The Linux Electronic Mail Administrator HOWTO

Table of Contents

Table of Contents

The Linux Electronic Mail Administrator HOWTO

Guylhem Aznar `<guylhem at metalab.unc.edu>`

v3.2, January 2000

---

*This document describes the setup, care and feeding of Electronic Mail (e−mail) under Linux. It is primarily intended for administrators, rather than users. (See the Mail−User's−HOWTO for information on user issues and user agents.) You need to read this if you plan to communicate locally or to remote sites via electronic mail. You probably do *not* need to read this document if don't exchange electronic mail with other users on your system or with other sites.*

4.Requirements

5.Choosing a Mail Transport Agent

6.Installing Transport Software

7.User Agent Administration

8.Handling remote mail

9.Acknowledgements

1.Introduction, copyright and standard disclaimer

1.1 Email and spamming

To send mail to anyone mentioned in this document, convert "at" in email addresses to "`@`".

This conversion is simple for humans, but not spammers' address harvesters; therefore it's useful to protect generous contributors from being spammed!

1.2 Goals

The intent of this document is to answer some of the questions and comments that appear to meet the definition of "frequently asked questions" about e−mail software under Linux in general and the version in the Linux Debian and RedHat distributions in particular.

1.3 New versions

New versions of this document will be periodically posted to comp.os.linux.announce, comp.answers and mail.answers. They will also be added to the various anonymous ftp sites who archive such information including *sunsite.unc.edu:/pub/Linux/docs/HOWTO*.

In addition, you should be generally able to find this document on the Linux WorldWideWeb home page at *http://sunsite.unc.edu/mdw/linux.html*.

1.4 Feedback

I am interested in any feedback, positive or negative, regarding the content of this document via e−mail. Definitely contact me if you find errors or obvious omissions.

I read, but do not necessarily respond to, all e−mail I receive. Requests for enhancements will be considered and acted upon based on that day's combination of available time, merit of the request and daily blood pressure :−)

Flames will quietly go to /dev/null so don't bother.

Feedback concerning the actual format of the document should go to the HOWTO coordinator : Tim Bynum (`howto at wallybox.cei.net`).

1.5 Copyright

1.6 Limited warranty

Of course, I disavow any potential liability for the contents of this document. Use of the concepts, examples, and/or other content of this document is entirely at your own risk.

2.Other sources of information

http://metalab.unc.edu/LDP/HOWTO/Mail−User−HOWTO.html.

2.2 USENET

There is nothing special about configuring and running mail software under Linux (any more). Accordingly, you almost certainly do *NOT* want to be posting generic mail−related questions to the comp.os.linux.* newsgroups.

Don't post in comp.os.linux hierarchy unless it's really linux specific, for example : "Which options was Debian 1.2 sendmail compiled with ?" or "RedHat 5.0 smail crashes when I run it".

Let me repeat that.

There is virtually no reason to post anything mail−related in the comp.os.linux hierarchy any more. There are existing newsgroups in the comp.mail.* hierarchy to handle *ALL* your questions.

IF YOU POST TO COMP.OS.LINUX. FOR NON−LINUX−SPECIFIC QUESTIONS, YOU ARE LOOKING IN THE WRONG PLACE FOR HELP. THE MAIL EXPERTS HANG OUT IN THE PLACES INDICATED ABOVE AND GENERALLY DO NOT RUN LINUX.*

POSTING TO THE LINUX HIERARCHY FOR NON−LINUX−SPECIFIC QUESTIONS WASTES YOUR TIME AND EVERYONE ELSE'S AND IT FREQUENTLY DELAYS YOUR GETTING THE ANSWER TO YOUR QUESTION.

GOOD PLACES are :

```
comp.mail.elm           the ELM mail system.
comp.mail.mh            The Rand Message Handling system.
comp.mail.mime          Multipurpose Internet Mail Extensions.
comp.mail.misc          General discussions about computer mail.
comp.mail.multi-media   Multimedia Mail.
comp.mail.mush          The Mail User's Shell (MUSH).
comp.mail.sendmail      the BSD sendmail agent.
comp.mail.smail         the smail mail agent.
comp.mail.uucp          Mail in the uucp environment.
```

2.3 Mailing Lists

There are many sendmail, smail and qmail mailing lists.

You can find addresses in /usr/doc/the_one_you_have_chosen.

2.4 Other documents from LDP

There is plenty of excellent material provided in the other Linux HOWTO documents and from the Linux DOC project.

In particular, you might want to take a look at the following:

- on your own computer in /usr/doc/ :−)
- the Linux Networking Administrators' Guide
- the Mail Users HOWTO
- the Serial Communications HOWTO
- the Ethernet HOWTO
- the UUCP HOWTO if you're fed via UUCP

2.5 Books

The following is a non−inclusive set of books that will help:

- "`Managing UUCP and USENET`" from O'Reilly and Associates is in my opinion the best book out there for figuring out the programs and protocols involved in being a USENET site.
- "`Unix Communications`" from The Waite Group contains a nice description of all the pieces (and more) and how they fit together.
- "`Sendmail`" from O'Reilly and Associates looks to be the definitive reference on sendmail−v8 and

sendmail+IDA. It's a "must have" for anybody hoping to make sense out of sendmail without bleeding in the process.

- "`The Internet Complete Reference`" from Osborne is a fine reference book that explains the various services available on Internet and is a great source for information on news, mail and various other Internet resources.
- "`The Linux Networking Administrators' Guide`" from Olaf Kirch of the Linux Documentation Project is available on the net and is also published by (at least) O'Reilly and SSC. It makes a fine one−stop shop to learn about everything you ever imagined you'd need to know about Unix networking.

3.How Electronic Mail Works

Now we'll explain the flow of information that typically takes place when two people to communicate by email. Let us suppose that Alice, on her machine `wonderland.com`, wants to send mail to Bob, on his machine `dobbs.com`. Both machines are connected to the Internet.

It helps to know that an Internet mail message consists of two parts; mail headers and a mail body, separated by a blank line. The mail headers contain the source and destination of the mail, a user−supplied subject line, the date it was sent, and various other kinds of useful information. The body is the actual content of the message. Here's an example:

```
From: "Alice" <alice@wonderland.com>
Message-Id: <199711131704.MAA18447@wonderland.com>
Subject: Have you seen my white rabbit?
To: bob@dobbs.org (Bob)
Date: Thu, 13 Nov 1997 12:04:05 -0500 (EST)
Content-Type: text

I'm most concerned.  I fear he may have fallen down a hole.
--
                                                >>alice>>
```

The arrangement and meaning of Internet mail headers are defined by an Internet standard called RFC822.

3.1 Mail between full−time Internet machines

Here's a diagram of the whole process −− I'll explain all the stages and terminology below.

```
                   +---------+          +-------+
+-------+  types   | sending |  calls   |sending|
| Alice |--------->|   MUA   |--------->|  MTA  |::::>::::
+-------+          |         |          |       |      ::   on the
                   +---------+          +-------+      ::   sending
                                                       ::   machine
.......................................................................
                             SMTP                      ::
```

```
::::::::::::::::::::::::::::<::::::::::::::::::::::::::::
::
::   +---------+          +-----+                +-------+
::   |receiving|  calls   |     |  delivers to   | Bob's |
::::>|   MTA   |--------->| LDA |===============>|mailbox|  on the
     |         |          |     |                |       |  receiving
     +---------+          +-----+                +-------+  machine
                                                   |   |
                                                   |   |
                    +----------------<-------------+   |
                    |                                  |
               +---------+         +-------+           |
               |  Bob's  |         | Bob's |<----------+
               | notifier|         |  MUA  |
               +---------+         +-------+
                    |                  |
                    |     +-----+      |
                    +----->| Bob |<----+
                          +-----+
```

To send mail, Alice will invoke a program called a mail user agent (or MUA for short). The MUA is what users think of as `the mailer'; it helps her compose the message, usually by calling out to a text editor of her choice. When she hits the MUA `send' button, her part of the process is done. Later in this HOWTO we will survey popular MUAs.

The MUA she uses immediately hands her message to a program called a mail transport agent (or MTA). Usually this program will be `sendmail`, though some alternative MTAs are gaining popularity and may appear in future Linux distributions. Later in this HOWTO we will also survey MTAs.

The MTA's job is to pass the mail to an MTA on Bob's machine. It determines Bob's machine by analyzing the To header and seeing the `dobbs.com` on the right−hand side of Bob's address. It uses that address to open an Internet connection to Bob's machine. The mechanics of making that connection are a whole other topic; for this explanation, it's enough to know that that connection is a way for Alice's MTA to send text commands to Bob's machine and receive replies to those commands.

The MTA's commands don't go to a shell. Instead they go to a service port on Alice's machine. A service port is a sort of rendezvous, a known place where Internet service programs listen for incoming requests. Service ports are numbered, and Alice's MTA knows that it needs to talk to port 25 on Bob's machine to pass mail.

On port 25, Bob's machine has its own MTA listening for commands (probably another copy of sendmail). Alice's MTA will go through a dialogue with Bob's using Simple Mail Transfer Protocol (or SMTP). Here is what an SMTP dialogue looks like. Lines sent by Alice's machine are shown with S:, responses from Bob's machine are shown with R:.

```
S: MAIL FROM:<alice@wonderland.com>
R: 250 OK
S: RCPT TO:<bob@dobbs.com>
R: 250 OK
S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: From: "Alice" <alice@wonderland.com>
S: Message-Id: <199711131704.MAA18447@wonderland.com>
S: Subject: Have you seen my white rabbit?
S: To: bob@dobbs.org (Bob)
S: Date: Thu, 13 Nov 1997 12:04:05 -0500 (EST)
```

```
S: Content-Type: text
S:
S: I'm most concerned.  I fear he may have fallen down a hole.
S: --
S:                                                  >>alice>>
S: .
R: 250 OK
```

Usually an SMTP command is a single text line and so is its response. The DATA command is an exception; after seeing that, the SMTP listener accepts message lines until it sees a period on a line by itself. (SMTP is defined by the Internet standard RFC821.)

