Life with qmail

Table of Contents

1. Introduction **1**
- 1.1. Audience 1
- 1.2. What is qmail? 1
- 1.3. Why use qmail? 1
- 1.4. History 2
- 1.5. Features 2
- 1.6. Related packages 4
- 1.7. Architecture 5
- 1.8. License 5
- 1.9. Comparison with other MTA's 5
- 1.10. Documentation 5
- 1.11. Support 7

2. Installation **11**
- 2.1. Installation Issues 11
- 2.2. Preparation 12
- 2.3. System requirements 12
- 2.4. Download the source 13
- 2.5. Build the source 13
- 2.6. Install ucspi-tcp 17
- 2.7. Install daemontools 17
- 2.8. Start qmail 18
- 2.9. Test the Installation 26

3. Configuration **29**
- 3.1. Configuration Files 29
- 3.2. Relaying 30
- 3.3. Multiple host names 31
- 3.4. Virtual domains 32
- 3.5. Aliases 32
- 3.6. qmail-users 33
- 3.7. Spam Control 34
- 3.8. Virus Scanning 34

4. Usage **35**
- 4.1. .qmail files 35
- 4.2. Sending messages 37
- 4.3. Environment Variables 38

5. Advanced Topics **41**
- 5.1. procmail 41
- 5.2. POP and IMAP servers 41
- 5.3. POP and IMAP clients 44
- 5.4. Multi-RCPT vs. Single RCPT delivery 45
- 5.5. VERP 46
- 5.6. Troubleshooting 46
- 5.7. Big Servers 49
- 5.8. Migrating from Sendmail to qmail 49
- 5.9. Mailing List Managers 49
- 5.10. Patches 50

Table of Contents

5. Advanced Topics
5.11. QMTP.....52
5.12. Rejecting Invalid Recipients During SMTP Dialogue.....52
5.13. TLS and STARTTLS.....52

A. Acknowledgments.....53

B. Related Packages.....55
B.1. dot-forward.....55
B.2. fastforward.....55
B.3. ucspi-tcp.....55
B.4. daemontools.....56
B.5. qmailanalog.....56
B.6. rblsmtpd.....56
B.7. serialmail.....57
B.8. mess822.....58
B.9. ezmlm.....58
B.10. safecat.....58
B.11. djbdns.....58
B.12. maildrop.....59
B.13. syncdir.....59

C. How Internet Mail Works.....61
C.1. How a message gets from point A to point B.....61
C.2. More information.....62

D. Architecture.....65
D.1. Modular system architecture.....65
D.2. File structure.....65
D.3. Queue structure.....66
D.4. Pictures.....66

E. Infrequently Asked Questions.....69
E.1. How frequently does qmail try to send deferred messages?.....69
E.2. Why can't I send mail to a large site with lots of MX's?.....70
E.3. What is QUEUE_EXTRA?.....70

F. Error Messages.....73

G. Gotchas.....75
G.1. qmail doesn't deliver mail to superusers......75
G.2. qmail doesn't deliver mail to users who don't own their home directory......75
G.3. qmail doesn't deliver mail to users whose usernames contain uppercase letters......75
G.4. qmail replaces dots (.) in extension addresses with colons (:)......75
G.5. qmail converts uppercase characters in extension addresses to lowercase......75
G.6. qmail doesn't use /etc/hosts......75
G.7. qmail doesn't log SMTP activity......76
G.8. qmail doesn't generate deferral notices......76
G.9. qmail is slow if /var/qmail/queue/lock/trigger is gone/has the wrong permissions/is a regular file......76

Table of Contents

G. Gotchas
G.10. DNS or IDENT lookups can make SMTP slow. 76
G.11. Carriage Return/Linefeed (CRLF) line breaks don't work. 76
G.12. qmail-send or tcpserver stop working if logs back up. 77
G.13. qmail-smtpd doesn't validate the local part of an address. 77
G.14. Firewalls can block remote access to your SMTP/POP3/IMAP server. 77
G.15. qmail-inject sets From field to anonymous if USER and LOGNAME aren't set. 77
G.16. qmail-send doesn't always exit immediately when killed. 78
G.17. Delivering to /dev/null doesn't throw messages away. 78
G.18. Modifying the queue while qmail-send is running is dangerous. 78

H. Frequently Asked Questions about Life with qmail 79
H.1. What version is Life with qmail? 79
H.2. Who owns Life with qmail? 79
H.3. How is Life with qmail licensed? 79
H.4. How can I be notified when new releases of LWQ are made available? 79
H.5. Where can LWQ contributors and fans talk about it? 79
H.6. Has Life with qmail been translated to language? 79
H.7. Is Life with qmail available in PostScript, PDF, plain text, or any other format beside HTML? 79
H.8. I used Life with qmail and it crashed my system/erased my hard disk/ruined my love life/killed my dog/etc. 79
H.9. How can I contribute to LWQ? 80
H.10. What's changed in this version of LWQ? 80

1. Introduction

1.1. Audience

Life with qmail is aimed at everyone interested in running *qmail*, from the rank amateur (*newbie*) who just installed Linux on a spare PC all the way up to the experienced system administrator or mail administrator. If you find it lacking or unclear, please let me know. Send comments to lwq@sill.org.

There's a wealth of information available on *qmail* from a variety of sources. Some is targeted to newbies, some assumes that the reader is more experienced. Life with qmail is an attempt to "glue" this information into a single source, filling in some of the cracks and assuming only that the reader has basic skills such as:

- Manipulating files/directories under UNIX
- Operating a web browser or FTP client
- Following directions

1.2. What is *qmail*?

qmail is an Internet Mail Transfer Agent (MTA) for UNIX-like operating systems. It's a drop-in replacement for the *Sendmail* system provided with UNIX operating systems. *qmail* uses the Simple Mail Transfer Protocol (SMTP) to exchange messages with MTA's on other systems.

Note: The name is "qmail", not "Qmail".

1.3. Why use *qmail*?

Your operating system included an MTA, probably *Postfix* or *Sendmail*, so if you're reading this document you're probably looking for something different. Some of the advantages of *qmail* over vendor-provided MTA's include:

1.3.1. Security

qmail was designed for high security. *Sendmail* has a long history of serious security problems. When *Sendmail* was written, the Net was a much friendlier place. Everyone knew everyone else, and there was little need to design and code for high security. Today's Internet is a much more hostile environment for network servers. *Sendmail*'s author, Eric Allman, and the current maintainer, Claus Assman, have done a good job of tightening up the program, but nothing short of a redesign can achieve *true* security.

1.3.2. Performance

qmail parallelizes mail delivery, performing up to 20 deliveries simultaneously, by default.

1.3.3. Reliability

Once *qmail* accepts a message, it guarantees that it won't be lost. *qmail* also supports a new mailbox format that works reliably *even over NFS* without locking.

1.3.4. Simplicity

qmail is smaller than any other equivalently-featured MTA.

Note: The official *qmail* web page, http://cr.yp.to/qmail.html covers the advantages of *qmail* more extensively.

1.4. History

qmail was written by Dan Bernstein (DJB), http://en.wikipedia.org/wiki/Daniel_J._Bernstein, a math professor now at the University of Illinois in Chicago. Dr. Bernstein is also well known for his work in the field of cryptography and for his lawsuit against the U.S. government regarding the publishing of encryption source code. See http://en.wikipedia.org/wiki/Bernstein_v._United_States or http://cr.yp.to/export.html for information regarding the lawsuit.

The first public release of *qmail*, beta version 0.70, occurred on January, 24, 1996. The first gamma release, 0.90, was on August, 1, 1996.

Version 1.0, the first general release, was announced on February, 20, 1997. The current version, 1.03, was released on June, 15, 1998.

The next release is expected to be an evaluation version of 2.0. Some of things that might appear in version 2 are covered at http://cr.yp.to/qmail/future.html.

1.5. Features

The *qmail* web page, http://cr.yp.to/qmail.html, has a comprehensive list of *qmail*'s features. This section is based heavily on that list.

1.5.1. Setup

- Automatic adaptation to your UNIX variant--no porting needed
- Automatic per-host configuration
- Quick installation--no big list of decisions to make

1.5.2. Security

- Clear separation between addresses, files, and programs
- Minimization of setuid code
- Minimization of root code
- Five-way trust partitioning--security in depth
- Optional logging of one-way message hashes, entire message contents, etc. (See What is QUEUE_EXTRA? in Appendix E.)

1.5.3. Message construction

- RFC 822 and RFC 1123 compliant
- Full support for address groups
- Automatic conversion of old-style address lists to RFC 822 format
- `sendmail` command for compatibility with current user agents

- Header line length limited only by memory
- Host masquerading (See defaulthost)
- User masquerading (See MAILUSER and MAILHOST)
- Automatic Mail-Followup-To creation (See QMAILMFTFILE)

1.5.4. SMTP service

- RFC 821, RFC 1123, RFC 1651, RFC 1652, and RFC 1854 compliant
- 8-bit clean
- RFC 931/1413/ident/TAP callback--can help track spammers/forgers
- Relay control--stops unauthorized relaying by outsiders
- No interference between relay control and aliases
- Automatic recognition of local IP addresses
- Per-buffer timeouts
- Hop counting
- Parallelism limit (via ucspi-tcp)
- Refusal of connections from known abusers (via ucspi-tcp)
- Relaying and message rewriting for authorized clients
- Optional RBL/ORBS support (via rblsmtpd)

1.5.5. Queue management

- Instant handling of messages added to queue
- Parallelism limits
- Split queue directory--no slowdown when queue gets big
- Quadratic retry schedule--old messages tried less often (see Appendix E)
- Independent message retry schedules
- Automatic safe queueing--no loss of mail if system crashes
- Automatic per-recipient checkpointing
- Automatic queue cleanups
- Queue viewing (See `qmail-qread`)
- Detailed delivery statistics (via qmailanalog)

1.5.6. Bounces

- QSBMF bounce messages--both machine-readable and human-readable
- HCMSSC support--language-independent RFC 1893 error codes
- Double bounces sent to postmaster

1.5.7. Routing by domain

- Any number of names for local host (See locals)
- Any number of virtual domains (See virtualdomains)
- Domain wildcards (See virtualdomains)
- Configurable "percent hack" support (See percenthack)
- UUCP hook

1.5.8. SMTP delivery

- RFC 821, RFC 974, and RFC 1123 compliant
- 8-bit clean

* Automatic downed host backoffs
* Artificial routing--smarthost, localnet, mailertable (See smtproutes)
* per-buffer timeouts
* Passive SMTP queue--perfect for SLIP/PPP (via serialmail)
* AutoTURN support (via serialmail)

1.5.9. Forwarding and mailing lists

* *Sendmail* `.forward` compatibility (via dot-forward)
* Hashed forwarding databases (via fastforward)
* *Sendmail* `/etc/aliases` compatibility (via fastforward)
* Address wildcards (See .qmail-default)
* Mailing list owners--automatically divert bounces and vacation messages
* VERPs--automatic recipient identification for mailing list bounces
* Delivered-To--automatic loop prevention, even across hosts

1.5.10. Local delivery

* User-controlled address hierarchy--fred controls fred-anything mbox delivery
* Reliable NFS delivery (See maildir)
* User-controlled program delivery: procmail etc. (See qmail-command)
* Optional new-mail notification (See qbiff)
* Optional NRUDT return receipts (See qreceipt)
* Conditional filtering (See condredirect and bouncesaying)

1.5.11. POP3 service

* RFC 1939 compliant
* UIDL support
* TOP support
* APOP hook
* modular password checking (via checkpassword)

1.6. Related packages

qmail follows the classic UNIX philosophy that each tool should perform a single, well-defined function, and complex functions should be built by connecting a series of simple tools into a "pipeline". The alternative is to build more and more complex tools that re-invent much of the functionality of the simpler tools.

It's not surprising, then, that *qmail* itself doesn't do everything everyone might want it to do. Here, then, are some of the most popular add-ons written for *qmail*. Of course, many standard UNIX utilities can also be plugged into *qmail*.

* dot-forward--a *Sendmail* `.forward` file compatibility add-on
* fastforward--a *Sendmail* alias database compatibility add-on
* ucspi-tcp--an *inetd* replacement
* daemontools--a set of tools for managing daemons and their logs
* qmailanalog--a set of qmail log file analysis tools
* serialmail--tools for mailing over slow networks
* mess822--tools for parsing Internet mail messages

- ezmlm--a mailing list manager for *qmail*

1.7. Architecture

Appendix D covers *qmail*'s functional and physical structure. In a nutshell, *qmail* consists of a series of programs (modules) that perform different tasks.

1.8. License

As of 2007-11-30, *qmail* 1.03 is in the public domain. See http://cr.yp.to/qmail/dist.html. This means that there are no legal limits to what you can do with it: you can copy it, give it away, sell it, modify it, rename it, or use pieces of it in copy-protected works, without any restrictions.

Other packages by Dan Bernstein, such as *daemontools* and *ucspi-tcp*, are copyrighted by the author, and are not distributed with a statement of user's rights. In http://cr.yp.to/softwarelaw.html, he outlines what he thinks your rights are under U.S. copyright law. See also http://en.wikipedia.org/wiki/License-free_software.

1.9. Comparison with other MTA's

A book could be written about this topic, but it would be tedious reading. Here's a quick comparison of *qmail* with some of the most common UNIX MTA's.

MTA	Maturity	Security	Features	Performance	Sendmailish	Modular
qmail	medium	high	high	high	addons	yes
Sendmail	high	low	high	low	x	no
Postfix	medium	high	high	high	yes	yes
exim	medium	low	high	medium	yes	no
Courier	low	medium	high	medium	optional	yes

Sendmailish means the MTA behaves like *Sendmail* in some ways that would make a switch from *Sendmail* to the alternative MTA more user-transparent, such as the use of `.forward` files, `/etc/aliases`, and delivery to `/var/spool/mail`.

