

# APPENDIX A

# How qmail Works

YOU DON'T NEED TO UNDERSTAND how qmail works to install or use qmail. And you don't have to be an auto mechanic to operate a car or a watchmaker to tell time. But if you really want to master qmail, knowing exactly how it does what it does is crucial.

Luckily, qmail's simple, modular design makes understanding how it works easy for a system as complex as a Mail Transfer Agent (MTA). This appendix takes a top-down approach: first looking at how the modules interact with each other, then looking at how each module does its job.

## High-Level Overview

The grand division in qmail is between the modules that accept new messages and place them into the queue and the modules that deliver them from the queue. We'll call these functions *receiving* and *sending*. The separation between receiving and sending is complete: Either of these functions can be fully operational while the other is shut down. Figure A-1 shows the high-level organization of qmail.

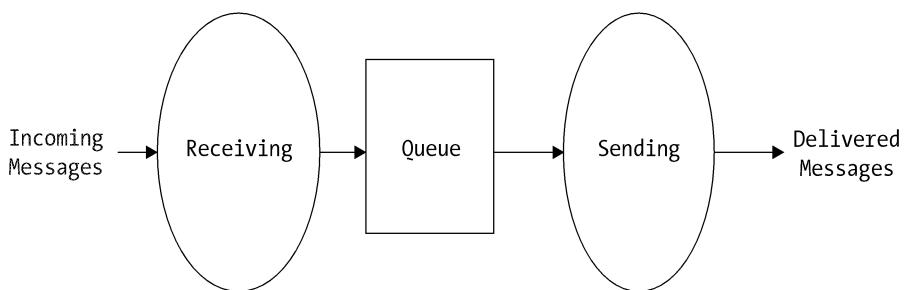
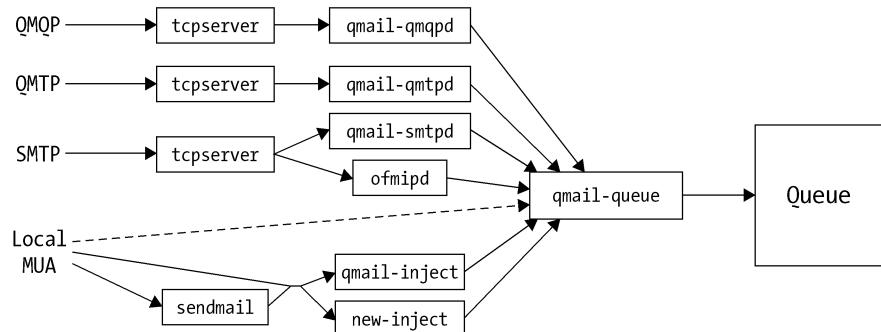


Figure A-1. High-level qmail organization

### Receiving

Messages enter the queue through two main routes: local injection using `qmail-inject` or `sendmail` and network injection using `qmail-smtpd`, `qmail-qmtpd`

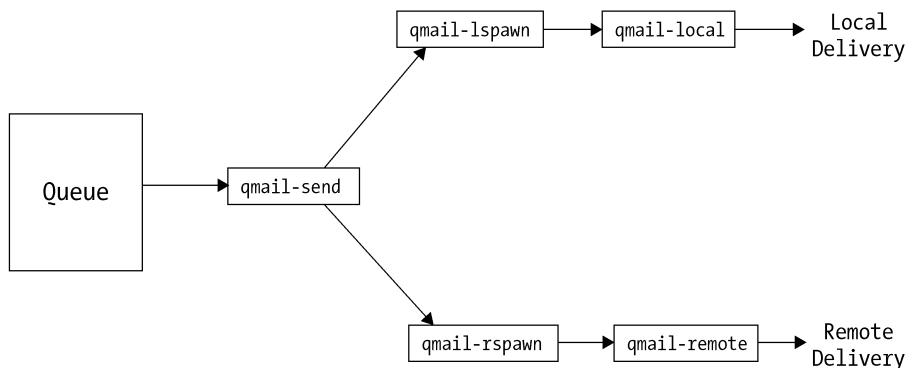
or qmail-qmqpd. Both of these routes use qmail-queue to actually inject their messages into the queue. Figure A-2 shows the organization of the receiving function.



*Figure A-2. The receiving function*

## Sending

Messages are delivered from the queue through two main routes: local delivery using qmail-local and remote delivery using qmail-remote. Both types of deliveries are dispatched by qmail-send through qmail-lspawn and qmail-rspawn, respectively. Figure A-3 shows the organization of the sending function.



*Figure A-3. The sending function*

## Receiving Modules

First we'll look at the modules comprising the receiving function: sendmail, qmail-inject, qmail-smtpd, qmail-qmtpd, qmail-qmqpd, and qmail-queue.

## Local Receiving Modules

Messages injected locally usually come in through `qmail-inject` or the `sendmail` wrapper. It's also possible to inject messages using `qmail-queue` directly, but this is uncommon.

### `sendmail`

The `sendmail` command is primarily a wrapper around `qmail-inject`. It accepts many of the `Sendmail` version's options and arguments, translates them to their `qmail-inject` equivalents, ignores irrelevant options, and runs `qmail-inject`.

### `qmail-inject`

`qmail-inject`'s job is ensuring that messages have RFC 2822-compliant headers before passing them on to `qmail-queue`. Chapter 4, "Using qmail," details how environment variables can adjust the appearance of messages passed through `qmail-inject`. The `qmail-header` man page details `qmail-inject`'s header manipulation.

### **Address Qualification**

For e-mail addresses listed in `From`, `To`, `Cc`, and other fields, `qmail-inject` ensures that they're in the format *localpart@qualifieddomain*.

- If an address consists only of a *localpart*, it appends `@defaulthost`.
- If an address consists only of *localpart@hostname*, it appends `.defaultdomain`.
- If an address looks like *localpart@hostname+*, it replaces the `+` with `.plusdomain`.

### **Recipients**

If no recipients are specified on the command line, `qmail-inject` looks for recipients in the `To`, `Cc`, `Bcc`, `Apparently-To`, `Resent-To`, `Resent-Cc`, and `Resent-Bcc` fields. All `Bcc` and `Resent-Bcc` fields are stripped from the header.

Because RFC 2822 requires that all messages have a To or CC field, qmail-inject adds one, if necessary, containing this:

```
Cc: recipient list not shown: ;
```

which is an empty *address group*.

### **Required Fields**

qmail-inject adds the following fields if they're not provided:

- From—the name of the user who invoked qmail-inject.
- Date—the current time in Greenwich Mean Time (GMT).
- Message-Id—not strictly required by RFC 2822, but handy for tracking messages. The value is `<timestamp.pid.qmail@qualifiedhostname>`, by default, where *qualifiedhostname* is constructed from *defaulthost* and *defaultdomain*.

### **Other Features**

qmail-inject also does the following:

- Resent- header fields—Resent fields are handled similarly to original fields: Resent-Cc is added if Resent-To and Resent-Cc are missing, and Resent-From and Resent-Date are added if necessary.
- Addresses listed in header fields must be separated by commas, so if qmail-inject sees addresses separated by spaces, it inserts commas. For example, this field:

```
To: carol david
```

will be rewritten as this:

```
To: carol, david
```

- Return-Path and Content-Length fields are stripped.

## Remote Receiving Modules

Messages received remotely usually come in through `qmail-smtpd`, `qmail-qmtpd`, or `qmail-qmpd`, depending upon the protocol used.

### `qmail-smtpd`

`qmail-smtpd` conducts an SMTP session on standard input and standard output, accepts one or more messages, and passes them on to `qmail-queue`. `qmail-smtpd` does *not* handle accepting network connections itself: It must be run by a network server such as `tcpserver` from the `ucspi-tcp` package (see Appendix B, “Related Packages”), `inetd`, or `xinetd`.

`qmail-smtpd` expects the environment variables listed in Table A-1 to be set.

Table A-1. TCP Environment Variables

VARIABLE	CONTENTS
PROTO	Always TCP
TCPLOCALHOST	Lowercased domain name of the local host, if available, or unset
TCPLOCALIP	Internet Protocol (IP) address of the local host in dotted-decimal form (x.x.x.x)
TCPLOCALPORT	Local port number of the SMTP session, in decimal
TCPREMOTEHOST	Lowercased domain name of the remote host, if available, or unset
TCPREMOTEINFO	A value, often a username, supplied by the remote host, obtained using the IDENT protocol (RFC 1413), if available, or unset
TCPREMOTEIP	IP address of the remote host in dotted-decimal form
TCPREMOTEPORT	Remote port number of the SMTP session, in decimal

`qmail-smtpd` begins by displaying a banner message like this:

```
220 dolphin.example.com ESMTP
```

The 220 is a status code that indicates “no problems.” See Appendix C, “An Internet Mail Primer,” for more information about Simple Mail Transfer Protocol (SMTP) and its status codes. `dolphin.example.com` is the local host name, and `ESMTP` advertises that `qmail-smtpd` implements some SMTP extensions.

After displaying the banner message, `qmail-smtpd` reads SMTP commands from the client via standard input and outputs its response to standard output. SMTP requires commands to be issued in a certain order and syntax, and `qmail-smtpd` enforces these restrictions. It also enforces other restrictions based on environment

variables such as RELAYCLIENT and DATABYTES and control files such as badmailfrom, databytes, and rcpthosts. (See Chapter 5, “Managing qmail,” for more information about these variables and control files.)

One of the most important checks qmail-smtpd performs is the validation of recipients. Recipients are specified in SMTP RCPT commands, which look like this:

```
RCPT TO:<localpart@domain>
```

If the RELAYCLIENT environment variable is set—meaning that the client is allowed to relay—the value of the variable is appended to *domain*, and the modified recipient is added to the list. RELAYCLIENT is usually set to the empty string, so the recipient address is not modified. However, if RELAYCLIENT is inadvertently set to something such as a single space character, this results in making all injections from relay-approved clients fail because of invalid domain names.



**NOTE** *There are legitimate uses of a non-empty RELAYCLIENT. The online qmail FAQ gives an example that uses RELAYCLIENT and control/virtualdomains to implement envelope rewriting (<http://cr.yp.to/qmail/faq/servers.html#network-rewriting>).*

If RELAYCLIENT is not set, qmail-smtpd looks in control/rcpthosts and control/morercpthosts.cdb for *domain*. If the domain isn’t listed, the recipient is rejected with this message:

```
553 sorry, that domain isn't in my list of allowed rcpthosts (#5.7.1)
```

Accepted messages are passed to qmail-queue to be placed in the queue. If successful, qmail-smtpd reports success, accepting responsibility for delivering the message or returning a bounce message to the sender if it’s not deliverable.

Note that qmail-smtpd converts SMTP carriage return/linefeed (CR-LF) newlines into Unix LF newlines, and returns a temporary error when it detects a bare LF—one not preceded by a CR.

qmail-smtpd adds a Received field to each message. This field looks like this:

```
Received: from unknown (HELO dolphin.example.com) (127.0.0.1)
by 0 with SMTP; 8 Aug 2001 16:02:00 -0000
```

where:

- unknown means that TCPREMOTEHOST was not set.

- dolphin.example.com was the parameter supplied by the client with the HELO command.
- 127.0.0.1 was the value of TCPREMOTEIP.
- 0 was the value of TCPLOCALHOST (in this case, set using the -l option to tcpserver).
- SMTP is the protocol.
- 8 Aug 2001 16:02:00 -0000 is the time and date at which qmail-smtpd received the message.

### *qmail-qmtpd*

qmail-qmtpd does pretty much the same thing that qmail-smtpd does. The difference is that it uses the Quick Mail Transfer Protocol (QMTP) instead of SMTP, which changes the commands and responses. qmail-qmtpd requires the same TCP environment variables listed in Table A-1 and honors the RELAYCLIENT and DATABYTES environment variables and rcpthosts, morercpthosts.cdb, and databytes control files.

### *qmail-qmqpd*

qmail-qmqpd works much like qmail-qmtpd except it does not do any relay control: All recipients are accepted unconditionally. Again, it requires the TCP environment variables listed in Table A-1.

### *qmail-queue*

qmail-queue's job is to accept messages and place them into the queue. It reads a single message from file descriptor zero and an envelope from file descriptor one. Chapter 4, "Using qmail," explains the format of the envelope and qmail-queue's exit status codes.

qmail-queue expects the envelope recipient addresses to be fully specified, including a local part (username, alias, extension address), an @, and a fully qualified domain name.

qmail-queue adds a Received field to the message that looks like this:

```
Received: (qmail 16707 invoked from network); 8 Aug 2001 16:02:00 -0000
```

where:

- 1607 is qmail-queue's process ID.
- invoked from network means qmail-queue was invoked by user qmaild.
- 8 Aug 2001 16:02:00 -0000 is the time and date at which qmail-queue processed the message.

The invoked from phrase may also indicate that qmail-queue was invoked by the user alias via qmail-local (invoked by alias) or user qmails via qmail-send (invoked for bounce).

### *How Messages Are Placed in the Queue*

To guarantee reliability, placing messages in the queue is done in four stages:

1. A file is created in /var/qmail/queue/pid named after qmail-queue's process ID. The file system assigns the file an inode number guaranteed to be unique on that file system. This is qmail's queue ID for the message.
2. The pid/pid file is renamed to mess/split/inode, and the message is written to the file.



**NOTE** split is the remainder left from dividing inode by the compile-time configuration setting conf-split. For example, if inode is 95 and conf-split is the default, 23, then split is 3 (95 divided by 23 is 4 with a remainder of 3.) Computer scientists call this operation modulo.

3. The file `intd/inode` is created and the envelope is written to it.
4. `intd/inode` is linked to `todo/inode`.

At the moment `todo/inode` is created, the message has been queued. `qmail-send` eventually (within 25 minutes) notices the new message, but to speed things up, `qmail-queue` writes a single byte to `lock/trigger`, a named pipe that `qmail-send` watches. When `trigger` contains readable data, `qmail-send` is awakened, empties `trigger`, and scans the `todo` directory.

`qmail-queue` kills itself if it hasn't successfully completed queuing the message after 24 hours.

## Sending Modules

Now we'll look at the modules comprising the sending function: `qmail-send`, `qmail-lspawn`, `qmail-rspawn`, `qmail-local`, and `qmail-remote`.

### *qmail-send*

`qmail-send` is the heart of qmail. It processes messages in the queue and dispatches them to `qmail-rspawn` and `qmail-lspawn`. Chapter 5, “Managing qmail,” covers `qmail-send`, but we'll examine `qmail-send`'s functions in the order that a message in the queue would experience them: preprocessing, delivery, and cleanup.

### *Preprocessing*

Preprocessing, like queuing, is done in stages:

1. Upon discovering `todo/inode`, `qmail-send` deletes `info/split/inode`, `local/split/inode`, and `remote/split/inode`, if they exist.
2. A new `info/split/inode` is created, containing the envelope sender address.
3. If the message has local recipients, they're added to `local/split/inode`.
4. If the message has remote recipients, they're added to `remote/split/inode`.
5. `intd/inode` and `todo/inode` are deleted.

At the moment `todo/inode` is deleted, the message is considered *preprocessed*.

Recipients are considered local if the domain is listed in `control/locals` or the entire recipient or domain is listed in `control/virtualdomains`. If the recipient is virtual, the local part of the address is rewritten as specified in `virtualdomains`.

## *Delivery*

Initially, all recipients in `local/split/inode` and `remote/split/inode` are marked *not done*, meaning that `qmail-send` should attempt to deliver to them. On its own schedule, `qmail-send` sends delivery commands to `qmail-lspawn` and `qmail-rspawn` using channels set up by `qmail-start`. When it receives responses from `qmail-lspawn` or `qmail-rspawn` that indicate successful delivery or permanent error, `qmail-send` changes their status in `local/split/inode` or `remote/split/inode` to *done*, meaning that it should not attempt further deliveries. When `qmail-send` receives a permanent error, it also records that in `bounce/split/inode`.

Bounce messages are also handled on `qmail-send`'s schedule. Bounces are handled by injecting a bounce message based on `mess/split/inode` and `bounce/split/inode`, and deleting `bounce/split/inode`.

When all of the recipients in `local/split/inode` or `remote/split/inode` are marked *done*, the respective local or remote file is removed.