Now Bob's MTA has Alice's message. It will add a header to the message that looks something like this:

```
Received: (from alice@wonderland.com)
        by mail.dobbs.com (8.8.5/8.8.5) id MAA18447
        for bob@dobbs.com; Thu, 13 Nov 1997 12:04:05 -0500
```

This is for tracking purposes in case of mail errors (sometimes a message has to be relayed through more than one machine and will have several of these). Bob's MTA will pass the modified message to a local delivery agent or LDA. On Linux systems the LDA is usually a program called `procmail`, though others exist.

The LDA's job is to append the message to Bob's mailbox. It's separate from the MTA so that both programs can be simpler, and so the MTA can concentrate on doing Internet things without worrying about local details like where the user mailboxes live.

Bob's mailbox will normally be a file called /usr/spool/mail/bob or /var/mail/bob. When he reads mail, he runs his own MUA (mail user agent) to look at and edit that file.

3.2 Notifiers

There's yet another kind of program that is important in the mail chain, though it does not itself read or transmit mail. It's a *mail notifier*, a program that watches your email in−box for activity and signals you when new mail is present.

The original notifier was a pair of Unix programs called biff(1) and comsat(8). The biff program is a front end that enables you to turn on the comsat service. When this service is on, the header of new mail will be dumped to your terminal as it arrived. This facility was designed for people using line−oriented programs on CRTs; it's not really a good idea in today's environment.

Most Unix shells have built−in mailcheck facilities that allow them to function as notifiers in a rather less intrusive way (by emitting a message just before the prompt when new mail is detected). Typically you can enable this by setting environment variables documented on the shell's manual page. For shells in the sh/ksh/bash family, see the MAIL and MAILPATH variables

Systems supporting X come with one of several little desktop gadgets that check for new mail periodically and give you both visible and audible indication of new mail. The oldest and most widely used of these is called *xbiff*; if your Linux has a preconfigured X desktop setup, xbiff is probably on it. See the xbiff(1) manual page for details.

3.3 Mail to part−time Internet machines

If you were reading carefully, you may have noticed that the information flow we described above depends on Alice's machine being able to talk to Bob's machine immediately. What happens if Bob's machine is down, or is up but not connected to the Internet?

If Alice's MTA can't reach Bob's immediately, it will stash Alice's message in a mail queue on `wonderland.com`. It will then retry sending the mail at intervals until an expiration time is reached, at which point a bounce message notifying Alice of the failure will be sent back to her. In the default configuration of the most popular MTA (sendmail), the retry interval is 15 minutes and the expiration time is 4 days.

3.4 Remote mail and remote−mail protocols

Many Linux users nowadays are connected to the Internet via ISPs (Internet Service Providers) and don't have their own Internet domains. Instead they have accounts on an ISP machine. Their mail gets delivered to a mailbox on that ISP machine. But typically these users want to read and reply to their mail using their own machines, which connect to the ISP intermittently using SLIP or PPP. Linux supports remote mail protocols to support this.

Note how this is different from the scenario we discussed in the last section. Mail sitting in a queue awaiting retransmission is not the same as mail dispatched to a server mailbox; mail in a queue is not considered to have been delivered and is subject to expiration, but mail delivered to an ISP server mailbox *is* considered `delivered' and can sit there indefinitely.

A remote−mail protocol allows mail on a server to be pulled across a network link by a client program (this is the opposite of normal delivery in which an MTA pushes mail to a receiving MTA). There are two remote−mail protocols in common use; POP3 (defined by the Internet standard RFC1939) and IMAP (defined by the Internet standard RFC2060). Effectively all ISPs support POP3; a growing number support IMAP (which is more powerful).

Here is what an example POP3 session looks like:

```
S: <client connects to service port 110>
R:    +OK POP3 server ready <1896.697170952@mailgate.dobbs.org>
S:    USER bob
R:    +OK bob
S:    PASS redqueen
R:    +OK bob's maildrop has 2 messages (320 octets)
S:    STAT
R:    +OK 2 320
S:    LIST
R:    +OK 2 messages (320 octets)
R:    1 120
R:    2 200
R:    .
S:    RETR 1
R:    +OK 120 octets
R:    <the POP3 server sends message 1>
R:    .
S:    DELE 1
```

```
R:    +OK message 1 deleted
S:    RETR 2
R:    +OK 200 octets
R:    <the POP3 server sends message 2>
R:    .
S:    DELE 2
R:    +OK message 2 deleted
S:    QUIT
R:    +OK dewey POP3 server signing off (maildrop empty)
S:  <client hangs up>
```

An IMAP session uses different commands and responses, but is logically very similar.

To take advantage of POP3 or IMAP, you need a remote mail client program to pull your mail. Some mail user agents have client capabilities built in (which one supports which is noted below), and the Netscape browser's mail facility supports both POP and IMAP natively.

The main drawback of POP client facilities built into MUAs is that you have to explicitly tell your mailer to poll the server; you don't get notified by xbiff(1) or equivalent, as you would for mail that is either local or delivered by a conventional SMTP `push' connection. Also, of course, not all MUAs can do POP/IMAP, so you may find yourself compromising on other features.

Your Linux probably comes with a program called fetchmail that is designed specifically to talk to remote−mail servers, fetch mail, and feed it into your normal mail delivery path by speaking SMTP to your listener.

Unless you need to keep your mail on the server (for example, because you move around between client machines a lot) fetchmail is probably a better solution than whatever POP/IMAP features your user agent has. Fetchmail can be told to run in background and poll your server periodically, so your xbiff(1) or other mail−notifier program will work as it would for SMTP mail. Also, fetchmail is rather more tolerant of various idiosyncracies and nonstandard server quirks than the clients in MUAs, and has better error recovery.

Here's a diagram showing how both cases (with and without fetchmail) work:

```
                   +---------+          +-------+
+-------+  types   | sending |  calls   |sending|
| Alice |--------->|   MUA   |--------->|  MTA  |::::>::::
+-------+          |         |          |       |      ::
                   +---------+          +-------+      ::   on the
                                                       ::   sending
                             SMTP                      ::   machine
 :::::::::::::::::::::::::::<::::::::::::::::::::::::::
 ::
.::................................................................
 ::
 ::   +---------+         +-----+          +-------+
 ::   |receiving|  calls  |     | delivers | Bob's |
 ::::>|   MTA   |-------->| LDA |=========>|server |::::>::::
      |         |         |     |    to    |mailbox|      ::  on the
      +---------+         +-----+          +-------+      ::   mail
                                                          ::  server
                          POP or IMAP                     ::
 :::::::::::::::::::::::::::<::::::::::::::::::::::::::::::
 ::
```

```
.::.........................................................................
 ::
 ::                  +-----------+
 ::                  |           |
 ::::::>:::::::::::::| fetchmail |::::::::               on the
 ::                  |           |      ::              receiving
 ::                  +-----------+      ::               machine,
 ::                                     ::             with fetchmail
 ::   ::::::::::::::::<:::::::::::::::::::
 ::   ::
 ::   ::   +---------+          +-----+                +-------+
 ::   ::   |receiving|  calls   |     |  delivers to   | Bob's |
 ::   ::::>|   MTA   |--------->| LDA |===============>|mailbox|
 ::        |         |          |     |                |       |
 ::        +---------+          +-----+                +-------+
 ::                                                      |   |
 ::                                                      |   |
 ::                       +----------------<-------------+   |
 ::                       |                                  |
 ::                  +---------+         +-------+           |
 ::                  |  Bob's  |         | Bob's |<----------+
 ::                  | notifier|         |  MUA  |
 ::                  +---------+         +-------+
 ::                       |                  |
.::.........................................................................
 ::                   .   |                  |
 ::     without       .   |                  |
 ::    fetchmail      .   |                  |
 ::                   .   |      +-----+     |
 ::   +----------+    .   +----->|     |<----+
 ::   |  Bob's   |    .          | Bob |
 :::::| POP/IMAP |----.--------->|     |
      |   MUA    |    .          +-----+
      +----------+    .
```

3.5 Mailbox formats

When incoming mail gets appended to a mailbox, it's up to the MTA to provide some kind of delimiters that tell where one message stops and the next begins.

Under Unix, the convention almost all mailers use is that each line beginning with ``From '' (the space is significant) begins a new message. If ``From '' occurs at the beginning of a line in text, a Unix MTA will generally prefix it with a greater−than sign, so it looks like ``>From ''. RFC822 headers follow this From line (which usually continues with the sender name and receipt date).

This convention originated with Unix Version 7, so this kind of mailbox is referred to as a ``V7 mailbox''; it is also sometimes called ``mbox format''. Unless otherwise noted, all programs mentioned in this HOWTO expect this format. It is not, however, quite universal, and tools expecting and generating different formats can confuse each other badly.

The four other formats to know about (and beware of!) are BABYL, MMDF, MH, and qmail maildir. Of these, MMDF is the simplest; it uses a delimiter line consisting four control−As (ASCII 001) characters followed by CR−LF. MMDF was an early and rather crude Internet mail transport; a descendant is still in use on SCO systems.

BABYL is another survival, from an early mail system at MIT. It is still used by Emacs's mail−reader mode.

MH and qmail maildir are `mailbox' formats which actually burst each mailbox into a directory of files, one per message. Running grep on such a `mailbox' will get you nowhere, since all grep will see are the directory bits.

Microsoft Outlook Express .mbx mailboxes can be converted to RFC822 format with mbx2mbox app.

4.Requirements

4.1 Hardware

There are no specific hardware requirements for mail under Linux.