Jonathan de Boyne Pollard has reviews of many Unix MTAs at http://homepages.tesco.net/~J.deBoynePollard/Reviews/UnixMTSes/. Another detailed comparison is available at http://www.geocities.com/mailsoftware42/.

1.10. Documentation

1.10.1. man pages

The *qmail* distribution comes with a complete set of `man` pages. After installation, they're in `/var/qmail/man`. You'll probably need to add that directory to your `MANPATH` environment variable.

Shell	Command
Bourne (/bin/sh)	`MANPATH=$MANPATH:/var/qmail/man; export MANPATH`
bash, Korn	`export MANPATH=$MANPATH:/var/qmail/man`
C Shell	`setenv MANPATH $MANPATH:/var/qmail/man`

At this point, commands in the format "`man` *`name-of-qmail-man-page`*" should display the appropriate man page.

The `man` pages are also available on-line in HTML format from:

- http://www.qmail.org/man/index.html

Note: The *qmail* man pages are loaded with information, but they require careful reading because they're written in a very dense, technical style. You might want to print off a set and read them through once to familiarize yourself with what's there and where it is. Very little information is repeated on multiple pages, so if you don't know where something is covered, it can be hard to find it.

1.10.2. Docs

The *qmail* distribution includes a series of documents that are installed under `/var/qmail/doc`. They include:

- `FAQ`: Frequently Asked Questions, with answers
- `INSTALL*`: Installation documentation
- `PIC.*`: Descriptions of how *qmail* performs key tasks. See the Architecture appendix for more information.
- Various other installation-related documentation

These docs are also available on-line from:

- http://www.qmail.org/man/index.html

1.10.3. FAQs

There are two official FAQ (Frequently Asked Questions, with answers) documents:

- `/var/qmail/doc/FAQ`, the plain text version, and
- The web FAQ at http://cr.yp.to/qmail/faq.html.

The web FAQ is more complete.

1.10.4. Books

1.10.4.1. The qmail Handbook

Dave Sill, the author of Life with qmail, has written a *qmail* book for Apress (http://www.apress.com/). This book, The qmail Handbook, covers everything in this guide, but goes into much more detail and also covers a lot of new ground.

For more information, see http://www.apress.com/catalog/book/1893115402/. To order this book from my bookstore, in association with Amazon.com, see http://www.amazon.com/exec/obidos/ASIN/1893115402/davesill.

1.10.4.2. Qmail Quickstarter: Install, Set Up and Run your own Email Server

Kyle Wheeler has written a *qmail* book for Packt (http://www.packtpub.com/). As the title suggests, this book is designed to help people new to *qmail* to set up a mail server.

To order this book from my bookstore, in association with Amazon.com, see http://www.amazon.com/exec/obidos/ASIN/1847191150/davesill.

1.10.4.3. qmail

John Levine has written a *qmail* book for O'Reilly & Associates (http://www.oreilly.com/). See http://qmail.gurus.com/ for more info including the Table of Contents and a sample chapter.

To order this book from my bookstore, in association with Amazon.com, see http://www.amazon.com/exec/obidos/ASIN/1565926285/davesill.

1.10.4.4. Running qmail

Richard Blum has written Running qmail, which is published by Sams. This book has received mixed reviews on the *qmail* mailing list.

For more information or to order this book, see http://www.amazon.com/exec/obidos/ASIN/0672319454/davesill.

1.10.4.5. qmail: Yuksek Performansli E-Posta Sunucu

Ismail Yenigul, et al, have written a Turkish-language *qmail* book. See http://www.acikakademi.com/catalog/qmail/.

1.10.5. List archives

The *qmail* e-mail mailing list, maintained by Dan Bernstein, is a valuable source of information. Web archives of the lists messages are kept at:

- http://www.ornl.gov/lists/mailing-lists/qmail/.
- http://tech.groups.yahoo.com/group/djb-qmail/?refstop=1 and
- http://securepoint.com/lists/html/Qmail/.

Most questions about *qmail* can be answered by searching the list archives first.

1.10.6. Other Web Sites

- http://cr.yp.to/qmail.html: the official *qmail* home page.
- http://www.qmail.org: the unofficial *qmail* home page. Contains lots of information about add-ons and patches, and links to many good *qmail* web pages on other sites.
- http://www.flounder.net/qmail/qmail-howto.html: Adam McKenna's HOWTO.

1.11. Support

1.11.1. Mailing lists

The following lists reside on list.cr.yp.to. In order to prevent harvesting of e-mail addresses by spammers, I'm avoiding the use of complete, valid addresses and "mailto" URL's.

The lists are managed by *ezmlm*, which uses different addresses to perform different functions:

- `*listname*@list.cr.yp.to`: the submission address. Messages sent here go out to all members of the list. Do **not** send subscribe/unsubscribe requests here: they won't work, and they'll annoy the subscribers.
- `*listname*-help@list.cr.yp.to`: the "help" address. Returns a list of command addresses and general usage information.
- `*listname*-subscribe`: send a blank message here to subscribe.
- `*listname*-unsubscribe`: send a blank message here to unsubscribe.

To specify a subscription/unsubscription address, say `joe@example.com`, send the message to:

- `*listname*-subscribe-joe=example.com@list.cr.yp.to`.

1.11.1.1. qmail

The main *qmail* mailing list. For discussion and questions/answers on most things related to *qmail*, except those with their own lists. Read Charles Cazabon's "12 Steps to qmail List Bliss" at http://pyropus.ca/personal/writings/12-steps-to-qmail-list-bliss.html before posting. Also read the FAQs and search the list archives before posting a question. When you ask questions, please try to include sufficient details to make it possible for people to respond:

- ***What did you do?*** What's your configuration? Include `qmail-showctl` output if you're not sure what's important. What action did you take? If this is a new installation, tell how you installed *qmail*.
- ***What did you expect to happen?*** What was the outcome you were trying to achieve? Don't assume the reader can guess.
- ***What did happen?*** Describe the actual result. Include log file clippings and copies of messages, with headers.

Note: The *qmail* list uses a utility called `qsecretary` to verify that messages posted to the list are not spam. Each message posted to the list will result in an e-mail confirmation request from `qsecretary`. Read the message and follow the directions to confirm your message--usually just replying to the `qsecretary` message will do the trick. Regular list posters often automate this process using autoresponders like Charles Cazabon's *pymsgauth*, available from http://pyropus.ca/software/pymsgauth/. *pymsgauth* verifies that message sent to the *qmail* list really came from you, so it won't automatically confirm forged messages sent to the list in your name.

1.11.1.2. qmailannounce

The *qmail* announcement mailing list. New releases are announced here. There's no submission address: it's a read-only list.

1.11.1.3. serialmail

For discussion of the *serialmail* package.

1.11.1.4. ezmlm

For discussion of the *ezmlm* mailing list manager.

1.11.2. Consultants

See http://www.qmail.org/top.html#paidsup for a list of commercial support providers.

1.11.3. FAQTS Knowledgebase

A database of *qmail*-related questions and answers is available at http://qmail.faqts.com/. If you have a question that the FAQ doesn't answer, try searching this knowledgebase. It's especially good at answering "how to" questions.

2. Installation

This section covers installing *qmail*. If you're an experienced system administrator, you can install *qmail* following the directions in `INSTALL` in the source distribution. The `INSTALL` directions are the **official** installation directions. They're more complex than the Life with qmail directions, and they assume that the reader is an experienced system and mail administrator. They're also outdated and don't reflect Bernstein's current recommended practices.

Note: If you choose to install using the following directions, you should read through the entire section to familiarize yourself with the overall process.

2.1. Installation Issues

2.1.1. Binary vs. source code

Before 2007-11-30, *qmail*'s restrictive licensing regarding the distribution of prebuilt packages meant that it was usually installed from a source code distribution. This may change in the future, expecially if *daemontools* and *ucspi-tcp* are placed in the public domain. For now, though, source code is still the preferred distribution method for *qmail*.

If you're not familiar with the distinction between source code and binaries, imagine ordering a pizza delivered to your house. The "binary" version of the pizza arrives ready-to-eat. The "source code" pizza comes as a kit containing flour, yeast, cheese, sauce, toppings, and directions for cooking the pizza yourself. Source code installations are a little more work for you, but if you follow the directions carefully, the result is the same--or even better. The self-baked pizza will be fresher, you can adjust the toppings to your preferences, and you'll know a lot more about your pizza and how it "works".

Safely running an Internet-accesible network service is not easy. An improperly configured service can put the host system at risk of attack or can be used to attack other sites--potentially exposing the administrator to legal liability. The more you know about how your network services work, the more likely they are to be properly configured and secure.

2.1.2. Tarball vs. OS-specific package

Some operating systems provide a mechanism for automating source code installations. Returning to the pizza analogy, they make it possible to package the ingredients and directions in such a way that you can just push a button and have the pizza bake itself.

Sounds great, doesn't it?

In practice, it might not be such a good idea. Assembling these packages is pretty difficult, and they might not do things the way they're supposed to. They're software, and like any software, they can have bugs. But even if they're bug free, the convenience they provide comes at a cost. You lose most of the advantages of the self-baked pizza: the ability to adjust the toppings to your personal preferences, and the knowledge of how the pizza was made and how it works.

If *qmail* was a pizza, the self-building approach might still be the way to go. But it's not: it's a fairly complicated system that the installer/maintainer needs to understand pretty well in order to be able to keep it working smoothly. The self-installing *qmail* is easier to install than the user-installed version, but the user-installed version is easier to configure and troubleshoot. You install *qmail* once on a system, but you will probably have several opportunities to reconfigure it or try to figure out why mail isn't flowing the way you think it should.

For this reason, I suggest installing *qmail* from scratch using the source code tarball, not a Red Hat RPM or other self-installing bundle.

2.2. Preparation

Before installing *qmail* on a system, especially if this is your first *qmail* installation, there are a few things you need to think about.

- If possible, install *qmail* on a "practice" system. This will give you a chance to make mistakes without losing important mail or interrupting mail service to your users.
- If you don't have a spare, and your system is already handling mail using *sendmail*, *smail*, or some other MTA, you can install and test most pieces of *qmail* without interfering with the existing service.
- When migrating a system from some other MTA to *qmail*--even if you've got some *qmail* experience under your belt--it's a good idea to formulate a plan.

2.3. System requirements

qmail will install and run on most UNIX and UNIX-like systems, but there are few requirements:

- About 10 megabytes of free space in the build area during the build. After the build, you can free all but 4 megabytes by removing the object files.
- A complete, functioning C development system including a compiler, system header files, and libraries. The build directions will show you how to tell if you've got the necessary parts.
- A few megabytes for the binaries, documentation, and configuration files.
- A safe filesystem for the queue. *qmail*'s reliability guarantee requires that the queue reside on a filesystem with traditional BSD FFS semantics. Most modern local filesystems meet these requirements with one important exception: the `link()` system call is often *asynchronous*--meaning that the results of the `link()` operation might not have been written to disk when the `link()` call returns. Bruce Guenter's *syncdir* library can be used to work around this problem. See syncdir in the Related Packages appendix for more information.
- Sufficient disk space for the queue. Small single-user systems only need a couple megabytes. Large servers may need a couple gigabytes.
- A compatible operating system. Most flavors of UNIX are acceptable. See `README` in the source tree for a list of known compatible releases.
- Access to a domain name server (DNS) is highly recommended. Without one, *qmail* can only send to remote systems configured in its `smtproutes` config file.
- Adequate network connectivity. *qmail* was designed for well-connected systems, so you probably don't want to try to use it for a mailing list server on a 28.8k dial-up. The *serialmail* package was designed to make *qmail* more compatible with poorly-connected systems. See the serialmail section in the Related Packages appendix for more information.

Note: The *qmail* `bin` directory must reside on a filesystem that allows the use of executable and `setuid()` files. Some OS distributions automatically mount `/var` with the `nosuid` or `noexec` options enabled. On such systems, either these options should be disabled or `/var/qmail/bin` should reside on another filesystem without these options enabled. The Create directories section describes how to use symbolic links to accomplish the latter. If `/var` is mounted `nosuid`, you'll probably see the following error message in the `qmail-send` logs:

```
delivery : deferral: Sorry,_message_has_wrong_owner._(#4.3.5)
```

Note: *qmail* won't install properly under Apple's OS X following these directions or the ones in the `INSTALL` file. Eben Pratt has documented procedures for installing under OS X, available from http://netdevice.com/qmail/#osx.

2.4. Download the source

OK, so you've got a system meeting the requirements ready for installing *qmail*. The first step is to download the source code for *qmail* and any other add-ons. You'll need *qmail*, of course, and you should probably also get *ucspi-tcp* and *daemontools*:

- *qmail*, http://www.qmail.org/netqmail-1.06.tar.gz
- *ucspi-tcp*, http://cr.yp.to/ucspi-tcp/ucspi-tcp-0.88.tar.gz
- *daemontools*, http://cr.yp.to/daemontools/daemontools-0.76.tar.gz

Retrieve these files using your web browser, web client (e.g., `wget`), or FTP client.

Note: If any of the links fail, it's probably because the package has been updated. In that case, you should go to http://cr.yp.to/software.html and follow the links to download the current version. It's possible that upgraded versions aren't compatible with the following instructions, so be sure to read the release notes in the "Upgrading from previous versions..." sections.

Note: This installation uses the *netqmail* distribution of *qmail*, which consists of the official *qmail* 1.03 tarball to which patches that fix a handful of bugs, deficiencies, and incompatibilities have been applied. See http://www.qmail.org/netqmail/ and the *netqmail* `CHANGES` file for more information.

