9 Email

Sendmail, mailing lists, email clients, and POP/IMAP (remote connections)

This chapter shows you how to set up sendmail, set up simple mailing lists for customers, clients, or groups of people, review some of the email clients available, and set up remote connections using the POP (Post Office Protocol) and IMAP (Internet Message Access Protocol) mail standards.

Email (electronic mail) was one of the first methods for communicating over what is now the Internet. Email allowed short messages to route across various machines. In the early days (1987), you often had to manually specify all the machines (or hops) an email message would have to route through to get to the end host. With TCP/IP as the communication method, things are a bit smarter in that you don't need to know how to get to the end host—as long as some machine along the way knows how to route the email for you. The technical name for the email that you get on the Internet is known as Simple Mail Transfer Protocol (SMTP). There are a few other protocols that work with SMTP, such as POP and IMAP, but we'll get to those later. You can find more information about SMTP in RFC 821.

The primary program that is the Mail Transport Agent (or MTA) is sendmail. There are others available, but sendmail is pretty much the standard for most Linux installations. You may have seen that really thick book with a bat on it that gets you into the nitty-gritty of sendmail. It gets really nitty-gritty.[1]

The upshot of all this is that you don't always need sendmail to set up a simple email server. The one thing you should know about sendmail is this: Always be sure you have the latest version. Each version of sendmail, while patching various bugs, seems to have a security problem with it. Sometimes it seems like you need to upgrade your sendmail software more often than the Linux kernel.

Enough of that. Let's get to the typical sendmail options you'll see on startup. Most Linux installations have sendmail kick in on bootup. Red Hat (being SYSV-ish) has an `/etc/rc.d/init.d/sendmail.init` script that runs on startup. An installation could start the actual command like this:

```
echo -n "Starting sendmail: "
daemon sendmail -bd -q1h
echo
touch /var/lock/subsys/sendmail
```

As you can see, the sendmail program starts with the `-bd` and `-q1h` options. The `-bd` signifies that sendmail should start up as a daemon and run in the background, waiting for connections, much the same way that a Web server is always running, waiting for a connection. The `-q1h` means that sendmail should go through its queue at least once an hour and try to send any data in its queue.

There are a number of reasons why an email message would be held in a queue. For example:

- You have a dialup-only connection to the Internet, and you send a note while you're not connected.
- The remote host is not dialed in (same thing in reverse).
- The remote host is down for upgrades or it crashed.
- There is no connection to the remote host (the network is down).

In any of these cases, you'll want to make sure that your email gets through. Leaving the email in sendmail's queue allows you to do this. A machine directly connected to the Internet or on a LAN can change this to

[1] If you have anything to do with setting up sendmail, get this book.

ten minutes (`-q10m`), since you're more likely to be connected to the Internet all the time. In case you're interested in what's currently in the queue, the `mailq` command will list what sendmail has. The `mailq` program will also list the reason why an email message is stuck in the queue. If there is a lot of email in the queue, you may have a network or configuration problem.

You can verify that sendmail is running by connecting to TCP port 25.

```
[markk@wayga ~ ]# telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 wayga.net ESMTP Sendmail 8.7.6/8.7.3; Fri, 18 Apr 1997 00:26:18 -
  0400
```

As you can see, the connection works, and there is a bit of information about the host we connected to: status number, hostname, type of SMTP (Extended SMTP), the `Sendmail` program (instead of qmail or the like) version `8.7.6`, and the current time and date that the computer thinks it is.

The configuration options for sendmail are typically located in `/etc/sendmail.cf`. Examine the file if you like, but don't change anything unless you know what you're doing! Small changes can render your email system useless. As always, back up files that you want to edit, just in case.

A few options you may want to note:

- `OA`—Can set another file that contains aliases. We'll get into this with majordomo.
- `Mlocal`—Specifies what program will act as the Mail Delivery Agent (MDA). This can be something like `/bin/mail` or `/usr/bin/procmail` (we'll get to this later, too!).
- `DS`—Contains a pointer to a "smart" relay host. If you have a very slow connection to the Internet or don't have DNS set up, you can specify another machine to accept all your email and deliver it for you. For dial-up links, you only have to send the email once, and a machine that is permanently connected to the Internet can hold it in its queue.
- `Cw`—This option works if you have one Internet host that has multiple names (for example, `wayga.net` and `ratatosk.org`). This entry will allow you to specify which hosts you will receive email as.