You'll need some sort of 'transport' software to connect to remote systems, which means either TCP/IP or uucp.

This could mean that you need a modem or ethernet card, depending on your setup. In most cases, you'll want the fastest modem you can afford, i.e. V90 57 600 bps currently. In general, you want to have a 16550 UART on your serial board or built into your modem to handle speeds of above 9600 baud.

If you don't know what that last sentence means, please consult the `comp.dcom.modems` group or the various fine modem and serial communications FAQs and periodic postings on USENET.

5.Choosing a Mail Transport Agent

Mail transport agents are the software that transfers mail from your local system to remote systems. It is very seldom necessary to mess with or replace your MTA on a modern Linux, and you're better off not fixing what isn't broken. Nevertheless, here's a survey to get you started on understanding what the tradeoffs are if you decide you need more security or performance than your system's default can offer.

(There are other Unix MTAs besides these, but you are quite unlikely to encounter them on a Linux box.)

Each has its own unique features, but the best compromise is qmail. It features high security (even if vmail is more secure), high speed (even if smail is faster for local uses) and ease of configuration. Of course, feel free to choose any mail software. The information provided here is intended to help you choose well.

Sendmail can be nice for many sites with complicated options, but I think its configuration is too hard for beginners while it is not very secure or very fast, so there is only a **really** outdated sendmail section in this HOWTO.

If you know what you're doing, choose sendmail (and you shouldn't be reading this HOWTO!); otherwise I generally recommend qmail.

Detailed descriptions of these programs follow.

5.1 sendmail

BSD sendmail is the grandaddy of Internet MTAs. It has outlasted a few would−be successors. Most Linux distributions now use it and have it preinstalled.

Sendmail has a long−standing reputation for being an administrator's nightmare −− hard to understand, tricky to configure, rife with security holes. As Internet technology and standards have stabilized, however, many of the sendmail options and configurable rules that gave rise to this reputation have ceased to require per−site tweaking (the effective demise of non−TCP/IP network layers like UUCP has helped a lot). Also, recent sendmail versions have an improved configuration system that insulates you from the legendary hideousness of the sendmail.cf configuration file. Most importantly, sendmail now normally comes preconfigured, and you should never need to touch it unless you have unusual requirements (such as needing to route mail over a non−TCP/IP network).

There is a sendmail home page at http://www.sendmail.org. It includes references to extensive documentation of sendmail, should you actually need to wrestle with custom−configuring it.

Other MTAs, if called as `sendmail', may mimic the semantics of sendmail's command−line options. This is convenient for mail user agents, which often assume they are talking to sendmail.

5.2 smail v3.2

Smail was the first serious attempt to replace sendmail. It has a simpler and much more comprehensible configuration system than sendmail's, and it's fairly secure. Some Linux distributions preinstall it rather than sendmail.

At one time smail's excellent support for mixed TCP/IP and UUCP sites was a major selling point for it, but as UUCP has declined, so has smail. Also, smail is less efficient than sendmail on high−volume connections.

As with sendmail, it is unlikely that you will need to tweak a preinstalled smail configuration.

(Very occasionally you might run across references to an `smail 2.5'. This program has been obsolete for a long time. Don't bother with it.)

5.3 qmail

The qmail program is a sendmail−compatible MTA designed specifically for high security. The author has a standing reward of $500 for publication of the first verifiable security hole; this reward has gone unclaimed since March 1997.

The qmail home page is at http://pobox.com/~djb/qmail.html.

5.4 exim

The exim program is similar to smail3, but with more features. It advertises particular strengths in spam−blocking and support of several virtual hosts (virtual DNS domains) on the same host.

The exim home page is at http://www.exim.org/.

I tried it on my own computer, it looks like a nice merge between smail configuration system and qmail security, moreover it has the advantage of being GPL.

A section explaining how to replace your current MTA by exim will be added soon.

6.Installing Transport Software

6.1 Qmail v1.03

Secured, fast and easy to use, this is my preferred MTA (mail transport agent).

Currently, no distribution comes with qmail preinstalled. We will focus on compiling and installing qmail, since this is the only tricky part: configuration is really straightforward.

Getting qmail

Go to www.qmail.org to download the latest version.

Uncompressing sources

Then decompress it by running:

```
mv qmail.tar.gz /usr/local/src
cd /usr/local/src ; tar -zxvf qmail.tar.gz
```

If you find a bz2 version (new and better compression format), just replace tar with:

```
bunzip2 qmail.tar.bz2
tar -xvf qmail.tar
```

Preparing for compilation

Now enter the qmail directory to examine the configuration defaults:

```
cd qmail; more conf-*
```

You shouldn't need to change any defaults, but you could (for example) specify an alternate installation directory or better compilation flags.

Now run:

```
mkdir /var/qmail
```

to create target dir.

If you haven't installed a Debian distribution, you'll need to add several user IDs for qmail's use: qmail's high security depends on that.

The fact that qmail is divided into modules running each under their own UID makes it much harder for an intruder to break your whole mail system or gain root access by abusing it.

So run:

```
# groupadd nofiles
# useradd -g nofiles -d /var/qmail/alias alias
# useradd -g nofiles -d /var/qmail qmaild
# useradd -g nofiles -d /var/qmail qmaill
# useradd -g nofiles -d /var/qmail qmailp
# groupadd qmail
# useradd -g qmail -d /var/qmail qmailq
```

```
# useradd -g qmail -d /var/qmail qmailr
# useradd -g qmail -d /var/qmail qmails
```

or hand–edit `/etc/passwd` and `/etc/group` to add these users by yourself.

Evan E. reported he had to use "−g groupid" parameter for a vanilla groupadd (Caldera 1.2), else groupadd reported this error : "A group with that name already exists."

For example you can respectively add:

```
qmail:*:2107:
nofiles:*:2108:
```

&

```
alias:*:7790:2108::/var/qmail/alias:/bin/true
qmaild:*:7791:2108::/var/qmail:/bin/true
qmaill:*:7792:2108::/var/qmail:/bin/true
qmailp:*:7793:2108::/var/qmail:/bin/true
qmailq:*:7794:2107::/var/qmail:/bin/true
qmailr:*:7795:2107::/var/qmail:/bin/true
qmails:*:7796:2107::/var/qmail:/bin/true
```

Now you can run

```
make setup check
```

to check your configuration, then :

```
./config
```

to configure qmail.

Attention, your server has to be resolvable by DNS or ./config will get confused.

If you don't have DNS access, you can give your server name directly via :

```
./config-fast foo.bar.com
```

Now you must install some aliases, since /etc/alias is not used by qmail unless you compile and install an optional package.

Here's my setup :

```
File : ".qmail-MAILER-DAEMON"
&postmaster
File : ".qmail-bin"
&root
File : ".qmail-daemon"
&root
File : ".qmail-decode"
&root
File : ".qmail-dumper"
&root
File : ".qmail-games"
&root
File : ".qmail-ingres"
&root
File : ".qmail-mailer-daemon"
&postmaster
File : ".qmail-manager"
&root
File : ".qmail-news"
&root
File : ".qmail-nobody"
&root
File : ".qmail-operator"
&root
File : ".qmail-postmaster"
&root
File : ".qmail-root"
&guylhem
File : ".qmail-system"
&root
File : ".qmail-toor"
&root
File : ".qmail-uucp"
&root
File : ".qmail-uucp-default"
|preline -dr /usr/bin/uux - -r -gC -a"${SENDER:-MAILER-DAEMON}" lm!rmail "($DEFAULT@$HOST)"
```

You need to create each of these file in `~alias`, replacing `&guylhem` in `.qmail-root` by your own login to get root mail.

ATTENTION UUCP USERS !

DO NOT TRUST THE QMAIL FAQ FOR UUCP, USE MY .qmail–uucp–default INSTEAD! ELSE YOU WILL NOT BE ABLE TO SEND ANY MAIL BY YOUR UUCP CONNEXION!

Now you'll need to decide in which format your users will get their mail.

Here's my suggestion :

- For NFS mounted home dirs, use MAILDIR format with a patch for local mail readers (patchs are available on www.qmail.org)
- If no patch is available, prefer MAILFILE format : any mail reader can read a file containing mail, people will only need to create an alias (for bash) or a setenv (for csh) for their mail reader

* Avoid /var/spool/mail/$USER format, too insecure

To fix the default format, read each file in `/var/qmail/boot` then copy the one you best like to `/var/qmail/rc`.

`home` or `proc` are safe choices, but prefer `home` for security reasons.

Configuring qmail

In /var/qmail/control, edit:

defaultdomain, me, plusdomain

* me is you local FQDN (full qualified domain name), for example on my machine it is `barberouge.linux.lmm.com`
* defaultdomain will be added to any host name without dots, including defaulthost, for example you can set it to `localnetwork` so any mail sent to joe@hisbox will be completed to be sent to joe@hisbox.localnetwork instead
* plusdomain is the exception: it is added to any host name that ends with a plus sign, including defaulthost (set in `me`) if it ends with a plus sign.

These 3 examples show you the power and ease of configuration of qmail!

locals, rcpthosts

If you want to support virtual domain names, just put additional names in these files. Any mail you receive for these names will be handled locally.

The difference between `locals` and `rcpthosts` is the latter isn't considered as a local alias, which is useful if you receive mail from some free email address like yahoo.com or lemel.fr while you also send mail to other users of these non local services, i.e. you don't want to handle locally mail send to someone@yahoo.com!

virtualdomains

There can you specify default outgoing mode, for example :

```
#:alias-uucp
```

if you don't want to send outgoing mail by uucp but by smtp (default) or

```
:alias-ucp
```

if you send your outgoing mail by uucp.