2.5. Build the source

2.5.1. Verify build environment

The first thing you need to do is make sure that you have the necessary tools to compile a program. How you determine this depends on what flavor of UNIX you're using. The easiest way to tell, although it's not guaranteed, is to *try* it.

Note: If any one of these tests passes, you can stop and go on to the next section.

- At a command line prompt, type `cc` and press Enter:

```
$ cc
cc: No input files specified
$
```

- If you get a similar response, you have a C compiler in your path. If not, it doesn't necessarily mean you don't have one installed. You might, but maybe it isn't in your path. Of course it could also mean that you *don't* have one. Try these:

  - `/usr/bin/cc`
  - `/usr/bin/gcc`
  - `/usr/local/bin/cc`
  - `/usr/local/bin/gcc`
  - `/usr/ccs/bin/cc`
- If none of these works, you'll have to try something little more platform specific. At the prompt try one of these, depending on which OS you're using:

  - Red Hat Linux: `rpm -qa | grep gcc` or `rpm -qa | grep egcs`
  - FreeBSD: includes GCC by default
- If you can't find a compiler installed, you'll have to locate one and install it. Contact your OS vendor or other OS support channel.

In this section we'll go through the actual steps of compiling *qmail*. A way to cut-n-paste will come in handy here, but isn't really necessary.

2.5.2. Unpack the distribution

If you made it this far, you have a working C compiler and copies of the tarballs. Next, copy or move the tarballs to the directory you want to do the work in. `/usr/local/src` is a good choice for *qmail* and *ucspi-tcp*. *daemontools* should be built under `/package`.

At this time you probably want to become root, if you're not already.

```
su
umask 022
mkdir -p /usr/local/src
mv netqmail-1.06.tar.gz ucspi-tcp-0.88.tar.gz /usr/local/src
mkdir -p /package
mv daemontools-0.76.tar.gz /package
chmod 1755 /package
```

Now you can unpack the packages.

```
cd /usr/local/src
gunzip netqmail-1.06.tar.gz
tar xpf netqmail-1.06.tar
gunzip ucspi-tcp-0.88.tar.gz
tar xpf ucspi-tcp-0.88.tar
rm *.tar      # optional, unless space is very tight
cd /package
gunzip daemontools-0.76.tar.gz
tar xpf daemontools-0.76.tar
rm *.tar      # optional, again
```

There should now be directories called `/usr/local/src/netqmail-1.06`, `/usr/local/src/ucspi-tcp-0.88`, and `/package/admin/daemontools-0.76`.

2.5.3. Create directories

Since *qmail*'s installation program creates the subdirectories as they're needed, you only need to create the *qmail* "home" directory:

```
mkdir /var/qmail
```

And on to the next section.

Note: If you want some or all of the *qmail* files to reside elsewhere than `/var`, this can be accomplished by creating symbolic links under `/var/qmail` pointing to the other locations.

For ***example***, the *qmail* configuration files can be stored in `/etc/qmail` by doing:

```
mkdir /etc/qmail
ln -s /etc/qmail /var/qmail/control
```

2.5.4. Create users and groups

The easiest way to create the necessary users and groups is to create a little script file to do it for you. In the source directory you'll find a file called `INSTALL.ids`. It contains the command lines for many platforms, so copying the file to another name and editing that is quick and easy.

```
cd /usr/local/src/netqmail-1.06
cp INSTALL.ids IDS
```

Then, using your favorite editor, remove all of the file **except** the lines you want. For example, here's what `IDS` would look like for FreeBSD after editing:

```
pw groupadd nofiles
pw useradd qmaild -g nofiles -d /var/qmail -s /nonexistent
pw useradd alias -g nofiles -d /var/qmail/alias -s /nonexistent
pw useradd qmaill -g nofiles -d /var/qmail -s /nonexistent
pw useradd qmailp -g nofiles -d /var/qmail -s /nonexistent
pw groupadd qmail
pw useradd qmailq -g qmail -d /var/qmail -s /nonexistent
pw useradd qmailr -g qmail -d /var/qmail -s /nonexistent
pw useradd qmails -g qmail -d /var/qmail -s /nonexistent
```

Then to run it, either use `chmod` to make it executable or run it with `sh`:

First method:

```
chmod 700 IDS
./IDS
```

Second method:

```
/bin/sh IDS
```

When the script finishes, all of your users and groups will be created and you can go on to the next section.

But what do you do if your system isn't listed in `INSTALL.ids`? You'll have to create them manually. Start by using your favorite editor and editing `/etc/group`. You need to add the following two lines to the end of the file:

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

Note: Make sure that 2107 and 2108 aren't already used. If they are used, select two numbers that aren't already in use.

Next, using `vipw` (most systems have it, if not you'll need to use your editor again but this time on `/etc/passwd`) add these lines to the end of the file:

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

Note: Make sure 7790-7796 aren't already in use and that 2107 and 2108 are the same group ids you used above. If any of these UID's are already being used, select numbers that aren't already in use.

You don't specifically need to add any of these lines to the *end* of the file, that's just the easiest way to explain it here.

You're now ready to continue on to the next section.

2.5.5. Do the build

Now you can start building *qmail*. Change to the `/usr/local/src/netqmail-1.05/netqmail-1.05` directory and let's get started:

```
cd /usr/local/src/netqmail-1.06
```

In the Verify Build Environment section, you located your C compiler. If it's not called `cc` or the directory it resides in isn't in your `PATH` environment variable, you'll need to edit `conf-cc` and `conf-ld`. Say your compiler is `gcc`, and it's in your `PATH`. Simply edit `conf-cc` and `conf-ld` and replace "cc" with "gcc".

Now type the following:

```
make setup check
```

After the build is complete, you'll need to do your post installation configuration. A couple of scripts are provided to make this job a lot easier.

If your DNS is configured properly, this script should be all you need at this point:

```
./config
```

If, for some reason, `config` can't find your hostname in DNS, you'll have to run the `config-fast` script:

```
./config-fast the.full.hostname
```

For example, if your domain is example.com and the hostname of your computer is dolphin, your config-fast line would look like this:

```
./config-fast dolphin.example.com
```

Note: On a small local LAN you might want to use a pseudo domain such as ".local". E.g., if your hostname is "mash", you could do `./config-fast mash.local`. If you do this, be sure to configure *qmail* to use a valid Internet domain name in return addresses. (See section 3, Configuration.)

---

qmail is now installed on your system and is ready to be run! The next section will guide you through the steps of starting and testing qmail.

2.6. Install ucspi-tcp

Earlier, you unpacked the *qmail*, *ucspi-tcp*, and *daemontools* tarballs. Now change to the *ucspi-tcp* directory:

```
cd /usr/local/src/ucspi-tcp-0.88
```

In the Do the build section, if you modified `conf-cc` and `conf-ld`, you'll need to make the same changes in this directory.

Then do:

```
patch < /usr/local/src/netqmail-1.06/other-patches/ucspi-tcp-0.88.errno.patch
make
make setup check
```

That's it. *ucspi-tcp* is installed.

2.7. Install daemontools

Change to the *daemontools* build directory:

```
cd /package/admin/daemontools-0.76
```

Once again, if you modified <conf-cc> and `conf-ld` during the *qmail* and *ucspi-tcp* builds, you'll need to make the same changes in the `src` directory.

Then do:

```
cd src
patch < /usr/local/src/netqmail-1.06/other-patches/daemontools-0.76.errno.patch
cd ..
package/install
```

On BSD systems (no `/etc/inittab`), you'll need to reboot at this point to start `svscan`, the master service control daemon.

Use "`ps -ef | grep svscan`" or "`ps waux | grep svscan`" to verify that `svscan` is running.

Note: Under Solaris, you'll have to modify the `/etc/inittab` entry that starts `svscan` from:

```
SV:123456:respawn:/command/svscanboot
```

to:

```
SV:123456:respawn:/command/svscanboot </dev/null >/var/log/svscan 2>&1
```

or:

```
SV:123456:respawn:/command/svscanboot </dev/null >/dev/msglog 2>&1
```

Depending upon whether you want error messages resutling from starting `svscan` to be sent to a log file or the system console. For an explanation of why this is necessary, see:

http://marc.theaimsgroup.com/?l=log&m=100327801309834&w=2

Note: A Slackware user reports that the `SV` `/etc/inittab` entry has to be moved before the `x1` entry or `svscan` won't be started at boot-up.

2.8. Start *qmail*

2.8.1. `/var/qmail/rc`

The `/var/qmail/boot` directory contains example *qmail* boot scripts for different configurations: `/var/spool/mail` vs. `$HOME/Mailbox`, using *procmail* or *dot-forward*, and various combinations of these. Feel free to examine these, but for our installation, we'll use the following script:

```
#!/bin/sh

# Using stdout for logging
# Using control/defaultdelivery from qmail-local to deliver messages by default

exec env - PATH="/var/qmail/bin:$PATH" \
qmail-start "`cat /var/qmail/control/defaultdelivery`"
```

Note: This script uses backquotes (`` ` ``), not single quotes ('). For best results, copy and paste the scripts in this guide instead of retyping them.

Use your editor to create the above `/var/qmail/rc`, then execute these commands:

```
chmod 755 /var/qmail/rc
mkdir /var/log/qmail
```

At this point you need to decide the default delivery mode for messages that aren't delivered by a .qmail file. The following table outlines some common choices.

Mailbox format	**Name**	**Location**	**defaultdelivery**	**Comments**
mbox	`Mailbox`	`$HOME`	`./Mailbox`	most common, works with most MUA's
maildir	`Maildir`	`$HOME`	`./Maildir/`	more reliable, less MUA support
mbox	*`username`*	`/var/spool/mail`	See `INSTALL.vsm`	traditional UNIX mailbox

See `INSTALL.mbox`, `INSTALL.maildir`, and `INSTALL.vsm` for more information.

To select your default mailbox type, just enter the *defaultdelivery* value from the table into `/var/qmail/control/defaultdelivery`. E.g., to select the standard *qmail* `Mailbox` delivery, do:

```
echo ./Mailbox >/var/qmail/control/defaultdelivery
```

Note: `defaultdelivery` isn't a standard *qmail* control file. It's a feature of the above `/var/qmail/rc` file. The *defaultdelivery* argument to `qmail-start` is the *contents* of a `.qmail` file that specifies delivery instructions to be followed when no actual `.qmail` is found. Putting these instructions in a separate control file eliminates the need to quote shell metacharacters in the delivery instructions and avoids messy multi-line command arguments.

2.8.2. System start-up files

2.8.2.1. The `qmailctl` script

If you were to manually execute the `/var/qmail/rc` script, *qmail* would be *partially* started. But we want *qmail* started up automatically every time the system is booted and we want it shut down cleanly when the system is halted.

This is accomplished by creating a startup/shutdown script like the following in `/var/qmail/bin/qmailctl`:

```
#!/bin/sh

# description: the qmail MTA
```

```
PATH=/var/qmail/bin:/bin:/usr/bin:/usr/local/bin:/usr/local/sbin
export PATH

QMAILDUID=`id -u qmaild`
NOFILESGID=`id -g qmaild`

case "$1" in
  start)
    echo "Starting qmail"
    if svok /service/qmail-send ; then
      svc -u /service/qmail-send /service/qmail-send/log
    else
      echo "qmail-send supervise not running"
    fi
    if svok /service/qmail-smtpd ; then
      svc -u /service/qmail-smtpd /service/qmail-smtpd/log
    else
      echo "qmail-smtpd supervise not running"
    fi
    if [ -d /var/lock/subsys ]; then
      touch /var/lock/subsys/qmail
    fi
    ;;
  stop)
    echo "Stopping qmail..."
    echo "  qmail-smtpd"
    svc -d /service/qmail-smtpd /service/qmail-smtpd/log
    echo "  qmail-send"
    svc -d /service/qmail-send /service/qmail-send/log
    if [ -f /var/lock/subsys/qmail ]; then
      rm /var/lock/subsys/qmail
    fi
    ;;
  stat)
    svstat /service/qmail-send
    svstat /service/qmail-send/log
    svstat /service/qmail-smtpd
    svstat /service/qmail-smtpd/log
    qmail-qstat
    ;;
  doqueue|alrm|flush)
    echo "Flushing timeout table and sending ALRM signal to qmail-send."
    /var/qmail/bin/qmail-tcpok
    svc -a /service/qmail-send
    ;;
  queue)
    qmail-qstat
    qmail-qread
    ;;
  reload|hup)
    echo "Sending HUP signal to qmail-send."
```

```
    svc -h /service/qmail-send
    ;;
  pause)
    echo "Pausing qmail-send"
    svc -p /service/qmail-send
    echo "Pausing qmail-smtpd"
    svc -p /service/qmail-smtpd
    ;;
  cont)
    echo "Continuing qmail-send"
    svc -c /service/qmail-send
    echo "Continuing qmail-smtpd"
    svc -c /service/qmail-smtpd
    ;;
  restart)
    echo "Restarting qmail:"
    echo "* Stopping qmail-smtpd."
    svc -d /service/qmail-smtpd /service/qmail-smtpd/log
    echo "* Sending qmail-send SIGTERM and restarting."
    svc -t /service/qmail-send /service/qmail-send/log
    echo "* Restarting qmail-smtpd."
    svc -u /service/qmail-smtpd /service/qmail-smtpd/log
    ;;
  cdb)
    tcprules /etc/tcp.smtp.cdb /etc/tcp.smtp.tmp < /etc/tcp.smtp
    chmod 644 /etc/tcp.smtp.cdb
    echo "Reloaded /etc/tcp.smtp."
    ;;
  help)
    cat <<HELP
   stop -- stops mail service (smtp connections refused, nothing goes out)
  start -- starts mail service (smtp connection accepted, mail can go out)
  pause -- temporarily stops mail service (connections accepted, nothing leaves)
   cont -- continues paused mail service
   stat -- displays status of mail service
    cdb -- rebuild the tcpserver cdb file for smtp
restart -- stops and restarts smtp, sends qmail-send a TERM & restarts it
doqueue -- schedules queued messages for immediate delivery
 reload -- sends qmail-send HUP, rereading locals and virtualdomains
  queue -- shows status of queue
   alrm -- same as doqueue
  flush -- same as doqueue
    hup -- same as reload
HELP
    ;;
  *)
    echo "Usage: $0 {start|stop|restart|doqueue|flush|reload|stat|pause|cont|cdb|queue|help}"
    exit 1
    ;;
esac
```

```
exit 0
```

This script is available via http://lifewithqmail.org/qmailctl-script-dt70.

