OSS praticals:

1. Joomla

- Install apache server: apt install apache2
- Restart it: systemctl start apache2
- Install php modules :
  apt install php php-xml php-mysql php-mbstring php-zip php-soap php-sqlite3 php-curl php-gd php-ldap php-imap php-common

- Install mysql server: apt install mysql-server
- Go to mysql server and do as below:

1. mysql -u root -p
2. Enter password: Rutu@0204
3. create database joomla;
4. use joomla;
5. create user 'Rutu369'@localhost identified by 'Rutuja@02042003'
6. grant all privileges on joomla.* to 'Rutu369'@localhost;
7. flush privileges;
8. exit

- Now change your dir to cd /var/www/
- Create a joomla dir: mkdir joomla
- cd joomla;
- Install joomla in curr dir using below command in terminal:
  wget
  https://downloads.joomla.org/cms/joomla4/4-2-3/Joomla_4-2-3-Stable-Full_Package.zip?format=zip

- unzip Joomla_4-2-3-Stable-Full_Package.zip?format=zip
- Provide necessary permissions to joomla dir
  chown -R www-data. ./
  chmod -R 755 ./

- Edit the config file of joomla
  nano /etc/apache2/sites-available/joomla.conf

  Paste below content in it:
  <virtualhost *:80>
  servername www.rutujajoomla.com
  documentroot /var/www/joomla
  </virtualhost>

- Disable the default site:
  a2dissite 000-default.conf
- Enable the joomla site
  a2ensite joomla.conf

- Enable rewrite mod
  a2enmod rewrite

- Restart the apache server
  systemctl restart apache

- Next go to joomla configuration page -> enter details and configure database by providing details that were used fro mysql database creation
- Checkout the site

2. Bugzilla
https://bugzilla.readthedocs.io/en/latest/installing/quick-start.html
Official document 👍

3. Rpm Package management
Steps:
Take a fedora CONTAINER

Steps:

1. **Install rpm-build package (This package holds the scripts and exe programs to pull the rpm packages using rpm package manager)**
   $yum install rpm-build rpmdevtools -y

2. **Very imp -> move to /root folder first then do following process**
3. **Create the dir structure in curr dir (inside rpmbuild -> create below 5 dir)**
   $rpmbuild
   rpmbuild/
   - **BUILD: (It stores the source code of program whose package is to be built)**
   - **RPMS: (Used to** (Built RPM packages for different architectures))
   - **SOURCES: (stores all tar.gz rpm source files)**
   - **SPECS: (Its a location where we create the rpm package config file)**
   - **SRPMS: (Stores all srpms files for curr rpm package)**
4. **Create a folder at rpmbuild dir level and then 2 files inside it**
   $mkdir format-1.0
   $cd format-1.0
   $nano format.conf -> add random text "this is a config file for rpm package"
   $nano format.txt -> add random text "this is the document file for given rpm"

(Save and quit-> move back the to parent dir)
5. Create the tar.gz version of the dir "format-1.0
   $tar czvf format-1.0.tar.gz format-1.0
   $mv format-1.0.tar.gz rpmbuild/SOURCES/

   (Move this format.tar.gz file to rpm/SOURCES/ dir)

6. Move to rpmbuild/SPECS dir to create a config file for rpm package
   $cd rpmbuild/SPECS/
7. Create the config file “format.spec” using below command
   $rpmdev-newspec format.spec
8. Modify the contents of this file using nano -> This file is useful to create the rpm env
   $nano format.spec
9. Replace the file contents with below content:

```
Name:          format
Version:       1.0
Release:       1.1
Summary:    Testing...
License:       GPL
URL:           https://github.com/rutuja369/Cpp/blob/main/hello.cpp
Source0:       format-1.0.tar.gz
BuildArch:     noarch

%description
my first rpm package

%build
cat > format.sh << EOF
#!/bin/bash
date
cal
EOF

%install
mkdir -p %{buildroot}/usr/bin
mkdir -p %{buildroot}/yahoo
install -m 755 format.sh %{buildroot}/usr/bin/format.sh
cp /root/format-1.0/*  %{buildroot}/yahoo/

%clean
rm -rf %{buildroot}

%files
/usr/bin/format.sh
/yahoo/format.txt
/yahoo/format.conf

%changelog
* Thu Dec 14 2023 Super User
-
```