## *Retry Schedules*

`qmail-send` uses a simple formula to determine the times at which messages in the queue are retried. If *attempts* is the number of failed delivery attempts so far, and *birth* is the time at which a message entered the queue (determined from the creation time of the `queue/info` file), then:

$$\text{nextretry} = \text{birth} + (\text{attempts} \times c)^2$$

where *c* is a retry factor equal to 10 for local deliveries and 20 for remote deliveries. Table A-2 shows the complete retry schedule for a remote message that's never successfully delivered, with the default *queuelifetime* of 604,800 seconds.

*Table A-2. Remote Message Retry Schedule*


---

<b>DELIVERY ATTEMPT</b>	<b>SECONDS</b>	<b>DAY-HOUR:MIN:SEC</b>
1	0	0-00:00:00
2	400	0-00:06:40
3	1,600	0-00:26:40
4	3,600	0-01:00:00
5	6,400	0-01:46:40
6	10,000	0-02:46:40
7	14,400	0-04:00:00
8	19,600	0-05:26:40
9	25,600	0-07:06:40
10	32,400	0-09:00:00
11	40,000	0-11:06:40
12	48,400	0-13:26:40
13	57,600	0-16:00:00
14	67,600	0-18:46:40
15	78,400	0-21:46:40
16	90,000	1-01:00:00
17	102,400	1-04:26:40
18	115,600	1-08:06:40
19	129,600	1-12:00:00
20	144,400	1-16:06:40
21	160,000	1-20:26:40
22	176,400	2-01:00:00
23	193,600	2-05:46:40
24	211,600	2-10:46:40
25	230,400	2-16:00:00
26	250,000	2-21:26:40
27	270,400	3-03:06:40
28	291,600	3-09:00:00
29	313,600	3-15:06:40
30	336,400	3-21:26:40
31	360,000	4-04:00:00
32	384,400	4-10:46:40
33	409,600	4-17:46:40

---

Table A-2. Remote Message Retry Schedule (continued)

DELIVERY ATTEMPT	SECONDS	DAY-HOUR:MIN:SEC
34	435,600	5-01:00:00
35	462,400	5-08:26:40
36	490,000	5-16:06:40
37	518,400	6-00:00:00
38	547,600	6-08:06:40
39	577,600	6-16:26:40
40	608,400	7-01:00:00

The local message retry schedule is similar, but because of the lower  $c$ , messages are retried twice as often.

### Cleanup

When both local/*split/inode* and remote/*split/inode* have been removed, the message is dequeued by:

1. Processing bounce/*split/inode*, if it exists.
2. Deleting info/*split/inode*.
3. Deleting mess/*split/inode*.

Partially queued and partially dequeued messages left when a system crash interrupts qmail-queue or qmail-send are deleted by qmail-send using qmail-clean, another long-running daemon started by qmail-start. Messages with a mess/*split/inode* file and possibly an intd/*inode*—but no todo, info, local, remote, or bounce—are safe to delete after 36 hours because qmail-queue kills itself after 24 hours. Similarly, files in the pid directory more than 36 hours old are also deleted.

### Local Sending Modules

Messages to be delivered locally are passed from qmail-send to qmail-lspawn, which invokes qmail-local to perform the delivery.

### *qmail-lspawn*

*qmail-lspawn* reads delivery commands from *qmail-send* on file descriptor 0 (zero), invokes *qmail-local* to deliver the messages, and reports the results to *qmail-send* on descriptor 1 (one).

Before invoking *qmail-local*, *qmail-lspawn* determines which local user controls the address so *qmail-local* can be started with the necessary user ID and group ID. *qmail-lspawn* first checks the *qmail-users* database, *users/cdb*. If the address is not listed there, it runs *qmail-getpw*. If *qmail-getpw* doesn't find a matching user, it gives control of the address to the alias user.

### *qmail-local*

*qmail-local* accepts a message on standard input with envelope information, delivery location, and default delivery instructions supplied as arguments. Before attempting delivery, it constructs a Delivered-To field based on the envelope and checks the message for an identical Delivered-To field. If it finds one, it bounces the message to prevent a mail loop. Chapter 4, "Using qmail," details the actual delivery process. If the delivery is successful, *qmail-local* returns an exit status of 0 (zero). All other codes indicate either permanent or temporary failure.

## *Remote Sending Modules*

Messages to be delivered remotely are passed from *qmail-send* to *qmail-rspawn*, which invokes *qmail-remote* to perform the delivery.

### *qmail-rspawn*

*qmail-rspawn* reads delivery commands from *qmail-send* on file descriptor 0 (zero), invokes *qmail-remote* to deliver the messages, and reports the results to *qmail-send* on descriptor 1 (one).

### *qmail-remote*

*qmail-remote* accepts a message on standard input with envelope information supplied as arguments. After attempting to deliver the message remotely via SMTP, it summarizes its results via reports printed to standard output. Chapter 5, "Managing qmail," explains the format of these reports.

The remote host is specified as one of `qmail-remote`'s arguments. It can be either a fully qualified domain name or an IP address. If it's a domain name, `qmail-remote` checks the Domain Name System (DNS) for a mail exchanger (MX) record for that domain. If the remote host is listed in `control/smtproutes`, `qmail-remote` uses the host specified in `smtproutes`.

# APPENDIX B

# Related Packages

QMAIL IS A COMPLETE Mail Transfer Agent (MTA), but many packages were either designed specifically to add new functionality to qmail or simply work well with qmail. This appendix lists some of these packages, describes them briefly, and provides links for more information.

The unofficial qmail home page, <http://www.qmail.org/>, is the definitive collection of information about qmail-related packages.

## checkpassword

checkpassword is the authentication package used by qmail-pop3d (really qmail-popup). The standard checkpassword package by Daniel Bernstein (<http://cr.yp.to/checkpwd.html>) authenticates users with the Unix password file.

Many alternative checkpassword implementations support other authentication mechanisms including Lightweight Directory Access Protocol (LDAP), Structured Query Language (SQL) databases, Pluggable Authentication Modules (PAM), Authenticated POP (APOP), and dedicated POP password files. The unofficial qmail home page contains a checkpassword section with links to these alternative implementations (<http://www.qmail.org/top.html#checkpassword>).



**TIP** *To verify that your checkpassword program is authenticating properly, you can test it from the command line using Perl. For example, if user paul's password is Lauren&Natalie, the command perl -e 'printf "%s\0%s\099\0", "paul", "Lauren&Natalie"' | /bin/checkpassword echo OK 3<&0 will output "OK" if the authentication succeeds and nothing if it fails.*

## Courier-IMAP

Courier-IMAP is an Internet Mail Access Protocol (IMAP) server often used with qmail because it supports maildir mailboxes. Chapter 10, “Serving Mailboxes,” covers installing, configuring, and using Courier-IMAP.

Courier-IMAP was written by Sam Varshavchik, who maintains a Web page for it (<http://www.inter7.com/courierimap/>).

## daemontools

The daemontools package contains a set of utilities for controlling and monitoring services. It’s highly recommended, especially for busy systems. Key utilities included are:

- `supervise`, which monitors a service and restarts it if it dies
- `svscan`, which monitors a service directory and starts `supervise`
- `svc`, which talks to `supervise` and allows one to stop, pause, or restart a service
- `multilog`, which maintains a log for a service, automatically rotating it to keep it under the configured size
- `setuidgid`, which runs a program for the superuser with a normal user’s user and group IDs

Gerrit Pape distributes the documentation for daemontools as `man` pages from <http://innominate.org/~pape/djb/>.

daemontools was written by Daniel Bernstein, who maintains a Web page for it (<http://cr.yp.to/daemontools.html>).

## djbdns

djbdns is a Domain Name System (DNS) server created by qmail’s author, Daniel Bernstein. Like qmail, it provides a secure, reliable, efficient, and modular alternative to the *de facto* standard, which in this case is Berkeley Internet Name Daemon (BIND).

If you’re running BIND now—either for providing authoritative name service for your domain(s) or as a caching-only server for improving lookup performance—you should consider switching to djbdns for the same reasons that you’re running qmail.

Even if you're not running a name server, you should install djbdns and run dnscache to enhance DNS lookup performance for all applications—not just qmail—and reduce outgoing DNS traffic.

The official djbdns Web site is <http://cr.yp.to/djbdns.html>. The unofficial djbdns Web site (<http://www.djbdns.org/>) is another valuable resource, as is “Life with djbdns,” a djbdns manual available on the Web (<http://www.lifewithdjbdns.org/>).

## **dot-forward**

Sendmail uses .forward files to allow users to control the delivery of messages they receive. qmail uses a similar mechanism: .qmail files. The dot-forward package gives qmail the ability to use .forward files. Systems running Sendmail or any other MTA that uses .forward files might want to consider using dot-forward to avoid having to convert existing .forward files to their .qmail equivalents—or simply to make the transition to qmail less visible to their users.

dot-forward is a small package, so it's easy to install and configure. Chapter 7, “qmail Configuration: Advanced Options,” covers installing and configuring dot-forward.

dot-forward was written by Daniel Bernstein, who maintains a Web page for it (<http://cr.yp.to/dor-forward.html>).

## **ezmlm**

ezmlm is a high-performance, easy-to-use mailing-list manager (MLM) for qmail. If you're familiar with LISTSERV or Majordomo, you know what a mailing-list manager does. For more information about mailing lists under qmail, including installing, configuring, and using ezmlm, see Chapter 9, “Managing Mailing Lists.”

ezmlm was written by Daniel Bernstein, who maintains a Web page for it (<http://cr.yp.to/ezmlm.html>).

## **ezmlm-idx**

ezmlm-idx is an add-on for ezmlm that adds many useful features such as multiple message archive retrieval, digests, message and subscription moderation, and

remote list maintenance. Chapter 9, “Managing Mailing Lists,” covers installing, configuring, and using ezmlm-idx.

Fred Lindberg and Fred B. Ringel created ezmlm-idx and maintain a Web page for it (<http://www.ezmlm.org/>).

## **fastforward**

fastforward is another Sendmail compatibility add-on. Sendmail uses a central alias database kept in a single file, usually /etc/aliases. qmail uses a series of dot-qmail files in /var/qmail/alias, one file per alias. If you’re migrating to qmail and you’ve got a Sendmail-format aliases file that you don’t want to convert, fastforward gives qmail the ability to use the aliases file as-is.

Chapter 7, “qmail Configuration: Advanced Options,” covers installing and configuring fastforward. fastforward was written by Daniel Bernstein, who maintains a Web page for it (<http://cr.yp.to/fastforward.html>).

## **getmail**

getmail is a POP3 client written in Python. It retrieves messages from a Post Office Protocol version 3 (POP3) server and delivers them locally to maildir or mbox mailboxes or programs.

Chapter 10, “Serving Mailboxes,” covers installing, configuring, and using getmail.

getmail was written by Charles Cazabon, who maintains a Web page for it (<http://www.qcc.sk.ca/~charlesc/software/getmail-2.0/>).

## **maildrop**

maildrop is a mail filter similar to Procmail. It provides a powerful filtering language and can be used in dot-qmail files to intercept junk mail or direct mail to different mailboxes.

maildrop was written by Sam Varshavchik, who maintains a Web page for it (<http://www.flounder.net/~mrsam/maildrop>).

## **mess822**

mess822 is a library and set of applications for parsing Request For Comments (RFC) 822 (currently RFC 2822)-compliant mail messages. The applications include:

- `ofmid`, a Simple Mail Transfer Protocol (SMTP) daemon that accepts messages from clients and rewrites `From` fields based on a database
- `new-inject`, a `qmail-inject` replacement that supports user-controlled host name rewriting
- `iftocc`, a dot-qmail utility for checking whether a message was sent to a specific address
- `822header`, `822field`, `822date`, and `822received`, which extract information from a message
- `822print`, pretty-prints a message

`mess822` was written by Daniel Bernstein, who maintains a Web page for it (<http://cr.yp.to/mess822.html>).

## **oMail-webmail**

`oMail-webmail` is a Web-based Mail User Agent (MUA) for qmail. It accesses `maildir` mailboxes directly, rather than through POP3 or IMAP like some other Web-based MUAs.

`oMail-webmail` was written by Olivier Müller, who maintains a Web page for it (<http://webmail.omnis.ch/omail.pl?action=about>).

## **oSpam**

`oSpam` is an anti-spam utility similar to TMDA (see “TMDA” in this appendix).

`oSpam` was written by Olivier Müller, who maintains a Web page for it (<http://omail.omnis.ch/ospam/>).

## **qlogtools**

qlogtools is a set of utilities used for producing and analyzing logs from services managed using daemontools, especially qmail. Particularly handy are:

- `qlogselect`, which extracts messages from a `multilog qmail-send` log file from a particular sender or time period
- `ta164n2tai`, which converts `multilog` timestamps to the format that `qmailanalog` requires (see “`qmailanalog`” in this appendix)
- `multitail`, which displays data appended to a named file, even if the file is cycled—like `multilog`’s `current`

`qlogtools` was written by Bruce Guenter, who maintains a Web page for it (<http://untroubled.org/qlogtools/>).

## **qmail-autoreader**

Autoresponders are utilities run from dot-qmail files that send a message in response to incoming mail. Although this sounds like a simple task, there are many pitfalls. For example, you probably don’t want to automatically respond to messages received from mailing lists. And you certainly don’t want to respond to other autoresponders, which could quickly result in thousands of messages bouncing back and forth between the two responders.

Bruce Guenter’s `qmail-autoreader` avoids these pitfalls and is easy to install and use. It’s available on the Web (<http://untroubled.org/qmail-autoreader/>).

## **qmail-qfilter**

`qmail-qfilter` is a utility that allows messages to be filtered before they’re passed to `qmail-queue`. Using `qmail-qfilter`, programs and scripts can be used to alter messages (add or adjust header fields, for example) or refuse unwanted messages (such as those with executable attachments).

Bruce Guenter wrote `qmail-qfilter` and maintains a Web page for it (<http://untroubled.org/qmail-qfilter/>).

## **Qmail-Scanner**

Qmail-Scanner is a virus-scanning harness for qmail. It works with most commercial Unix virus scanners and includes a built-in scanner that allows administrators to quickly and easily block Windows-based malware before it's been added to the commercial scanner's signature database. Chapter 12, "Understanding Advanced Issues," covers installing, configuring, and using Qmail-Scanner.

Qmail-Scanner was written by Jason Haar, who maintains a Web page for it (<http://qmail-scanner.sourceforge.net/>).

## **qmail-vacation**

qmail-vacation is a special-purpose autoresponder (see the "qmail-autoresponder" section in this appendix) for notifying senders that the recipient won't be reading their message immediately, for example, because they're on vacation. qmail-vacation allows users to easily enable and disable these autoresponses.

qmail-vacation was written by Peter Samuel and is available from the Web (<http://www.gormand.com.au/peters/tools/>).

## **qmailanalog**

qmailanalog processes qmail-send's log files and produces a series of reports that tell you how much and what kind of work the system is doing. If you need statistics about how many messages are being sent or received, how big they are, and how quickly they're being processed, qmailanalog is what you need.

qmailanalog relies on log-entry timestamps in the fractional second format used by *accustamp*, an obsolete time-stamping utility. To use it with logs generated by *multilog*, which are in Temps Atomique International 64-bit, nanosecond precision (TAI64N) format, you'll need to translate them into the old format. One program to do that is available from <http://www.qmail.org/tai64nfrac>.

qmailanalog was written by Daniel Bernstein, who maintains a Web page for it (<http://cr.yp.to/qmailanalog.html>).

## *Using qmailanalog*

Installing qmailanalog following the directions in INSTALL is straightforward, but running it is a little tricky, especially because it doesn't understand TAI64N timestamps:

1. First, install qmailanalog using the directions in INSTALL.
2. Download tai64nfrac to /usr/local/src/tai64nfrac.c, edit the file and remove everything above the line containing /\* \$Id\$. Compile the program using:

```
# cc -o /usr/local/bin/tai64nfrac /usr/local/src/tai64nfrac.c
#
```

3. Process one or more qmail-send log files through tai64nfrac. The input log entries should look something like this:

```
@400000003b88ef5c313ae1e4 status: local 0/10 remote 0/20
```

In particular, each line must start with a TAI64N timestamp (the @400000003b88ef5c313ae1e4 part) followed by a qmail-send log message (the status: local 0/10 remote 0/20 part).