Create the script using your editor or by downloading it with your web browser (recommended).

Make the `qmailctl` script executable and link it to a directory in your path:

```
    chmod 755 /var/qmail/bin/qmailctl
    ln -s /var/qmail/bin/qmailctl /usr/bin
```

2.8.2.2. The `supervise` scripts

Now create the `supervise` directories for the *qmail* services:

```
    mkdir -p /var/qmail/supervise/qmail-send/log
    mkdir -p /var/qmail/supervise/qmail-smtpd/log
```

Create the `/var/qmail/supervise/qmail-send/run` file:

```
#!/bin/sh
exec /var/qmail/rc
```

Create the `/var/qmail/supervise/qmail-send/log/run` file:

```
#!/bin/sh
exec /usr/local/bin/setuidgid qmaill /usr/local/bin/multilog t /var/log/qmail
```

Create the `/var/qmail/supervise/qmail-smtpd/run` file:

```
#!/bin/sh

QMAILDUID=`id -u qmaild`
NOFILESGID=`id -g qmaild`
MAXSMTPD=`cat /var/qmail/control/concurrencyincoming`
LOCAL=`head -1 /var/qmail/control/me`

if [ -z "$QMAILDUID" -o -z "$NOFILESGID" -o -z "$MAXSMTPD" -o -z "$LOCAL" ]; then
    echo QMAILDUID, NOFILESGID, MAXSMTPD, or LOCAL is unset in
    echo /var/qmail/supervise/qmail-smtpd/run
    exit 1
fi

if [ ! -f /var/qmail/control/rcpthosts ]; then
    echo "No /var/qmail/control/rcpthosts!"
    echo "Refusing to start SMTP listener because it'll create an open relay"
    exit 1
fi

exec /usr/local/bin/softlimit -m 2000000 \
```

```
/usr/local/bin/tcpserver -v -R -l "$LOCAL" -x /etc/tcp.smtp.cdb -c "$MAXSMTPD" \
    -u "$QMAILDUID" -g "$NOFILESGID" 0 smtp /var/qmail/bin/qmail-smtpd 2>&1
```

Note: `concurrencyincoming` isn't a standard qmail control file. It's a feature of the above script. Also, that's `-1` (dash one) on the `LOCAL` line and `-l` (dash ell) on the `tcpserver` line.

Note: Under Solaris, the normal `id` program won't work right in this script. Instead of `id`, use `/usr/xpg4/bin/id`, e.g.:

```
QMAILDUID=`/usr/xpg4/bin/id -u qmaild`
NOFILESGID=`/usr/xpg4/bin/id -g qmaild`
```

Note: The memory limit specified in the `softlimit` command may need to be raised depending upon your operating system and hardware platform. If attempts to connect to port 25 fail or remote systems are unable to send you mail, or you see a message like:

```
/usr/local/bin/tcpserver: error while loading shared libraries:
libc.so.6: failed to map segment from shared object: Cannot
allocate memory
```

try raising it to 3000000 or 4000000.

Create the `concurrencyincoming` control file:

```
echo 20 > /var/qmail/control/concurrencyincoming
chmod 644 /var/qmail/control/concurrencyincoming
```

Create the `/var/qmail/supervise/qmail-smtpd/log/run` file:

```
#!/bin/sh
exec /usr/local/bin/setuidgid qmaill /usr/local/bin/multilog t /var/log/qmail/smtpd
```

Make the run files executable:

```
chmod 755 /var/qmail/supervise/qmail-send/run
chmod 755 /var/qmail/supervise/qmail-send/log/run
chmod 755 /var/qmail/supervise/qmail-smtpd/run
chmod 755 /var/qmail/supervise/qmail-smtpd/log/run
```

Then set up the log directories:

```
mkdir -p /var/log/qmail/smtpd
chown qmaill /var/log/qmail /var/log/qmail/smtpd
```

Finally, link the `supervise` directories into `/service`:

```
ln -s /var/qmail/supervise/qmail-send /var/qmail/supervise/qmail-smtpd /service
```

The `/service` directory is created when *daemontools* is installed.

Note: The *qmail* system will start automatically shortly after these links are created. If you don't want it running yet, do:

```
qmailctl stop
```

2.8.2.3. SMTP Access Control

Allow the local host to inject mail via SMTP:

```
echo '127.:allow,RELAYCLIENT=""' >>/etc/tcp.smtp
qmailctl cdb
```

2.8.3. Stop and disable the installed MTA

Although it's possible to run both *qmail* and your existing MTA, which is probably *Sendmail*, simultaneously, I don't recommend it unless you know what you're doing. And, frankly, if you're reading this, you probably don't know what you're doing. :-)

If your existing MTA is *Sendmail*, you should be able to stop it by running the `init.d` script with the "stop" argument. E.g., one of these should work:

```
/etc/init.d/sendmail stop
/sbin/init.d/sendmail stop
/etc/rc.d/init.d/sendmail stop
```

If you can't find an `init.d/sendmail` script, you can locate `sendmail`'s PID using "`ps -ef|grep sendmail`" or "`ps waux|grep sendmail`" and stop it using:

```
kill PID-of-sendmail
```

If your MTA isn't *Sendmail*, check its documentation for the correct shutdown procedure.

You should also consider removing the old MTA completely from the system. At least disable the `init.d` script so it doesn't try to start up again when the system is rebooted.

For Red Hat Linux, removing *Sendmail* can be accomplished by:

```
rpm -e --nodeps sendmail
```

Note: If you're using an RPM-based Linux distribution like Red Hat, removing the MTA package might cause problems down the road. Utilities that update the system might try to reinstall Sendmail, or MUA packages might not install because they can't tell an MTA is installed. Mate Wierdl provides a stub package called "fake_mta" that can be installed to prevent these problems. Simply install the RPM available from ftp://ftp.csi.hu/mw/fake_mta-1-10memphis.noarch.rpm .

Verify that nothing is listening to the SMTP port (25). Culprits could be the old MTA, `inetd`, or `xinetd`. The following command should produce no output (unless the qmail-smtpd service is running):

```
netstat -a | grep smtp
```

If something is running, make sure it's not *qmail* by doing:

```
qmailctl stop
```

The repeat the `netstat` check:

```
netstat -a | grep smtp
```

If you still get output from that command you'll have to locate the culprit and fix it before *qmail*'s SMTP service will run.

Lastly, replace any existing `/usr/lib/sendmail` with the *qmail* version:

```
mv /usr/lib/sendmail /usr/lib/sendmail.old                    # ignore errors
mv /usr/sbin/sendmail /usr/sbin/sendmail.old                  # ignore errors
chmod 0 /usr/lib/sendmail.old /usr/sbin/sendmail.old          # ignore errors
ln -s /var/qmail/bin/sendmail /usr/lib
ln -s /var/qmail/bin/sendmail /usr/sbin
```

Note: It's important to create the `sendmail` links, regardless of the previous MTA, if any. The `sendmail` command is invoked by many applications for sending mail.

The last step is to create a couple of system aliases.

2.8.4. Create System Aliases

There are three system aliases that should be created on all *qmail* installations:

Alias	Purpose
`postmaster`	RFC 2821 required, points to the mail adminstrator (you)
`mailer-daemon`	de facto standard recipient for some bounces
`root`	redirects mail from privileged account to the system administrator
`abuse`	de facto standard recipient for abuse complaints

To create these aliases, decide where you want each of them to go (a local user or a remote address) and create and populate the appropriate `.qmail` files. For example, say local user `dave` is both the system and mail administrator:

```
echo dave > /var/qmail/alias/.qmail-root
echo dave > /var/qmail/alias/.qmail-postmaster
ln -s .qmail-postmaster /var/qmail/alias/.qmail-mailer-daemon
ln -s .qmail-postmaster /var/qmail/alias/.qmail-abuse
chmod 644 /var/qmail/alias/.qmail-root /var/qmail/alias/.qmail-postmaster
```

See `INSTALL.alias` for more details.

2.8.5. Start *qmail*

If you stopped *qmail* above after creating the links in `/service`, you should restart it now:

```
qmailctl start
```

2.9. Test the Installation

qmail should now be running. First run `qmailctl stat` to verify that the services are up and running:

```
# qmailctl stat
/service/qmail-send: up (pid 30303) 187 seconds
/service/qmail-send/log: up (pid 30304) 187 seconds
/service/qmail-smtpd: up (pid 30305) 187 seconds
/service/qmail-smtpd/log: up (pid 30308) 187 seconds
messages in queue: 0
messages in queue but not yet preprocessed: 0
```

All four services should be "up" for more than a second. If they're not, you've probably got a typo in the associated run script or you skipped one or more steps in creating the necessary files, directories, or links. Go back through the installation step-by-step and double check your work. You can also download and run the `inst_check` script, available from http://lifewithqmail.org/inst_check. For example:

```
# sh inst_check
! /var/log/qmail has wrong owner, should be qmaill
...try: chown qmaill /var/log/qmail
#
```

If `inst_check` finds problems, fix them and re-run it. When everything looks right, `inst_check` will report:

```
Congratulations, your LWQ installation looks good!
```

The `readproctitle` program keeps a log of error messages generated by services managed by svscan. To see these messages, use `ps` or some other process listing command. For example, you might see something like:

```
# ps -efl | grep "service errors" | grep -v grep
000 S root      1006  1001  0  76   0    -   334 pipe_w Mar31 ?        00:00:00
readproctitle service errors: ...unable to start qmail-smtpd/run: exec format error
#
```

In this case, the problem is that there is an error in the first line of the `/service/qmail-smtpd/run` script--most likely caused by the file being is DOS format (CR-LF line endings instead of Unix's LF-only).

It sometimes helps to run a service manually in order to find configuration problems. For example, if your `qmail-smtpd/log` service isn't running, do:

```
cd /service/qmail-smtpd/log
svc -d .
./run
if no errors, enter a line of text and press ENTER
if still no errors, enter CTRL-D (end of file)
```

At this point, you should be able to identify the problem and fix it. Once that's done, return to the service's directory, if necessary, and do:

```
svc -u .
```

Once the services are all up with >1 second uptime, follow the instructions in `TEST.deliver` and `TEST.receive` to verify that they're working correctly. Note that using these instructions, logging will be accomplished by `multilog` to `/var/log/qmail`, not `splogger` to something like `/var/log/maillog`.

Note: If you chose maildir mailbox format as the default delivery method, you will need to create a `Maildir` directory in your home directory and `alias`'s home directory before trying these instructions. See the maildir section to see how to properly create this directory.

---

3. Configuration

You've got *qmail* installed, from the recommended source tarball method, one of the self-compiling packages, or a var-qmail package. This section contains information you will need to configure *qmail* to make it work the way you want it to.

3.1. Configuration Files

All of *qmail*'s system configuration files, with the exception of the `.qmail` files in `~alias`, reside in `/var/qmail/control`. The `qmail-control` man page contains a table like the following:

Control	Default	Used by	Purpose
badmailfrom	*none*	qmail-smtpd	blacklisted From addresses
bouncefrom	MAILER-DAEMON	qmail-send	username of bounce sender
bouncehost	me	qmail-send	hostname of bounce sender
concurrencyincoming	*none*	/service/qmail-smtpd/run	max simultaneous incoming SMTP connections
concurrencylocal	10	qmail-send	max simultaneous local deliveries
concurrencyremote	20	qmail-send	max simultaneous remote deliveries
defaultdelivery	*none*	/var/qmail/rc	default .qmail file
defaultdomain	me	qmail-inject	default domain name
defaulthost	me	qmail-inject	default host name
databytes	0	qmail-smtpd	max number of bytes in message (0=no limit)
doublebouncehost	me	qmail-send	host name of double bounce sender
doublebounceto	postmaster	qmail-send	user to receive double bounces
envnoathost	me	qmail-send	default domain for addresses without "@"
helohost	me	qmail-remote	host name used in SMTP HELO command
idhost	me	qmail-inject	host name for Message-ID's
localiphost	me	qmail-smtpd	name substituted for local IP address
locals	me	qmail-send	domains that we deliver locally
me	*FQDN of system*	various	default for many control files
morercpthosts	*none*	qmail-smtpd	secondary rcpthosts database
percenthack	*none*	qmail-send	domains that can use "%"-style relaying
plusdomain	me	qmail-inject	domain substituted for

			trailing "+"
qmqpservers	*none*	qmail-qmqpc	IP addresses of QMQP servers
queuelifetime	604800	qmail-send	seconds a message can remain in queue
rcpthosts	*none*	qmail-smtpd	domains that we accept mail for
smtpgreeting	me	qmail-smtpd	SMTP greeting message
smtproutes	*none*	qmail-remote	artificial SMTP routes
timeoutconnect	60	qmail-remote	how long, in seconds, to wait for SMTP connection
timeoutremote	1200	qmail-remote	how long, in seconds, to wait for remote server
timeoutsmtpd	1200	qmail-smtpd	how long, in seconds, to wait for SMTP client
virtualdomains	*none*	qmail-send	virtual domains and users

For more information about a particular control file, see the man page for the module listed under "Used by".

