



Email System Using Sockets

By:

Neha Pai (653)

Nidhi Nirmal (650)

Ritika Kumar (663)

Shivani Rana (660)

Introduction

- Sockets provide an interface for programming networks at the transport layer. Network communication using Sockets is very much similar to performing file I/O. In fact, socket handle is treated like file handle.
- The streams used in file I/O operation are also applicable to socket-based I/O. Socket-based communication is independent of a programming language used for implementing it. That means, a socket program written in Java language can communicate to a program written in non-Java (say C or C++) socket program.
- A server (program) runs on a specific computer and has a socket that is bound to a specific port. The server listens to the socket for a client to make a connection request. If everything goes well, the server accepts the connection.
- Upon acceptance, the server gets a new socket bound to a different port. It needs a new socket (consequently a different port number) so that it can continue to listen to the original socket for connection requests while serving the connected client.

- There are four types of sockets available to the users. The first two are most commonly used and the last two are rarely used.
- **Stream Sockets** – Delivery in a networked environment is guaranteed. If you send through the stream socket three items "A, B, C", they will arrive in the same order – "A, B, C". These sockets use TCP (Transmission Control Protocol) for data transmission. If delivery is impossible, the sender receives an error indicator. Data records do not have any boundaries.
- **Datagram Sockets** – Delivery in a networked environment is not guaranteed. They're connectionless because you don't need to have an open connection as in Stream Sockets – you build a packet with the destination information and send it out. They use UDP (User Datagram Protocol).
- **Raw Sockets** – These provide users access to the underlying communication protocols. These sockets are normally datagram oriented, though their exact characteristics are dependent on the interface provided by the protocol. Raw sockets are not intended for the general user; they have been provided mainly for those interested in developing new communication protocols, or for gaining access to some of the more cryptic facilities of an existing protocol.
- **Sequenced Packet Sockets** – They are similar to a stream socket, with the exception that record boundaries are preserved. This interface is provided only as a part of the Network Systems (NS) socket abstraction. Sequenced-packet sockets allow the user to manipulate the Sequence Packet Protocol (SPP) or Internet Datagram Protocol (IDP) headers on a packet or a group of packets, either by writing a prototype header along with whatever data is to be sent, or by specifying a default header to be used with all outgoing data, and allows the user to receive the headers on incoming packets.