Testing qmail

Now it is configured, try:

```
sh -cf '/var/qmail/rc &'
```

to launch qmail (it won't interfere with your local MTA), then:

```
echo to: mylogin | /var/qmail/bin/qmail-inject
```

You should receive this mail in the format you've chosen in `/var/qmail/boot/`.

Removing your other MTA

If this test was successful, just kill your previous MTA:

killall −STOP daemon_name ; if any children are running, you should killall −CONT their_name, wait, killall −STOP again, and repeat ad nauseam.

If there aren't any children, killall −TERM and then killall −CONT.

Remove it (how you can do this depends on the distribution you installed, for example rpm −e −−nodeps on RedHat, Caldera and Suse, or dpkg −r −−force−depends on Debian) then run:

```
# ln -s /var/qmail/bin/sendmail /usr/lib/sendmail
# ln -s /var/qmail/bin/sendmail /usr/sbin/sendmail
```

Now set up qmail−smtpd in /etc/inetd.conf (all on one line):

```
smtp stream tcp nowait qmaild /var/qmail/bin/tcp-env tcp-env /var/qmail/bin/qmail-smtpd
```

If you are using a old non−SYSV−init distribution like redhat, just add to your boot scripts:

```
sh -cf '/var/qmail/rc &'
```

Usually this should be /etc/rc.local but your mileage may vary.

For actual SYSV−init compliant distributions (RedHat, Caldera, Suse, Debian), add this script to /etc/init.d/ or /etc/rc.d/init.d/ :

DEBIAN version:

```
#!/bin/sh

test -x /var/qmail/rc || exit 0

case "$1" in
  start)
     echo -n "Starting mta: "
     sh -cf '/var/qmail/rc &'
     echo "qmail."
     ;;
  stop)
     echo -n "Stopping mta: "
     killall qmail-lspawn
     echo "qmail."
     ;;
  restart)
     echo -n "Restarting mta: "
     killall -HUP qmail-lspawn
     killall -ALRM qmail-lspawn
     echo "qmail."
     ;;
  *)
     echo "Usage: /etc/init.d/qmail {start|stop|restart}"
     exit 1
esac

exit 0
```

REDHAT version:

```
#!/bin/sh
#
# qmail      This shell script takes care of starting and stopping qmail.
#
# description: qmail is a Mail Transport Agent, which is the program \
#              that moves mail from one machine to another.
# processname: qmail
# config: /var/qmail/control/

# Source function library.
. /etc/rc.d/init.d/functions

# Source networking configuration.
. /etc/sysconfig/network
```

```
export PATH=$PATH:/var/qmail/bin

# Check that networking is up.
[ ${NETWORKING} = "no" ] && exit 0

[ -f /usr/sbin/sendmail ] || exit 0

# See how we were called.
case "$1" in
  start)
        # Start daemons.
        echo -n "Starting qmail: "
        qmail-start '|preline procmail' splogger qmail &
        touch /var/lock/subsys/qmail
        echo
        ;;
  stop)
        # Stop daemons.
        echo -n "Shutting down qmail: "
        killproc qmail-lspawn
        echo
        rm -f /var/lock/subsys/qmail
        ;;
  restart)
        $0 stop
        $0 start
        ;;
  status)
        status qmail
        ;;
  *)
        echo "Usage: qmail {start|stop|restart|status}"
        exit 1
esac

exit 0
```

And make symlinks to each /etc/rc.d/rcN.d/, for example:

```
ln -sf /etc/init.d/qmail /etc/rc1.d/K19qmail
```

If the first letter is K, you will kill qmail on this runlevel (1 for single mode or 6 for boot), but if the first letter is S, you will start qmail on this runlevel (each others runlevel).

- How to decide whether you should put a K or a S? Do what the majority of dæmons in this runlevel do!
- What number should you put after K or S? The number next to your network daemon. That means the MTA will be started and stopped respectively after and before the network daemon. Without this, your network will be unreachable while the MTA would expect it to work.

RedHat, Caldera and Suse will use /etc/rc.d/ instead of plain /etc/ for Debian distribution, i.e. /etc/rc.d/rc1.d or /etc/rc.d/init.d for example.

That's all, folks!

No need to reboot (remember, you're using linux, not some other cheap OS!) for the modifications to take effect, just run:

```
killall inetd
init 1
```

To go to single user mode, then:

```
init 2
```

to go back to your default runlevel (indicated in /etc/inittab with initdefault label).

You could also hand−start qmail script but "init" method will show you if qmail script is well positioned, i.e. launched after network scripts but before any program which depends on email to warn you (like inn).

6.2 Smail v3.1

Smail3.1 seems to be a de−facto standard transport agent for uucp−only sites and for some smtp sites. It's easy to configure, it compiles without patching from the sources and it's fairly secure.

Configuring smail

Install the smail binary from your distribution (I recommend you choose this) or get the smail sources and build smail. If you're building smail from sources, you need to have the following in your os/linux file so that 'sed' gives you shell scripts that work properly.

```
CASE_NO_NEWLINES=true
```

Once it's installed, config. files will certainly go in /etc/smail (but your mileage may vary if you use old distributions); let's start editing them!

"config" file

```
# From
smart_path=polux
smart_transport=uux

# To
hostname=barberouge
domains=linux.lmm.com

visible_name=barberouge.linux.lmm.com
uucp_name=barberouge.linux.lmm.com

# max_message_size=512k
# auth_domains=foo.bar
# more_hostnames=barberouge.polux.freenix.fr
```

Well, first, who is feeding you? I'm fed by "polux" via uucp (i.e. uux transport); naturally you need to change this file according to your own situation. For example, you could by fed by "bargw.bar.foobar.com" via "smtp", in that case you don't need a transport file and can define "−transport_file " to indicate you don't need one.

You can also use "postmaster_address = yourname", hide the network topology in outgoing addresses (if you're a gateway) using "visible_name", set which aliases address can also be used for the email you receive, using "more_hostnames".

See smail documentation for more details or the examples in /usr/doc/smail/examples to see if any match your situation.

"directors" file

```
# aliasinclude - expand ":include:filename" addresses produced by alias files
# This entry and the next one are pretty much boiler-plate.  Reasons
# for making significant changes are few.  The sole purpose of these
# is to match and expand addresses of the form:
#       :include:pathname
# which may occur in alias files or mailing-list/forward files
# (produced by any director with a driver of forwardfile).
aliasinclude:
        driver = aliasinclude,          # use this special-case driver
        nobody;                         # associate nobody user with addresses
                                        #  when mild permission violations
                                        #  are encountered
        copysecure,                     # get permissions from alias director
        copyowners,                     # get owners from alias director



# forwardinclude - expand ":include:filename" addrs produced by forward files
forwardinclude:
        driver = forwardinclude,        # use this special-case driver
        nobody;
        copysecure,                     # get perms from forwarding director
        copyowners,                     # get owners from forwarding director
```

```
# aliases - search for alias expansions stored in a database
# This is the standard aliases file.  It is used for generic things,
# like mapping root, postmaster, MAILER-DAEMON and uucp to site
# admins, creating some small system alias expansions, and such.  In
# this site configuration, the aliases file is used mostly for
# machine-specific aliasing/forwarding information.  Global forwarding
# information should be put in the "forward" database.
aliases:
        driver=aliasfile,               # general-purpose aliasing director
        -nobody,                        # all addresses are associated
                                        # with nobody by default, so setting
                                        # this is not useful.
        sender_okay,                    # don't remove sender from expansions
        owner=owner-$user;              # problems go to an owner address
        file=/etc/aliases,
        modemask=002,                   # should not be globally writable
        optional,                       # ignore if file does not exist
        proto=lsearch,                  # unsorted ASCII file


# forward - search for expansions stored in a forwarding database
# This is the subdomain-wide user forwarding database.  Entries are
# maintained here for current or past users, to forward their mail to
# their preferred mail-reading machine.  The forward database is
# shipped around the TCP/IP network as changes are made, to keep the
# network consistent.
#forward:
#       driver = aliasfile,             # general-purpose aliasing director
#       -nobody,                        # all addresses are associated
#                                       # with nobody by default, so setting
#                                       # this is not useful.
#       owner = real-$user;             # problems go to an owner address
#
#       file = /etc/forward,
#       modemask = 002,
#       proto = dbm,                    # use dbm(3X) library for access


# dotforward - expand .forward files in user home directories
# For users that have an entry in the "forward" database, a ".forward"
# file is only used if it is on the "home" machine, as identified in
# the forward database.  If used, it is treated as a list of addresses
# to which mail should be delivered, rather than (or in addition to)
# the user identified in the local address.
dotforward:
        driver = forwardfile,           # general-purpose forwarding director
        owner = postmaster, nobody, sender_okay;

        file = ~/.forward,              # .forward file in home directories
        checkowner,                     # the user can own this file
        owners = root,                  # or root can own the file
        modemask = 002,                 # it should not be globally writable
        caution = daemon:root,          # don't run things as root or daemon
        # be extra careful of remotely accessible home directories
        unsecure = "~uucp:/tmp:/usr/tmp:/var/tmp"


# forwardto - expand a "Forward to " in user mailbox files
# This emulates the V6/V7/System-V forwarding mechanism which uses a
# line of forward addresses stored at the beginning of user mailbox files
```

```
# prefixed with the string "Forward to "
forwardto:
        driver = forwardfile,
        owner = postmaster, nobody, sender_okay;

        file = /var/spool/mail/${lc:user},      # point at user mailbox files
        forwardto,                        # enable "Forward to " function
        checkowner,                       # the user can own this file
        owners = root,                    # or root can own the file
        modemask = 0002,                  # under System V, group mail can write
        caution = daemon:root             # don't run things as root or daemon


# user - match users on the local host with delivery to their mailboxes
user:   driver = user;                    # driver to match usernames
        transport = local                 # local transport goes to mailboxes


# real_user - match usernames when prefixed with the string "real-"
# This is useful for allowing an address which explicitly delivers to a
# user's mailbox file.  For example, errors in a .forward file expansion
# could be delivered here, or forwarding loops between multiple machines
# can be resolved by using a real-username address.  Also, users that
# wish to use mail as a means of transferring data to a machine that
# is not their "home" machine can mail to real-login-name@remote.host.
real_user:
        driver = user;
        transport = local,
        prefix = "real-"                  # for example, match real-root


# lists - expand mailing lists stored in a list directory
# mailing lists can be created simply by creating a file in the
# /etc/smail/lists directory.
lists:  driver = forwardfile,
        caution,                          # flag all addresses with caution
        nobody,                           # and then associate the nobody user
        owner = owner-$user;              # system V sites may wish to use
                                          # o-$user, as owner-$user may be
                                          # too long for a 14-char filename.
        file = lists/${lc:user}           # lists is under $smail_lib_dir


# owners - expand mailing lists stored in a list owner directory
# mailing lists owner lists can be created simply by creating a file
# in the /etc/smail/lists/owner directory.  Mailing list owners
# are sent locally generated errors dealing with a mailing list of the
# same name.  To create an owner list for a mailing list, create a
# file with the name of the list in /etc/smail/lists/owner.  This
# will create a list address of owner-listname, as is used by the
# "lists" director above.
owners: driver = forwardfile,
        caution,                          # flag all addresses with caution
        nobody,                           # and then associate the nobody user
        owner = postmaster;               # system V sites may wish to use
                                          # o-$user, as owner-$user may be
                                          # too long for a 14-char filename.
        prefix = "owner-",
        file = lists/owner/${lc:user}     # lists is under $smail_lib_dir


# request - expand mailing lists stored in a list request directory
```

```
# mailing lists request lists can be created simply by creating a file
# in the /etc/smail/lists/request directory.  Request addresses
# are typically used as a standard address for queries about a mailing
# list.  For example, requests for additions or deletions to a list
# will generally be sent to "list-request", which should be set up to
# forward to the appropriate person or persons.
request: driver = forwardfile,
        caution,                        # flag all addresses with caution
        nobody,                         # and then associate the nobody user
        owner = postmaster;             # system V sites may wish to use
                                        # o-$user, as owner-$user may be
                                        # too long for a 14-char filename.
        suffix = "-request",
        file = lists/request/${lc:user} # lists is under $smail_lib_dir
```