3.2. Relaying

3.2.1. Introduction

What is *relaying*? It's when an MTA accepts a message via SMTP that doesn't *appear* to be either **for** a local address or **from** a local sender.

In the pre-spam days, it was common for MTA's to be configured as *open* relays: promiscuous servers that would accept mail from anyone, for anyone.

Most MTA's now are configured to either completely disable relaying, or to only a allow certain trusted users or systems to use them as a relay.

Chris Johnson has written a very nice document on the topic for *qmail* users. I encourage you to visit http://www.palomine.net/qmail/relaying.html.

3.2.2. Disabling relaying

If you follow the official directions for installing *qmail*, relaying will be turned off by default. This is accomplished by populating the file `/var/qmail/control/rcpthosts` with the fully-qualified domain names listed in `locals` and `virtualdomains` (the local hosts). The name of the control file, `rcpthosts`, comes from the SMTP `RCPT` (recipient) command. In an SMTP session, RCPT is used to specify the addresses of the recipients of a message. `rcpthosts`, then, lists the valid hostnames that can appear in a RCPT address.

3.2.3. Allowing selective relaying

Most single-user and small workgroup servers can disable relaying completely, but if you have to support a distributed user community, you'll need a way to allow your users, and *only* your users, to use your system as a relay. This is accomplished by using *tcpserver* to set the `RELAYCLIENT` environment variable, which tells `qmail-smtpd` to override the `rcpthosts` file.

If you follow the installation instructions in this document, selective relaying will be enabled by default. To give a client relay access, add an entry to `/etc/tcp.smtp` like:

```
IP address of client:allow,RELAYCLIENT=""
```

Then rebuild the SMTP access database by doing:

```
qmailctl cdb
```

or:

```
tcprules /etc/tcp.smtp.cdb /etc/tcp.smtp.tmp < /etc/tcp.smtp
chmod 644 /etc/tcp.smtp*
```

If you followed the official installation instructions, Chris Johnson has written another very nice document on how to configure *qmail* to allow selected hosts to relay. See http://www.palomine.net/qmail/selectiverelay.html.

3.2.4. Relaying to a smart host

For anyone setting up a mail server on a typical home broadband service, there is a good chance that your IP address will get blacklisted by organizations like SORBS (http://www.dnsbl.sorbs.net/lookup.shtml) in an effort to block spam. Most ISPs provide an SMTP server that will relay all mail from their customers, and such servers are usually not blacklisted. For example, the Road Runner service in Cincinnati, Ohio, has `smtp-server.cinci.rr.com` available to their customers. You can tell qmail to route all outgoing SMTP traffic through that server by doing:

```
echo ":smtp-server.cinci.rr.com" > /var/qmail/control/smtproutes
```

The `smtproutes` file can perform more routing functions than this; see the `qmail-remote` man page for more details.

3.3. Multiple host names

If your system is known by more than one name, e.g., all addresses of the form *`user`*`@host1.example.com` can also be written as *`user`*`@example.com` or *`user`*`@mail.example.com`, then you need to tell *qmail* this so it'll know which addresses it should deliver locally and which messages it should accept from remote systems.

To do this, just add all of the names to two control files:

- `rcpthosts`, which tells `qmail-smtpd` to accept mail addressed to these hosts, and
- `locals`, which tells `qmail-send` that addresses on these hosts are to be delivered locally.

Send `qmail-send` a `HUP` (hangup) signal to tell it to reread `locals`. If you have `qmailctl`, you can do:

```
qmailctl reload
```

3.4. Virtual domains

Virtual domains are similar to the multiple host names discussed in the previous section, but there are some important differences. First, if `example.net` hosts the virtual domain `virtual.example.com`, it's generally **not** true that messages sent to `joe@example.net` should end up in the same mailbox as messages sent to `joe@virtual.example.com`. The namespace for each virtual domain is distinct.

With *qmail*, virtual domains are configured in the `virtualdomains` file, which consists of one or more entries of the form:

```
user@domain:prepend
```

qmail converts *`user@domain`* to *`prepend-user@domain`* and treats the result as if *`domain`* was local. The *`user@`* part is optional. If it's omitted, the entry matches *all* `@`*`domain`* addresses.

Returning to the example scenario above, if the example.net mail administrator wanted to create a virtual domain, `virtual.example.com`, under the administrative control of user `john`, the following entry in `virtualdomains` would accomplish that:

```
virtual.example.com:john
```

An incoming message to `joe@virtual.example.com` would be rewritten as `john-joe@virtual.example.com` and delivered locally. See the .qmail section, and the extension addresses subsection for more information about how john can manage his virtual domain.

As with multiple host names, all virtual domains must be listed in `rcpthosts` so `qmail-smtpd` will know to accept messages addressed to them. However, unlike multiple host names, virtual domains **must not** be added to `locals`.

After modifying `virtualdomains`, send `qmail-send` a `HUP` (hangup) signal to tell it to reread the file. If you have `qmailctl`, you can do:

```
qmailctl reload
```

Don't forget to add virtual domains to `rcpthosts`, too.

Note: Domain name server (DNS) mail exchanger (MX) records must be set up to direct messages for virtual domains to the appropriate mail server. This is a job for the name server administrator and is beyond the scope of this guide.

3.5. Aliases

qmail's standard aliasing mechanism is a natural outgrowth of *qmail*'s local delivery mechanism. `qmail-local` attempts to deliver a message addressed to *`localpart`*`@host` to a local user named *`localpart`*. If no matching user is found, the message is delivered to the `alias` user, a pseudo-user on all *qmail* systems whose home directory is usually `/var/qmail/alias`.

For example, say you want to create an `info@example.com` alias that forwards messages to user `tom`. On `example.com`, do, as user `root`:

```
echo \&tom > /var/qmail/alias/.qmail-info
```

The `.qmail` section and extension addresses subsection describe how to create `.qmail` files that specify which aliases exist, and what to do with messages sent to them.

The Gotchas appendix covers a couple of tricky cases regarding the usage of alias--aliases containing uppercase characters and dots ('.')--and `man dot-qmail` contains complete documentation of the usage of `.qmail` files.

Note that because of the way aliases are implemented in *qmail*, an alias can **never** override a valid user's deliveries. E.g., if `rachel` is a normal user, `~alias/.qmail-rachel` will not be used.

The *fastforward* package provides an alternative aliasing mechanism that puts multiple aliases in a single file compatible with *Sendmail*'s alias database.

The next section, qmail-users, describes another mechanism that can be used to implement aliases.

3.6. qmail-users

qmail-users is a system for assigning addresses to users. A series of configuration files resides under `/var/qmail/users`. The `assign` file is a table of assignments. There are two kinds of assignments: simple and wildcard.

Note: `assign` contains a series of assignments, one per line, followed by a line containing a single dot (.). If you create `assign` manually, don't forget the dot line.

3.6.1. Simple assignment

A simple assignment looks like:

=*address*:*user*:*uid*:*gid*:*directory*:*dash*:*extension*:

What this means is that messages received for *address* will be delivered as user *user*, with the specified *uid* and *gid*, and the file *directory*/.qmail*dashextension* will specify how the messages are to be delivered.

3.6.2. Wildcard assignment

A wildcard assignment looks like:

+*prefix*:*user*:*uid*:*gid*:*directory*:*dash*:*prepend*:

What this means is that messages received for addresses of the form *prefixrest* will be delivered as user *user*, with the specified *uid* and *gid*, and the file *directory*/.qmail*dashprependrest* will specify how the messages are to be delivered.

3.6.3. qmail-user programs

qmail-user has two helper programs: `qmail-newu` and `qmail-pw2u`.

`qmail-newu` processes the `assign` file and generates a constant database (CDB) file called `cdb` in `/var/qmail/users`. CDB is a binary format that can be accessed quickly by `qmail-lspawn`, even when there are thousands of assignments.

`qmail-pw2u` converts the system user database, `/etc/passwd`, into a series of assignments suitable for `assign`. `qmail-pw2u` uses a set of files to modify the translation rules.

- `include`: users to include
- `exclude`: users to exclude
- `mailnames`: alternative "mailnames" for users
- `subusers`: extra addresses handled by a user, with an optional `.qmail` extension
- `append`: miscellaneous assignments

Note: If you use `qmail-pw2u`, don't forget to re-run `qmail-pw2u` and `qmail-newu` whenever you add users, remove users, or change UID's or GID's. A typical sequence would be:

```
qmail-pw2u </etc/passwd >/var/qmail/users/assign
qmail-newu
```

3.7. Spam Control

Chris Hardie has written an excellent *qmail* Anti-Spam HOWTO. It's available from http://www.summersault.com/chris/techno/qmail/qmail-antispam.html.

3.8. Virus Scanning

Jason Haar has written Qmail-Scanner, a content scanning harness for *qmail*. See http://qmail-scanner.sourceforge.net/ for more information.

Qmail-Scanner includes a simple "policy-blocking" component (e.g. block *.scr, or block "Subject: Yellow!") as well as directly supporting many different antivirus "plugins" including the ClamAV Antivirus scanner available from http://www.ClamAV.net.

4. Usage

This section covers the usage of *qmail* by normal users. If you read or send mail on a *qmail* system, this is where you'll find information about how to do that with *qmail*.

4.1. `.qmail` files

Delivery of a user's mail is usually controlled by one or more ".qmail" (pronounced *dot kyoo mail*) files--files in the user's home directory with names beginning with `.qmail`. The `dot-qmail` man page describes `.qmail` file usage.

`.qmail` files contain a list of delivery instructions, one instruction per line. The first character of the line determines what kind of delivery is involved:

Character	**Delivery Type**	**Value**
#	none *(comment)*	ignored
|	program	command to be run by shell
/ or .	mbox *(if last char isn't a /)*	pathname of mbox *(including the / or .)*
/ or .	maildir *(if last char is a /)*	pathname of maildir *(including the / or .)*
&	forward	address to forward message
letter or number	forward	address to forward message *(including the first char)*

4.1.1. program delivery

When a program delivery instruction is encountered, *qmail* starts a shell (`/bin/sh`) to execute the command and feeds the command a copy of the incoming message on standard input. The `qmail-command` man page documents the details of this process.

Program delivery is very powerful, and can be used to implement a wide range of functionality such as message filtering, automatically responding to messages, and delivery via third-party delivery agents such as *procmail*.

E.g.:

```
|preline /usr/ucb/vacation djb
```

This causes *qmail* to start `preline`, pass it `/usr/ucb/vacation` and `djb` as arguments, and provide a copy of the message on standard input.

4.1.2. mbox delivery

Mbox is the standard UNIX mailbox format in which multiple messages are stored in a single file and messages are headed with a "`From `" line. This line looks like a header field, but it isn't one: it's just something the delivery agent adds so mail readers can tell where each message begins.

E.g.:

```
./Mailbox
```

This causes messages to be appended to `$HOME/Mailbox`, with a "`From `" line prepended. A simple mbox mailbox with a single message looks like:

```
From user1@example.net Thu May 13 18:34:50 1999
Received: (qmail 1287205 invoked from network); 13 May 1999 18:34:49 -0000
From: user1@example.net
To: user2@example.com
Subject: hey

What's up?
```

The first line was added at delivery by *qmail*.

4.1.3. maildir delivery

Maildir is a mailbox format created by Dan Bernstein to address the shortcomings of the mbox format. A maildir mailbox is a directory containing three subdirectories, `new`, `cur`, and `tmp`. Each message in a maildir mailbox is in a separate file in one of the subdirectories, depending upon its status: `new` is for unread messages, `cur` is for messages that have been seen, and `tmp` is for messages in the process of being delivered. The `maildir` man page describes the format of a maildir in detail.

One of the benefits of the maildir format is that, even though it doesn't use locking to prevent simultaneous updates from different delivery agents, it's reliable. This means maildir mailboxes can safely reside on NFS-mounted filesystems.

E.g.:

```
./Maildir/
```

This causes messages to be saved in `$HOME/Maildir`, a maildir-format mailbox.

Note: `qmail-local` can deliver mail to maildir mailboxes, but it can't create them. Maildir mailboxes should be created with the `maildirmake` program that comes with *qmail*. E.g., "`maildirmake ~/Maildir`". Be sure to run `maildirmake` as the owner of the maildir, **not** as root. Your `useradd` or `adduser` command might support a "skeleton" directory, e.g. `/etc/skel`, where you can create a maildir that will be copied for all new users.

4.1.4. forward delivery

Forward deliveries causes the message to be resent to the specified address. Addresses specified in `.qmail` files can't contain comment fields or extra spaces.

These are **wrong**:

```
&<user@example.com>
& user@example.com
&Joe User <user@example.com>
```

These are correct:

```
&user@example.com
user@example.com
&user
```

The first two cause `user@example.com` to receive a copy of the message. The last sends a copy to the local user `user`.

4.1.5. extension addresses

qmail supports user-controlled extension addresses. In addition to the base address, *`username@hostname.domain`*, users can receive mail at *`username-extension@hostname.domain`*. For the remainder of this section, I'll leave off the "*@hostname.domain*" part since we're considering actions that take place on the local system.

The delivery instructions for *`username`* are in `~`*`username`*`/.qmail` and the delivery instructions for *`username-extension`* are in `~`*`username`*`/.qmail-`*`extension`*.

For example, `dave-lwq@sparge.example.com` is controlled by `~dave/.qmail-lwq` on host sparge.

Extensions can have multiple fields, e.g., `dave-list-qmail`, controlled by `~dave/.qmail-list-qmail`. In this example, `dave-list-qmail` is subscribed to the qmail mailing list, and `~dave/.qmail-list-qmail` files the list messages in a separate mailbox.