For example:

```
# tai64nfrac < /var/log/qmail/* > logs.frac
#
```

The lines in logs.frac should look like this:

```
998829906.825942500 status: local 0/10 remote 0/20
```

4. Process the fractional-timestamp log file using matchup. Because the log files may contain entries for messages that haven't been delivered yet, matchup will write those entries to file descriptor 5. Save them to a file for inclusion in the next matchup run. For example:

```
# /usr/local/qmailanalog/bin/matchup < logs.frac > logs.match 5> logs.cont
#
```

Each line in logs.match contains all of the relevant information for a single delivery attempt.

5. Use the scripts in /usr/local/qmailanalog/bin with names starting with z to produce a report from logs.match. For example, to produce an overall summary report:

```
/usr/local/qmailanalog/bin/zoverall < logs.match
```

Each report includes an explanation of its output.

6. Use the scripts in /usr/local/qmailanalog/bin with names starting with x to extract entries for particular messages, senders, or recipients. These extracted entries can then be passed through a z script. For example, for a report on the recipient hosts of messages from root@dolphin.example.com, you could do this:

```
# cd /usr/local/qmailanalog
# bin/xsender root@dolphin.example.com < log.match | bin/zrhosts
...report...
```

## **safecat**

safecat reliably delivers a message to a maildir mailbox. It's useful for filing messages using filters such as Procmail. For example, the following recipe files all messages in \$HOME/Maildir:

```
:0w
|safecat Maildir/tmp Maildir/new
```

safecat was written by Len Budney, who maintains a Web page for it (<http://www.pobox.com/~lbudney/linux/software/safecat.html>).

## **serialmail**

qmail was designed for systems with full-time, high-speed connectivity. serialmail is a set of tools that help adapt qmail to intermittent, low-speed connectivity. With serialmail on such a system, qmail can be configured to deliver all remote mail to a single spool maildir. The serialmail maildirsmtp command can be used to upload the maildir to the Internet Service Provider's (ISPs) mail hub after the connection is brought up. If the ISP supports QMTP (see Chapter 7, "Configuring qmail: Advanced"), maildirqmtp can also be used.

serialmail can be used on the ISP side of a dialup connection to implement the AutoTURN mechanism, where an SMTP connection by a client causes the

server to initiate a connection back to the client for delivering messages queued on the server. This is similar to the SMTP ETRN function but doesn't require the client to issue the ETRN command. AutoTURN is documented in the AUTOTURN file in the serialmail source directory.

serialmail was written by Daniel Bernstein, who maintains a Web page for it (<http://cr.yp.to/serialmail.html>).

## SqWebMail

SqWebMail is a Web-based MUA like oMail-webmail (see the “oMail-webmail” section in this appendix). SqWebMail also access maildir mailboxes directly, not through POP3 or IMAP, for improved performance.

SqWebMail was written by Sam Varshavchik, who maintains a Web page for it (<http://inter7.com/sqwebmail/>).

## syncdir

The syncdir package for Linux is a library that provides wrapped versions of the link(), open(), rename(), and unlink() system calls that force their changes to be written to disk immediately. This is useful because qmail relies upon the Berkeley Software Distribution (BSD) behavior of these operations to ensure that the queue is crash proof.

To use this library, build and install the library using `make` and `make install`, append `-lsyncdir` to the command in `conf-ld` in the qmail source directory, and rebuild qmail.

syncdir was written by Bruce Guenter. It's available from the Web (<http://untroubled.org/syncdir/>).

## TMDA

TMDA (Tagged Message Delivery Agent) is an anti-spam tool similar to oSpam (see the “oSpam” section in this appendix). TMDA allows the user to maintain a *whitelist*, a list of pre-approved senders. Messages from all other senders are held until the sender responds to a confirmation request. Because most spammers use invalid return addresses or are too lazy to respond to confirmation requests, their messages are not delivered.

TMDA is covered in Chapter 8, “Controlling Junk Mail.”

TMDA was written by Jason Mastaler, who maintains a Web page for it (<http://software.libertine.org/tmda/>).

## **ucspi-tcp**

qmail's SMTP server doesn't run as a stand-alone daemon. A helper program such as `tcpserver`, `inetd`, or `xinetd` runs as a daemon. When it receives a Transmission Control Protocol (TCP) connection to port 25, the SMTP port, it runs `qmail-smtpd`.

`Inetd` is the de facto standard network server "super-server." It can be configured through `/etc/inetd.conf` to run `qmail-smtpd`, but the recommended server tool for qmail is `tcpserver`, which is part of the `ucspi-tcp` package. `ucspi-tcp` is an acronym for Unix Client-Server Program Interface for TCP, and it's pronounced *ooks-pie tee see pee*.

Chapter 2, "Installing qmail," compares `tcpserver` to `inetd`. Chapter 8, "Controlling Junk Mail," covers installing and configuring `rlsmtpd`, an `ucspi-tcp` tool for checking DNS blacklists.

Gerrit Pape distributes the documentation for `ucspi-tcp` as `man` pages from <http://innominate.org/~pape/djb/>.

`ucspi-tcp` was written by Daniel Bernstein, who maintains a Web page for it (<http://cr.yp.to/ucspi-tcp.html>).

## **VMailMgr**

`VMailMgr` (Virtual Mail Manager) is a virtual domain and virtual user management add-on for qmail. `VMailMgr` allows a single Unix user, the virtual domain manager, to add and remove mail users in the domain. The virtual users don't require Unix accounts, and can retrieve their mail via POP3 or IMAP.

Chapter 11, "Managing Virtual Domains and Users," covers installing, configuring, and using `VMailMgr`.

`VMailMgr` was created by Bruce Guenter, who maintains a Web site for it (<http://www.vmailmgr.org/>).

## **Vpopmail**

`Vpopmail` is a virtual domain and virtual user management add-on for qmail similar to `VMailMgr` (see the previous section).

Chapter 11, "Managing Virtual Domains and Users," covers installing, configuring, and using it.

`Vpopmail` is maintained by Ken Jones, who also maintains a Web site for it (<http://www.inter7.com/vpopmail/>).

## APPENDIX C

# How Internet Mail Works

ALTHOUGH INTERNET MAIL IS one of the most heavily used Internet services, many users—and a surprising percentage of system administrators—don't really understand how it works. This appendix provides some background about how Internet mail works and includes pointers to more detailed sources of information.

### How a Message Gets from Point A to Point B

When a user on one host sends a message to a user on another host, many things happen behind the scenes of which you may not be aware.

Let's say Alice, `alice@alpha.example.com`, wants to send a message to Bob, `bob@beta.example.com`. Here's what happens:

1. Alice composes the message with her mail user agent (MUA), something such as Mutt or Pine. She specifies the recipient in a To field, the subject of the message in a Subject field, and the text of the message itself. It looks something like this:

```
To: bob@beta  
Subject: lunch
```

How about pizza?

2. When she's satisfied with the message, she tells the MUA to send it.
3. At this point, the MUA can add additional header fields such as Date and Message-Id and modify the values Alice entered (for example, it could replace `bob@beta` with `Bob <bob@beta.example.com>`).
4. Next, the MUA *injects* the message into the mail system in one of two ways: It can run a program provided by the mail system for the purpose of injecting messages, or it can open a connection to the Simple Mail Transfer Protocol (SMTP) port on either the local system or a remote mail

server. For this example, we'll assume the MUA uses a local injection program to pass messages to the MTA. The details of the injection process vary by MTA, but on Unix systems the `sendmail` program is a *de facto* standard. With this program, the MUA puts the header and body in a file, separated by a blank line, and passes the file to the `sendmail` program.

5. If the injection succeeds—the message was syntactically correct and `sendmail` was invoked properly—the message is now the MTA's responsibility. Details vary greatly by MTA, but generally the MTA on alpha examines the header to determine where to send the message, opens an SMTP connection to beta, and forwards the message to the MTA on the beta system. The SMTP dialogue requires messages to be sent in two parts: the *envelope*, which specifies the recipient's address (`bob@beta.example.com`) and the return address (`alice@alpha.example.com`), and the message itself, which consists of the header and body.
6. If the beta MTA rejects the message, perhaps because there's no user bob on the system, the MTA on alpha sends a *bounce* message to the return address, `alice@alpha.example.com`, to notify her of the problem.
7. If the beta MTA accepts the message, it looks at the recipient's address, determines whether it's local to beta or on a remote system. In this case, it's local, so the MTA either delivers the message itself or passes it to a message delivery agent (MDA) like `/bin/mail`, `qmail-local`, or `Procmail`.
8. If the delivery fails, perhaps because Bob has exceeded his mail quota, the beta MTA sends a bounce message to the envelope return address, `alice@alpha.example.com`.
9. If the delivery succeeds, the message waits in Bob's mailbox until his MUA reads it and displays it.

## Envelopes vs. Headers

When you send a letter by “snail mail”—the old-fashioned physical delivery method—you write a letter that looks something like this:

To: Jane Doe  
123 Main Street  
Springfield, Anystate 99999

Dear Jane,

Blah blah blah...

Your friend, John Q. Public

You then place the letter in an envelope with Jane's address, and your address—so the postal service can return your letter to you if it can't deliver it for some reason.

Internet mail works much the same. When you send a letter by e-mail you construct a message that looks like this:

```
From: "John Q. Public" <jqpublic@isp.example.net>
To: "Jane Doe" <jane@doe.example.com>
Subject: Blah
```

Blah blah blah...

-John

When you hit the Send button on your MUA, either the MUA or the MTA that receives the message constructs an envelope for it. As with snail mail, the envelope contains the recipient's address—which is required for delivering the message, of course—and the sender's address, which might be necessary for notifying the sender that the letter was undeliverable. Unlike snail mail, the Internet mail envelope does not require a stamp.

Because the envelope is constructed automatically for the user, many users don't even realize it exists. They mistakenly believe that the header of the message is the envelope. This belief works fine for simple person-to-person messages such as the one in the previous example, where the envelope is constructed from the header. It fails miserably when that's not the case, such as most spam, messages received from mailing lists, and messages received via Bcc (blind carbon copy).

For example, let's look at message sent with a Bcc header field. The most common implementation of Bcc is for the sending MUA/MTA to strip the Bcc field from the message after adding the addresses listed to the envelope. So, a message that looks like this:

```
From: "John Q. Public" <jqpublic@isp.example.net>
To: "Jane Doe" <jane@doe.example.com>
Bcc: "John Doe" <john@doe.example.com>
Subject: Blah
```

Blah blah blah...

-John

when it's submitted by the sender, arrives in both recipients' mailboxes without the Bcc field. This is accomplished by creating an envelope that looks like this:

*Sender:jqpublic@isp.example.net  
Recipients:jane@doe.example.com, john@doe.example.com*

MTAs have different ways of storing the envelopes of messages in their queues. With qmail, envelope senders are stored under /var/qmail/queue/info, local recipients are stored under /var/qmail/queue/local, and remote recipients are stored under /var/qmail/queue/remote. Messages (header plus body) are stored under /var/qmail/queue/mess.

When messages are sent via SMTP, the MAIL command is used to send the envelope sender, and the RCPT command is used to send envelope recipients. Messages are sent using the DATA command, which must be preceded by the MAIL and RCPT commands.

## **Finding More Information**

For information about how Internet mail works, see one or more of the following documents by the author of qmail:

- Internet mail (<http://cr.yp.to/im.html>)
- SMTP (<http://cr.yp.to/smtp.html>)
- Internet mail message header format (<http://cr.yp.to/immhf.html>)

Detailed information about the Internet mail standard is contained in the Requests for Comment (RFCs).

## *Internet Mail RFCs*

RFCs are the official documentation of the Internet. Most of these are well beyond the commentary stage and actually define Internet protocols such as the Transmission Control Protocol (TCP), the File Transfer Protocol (FTP), Telnet, and the various mail standards and protocols:

- RFC 821, Simple Mail Transfer Protocol, obsoleted by RFC 2821  
<http://www.ietf.org/rfc/rfc0821.txt>
- RFC 822, Standard for the Format of ARPA Internet Text Messages, obsoleted by RFC 2822  
<http://www.ietf.org/rfc/rfc0822.txt>

- RFC 931, Authentication Server  
<http://www.ietf.org/rfc/rfc0931.txt>
- RFC 974, Mail Routing and the Domain System  
<http://www.ietf.org/rfc/rfc0974.txt>
- RFC 1123, Requirements for Internet Hosts—Application and Support  
<http://www.ietf.org/rfc/rfc1123.txt>
- RFC 1413, Identification Protocol  
<http://www.ietf.org/rfc/rfc1413.txt>
- RFC 1423, Privacy Enhancement for Internet Electronic Mail: Part III:  
Algorithms, Modes, and Identifiers  
<http://www.ietf.org/rfc/rfc1423.txt>
- RFC 1651, SMTP Service Extensions  
<http://www.ietf.org/rfc/rfc1651.txt>
- RFC 1652, SMTP Service Extension for 8bit-MIMEtransport  
<http://www.ietf.org/rfc/rfc1652.txt>
- RFC 1806, Content-Disposition Header  
<http://www.ietf.org/rfc/rfc1806.txt>
- RFC 1854, SMTP Service Extension for Command Pipelining  
<http://www.ietf.org/rfc/rfc1854.txt>
- RFC 1891, SMTP Service Extension for Delivery Status Notifications  
<http://www.ietf.org/rfc/rfc1891.txt>
- RFC 1892, The Multipart/Report Content Type for the Reporting of Mail  
System Administrative Messages  
<http://www.ietf.org/rfc/rfc1892.txt>
- RFC 1893, Enhanced Mail System Status Codes  
<http://www.ietf.org/rfc/rfc1893.txt>
- RFC 1894, An Extensible Message Format for Delivery Status Notifications  
<http://www.ietf.org/rfc/rfc1894.txt>
- RFC 1939, Post Office Protocol, Version 3  
<http://www.ietf.org/rfc/rfc1939.txt>

- RFC 1985, SMTP Service Extension for Remote Message Queue Starting (ETRN)  
<http://www.ietf.org/rfc/rfc1985.txt>
- RFC 1991, PGP Message Exchange Formats  
<http://www.ietf.org/rfc/rfc1991.txt>
- RFC 2015, MIME Security with Pretty Good Privacy (PGP)  
<http://www.ietf.org/rfc/rfc2015.txt>
- RFC 2045, MIME Internet Message Bodies  
<http://www.ietf.org/rfc/rfc2045.txt>
- RFC 2046, MIME Media Types  
<http://www.ietf.org/rfc/rfc2046.txt>
- RFC 2047, MIME Headers  
<http://www.ietf.org/rfc/rfc2047.txt>
- RFC 2048, MIME Registration Procedures  
<http://www.ietf.org/rfc/rfc2048.txt>
- RFC 2049, MIME Conformance Criteria  
<http://www.ietf.org/rfc/rfc2049.txt>
- RFC 2142, Mailbox Names for Common Services  
<http://www.ietf.org/rfc/rfc2142.txt>
- RFC 2183, Content Disposition Header  
<http://www.ietf.org/rfc/rfc2183.txt>
- RFC 2821, Simple Mail Transfer Protocol  
<http://www.ietf.org/rfc/rfc2821.txt>
- RFC 2822, Internet Message Format  
<http://www.ietf.org/rfc/rfc2822.txt>

A comprehensive list of mail-related RFCs is available from the Internet Mail Consortium at <http://www.imc.org/mail-standards.html>.

# APPENDIX D

# qmail Features

CHAPTER 1 LISTED QMAIL’s features in a readable, newbie-friendly format. This appendix does it again in more detail. If you really need to understand a feature, perhaps to explain it to someone else, this appendix should help. This should also help explain the feature list on the official qmail Web site (<http://cr.yp.to/qmail.html>).

## Setup Features

qmail includes the following setup features:

- **Adaptable.** During the build process, qmail automatically adapts itself to most Unix and Linux distributions, obviating the need for manual porting by a system programmer.
- **Automatic configuration.** Basic per-host configuration is done automatically by qmail using the config and config-fast scripts.
- **Quick installation.** Setting up a basic installation is easy and doesn’t require lots of decision making.