9.1 Using m4 Files

One option that many versions of sendmail has is the option to have your `sendmail.cf` file automatically generated. This will allow you to keep a small file that has most of the options you wish to use. When upgrading sendmail, you need to recompile only `sendmail.cf` to have the latest updates to that file. The m4 macro language is excellent for doing this, and sources for various types of `sendmail.cf` files are included.

If you decide to use the m4 files, be sure to download the source code for sendmail, as the Red Hat distribution does not include m4 files by default. The m4 processor is installed by default, however, and is located in `/usr/bin`.

9.2 You Have Mail!

Once email has arrived at your system, there are a few things that happen to it before you actually get alerted to new email. First, sendmail checks to see if you have a `.forward` file in your home directory. If so, then sendmail will forward the email to whatever address is specified in the `.forward` file. If you're using `procmail` and you have a `.procmailrc` file, sendmail will run `procmail` and follow the rules in it. After that, sendmail appends the email to a file called `/var/spool/mail/<USER>`, where `<USER>` is the username. This file does not need to be created when making a new user; sendmail will take care of that the first time the user gets an email message.

Creating mail aliases allows incoming mail to get routed to a different user or program, even if there is no user account with that name on the system. This will have a larger impact once we get to managing mailing lists. For now, you may want to examine the `/etc/aliases` file to see what aliases are currently defined. This is a method of having a mail alias that really sends the mail to someone else, or a group of people. For example, a common `/etc/aliases` file may contain the following:

```
MAILER-DAEMON:  postmaster
postmaster: root

# General redirections for pseudo accounts.
bin:        root
daemon:     root
games:      root
nobody:     root
uucp:       root
```

```
# Well-known aliases.
manager:    root
operator:   root

# Person who should get root's mail
root:       mark
mfk:        mark
bren:       brenda
bdk:        brenda
```

You can see that there are a number of aliases listed, most of them pointing to `root`. In some cases (`bin` and `nobody`), the accounts exist, but no one should ever log into those accounts. In this case, any email that gets sent to `bin` or `daemon` merely gets forwarded to `root`. Also, note that `postmaster` is set to `root`; in most situations, the person who administers the mail also administers the rest of the system. The `MAILER-DAEMON` is mostly for mail errors. These errors get sent to `root` as well.

Later on, you see a few user aliases. These aliases are set up so that anyone sending email to `mfk@wayga.net` will go automatically into the `mark` account. There is no `mfk` account, but people can send email to that address. The same is true of anyone sending email to `bren@wayga.net` or `bdk@wayga.net`. A small business could have email aliases of `mark.komarinski`, `mfk`, `m.komarinski`, `markk`, `mark.k`, and any other unique permutation of a name in case outside customers forget an email address. It is something that's often overlooked, but it is very important to keep in touch with customers.

Once the mail is delivered to the correct account, a process called `comsat` runs; `comsat` is what actually tells the shell that you have new mail. Once the shell gets that information, it may notify you of it. The `biff` program tells you if you'll receive notification, and it also lets you turn it on or off. The `y` option to `biff` will turn mail prompting on, and `n` will turn it off. There are also X versions of `biff` to give you graphical notification that email has arrived.[2]

Now that you've received an email message, how are you going to read it? This is the job of the MUA, or Mail User Agent. This is the only area the end-user is going to interact with directly. There are dozens of programs available from the low-feature (mail) to the high-feature, full-screen (elm, pine, mutt) to the graphical (Netscape).

[2]Why name the command `biff`? Turns out that's the name of the author's dog. Under UNIX, if you write it, you name it.

What is used for mail reading is pretty much up to the user. For example, many people prefer pine over elm. If you plan on having many users, it's best to install a few MUAs aside from `/bin/mail`. Red Hat's installation provides a number of different MUAs to suit everyones' needs. Each MUA has advantages.

9.3 MIME

When SMTP was first designed, it would handle only 7-bit (plain ASCII) data. Hence the "Simple" in the protocol name. With things like JPEG images, sound files, application data, and compressed files to be sent through email, there had to be some way of sending binary (8-bit) data through a 7-bit stream. The first method of handling this was via the `uuencode` program. This program would turn two bytes of 8-bit data into three bytes of 7-bit data. The file size was two-thirds larger, but it would now squeeze through the 7-bit path that SMTP provided. Another problem in addition to the file size was that the remote side had to decode the data (`uudecode`). This turned into a tedious procedure, the sender encoding the data and manually including it in a mail file and the receiver trying to strip out all the mail headers, text, and signatures to get to the `uuencode` file and then decode it.

