

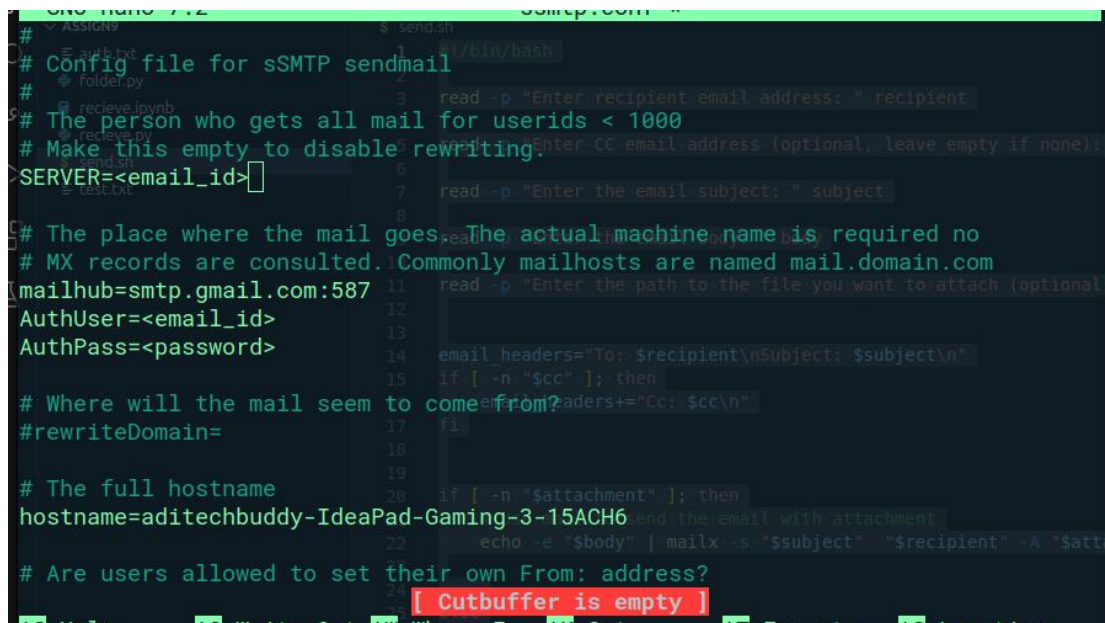
Assignment-10(Email_w_terminal)

Name:Aditya Rathor
Roll No: B22AI044

Q1(Sending mail)

<Intial setup to done after installing ssmtp>: so that it can connect with gmail using port 587 for sending mails

Whereas here it should be ensured that password should be for app password which has to be enabled and extracted from the google account and not the actual email_id's password



```
# Config file for sSMTP sendmail
#
# The person who gets all mail for userids < 1000
# Make this empty to disable rewriting.
SERVER=<email_id>
# The place where the mail goes. The actual machine name is required no
# MX records are consulted. Commonly mailhosts are named mail.domain.com
mailhub=smtp.gmail.com:587
AuthUser=<email_id>
AuthPass=<password>
# Where will the mail seem to come from?
#rewriteDomain=
# The full hostname
hostname=aditechbuddy-IdeaPad-Gaming-3-15ACH6
# Are users allowed to set their own From: address?
Cutbuffer is empty
```

Source code:

```
#!/bin/bash

read -p "Enter recipient email address: " recipient

read -p "Enter the email subject: " subject

read -p "Enter the email body: " body
```

```
read -p "Enter the path to the file you want to attach (optional, leave empty if none):" attachment
```

```
email_headers="To: $recipient\nSubject: $subject\n"
if [ -n "$attachment" ]; then
# Use mail to send the email with attachment
echo -e "$body" | mailx -s "$subject" "$recipient" -A "$attachment"

else
echo -e "$email_headers\n$body" | ssmtp $recipient

fi
```

Created a shell script to send mail using ssmtp (simple mail transfer protocol) client which is used as a send only mail server

Taking input from the user such as the email address, subject, body and attachment(optional).