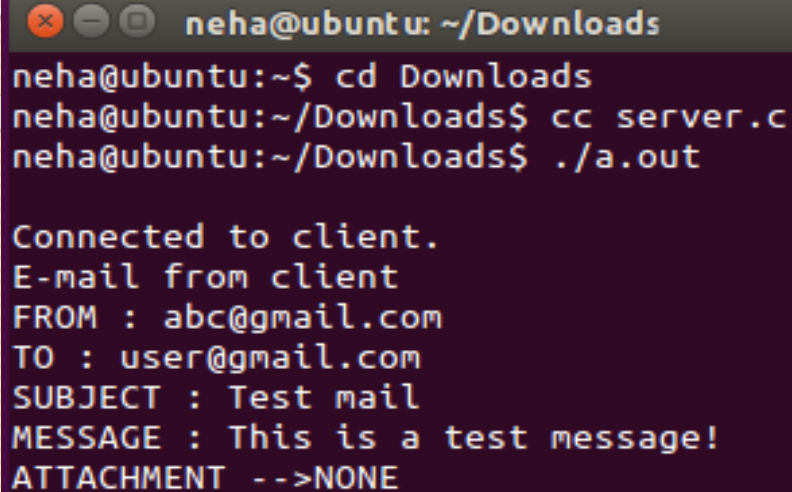
Working Output

Client sending email without an attachment:

```
neha@ubuntu: ~/Downloads
neha@ubuntu:~$ cd Downloads
neha@ubuntu:~/Downloads$ cc client.c
client.c: In function 'main':
client.c:20:1: warning: 'gets' is deprecated (declared at /usr/include/stdio.h:638) [-Wdeprecated-declarations]
gets(from);
^
client.c:22:1: warning: 'gets' is deprecated (declared at /usr/include/stdio.h:638) [-Wdeprecated-declarations]
gets(to);
^
client.c:24:1: warning: 'gets' is deprecated (declared at /usr/include/stdio.h:638) [-Wdeprecated-declarations]
gets(sub);
^
client.c:26:1: warning: 'gets' is deprecated (declared at /usr/include/stdio.h:638) [-Wdeprecated-declarations]
gets(msg);
^
/tmp/ccxrsk4.o: In function 'main':
client.c:(.text+0xe2): warning: the 'gets' function is dangerous and should not be used.
neha@ubuntu:~/Downloads$ ./a.out

From: abc@gmail.com
To: user@gmail.com
Subject: Test mail
Enter Message: This is a test message!
Attach a file (y) or (n) : n
neha@ubuntu:~/Downloads$
```

Server receiving an email without an attachment:

A terminal window with a dark purple background and a grey title bar. The title bar contains three window control icons (close, minimize, maximize) and the text 'neha@ubuntu: ~/Downloads'. The terminal shows a sequence of commands and their outputs. The commands are 'cd Downloads', 'cc server.c', and './a.out'. The outputs are 'Connected to client.', 'E-mail from client', and the email headers: 'FROM : abc@gmail.com', 'TO : user@gmail.com', 'SUBJECT : Test mail', 'MESSAGE : This is a test message!', and 'ATTACHMENT -->NONE'. A white cursor is visible on the line following the attachment status.

```
neha@ubuntu: ~/Downloads
neha@ubuntu:~$ cd Downloads
neha@ubuntu:~/Downloads$ cc server.c
neha@ubuntu:~/Downloads$ ./a.out

Connected to client.
E-mail from client
FROM : abc@gmail.com
TO : user@gmail.com
SUBJECT : Test mail
MESSAGE : This is a test message!
ATTACHMENT -->NONE
```

Client sending an email with attachment:

```
neha@ubuntu: ~/Downloads
neha@ubuntu:~$ cd Downloads
neha@ubuntu:~/Downloads$ cc client.c
client.c: In function 'main':
client.c:20:1: warning: 'gets' is deprecated (declared at /usr/include/stdio.h:638) [-Wdeprecated-declarations]
  gets(from);
  ^
client.c:22:1: warning: 'gets' is deprecated (declared at /usr/include/stdio.h:638) [-Wdeprecated-declarations]
  gets(to);
  ^
client.c:24:1: warning: 'gets' is deprecated (declared at /usr/include/stdio.h:638) [-Wdeprecated-declarations]
  gets(sub);
  ^
client.c:26:1: warning: 'gets' is deprecated (declared at /usr/include/stdio.h:638) [-Wdeprecated-declarations]
  gets(msg);
  ^
/tmp/ccxrsk4.o: In function `main':
client.c:(.text+0xe2): warning: the `gets' function is dangerous and should not be used.
neha@ubuntu:~/Downloads$ ./a.out

From: abc@gmail.com
To: user@gmail.com
Subject: Test mail
Enter Message: This is a test message!
Attach a file (y) or (n) : n
neha@ubuntu:~/Downloads$ ./a.out

From: user1@gmail.com
To: user2@gmail.com
Subject: Hello there!
Enter Message: An attachment will be sent.
Attach a file (y) or (n) : y

Enter the file name : abc.txt
neha@ubuntu:~/Downloads$
neha@ubuntu:~/Downloads$
```