In the early 1990s, the Multipurpose Internet Mail Extension (MIME) was developed to do this, and it is used today on the Web for determining file types. MIME encoded each data type such that any MIME-capable mail reader would understand what kind of data was in the message. Once a file type was detected, each individual user could have a file (`.mailcap`) that would list what file types you knew about, and what application to run once you found that file type. For example, if a JPEG image were sent, the receiving side might start up the `xv` program and display the image in an X window. Another user might have it set up so the image was converted into PostScript and sent directly to the printer.[3] Most mailers today have the ability to handle MIME mail, and there should be little to no interaction that you as an administrator have to worry about.

[3]This is a good example of "the UNIX way"—there's more than one way to do anything.

9.4 The .forward File

When mail comes in, the `.forward` file is checked. If it exists, the email is sent to the address listed in the file. This is helpful if you have multiple accounts on the Internet and want all the email to come to one location. An example of `.forward` shows another feature of the file—instead of an email address, you can have a command. The `vacation` program, which isn't available in Red Hat (but is on the CD-ROM), allows you to put a program in the `.forward` file. Each time email comes in, the program is run. The `vacation` program stores the email and then sends a custom message back to the email sender, notifying them that you're on vacation (or out of town, out of the office, out of your mind, etc.). The sender then knows that the email got through.

Procmail

If `procmail` is installed as the Mail Delivery Agent (MDA) on your system (check the `Mlocal` entry in your `sendmail.cf` to find out), you can immediately use it to filter all your incoming mail. If not, you can put a reference to `procmail` in your `.forward` to run each time new mail arrives. `Procmail` is used to process mail as it comes in. This can include putting all email from the "Plan9 MUSH Advisory Committee" in one mailbox separate from everything else, or filtering all that annoying email you get from `billg@microsoft.com` by sending it to `/dev/null`.[4]

If `procmail` is set up to be the default local mail handler, setting `procmail` up for use is easy. Create a `.procmailrc` file and start adding rules for `procmail` to filter. If `procmail` is not your local mail handler, you have three options:

1. Make `procmail` the local mail handler (the man page has the suggested changes you can make to the `/etc/sendmail.cf file`). If you're

[4] Anything sent to `/dev/null` is immediately discarded. This is called the "bit bucket." Thanks to the fact that Linux is a 32-bit OS, its bit bucket is one of the fastest around.

using m4 configuration files, add `FEATURE(local_procmail)` to your file.

2. Run `procmail` periodically, or use `cron` to filter out an existing mailbox. This still allows you to use `.forward` (in the event of a vacation and the like), but the mail is not always sorted and you will have to wait.

3. Put a reference to `procmail` in your `.forward` file. Note that this won't let you forward all that easily. In the `.forward` file, put a line in that looks like the following:

```
"|IFS=' '&&exec /usr/local/bin/procmail -f-||exit 75
  #YOUR_USERNAME"
```

As you can guess, option 1 is the best, as it requires the least amount of effort on everyone's part.[5] Here's a copy of a sample `.procmailrc` file:

```
PATH=/bin:/usr/bin:/usr/bin
MAILDIR=$HOME/Mail      #you'd better make sure it exists
DEFAULT=$MAILDIR/mbox   #completely optional
LOGFILE=$MAILDIR/from   #recommended

:0:
* ^From.*mark
from_me

:0
* ^Subject:.*Flame
/dev/null
```

There are three sections to matching a line and performing some action on it, also called a recipe. A recipe starts with a line that has at least `:0`. After the `:0`, you can include some flags (for example, `H` will scan only the headers). Pattern matching is done for any lines that start with an asterisk (`*`). The rest of the line is sent to `egrep`[6] literally. If all of the lines match (the lines are ANDed together), then the appropriate action is taken. If not, it moves on to the next recipe.

There can be only one action line, and this is the one that does not start with an `*` or `:`. An action starting with `!` will forward the program to the spec-

[5] Except the system administrator, who has to install the software and make sure that sendmail isn't busted. Oh wait. That's you. Scratch that.

[6] `egrep` = extended grep, and the syntax is not exactly the same as with plain `grep`.

ified user. An action starting with | will start a shell and pipe the email to that program for processing. Anything else will be assumed to be a filename to append (or create if it doesn't already exist).

