MINI PROJECT REPORT

E-MAIL SYSTEM

SUBMITTED IN PARTIAL FULFILMENT OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

by

Jenny Lawrance

Under the guidance of

Mrs.Priya Chandran

Department of Computer Engineering
National Institute of Technology, Calicut

National Institute of Technology, Calicut
Department of Computer Engineering

Certified that this Mini Project Report entitled

E-MAIL SYSTEM

is a bonafide report of the mini project done by

Jenny Lawrance

in partial fulfilment of the degree of
Bachelor of Technology
under our guidance

Mrs.Priya Chandran
Guide
Asst. Professor
Dept.of Computer Engineering

Dr.V.K.Govindan
Professor and
Head
Dept.of Computer Engineering

Acknowledgement

I would like to express my heartfelt gratitude to all those who helped me to bring out this project. Firstly, I would like to thank my guide, Mrs.Priya Chandran, Asst Professor,Department of Computer Science and Engineering,NIT Calicut, for her guidance and co-operation. I also thank Dr.Vineeth Kumar Paleri, Asst. Professor, Department of Computer Science and Engineering, for co-ordinating our mini project. I also acknowledge the advice and guidance given to me by my friends, classmates and seniors.

Jenny Lawrance

Abstract

This is a project implemented in C Language on Linux Platform. The project involves two subsystems, a client and a server. The mail client delivers mail from the client to the server. The server in turn delivers the mail to the appropriate mail box of the intended recipient(s). The underlying protocol implemented is SMTP (Simple Mail Transfer Protocol) which supports text messaging. The client is also provided with authorization facility, to prevent unwanted messages circulating in the system.

Contents

1 Introduction **1**
1.1 Electronic Mail 1
1.2 SMTP 2
1.3 TCP/IP Standards for electronic mail 2

2 Requirements **3**

3 Functionality **3**

4 Implementation **4**

5 Conclusion **5**

6 APPENDIX **6**
6.1 Pseudocode 6
6.1.1 SERVER 6
6.1.2 CLIENT 7

1 Introduction

1.1 Electronic Mail

An electronic mail facility allows the user to send memos across the Internet. Today it is one of the widely used application services. E-mail offers a fast, convenient method of transferring information. Its interesting to note the fact that most users send files with electronic mail than with file transfer protocols. Mail delivery is different from the other conventional network protocols, which is evident from the fact that these protocols use timeout and retransmission for individual segments if no acknowledgement returns. In case of Electronic mail, the system must provide for instances when the remote machine is temporarily unreachable. The user does not want the transfer to abort merely because the destination is temporarily unavailable.

To handle with delayed delivery, mail system use a technique of spooling. When the user sends a mail message, the system places a copy in its private storage (spool) area along with identification of the sender, recipient, destination machine and the time of deposit. The system then initiates the actual transfer as a background activity.

1.2 SMTP

The TCP/IP protocol suite specifies a standard for the exchange of mail between machines. This standard specifies the exact format of messages a client on one machine uses to transfer mail to a server on another. This protocol focuses on specifically on how underlying mail delivery sysytem passes messages across the Internet from one machine to another. It doesn't specify how the mail server accepts the mail from the user or how the interface presents the user with the incoming mail. Also SMTP doesn't specify how the mail is stored or how frequently mail system attempts to send messages.

Basic Policy in SMTP

1.Server sends READY FOR MAIL message
2.Client on reciept of this message responds with HELO message.
3.Server responds by identifying itself.
.....Connection is Setup now.....
4. Client sends MAIL message that contains sender identification11
5. Server responds by Acknowledging this message with OK
6. Client sends RCPT : contains information of recipients
7. Server acknowledges each of the recipients by OK, or rejects it with
NO SUCH USER HERE message
8. Client sends DATA message
9. Server responds by issuing START MAIL INPUT
10. Client sends data
11. At the end of data client sends termination sequence
12. Server accepts it with an OK
13. Client Sends QUIT to terminate the connection if it has no more mails to send
14. Server acknowledges the QUIT message and connection is closed.

1.3 TCP/IP Standards for electronic mail

The mail message is to consist of two parts: a mail header and the mail body. In particular the standard specifies that headers contain readable text, divided into lines that consist of a keyword followed by a colon and a value.

The mail header is to consist of the following keywords: 1. To:

2. From:

3. Reply-To: (Optional)

This header is followed by a blank line which signifies the end of the header. The following part is taken as the message body.

2 Requirements

A computer running on Unix/Linux platform with secondary storage atleast (about 100MB) is the fundamental requirement. The server program is made to run on this machine, using port 25 (Standard port for SMTP transfer) for data transfer between itself and the client. It is very essential that no other process is using this port. If the existing Unix/Linux is running SendMail program, another port is chosen. The client program can run on any machine which is physically connected to the server machine.