Mails without attachment can be sent using ssmtp directly but for sending along with an attachment requires special configuration and thus mailx, a command utility tool in Linux, is used

Q2(Receiving Mail)

```
import imaplib
import email
from getpass import getpass
from email.header import decode_header

# Email credentials
email_address = input("Enter your email address: ")
email_password = getpass("Enter your email password: ")
```

```
# IMAP settings for popular email providers
imap_host = "imap.gmail.com" # Gmail
imap_port = 993
```

```
mail = imaplib.IMAP4_SSL(imap_host, imap_port)
try:
mail.login(email_address, email_password)
except:
```

```
print("error during authentication")
```

```
mailbox_choice = input("Which mailbox do you want to view?  
(inbox/spam/sent/starred): ").lower()
```

```
if mailbox_choice == "inbox":  
    mail.select("inbox")  
elif mailbox_choice == "spam":  
    mail.select("[Gmail]/Spam")  
elif mailbox_choice == "starred":  
    mail.select('[Gmail]/Starred')  
else:  
    print("Invalid choice. Viewing Inbox by default.")  
    mail.select("inbox") # Default to Inbox if invalid choice
```

```
num_emails = int(input("How many emails do you want to see? "))
```

```
status, email_ids = mail.search(None, "ALL")
```

```
if status == "OK":  
    email_ids = email_ids[0].split()[-num_emails:] # Get the last 'num_emails' IDs  
    for e_id in reversed(email_ids): # Iterate in reverse to display newest emails first  
        # Fetch the email by ID  
        status, data = mail.fetch(e_id, "(RFC822)")  
        if status == "OK":  
            raw_email = data[0][1]  
            # Parse the raw email using email library  
            msg = email.message_from_bytes(raw_email)  
            # Extract relevant details  
            sender = msg["From"]  
            subject_encoded = msg["Subject"]  
            subject_decoded = decode_header(subject_encoded)[0][0]  
            if isinstance(subject_decoded, bytes):  
                subject = subject_decoded.decode()  
            else:  
                subject = subject_decoded
```

```
        # For simplicity, display first text part of the email as body  
        for part in msg.walk():  
            if part.get_content_type() == "text/plain":  
                body = part.get_payload(decode=True).decode().replace('\r\n', '\n')  
                break  
        else:  
            body = "No text body found"  
        # Display the email details  
        print(f'From: {sender}')
```

```
print(f"Subject: {subject}")
print(f"Body: {body}")
print("-" * 50)
```

```
else:
    print("error during Authentication")
```

```
mail.logout()
```

Created a python script tha retrieve and display emails from an email account using the IMAP (Internet Message Access Protocol) protocol

Using the imaplib and email libraries in Python to connect to the email server, authenticate the user, and fetch emails from selected mailboxes.

Functionality:

Authentication:

The script prompts the user to enter their email address and password or retrieves them from a file (commented out for security reasons).

The getpass library is used to securely input the password without displaying it on the screen.(use app password)

Mailbox Selection and number of emails:

The user is prompted to select a mailbox to view (inbox, spam, sent, starred).

IMAP mailbox names are specific to Gmail in this script, but they can be adjusted for other email providers.

Also asking from user the number of emails it wants to seeon the screen

Email Retrieval:

The script fetches the specified number of emails from the selected mailbox.

It retrieves email IDs using IMAP search and fetches each email's details using IMAP fetch with (RFC822) parameter.

(RFC822) is a parameter used in IMAP (Internet Message Access Protocol) commands to specify the format in which email messages should be fetched.

The email details extracted include sender, subject, and body.

Email Parsing:

The email library is used to parse the raw email data fetched from the server.

The sender's email address, subject (decoded from any encoded format), and the first text/plain part of the email body are extracted and displayed.