## Security Features

qmail includes the following security features:

- **Compartmentalization of delivery targets.** There is a clear distinction between addresses, files, and programs that prevents attackers from writing to security-critical files and executing arbitrary programs with elevated privileges.
- **Minimization of setuid() code.** Only one module, qmail-queue, runs setuid().
- **Minimization of root code.** Only two modules runs as root: qmail-start and qmail-lspawn.

- **Five-way trust partitioning.** Five qmail-specific user IDs are used to partition trust within the qmail system. A compromise to the system should be contained to one partition.
- **Logging.** Using the QUEUE\_EXTRA compile-time option, logging of one-way message hashes, entire message contents, or other desired information is possible for all messages or subsets of messages (for example, messages from or to a specified user or domain).

## Message Construction

qmail includes the following message construction features:

- **RFC compliant.** Messages built by qmail-inject comply with the Internet RFCs 2822 (message format) and 1123 (requirements for Internet hosts).
- **Address groups.** Full support is provided for RFC 2822 address groups.
- **sendmail hook.** A sendmail command is included for compatibility with current user agents.
- **Long header fields.** Header line length is limited only by the available system memory.
- **Host masquerading.** Local hosts can be hidden behind a public mail relay.
- **User masquerading.** Local users can be hidden behind aliases on a mail server.
- **Automatic Mail-Followup-To creation.** The Mail-Followup-To header field is used by the author of a message to direct replies to mailing-list messages to the appropriate address or addresses.

## SMTP Service

qmail includes the following Simple Mail Transfer Protocol (SMTP) service features:

- **RFC compliant.** Complies with RFC 2821 (SMTP), RFC 1123, RFC 1651 (ESMTP), RFC 1652 (8-bit MIME), and RFC 1854 (pipelining).
- **8-bit clean.** Accepts 7-bit ASCII characters as well as 8-bit extended characters.

- **Supports IDENT (RFC 931/1413/TAP) callback.** This allows cooperating mail administrators to determine the identity of users abusing the system.
- **Relay control.** qmail automatically denies unauthorized relaying by outsiders.
- **Automatic recognition of local Internet Protocol (IP) addresses.**  
Messages to jessica@[192.168.1.3] are recognized as local by qmail-smtpd if 192.168.1.3 is a local IP address.
- **Per-buffer timeouts.** Each new buffer of data from the remote SMTP client has its own time limit.
- **Hop counting for detection of looping messages.** Messages that pass through more than 100 delivery hops are rejected.
- **Parallelism limit.** Using tcpserver, which is part of the ucspi-tcp package, the number of concurrent incoming SMTP sessions can be controlled.
- **Refusal of connections from known abusers.** Using tcpserver, specific hosts and domains can be refused access to the SMTP server.
- **Relaying and message rewriting for authorized clients.** The RELAYCLIENT environment variable can be used to allow authorized hosts to relay or to modify header fields for specified hosts.
- **Optional RBL support.** Using rblsmtpd, from the ucspi-tcp package, access can be denied to known senders of junk e-mail—also known as *spam*.

## Queue Management

qmail includes the following queue management features:

- **Instant handling of messages added to queue.** New messages are always delivered immediately, subject to resource availability.
- **Parallelism limits.** The number of simultaneous local and remote deliveries is limited and configurable.
- **Split queue directory.** Some Unix file systems experience significant slow downs with large directories. qmail splits the queue into a configurable (at compilation) number of subdirectories to keep the number of files per directory low.

- **Quadratic retry schedule.** Undelivered messages in the queue are retried less frequently as they age—the longer a host has been unreachable, the less likely it is to be reachable soon.
- **Independent message retry schedules.** Each message has its own retry schedule. If a long-down host comes back, qmail won't immediately flood it with a huge backlog.
- **Automatic safe queuing.** No mail is lost if the system crashes.
- **Automatic per-recipient checkpointing.** Each successful delivery of a message to multiple recipients is recorded, preventing the sending of duplicates in the event of a crash.
- **Automatic queue cleanups.** Interrupted queue injections can leave partially injected messages in the queue. `qmail-send` automatically cleans these out after 36 hours. `qmail-clean` removes messages after successful delivery.
- **Queue viewing.** `qmail-qread` displays the current contents of the queue.
- **Detailed delivery statistics.** The `qmailanalog` package analyzes the `qmail-send` logs and produces delivery statistics.

## Bounces

qmail includes the following non-deliverability report (bounce) features:

- **qmail-send Bounce Message Format (QSBMF) bounce messages.** qmail's bounce messages are in a format that's both machine-readable and human-friendly.
- **Hash Convention for Mail System Status Codes (HCMSSC) support.** Bounce messages include language-independent RFC 1893 error codes.
- **Double bounces sent to postmaster.** Undeliverable bounce messages often indicate configuration errors, so they're delivered to the postmaster alias, by default.

## Routing by Domain

qmail includes the following features for routing messages by domain name:

- **Unlimited names for local host.** The local system can have any number of aliases.
- **Unlimited virtual domains.** One host can support any number of virtual domains—each with a separate name space.
- **Domain wildcards.** Using the virtual domains support, domains can be wildcard matched for special routing.
- **Configurable “percent hack” support.** Sendmail-style routed addresses—for example, `molly@example.com@example.net`—can be supported.

## SMTP Delivery

qmail includes the following SMTP delivery features:

- **RPC compliant.** Complies with RFC 2821 (SMTP), RFC 974 (Mail Routing), and RFC 1123.
- **8-bit clean.** Sends 7-bit ASCII characters as well as 8-bit extended characters.
- **Automatic downed host backoffs.** If a host is unreachable, qmail waits an hour before trying again.
- **Artificial routing.** Default routes—for example, via DNS MX records—can be overridden using qmail’s `smtproutes` configuration file, which is equivalent to Sendmail’s `mailertable`.
- **Per-buffer timeouts.** Each new buffer of data to the remote SMTP server has its own time limit.
- **Passive SMTP queue.** Mail can be queued to a mailbox for scheduled delivery using the `serialmail` package. This is useful for SLIP/PPP.
- **AutoTURN support.** Using the `serialmail` package, clients can tell the server to send them their queued mail.

## Forwarding and Mailing Lists

qmail supports forwarding and mailing lists:

- **Sendmail .forward compatibility.** Using the dot-forward package, Sendmail-style .forward files can be used.
- **Hashed forwarding databases.** The fastforward package implements a high-performance forwarding database.
- **Sendmail /etc/aliases compatibility.** The fastforward package includes a clone of the newaliases command that supports Sendmail-style alias databases.
- **Address wildcards.** Using .qmail-default, .qmail-something-default, and so on, users and mail administrators can specify the disposition of messages to multiple addresses.
- **Mailing-list owners.** If a message is forwarded to .qmail-something, and .qmail-something-owner exists, it automatically gets a return address of `user-something-owner@domain`. This diverts bounces and vacation messages from going to the sender.
- **Variable Envelope Return Path (VERP) support.** VERP allows reliable automatic recipient identification for mailing-list bounces.
- **Delivered-To header field.** Each “final” delivery causes the addition of a Delivered-To header field containing the recipient address. If a message already contains a Delivered-To for the current recipient, the message is rejected. This enables automatic loop prevention, even across hosts.
- **Automatic subscription management.** The ezmlm package allows users to subscribe and unsubscribe themselves from mailing lists. It also tracks bounces and removes invalid addresses.

## Local Delivery

qmail supports the following local delivery features:

- **User-controlled address hierarchy.** User lucy controls mail addressed to `lucy-anything@domain`.

- **Supports Unix mbox mailboxes.** Supports the traditional Unix mailbox format: multiple messages in one file, separated by From lines.
- **Supports maildir mailboxes.** Provides reliable delivery to mailboxes—even over Network File System (NFS)—using the maildir mailbox format.
- **User-controlled program delivery.** Users can direct messages to filters like Procmail or maildrop, custom scripts, vacation reminders, and so on.
- **Optional new-mail notification.** The qbiff program can be used to notify users upon receipt of new messages.
- **Optional Notice-Requested-Upon-Delivery-To (NRUDT) return receipts.** The qreceipt program can be used to respond to NRUDTs.
- **Conditional filtering.** The condredirect and bouncesaying programs can be used to conditionally intercept messages.

## POP3 Service

qmail includes a POP3 server with these features:

- **RFC compliant.** Complies with RFC 1939 (POP3).
- **UIDL support.** qmail-pop3d implements the optional UIDL command, which lists the unique ID of one or more messages.
- **TOP support.** qmail-pop3d implements the optional TOP command, which returns the header and beginning of a specified message.
- **Modular password checking.** The checkpassword package, available separately, implements password validation. Versions supporting different authentication methods and user databases are available.
- **Authenticated Post Office Protocol (APOP) hook.** APOP is available using an alternative checkpassword module.

# APPENDIX E

# Error Messages

MAIL TRANSFER AGENTS (MTAs) are complex systems, and there are thousands of things that go wrong in the process of accepting a message and delivering it locally or remotely. Because qmail was implemented with high reliability as a goal, it's particularly careful in checking for error conditions. Its error messages are generally very descriptive, but sometimes it can be difficult to pinpoint the exact cause of the problem that qmail is complaining about. Rather than attempting to explain each of the hundreds of error messages qmail can generate, this appendix provides some guidance for interpreting these messages.

Error messages can show up in three places: in an interactive shell session, in one of the log files, or in a bounce message.

## Interactive Error Messages

Interactive errors are usually the result of “pilot error”—incorrect syntax, permission problems, typographic errors, and so on. First determine whether the error message is coming from the shell or the qmail command you’re running. If the message is from a qmail program, consult the `man` page or Chapter 3 (for user commands) or Chapter 5 (for management commands).

If you run across an error message and you can’t figure out what the problem is, try searching the archives of the qmail mailing list. Chances are good that someone has already been there, asked that question, and gotten an answer. The list search engine is at <http://www-archive.ornl.gov:8000/>.

## Log Messages

The only logs written by qmail are produced by `qmail-send`. (The `qmail-smtpd` logs are really from `tcpserver`). The errors logged by `qmail-send` come from itself, `qmail-local`, or `qmail-remote`.

### `qmail-send` Messages

Errors logged by `qmail-send` contain the string “alert:” if the problem is critical or the string “warning:” if the problem is serious but not crippling. Errors of either severity are serious and should be investigated immediately.

Critical problems include the inability to access the qmail home directory, the queue, or the control files; the inability to talk to its helpers: qmail-lspawn, qmail-rspawn, or qmail-clean; and the inability to append to a queue/bounce file. Verify that the directory it's complaining about exists and has the right owner/group/mode. The easiest way to do this is to stop qmail and do make check from the build directory as root.

The message “alert: cannot start: hath the daemon spawn no fire?” means that the communication channels to qmail-lspawn, qmail-rspawn, or qmail-clean weren’t set up—perhaps because qmail-start had trouble starting them.

## *qmail-local Messages*

Errors logged by qmail-local are usually temporary and related to permission problems or full file systems (including file systems with space but no free inodes).

Two permanent errors are:

- This message is looping: it already has my Delivered-To line. (#5.4.6). A looping message is a message delivered twice to the same address—usually due to a dot-qmail forwarding instruction that forwards to an address that forwards it back. Check the Received fields and logs to find the culprit.
- Sorry, no mailbox here by that name. (#5.1.1). This is obvious enough, but sometimes you get this message when you’re *sure* the mailbox exists. Well, you’re probably wrong. The problem is usually because of qmail’s narrow definition of a valid mail user, as detailed in the `qmail-getpw` man page and in Chapter 5, “Managing qmail.” Typical problems include uppercase characters in the username and home directories not owned by the user.



**TIP** See the “RFC 1893 Status Codes” section later in this appendix for an explanation of the “(#x.x.x)” codes.

Common temporary errors include:

- Uh-oh: home directory is writable. (#4.7.0).
- Uh-oh: .qmail file is writable. (#4.7.0).

Both of these errors indicate that either the user's home directory or dot-qmail file is writable by users that the *conf-patrn* compile-time configuration setting prohibits. (See Chapter 2, "Installing qmail".)

## qmail-remote *Messages*

qmail-remote generates its own messages and relays messages from remote hosts. Remote messages are sent by various MTAs but are usually pretty easy to interpret.

Common qmail-remote error messages include:

- Sorry, I wasn't able to establish an SMTP connection. (#4.4.1). qmail-remote has not been able to connect to the remote server. Most likely the remote server is down, but it could also indicate a network problem anywhere between the two systems.
- CNAME lookup failed temporarily. (#4.4.3). qmail-remote tried to look up the Internet Protocol (IP) address of the remote server in the Domain Name System (DNS) and received a temporary error.
- Sorry, I couldn't find any host by that name. (#4.1.2). Again, qmail-remote was unable to find an IP address. This is a temporary error, so it'll keep trying.
- Sorry. Although I'm listed as a best-preference MX or A for that host, it isn't in my control/locals file, so I don't treat it as local. (#5.4.6). qmail-remote looked up the IP address for a remote host and found that it was the local host. However, the host wasn't listed in control/locals or the delivery would have been given to qmail-local. Either fix control/locals or your DNS records.

## Bounce *Messages*

qmail's bounce messages are in a format called QSBMF (qmail-send bounce message format), which is documented on the Web (<http://cr.yp.to/proto/qsbmf.txt>). RFCs 1892, 1893, and 1894 define another bounce message format called Delivery Status Notification (DSN). The status codes defined in RFC 1893 are also used by QSBMF and other non-DSN bounce message formats. Some MTAs still use their own ad-hoc bounce message formats.

## QSBMF

QSBMF messages are designed to be simultaneously human-friendly and easily parsed by automated bounce handlers. Listing E-1 shows a typical bounce message.

### *Listing E-1. A QSBMF bounce message*

```
From: MAILER-DAEMON@dolphin.example.com
To: dave@dolphin.example.com
Subject: failure notice

Hi. This is the qmail-send program at dolphin.example.com.
I'm afraid I wasn't able to deliver your message to the following addresses.
This is a permanent error; I've given up. Sorry it didn't work out.
```

```
<nosuchuser@dolphin.example.com>:
Sorry, no mailbox here by that name. (#5.1.1)
```

-- Below this line is a copy of the message.

```
Return-Path: <dave@dolphin.example.com>
Received: (qmail 3458 invoked by uid 500); 26 Aug 2001 21:56:48 -0000
Date: 26 Aug 2001 21:56:48 -0000
Message-ID: <20010826215648.3457.qmail@dolphin.example.com>
From: dave@dolphin.example.com
to: nosuchuser@dolphin.example.com
```

The body of a QSBMF message consists of four parts: an introductory paragraph, a series of one or more recipient paragraphs, a break paragraph, and a copy of the original message. Blank lines separate the paragraphs.

In this case, the introductory paragraph is:

```
Hi. This is the qmail-send program at dolphin.example.com.
I'm afraid I wasn't able to deliver your message to the following addresses.
This is a permanent error; I've given up. Sorry it didn't work out.
```

The initial string, Hi. This is the... , identifies the message as QSBMF. This paragraph is intended for human readers and identifies the source of the message.

The recipient paragraph in this example is this:

```
<nosuchuser@dolphin.example.com>:
Sorry, no mailbox here by that name. (#5.1.1)
```

The first line identifies the problematic recipient address, and the second line is a description of the problem. The (#5.1.1) is an RFC 1893 status code—these will be explained in the next section.

The break paragraph starts with a - character. In this example, it's this line:

-- Below this line is a copy of the message.

The remainder of the bounce message is the copy of the original message.

An interesting special case is that of the double bounce: the bounce message sent to the postmaster when a bounce message is undeliverable. With double bounces, the message included is the original bounce message—a QSBMF bounce enclosed in another QSBMF bounce.

## RFC 1893 Status Codes

These are three-digit codes displayed as *C.S.D*, where *C* is the *class* sub-code, *S* is the *subject* sub-code, and *D* is the *detail* sub-code.

The class sub-code is one of three values: 2 (success), 4 (temporary error), or 5 (permanent error). They correlate with the initial digits of SMTP status codes.

The subject sub-code has seven possible values, as listed in Table E-1.