These two examples show that all mail that has a `From` line and ends in `mark` (which will be any email sent by me) will be put in the `~/Mail/from_me` mailbox. The second example shows that all email that starts with the word "`Subject`" and ends in "`Flame`" will be sent to `/dev/null` (the bit bucket).

The thing to know if you just want to delete mail out-of-hand is that you have to be absolutely sure that your logic is correct. For example, if you delete all mail that has the word "`Flame`" in it, then you could potentially lose messages with a subject such as "Flame broiled burgers free until 10 AM." OK, so it's a poor example. You may want to store mail you want to delete for a while and then examine it just to be sure you don't lose anything important.

9.5 Mailing Lists

Mailing lists allow larger groups of people to communicate via email. In many cases, it is preferable to using USENET since mailing lists do not need approval from anyone other than the administrator to be created.

Anyone with a machine connected to the Internet can create a mailing list on any subject. Getting members to subscribe to it is a different issue, however.

Mailing lists can control who subscribes. The mailing list administrator can make it so anyone can subscribe, or he/she can make it so that the administrator is the only one who can add new members to the mailing list. The administrator can also remove offending members if a problem arises.

Mailing lists can control content. This isn't as bad as it sounds, since it is the alternative to the anarchy that is USENET. The administrator can set the list up to be open and provide a healthy discussion area (such as for developers working on a piece of software). Alternatively, the list can be set up such that the administrator has to approve each piece of email that gets sent to the list, like moderated USENET groups. Moderated lists are good for news releases, humor lists, or any other list where the amount of discussion is low.

Majordomo

The best mailing list program in use has to be majordomo. Majordomo provides all the features listed above, plus adds a few other features, such as

sending out regular digests, or archiving for future use. We'll get back to these later on.

The first thing that needs to be set up is majordomo itself, if it's not already installed. The source for it is located on the CD-ROM. You need to have Perl 5 installed on the system (as well as GCC, of course) to use it. First, create a majordomo user. Given the eight-character limitations of Linux's usernames, the username will have to be shortened to `majordom`. Thanks to `/etc/aliases`, no one ever has to know this. You should also create a home directory for the `majordom` user. This can be `/home/majordom` if you like, or `/usr/local/majordom` if you prefer, or anywhere you have space.

`Su` to `majordom`, copy the `majordom1.94.1.tar.gz` file to the home directory, and uncompress/untar the file. Make any kinds of modifications to the `majordomo.cf` file; then just use the `make` command to create it. Once installed (which should be in the majordomo home directory), you need to make only one change to the `/etc/sendmail.cf` file. You should already have[8] one OA (or O alias file) entry listed. Underneath that, put in the following:

```
OA/home/majordomo/majordomo.aliases
```

This sets up an additional file to handle aliases. It also makes it easier for the majordomo administrator to add new mailing lists, and that person does not need root access to modify the lists.

Now, on to creating the lists themselves. Let's create a (fictional) list called `gastro`. It will be an open list for sharing recipes, so anyone can join, and anyone can post a note to the group.

In the `~majordom/majordomo.aliases` file (we'll call it the alias file from now on, in case you chose to use `/etc/aliases` instead), add the following:

```
gastro:                    "|/home/majordom/wrapper resend -l gastro
  gastro-list"
gastro-list:                      :include:/home/majordom/lists/gastro
owner-gastro:                       mark
gastro-owner:                       mark
gastro-request:                        "|/home/majordom/wrapper
  majordomo -l gastro"
gastro-approval:                       mark
```

[7]You set up a password for the `majordom` user, right??

[8]Note that this is a bit of a security risk, as root should really be the only person who can create new aliases. If you want, leave this statement out and use `/etc/aliases` instead of the `majordomo.aliases` file.

This sets up the basics. The list itself will be gastro@host, the owner will be mark, and any requests to join the list will be sent to majordomo as well. Normally, users would send email to majordomo@host to subscribe, but different mailing list software works in different ways.

Once the aliases are set up, you can initialize the lists using:

```
echo "config gastro gastro.admin" | mail majordomo
```

and the lists will have some default configurations put in. If you choose to change the defaults (and you should!), edit the ~majordom/lists/gastro.config file. This has all the configuration settings for the particular list.