You shouldn't need to change anything here, only mailing list options if you intend to run some using smail, or forwards options if, for example, you want to disable forwarding.

"fidopaths" file

```
.f105.n324.z2.fidonet.org       f105.n324.z2.fidonet.org!%s
.n324.z2.fidonet.org            f105.n324.z2.fidonet.org!%s
.z2.fidonet.org                 f105.n324.z2.fidonet.org!%s
.fidonet.org                    f105.n324.z2.fidonet.org!%s
```

Create such a file only if you're using ifmail and FIDO.

"routers" file

```
# forces - force certain paths
# This database exists as a means of hardcoding the paths to various
# machines or domains.  It is for use in creating temporary tweaks to
# the other routing databases.  To change the database, edit the file
# maps/force.path and type "make" in the maps/ subdirectory.
forces:
        driver = pathalias,             # router to search paths file
        method = /etc/smail/maps/table; # transports are in this file
        file = forcepaths,              # file containing force path info
        proto = lsearch,                # use the sorted path file
        optional,
        reopen                          # close when not being used


uucp_neighbors:
        driver=uuname,                  # use a program which returns neighbors
        transport=uux;
        cmd="/usr/bin/uuname -a",       # specifically, use the uuname program
```

```
#        domain=uucp                    # strip ending ".uucp"


# smart_host - a partially specified smarthost director
# If the config file attribute smart_path is defined as a path from the
# local host to a remote host, then hostnames not matched otherwise will
# be sent off to the stated remote host.  The config file attribute
# smart_transport can be used to specify a different transport.
# If the smart_path attribute is not defined, this router is ignored.
smart_host:
        driver = smarthost,             # special-case driver
        transport = uux                 # by default deliver over UUCP
#       path=phreak


# ifmail - to send mails to fidonet and vice versa
ifmail:
        driver=pathalias,
        transport=ifmail;
        file=fidopaths,
        proto=lsearch
```

You should only include ifmail chapter if you use ifmail for FIDO mails. Note you can also change transport mode from "uux" (ie UUCP) to, for example, "smtp" or even 'hardcode the paths to various machines or domains' in "/etc/smail/maps/table".

This is useful if you want outgoing mail for your local network to be delivered immediately, since there's no need for it to be routed to your uucp connexion of your internet access.

"transports" file

```
# local - deliver mail to local users
# Tell smail to append directly to user mailbox files in the /var/spool/mail
# directory.
#local: driver = appendfile,            # append message to a file
#       -return_path,                   # include a Return-Path: field
#       local,                          # use local forms for delivery
#       from,                           # supply a From_ envelope line
#       unix_from_hack;                 # insert > before From in body
#
#       file = /var/spool/mail/${lc:user},      # use this location for Linux
#                                               # Note, mail spool must be 1777
#       file = ~/mailfile,              # use this location for better security
#       group = mail,                   # group to own file for System V
#       mode = 0660,                    # under System V, group mail can access
#       suffix = "\n",                  # append an extra newline
#       append_as_user,


# This allows each user to have a ~/.procmailrc file to control filtering
# of mail and saving mail from mail lists in separate mailboxes if they wish.
local:  +inet,
        -uucp,
```

```
        driver = pipe,                  # append message to a file
        return_path,                    # include a Return-Path: field
        local,                          # use local forms for delivery
        from,                           # supply a From_ envelope line
        unix_from_hack;                 # insert > before From in body

        cmd = "/usr/bin/procmail",      # use procmail for local delivery
        parent_env,                     # environment info from parent addr
        pipe_as_user,                   # use user-id associated with address
        umask = 0022,                   # umask for child process
#       -ignore_status,                 # exit status should be believed
#       -ignore_write_errors,           # retry on broken pipes


# pipe - deliver mail to shell commands
# This is used implicitly when smail encounters addresses which begin with
# a vertical bar character, such as "|/usr/lib/news/recnews talk.bizarre".
# The vertical bar is removed from the address before being given to the
# transport.
#pipe:  driver = pipe,                  # pipe message to another program
#       return_path, local, from, unix_from_hack;
#
#       cmd = "/bin/sh -c $user",       # send address to the Bourne Shell
#       parent_env,                     # environment info from parent addr
#       pipe_as_user,                   # use user-id associated with address
#       umask = 0022,                   # umask for child process
#       -log_output,                    # do not log stdout/stderr
#       ignore_status,                  # exit status may be bogus, ignore it
#       ignore_write_errors,            # ignore broken pipes


# file - deliver mail to files
# This is used implicitly when smail encounters addresses which begin with
# a slash or squiggle character, such as "/usr/info/list_messages" or
# perhaps "~/Mail/inbox".
#file:  driver = appendfile,
#       return_path, local, from, unix_from_hack;
#
#       file = $user,                   # file is taken from address
#       append_as_user,                 # use user-id associated with address
#       expand_user,                    # expand ~ and $ within address
#       check_path,
#       suffix = "\n",
#       mode = 0644


# uux - deliver to the rmail program on a remote UUCP site
#
# As many as five recipient addresses will be delivered to the remote
# host in one UUCP transaction.
uux:    driver = pipe,
        -uucp,
        inet,
#       uucp,                           # use UUCP-style addressing forms
        from,                           # supply a From_ envelope line
        max_addrs = 5,                  # at most 5 addresses per invocation
        max_chars = 200;                # at most 200 chars of addresses
# the -r flag prevents immediate delivery, parentheses around the
# $user variable prevent special interpretation by uux.
        cmd = "/usr/bin/uux - -r -g$grade $host!rmail $((${strip:user})$)",
#        cmd="/usr/bin/uux - $host!rmail $(($user)$)",
        ignore_write_errors,            # ignore broken pipes
```

```
        umask = 0022,
#       pipe_as_sender,


# uux_one_addr - deliver mail over UUCP to a remote host that can take
#               one address at a time.
# This is often necessary when delivering to a site running an unmodified
# version of 4.1BSD.
uux_one_addr:
        driver = pipe,
        uucp,                           # use UUCP-style addressing forms
        from;                           # supply a From_ envelope line
# the -r flag prevents immediate delivery
        cmd = "/usr/bin/uux - -r -g$grade $host!rmail (${strip:user})",
        umask = 0022,
        pipe_as_sender


queueonly:
        driver = pipe;                  # send the message to a pipe
        cmd = "/usr/lib/sendmail -Q -f $sender -bm $user",
                                        # use getmail for local delivery
        user=root,                      # execute getmail as "root"
        group=mail,                     # execute getmail as "mail"
        parent_env,                     # environment info from parent addr
        -pipe_as_user,                  # use user-id associated with address
        umask = 0007,                   # umask for child process

# to deliver the message.  The smtp transport is included only if BSD
# networking exists.
# The uucp attribute can be specified for transfers within the UUCP
# zone.  The inet attribute must be specified for transfers within the
# Internet.
# NOTE: This is hardly optimal, a backend should exist which can handle
#       multiple messages per connection.
# ALSO: It may be necessary to restrict max_addrs to 100, as this is the
#       lower limit SMTP requires an implementation to handle for one
#       message.
smtp:   driver=tcpsmtp,
        inet,                           # if UUCP_ZONE is not defined
#       uucp,                           # if UUCP_ZONE is defined
        -max_addrs, -max_chars;         # no limit on number of addresses

        short_timeout=5m,               # timeout for short operations
        long_timeout=2h,                # timeout for longer SMTP operations
        service=smtp,                   # connect to this service port
# For internet use: uncomment the below 4 lines
       use_bind,                       # resolve MX and multiple A records
       defnames,                       # use standard domain searching
       defer_no_connect,               # try again if the nameserver is down
       local_mx_okay,                  # fail an MX to the local host


ifmail:
        from,received,max_addrs=5,max_chars=200,
        driver=pipe;
        pipe_as_sender,
        cmd="/usr/local/bin/ifmail -x9 -r$host $((${strip:user})$)"
```

You should include an ifmail chapter only if you use ifmail for FIDO mail. Apart from that, you shouldn't need to edit anything in this file which defines transport agents (like uux, smtp ...) you can use as parameters in other config. files.