*Table E-1. RFC 1893 Subject Sub-Codes*

CODE	NAME	MEANING
x.0.x	Other or Undefined Status	Problem unknown or undefined
x.1.x	Address Status	Problem with sender or recipient address syntax or validity
x.2.x	Mailbox Status	Problem with the recipient's mailbox
x.3.x	Mail System Status	Problem with the recipient host's mail system
x.4.x	Network and Routing Status	Problem with network or routing
x.5.x	Mail Delivery Protocol Status	Problem with mail delivery
x.6.x	Message Content or Media Status	Problem with message content or format
x.7.x	Security or Policy Status	Problem with security or policy

The detail sub-codes vary with subject sub-code. The only valid detail sub-code for subject sub-code 0 is 0: If an MTA doesn't know what subject sub-code applies, it doesn't make sense to categorize it at a lower level. The detail sub-codes for the other subject sub-codes are listed in Tables E-2 through E-8.

*Table E-2. RFC 1893 Address Status Detail Sub-Codes*


---

<b>CODE</b>	<b>MEANING</b>
x.1.0	Unknown problem with an address specified in this message
x.1.1	Nonexistent recipient (part left of @)
x.1.2	Invalid destination host (part right of @)
x.1.3	Bad destination address syntax
x.1.4	Ambiguous destination
x.1.5	Valid destination
x.1.6	Recipient has moved without a forwarding address
x.1.7	Bad sender's address syntax
x.1.8	Bad sender's host

---

*Table E-3. RFC 1893 Mailbox Status Detail Sub-Codes*


---

<b>CODE</b>	<b>MEANING</b>
x.2.0	Unknown problem with an existing mailbox
x.2.1	Mailbox disabled
x.2.2	Mailbox full
x.2.3	Message too big
x.2.4	Problem sending to mailing list

---

*Table E-4. RFC 1893 Mail System Status Detail Sub-Codes*


---

<b>CODE</b>	<b>MEANING</b>
x.3.0	Unknown/other problem with destination host's mail system
x.3.1	Mail system full
x.3.2	Not accepting messages
x.3.3	Mail system doesn't support requested feature
x.3.4	Message too big
x.3.5	Mail system misconfigured

---

*Table E-5. RFC 1893 Network and Routing Status Detail Sub-Codes*

<b>CODE</b>	<b>MEANING</b>
x.4.0	Unknown/other network problem
x.4.1	No answer from host
x.4.2	Bad connection
x.4.3	Directory service failure
x.4.4	Unable to route
x.4.5	Mail system congestion
x.4.6	Routing loop detected
x.4.7	Delivery time expired

*Table E-6. RFC 1893 Mail Delivery Protocol Status Detail Sub-Codes*

<b>CODE</b>	<b>MEANING</b>
x.5.0	Unknown/other problem delivering to next hop
x.5.1	Invalid command
x.5.2	Syntax error
x.5.3	Too many recipients
x.5.4	Invalid command arguments
x.5.5	Wrong protocol version

*Table E-7. RFC 1893 Message Content or Message Media Status Detail Sub-Codes*

<b>CODE</b>	<b>MEANING</b>
x.6.0	Unknown/other problem with message content
x.6.1	Media (format) not supported
x.6.2	Conversion necessary but prohibited
x.6.3	Conversion necessary but not supported
x.6.4	Message converted but with data loss
x.6.5	Conversion failed

*Table E-8. RFC 1893 Security or Policy Status Detail Sub-Codes*

<b>CODE</b>	<b>MEANING</b>
x.7.0	Unknown/other security problem
x.7.1	Delivery not authorized, message refused
x.7.2	Delivery to mailing list prohibited
x.7.3	Security conversion required but not possible
x.7.4	Security features not supported
x.7.5	Cryptographic failure
x.7.6	Cryptographic algorithm not supported
x.7.7	Message integrity failure

## APPENDIX F

# Gotchas

FOR THE MOST PART, qmail works the way people expect it to work. There are, however, a few *gotchas*: quirks in qmail's behavior that frequently cause problems for beginners.

### *qmail Doesn't Deliver to Superusers*

To prevent the possibility of qmail-local running commands as a privileged user, qmail ignores all users whose user ID is zero. This is documented in the `qmail-getpw` man page.

That doesn't mean qmail won't deliver to root, it just means that such a delivery will have to be handled by a non-privileged user. Typically, one creates an alias for root by populating `/var/qmail/alias/.qmail-root` with an entry that forwards to the system administrator's unprivileged account.

### *qmail Doesn't Deliver to Users Who Don't Own Their Home Directory*

This is another security feature and just good general practice. This is documented in the `qmail-getpw` man page.

### *qmail Doesn't Deliver to Users Whose Usernames Contain Uppercase Letters*

qmail converts the entire “local part”—everything before the @ in an address—to lowercase. The man page doesn't come out and say that, but the code does. The fact that it ignores users with uppercase characters is documented in the `qmail-getpw` man page.

## *qmail Replaces Dots (.) in Extension Addresses with Colons (:)*

This is another security feature. The purpose is to prevent extension addresses from backing up the file tree using . . . Without this restriction, a malicious user could attempt a delivery to joe-.../jane/foo@example.com hoping to attempt delivery via the dot-qmail file ~joe/.qmail-.../jane/foo—perhaps disclosing the contents of ~jane/foo in the form of a bounce message. By replacing dots with colons, qmail ensures that all dot-qmail files for a user are under their home directory. This is documented in the `qmail-local` man page.

## *qmail Converts Uppercase Characters in Extension Addresses to Lowercase*

This is another result of the fact that qmail lowerscases the entire local part of addresses, and it is documented in the `qmail-local` man page.

## *qmail Doesn't Use /etc/hosts*

qmail *never* uses `/etc/hosts` to determine the Internet Protocol (IP) address associated with a host name. If you use names in control files, qmail must have access to a name server.

It *is* possible to run qmail on systems without access to a name server. Hosts in control files can be specified by IP address by enclosing them in square brackets ([ ]), for example:

[10.1.2.219]

Actually, the square brackets aren't always necessary—but it's a good idea to use them anyway.

## *qmail Doesn't Log SMTP Activity*

For whatever reasons, qmail doesn't log SMTP connections, rejections, invalid commands, or valid commands. `tcpserver` can be used to log connections, and `recordio` can be used to log the entire SMTP dialogue. `recordio` is part of the `ucspi-tcp` package (see Appendix B, “Related Packages”). The procedure is documented in Chapter 7, “Troubleshooting qmail.”

## *qmail Doesn't Generate Deferral Notices*

If Sendmail is unable to deliver a message within a few hours, typically four, it sends a deferral notice to the originator. These notices look like bounce messages but don't indicate that the delivery has failed permanently yet.

qmail doesn't send such warnings. A temporarily undeliverable message will only be returned to the originator after it spends at least *queuelifetime* seconds in the queue.

## *qmail Is Slow If trigger Is Wrong*

qmail-queue and qmail-send communicate via a named pipe called /var/qmail/queue/lock/trigger. If this pipe gets messed up, qmail-send doesn't notice new messages for up to 25 minutes.

The best way to ensure that trigger is set up right is to run make check from the source directory. If that's not possible, make sure it looks like this:

```
# ls -l /var/qmail/queue/lock/trigger
prw-w-w- 1 qmails  qmail      0 Jul  5 21:25 /var/qmail/queue/lock/trigger
#
```

Pay particular attention to the p at the beginning of the line, which says that it's a named pipe, the mode (must be world-writable), the owner, and the group.

## *DNS or IDENT Lookups Can Make SMTP Slow*

If qmail-smtpd is slow to respond to connections, the problem is probably because of Domain Name System (DNS) reverse lookups or IDENT lookups having to time out. If the problem is DNS-related, the best fix is to correct your DNS configuration. Another approach is to configure tcpserver not to attempt the lookups. That might be an option if you don't need the information these lookups provide. This can be accomplished by removing the -h, -p, and -r options and adding -H, -P, -R, and -l 0 (*ell zero*).

## *qmail-smtpd Accepts Mail for All Recipients*

The control/rcpthosts file specifies the hosts for which qmail-smtpd will accept messages (unless RELAYCLIENT is set to allow relaying). However, qmail-smtpd does not attempt to validate the recipient. If an invalid recipient is specified, qmail-send will generate a bounce message. This is often a problem for simplistic

open-relay testing programs that wrongly assume successful SMTP injection means successful delivery.

For example, a common relay test is to send a message to *recipient%testhost@yourdomain*, which relies on the Sendmail percent hack: stripping *@yourdomain*, replacing the % with @, resulting in *recipient@testhost*, and reinjecting the message to the new address.

With qmail, unless the control/percenthack file is in use, such a test merely tries to deliver a message to the local mailbox *recipient%testhost*, which probably doesn't exist. The result is a bounce message sent to the return address specified by the relay tester.

### *qmail-smtpd Doesn't Automatically Relay for the Local Host*

By default, *qmail-smtpd* doesn't accept messages for remote hosts—even if the SMTP client is a Mail User Agent (MUA) running on the local host. If you want to allow local MUAs to inject mail via SMTP, you must enable selective relaying and grant relay access to 127.0.0.1, the local host. See Chapter 3, “Configuring qmail: The Basics,” for more information about selective relaying.

# Index

## SYMBOLS

/ (trailing slashes), and qmail configuration, 242  
/bin/mail, 240  
/etc/group, 60  
/etc/hosts gotcha, 466  
/var/log, 203  
/var/qmail  
    and file structure, 17  
    symbolic links under, 33  
/var/qmail/rc script, 67  
. (dot)  
    in extension addresses gotcha, 466  
.forward, and delivery disposition, 237  
.qmail. *See* dot-qmail  
-i qmail-inject option,  
    and QMAILINJECT, 130  
-t sendmail option, 124  
@ (at sign), in POP3 usernames, 383

## A

A Mail Virus Scanner (AMaViS), 404  
addresses  
    distinction from programs and files, 4  
    extension addresses, 141  
        case of characters gotcha, 466  
        dots (.) in extension addresses  
            gotcha, 466  
    filtering messages and, 145–146  
    junk mail control and, 287, 294  
    receiving messages and, 141–142  
qmail-inject  
    name-address format and, 131  
    address-comment format and, 131

qualification and, 419  
setting envelope sender addresses, 129–130  
tagged addresses, 294  
adjusting message lifetimes/retry schedules, 192  
advantages and disadvantages of qmail, 3–9  
disadvantages, 9, 16  
performance, 6–7  
reliability, 7  
security, 3–6  
simplicity, 8–9  
advisory spam controls, defined, 285, 287  
AIX, creating users and groups in, 60  
aliases, 116  
    /var/qmail directory, 17  
advanced configuring, 245–249  
configuring, 116–117  
    creating system aliases, 49, 77  
    support of host name aliases, 14  
Allbery, Russ, and Majordomo FAQs, 305, 313  
alarm function in qmailctl, 162  
AMaViS, 404  
Anti-Spam HOWTO Web site, 285  
APOP authentication, 358–362  
Apress Web site, 27  
architecture. *See* file structure; modular system architecture  
assign, and single dot (.), 117  
at sign (@), in POP3 usernames, 383  
authenticated relaying, 113–115  
authentication  
    of IMAP, 367–369

- authentication (*continued*)  
of POP3, 353  
of SMTP, 264–273
- B**  
backup mail exchanger, 226–228  
badmailfrom  
control file, 95  
controlling junk mail and, 288–289  
Berkeley Internet Name Daemon  
(BIND), 432  
Bernstein, Daniel J.  
checkpassword and, 431  
cryptography and, 15  
djbdns and, 432  
dot-forward and, 433  
efficient code of, 9  
ezmlm and, 297, 433  
fastforward and, 434  
mess822 and, 435  
qmail security guarantee and, 4  
qmailanalog and, 437  
qmail's beginnings and, 1, 15  
QMTP and, 254  
rlsmtpd and, 289  
serialmail and, 439–440  
ucspi-tcp and, 441  
VERP and, 395  
big-concurrency patch, 252  
big-todo patch, 277  
bin /var/qmail directory, 17  
binaries, and installation of qmail, 28  
BIND (Berkeley Internet Name Daemon),  
432  
blacklisting, 292, 293  
blacklists. *See under* DNS  
boot /var/qmail directory, 17  
boot script  
configuring at installation, 57–58  
setting up, 45–46  
bounce  
messages, and error messages,  
459–464
- QSBMF, 460–461  
RFC 1893 status codes,  
461–464  
queue subdirectory contents, 18  
bouncefrom control file, 95  
bouncehost control file, 95–96  
bounces, handling of, 13–14  
bouncesaying dot-qmail utility, 148–149  
rc.local  
/var/qmail/bin/qmailctl and system  
boots, 71–72  
updating, 46  
Budney, Len, and safecat, 439  
buffer size, and DNS patches, 252  
build environments, verification for  
installation  
quick-start and, 38–39  
step-by-step installation and, 53–54  
building qmail  
quick-start and, 42–43  
step-by-step installation and, 61–63  
building source, 55–66  
compile-time configuration settings,  
57–58  
creating directories, 59  
creating users and groups, 59–61  
daemontools, installing, 64–66  
qmail, installing, 61–63  
ucspi-tcp, installing, 63–64  
unpacking distributions, 55–57
- C**  
C runtime library, defined, 6  
caching, and DNS, 284  
Cazabon, Charles, getmail and, 371, 434  
Cc fields, and troubleshooting, 215  
cdb function in qmailctl, 164  
certificates, self-signed vs. CA-signed,  
266  
channels for qmail-send, listed, 179  
checkpassword programs  
qmail-pop3d and, 329, 431

- using different, 359
- CNAME lookup failed temporarily error
  - message, 459
- code, *See also* source-code
  - efficiency of, 9
  - security and, 6
- code listings
  - /var/qmail/rc script, 67
  - a simple mailing list, 296
  - env example file, 135
  - msg example file, 125–126, 135
  - qmail-procmail, 147
  - qmailctl script, 69–71
- commands
  - ezmlm command addresses, 304–305
  - ezmlm-sub and ezmlm-unsub, 304
  - for listing processes, 200
  - grep commands, delivery and, 141
  - maildirmtp command, 232
  - openssl command, 267
  - qmail management commands,
    - 165–187
    - qmail-clean, 165
    - qmail-getpw, 166–167
    - qmail-local, 168–169
    - qmail-lspawn, 169
    - qmail-newmrh, 170
    - qmail-newu, 170
    - qmail-pop3d, 170–171
    - qmail-popup, 171–172
    - qmail-pw2u, 172–175
    - qmail-qmqpc, 175
    - qmail-qmqpd, 175
    - qmail-qmtpd, 176
    - qmail-qread, 176–177
    - qmail-qstat, 177
    - qmail-remote, 177–179
    - qmail-rspawn, 179
    - qmail-send, 179–180
    - qmail-showctl, 180–182
    - qmail-smtpd, 183
    - qmail-start, 184
    - qmail-tcpok, 184–185
    - qmail-tcpt, 185
- splogger, 185–186
- tcp-env, 186–187
- sendmail command, 419
- setlock command, 232
- comments (personal names), setting, 129
- compile-time settings, configuring at
  - installation, 57–58
- concurrenties, tuning, 278
- concurrencyimap control file, 347, 365
- concurrencyincoming control file, 96
- concurrencylocal control file, 96–97, 252
- concurrencypop3 control file, 330,
  - 335, 340
- concurrencyremote control file, 97, 252
- condredirect dot-qmail utility, 150
- conf-split, adjusting, 275–277
- configuring
  - aliases, 116–117
  - at installation, 66–77
    - boot script, 66–68
    - SMTP access control, 76
    - system aliases, 77
    - system startup files, 68–76
  - getmail, 372
  - multiple host names, 115
  - qmail-users mechanism, 117–119
  - scalable servers, 395–401
    - mail, 396–399
    - mailbox delivery and service,
      - 399–401
  - system startup files, 73–76
  - virtual domains, 115–116
  - VMailMgr, 380–381
  - Vpopmail, 386–387
- configuring qmail
  - aliases, 116–117
  - at installation, compile-time settings,
    - 57–58
- control files reference, 93–110, *See also under* files
  - badmailfrom, 95
  - bouncefrom, 95
  - bouncehost, 95–96
  - concurrencyincoming, 96

configuring qmail (*continued*)  
    concurrencylocal, 96–97  
    concurrencyremote, 97  
    datatypes, 97–98  
    defaultdelivery, 98–99  
    defaultdomain, 99  
    defaulthost, 99–100  
    doublebouncehost, 100  
    doublebounceto, 100  
    envnoathost, 101  
    helohost, 101  
    idhost, 101  
    localiphost, 102  
    locals, 102  
    me, 102–103  
    morerecpthosts, 103  
    percenthack, 103–104  
    plusdomain, 104  
    qmqp servers, 104  
    queuelifetime, 104–105  
    rcpthosts, 105–106  
    smtpgreeting, 106  
    smtproutes, 107  
    timeoutconnect, 108  
    timeoutremote, 108  
    timeoutsmtpd, 108–109  
    understanding format of, 94  
    virtualdomains, 109–110  
multiple host names, 115  
qmail-users mechanism,  
    117–119  
relaying, 110–115  
    allowing selective, 112–115  
    control of, 111–112  
    disabling, 112  
    virtual domains, 115–116  
configuring qmail, advanced options,  
    225–284  
migrating from Sendmail to qmail,  
    237–249  
    aliases, 245–249  
    dot-forward package, 237–239  
    mailbox location and format,  
        239–245

modifying source-code, 249–254  
frequently used patches, 251–254  
installing patches, 250–251  
performance tuning, 273–284  
    determining problems, 274  
    tuning network, 283–284  
    tuning qmail, 275–278  
    tuning system hardware, 280–283  
    tuning system software, 279–280  
QMQP, 258–264  
    nullmailer and, 264  
    setting up QMQP clients (mini-qmail), 261–264  
    setting up QMQP services, 259–261  
QMTP, 254, 255–258  
    basics, 255–258  
    setting up QMTP services,  
        254–255  
securing SMTP, 264–273  
    SSL-wrapped SMTP, 269–273  
    STARTTLS, 265–269  
setting up typical configurations,  
    226–236  
    backup mail exchanger, 226–228  
    dial-up client, 230–233  
    general purpose mail server, 226  
    mailbox servers, 235–236  
    null client, 228–230  
    smart host, 233–235  
connectivity, and tuning networks, 283  
consultants, for support, 25  
cont function in qmailctl, 164  
control /var/qmail directory, 17  
control files  
    badmailfrom, 95  
    bouncefrom, 95  
    bouncehost, 95–96  
    concurrencyimap, 347, 365  
    concurrencyincoming, 96  
    concurrencylocal, 96–97  
    concurrencypop3, 330,  
        335, 340  
    concurrencyremote, 97  
    datatypes, 97–98