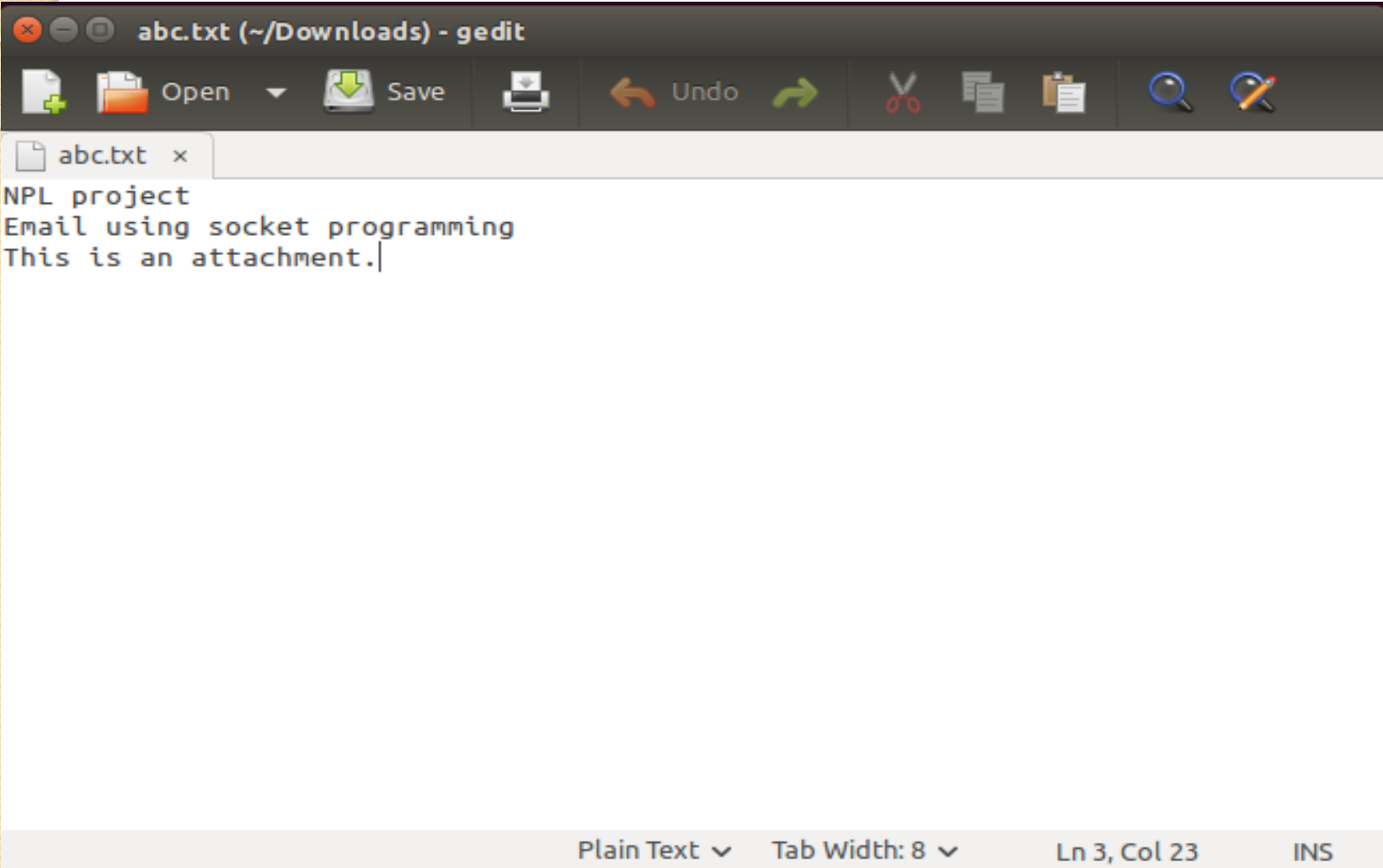
Server receiving a mail with an attachment:

```
neha@ubuntu: ~/Downloads
neha@ubuntu:~$ cd Downloads
neha@ubuntu:~/Downloads$ cc server.c
neha@ubuntu:~/Downloads$ ./a.out

Connected to client.
E-mail from client
FROM : abc@gmail.com
TO : user@gmail.com
SUBJECT : Test mail
MESSAGE : This is a test message!
ATTACHMENT -->NONE

Connected to client.
E-mail from client
FROM : user1@gmail.com
TO : user2@gmail.com
SUBJECT : Hello there!
MESSAGE : An attachment will be sent.
ATTACHMENT -->abc.txt
CONTENTS : NPL project
Email using socket programming
This is an attachment.
█
```


Attached file:



The image shows a screenshot of a gedit text editor window. The title bar reads "abc.txt (~/Downloads) - gedit". The menu bar includes "Open", "Save", "Undo", and "Redo". The toolbar contains icons for file operations and editing. The text area displays the following content:

```
abc.txt x
NPL project
Email using socket programming
This is an attachment.
```

The status bar at the bottom indicates "Plain Text", "Tab Width: 8", "Ln 3, Col 23", and "INS".