`.qmail` files can be *wildcarded* using `-default`. So `dave-list-qmail` could also be handled by `~dave/.qmail-list-default`. This would allow one catch-all `.qmail` file to handle all `dave-list-`*`whatever`* addresses. Note that `dave-list` wouldn't be handled by `~dave/.qmail-list-default` because it doesn't match the "-" after "list".

qmail uses the closest match it finds. E.g., when a message comes in addressed to `dave-list-qmail`, it'll use the first one of the following that it finds:

```
.qmail-list-qmail
.qmail-list-default
.qmail-default
```

If no matching `.qmail` file is found, the delivery fails and the message bounces back to the sender.

4.2. Sending messages

Mail users usually don't use the MTA directly to send messages. Typically, messages are composed and sent using a Mail User Agent (MUA) such as *pine* or *mutt*, which then calls the MTA to deliver the message. The process of handing a message to the MTA is called *injection*.

There are two ways to inject messages into most MTA's: via the Simple Mail Transfer Protocol, SMTP, or using a program provided by the MTA for that purpose.

4.2.1. SMTP

MUA's can open a TCP connection to port 25, the standard SMTP port, on the local host or a designated mail server. The MUA and the MTA then engage in a dialogue that results in either:

- the message being transfered to the MTA, or
- a error status being returned to the MUA

SMTP has no mechanism for authentication, so no username or password is required to send a message. However, many MTA's refuse to accept messages that don't appear to be either from or for a local user. If a properly formatted message is rejected, relaying restrictions are the most likely cause. See the Relaying section for more information about relay configuration.

4.2.2. `/var/qmail/bin/sendmail`

For many years, *Sendmail* was *the* UNIX MTA. It was so ubiquitous, that many programmers just assumed that it was the MTA. As a result, *Sendmail*'s local injection mechanism became the standard Application Programmer's Interface (API) for local mail injection. *qmail* and other non-*Sendmail* MTA's provide a `sendmail` program that works the same way as the real *Sendmail*'s `sendmail` for local injection.

The *qmail* `sendmail`, which is normally in `/var/qmail/bin/sendmail`, usually replaces the *Sendmail* `sendmail` on *qmail* systems. Typical locations of the `sendmail` program include:

- `/usr/lib/sendmail`
- `/usr/sbin/sendmail`

On a *qmail* system, "`ls -l` *`path-to-sendmail`*" should show that `sendmail` is a symbolic link to `/var/qmail/bin/sendmail`:

```
$ ls -l /usr/lib/sendmail
lrwxrwxrwx   1 root     root           29 Feb 19 11:04 /usr/lib/sendmail -> /var/qmail/bin/sendmail
```

4.2.3. `qmail-inject`

In addition to emulating the `sendmail` API, *qmail* has its own injection program: `qmail-inject`. In fact, `sendmail` is just a wrapper around `qmail-inject`.

As an API, `sendmail` is probably better because it's much more widely available. The *qmail* API provided by `qmail-inject` will only work on systems with *qmail*, but the `sendmail` interface is nearly universal.

For example, to send a blank message to `joe@example.com`:

```
echo To: joe@example.com | /var/qmail/bin/qmail-inject
```

4.3. Environment Variables

Some *qmail* programs set or use environment variables. The following table lists these variables and describes their use.

Name	**Man page**	**Set or used**	**Purpose**
DATABYTES	`qmail-smtpd`	used	Overrides `control/databytes`

DEFAULT	`qmail-command`	set	Portion of address matching "-default" in a `.qmail` file name.
DTLINE	`qmail-command`	set	Delivered-To header field
EXT	`qmail-command`	set	The address extension
EXT2	`qmail-command`	set	Portion of `EXT` following first dash
EXT3	`qmail-command`	set	Portion of `EXT` following second dash
EXT4	`qmail-command`	set	Portion of `EXT` following third dash
HOME	`qmail-command`	set	The user's home directory
HOST	`qmail-command`	set	The *domain* part of the recipient address
HOST2	`qmail-command`	set	Portion of `HOST` preceding last dot.
HOST3	`qmail-command`	set	Portion of `HOST` preceding second-to-last dot
HOST4	`qmail-command`	set	Portion of `HOST` preceding third-to-last dot
LOCAL	`qmail-command`	set	The *local* part of the recipient address
LOGNAME	`qmail-inject`	used	User name in From header field (4)
MAILHOST	`qmail-inject`	used	Host name in From header field (2)
MAILNAME	`qmail-inject`	used	Personal name in From header field (2)
MAILUSER	`qmail-inject`	used	User name in From header field (2)
NAME	`qmail-inject`	used	Personal name in From header field (3)
NEWSENDER	`qmail-command`	set	Forwarding sender address (see "man dot-qmail")
QMAILDEFAULTDOMAIN	`qmail-inject`	used	Overrides `control/defaultdomain`
QMAILDEFAULTHOST	`qmail-inject`	used	Overrides `control/defaulthost`
QMAILHOST	`qmail-inject`	used	Host name in From header field (1)
QMAILIDHOST	`qmail-inject`	used	Overrides `control/idhost`
QMAILINJECT	`qmail-inject`	used	Specify various options (see next table)
QMAILMFTFILE	`qmail-inject`	used	File containing list of mailing list addresses for Mail-Followup-To generation
QMAILNAME	`qmail-inject`	used	

			Personal name in From header field (1)
QMAILPLUSDOMAIN	`qmail-inject`	used	Overrides `control/plusdomain`
QMAILSHOST	`qmail-inject`	used	Host name in envelope sender address
QMAILSUSER	`qmail-inject`	used	User name in envelope sender address
QMAILUSER	`qmail-inject`	used	User name in From header field (1)
RECIPIENT	`qmail-command`	set	Envelope recipient address
RELAYCLIENT	`qmail-smtpd`	used	Ignore `control/rcpthosts` and append value to recipient address
RPLINE	`qmail-command`	set	Return-Path header field
SENDER	`qmail-command`	set	Envelope sender address
UFLINE	`qmail-command`	set	UUCP-style "From " line
USER	`qmail-command`	set	The current user
USER	`qmail-inject`	used	User name in From header field (3)

QMAILINJECT Flags

Letter	**Purpose**
c	Use address-comment style for the From field
s	Do not look at any incoming Return-Path field
f	Delete any incoming From field
i	Delete any incoming Message-ID field
r	Use a per-recipient VERP
m	Use a per-message VERP

5. Advanced Topics

5.1. *procmail*

procmail is a popular Message Delivery Agent (MDA). The function of an MDA is to accept a message from the MTA for a specific user or mailbox, and deliver the message according to the user's desires. *procmail* can be used to "filter" messages by the content of various header fields or the body of the message. For example, messages from a particular person can be directed to a mailbox for just that person.

There are a couple tricks to running *procmail* with *qmail*. First, procmail is usually built to deliver to an mbox mailbox in `/var/spool/mail`. You can rebuild *procmail* to default to `$HOME` or you can instruct users not to rely on *procmail* to default the location of the mbox. Unless you patch it for `$HOME` delivery, `procmail` will still use `/var/spool/mail` for temporary files.

Another problem is that `qmail-command` and `procmail` don't have a common understanding of which exit codes mean what. `procmail` uses the standard UNIX exit codes: zero means *success*, nonzero means *failure*, and the cause of the failure is indicated by `/usr/include/sys/errno.h`. `qmail-command` uses certain nonzero codes to indicate permanent errors and the rest are considered temporary. A small shell script wrapper can be used to translate the exit codes for `qmail-command`. Such a wrapper was posted to the *qmail* list and is available from the archives at http://www.ornl.gov/lists/mailing-lists/qmail/1998/04/msg00487.html.

Also, older versions of *procmail* (prior to 3.14) don't deliver directly to maildir-format mailboxes. Your best bet is to upgrade to the current version of *procmail*. Another approach is *safecat*, a program that writes a message on standard input to a specified maildir. Users can write *procmail* recipes (delivery instructions) that use *safecat* to file the message. You can also skip *procmail* altogether, and use *maildrop*.

Finally, `procmail` expects the messages it receives to be in mbox format. Normal *qmail* program deliveries include only the actual mail message, not including a "`From` " line. The `preline` command can be used to format the message as `procmail` expects. The wrapper linked above includes `preline`.

For example, let's say user "dave" wants his mail to be processed by `procmail`. His system administrator has built `procmail` to deliver to `$HOME` by default, and has provided the exit code wrapper linked above, called `/usr/local/bin/qmail-procmail`. His `.qmail` file should look like:

```
|/usr/local/bin/qmail-procmail
```

5.2. POP and IMAP servers

qmail includes a POP server, *qmail-pop3d*, but it's not configured and installed as part of the *qmail* installation process. You can also use one of the other POP or IMAP servers available, although most of them were written for *Sendmail* and will require some work to use with *qmail*.

5.2.1. *qmail-pop3d*

qmail-pop3d is the POP server included with *qmail*. It's a fine POP server, and many *qmail* sites use it. It's modular, and supports multiple authentication schemes via alternative authentication modules.

Note: *qmail-pop3d* supports **only** maildir-format mailboxes, so if you have users logging into the POP server and running MUA's locally, they all have to support maildir. If all of your users read mail via POP, the mailbox format on the server is not an issue.

5.2.1.1. Architecture of *qmail-pop3d*

A *qmail-pop3d* server consists of three modules:

- `qmail-popup`--gets username/password
- `checkpassword`--authenticates username/password
- `qmail-pop3d`--the POP daemon

Typically, `qmail-popup` is run via `inetd` or `tcpserver`, listening to port 110, the POP3 port. When a connection is made, it prompts for the username and password. Then it invokes `checkpassword`, which verifies the username/password and invokes `qmail-pop3d` if they match.

5.2.1.2. Installation of *qmail-pop3d*

1. Completely install and test *qmail*. If you want all users to have POPable mailboxes, make sure *defaultdelivery* is set to `./Maildir/`. If you installed the `/var/qmail/rc` script from the Installation section, this is configured in `control/defaultdelivery`. If not, it's probably in `/var/qmail/rc` on the `qmail-start` command line.

2. Download a `checkpassword` program from http://www.qmail.org/top.html#checkpassword. The standard `checkpassword`, http://cr.yp.to/checkpwd.html, is a good choice if you don't need anything fancy.

3. Compile and install `checkpassword` according to the directions. Make sure you install it as `/bin/checkpassword`.

Note: If you install the standard `checkpassword`, don't forget to apply the `errno` patch after unpacking the source:

```
patch < /usr/local/src/netqmail-1.06/other-patches/checkpassword-0.90.errno.patch
```

4. `mkdir /var/qmail/supervise/qmail-pop3d`

5. Create a `/var/qmail/supervise/qmail-pop3d/run` script containing:

```
#!/bin/sh
exec /usr/local/bin/softlimit -m 2000000 \
    /usr/local/bin/tcpserver -v -R -H -l 0 0 110 /var/qmail/bin/qmail-popup \
        FQDN /bin/checkpassword /var/qmail/bin/qmail-pop3d Maildir 2>&1
```

where *FQDN* is the fully qualified domain name of the POP server you're setting up, e.g., `pop.example.net`.

Note: The memory limit specified in the `softlimit` command may need to be raised depending upon your operating system and hardware platform. If attempts to connect to port 110 fail or POP3 connections fail mysteriously, or you see a message like:

```
/usr/local/bin/tcpserver: error while loading shared libraries:
libc.so.6: failed to map segment from shared object: Cannot
allocate memory
```

try raising it to 3000000 or 5000000.

---

6. `mkdir /var/qmail/supervise/qmail-pop3d/log`

7. Create a `/var/qmail/supervise/qmail-pop3d/log/run` script containing:

```
#!/bin/sh
exec /usr/local/bin/setuidgid qmaill /usr/local/bin/multilog t \
    /var/log/qmail/pop3d
```

8. Set up the log directory and permissions on the `run` scripts, and link the service into `/service`:

```
chmod +t /var/qmail/supervise/qmail-pop3d   # if daemontools < 0.75
mkdir /var/log/qmail/pop3d
chown qmaill /var/log/qmail/pop3d
chmod 755 /var/qmail/supervise/qmail-pop3d/run
chmod 755 /var/qmail/supervise/qmail-pop3d/log/run
ln -s /var/qmail/supervise/qmail-pop3d /service
```

9. Add the following to `qmailctl`'s "start" section:

```
if svok /service/qmail-pop3d ; then
  svc -u /service/qmail-pop3d /service/qmail-pop3d/log
else
  echo qmail-pop3d supervise not running
fi
```

10. Add the following to `qmailctl`'s "stop" section:

```
echo "  qmail-pop3d"
svc -d /service/qmail-pop3d /service/qmail-pop3d/log
```

11. Add the following to `qmailctl`'s "stat" section:

```
svstat /service/qmail-pop3d
svstat /service/qmail-pop3d/log
```

12. Add the following to `qmailctl`'s "pause" section:

```
echo "Pausing qmail-pop3d"
svc -p /service/qmail-pop3d
```

13. Add the following to `qmailctl`'s "cont" section:

```
echo "Continuing qmail-pop3d"
svc -c /service/qmail-pop3d
```

14. Add the following to `qmailctl`'s "restart" section:

```
echo "* Restarting qmail-pop3d."
svc -t /service/qmail-pop3d /service/qmail-pop3d/log
```

5.2.2. *Qpopper*

If you need a POP daemon that works with mbox-format mailboxes, you can use Qualcomm's *Qpopper*. *Qpopper* is available from http://www.eudora.com/products/unsupported/qpopper/.

5.2.3. *Binc IMAP*

Andreas Hanssen has written the *Binc IMAP* server. Binc IMAP is designed to use the same authentication mechanism (`checkpassword`) that `qmail-pop3d` uses, so it's a good fit for *qmail* servers. Like `qmail-pop3d`, it supports only Maildir mailboxes. See: http://www.bincimap.org/.