- defaultdelivery, 98–99
- defaultdomain, 99
- defaulthost, 99–100
- doublebouncehost, 100
- doublebounceto, 100
- envnoauthhost, 101
- helohost, 101
- idhost, 101
- localiphost, 102
- locals, 102
- me, 102–103
- morerecphosts, 103
- overriding qmail-inject's control files, 130
- percenthack, 103–104
- plusdomain, 104
- qmqpsservers, 104
- queuelifetime, 104–105
- rcpthosts, 105–106
- smtpgreeting, 106
- smtproutes, 107
- timeoutconnect, 108
- timeoutremote, 108
- timeoutsmtpd, 108–109
- virtualdomains, 109–110
- copyright of qmail, 19–20
- Courier, 11
- Courier-IMAP, 350–353
  - basics, 432
  - installing, 350–352
  - SSL wrapping, 362–364
  - testing, 352–353
- CPU (Central Processing Unit) tuning, 280
- CRAM-MD5 authentication, 367–369
  
- D**
- daemons, checking, 79–80
- daemontools, *See also* multilog
  - basics, 432
  - defined, 37
- quick-start installation, 43–45
- step-by-step installation, 64–66
- datatypes control file, 97–98
- Date fields, troubleshooting, 214
- datemail utility, 150–151
- Davis, Christopher K., and DNS patches, 252
- defaultdelivery values, mailbox formats and locations and, 68
- defaultdomain control file, 99
- deferral notices gotcha, 467
- Delivered-To field, and troubleshooting, 211–212
- delivery
  - conditional, 141
  - distributing, 399–400
  - dot-qmail delivery types listed, 136
  - error handling, 139–140
  - forward delivery, 138–139
  - local delivery options, 15
  - maildir delivery, 138
  - mbox delivery, 137
  - migrating to home directory delivery, 241–242
  - multiple deliveries, 139–141
  - qmail-send log messages of, 203–206
  - program delivery, 137
  - qmail-send and, 426
  - Sendmail-style delivery, 240–241
  - simple mailing lists and, 296
  - single-recipient vs. multiple-recipient, 391–394
  - unsuccessful, 204–206, 218, 220–222
- diagnostic tools for queues, 188–191
- dial-up client configuration, 230–233
- directories, *See also* subdirectories
  - /var/qmail listed, 17
  - creating at installation, 59
  - home directories
    - migrating to home directory delivery, 241–242
    - moving mailboxes to, 243–244

- directories (*continued*)  
    installation directory comparison for  
        qmail+patches vs. tarball, 90  
    master  
        creating, 40  
        in /var/qmail, 33  
disadvantages  
    of modular approach, 16  
    of qmail, 9  
disk I/O tuning, 281  
disk interfaces, and tuning disk I/O, 281  
distances, and MX records, 226  
djbdns server package  
    basics, 432–433  
    dnscache and, 251–252  
DNS  
    blacklists, 289–290  
    directing mail to virtual domains and,  
        380, 386  
    DNS caching, and tuning networks,  
        284  
    DNS lookups and SMTP slowness  
        gotcha, 467  
    DNS patches, 251–252  
    dnscache, and djbdns, 251–252  
doc /var/qmail directory, 17  
documentation, 20–23  
documents installed under  
    /var/qmail/doc, 21  
domain (host), setting in messages,  
    128–129  
domain mailboxes, and Fetchmail, 370  
domains, and routing by, 14  
doqueue function in qmailctl, 162  
dot(.)  
    in extension addresses gotcha, 466  
dot-forward package, 237–239, 433  
dot-qmail files, 136  
dot-qmail utilities, 148–156  
    bouncesaying, 148–149  
    condredirect, 150  
    except, 151  
    forward, 152  
    preline, 155  
    qreceipt, 155–156  
doublebouncehost control file, 100  
doublebounceto control file, 100
- E**
- e-mail information, 446  
EICAR (European Institute for Computer  
    Anti-Virus Research), 413–414  
822field, and extracting header fields, 141  
elq utility, 151  
env example file listing, 135  
envelopes  
    envelope return paths, variable,  
        394–395  
    envelope sender address, setting,  
        129–130  
    vs. headers, 444–446  
environment variables  
    qmail-command, 142–144  
    qmail-inject, 127–133  
    RELAYCLIENT, 422  
    requirements for qmail-smtpd,  
        421–423  
    tcp-env, 187  
envnoahost control file, 101  
error messages, 457–464  
    beginning with “Sorry . . .” 458–459  
    bounce, 459–463  
    interactive, 457  
    log, 457–459  
errors, *See also* fixing common problems;  
    troubleshooting  
    in multiple deliveries, 139–140  
European Institute for Computer Anti-  
    Virus Research (EICAR),  
        413–414  
example header analysis, troubleshooting,  
    216–217  
except dot-qmail utility, 151  
Exim, 11–12  
exit status codes, qmail-queue, 134  
Ext2 queue workarounds, 34

- extension addresses, 141
    - dots (.) in gotcha, 466
    - filtering messages and, 145–146
    - junk mail control and, 287, 294
  - ezmlm, 297–305
    - basics, 297, 433–434
    - installing, 298–301
    - subscribing and unsubscribing, 304–305
    - testing, 301–302
    - understanding ezmlm-idx, 298
    - using, 302–304
    - VERP and, 297, 395
  - ezmlm-idx, 298, 433–434
    - `ezmlm@list.cr.yp.to` mailing list, 25
- F**
- FAQs sites, 21
  - fastforward
    - basics, 434
    - implementing aliases using, 247–249
  - features of qmail, 449–455
    - basics, 12–15
    - message construction features, 450
    - POP3 service features, 455
    - queue management features, 451–452
    - routing by domain features, 453
    - security features, 449–450
    - setup features, 449
    - SMTP delivery features, 453
    - SMTP service features, 450–451
  - Fetchmail, 369–370
  - fields
    - `822field`, 141
    - required, 420
    - Resent- fields, and qmail-inject, 420
    - troubleshooting Cc fields, 215
    - troubleshooting using
      - Date fields, 214
      - Message-ID fields, 214
- Received fields, 212–213
    - Resent- fields, 215
    - To fields, 215
  - file structure, 17
  - file systems
    - queue, and requirements, 33–35
    - selecting, 279–280
  - files, *See also* control files
    - creating basic configuration files, 43
    - dot-qmail files, 136, 287
    - for Majordomo lists, 313
    - information storage, and inodes, 204
    - names of in grep output, 193
    - PIC files, 19
    - system startup files configuration, 68–76
      - qmail services, 73–76
      - `qmailctl` script, 68–73
  - filtering mail, 144–148
    - extension addresses, 145–146
    - junk mail, 292–293
    - maildrop, 148
    - procmail, 146–148
  - fixing common problems, *See also* error messages; troubleshooting
    - local users can't send mail, 223
    - mail not accepted from remote hosts, 219
    - mail not delivered to local user, 220–221
    - mail not delivered to remote address, 221–222
    - mail not retrievable by users, 222
  - flagging junk mail, 292, 293
  - flush function in `qmailctl`, 162
  - formats
    - boot script and mailbox formats, 66–68
    - mailbox formats, 243
    - selection for mailboxes, 29–32
  - forward deliveries, 138–139
  - forward dot-qmail utility, 152
  - forwarding support, 14
  - FreeBSD, 60

## From field

and troubleshooting, 215

setting, 129

## functions

mail servers functions, 396

MLMs

basic functions, 325

list maintenance function, 325

resending function, 325

## qmailctl

alarm function, 162

cdb function, 164

cont function, 164

doqueue function, 162

flush function, 162

help function, 165

hup function, 163

pause function, 163

queue function, 164

reload function, 163

restart function, 162

script functions, 161–165

start function, 161

stat function, 163, 189, 201

## G

general purpose mail server, 226

getmail, 371–374, 434

GNU Privacy Guard (GnuPG), 264

gotchas, 465–468

Greenwich Mean Time (GMT) in message headers, 213

grep command, file matches and, 193

groups, creating, 41–42, 59–61

Guenter, Bruce

qlogtools and, 436

qmail-autoreponder and, 436

qmail-qfilter and, 436

QMAILQUEUE patch and, 405

relay-ctrl package and, 114

syncdir and, 440

VMailMgr and, 377, 441

## H

Haar, Jason, and Qmail-Scanner, 404–414, 437

Hampton, Catherine A., and SpamBouncer, 293

hard delivery errors, 140

Hardie, Chris, and anti-spam, 285

hardware. *See* system hardware

harnesses, defined, 404

Hazel, Philip, and Exim, 11

header fields. *See* message headers, using to troubleshoot

headers vs. envelopes, 444–446

helohost control file, 101

help function in qmailctl, 165

history of qmail, 1, 15

host-based relaying, 112–113

host (domain) setting in messages, 128–129

host names. *See* multiple host names

hup function in qmailctl, 163

## I

I/O. *See* disk I/O

IDENT lookups and SMTP slowness

gotcha, 467

idhost control file, 101

IMAP (Internet Mail Access Protocol) servers

basic description, 327

Courier-IMAP, 350–353

installing, 350–352

testing, 352–353

distributed service, 401

factors in selecting, 327–328

retrieving mail, 369–374

Fetchmail, 369–370

getmail, 371–374

securing, 362–369

enabling SSL wrapper, 363–364

proxy-wrapping, 364–367

CRAM-MD5 authentication, 367–368  
wrapping with SSL, 362–367

University of Washington IMAP, 344–350  
installing with maildir support, 344–349  
testing, 349–350

inetd  
inetd.conf entries, 187  
limitations of, 35–36  
running of qmail-pop3d and, 329

info queue subdirectory contents, 18

init.d  
linking directories, 46–47  
qmailctl System V init.d script, 160

injection defined, 121

inodes defined, 204

installation, 27–91  
from RPMs, 87–90  
assumptions about, 88  
cautions, 90  
downloading RPMs, 88  
installing RPMs, 88–90  
selecting RPMs, 87–88

INSTALL file and, 27

preparation for, 28–38  
binary and source-codes, 28–29  
files placement, 32–35  
selecting mailbox formats and locations, 29–32  
support utilities, 35–38  
tarballs and operating system-specific packages, 28–29

quick-start instructions, 38–51  
boot script setup, 45–46  
daemontools, installing, 43–45  
installed mailer, stopping and disabling, 50  
locating source, 39–40  
logging directories setup, 49  
master directory, creating, 40  
qmail, installing, 42–43  
qmailctl script, installing, 46

services setup, 47–49  
SMTP access controls setup, 49  
starting, 50–51  
system aliases, creating, 49  
system requirements, 38  
System V-style init.d, populating, 46–47  
testing installation, 51  
ucspi-tcp, installing, 43  
unpacking distribution, 40  
users and groups, creating, 41–42  
verifying build environment, 38–39

installation step-by-step procedures, 52–87

building source, 55–66  
compile-time configuration settings, 57–58  
daemontools, installing, 64–66  
directories, creating, 59  
installing ucspi-tcp, 63–64  
qmail, installing, 61–63  
unpacking distributions, 55–57  
users and groups, creating, 59–61

configuring, 66–77  
boot script, 66–68  
SMTP access control, 76  
system aliases, 77  
system startup files, 68–76

installed mailers, stopping and disabling, 77–78  
product overview, 86–87  
starting qmail, 78–79  
testing installations, 79–86  
preparation for, 52–55

inst\_check script, 51

intd directories, splitting, 253

intd queue subdirectory contents, 18

Internet  
connectivity, and tuning networks, 283  
Web sites for information about, 446  
protocols, and RFCs, 446

- Internet Mail Access Protocol. *See* IMAP  
(Internet Mail Access Protocol)
- Internet Mail RFCs, 446–448
- Internet RFC-2822, Internet Message Format defined, 211
- J**
- Jones, Ken, and Vpopmail, 441
- junk mail, 285–294
- preventing, 286–287
  - system-level controls, 287–292
    - rlsmtpd, 289–292
    - using badmailfrom, 288–289
  - user-level controls, 292–294
    - address revocation and auditing, 294
    - filtering, 292–293
    - TMDA, 293
- K**
- keyword searching messages, 292, 293
- L**
- L-Soft's LISTSERV MLM, 324
- languages (programming)
  - Python
    - getmail and, 371
    - in Mailman, 315
  - SQL (Structured Query Language), 402–403
- languages (spoken), selection of in ezmlm-idx, 300
- LDAP, 401
- LDAP PAM, 402
- legality of logging messages, 210
- license, 19–20
- Lindberg, Fred, and ezmlm-idx, 298, 434
- link()
  - calls and installation and, 34
  - patch for, 254
- links, symbolic under /var/qmail, 33
- Linux, creating users and groups in, 59–60
- Listar MLM, 324
- list.cr.yp.to mailing lists, 23–25
- listings. *See* code listings
- LISTSERV MLM, 324
- local parts (usernames), setting in messages, 128
- local queue subdirectory contents, 18
- localhost or IP address 127.0.0.1, 217
- localhost, reference to local host, 217
- localiphost control file, 102
- locals control file, 102
- location
  - queue location requirements, 33–35
  - mailbox format selection and location, 29–32
- lock queue subdirectory contents, 18
- logs and logging
  - and SMTP activity gotcha, 466
  - installation and, 36
  - legality of logging messages, 210
  - log messages, 457–459
  - qmailctl script and, 165
  - testing after installation, 81
  - troubleshooting, 201–210
    - extended message logging, 209–210
    - multilog, 202
    - qmail-send log messages, 203–206
    - spllogger, 202–203
    - tcpserver log messages, 206–207
    - using recordio to log SMTP sessions, 207–208
    - using QUEUE\_EXTRA, 450
  - looping messages, 458