Note I commented out some parts, like "pipes" or "file", to enhance security.

"maps/" directory

It contains `map` and `table` files:

First, `map` file

```
#N      foo.bar foo2.bar2
#S      AT 486/RedHat Linux 1.2.13
#O      organization
#C      contact
#E      administration (email)
#T      phone
#P      address
#R
#U      hosts connected via uucp
#W      created/edited by
#
hname polux

hname linux.eu.org

hname = polux
hname = polux.linux.eu.org
```

Once again, edit this file to match you situation (I'm fed by polux.linux.eu.org).

Now `table` file

```
*       uux
```

You can define different transports to different paths, for example "smtp" for the machines in your local network, "uux" (i.e. uucp) for the rest of the world or vice−versa (I'm using uucp for any outgoing mail, therefore I use "*"!).

Other good examples

The previous files are the one I currently use for my site, you shouldn't encounter any problem using them as samples/basis for your own files.

The following files are provided only as good examples to configure smail a different way.

```
#ident "@(#) transports,v 1.2 1990/10/24 05:20:46 tron Exp"

# See smail(5) for a complete description of the contents of this file.

# local - deliver mail to local users
#
# Tell smail to append directly to user mailbox files in the /usr/mail
# directory.
local:  driver = appendfile,            # append message to a file
        return_path,                    # include a Return-Path: field
        local,                          # use local forms for delivery
        from,                           # supply a From_ envelope line
        unix_from_hack;                 # insert > before From in body

        file = /usr/mail/${lc:user},    # use this location for System V
        group = mail,                   # group to own file for System V
        mode = 0660,                    # under System V, group mail can access
        suffix = "\n",                  # append an extra newline
        append_as_user,

# pipe - deliver mail to shell commands
#
# This is used implicitly when smail encounters addresses which begin with
# a vertical bar character, such as "|/usr/lib/news/recnews talk.bizarre".
# The vertical bar is removed from the address before being given to the
# transport.
pipe:   driver = pipe,                  # pipe message to another program
        return_path, local, from, unix_from_hack;

        cmd = "/bin/sh -c $user",       # send address to the Bourne Shell
        parent_env,                     # environment info from parent addr
        pipe_as_user,                   # use user-id associated with address
        umask = 0022,                   # umask for child process
        -log_output,                    # do not log stdout/stderr
        ignore_status,                  # exit status may be bogus, ignore it
        ignore_write_errors,            # ignore broken pipes

# file - deliver mail to files
#
# This is used implicitly when smail encounters addresses which begin with a
# slash or squiggle character, such as "/usr/info/list_messages" or perhaps
# "~/Mail/inbox".
file:   driver = appendfile,
        return_path, local, from, unix_from_hack;

        file = $user,                   # file is taken from address
        append_as_user,                 # use user-id associated with address
        expand_user,                    # expand ~ and $ within address
        suffix = "\n",
        mode = 0644

# uux - deliver to the rmail program on a remote UUCP site
#
# As many as five recipient addresses will be delivered to the remote host in
# one UUCP transaction.
```

```
uux:    driver = pipe,
        uucp,                           # use UUCP-style addressing forms
        from,                           # supply a From_ envelope line
        max_addrs = 5,                  # at most 5 addresses per invocation
        max_chars = 200;                # at most 200 chars of addresses

        # the -r flag prevents immediate delivery, parentheses around the
        # $user variable prevent special interpretation by uux.
        cmd = "/usr/bin/uux - -r -g$grade $host!rmail $((${strip:user})$)",
        umask = 0022,
        pipe_as_sender

# uux_one_addr - deliver mail over UUCP to a remote host that can take one
# address at a time.
#
# This is often necessary when delivering to a site running an unmodified
# version of 4.1BSD.
uux_one_addr:
        driver = pipe,
        uucp,                           # use UUCP-style addressing forms
        from;                           # supply a From_ envelope line

        # the -r flag prevents immediate delivery
        cmd = "/usr/bin/uux - -r -g$grade $host!rmail (${strip:user})",
        umask = 0022, pipe_as_sender

# demand - deliver to a remote rmail program, polling on demand
demand: driver = pipe,
        uucp, from, max_addrs = 5, max_chars = 200;

        # with no -r flag, try to contact remote site immediately
        cmd = "/usr/bin/uux - -g$grade $host!rmail $(($user)$)",
        umask = 0022, pipe_as_sender

# uusmtp - deliver to the rsmtp program on a remote UUCP site
#
# Deliver using a simple Batched SMTP protocol to the remote machine.
# This allows much more arbitrary addresses to be used.  It also
# removes the limit on recipient addresses per invocation of uux.
uusmtp: driver = pipe,
        bsmtp,                          # send batched SMTP commands
        -max_addrs,                     # there is no limit on the number or
        -max_chars;                     #   total size of recipient addresses.

        # supply -r to prevent immediate delivery, the recipient addresses
        # are stored in the data sent to the standard input of rsmtp.
        cmd = "/usr/bin/uux - -r -g$grade $host!rsmtp",
        umask = 0022, pipe_as_sender

# demand_uusmtp - deliver to a remote rsmtp program, polling on demand
demand_uusmtp:
        driver = pipe,
        bsmtp, -max_addrs, -max_chars;

        # with no -r flag, try to contact remote site immediately
        cmd = "/usr/bin/uux - -g$grade $host!rsmtp",
        umask = 0022, pipe_as_sender

# smtp - deliver using SMTP over TCP/IP
#
# Connect to a remote host using TCP/IP and initiate an SMTP conversation to
# deliver the message.  The smtp transport is included only if BSD networking
```

```
# exists.

# NOTE: It may be necessary to restrict max_addrs to 100, as this is the
#       lower limit SMTP requires an implementation to handle for one
#       message.
smtp:   driver = smtp,
        -max_addrs,
        -max_chars
```

```
#ident "@(#) table,v 1.2 1990/10/24 05:20:31 tron Exp"

# This file names the transports that are to be used in delivering
# to specific hosts from bargw.

#host           transport
#--------       ---------
curdsgw         demand_uusmtp   # deliver using batched SMTP
oldbsd          uux_one_addr    # 4.1BSD sites cannot take more than one addr
sun             demand          # call sun when their is mail to send
*               uux             # for all others, poll at intervals
```

Restarting inetd

To run smail as a smtp daemon, add one of the following to /etc/inetd.conf:

```
smtp stream tcp nowait  root  /usr/bin/smtpd smtpd
```

```
or:
```

```
smtp stream tcp nowait  root  /usr/sbin/tcpd  /usr/sbin/in.smtpd
```

Outgoing mail gets sent automatically, when using elm.

Smail with smtp

Generally, ISPs use smtp, therefore you shouldn't have any problems sending your mail. If your internet link is down when you send mail, then the mail sits in "`/var/spool/smail/input`". When the link next comes up, "`runq`" is run which causes the mail to be sent. However, receiving mail is **the** problem since your provider has many clients to look after, not only you!

Usually, you can retrieve your mail via the POP protocol, see POP section below.

6.3 OUTDATED SECTION: Sendmail+IDA

For big sites, sendmail is worth choosing, due to the "incredible ease of use", (very relative feeling when you know qmail) but you must decide which you want between sendmail+IDA and sendmail 8.x:

- If you use an old kernel (1.0): sendmail+IDA
- If you use a not so old kernel (1.2): sendmail+IDA and source code editing
- Recent kernel (2.0) will choose sendmail 8.x

Remember, linux newbies or people concerned by security / ease of configuration should rather try using smail or qmail, which are easier to use and safer.

Source installation

If your distribution doesn't provide you with a ready−to−install sendmail package (.rpm for RedHat, Caldera and Suse, .deb for Debian) just download the sources and run:

- cd / ; tar −zxvf sendmail5.67b+IDA1.5.tgz

- cd to /usr/local/lib/mail/CF and copy the sample.m4 local.m4 file to "yourhostname.m4".

Edit out the distributed hostname, aliases, smarthost and put in the correct one for your site. The default file is for a uucp−only site (no longer in 8.x) who has domainized headers and who talks to a smart host. Then "make yourhostname.cf" and move the resulting file to /etc/sendmail.cf

- if you are uucp−only, you do *NOT* need to create any of the tables mentioned in the README.linux file.

You'll just have to touch the files so that the Makefile works. Just edit the .m4 file, make sendmail.cf and start testing it.

- if you're uucp−only and you talk to sites in addition to your "smart−host", you'll need to add uucpxtable entries for each (or mail to them will also go through the smart host) and run dbm against the revised uucpxtable.

- If you run Rich Braun's original binary distribution of 5.67a, you'll need to freeze the configuration if you change your .cf file with "/usr/lib/sendmail −bz" to make the changes take effect.

You should also update your version to at least 5.67b since there is a nasty security hole in 5.67a and earlier. Another nice thing is that if you have mail.debug set and you run syslogd, your incoming and outgoing mail messages will get logged. See the "/etc/syslog.conf" file for details.

The sources for sendmail+IDA can be found at vixen.cso.uiuc.edu ; they require no patching to run under Linux if you're running something like a kernel of 1.00.

