E-Mail Servers

Alex S.*

1 Introduction

Controlling e-mail servers directly can be quite interesting, and fun. This document will show some very basic ways of sending and receiving e-mail using the mail servers directly.

2 Sending E-Mail

Most e-mail is sent using SMTP (simple mail transfer protocol) servers. The mail programs logs into that server, sends e-mail, and logs you off. You can do that yourself! SMTP servers reside on port 25 of your e-mail server. Usually, the name is "mail.yourisp.com". You telnet into that server using a telnet program, which is available under both UNIX and Windows (and Apple). You telnet to port 25. i.e.:

```
telnet mail.yourisp.com 25
```

Once you've logged in, the server expects a greeting. The greeting is in the form of "HELO yourip." Luckily, you don't need to type in your IP (most of the time) and can just use a dot in it's place, i.e.:

```
HELO .
```

You should note that the "dot" has spaces on *both sides*! Once you've said `HELO`, you can start sending e-mail. First, you specify who's sending e-mail. This has to be an e-mail address with a valid server (the server's name has to exist, and DNS server has to verify it[1].) Usually, this would be your real e-mail; but if you're using this approach, obviously, you'd want to avoid your real e-mail. Thus, you can make up any e-mail! For example: root@yahoo.com would do pretty well. It will make it seem as though the e-mail has some from there... ;-) (NOTE: do not use yahoo.com as your server! This is just an example.) You continue by specifying:

```
MAIL FROM: root@yahoo.com
```

*alex@theparticle.com

[1]Most modern servers require this.

You could also have said:

```
MAIL FROM: <root@yahoo.com>
```

It doesn't really matter as long as you tell the server who "you are". Then, you have to specify the location where you want to send your e-mail; you do this:

```
RCPT TO: someone@somewhere.com
```

You could also have said:

```
RCPT TO: <someone@somewhere.com>
```

Once you specified the location of your recipient, you can begin the e-mail by typing:

```
DATA
```

Note, that the first several lines you type after `DATA` might have special meaning. For example, after `DATA`, you can type:

```
FROM: Prof.Phreak <profphreak@nowhere.com>
TO: Recipient Name <someone@nowhere.com>
SUBJECT: The subject of the e-mail...
```

You can of course, type in more info, most of the rest are pretty much not standard, and will not be compatible with text only e-mail readers. Note that you don't need these lines; they're only for the receiving program to put info for the user "who sent you the e-mail" mgs. Avoiding them will simply leave the e-mail with no subject, the "TO" field will have "undisclosed recipient;" and the "FROM" field will have your e-mail you've specified in the "MAIL FROM" field. After these three lines, you can begin your e-mail text. The text can be anything, and multi-lined. You end the e-mail by typing a *dot* at the beginning of a line. For example, you end the e-mail this way:

```
see you later ;-)
.
```

See that little *dot* at the last line? To quit from the server, you simply type `QUIT`, i.e.:

```
QUIT
```

After `QUIT`, the server disconnects you. Notice that the only revealing thing in the e-mail will be your IP address. Which if you use a publicly available workstation (the library), and/or telnet into several other servers before hand, will be hard to trace.

3 Receiving E-Mail

Receiving e-mail using a mail server directly isn't much fun. (and most of the times, I'd definitely recommend against it.) Anyway, there are times where this comes in useful. (for example, deleting junk mail without downloading it.)

Most people read e-mail using a POP3 mail server. A POP3 server resides on port 110, usually referred to as "pop3"; you telnet into that server by:

```
telnet mail.yourisp.com pop3
```

You could also have said:

```
telnet mail.yourisp.com 110
```

As soon as you logon, the server is in its *authentication* mode, and expects you to type in your username and password. You do this by:

```
USER username
PASS password
```

If you make an error, you can try again—it will let you know that the password/username is not valid. Once you logon, you'll be in the *transfer* mode. At that point, you can see how many msgs you got, and read them. For example, you can type:

```
STAT
```

To give you statistics about your e-mail; basically, the size of e-mails. Similar goes for the `LIST` command, i.e.:

```
LIST [msg]
```

The `[msg]` there is optional, if you want to find out stuff about an individual e-mail, you use it. (I almost always use `LIST` with no parameters). After you know how many e-mails you got, you can read (well, almost) each individual e-mail by typing:

```
RETR msg
```

Where "msg" is the e-mail number gotten from `LIST` command. The only disadvantage is that if the e-mail is large, it just zoomes past you, and you can hardly read it—that's why I suggest an e-mail program.