5.2.4. *Dovecot*

Timo Sirainen has written *Dovecot*, a IMAP and POP server that supports both mbox and maildir mailboxes. It was designed to be secure. It's available from http://www.dovecot.org/.

5.2.5. *imap-maildir*

David R. Harris has cleaned up the patch that adds maildir support to the University of Washington IMAP server and documented the installation process. See http://www.davideous.com/imap-maildir/.

5.2.6. *Courier-IMAP*

Sam Varshavchik has written an IMAP server that supports maildir mailboxes **only**. It's available from http://www.courier-mta.org/imap/.

5.2.7. *Cyrus*

Carnegie Mellon University's Project Cyrus includes an IMAP server. It's available from http://asg.web.cmu.edu/cyrus/imapd/. Rick Updegrove has written a `qmail2cyrus` wrapper for delivering messages to a Cyrus mail store. This wrapper is available from http://msgs.securepoint.com/cgi-bin/get/qmail0308/41/1/1.html.

5.3. POP and IMAP clients

5.3.1. *fetchmail*

fetchmail is a program that retrieves mail from a POP or IMAP server and re-injects it locally. *fetchmail* has no trouble retrieving mail from *qmail* servers, but there are a couple tricks for making it work well on a *qmail* client.

Here's a sample `.fetchmailrc` for a user on a *qmail* system:

```
poll mail.example.net proto pop3 nodns
    user dsill with password flubgart is dave here
    fetchall forcecr
```

This instructs *fetchmail* to connect to mail.example.net via POP3, log in as user `dsill`, password `flubgart`, retrieve all messages, and deliver them to dave@localhost. The `forcecr` causes *fetchmail* to end each line with a carriage return when injecting the message on the local system via SMTP. *qmail* requires this.

5.3.2. *getmail*

getmail is a program that retrieves mail from a POP server and delivers it to a maildir mailbox. It's actually a Python script, so you may need to install the Python interpreter before you can use *getmail*

getmail was written by Charles Cazabon, who maintains a web page for it at http://pyropus.ca/software/getmail/.

5.4. Multi-RCPT vs. Single RCPT delivery

Say you're an MTA, and one of your users sends a message to three people on hostx.example.com. There are several ways you could do this.

1. You could open an SMTP connection to hostx, send a copy of the message to the first user, send a copy to the second user, send a copy to the third user, then close the connection.
2. You could start three processes, each of which opens an SMTP connection to hostx, sends a copy of the message to one of the users, then closes the connection.
3. You could open an SMTP connection to host, send a copy of the message addressed to *all three users*, then close the connection.

The first method is clearly inferior to the third. Even if the message is tiny, it'll take at least as long. And if the message is large, it'll take a lot longer *and* use more network bandwidth.

So scratch that one.

The second and third methods are a little more interesting.

The third method only opens one connection to hostx, and only sends one copy of the message. That makes for efficient use of bandwidth.

The second method uses multiple connections and sends multiple copies of the message. That "wastes" bandwidth, but due to the nature of the SMTP protocol, requires fewer round-trip delays, and is faster than the third method. It's also simpler than the third method, so the MTA can be coded in a more straightforward manner. And finally, because each recipient gets their own copy of the message, it's possible for the MTA to implement VERPs (see next section).

qmail always uses the second method (single RCPT). There are no patches to implement the third method (multiple RCPT)--it would require **major** work.

Although there are pathological cases where it can be slower than multiple RCPT, the simplicity and VERP advantages outweigh that overall.

Single RCPT delivery *does* use more bandwidth than multiple RCPT delivery, but the difference is often exaggerated. Most messages have, at most, a couple recipients, and they're usually on separate hosts, so multi-RCPT delivery buys them nothing. Even on a list server, where multi-RCPT delivery could help, the potential gains are small because SMTP uses only a fraction of the bandwidth over most links--HTTP usually gets the lion's share.

For example, if 10% of your uplink's bandwidth goes to SMTP, and your SMTP bandwidth could be reduced by, say, 25%, by using multi-RCPT delivery, that would only drop your SMTP bandwidth to 7.5%.

5.5. VERP

When a message is undeliverable, the MTA responsible is supposed to return a bounce message to the envelope return path (ERP). The bounce message should include the address of the recipient, the reason the message is undeliverable, and whether the problem is temporary or permanent. Some MTA's don't do the right thing, though. They might send the bounce to the address in the From header field, or the bounce might not identify the recipient.

For most user-to-user messages, these problems aren't too bad. One can usually figure things out based on the timing of the bounce or the contents. For mailing lists, the problem of bad bounces is more serious. Subscribers move, forwarding mail to their new address. If the new address starts having delivery problems, it can be impossible to tell which subscriber's mail is bouncing if the bounce message only includes the new address.

Dan Bernstein came up with a solution to this problem called VERP (Variable Envelope Return Path). Using VERPs, each message sent to each subscriber to a list has a unique return path. This allows a bounce handler to identify the problem subscriber.

For example, a typical non-VERP'ed mailing list has a return address of the form `listname-owner@domain`. For a VERP'ed list, the return address would look like `listname-owner-subscriber=sdomain@ldomain`, where the subscriber's address, `subscriber@sdomain`, is embedded between the "owner" and the "`@`". (The "`@`" in the subscriber's address is replaced with an "=".)

The *ezmlm* list manager uses VERPs to automatically handle bounces. It even provides subscribers with temporary delivery problems with a list of the messages they missed so they can retrieve them from the archive.

Russell Nelson wrote a bounce manager for *Majordomo* under *qmail*, but he no longer maintains it. It's available from http://www.qmail.org/bounceman-0.4.shar.

5.6. Troubleshooting

5.6.1. Processes

A properly-running, complete, but minimal *qmail* installation should always have the following four processes:

- `qmail-send` running as user `qmails`
- `qmail-clean` running as user `qmailq`
- `qmail-rspawn` running as user `qmailr`

- `qmail-lspawn` running as user `root`

Depending upon your flavor of UNIX, one of the following two commands should list these processes, and possibly a few more:

```
ps -ef | grep qmail
ps waux | grep qmail
```

For example:

```
[dave@sparge dave]$ ps waux|grep qmail
dave      2222  0.0  0.8   836   348  p4 S    10:25   0:00 grep qmail
qmaild     351  0.0  1.0   840   400  ?  S N  12:43   0:00 /usr/local/bin/tcpserver -v -x /etc/tcp.smtp.cdb -u 49491 -g 31314 0 smtp /var/qmail/bin/qmail-smtpd-
qmaild    2220  0.0  1.0   844   420  ?  S N  10:25   0:00 /usr/local/bin/tcpserver -v -x /etc/tcp.smtp.cdb -u 49491 -g 31314 0 smtp /var/qmail/bin/qmail-smtpd-
qmaill     365  0.0  0.8   748   344  ?  S N  12:43   0:00 splogger qmail
qmailq 368 0.0 0.7 736 292 ? S N 12:43 0:00 qmail-clean
qmailr 367 0.0 0.6 732 272 ? S N 12:43 0:00 qmail-rspawn
qmails 350 0.0 0.8 776 336 ? S N 12:43 0:00 qmail-send
root       340  0.0  0.6   724   252  ?  S N  12:43   0:00 /usr/local/sbin/supervise /var/supervise/qmail-send /var/qmail/rc
root       341  0.0  0.6   724   252  ?  S N  12:43   0:00 /usr/local/sbin/supervise /var/supervise/tcpserver-qmail /usr/local/bin/tcpserver -v -x /etc/tcp.smtp
root 366 0.0 0.7 736 276 ? S N 12:43 0:00 qmail-lspawn ./Mailbox
[dave@sparge dave]$
```

If you run *qmail* or `qmail-smtpd` under `supervise`, as in the example above, you should see those processes as well. And if run `qmail-smtpd` under `tcpserver`, you should see a parent `tcpserver` process *plus* an additional `tcpserver` process for each active *incoming* SMTP connection.

If you use `splogger` (or `multilog` or `cyclog`) to handle logging, you'll have a `splogger` (or `multilog` or `cyclog`) process or two running as user `qmaill`.

Also, if *qmail* is busy delivering messages locally or remotely, you'll see up to `concurrencylocal` `qmail-local` processes and up to `concurrencyremote` `qmail-remote` processes.

5.6.2. Logs

5.6.2.1. `multilog`

`multilog`, which is part of the *daemontools* package, logs messages to a series of files in a specified directory.

The log directory is specified on the `multilog` command line, so you can find it by examining your *qmail* startup scripts.

The number of files in the log directory, and the maximum size of each file, are determined by `multilog` options. The log file names are the TAI (Temps Atomique International) timestamps of the time at which the file was started. The `tai64nlocal` command, also from *daemontools*, converts TAI timestamps into local, human-readable timestamps.

A typical `multilog` log entry looks like:

```
@4000000038c3eeb104a6ecf4 delivery 153: success: did_1+0+0/
```

"@4000000038c3eeb104a6ecf4" is the optional, but recommended, TAI timestamp. "delivery 153: success: did_1+0+0/" is the log message itself.

5.6.2.2. `splogger`

`splogger` uses the *syslog* logging system to timestamp messages and send them to the *syslog* daemon. *Syslog* is configured in `/etc/syslog.conf`. Messages sent to *syslog* have a *facility* and *priority*. Entries in `/etc/syslog.conf` filter on the facility and priority to direct the messages to the desired log file, remote log host, or the console. `splogger` logs to the `mail` facility, by default, so `grep`'ing the `syslog.conf` file for "mail" should show the disposition of *qmail*'s log messages.

Typical locations include:

- `/var/log/syslog`
- `/var/adm/SYSLOG`
- `/var/log/maillog`

A typical *syslog* log entry looks like:

```
Jun  3 11:35:23 sparge qmail: 928424123.963558 delivery 153: success: did_1+0+0/
```

"Jun 3 11:35:23" is the *syslog* timestamp.

"sparge" is the name of the system that sent the message.

"qmail:" is the tag `splogger` places on all *qmail* log entries.

"928424123.963558" is an optional TAI timestamp (see next section).

"delivery 153: success: did_1+0+0/" is the log message itself.

5.6.2.3. Log messages

Here's a typical log sequence for a message sent to a remote system from the local system:

```
1 @4000000038c3eeb027f41c7c new msg 93869
2 @4000000038c3eeb027f6b0a4 info msg 93869: bytes 2343 from <dave@sill.org> qp 18695 uid 49491
3 @4000000038c3eeb02877ee94 starting delivery 2392: msg 93869 to remote lwq@w3.to
4 @4000000038c3eeb0287b55ac status: local 0/10 remote 1/20
5 @4000000038c3eeb104a13804 delivery 2392: success: 209.85.127.177_accepted_message.
   /Remote_host_said:_250_CAA01516_Message_accepted_for_delivery/
6 @4000000038c3eeb104a4492c status: local 0/10 remote 0/20
7 @4000000038c3eeb104a6ecf4 end msg 93869
```

Line 1 indicates that *qmail* has received a new message, and its queue ID is 93869. The queue ID is the i-node number of the `/var/qmail/queue/mess/NN/` file--the queue file that contains the message. The queue ID is guaranteed to be unique as long as the message remains in the queue.

Line 2 says that the message is from `dave@sill.org` and is 2343 bytes.

Line 3 says `qmail-remote` is starting to deliver the message to `lwq@w3.to`, and it's assigning the ID 2392 to the delivery.

Line 4 says 0 local deliveries and 1 remote delivery are pending.

Line 5 says delivery 2392 is complete and successful, and it returns the remote server's response, which often contains information the remote mail administrator would find helpful in tracking a delivery. In this case, the "CAA01516" is the remote system's delivery ID.

Line 6 says 0 local deliveries and 0 remote deliveries are pending, i.e., the delivery is complete.

Line 7 says that the message has been delivered completely and removed from the queue. At this point, the queue ID, 93869, is reusable for another delivery.

5.7. Big Servers

See also *qmail-ldap*.

5.7.1. Scalable parallelism

Use a fast NFS network file server to store user directories. Set up multiple equal-preference SMTP servers delivering to maildir mailboxes on the file server.

5.8. Migrating from *Sendmail* to *qmail*

Check Dan Bernstein's *Sendmail->qmail* page at http://cr.yp.to/qmail/sendmail.html.

5.9. Mailing List Managers

Mailing list managers (MLM's) are systems that help list owners run mailing lists. Their duties fall into two main divisions: managing the lists of subscribers, and controlling the resending of messages to the subscribers.

Most (all?) UNIX mailing list managers can be made to work with *qmail*.

5.9.1. *ezmlm*

ezmlm was written by Dan Bernstein, the author of *qmail*. It was written for use with *qmail*, and relies on several features of *qmail*. Most notably, it uses VERPs to reliably process bounce messages. *ezmlm* is somewhat unique among MLM's in that it doesn't process commands sent to a central MLM address: it appends the command to the name of the list. E.g., to subscribe to the "foo@list.example.net" list, one sends a message to "foo-subscribe@list.example.net".

For more information about *ezmlm*, see http://www.ezmlm.org/, the unofficial *ezmlm* web site, and the official home of *ezmlm-idx*, a very nice add-on that includes many useful features.

5.9.2. *Majordomo*

Majordomo is one of the most popular UNIX MLMs. It works fine with *qmail* provided a few simple changes are made. Russ Allbery has written a FAQ about *qmail*/*Majordomo* available from http://web.archive.org/web/20050308091420/http://www.eyrie.org/~eagle/faqs/mjqmail.html.

5.10. Patches

Various source code patches are available for *qmail*. To install a patch, download it, cd to the *qmail* source tree, and apply it using the `patch` command.

```
cd /usr/local/src/qmail/qmail-1.03
patch -p0 </tmp/patchfile
```

Note: See the `patch man` page for more information. This is just an example. Also, you might need to use a current version of GNU patch to successfully apply some patches. See http://www.gnu.org/software/patch/patch.html.