10. **Run the command to exe this config file**
    $ rpmbuild -ba format.spec
    (here -ba -> build all both bin files and src rpm)

NOTE: after executing this cmd it should end exe with a “exit 0” status

11. **Your new rpm package will be created in rpmbuild/RPMS/noarch/**

12. **Now you need to install this package staying in same dir ->**
    rpmbuild/RPMS/noarch
    $rpm -ivh format-1.0.noarch.rpm

13. **Now check ls /**
    A yahoo dir is created move into it by navigating to cd /
    $ cd /
    $ format.sh (it gives the date and calender format which was mentioned in the config file format.spec)
14. **To checkout your document and src files move to yahoo dir**
    $ cd yahoo
    $ ls
    Checkout the format.conf and format.txt files

4. Debian package management

Step 1: Set Up the Directory Structure

```
calc
└── calculator
    ├── DEBIAN
    │   └── control
    ├── usr
    │   └── bin
    │       └── executablefile
```

Open a terminal and navigate to the location where you want to create the Debian package.

Create a calc folder

Inside this calc folder create another folder , name it DEBIAN

```
# Create a directory named "calculator"
 --mkdir calculator
```

Navigate to the " calculator " directory
--cd calculator

Create a directory named "DEBIAN" inside " calculator "
--mkdir DEBIAN

Step 2: Create the Control File
Now, create the control file inside the DEBIAN directory. You can use a text editor like nano or vim. Here, I'm using nano:

--nano DEBIAN/control

Inside the text editor, add the following control file information:

Package: calculator
Version: 1.0
Section: custom
Priority: optional
Architecture: all
Essential: no
Installed-Size: 1024
Maintainer: ShikhaChoudhari
Description: Display String.

Step 3: Create Additional Directories
Now, create the additional directories required for the Debian package:

Inside "calculator" directory, create "usr" directory
--mkdir -p usr/bin

Step 4: Write a Simple cpp Program
Inside the **usr/bin** directory, you can create a simple cpp program. Let's create a file named **calc**:
--nano usr/bin/calc.cpp

Write a simple cpp program in the editor:

```
#include <iostream>
using namespace std;

int main() {
    char operation;
    float num1, num2;

    cout << "Enter operator (+, -, *, /): ";
    cin >> operation;

    cout << "Enter two numbers: ";
    cin >> num1 >> num2;

    switch (operation) {
        case '+':
```

```
            cout << "Result: " << num1 << " + " << num2 << " = " << num1 +
num2;
            break;
        case '-':
            cout << "Result: " << num1 << " - " << num2 << " = " << num1 -
num2;
            break;
        case '*':
            cout << "Result: " << num1 << " * " << num2 << " = " << num1 *
num2;
            break;
        case '/':
            if (num2 != 0)
                cout << "Result: " << num1 << " / " << num2 << " = " << num1 /
num2;
            else
                cout << "Error! Division by zero is not allowed.";
            break;
        default:
            cout << "Error! Invalid operator.";
            break;
    }

    return 0;
}
```

Save and exit the text editor.

Step 5: Compile and run the cpp Program
Compile the cpp program using the command:
-- g++ calc.cpp -o calc
--./calc

Step 6: Create the Debian Package
Now, it's time to create the Debian package using the **dpkg-deb** command. Ensure that the **build-essential** package is installed on your system:

--sudo apt-get install build-essential

Then, build the Debian package:

--dpkg-deb --build calculator

Step 7: Install the Debian Package
Once the package is built, you can install it on your system:
(here before installing package)

--sudo dpkg -i calculator.deb

Step 8: Run the Program
After installation, you can run your program:

--calc

***Docker: Write a C program to create singly linked list and containerize it.**

I] Singlyll.cpp

```
#include <bits/stdc++.h>
using namespace std;

class Node {
public:
    int data;
    Node* next;
    Node(int val){
        this->data = val;
        this->next = NULL;
    }
};
/* Algorithm to insert element at the head of the list
      1. Create a new node with the data
      2. Make next of new node as head and previous as NULL
      3. Move the head to point to new node
*/

void insertAtHead(Node* &head, int val){
    Node* new_node = new Node(val);
```

```
    new_node->next = head;          //for adding at the start our new node would be new head
    head = new_node;
}

/* Algorithm to insert element after given position
        1. Create a new node with the data
        2. Make next of new node as next of prev node
        3. Make prev node as next of new node
*/

Node* insertAfter(Node* head, int new_val, int position){
    if(position == 0){
        Node* a = new Node(new_val);
        a->data = new_val;
        a->next = head;
        return a;
    }
    else{
        int i;
        Node* a = head;
        for(i = 1; i < position; i++)
            a = a->next;
        Node* tmp = new Node(new_val);
        tmp->data = new_val;
        tmp->next = a->next;
        a->next = tmp;
        return head;
    }
```

```
}


/* Algorithm to insert element at the end of the list
      1. Create a new node with the data
      2. Move the last node to point to new node
      3. Make the new node as last node
*/
void insertAtEnd(Node* &head, int val){
   Node* new_node = new Node(val);


   if(head==NULL){            // if list is empty then new node should be 1st element of the list i.e. it will be the head node.
       head = new_node;
       return;
   }
   Node*  temp = head;       //temp variable to not tamper with the values of head
   while(temp->next != NULL){
       temp = temp->next;
   }
   temp->next = new_node;   // when temp cha next is NULL temp will bew out of while loop n then we set temp cha next to our new node
}


/* Algorithm to search an element in the list
      1. Move the current to point to head
      2. If current is NULL then the element is not present in the list
      3. If current is not NULL then compare the data of current with the element to be searched
      4. If the data is equal then return the current
      5. If the data is not equal then move the current to point to next of current
```

```
*/
bool search(Node* head, int val){
    Node* current = head; // Initialize current
    while (current != NULL)
    {
        if (current->data == val)
            return true;
        current = current->next;
    }
    return false;
}

/* Algorithm to delete an element from the list
      1. Move the current to point to head
      2. If current is NULL then the element is not present in the list
      3. If current is not NULL then compare the data of current with the element to be deleted
      4. If the data is equal then make the next of current as head
      5. If the data is not equal then move the current to point to next of current
*/
void deleteNode(Node* &head, int val){

    Node* temp = head;
    Node* prev = NULL;

    if (temp != NULL && temp->data == val) // If head node itself holds the key to be deleted
    {
        head = temp->next; // Changed head
        delete temp;            // free old head
```

```
        return;
    }
    else{                           // Else Search for the key to be deleted,
        while (temp != NULL && temp->data != val)
        {
            prev = temp;                    //keep track of the previous node as we need to change
'prev->next' */
            temp = temp->next;
        }
        if (temp == NULL)  // If key was not present in linked list
          return;

        prev->next = temp->next;   // Unlink the node from linked list
        delete temp; // Free memory
    }
}

/* Algorithm to print  the list
      1. Move the current to point to head
      2. If current is NULL then the list is empty
      3. If current is not NULL then print the data of current and move the current to point to next of
current
*/
void printList(Node* n)    //to print the linked list
{
    if(n != NULL) {
        cout << n->data << " ";
        printList(n->next);
    }
```

```
}

// Driver code
int main()
{
    Node* head = NULL;
    int choice, data, position;

    while(1){
        cout<<"\n 1.Insert element at the end";
        cout<<"\n 2.Insert element after the :";
        cout<<"\n 3.Insert element at the start";
        cout<<"\n 4.Search for the element in the list";
        cout<<"\n 5.Delete node in a list";
        cout<<"\n 6.Exit";
        cout<<"\n Enter your choice: ";
        cin>>choice;
       if(choice>5){
           break;
       }
       switch(choice){
          case 1:
           cout<<"\n Enter data to be inserted at the end: ";
           cin>>data;
           insertAtEnd(head, data);
           printList(head);
           break;
```

```
case 2:
 cout<<"\n Enter data to be inserted : ";
 cin>>data;
 cout<<"\n Enter index at which element is to be inserted: ";
 cin>>position;
 insertAfter(head, data, position);
 printList(head);
 break;

case 3:
 cout<<"\n Enter data to be inserted at the start: ";
 cin>>data;
 insertAtHead(head, data);
 printList(head);
 break;

case 4:
 cout<<"\n Enter data to be searched : ";
 cin>>data;
 search(head, data)? cout<<"Found" : cout<<"Not found";
 break;

case 5:
 cout<<"\n Enter element to be deleted : ";
 cin>>data;
 deleteNode(head, data);
 printList(head);
```

```
            break;
        }
    }

    return 0;
}
```