**M**

mail, *See also* delivery; junk mail; messages  
 configuring scalable servers and, 396–399  
 filtering, 144–148  
   extension addresses, 145–146  
 maildrop, 148  
 procmail, 146–148  
 mail problems  
   local users can't send mail, 223  
   mail not accepted from remote hosts, 219  
   mail not delivered to local user, 220–221  
   mail not delivered to remote address, 221–222  
   mail not retrievable by users, 222  
   retrieving, 369–374  
     Fetchmail, 369–370  
     getmail, 371–374  
 mail exchangers  
   backup mail exchanger, 226–228  
   MX, and performance, 6–7  
 Mail-Followup-To, 133  
 mail hubs, 233–235  
 mail servers, *See also specific servers*  
   functions of, 396  
 Mail User Agents (MUAs)  
   changing mailbox formats and, 243  
   explained, 2–3  
   listed, 2  
 mailbox servers, *See also* IMAP (Internet Mail Access Protocol) servers;  
   POP3 servers  
   basics, 235–236  
   changing mailbox formats and, 243  
 mailboxes  
   configuring boot script and, 66–68  
   configuring scalable servers and, 399–401  
   defined, 29

domain mailboxes  
   and Fetchmail, 370  
   and getmail, 373–374  
 format selection and location, 29–32  
 location and format basics, 239–245  
 migrating to maildir-format mailboxes, 243–245  
 moving to home directories, 241–242  
 service, 400–401  
 serving, 327–328, *See also* IMAP (Internet Mail Access Protocol); POP3 servers  
 utilities. *See* dot-qmail utilities  
 maildir mailbox format  
   delivery and, 138  
   installation and, 31–32  
   installing UW-IMAP with support of, 344–349  
   mailbox locations and, 32  
   migrating to maildir-format mailboxes, 243–245  
   reliability and, 7  
   support by Courier-IMAP, 350  
   support by qmail-pop3d, 328  
   support by SolidPOP, 338  
   vs. mbox mailbox format, 31  
 maildir2mbox utility, 152–153  
 maildirmake command, 31, 153  
 maildirsmtp command, 232  
 maildirwatch utility, 153  
 Maildrop, 148, 434  
 mailing list management, 295–326  
   ezmlm, 297–305  
     basics, 297  
     ezmlm-idx, 298  
     installing, 298–301  
     subscribing and unsubscribing, 304–305  
     testing, 301–302  
     using, 302–304  
 Mailman, 315–324  
   basics, 315–316

- mailing list management (*continued*)
  - creating mailing lists,
    - 320–324
  - installing, 316–320
  - subscribing, 324
- Majordomo, 305–315
  - basics, 305–306
  - creating lists with, 312–314
  - installing, 306–312
  - subscribing to Majordomo lists, 314–315
- other MLMs, 324–325
  - simple mailing lists, 295–297
- mailing list managers. *See* MLMs (mailing list managers)
- mailing lists
  - in ezmlm
    - creating, 302–304
    - subscribing and unsubscribing, 304–305
  - in Mailman
    - creating, 320–324
    - subscribing, 324
  - in Majordomo
    - creating, 312–314
    - subscribing, 314–315
  - list archives, 23
    - simple, 295–297
  - subscribing extension addresses to, 146
    - support for qmail, 23–25
    - support of, 14
  - Mailman, 315–324
    - basics, 315–316
    - creating mailing lists, 320–324
    - installing, 316–320
    - subscribing, 324
  - mailto utility, 154
  - Majordomo, 305–315
    - basics, 305–306
    - creating lists with, 312–314
    - installing, 306–312
    - subscribing to Majordomo lists, 314–315
- make check, 190–191
- make, failures in VMailMgr installation, 379
- man
  - man pages, 20–21
  - MANPATH, setting, 20
- man /var/qmail directory, 17
- management of qmail, *See also* queue management
  - management commands, 165–187
    - qmail-clean, 165
    - qmail-getpw, 166–167
    - qmail-local, 168–169
    - qmail-lspawn, 169
    - qmail-newmrh, 170
    - qmail-newu, 170
    - qmail-pop3d, 170–171
    - qmail-popup, 171–172
    - qmail-pw2u, 172–175
    - qmail-qmqpc, 175
    - qmail-qmqpd, 175
    - qmail-qmtpd, 176
    - qmail-qread, 176–177
    - qmail-qstat, 177
    - qmail-remote, 177–179
    - qmail-rspawn, 179
    - qmail-send, 179–180
    - qmail-showctl, 180–182
    - qmail-smtpd, 183
    - qmail-start, 184
    - qmail-tcpok, 184–185
    - qmail-tcptp, 185
    - splogger, 185–186
    - tcp-env, 186–187
  - qmailctl script, 159–165
    - interactive interface, 160–165
    - logging, 165
    - System V init.d script, 160
- mandatory spam controls defined, 285, 287
- Mastaler, Jason, and TMDA, 293, 440
- mbox mailbox format, 29–30
  - and mailbox locations, 32

configuring home directories  
     deliveries, 242  
 delivery and, 137  
 SolidPOP support of, 338  
     vs. maildirs, 31  
 me control file, 102–103  
 mess queue subdirectory contents, 18  
 mess822, 435  
 message construction features, 450  
 message headers, using to troubleshoot,  
     211–217  
     Cc fields, 215  
     Date fields, 214  
     Delivered-To field, 211–212  
     example header analysis,  
         216–217  
     From field, 215  
     Greenwich Mean Time (GMT) and,  
         213  
     Message-ID fields, 214  
     Received fields, 212–213  
     Resent- fields, 215  
     Return-Path field, 211  
     To fields, 215  
 Message-ID fields, and troubleshooting,  
     214  
 Message Transfer Agents. *See* MTAs  
     (Message Transfer Agents)  
 messages, *See also* receiving messages;  
     sending messages  
     adjusting message lifetimes/retry  
         schedules, 192  
     compatibility with Sendmail, 13  
     construction of, 13  
     error messages, 457–464  
         bounce messages, 459–464  
         log messages, 457–459  
     how qmail sends and receives,  
         417–418  
     how they travel, 443–444  
     logging, 209  
     retry schedule, 427–428  
     removing messages from queues,  
         193–195

mini-qmail, *See also* QMTP (Quick Mail Transfer Protocol)  
     defined, 258  
     nullmailer and, 264  
     support of clients, 230  
 MLMs (mailing list managers), *See also*  
     ezmlm; mailing list management; Mailman; Majordomo  
     basic functions of, 325  
     defined, 295  
     list maintenance function,  
         325  
     other Unix MLMs, 324–325  
     resending function, 325  
 moderated mailing lists defined, 2  
 modular system architecture  
     adding features and, 9  
     basics, 16  
     disadvantages of, 16  
     performance and, 7  
     security and, 6, 16  
 modules  
     listed, 16  
     receiving modules, 418–425  
         local, 419–420  
         qmail-queue, 423–425  
         remote, 421–423  
     sending local modules,  
         428–429  
 monolithic defined, 5  
 morerecphosts control file, 103  
 msg example file listing, 125–126, 135  
 MTAs (Message Transfer Agents)  
     comparison of various, 8, 10, *See also*  
         specific MTAs  
     defined, 1  
     interactions with MUAs, 3  
     monolithic, and security, 5  
 MUAs (Mail User Agents)  
     changing mailbox formats and, 243  
     explained, 2–3  
     listed, 2  
 Müller, Olivier, oMail-webmail and  
     oSpam and, 435

- multilog  
    advantages of, 36  
    troubleshooting, 202  
    while using recordio, 208
- multiple host names, configuring, 115
- multiple-recipients  
    delivery to. *See delivery*  
    multiple RCPT commands, 391–394
- MX (mail exchanger), and performance, 6–7
- MX records  
    backup mail exchanger and, 227–228  
    directing mail to virtual domains and, 380, 386  
    distributing incoming load with, 397–398  
    for smart hosts, 234
- null client and, 229–230
- nullmailer, 264
- N**
- name-address format, 131
- Nelson, Russell, and qmail-qsanity  
    script, 191
- network services, and installation, 35–36
- network tuning, 283–284
- null client, 228–230
- nullmailer, 264
- O**
- oMail-webmail, 435
- open mailing lists, defined, 2
- openssl command, and makefile, 267
- operating systems, *See also* system-level  
    controls for junk mail  
    creating users and groups in various, 59–61  
    installation requirements, 38, 52–53  
    selecting, 279  
    tarballs and operating system-specific  
        packages, 28–29
- options, *See also* configuring qmail,  
    advanced options  
    -i option, and QMAILINJECT, 130  
    -t option, and sendmail injection, 124  
    finding configurable options for  
        SolidPOP, 339  
    in ezmlm, 303  
    local delivery options, 15  
    specifying for qmail-inject injection, 130–132
- oSpam, 435
- P**
- packages related to qmail,  
    431–441  
    checkpassword, 431  
    Courier-IMAP, 432  
    daemontools, 432  
    djbdns, 432–433  
    dot-forward, 433  
    exmlm, 433  
    exmlm-idx, 433–434  
    fastforward, 434  
    getmail, 434  
    maildrop, 434  
    mess822, 435  
    oMail-webmail, 435  
    oSpam, 435  
    qlogtools, 436  
    qmail-autoreponder, 436  
    qmail-qfilter, 436  
    Qmail-Scanner, 437  
    qmail-vacation, 437  
    qmailanalog, 437–439  
    safecat, 439  
    serialmail, 439–440  
    SqWebMail, 440  
    syndir, 440  
    TMDA, 440  
    ucspi-tcp, 441  
    VMailMgr, 441  
    Vpopmail, 441

- Pape, Gerrit, and man pages, 432, 441
- PAM, 402
- parsing, and mail security, 6
- patches
- big-todo patch, 277
  - ezmlm-idx, 299
  - STARTTLS, 265–269
  - frequently used, 251–254
  - installing, 249–251
- pause function in qmailctl, 163
- percenthack control file, 103–104
- performance
- MX (mail exchanger) and, 6–7
  - tuning, 273–284
    - determining problems, 274
    - network, 283–284
    - qmail, 275–278
    - system hardware, 280–283
    - system software, 279–280
- personal names (comments), setting in messages, 129
- PGP (Pretty Good Privacy), 264
- PIC files, 19
- pictures, and PIC files, 19
- pid queue subdirectory contents, 18
- pinq utility, 154
- plusdomain control file, 104
- POP (Post Office Protocol)
- basic description, 327
  - factors in selecting, 327–328
- POP3 servers
- distributed service, 401
  - features of, 455
  - qmail-pop3d
    - architecture of, 329
    - installing, 329–332
    - testing, 332–333
    - using, 328
- Qpopper, 333–338
- installing, 334–337
  - testing, 337–338
- retrieving mail, 369–374
- Fetchmail, 369–370
- getmail, 371–374
- securing, 353–362
- APOP authentication, 358–362
  - wrapping with SSL, 354–358
- services, 15
- SolidPOP, 338–343
- installing, 339–342
  - service, 343
- Post Office Protocol. *See* POP (Post Office Protocol)
- Postfix, 11
- predate utility, 154–155
- preference, and MX records, 226
- preline dot-qmail utility, 155
- preprocessing, 425–426
- Pretty Good Privacy (PGP), 264
- proactive spam controls defined, 285, 287
- problems. *See also* fixing common problems; troubleshooting
- processes, troubleshooting, 200–201
- Procmail, 146–148
- programs
- program delivery instructions, 137
  - untrusting, and qmail security, 5
- protocols, and RFCs, 446
- proxy-wrapping, 364–367
- Python
- getmail and, 371
  - in Mailman, 315

## Q

- qail utility, 155
- qlogtools, 436
- qmail, *See also* installation; installation, step-by-step procedures; management of qmail
- basics, 2–3, 25
  - bounces, 452
  - building during installation, 61–63
  - disadvantages of, 9

- qmail (continued)*
  - features, 12–15, 449–455
    - message construction, 450
    - POP3 service, 455
    - queue management, 451–452
    - routing by domain, 453
    - security, 449–450
    - setup, 449
    - SMTP delivery, 453
    - SMTP service, 450–451
  - forwarding and mailing lists, 454
  - history of, 1, 15
  - local delivery, 454–455
    - vs. Sendmail, 2
- qmail-autoreader*, 436
- qmail-clean*
  - described, 16
  - management command, 165
- qmail-command* environment variables
  - listed, 143
  - settings listed, 144
- qmail-getpw*, 166–167
- qmail*, how it works, 412–430
  - high-level overview, 417–418
  - receiving modules, 418–425
    - local, 419–420
    - qmail-queue*, 423–425
    - remote, 421–423
  - sending modules, 425–430
    - local, 428–429
    - qmail-send*, 425–428
    - remote, 429–430
- qmail-inject*
  - local receiving modules and, 419
  - VERP and, 395
- qmail-inject* injection, 126–133
  - envelope sender address, setting, 129–130
  - environment variables, 127–133
  - From field, setting, 127–129
  - injection described, 16
  - Mail-Followup-To, setting, 133
  - overriding control files, 130
- specifying options, 130–132
- qmail-ldap*, 402
- qmail-local*
  - basics, 429
  - described, 168–169
  - error messages, 458–459,
  - module, 16
- qmail-lspawn*, 16, 169, 429
- qmail-newu*, 170
- qmail-pop3d*, *See also POP3 servers*
  - described, 170–171
  - installing, 329–332
  - using, 328
- qmail-popup*, 171
- qmail-procmail* listing, 147
- qmail-pw2u*
  - adding users and, 119
  - described, 172–175
- qmail-qfilter*, 436
- qmail-qmqpd*, 175, 423
- qmail-qmtpd*, 176, 423
- qmail-qread* queue diagnostic tool, 176–177, 190
- qmail-qsanity*, 191
- qmail-qstat* queue diagnostic tool, 188–189
- qmail-queue*
  - described, 16
  - injection, and sending messages, 134–136
  - receiving modules and, 423–425
- qmail-read* queue diagnostic tool, 188–189
- qmail-remote*
  - basics, 177–179, 429–430
  - error messages, 459
- qmail-rspawn*, 16, 179, 429
- Qmail-Scanner*, 404–414, 437
  - installing, 404–414
  - maintaining, 414
- qmail-send*
  - basics, 16, 179–180, 425–428
  - error messages, 457–458
  - log messages, 203–206

module, 16  
 removing messages from queue  
     and, 193  
`qmail-showctl`, 180–182  
`qmail-smtpd`  
     described, 16, 183  
     gotchas, 467–468  
     remote receiving module, 421–423  
`qmail-start`, 184  
`qmail-tcpok`, 184–185  
`qmail-tcpt0`, 185  
`qmail-users` mechanism, configuring,  
     117–119  
`qmail-vacation`, 437  
`qmail@list.cr.yp.to` mailing list, 24  
`qmail+patches` installation, vs. tarball  
     installation, 90  
`qmailanalog`, 437–439  
`qmailannounce@list.cr.yp.to` mailing  
     list, 24  
`qmailctl` functions  
     `alarm` function, 162  
`qmailctl` functions (*continued*)  
     `cdb` function, 164  
     `cont` function, 164  
     `doqueue` function, 162  
     `flush` function, 162  
     `help` function, 165  
     `hup` function, 163  
     `pause` function, 163  
     `queue` function, 164  
     `reload` function, 163  
     `restart` function, 162  
     script functions, 161–165  
     `start` function, 161  
     `stat` function, 163, 189, 201  
`qmailctl` script  
     installing, 46  
     system startup files configuration and,  
         68–73  
     troubleshooting processes and, 201  
     using, 159–165  
         interactive interface, 160–165  
         logging, 165

System V init.d script, 160  
`QMAILHOST`, and setting envelope  
     sender addresses, 130  
`QMAILINJECT`, and overriding control  
     files, 130–132  
`QMAILQUEUE` patch, 405, 412  
`QMAILUSER`, and setting envelope  
     sender addresses, 129  
`QMQP` (Quick Mail Queuing Protocol)  
     advanced configuration, 258–264  
     `nullmailer` and, 264  
     setting up QMQP clients (mini-qmail),  
         261–264  
     setting up QMQP services, 259–261  
`qmqpserver`s control file, 104  
`QMTP` (Quick Mail Transfer Protocol),  
     254, 255–258  
     advanced options, 254–258  
     basics, 255–258  
     patch, 253  
     setting up QMTP services,  
         254–255  
     support for, 253  
     vs. SMTP, 256–258  
`Qpopper`, 333–338  
     installing, 334–337  
     testing service, 337–338  
`receipt dot-qmail` utility,  
     155–156  
`QSBMF` error messages, 460–461  
`queue`, *See also* queue management  
     basics of structure, 17–18  
     cleanup, 428  
     how messages are placed in,  
         424–425  
     location requirements, 33–35  
     preprocessing, 425–426  
`queue /var/qmail` directory, 17  
`queue-fix`, 196–197  
`queue` function in `qmailctl`, 164  
`queue` management, 187–197  
     checking, 188–191  
         `make check`, 190–191  
     `qmail-qread`, 190

