3 from Github **https://github.com/clouds lab/cloud sim/releases/tag/cloud sim-3.0.3**
Step 2: Open terminal and write **$sudo apt** to install ant.
Step 3: From cloud sim folder open **Example.txt** file
Step 4: copy line number 43 to compiling and paste it in the terminal and change the version to 3.0.3

javac-classpath jars/cloudsim/3.0.3.jar: examples/org/cloudbus/cloudsim/examples/cloud sim Example1.java

Step 5: Now copy line no 44 for running the code

java-clsspath jars/cloudsim-3.0.3.jar: examples org.cloud.bus.cloudsim.examples.cloudsim Examples

OUTPUT :
Simulation finished

Cloudlet ID	STATUS	Data centre ID	VM ID	Time	Start time	Finish Time
0	SUCCESS	2	0	400	0.1	4001

Cloudsim Example1 finished !

Observations:

During the simulation using CloudSim, users can observe:
Performance metrics like response time, throughput, and resource utilization.
The behavior of different scheduling and allocation algorithms.
Impact of varying workloads on the cloud infrastructure.

Result:

Upon completion of the simulation:
Users gain insights into how cloud-based systems perform under specified conditions.
Performance metrics and analysis help in evaluating the efficiency of resource allocation policies and system configurations.

Precautions:

Ensure proper understanding of the CloudSim framework and its APIs to set up the simulation accurately.
Use appropriate parameters and workload models to simulate realistic scenarios.
Avoid overloading the simulation with unnecessary complexity initially; start with simpler scenarios and gradually increase complexity.
Verify the simulation results by comparing them with expected outcomes and known cloud computing principles.

EXPERIMENT No-10

Aim:

The aim is to implement and incorporate a custom scheduling algorithm into CloudSim to observe its behavior and performance within a simulated cloud computing environment.

Learning Outcomes:

Upon completion, learners should be able to:
Understand the principles of scheduling in cloud computing.
Gain hands-on experience in implementing a custom scheduling algorithm.
Integrate the custom algorithm into the CloudSim simulation framework.
Analyze and evaluate the performance of the custom scheduling algorithm under various conditions.

Prerequisites:

Proficiency in Java programming language.
In-depth understanding of cloud computing concepts, particularly scheduling algorithms.
Familiarity with CloudSim simulation framework and its APIs.
Access to CloudSim libraries and related documentation.

Equipment Required:

Computer with Java Development Kit (JDK) installed.
Text editor or Integrated Development Environment (IDE) for Java development.
Internet access to reference additional resources and documentation.

Theory:

CloudSim provides various built-in scheduling algorithms for simulating cloud environments. To run a scheduling algorithm not present in CloudSim, you'll need to create a new class implementing your custom algorithm logic. The implementation typically involves defining scheduling policies based on specific criteria like task prioritization, resource allocation, etc. The newly created algorithm should then be integrated into the CloudSim framework to be used within the simulation.

Procedure:

Step 1: Follow the steps from 1 to 5 from assignment 5.
Step 2: Open **github .com/Koushal2001**
Step 3: copy the data centre
Broker.java -> Paste on -> Open sources on cloud sim folder -> org ->? Cloud bus -> cloud sim -> paste a Data centre Broker.java
Step 4: Make a new file **simulation.java** & do the same as above.

Step 5: Now open terminal in the same directory and compile the cloud using **javac-classpath jars/cloudsim-**

3.0.3.jar:exaples/org/cloudbus/cloudsim/examples/simulation.java

Step 6: Run

OUTPUT :

Cloudlet ID	STATUS	Data centre ID	VM ID	Time	Start Time	Finish Time	User ID
8	SUCCESS	3	8	1.01	0.2	1.21	4
5	SUCCESS	2	5	1.67	0.2	1.87	4
7	SUCCESS	3	7	2.19	0.2	2.4	4
0	SUCCESS	2	0	2.2	0.2	2.4	4
9	SUCCESS	3	9	2.3	0.2	2.5	4
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
39	SUCCESS	3	9	1.94	6.87	8.82	4
35	SUCCESS	2	5	2.49	6.6	9.09	4
32	SUCCESS	2	2	1.55	7.69	9.23	4
31	SUCCESS	2	1	2.32	7.02	9.34	4
34	SUCCESS	2	4	2.36	7.8	10.15	4

Simulation finished !

EXPERIMENT No-11

Aim:

The aim is to transfer a text file between different virtual machines securely using the scp command via their IP addresses.

Learning Outcomes:

Upon completion, learners should be able to:
Understand the basics of securely transferring files between remote machines using scp.
Gain familiarity with using IP addresses for communication between virtual machines.
Learn how to authenticate and transfer files securely over a network using scp.

Prerequisites:

Access to multiple virtual machines with SSH (Secure Shell) enabled.
The text file you want to transfer.
Knowledge of the destination virtual machine's IP address and SSH credentials.

Equipment Required:

Two or more virtual machines (VMs) with SSH enabled.
Access to the terminal or command line interface on each VM.
The text file you intend to transfer.

Theory:

scp is a command-line utility used for securely copying files between hosts on a network. It uses SSH for authentication and encryption. When transferring files between virtual machines using scp, you'll need the IP address of the destination machine, the username, and access permissions to the remote system.

Procedure:

Step 1: Open virtual box, go to tools then click on **Network** select **NAT Networks,** click on **create.** Double click on new created **NAT Network,** change the name to **BWU** give IP address **192.168.20.0/24,** check the **Enable DHCP** box and hit **Apply .**

Step 2: Create two virtual machines **User1** and **User2.** Launch both virtual machines.

Step 3: On the top left there is a button **Machine** click on it, next click on **Settings** go to **Network tab,** change the **Attached to** into **NAT Network** automatically **BWU** is selected. Go to **Promiscuous mode** and select **Allow all** and click **OK.**

Do the same step3 for both User1 and User2.

Step 4: Launch User1, right click on Desktop then
Create Document -> text -> plain text, That will create a **.txt** file, rename the file as **hi.txt.** open **hi.txt** and write something then click on **Save.**

Step 5: Open terminal on both User1 and User2. Find the machine IP address by **ifconfig.** User1 has IP address 192.168.20.5 and User2 has 192.168.20.4.
Then write **ping192.168.20.4** in User1 terminal and waiting for connection with User2.

Step 6: Write **cd Desktop** then **ls** to check the available files.

Step 7: Write **scp hi.txt user@192.168.20.4:/home/user**
-> Are you sure you want to continue connecting (Yes/No/[Fingerprint])? Select **Yes.**
-> user@192.168.20.4's password : toor (*default password for parrot OS)

OUTPUT :

Hi.txt	100%	0	0.0KB/S	00.00

Step 8: Open the terminal of User2 write **ls**

OUTPUT :

Documents	**hi.txt**	Pictures	Templates
Desktop	Downloads	Music	Videos

***hi.txt** file is transferred from **User1 to User2** by **scp** command.

Observations:

During the scp command execution, observe the progress of the file transfer in the terminal or command line interface.
Check for any error messages or warnings during the transfer process.

Result:
Upon successful execution of the scp command, the text file will be securely transferred from the source VM to the specified directory on the destination VM using its IP address.

Precautions:

Ensure that SSH is properly configured and enabled on both source and destination VMs.
Verify the IP address and login credentials of the destination VM to avoid connection errors.
Protect sensitive data during transfer by using secure and encrypted connections.
Double-check file paths and permissions to avoid unintentional overwriting or access issues.