If you're running a kernel > 1.1.50, you get the fun of reversing most of the Linux−specific patches that are now in the vanilla sources. (I *did* told you this sendmail was only for old kernels:−)

It's extremely obvious where this needs to be done: just type "make" and when it blows up, go to that line in the sources and comment out the Linux−specific code that's in there.

If you're going to run sendmail+IDA, I strongly recommend you go to the sendmail5.67b+IDA1.5 version since all required Linux−specific patches are now in the vanilla sources and several security holes have been plugged that WERE (!!!) in the older version you may have grabbed or built before about December 1st, 1993.

Now linux kernel is 2.0, you should use sendmail 8.x instead of sendmail+IDA, but I already told you'd better choose sendmail 8.x:−)

The sendmail.m4 file

Sendmail+IDA requires you to set up a `sendmail.m4` file rather than editing the `sendmail.cf`file directly. The nice thing about this is that it is simple to set up mail configurations that are extremely difficult (if not totally impossible for most people to set up correctly) in smail or traditional sendmail.

The sendmail.m4 file that corresponds to the above smail example looks like the following:

```
dnl #------------------ SAMPLE SENDMAIL.M4 FILE ------------------
dnl #
dnl # (the string 'dnl' is the m4 equivalent of commenting out a line)
dnl # (well, not exactly, but use it for this purpose if you must :-)
dnl # you generally don't want to override LIBDIR from the compiled in paths
dnl #define(LIBDIR,/usr/local/lib/mail)dnl    # where all support files go
define(LOCAL_MAILER_DEF, mailers.linux)dnl    # mailer for local delivery
define(POSTMASTERBOUNCE)dnl                   # postmaster gets bounces
define(PSEUDODOMAINS, BITNET UUCP)dnl         # don't try DNS on these
dnl #
dnl #-------------------------------------------------------------
dnl #
dnl # names we're known by
define(PSEUDONYMS, myhostname.subdomain.domain myhostname.UUCP)
dnl #
dnl # our primary name
define(HOSTNAME, myhostname.subdomain.domain)
dnl #
dnl # our uucp name
define(UUCPNAME, myhostname)dnl
dnl #
dnl #-------------------------------------------------------------
dnl #
define(UUCPNODES, |uuname|sort|uniq)dnl       # our uucp neighbors
define(BANGIMPLIESUUCP)dnl                    # make certain that uucp
define(BANGONLYUUCP)dnl                       #  mail is treated correctly
define(RELAY_HOST, my_uucp_neighbor)dnl       # our smart relay host
```

```
define(RELAY_MAILER, UUCP-A)dnl              # we reach moria via uucp
dnl #
dnl #--------------------------------------------------------------------
dnl #
dnl # the various dbm lookup tables
dnl #
define(ALIASES, LIBDIR/aliases)dnl           # system aliases
define(DOMAINTABLE, LIBDIR/domaintable)dnl   # domainize hosts
define(PATHTABLE, LIBDIR/pathtable)dnl       # paths database
define(GENERICFROM, LIBDIR/generics)dnl      # generic from addresses
define(MAILERTABLE, LIBDIR/mailertable)dnl   # mailers per host or domain
define(UUCPXTABLE, LIBDIR/uucpxtable)dnl     # paths to hosts we feed
define(UUCPRELAYS, LIBDIR/uucprelays)dnl     # short-circuit paths
dnl #
dnl #--------------------------------------------------------------------
dnl #
dnl # include the 'real' code that makes it all work
dnl # (provided with the source code)
dnl #
include(Sendmail.mc)dnl                        # REQUIRED ENTRY!!!
dnl #
dnl #------------ END OF SAMPLE SENDMAIL.M4 FILE -------
```

Defining a local mailer

Unlike most Unix distributions, Linux did not come with a local mail delivery agent by default.

Slackware did! Well at least it is offered by the easy−to−use−but−longwinded installation script. It uses procmail.

Now, deliver or procmail is generally installed, with a default sendmail setup to handle local mail, so no complexity will be added to this already very complex setup. I recommend using the commonly available `deliver` or `procmail` programs, which can be optional packages in a some Linux distributions.

In order to do so, you need to define a `LOCAL_MAILER_DEF` in the `sendmail.m4` file that points to a file that looks like:

```
# -- /usr/local/lib/mail/mailers.linux --
#     (local mailers for use on Linux )
Mlocal, P=/usr/bin/deliver, F=SlsmFDMP, S=10, R=25/10, A=deliver $u
Mprog,  P=/bin/sh,       F=lsDFMeuP,   S=10, R=10, A=sh -c $u
```

There is a also built−in default for `deliver` in the `Sendmail.mc` file that gets included into the `sendmail.cf` file. To specify it, you would not use the `mailers.linux` file but would instead define the following in your sendmail.m4 file:

```
dnl --- (in sendmail.m4) ---
define(LOCAL_MAILER_DEF, DELIVER)dnl       # mailer for local delivery
```

Unfortunately, Sendmail.mc assumes deliver is installed in /bin, which is not the case with Slackware1.1.1 (which installs it in /usr/bin). In that case you'd need to either fake it with a link or rebuild deliver from sources so that it resides in /bin. Please note procmail is generally better than deliver, for example for mail filtering.

The sendmail+IDA dbm tables

Setting up special behavior for sites or domains is done through a number of optional `dbm` tables rather than editing the `sendmail.cf` file directly.

Refer to the July−1994 issue of `Linux Journal` (if you can still find it:−), to the docs in the sources, or to the sendmail chapter in the newest version of the Linux Documentation Project `Networking Administration Guide` which will be available real−soon−now for more details.

- mailertable − defines special behavior for remote hosts or domains.
- uucpxtable − forces UUCP delivery of mail to hosts that are in DNS format.
- pathtable − defines UUCP bang−paths to remote hosts or domains.
- uucprelays − short−circuits the pathalias path to well−known remote hosts.
- genericfrom − converts internal addresses into generic ones visible to the outside world.
- xaliases − converts generic addresses to/from valid internal ones.
- decnetxtable − converts RFC−822 addresses to DECnet−style addresses.

So which entries are really required?

When not using any of the optional dbm tables, sendmail delivers mail via the `RELAY_HOST` and `RELAY_MAILER)` defined in the sendmail.m4 file used to generate sendmail.cf. It is easily possible to override this behavior through entries in the domaintable or uucpxtable.

A generic site that is on Internet and speaks Domain Name Service, or one that is UUCP−only and forwards all mail via UUCP through a smart `RELAY_HOST`, probably does not need any specific table entries at all.

Virtually all systems should set the `DEFAULT_HOST` and `PSEUDONYMS` macros, which define the canonical site name and aliases it is known by.

If all you have is a relay host and relay mailer, you don't need to set these defaults since it works automagically. UUCP hosts will probably also need to set `UUCPNAME` to their official UUCP name.

They will also probably set `RELAY_MAILER` and `RELAY_HOST` which enable smart−host routing through a mail relay.

The mail transport to be used is defined in `RELAY_MAILER` and should usually be UUCP−A for UUCP sites. If your site is SMTP−only and talks `Domain Name Service', you would change the `RELAY_MAILER`.

If you're a SLIP site, you might want to take the easy way out and just forward all outgoing mail to your service provider to do the right thing with. To do so, you'd want to define ISOLATED_DOMAINS and VALIDATION_DOMAINS to be your domain, you'd also want to define RELAY_HOST to be your service provider and RELAY_MAILER to be TCP. Of course, you want to ask permission before you set any system up as your general purpose relay.

6.4 Sendmail 8.x

Sendmail 8.7.x from Berkeley was the latest major revision after sendmail5. It had wonderful built−in support for building under Linux : just "make linux" and all was set.

You'll probably be best served by grabbing one of the various binary distributions off of the usual Linux archive sites rather than fighting things like Berkeley dbm yourself.

There's a nice distribution of sendmail 8.6.12 from Jason Haar − `j.haar at lazerjem.demon.co.uk` on sunsite.unc.edu in /pub/Linux/system/Mail/delivery/sendmail−8.6.12−bin.tgz that has the source documentation and a very nice quickie description of how to run sendmail v8 for common configurations.

The bottom line with sendmail v8 is that you want to configure the bare minimum necessary to get the job done ; the following is an example that should get you close at least.

A sample 8.7.x mc file

Much like sendmail+IDA, sendmail v8 uses m4 to process a config file into a full sendmail.cf that sendmail uses. The following is my current mc file for my site (ppp to Internet for outgoing mail, uucp for incoming mail).

```
dnl divert(-1)
#-----------------------------------------------------------------------
#
# this is the .mc file for a linux host that's set up as follows:
#
#       - connected to Internet for outbound mail (ppp here)
#       - connected via UUCP for incoming mail
#       - domainized headers
#       - no local mailer (use 'deliver' instead)
#       - no DNS running so don't canonicalize outgoing via DNS
#       - all non-local outbound mail goes to the RELAY_HOST over smtp
#           (we run ppp and let our service provider do the work)
#
#                                        vds 3/31/95
#
#-----------------------------------------------------------------------
include(`../m4/cf.m4')
VERSIONID(`linux nodns relays to slip service provider smarthost')dnl
Cwmyhostname.myprimary.domain myhostname.UUCP localhost
OSTYPE(linux)
FEATURE(nodns)dnl
FEATURE(always_add_domain)dnl
FEATURE(redirect)
FEATURE(nocanonify)
dnl MAILER(local)dnl
MAILER(smtp)dnl
MAILER(uucp)dnl
define(`RELAY_HOST', smtp:my.relay.host.domain)
define(`SMART_HOST', smtp:my.relay.host.domain)
```

```
define(`UUCP_RELAY', smtp:my.relay.host.domain)
define(`LOCAL_MAILER_PATH', `/bin/deliver')
define(`LOCAL_MAILER_ARGS', `deliver $u')
```

Sendmail v8 tidbits

There are a few differences I suppose to the 'IDA bigots' among us. So far, I've found the following:

Instead of 'runq', you type 'sendmail –q' to run the queue!