- queue management (*continued*)  
    qmail-qsanity, 191  
    qmail-qstat, 188–189  
features of, 451–452  
introduction to, 13  
modifying, 191–197  
    adjusting message lifetimes/retry schedules, 192  
    making corrupt queues consistent, 196  
    recreating empty queues, 196–197  
    removing messages from queues, 193–195  
    tuning, 275–278  
queuelifetime control file, 104–105  
QUEUESMTPEXTRA compile-time configuration variable, 209  
Quick Mail Transfer Protocol. *See* QMTP  
    (Quick Mail Transfer Protocol)  
quick-start installation. *See* installation
- R**
- RAID (Redundant Arrays of Inexpensive Disk), and tuning system hardware, 282–283  
RAM (Random-Access Memory)  
    swap space memory vs., 375  
    tuning, 280–281  
rlsmtpd, for junk mail control, 289–292  
rcphosts control file, 105–106  
reactive spam controls defined, 285, 287  
Received header fields, and troubleshooting, 212–213  
receiving messages, 136–148  
    dot-qmail files, 136  
    extension addresses, 141–142  
    filtering mail, 144–148  
        extension addresses, 145–146  
        maildrop, 148  
        procmail, 146–148  
    forward delivery, 138–139  
    maildir delivery, 138
- mailbox delivery, 137  
multiple deliveries, 139–141  
program delivery, 137  
qmail-command environment variables, 142–144  
receiving modules, 418–425  
    local, 419–420  
    qmail-queue, 423–425  
    remote, 421–423  
recipients  
    qmail-inject and, 419–420  
    Sendmail vs. qmail and, 124  
recordio, to log SMTP sessions, 207–208  
Red Hat Linux 7.1, installing from RPMs and, 87–90  
Red Hat Package Manager packages  
    (RPMs), installing from, 87–90  
    assumptions about, 88  
    cautions, 90  
    downloading RPMs, 88  
    installing RPMs, 88–90  
    selecting RPMs, 87–88  
Redundant Arrays of Inexpensive Disk technology (RAID), and tuning system hardware, 282–283  
RELAYCLIENT, 76, 113, 422  
relaying, 110–115  
    allowing selective, 112–115  
    control of, 111–112  
    defined, 110  
    disabling, 112  
relays, 233–235  
    relay-after-IMAP, 114  
    relay-after-POP, 114  
    relays defined, 76  
reliability of qmail, 7  
reload function in qmailctl, 163  
remote message retry schedule, 427–428  
remote sending modules, 429–430  
remote queue subdirectory contents, 18  
removing messages from queue, 193–195  
report codes, qmail-remote listed, 178  
Resent- fields, and troubleshooting, 215  
restart function in qmailctl, 162

- retry schedules, 426–428
  - Return-Path header field, and troubleshooting, 211
  - RFC 1893 status codes, 461–464
  - RFCs, 446–448
  - Ringel, Fred B., and ezmlm-idx, 298, 434
  - root code
    - minimization of, 449
    - security and, 5
  - routing
    - by domain features, 453
    - domains and, 14
    - reliability and, 7
  - RPMs (Red Hat Package Manager packages), installing from, 87–90
    - assumptions about, 88
    - cautions, 90
    - downloading RPMs, 88
    - installing RPMs, 88–90
    - selecting RPMs, 87–88
- S**
- safecat, 439
  - Samuel, Peter, and qmail-vacation, 437
  - scalable servers, configuring, 395–401
    - incoming mail, 397–399
    - mailbox delivery, 399–400
    - mailbox service, 400–401
    - outgoing mail, 396–397
  - scripts
    - /var/qmail/rc script, 67
    - boot script
      - configuring at installation, 57–58
      - configuring for mailboxes, 66–68
      - setup, 45–46
    - qmailctl as an init.d script, 160
    - inst\_check script, 51
    - qmail-qsanity script, 191
    - qmailctl script, 69–71, 159–165
      - installing, 46
      - logs and logging and, 165
  - script functions, 161–165
  - startup files configuration script, 68–73
  - Secure Sockets Layer (SSL), and STARTTLS, 265
  - security, *See also* virus scanning
    - basic features, 12
    - code and, 6
    - features, 449–450
    - modular system architecture and, 6, 16
    - monolithic MTAs and, 5
    - MTAs, monolithic, and security, 5
    - of qmail, 3–6
    - parsing and, 6
    - Sendmail and, 3–5, 10–11
    - setuid() programs and, 4–5
    - SMTP security, and STARTTLS, 265–269
  - sending messages, 121–136
    - qmail-inject injection, 126–133
      - overriding control files, 130
      - setting envelope sender addresses, 129–130
      - setting From fields, 127–129
      - setting Mail-Followup-To, 133
      - specifying options, 130–132
    - qmail-queue injection, 134–136
    - Sendmail injection, 123–126
    - SMTP injection, 122–123
  - sending modules, 425–430
    - local, 428–429
    - qmail-send, 425–428
    - remote, 429–430
  - Sendmail
    - complexity of, 8–9
    - converting Sendmail-style aliases, 245–249
    - delivering mail, 240–241
    - dot-forward and, 433
    - history of, 1, 10–11
    - linking to rc directories and, 46–47
    - migration from, 237–249
    - multiple processes and, 7

- Sendmail (*continued*)
  - process IDs and, 78
  - security and, 3–5, 10–11
  - Sendmail injection, 123–126
  - vs. qmail, 2
- sendmail command, and local receiving modules, 419
- serialmail, 439–440
- serialmail@list.cr.yp.to mailing list, 25
- services
  - configuring system startup files, 73–76
  - setting up during quick-start, 47–49
  - stopping all, 51, 79
- setlock command, 232
- setuid() programs
  - minimization of with qmail, 449
  - security and, 4–5
- setuidgid, 37
- setup features, 12, 449
- shells, in MANPATH, 20
- signatures defined, 403
- simple assignments in qmail users, 117–118
- Simple Mail Transfer Protocol. *See* SMTP
- single-drive performance, 281
- single-recipient delivery. *See* delivery
- size of qmail vs. other MTAs, 8
- smart hosts, 233–235
- SMTP
  - activity logging gotcha, 466
  - authenticated, 114–115
  - delivery, 14
  - delivery features, 453
  - finding information about, 446
  - injection, 122–123
  - limitations in mail retrieving, 327
  - service features, 450–451
  - services, 13
  - setting up access control, 49, 76
  - SSL-wrapped, 269–273
    - installing Stunnel, 269–271
    - setting up services, 271–273
  - talking to local systems via, 183
- transparently distributed service, 398–399
- troubleshooting SMTP sessions, 207–208
- vs. QMTP, 256–257
- smtpgreeting control file, 106
- smtproutes control file, 107
- soft errors, and delivery, 140
- softlimit, 37
- software. *See* system software
- Solaris
  - creating run files, 259
  - id programs and services setup, 48
  - creating users and groups in, 59–60
- SolidPOP, 338–343
  - installing, 339–342
  - testing service, 343
- source-code, *See also* building source
  - downloading for installation, 54–55
  - installing from, 28–29
  - modifying, 249–254
  - frequently used patches, 251–254
  - installing patches, 250–251
- source files. *See* files
- spam, *See also* junk mail
  - defined, 285
  - mandatory control of, 285
- SpamBouncer, 293
- split, defined, 424
- splogger
  - described, 185–186
  - understanding logs and, 202–203
- spoofing, defined, 112
- SQL (Structured Query Language), 402–403
- SqWebMail, 440
- SSL (Secure Sockets Layer), and STARTTLS, 265
- SSL-wrapping
  - IMAP, 362–367
  - enabling SSL wrapper, 363–364
  - proxy-wrapping, 364–367
  - RAM-MD5 authentication, 367–368

- wrapping with SSL, 362–367
- POP3, 354–358
- SMTP, 269–273
  - installing Stunnel, 269–271
  - setting up SSL-wrapped services, 271–273
- SSLWrap, and POP3, 354
- standard C library defined, 6
- start function in qmailctl, 161
- STARTTLS, installing patch for SMTP security, 265–269
- stat function in qmailctl, 201
- stop function in qmailctl, 162
- Structured Query Language (SQL), 402–403
- Stunnel**
  - installing, 269–271
  - POP3 and, 354, 355
- subdirectories
  - queue, 18
  - split, 253
- subscribing to mailing lists, 304–305
- superusers
  - defined, 5
  - gotcha, 465
  - procedure to become, 55
  - qmail-qread and superuser privileges, 125
- supervise service monitor, 37
- support for qmail, 23–25
- svc service control program, 37
- svscan
  - described, 37
  - during daemontools installation, 65
- svstat, and service status display, 37
- swap space defined, 375
- symbolic links, under /var/qmail, 33
- syncdir, 440
- Syslog
  - advantages and disadvantages of, 36
  - troubleshooting splogger and, 202–203
- system accounts, and compile-time settings, 58
- system hardware, tuning, 280–283
- system-level controls for junk mail, 287–292
  - rlbsmtpd, 289–292
  - using badmailfrom, 288–289
- system software, tuning, 279–280
- System V-based systems
  - /var/qmail/bin/qmailctl, and system boots, 72–73
  - init.d script
    - populating, 46–47
    - qmailctl script and, 160
- systems. *See* operating systems; system-level controls for junk mail

## T

- tagged addresses, 294
- Tagged Message Delivery Agent (TMDA), 293, 440
- tai64nlocal, 37
- tarballs
  - copying or moving to directories, 40
  - installing dot-forward packages and, 237, 238
  - installing from, 28–29
  - locating source tarballs, 39–40
  - unpacking during installation, 40, 55–57
  - vs. installing with qmail+patches, 90
- tarpitting, 288
- tcp-env management command, 186–187
- tcpserver logs
  - running of qmail-pop3d and, 329
  - troubleshooting, 206–207
  - using recordio and, 208
- teergrubing, 288
- telnet command
  - initiating SMTP sessions and, 217–218
- test messages
  - group membership tests, 84
  - invalid local user to invalid local user, 84

- test messages (*continued*)  
  local user to local postmaster, 83  
  local user to local user, 81–82  
  local user to nonexistent local  
    address, 82–83  
  local user to valid remote address, 83  
  MUA, 86  
  remote user to invalid local user, 86  
  remote user to local user, 86  
  SMTP server tests, 85
- testing  
  checkpassword, 431  
  Courier-IMAP, 352–353  
  EICAR virus test file, 413–414  
  ezmlm, 301–302  
  qmail-pop3d service,  
    332–333  
  Qpopper service, 337–338  
  quick-start installation, 51  
  rlsmtpd, 290–292  
  SolidPOP service, 343  
  step-by-step installation,  
    79–86  
    daemons, 79–80  
    logs, 81  
  troubleshooting and, 217–218  
  UW-IMAP service, 349–350  
  VMailMgr, 382–383  
  Vpopmail, 388–389
- timeoutconnect control file, 108  
timeoutremote control file, 108  
timeoutsmtpd control file, 108–109
- TMDA (Tagged Message Delivery Agent)  
  basics, 440  
  for junk mail control, 293
- To fields, and troubleshooting, 215
- todo directories, 253
- todo queue subdirectory contents, 18
- trailing slashes (/), and configuration, 242
- troubleshooting, 199–223, *See also* error  
  messages  
    first attempt to send messages, 114  
    fixing common problems  
      local users can't send mail, 223
- mail not accepted from remote  
  hosts, 219
- mail not delivered to local user,  
  220–221
- mail not delivered to remote  
  address, 221–222
- mail not retrievable by users, 222
- information about, 23
- logs, 201–210  
  extended message logging,  
    209–210  
  multilog, 202  
  qmail-send log messages, 203–206  
  spllogger, 202–203  
  tcpserver log messages, 206–207  
  using recordio to log SMTP  
    sessions, 207–208
- multiple deliveries, 139–140
- process monitoring, 200–201
- testing, 217–218
- using message headers, 211–217  
  Cc fields, 215  
  Date fields, 214  
  Delivered-To field, 211–212  
  example header analysis, 216–217  
  From field, 215  
  Message-ID fields, 214  
  Received fields, 212–213  
  Resent- fields, 215  
  Return-Path field, 211  
  To fields, 215
- U**
- UBE (Unsolicited Bulk E-mail). *See* junk  
  mail
- UCE (Unsolicited Commercial E-mail).  
  *See* junk mail
- ucspi-tcp  
  described, 36, 441  
  installing, 43, 63–64
- Uh-oh: home directory is writable, error  
  message, 458–459

- Uh-oh: qmail file is writable, error message, 458–459
- umask command, 56
- uncoupling dot-qmail deliveries, 140
- University of Washington IMAP (UW-IMAP), 344–350
  - installing with maildir support, 344–349
  - testing, 349–350
- Unix Client-Server Program Interface for TCP. *See* ucspi-tcp
- Unix MLMs, 324–325, *See also* ezmlm; Mailman; Majordomo
- Unsolicited Bulk E-mail (UBE). *See* junk mail
- Unsolicited Commercial E-mail (UCE). *See* junk mail
- unsubscribing from mailing lists, 304–305
- user-level controls, for junk mail, 292–294
  - address revocation and auditing, 294
  - filtering, 292–293
  - TMDA, 293
- usernames
  - setting usernames (local parts) in messages, 128
  - containing at-sign (@) and POP3, 383
  - with upper case letters gotcha, 465
- users
  - creating at installation
    - with quick-start, 41–42
    - with step-by-step installation, 59–62
  - home directories gotcha, 465
  - local users can't send mail, 223
  - prevention of junk mail and, 286–287
  - test messages for local users, 81–84
  - usernames with upper case letters gotcha, 465
- users /var/qmail directory, 17
- utilities
  - user utilities, 148–156
    - bouncesaying, 148–149
- condredirect, 150
- datemail, 150–151
- elq, 151
- except, 151
- forward, 152
- maildir2 mbox,
- 152–153
- maildirmake, 153
- maildirwatch, 153
- mailsubj, 154
- ping, 154
- predate, 154–155
- preline, 155
- qail, 155
- qreceipt, 155–156
- support utilities and installation, 35–38
- VMailMgr utilities, 377
- UW-IMAP, 344–350
  - installing with maildir support, 344–349
  - testing, 349–350

## V

- Variable Envelope Return Paths. *See* VERP (Variable Envelope Return Path)
- Varshavchik, Sam
  - Courier and, 11, 432
  - maildrop and, 434
  - SqWebMail and, 440
- Venema, Wietse
  - Postfix and, 11
  - tcp\_wrappers utility and, 36
- VERP (Variable Envelope Return Paths)
  - ezmlm and, 297, 394–395
  - handling bounces, 395
  - qmail-inject and, 395
- vipw, and creating users and groups, 60
- virtual, defined, 375

- virtual domains, *See also* VMailMgr;
  - Vpopmail
    - configuring, 115–116
    - creating mailing lists in, 303–304
    - described, 375
    - importance of, 375
    - qmail and, 376
  - virtual users
    - definitions of, 375
    - qmail and, 376
  - virtualdomains control file, 109–110
  - virus scanning, 403–414
    - AMaViS, 404
    - Qmail-Scanner, 404–414
    - signatures and, 403
  - VMailMgr, 377–383
    - basics, 441
    - configuring, 380–381
    - core utilities, 377
    - installing, 377–379
- testing, 382–383
- vs. Vpopmail, 376–377
- Vpopmail, 383–389
  - basics, 441
  - configuring, 386–387
  - installing, 383–386
  - testing, 388–389
  - vs. VMailMgr, 376–377
- W**
  - whitelisting, 292, 293
- wildcard assignments, in qmail users, 118
- wrapping. *See* SSL-wrapping
- write caching, 35
- X**
  - xinetd, and running qmail-pop3d, 329