Now that your mailing list is set up, you should be able to email to the list by just sending mail to gastro@host, and the email will be sent to all the subscribers of the gastro list. There are a number of extra features that majordomo has, and you should check out the Web site for more information and to get the latest updates.

9.6 Qmail: An Alternative to Sendmail

One of the better known replacements for sendmail is called qmail. There are two reasons for its popularity: ease of use and security. This doesn't mean that qmail is best for everyone, since there are a number of changes that your system has to go through to use qmail. It's also not perfect in a large installation (yet) because of some of the ways that sendmail and qmail differ. The configuration for qmail is much simpler than sendmail, but it doesn't offer the "Swiss army knife" of Mail Transport Agents (MTAs) that many expect to see.

Here's a few of the ways in which qmail and sendmail differ in methodology:

- *Mail file location*—Sendmail prefers to use /usr/spool/mail/user as a location for incoming mail. Qmail prefers to use ~user/Mailbox as a location for storing mail files. This is for two reasons. First, putting the mail in the user's home directory provides for better quota checking and prevents overfilling a central location. Second, there are fewer security problems since there isn't a central location to store files.
- *Permissions*—Sendmail runs almost always as root, or with root privileges, to write files and so forth. Qmail runs as root only when necessary and otherwise runs as a qmail user, which has no special permissions.
- *Mailing list integration*—Sendmail has external programs to create and it administers mailing lists (majordomo and listproc). Qmail supports a

method where individual users can have their own mailing lists and administer them themselves with no extra accounts or permissions.

- *Size*—Sendmail is a monolithic MTA, where everything is done in the program itself. Qmail has a number of smaller programs to handle each aspect of mail receipt, delivery, sending, and so forth.

Installing Qmail

You can download qmail in either RPM format or source code (also called a tarball, due to the fact it's in tar format). Both files can be found at `http://www.qmail.org/`. If you download in RPM format, be sure to remove the sendmail RPM from your system to avoid conflicts. The RPM is in source format, but handles many of the functions of building and installing qmail, and adding the users and setting up initialization scripts for use with Red Hat.

After you get the `qmail-1.03.tar.gz` file, untar and uncompress the file. This will create a `qmail-1.03` directory with the source code in it. Review the `INSTALL` file so you know what the full install procedure is. This procedure allows you to build and install qmail without affecting the sendmail you probably already have installed until you're satisfied that qmail is working correctly. Qmail resides in the `/var/qmail` directory, so you'll need to create that directory and remember that so you can monitor that directory. Next, you'll need to create users for qmail to use. Since security is a big component of qmail, these new users will need to be created to separate qmail from the root user and regular users.

Now for the simple part. Enter `make` to build qmail. The compile should take a few minutes to complete, and you'll be ready for the installation and testing. The biggest problem you may run across is the fact that your home directory and `.qmail` files cannot be group- or world-writable. This will show up in the logs as something like:

```
Feb  8 16:50:00 wayga qmail: 886974600.032889 delivery 10:
  deferral: Uh-oh:_.qmail_file_is_writable._(#4.7.0)/
```

In this event, just find the `.qmail` file it's referring to and change the permissions. You'll also want to check for other copies of the sendmail program floating around the filesystem (like in `/usr/sbin/`, for example), and change those to use qmail. Also while testing, remember that mail is no longer being delivered to `/usr/spool/mail/user`, but to `~user/Mailbox`.

Once the testing and configuration are done (the `INSTALL` file is pretty thorough on this), you can then tell your users about the changes and start using qmail.

A few notes you should know of if you're considering changing MTAs:

- If you're upgrading or changing software to something you're not used to, always use a staging machine. There are instructions in the FAQ file on how to upgrade slowly from sendmail to qmail with a minimum of fuss and trouble. This upgrade path allows you to use both qmail and sendmail until you get the bugs worked out and move to qmail completely.
- You may not want to put qmail on a connection that goes down frequently. One of the features of qmail is that it continually tries to deliver mail. If the Internet link goes down often, you could be in a situation where qmail is trying to deliver lots of mail at the same time. If you have a dialup link, it might be worth it to check out the serialmail program, which is designed to send out email on demand (i.e., when the link is up).
- In addition to the `/usr/spool/mail` directory changing, the `.forward` file is ignored in favor of the `.qmail` file. The `.qmail` file has a number of features that can't be put in `.forward`, but programs that operate on `.forward` won't work anymore.