II] Dockerfile

```
# Use a lightweight base image
FROM gcc:latest

# Set the working directory inside the container
WORKDIR /app

# Copy the C++ source code into the container
COPY . .

# Compile the C++ code
RUN g++ -o singlyll singlyll.cpp

# Specify the command to run your application
CMD ["./singlyll"]
```

Commands:

```
sudo docker build -t mycppapp .
sudo docker run -it mycppapp
```

Server Configurations:

1. **Telnet:**

https://youtu.be/Mszf9mAY1D8?si=l8NspUDXoLCduFGr

Steps: For Server

1. Install telnet

   $sudo apt-get install xinetd telnetd

2. Edit "/etc/inetd.conf" using nano and add following line in it and save

   telnet stream tcp nowait telnetd /usr/sbin/tcpd /usr/sbin/in.telnetd

3. Now edit "/etc/xinetd.conf" file using nano and add below content in it and save

   instances=60

   log_type =SYSLOG authpriv

   log_on_success= HOST PID

   log_on_failure= HOST

   cps= 25 30

4. Restart the telnet server using

   $sudo /etc/init.d/xinetd restart

Steps: For Client (another pc)

1. Run this command

   $telnet <ip_of_server_pc>

2. Next login using server ka username and password -> done

2. FTP

$ apt-get update

$ apt-get install vsftpd

$ nano /etc/vsftpd.conf

Make the changes in this file

Uncomment–

anonymous_enable=NO

local_enable=YES

write_enable=YES

ascii_upload_enable=YES

ascii_download_enable=YES

Add at last

user_sub_token=$USER

local_root=/home/$USER/ftp

pasv_min_port=10000

pasv_max_port=10100

userlist_deny=NO

$ ufw allow from any port 20,21,10000:10100 proto tcp

Allow traffic from these ports

$s**udo adduser abc** – create a new user set password for it and enter the information for user

$**mkdir /home/abc/ftp-** create a new folder for “abc” user

$ sudo chown nobody:nogroup /home/abc/ftp

$ chmod a-w /home/abc/ftp

$ mkdir /home/abc/ftp/upload

$ sudo chown abc:abc /home/abc/ftp/upload

$echo "My FTP Server" | sudo tee /home/abc/ftp/upload/demo.txt

$ echo "abc" | sudo tee /etc/vsftpd.userlist (creating the userlist and adding abc to it)

$ sudo systemctl restart vsftpd (restarting vsftpd)

Now login with the new created user in another terminal

To login user

$ su - abc

$ftp localhost

Then enter the username and password to login to ftp

$ls

$ put file.txt

$ get file1.txt //it uploaded by server on ftp

Docker Practicals

1. *Docker: Write a python program to perform arithmetic operations and create Docker image accordingly.

```
# arithmetic_operations.py


def add(x, y):
    return x + y


def subtract(x, y):
    return x - y


def multiply(x, y):
    return x * y


def divide(x, y):
    if y != 0:
        return x / y
    else:
        return "Error: Division by zero"


if __name__ == "__main__":
    num1 = float(input("Enter the first number: "))
    num2 = float(input("Enter the second number: "))


    print(f"\nResults:")
```

```
    print(f"Sum: {add(num1, num2)}")
    print(f"Difference: {subtract(num1, num2)}")
    print(f"Product: {multiply(num1, num2)}")
    print(f"Quotient: {divide(num1, num2)}")
```

Dockerfile:

```
# Use an official Python runtime as a parent image
FROM python:3.9

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run arithmetic_operations.py when the container launches
CMD ["python", "arithmetic_operations.py"]
```