3 Functionality

The client program opens a connection with the server, after which mail transfer occurs between the two. Any user has to register before using the service of the system. This interface is provided at the client side. After registering, the client is given accessto his inbox and outbox where he can view his mail, both send and recieved. To send a mail, the client is at first provided with an editor to edit the mail. The mail once edited is sent to the recipients by the client program. It is essential that the recipients of the mail have already registered in the system. Else reject for that recipient is returned from the server. The server on the other end, waits for requests from the client on the port 25. Once a client request is obtained, the server forks a new process to take care of this request. Meanwhile the server continues to listen to requests on the port 25. The new process created by the server does the processing of the client request. The possible requests are

1. Login for existing users
2. Sign Up for new users
3. Recieve mail from the users who have logged in.
4. End the new process once the client is done using the system.

4 Implementation

Client program:

1. Initial function to establish a connection to the server
2. Function to Login a user
This function will accept the login and password from the end
user, send the same to the server, where the validity
of the user is verified.
3. Function to Register a new user
This function will accept the login and password, send it to the server, where the validity of the user name is verified.
4. Function to send mail
This will open a editor for the user to edit the mail, save the mail
and send it to the server according to the SMTP Protocol
5. Function to manage Inbox/Outbox
This gives the users options to delete/view mail in their folders.

Server Program:

1. Function for intial connection establishment and creation of new process
for each new client.
2. Function to login a user
3. Function to register a new user
4. Function to recieve mail and store it appropriately
For checking the validity of a user (which is needed in login, register and in recieve mail), the server uses a file PASSWD, which contains the login name and password of the registered users.

5 Conclusion

SMTP promises reliable mail transfer between machine's since it is based on TCP protocol. However, it supports only ASCII based messages, which is a disadvantage. MIME Extension To SMTP is a good option for mail transfer, and is successfully implemented in the internet today. This project based on SMTP, was successfully implemented and tested. To upgraded the system to support image based messages, and to implement a protocol to retrieve mail from server, are the further improvements which can be done on this project.

6 APPENDIX

6.1 Pseudocode

6.1.1 SERVER

Initialization

1. Create Socket socket sock = socket(AF-INET, SOCK-STREAM, 0);
2. Bind socket with port no and server's IP address bind(sock,"SERVER PARAMETERS");
3. Accept the client accept(clisock,"CLIENT PARAMETERS");
4. call Fork() to create a new process to execute client request

Login

1. Get "login" and "password" from client.
2. Open PASSWD file, read every pair of strings, check for "login" and "password" in the strings. If yes, return ACCEPT message to client. Else send REJECT Message.
3. Wait for further request from client.

New User SignUp

1. Get "login" and "password" from client.
2. Check for same login in PASSWD file. If yes, return REJECT else return ACCEPT to client. Append "login" and "password" to PASSWD.
3. Wait for further request from client.

Recieve mail

1. Recieve Sender's name.
2. Recieve the name of recipients one by one.
3. Check whether each recipient is valid by calling isvalidrecpt(char *name).
4. If yes, create a new file in recipients INBOX with name :sender/Date+Time of mail reciept. Let fp[i] be the file pointer. Date is retrieved using UNIX timeb.h , ctime.h's ftime(),ctime() functions.
5. When client sends DATA command, acknowledge the request.
6. Check for termination character sequence. If not end of message, copy it to the file pointed by fp[i] for i=0 to number of recipients.
7. Close the file 's fp[i], on reciept of End Of Message.
8. Wait for further request from client.

Exit

1. Close the client socket using close(clisock); 2. Exit the child process

6.1.2 CLIENT

//Initialization...

1. Create socket with server port and IP
2. Connect to the server port connect(sock,"SERVER PARAMETER");
3. Wait for request from user.

Login:

1. Accept "login" and "password" from user
2. Send "login" and "password" to server.
3. Recieve reply from Server: if ACCEPT, change directory to this user.
wait for further requests else return;

New User:

Step 1,2 Same as login
3. If ACCEPT, create new directory for user's OUTBOX using "mkdir(OUTBOX)"
4. If reject, end.

Send Mail

1. Open editor "system("vim temp")"
2. Retrieve recipients name from this file.
3. Send Recpt. name one by one.
4. When done, send DATA.
5. Read character by character from the file, send it to server.
6. When done, send termination sequence.

Check Mail

1. Change directory to users INBOX.
2. Display file using "system ("ls")"
3. Scan file name to view from user
4. Open the file, and display it to standard output.

Folder Options.

1. Change directory appropriately.
2. List the files.
3. Get options from user
to delete file, call function "remove(char *filename) or "unlink(char *filename)".
to view open the file and display it to standard output.

References

[1] DOUGLAS E. COMER [Jan 2000], *Internetworking with TCP/IP Principles, protocols, and architectures*, Pearson Education Asia.

[2] STEVENS, W. R. [1998], *UNIX Network Programming, 2nd edition*, Prentice-Hall, Upper Saddle River, New Jersey