Once qmail is configured and working properly, you'll need to make sure that you can deliver mail to each user (check their mail file and the `syslog` output in `/var/log` to make sure mail is being delivered properly).

There are a few things that you will notice once qmail is installed. For the benefit of your users, you should warn them ahead of time.

- As previously mentioned, mail now gets stored in `~user/Mailbox` instead of `/usr/spool/mail/user`. You'll need to make sure that the `MAIL` environment variable is changed (see `/etc/profile`). There are two problems with the `/usr/spool/mail` approach to delivering mail. First, it's a security risk. Second, the way sendmail does local mail delivery is prone to losing mail if the mail directory is mounted via NFS or if the server goes down while writing email. The "Mailbox" format in qmail makes sure the email is written to the disk before it reports success back to the main qmail program. If there is a locking problem or the machine goes down while a message is being delivered, it will get delivered when the machine restarts or the lock goes away.
- Qmail does not look in the `.forward` file anymore for forwarding email to other users. The replacement file, `.qmail`, handles this. The nice thing about this is it allows you as a user to create your own simple mail-

ing lists. For example, if you wanted to create a simple mailing list related to this book, you could create a `.qmail-linux` file in your home directory. That file then would get a list of the people you wanted to be on the list. Once this was done, any email sent to your address would get sent to the entire list. You could even have a `.qmail-default` to catch all `enry-* email` that came in. Note that this isn't a very full-featured mailing list system as it doesn't handle automatic subscribes or unsubscribes. This is where ezmlm comes in (see next section).

- Qmail also does not use `/etc/aliases` at all. There is a global way of setting up email aliases, which is a bit easier to use. The alias user is used by qmail to handle addresses that don't exist. For example, you could have `~alias/.qmail-markk` contain the line:

```
enry@wayga.net
```

  to forward email that comes in for `markk@wayga.net` to `enry@wayga.net`. A common use of this is setting up postmaster, abuse, or root aliases.
- The `root` user does not receive email. Due to the number of security problems this could cause, it was determined that `root` should be aliased to someone else (like the actual administrator of the machine). This makes cracking a root account much harder.

Ezmlm

The ezmlm package allows for easy mailing list setup. It has all the functionality commonly needed in mailing list applications: automatic subscribe/unsubscribe, moderated lists, and digest creation (handled by `ezmlm-idx`). Unlike many other mailing list applications, ezmlm is command-based—there should be no need for the mailing list administrator to edit files. Another feature of ezmlm (and qmail) is that mailing list creation and administration do not require access to the root account.

By default, many of the ezmlm programs get installed in `/usr/local/bin/ezmlm`. The install instructions are in the `INSTALL` file (located in the `ezmlm-0.53` directory) and are straightforward. Once installed and tested, users can create their own mailing lists with the following commands:

```
ezmlm-make ~/list ~/.qmail-list user-list host
```

This creates the mailing list. The mailing list itself is contained in `~/list`. This mailing list is not moderated in that anyone can subscribe and unsub-

scribe. If you want a private mailing list where only the moderator can add and remove members, you can add the `-P` option. Once the mailing list is created, you can just remove `~/list/public` to create a moderated list or create the file to create a public list:

```
ezmlm-sub ~/list email
```

This subscribes `email` to `list`. The new member is subscribed to the list.

```
ezmlm-unsub ~/list email
```

This unsubscribes `email` from `list`.

```
ezmlm-list ~/list
```

This lists the members of `list`.

Once a list is created, the `user-list-subscribe` and `user-list-unsubscribe` email addresses are created for users to subscribe and unsubscribe. If the mailing list is moderated, these mail addresses will not work and the moderator will have to manually add new members. An extension to ezmlm, `ezmlm-idx`, provides for more moderation and archiving.

Ezmlm Files

Once a mailing list is created, you'll find the following files in the directory:

`Archive/`	Contains an archive of the mailing list. Messages are numbered from 1 on. If the `archived` file exists, no messages will be archived.
`Archived`	If this file exists, ezmlm will archive messages.
`Headeradd`	Contains a list of headers that will be added to email sent by ezmlm.
`Headerremove`	Has a list of headers that will be removed from email that gets sent out by ezmlm.
`Key`	This file has some random binary data to prevent forging subscription requests.
`Num`	Contains the number of messages sent so far.