6.5 Local Delivery Agents

Unlike most operating systems, Linux did not have mail "built–in": you needed a program to deliver the local mail, like "lmail", "procmail" or "deliver".

However, every recent distribution includes a local mailer now!

Documentation for how to use either for local delivery is in the sendmail5.67b+IDA1.5 binary release (on sunsite) mentioned above.

7.User Agent Administration

7.1 Mutt

You should have no problem compiling, installing, or running mutt. Users of qmail can either get the patch or run it with –f flag to read their local mail folder.

If mutt bothers you with an "unknown terminal error" after a distribution upgrading, just recompile it.

7.2 Elm

Elm compiles, installs and runs flawlessly under Linux. For more information, see the elm sources and installation instructions. Elm and filter need to be mode 2755 (group mail) with /var/spool/mail mode 775 and group mail.

Qmail users can get a patch to use nifty qmail features, or will run elm with the −f flag to point to their local mail folder.

One thing you want to be aware of is that if you have Elm compiled to be MIME−able, you need metamail installed and in the standard path or Elm will not be able to read MIME mail you've received. Metamail is available on `thumper.bellcore.com` and of course via "archie".

If you use a binary distribution, you'll need to create a "/usr/local/lib/elm/elm.rc" file to override the compiled−in hostname and domain information:

- replace "subdomain.domain" with your domain name replace
- "myhostname" with you un−domainized hostname replace

```
#---------- /usr/local/lib/elm/elm.rc ------------------
#
# this is the unqualified hostname
hostname = myhostname
#
# this is the local domain
hostdomain = subdomain.domain
#
# this is the fully qualified hostname
hostfullname = myhostname.subdomain.domain
#
#-------------------------------------------------------
```

One thing you want to be aware of is that if you have Elm compiled to be MIME enabled, you need metamail installed and in your path or Elm will not be able to read MIME mail you've received. Metamail is available on `thumper.bellcore.com` and of course via "archie".

In the "too cool to be true" category, there is a distribution of Elm−2.4.24 that is "PGP−aware". To try it, grab the file `ftp://ftp.viewlogic.com/pub/elm-2.4pl24pgp3.tar.gz`, which is elm2.4.24 with PGP hooks added. You configure and build it the same way you do normal Elm, which means you probably need to add the patches mentioned above. For what it's worth, I run it here and like it a lot. Of course, there must be more recent versions available, including elm−ME+.

While this item is not Linux−specific, it's perceived (wrongly) to be a nagging Elm bug nevertheless. We've heard that Elm sometimes fails with a message that it's unable to malloc() some massive number of bytes. The identified workaround is to remove the post−processed global mail aliases (aliases.dir and aliases.pag).

`THIS IS NOT A BUG IN ELM`, it's an error in configuration of Elm by whomever you got your binary distribution of Elm from.

Elm has an enhanced and non−compatible, format for aliases ; you need to ensure that the path Elm uses for aliases is different from the path sendmail/smail uses. From the volume of reports of this problem, it's apparent that at least one major distribution 'on the street' has in the past been misconfigured. (from `scot at catzen.gun.de (Scot W. Stevenson)`)

The current metamail package requires csh for some of its scripts. Failure to have csh (or tcsh) will cause most interesting errors...

7.3 Mailx

If you don't have a local `mailx` program, save yourself the pain −− just go and grab the mailx kit from Slackware 2.1.0 or later, which has a nice implementation of mailx5.5. If you're into building from sources, mailx v5.5 compiles without patching under Linux if you have `"pmake"` installed.

If anybody is still using it, I strongly recommend removing the old "edmail" stuff from SLS1.00 and replacing it with mailx.

8.Handling remote mail

This section describes using POP or IMAP to handle remote mail.

Other options include nfs−mounting the spool partition on client machines (Danger Will Robinson! Is everyone using the same lock method?) or using a mail−to−web gateway (quite popular now).

8.1 History

On a workstation network, mail has always been a problem:

- Either you use "user@computer.foo.com" with problems when "computer" is down, making your network known to the people outside, having different addresses for a same person switching to another computer, ...
- Or you take a mail hub, "mailhost.foo.com" with rules for rewriting, so every user seems to post from the same address, even if they are on different computers.

But in that case, how can users read their mail?

Using a rsh with elm? :−)

It would overload our mail hub! One method was forwarding or UUCP, smtp, etc. but it's too complicated.

Then came POP/IMAP, both with security problems at the beginning, (now fixed using ssh on new versions): a mail program has sometimes to be set locally (like qmail, smail or vmail if, for example, you use elm, but mozilla will avoid that!) however, getting and sending Email is simpler.

8.2 Getting mail

Here come POP's main drawbacks:

- the password is sent as a clear text on the network,
- you must choose a POP−aware mailer; many do now (like Pine, Emacs, Mozilla, Netscape, Mutt, IE, Pegasus, Eudora, Claris...),
- when a user may roam (read mail from different machines) having e−mail popped on the computer used yesterday can be a nuisance,
- some POP servers (e.g. qpopper, ipop3d) on high−use servers can load the machine significantly. Consider controlling options (such as not leaving mail on the server) and/or changing the pop server (e.g. cucipop), as well as avoiding running it from inetd.

The password problem can be solved creating a crypted "channel" to have POP on it or using APOP or RPOP extensions. The mail reader problem can be solved either by changing mail reader (don't underestimate the effort required to re−educate users!) or by using a POP "mail sucker" with a local mail program.

IMAP can be preferable to POP in various situations like remote (and especially roaming) access, while you restrict POP to a LAN where snooping of passwords isn't so much of a concern. Mark Aitchison reported a solution here is to use hosts.deny and hosts.allow files (please see Net−3 HOWTO ; this assumes you are starting pop from inet).

The policy of leaving mail on the server or not has implications for server disk space and easier backup/security of the mail, as well as allowing roaming, so the best solution depends on the type of organization. Of course, this will not ensure your mail can't be read, but nobody will be able to delete it ; if all your mail is pgp encrypted this is a better solution.

Here are some pop programs worth trying:

- gwpop (a Good Way to POP) is very protected since it creates a crypted "channel" and puts mail directly in the "spool" ; however, it depends on Perl.
- popclient, simple to use: For example if your login is john and your password PrettySecret, you will run:

  ```
  $ popclient -3 -v mail.acme.net -u john -p "PrettySecret" -k -o JOHN-INET-MAIL
  ```

  It is strongly discouraged in case of multi−user machine; other user can see your password by, for example with "ps auxw"
- fetchmail, which is actively supported and incredibly simple to use: it is configured in `~/.fetchmailrc`, so you only need to run `fetchmail` when you want to retrive your mail. Here's my .fetchmailrc:

  ```
  poll mail.server protocol pop3:
          forcecr
          password PrettySecret;
  ```

Don't forget to "chmod 600 /.fetchmailrc" or fetchmail will ask for it. Please note that the forcecr option is needed to use fetchmail with qmail, which strictly respects RFCs.

8.3 Sending mail

For this, you must use smtp−aware mail software, like qmail, smail, vmail or mozilla (this one does everything: mail reader, POP receive, smtp send!)

Go to one of the previous sections to install and configure the one you like best. Then, when you will reach "Testing", try to send some mail to a local account on the mail hub.

8.4 Reading mail

If your program doesn't do everything itself, you can install elm, pgp, mush, pine ... many good programs are freely available for linux platforms!

8.5 Testing

To check whether your mail server has pop, try:

```
$ telnet mailhost 110
```

If it works, you will get something like "OK Pop server (...) starting": type "quit"!

To install a ssh crypted "channel", first test your mail server typing:

```
$ ssh mailhost date
```

If you get the date, you should be OK. Please note ssh will not ask for a password, therefore you must create a ".shosts" file on the mail server, containing client's name. To test ssh port redirection (which gwpop uses), type:

```
$ ssh −n −f −L 12314:localhost:110 mailhost sleep 30

then
```

```
$ telnet localhost 12314
```

Then will you hopefully see mail hub's pop banner. If you don't use ssh, don't forget to comment out $ssh on gwpop script. To check whether procmail is running, try "procmail −v"

8.6 Using

Now you can edit gwpop Perl script to check everything is ok, then run gwpop:

```
$ gwpop −v your−username
POP password on mailhost: yoursecretpasword
```

If gwpop "error messages" are normal, the mail from mail hub will be downloaded to your local machine wherever you told gwpop to put it. (please test with some mail!).

You can also use gwpop as a daemon:

```
$ gwpop −d $HOME/tmp your−username
```

gwpop messages are then sent to syslog and gwpop will run endlessly ; a "HUP" signal will force gwpop to get your mail.

You can get POP software here used on:

```
ftp://ftp.unina.it/pub/Unix/pkgs/network/mail/gwpop
ftp://ftp.informatik.rwth−aachen.de/pub/packages/procmail
http://www.cs.hut.fi/ssh/
```

9.Acknowledgements

The following people have helped in the assembly of the information and experience that helped make this document possible:

Steve Robbins, Ian Kluft, Rich Braun, Ian Jackson, Syd Weinstein, Ralf Sauther, Martin White, Matt Welsh, Ralph Sims, Phil Hughes, Scot Stevenson, Neil Parker, Stephane Bortzmayer and especially many thanks to

Vince Skahan for his huge contribution.

Eric S. Raymond edited this document, correcting some mistakes and transplanting the section on ``How Electronic Mail Works'' from his Mail User's HOWTO.

Hitoshi Hayakawa checked qmail section, Jun Morimoto added various notes about popclient & fetchmail and Takeo Nakano ispell'ed the document :-)

If I forgot anybody, my apologies: just email me!