To delete an e-mail you type:

```
DELE msg
```

This deletes (marks as deleted) the `msg` number pointer to by "`msg`". I say "marks as deleted" because it doesn't actually get deleted at that point. It will only be deleted when you leave the server. You can actually undelete the e-mails by typing:

```
RSET
```

That's it for the standard (required) POP3 commands. *Most* servers implement many optional commands as well—which I won't describe here. However, I'll mention the most useful one (at least to me), the:

```
TOP msg n
```

Where the `msg` is the `msg` to display, and `n` is the number of lines to display. Thus, you can see the header of an e-mail, without reading (or downloading) the whole e-mail.

You leave the server by typing `QUIT`, i.e.:

```
QUIT
```

At this point, the server deletes all the `msgs` marked as deleted (cleans up stuff), and waits for the next login.

A suggestion is not to take up the POP3 server for a long time—since when you're logged in, it locks everything, and somebody who's sending you e-mail might not be able to get through on some configurations[2].

4 Other Related Protocols

Over the years, many e-mail related protocols have been developed and deployed. Most notebly, MINE (which handles different data types within e-mails), and IMAP (which allows for much more flexible e-mail manipulation on the e-mail server).

There are also protocols which won't be mentioned, like tunneling XML SOAP envelopes over the e-mail system (ie: web-services that work over SMTP as opposed to over HTTP).

4.1 IMAP

Internet Message Access Protocol is a standard protocol for accessing e-mail from your local server. It is mostly an advanced version of POP3, and is in line with many webmail applications.

IMAP is a client/server protocol in which e-mail is received and held for you by your Internet server. You can view just the heading and the sender of the mail and then decide whether to download the mail. You can also create and manipulate folders or mailboxes on the server, delete messages etc.

You can read up a lot more on this protocol in [RFC2060].

[2] I haven't seen such a configuration in a long while—but hey, it used to be true ∼10 or so years ago.

4.2 MIME

Multipurpose Internet Mail Extension, a standard system for identifying the type of data contained in a file based on its extension. MIME is an Internet protocol that allows you to send binary files across the Internet as attachments to e-mail messages. This includes graphics, photos, sound and video files, and formatted text documents.

5 Implementation

This section presents some Perl code to send/recieve e-mail.

5.1 Sending E-Mail

For this code to run, you need `Net::SMTP` package. If it is not already installed, you can easily find it online and install it. The code below connects to the SMTP server, creates a new message and sends it. Obviously you'll likely want to change the actual names and e-mails before running it.

```
#!/usr/bin/perl

use Net::SMTP;
$smtp = Net::SMTP->new('mail.earthlink.net', Timeout => 60);
# FROM
$smtp->mail('from@email.com') or die $!;
# TO Addresses
$smtp->recipient('to@email.com') or die $!;
$smtp->data(q{To: blahname <to@email.com>
From: yourname <from@email.com>
Subject: Blah blah blah and more blah

this is the body

-bob
}) or die $!;
$smtp->quit;
```

5.2 Recieving E-Mail

For this code to run, you need `Net::POP3` package. If it is not already installed, you can easily find it online and install it. The code below connects to the POP3 server, downloads all messages and saves them as local files in the current directory. Obviously you'll likely want to change the actual names and e-mails before running it (especially the login information).

```
#!/usr/bin/perl

use Net::POP3;

$pop = Net::POP3->new('mail.earthlink.net', Timeout => 60);
$nummessages = $pop->login("username","password") or die;
print "There are: ".$nummessages." messages.\n";

($numelements,$sizeofmbox) = $pop->popstat();
print qq{numelements: $numelements
sizeofmbox: $sizeofmbox\n};

$list = $pop->list();
for $msgnum (sort {$a <=> $b} keys %$list){
    $uidl = $pop->uidl($msgnum);
    print "msg: $msgnum; size: ".
        $list->{$msgnum}."; id: ".$uidl."\n";
    open $fh,">$uidl.txt" or die $!;
    $pop->get($msgnum,$fh);
    close $fh;
}

$highestmsgnum = $pop->last();
print "highestmsgnum: $highestmsgnum\n";
print "disconnecting...\n";
$pop->quit();
```

6 Conclusion

That's it for handling the e-mail servers directly. Most of these though, are more conveniently used in a small networking program. (writing your own mail client...) In which case, you'd want to take a look at [RFC1939] document for POP3, and [RFC822] document for SMTP.

A very important note: most of these servers don't echo back the characters, thus, if you're typing, you won't see your text appear in the telnet window. (To fix it under Windows, you setup your terminal window to echo the characters locally.) Under UNIX, you can just telnet into another machine, that way, that other machine will echo the characters to you (not the mail server).

Good luck! (and don't use this info to send hard to trace junk mail)