`Public`	If this exists, users can subscribe and unsubscribe without the help of a moderator. If it doesn't exist, only the moderator can add or remove users.
`Text`	Directory containing files that ezmlm will send back as administrative messages. You can change these files if you want.

9.7 Remote Email (POP and IMAP)

POP (Post Office Protocol) and IMAP (Internet Message Access Protocol) are two of the standards for receiving email from a remote machine. The best example of remote email use is in an office situation where there is a Linux server and Windows (or Mac, or other UNIX) clients also running on the network. The clients can receive their email in a number of ways (Eudora, Netscape, PC-Pine, etc.) and still have one central mail server.

Each protocol has strengths, weaknesses, and assorted Mail User Agents (MUAs) that support them. Both are available in a Red Hat install setup ready to run. Authentication of email is done by the same PAM scheme that is used to log in. One of the advantages of using POP or IMAP is that little user setup is required. You can typically install and run client software without any trouble.

POP

POP (Post Office Protocol, defined in RFCs 937 and 1225) was one of the first methods for retrieving and sending email to/from a remote machine. The protocol is rather simple. You can get, list, and delete mail. That's about it. The good thing about it, though, is that you can pick up your email at the office from home. The downside to POP[9] is that the email on the server is often deleted once it gets sent to you. So, while you can get your email from work to home, you then have to send the email back from home to work, or else you won't have any email at work. Netscape, Outlook, and most PC-based Internet email programs handle POP.

[9]To be fair, this is a problem with the applications that use it, not the protocol itself. POP can be set up to leave email on the server, which can fill the server up very quickly.

The POP server (`popper`) is included in a typical Red Hat distribution as part of the `imap-4.5` RPM. To use it, configure the client software so that the server matches the Linux box. The username and password are the same as in `/etc/passwd`. Remember that any email downloaded gets stored on the local machine, so make sure that user drives are backed up regularly. One option to look at is the `Don't delete from server` option, which might be named differently on some clients. This option leaves the email on the server after delivery. If you have two locations where you want to read your email from (work and home, for example), you can leave this option off on the office email system, and turn it on at home. This way, all your email gets downloaded at work, and you can only get new email from home. IMAP is a much better solution to this problem.

IMAP

IMAP (RFC 2060) was created to handle some of the deficiencies of POP. IMAP is designed to be a client/server setup, where the server keeps the mail until the client requests that it be deleted. In POP, all the email is downloaded at once, which is a bit of a problem for large messages over slow links. IMAP allows for downloading only the body of a message when it is required, allowing you to quickly download header information and then read only the messages you want to read. Email can be stored in different files (often called folders) to allow you to organize your email easier. These folders remain on the server side and can be accessed by the client software at any time.

Another big feature of IMAP is disconnected mode, sometimes called offline mode. If the client supports it, you can have the client download one or more folders to your local machine, usually by selecting an option called `go offline`, `download folders`, or `offline mode`. You can then read, modify, delete, reply, and so on without being connected to a network. When you return to your network, you can return to online mode and synchronize the client and server. A list of clients and their support for IMAP can be found at the official IMAP home page, `http://www.imap.org`. Both Netscape Communicator and MS Outlook support IMAP and offline mode.

The IMAP server is included with Red Hat in the imap-4.5 RPM, and is configured to get files from `/var/spool/mail`. To configure it for use with qmail, you'll need to recompile the server. Client setup is about the same as for POP; set the server name and the username/password. One option that you may want to look at is the `server directory` option. This option tells the client what directory on the server contains the mail folders (not the `mbox` or new mail – that's hard-coded on the server). By default, this is the user's

home directory, and if a user does not store other files in that directory, you're okay. However, if you share user home directories via Samba or NFS, you'll want to change the server directory to something like `~user/Mail`. The downside to this is that you have to set it each time you configure someone for IMAP. The benefit is that your users won't confuse their email with their regular files.

By default, `mbox` (mail that was read in and not put in another mail folder) is stored in `~user/mbox`, which is actually the same file used by the UNIX `mail` program. New mail is checked for in `/var/spool/mail`. If you need to change this, you'll have to recompile the software. The easy way to do this is to get the `.src.rpm` file from the Red Hat FTP site or source CD-ROM and compile it. You can find out more about installing and compiling source RPM files in the RPM chapter.

9.8 Summary

- Sendmail is the default way of sending email.
- You might benefit from using qmail.
- MIME allows you to email binary data from one person to another.
- POP and IMAP allow you to get your email from remote locations.