Stop *qmail* by killing `qmail-send` or, if you installed the `qmailctl` script in the Installation section, do:

```
qmailctl stop
```

Then rebuild and install the new binaries:

```
make setup check
```

And restart *qmail*:

```
qmailctl start
```

Finally, test *qmail*--especially the part you patched.

Note: Although http://www.qmail.org/ lists many patches for *qmail*, **none** of the them have been approved by the author of qmail. They may introduce security, reliability, efficiency, or functionality problems not present in *qmail*. Most *qmail* installations only require the some of the Recommended patches. **You should not install any patches that you don't clearly require.**

5.10.1. Recommended Patches

qmail.org has a "Recommended Patches" section: http://qmail.org/top.html#patches. These patches address the few known bugs in qmail.

Note: all of the Recommended Patches have been included in the *netqmail* distribution. See http://www.qmail.org/netqmail/.

5.10.1.1. `errno.h` patches

This patch fixes a problem with missing `errno.h` inclusions. See http://article.gmane.org/gmane.mail.qmail.general/13960 for a detailed explanation and the patch itself.

Mate Wierdl has `errno.h` patches for all of Dan Bernstein's software including *qmail*, *daemontools*, and *ucspi-tcp*. These patches are available from http://www.thedjbway.org/patches/djb_errno_patches.tgz.

5.10.1.2. qmail-local TAB patch

This patch fixes a minor bug in the parsing of `.qmail` files that start with TAB characters.
http://www.ornl.gov/lists/mailing-lists/qmail/2000/10/msg00696.html

5.10.1.3. IP 0.0.0.0 patch

This patch causes the IP address 0.0.0.0 to be recognized as local.
http://www.suspectclass.com/~sgifford/qmail/qmail-0.0.0.0.patch

5.10.2. DNS

Historically, DNS responses have been limited to 512 bytes. Some large sites have started returning MX responses longer than that. *qmail* and many other programs have a problem with Domain Name Server (DNS) queries that return very large results. There are two ways to fix this in *qmail* and one workaround that might be sufficient for some applications.

5.10.2.1. Christopher K. Davis' patch, http://www.ckdhr.com/ckd/qmail-103.patch

This is an adaptation of a patch by Chuck Foster that should work with any resolver library, no matter how old, and uses a guard byte to avoid the "number of bytes placed in the buffer" library bug. It reallocates only once, to 65536, rather than just to the size needed, so it can be less memory-efficient than Chuck's patch (though, like his patch, it only reallocates if the response is larger than PACKETSZ, which defaults to 512 bytes). After reallocating, it forces a TCP query, rather than requiring the resolver library to do so (avoiding an extra round-trip between qmail and the name server, though if they're on the same machine or local network this is not a big worry).

5.10.2.2. Bump the packet buffer size up to 65536

Works with recent BIND resolver libraries, which will automatically do a TCP query within the library code if the reply comes back with the truncation bit set. This is the simplest fix, though it's also *potentially* the most wasteful of memory, depending on how your system handles paging. To do this, just replace `PACKETSZ` with `65536` in `dns.c` and rebuild *qmail*.

5.10.2.3. Run `dnscache` from *djbdns*

`dnscache` is, as the name implies, a caching DNS server. It knows how to handle large DNS responses and removes unnecessary information from them, so the response it returns is usually much smaller than the direct response. It also generally improves DNS lookup performance for all services that use DNS. Because it doesn't require patching *qmail*, this might be an acceptable workaround. Unfortunately, it's not a complete fix because responses can still be too large for *qmail*. See the djbdns section under Related Packages for more information.

5.10.3. *qmail-ldap*

This patch, by Andre Oppermann, et al, implements Lightweight Directory Access Protocol (LDAP) support in qmail. LDAP is like a network phone book. Using *qmail-ldap*, it should be possible for a POP server to serve many thousands of users. See http://www.nrg4u.com/.

5.11. QMTP

QMTP is the Quick Mail Transfer Protocol, an SMTP replacement protocol designed by Dan Bernstein. The protocol is defined at http://cr.yp.to/proto/qmtp.txt. QMTP is similar to SMTP, but is simpler, faster, and incompatible with SMTP. *qmail* includes a QMTP server, `qmail-qmtpd`, which is run very much like `qmail-smtpd`. QMTP usually uses port 209.

qmail doesn't include a QMTP *client*, but the *serialmail* package does. `maildirqmtp` takes a maildir mailbox and delivers the messages it contains to designated QMTP server via QMTP.

QMTP is not a drop-in replacement for SMTP, and is not yet in widespread use across the Internet.

Russ Nelson has a patch for `qmail-remote` that supports QMTP. It's available from http://www.qmail.org/qmail-1.03-qmtpc.patch. He also has a tarball that can be extracted in `/service` to enable a QMTP service. It's available from http://www.qmail.org/qmtpd-service.tar.gz.

5.12. Rejecting Invalid Recipients During SMTP Dialogue

When a remote server connects to `qmail-smtpd` and offers it a message, `qmail-smtpd` checks the recipient addresses against the contents of `control/rcpthosts`. If the host or domain after the `@` symbol is listed in `control/rcpthosts`, `qmail-smtpd` accepts the message, it's placed in the queue, and `qmail-send` attempts delivery. If the local recipient is invalid--there's no user or alias by that name--`qmail-send` generates a bounce message and sends it to the return address specified during the SMTP dialogue.

In a well-behaved world, either strategy would be fine. Unfortunately, there are a lot of poorly-behaved spammers out there. Some will attempt to deliver messages to recipients that "might" exist on your server--using a database of common names, a dictionary, or even a generated list of all possible alphanumeric strings.

On a qmail system, such spam attacks can inflict a substantial load on the system, fill the queue with junk, and delay the delivery of valid messages.

Some MTAs validate the local recipient during the SMTP dialogue and refuse to accept the message if the recipient is invalid. This saves the server from a lot of unnecessary work, but has a negative side effect, too. Using this validation, spammers can quickly determine which addresses are valid.

There are several ways to implement recipient validation during the SMTP dialogue with *qmail*. Eben Pratt has assembled a list of them at http://netdevice.com/qmail/rcptck/. Most of these solutions require maintaining a database of valid or invalid recipients or patterns. One that doesn't is Paul Jarc's qmail-realrcptto, available from http://code.dogmap.org./qmail/.

5.13. TLS and STARTTLS

Scott Gifford has written a very thorough and detailed step-by-step guide to using transport layer security (TLS) with *qmail*. The guide covers STARTTLS for SMTP and STLS for POP3D specifically for *netqmail*. It's available at http://www.suspectclass.com/~sgifford/ucspi-tls/ucspi-tls-qmail-howto.html.

A. Acknowledgments

First, thanks to Dan Bernstein for designing and writing such a powerful and elegant system. After nearly ten years of use, *qmail* still impresses me.

I'd also like to thank the members of the *qmail* mailing list. Two members deserve special mention. The first is Russ Nelson, one of the most helpful, patient, knowledgeable, and *funny* contributors. His contributions to the *qmail* community are second only to DJB's. The second is Charles Cazabon, who's close on Russ' heels. Charles is currently the major contributor to the mailing list, answering more questions correctly than anyone else. Charles has also written a couple of very useful utilities, getmail and pymsgauth, and was technical editor for The qmail Handbook where his contributions were *critical* to the success of the book, and for which he has received too little reward and recognition.

Thanks also to everyone who reviewed or contributed to this document, including:

- Vince Vielhaber
- Chris Green
- Christopher K. Davis
- Scott Schwartz
- Fred Lindberg
- Russell P. Sutherland
- Alex Miller
- Tim Hunter
- Frank D. Cringle
- Mahlon Smith
- Rogerio Brito
- Tony Hansmann
- Matthias Andree
- Tillman Hodgson
- Stefan Witzel
- Scott Gifford
- too many others to mention...

Special thanks to Henning Brauer for donating the lifewithqmail.org domain *and* hosting it!

Special thanks also to Michael M. Kadrie of ATLAS Design Group, http://www.atlasdesigngroup.com, for the nifty new *qmail* logo!

Life with qmail was written using Simple Document Format (SDF), a very cool Perl-based markup language that generates HTML, plain text, PostScript, POD, and other formats. It made the job *much* easier. See http://search.cpan.org/author/IANC/sdf-2.001/ for more information.

B. Related Packages

B.1. *dot-forward*

Sendmail uses `.forward` files, pronounced *dot forward*, 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: easy to install and configure. The source is available from http://cr.yp.to/software/dot-forward-0.71.tar.gz.

dot-forward was written by Dan Bernstein, who maintains a web page for it at http://cr.yp.to/dot-forward.html.

B.2. *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 you don't want to convert, *fastforward* gives *qmail* the ability to use the aliases file as-is.

The source is available from http://cr.yp.to/software/fastforward-0.51.tar.gz.

fastforward was written by Dan Bernstein, who maintains a web page for it at http://cr.yp.to/fastforward.html.

B.3. *ucspi-tcp*

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

Inetd is the standard network server "super-server". It can be configured through `/etc/inetd.conf` to run `qmail-smtpd`, but the recommended tool 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*.

`tcpserver` is preferred over *inetd* because:

- `tcpserver` allows one to limit the number of simultaneous connections to a service. *Inetd* has a connection-rate limiting mechanism that temporarily disables services that are "too" busy.
- `tcpserver` can be configured to deny access to certain hosts or to recognize local hosts and flag them so `qmail-smtpd` can treat them differently.
- `tcpserver` is the only server supported by the author of *qmail*.

The source is available from http://cr.yp.to/ucspi-tcp/ucspi-tcp-0.88.tar.gz.

Gerrit Pape distributes the documentation for *ucspi-tcp* as `man` pages from
http://smarden.org/pape/djb/.

ucspi-tcp was written by Dan Bernstein, who maintains a web page for it at
http://cr.yp.to/ucspi-tcp.html.

B.4. *daemontools*

The *daemontools* package contains a set of utilities for controlling and monitoring services. It's not mandatory, but it's highly recommended, especially for busy systems. It includes:

- `supervise`, which monitors a service and restarts it if it dies.
- `svc`, which talks to `supervise` and allows one to stop, pause, or restart the service.
- `multilog`, which maintains a log for a service, automatically rotating it to keep it under the configured size.
- `setuidgid`, which runs programs for the superuser with a normal user's UID and GID.

The source for *daemontools* is available from: http://cr.yp.to/daemontools/daemontools-0.76.tar.gz.

Gerrit Pape distributes the documentation for *daemontools* as `man` pages from
http://smarden.org/pape/djb/.

daemontools was written by Dan Bernstein, who maintains a web page for it at
http://cr.yp.to/daemontools.html.

B.5. *qmailanalog*

qmailanalog processes *qmail*'s log file and produces a series of reports that tell one 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.

As a bonus, the `matchup` program combines *qmail*'s multiple log lines per delivery into one--not unlike the familiar *Sendmail* logs.

The source for *qmailanalog* is available from http://cr.yp.to/software/qmailanalog-0.70.tar.gz.

qmailanalog was written by Dan Bernstein, who maintains a web page for it at
http://cr.yp.to/qmailanalog.html.

Note: *qmailanalog* relies on log entry timestamps in the fractional second format used by `accustamp`. In order to use it with logs generated by `multilog`, which are in 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.

B.6. *rblsmtpd*

If you've never been spammed, consider yourself *very* lucky. Most e-mail users are all too familiar with Unsolicited Bulk E-mail (UBE), aka "spam". Most of it is advertisements for sex sites, chain letters, and other scams. Back in the days of old, up until around 1998 or so, most MTA's on the

Internet were *open relays*, i.e., they would accept mail from anyone *for* anyone, even if neither sender nor recipient was local. Spammers use open relays, if they can find any, to deliver their spam. It covers their tracks, redirects the backlash toward the "innocent" relay site, and saves them lots of CPU time and network bandwidth.

Such open relays are considered **very** bad form these days, and several anti-spam vigilante groups have created a mechanism for identifying open relays and other common sources of spam so they can avoid accepting SMTP connections from them.

rblsmtpd is an RBL SMTP Daemon. It sits between `tcpserver` and `qmail-smtpd` and rejects connections from systems identified on one of these lists.

For example, to run `rblsmtpd` under `tcpserver`, try something like:

```
#!/bin/sh
QMAILDUID=`id -u qmaild`
NOFILESGID=`id -g qmaild`
MAXSMTPD=`cat /var/qmail/control/concurrencyincoming`
exec /usr/local/bin/softlimit -m 2000000 \
  /usr/local/bin/tcpserver -v -R -H -l 0 -x /etc/tcp.smtp.cdb -c "$MAXSMTPD" \
    -u "$QMAILDUID" -g "$NOFILESGID" 0 smtp /usr/local/bin/rblsmtpd\
    -r relays.ordb.org /var/qmail/bin/qmail-smtpd 2>&1
```

rblsmtpd was previously available as a separate utility, but is now bundled with ucspi-tcp.

rblsmtpd was written by Dan Bernstein, who maintains a web page for it at http://cr.yp.to/ucspi-tcp/rblsmtpd.html.

Charles Cazabon has a patch that removes the default RBL hardcoded into `rblsmtpd` since it's no longer free. The patch is available from http://pyropus.ca/software/misc/rblsmtpd-nodefaultrbl.patch.

B.7. *serialmail*

qmail was designed for systems with full time, high speed connectivity. *serialmail* is a set of tools that make *qmail* better suited to intermittent, low speed connectivity. With *serialmail* on such a system, *qmail* is configured to deliver all remote mail to a single maildir. The *serialmail* `maildirsmtp` command is used to upload the maildir to the ISP's mail hub when the connection is brought up. If the ISP supports QMTP (see QMTP under Advanced Topics), `maildirqmtp` can also be used.

serialmail can be used on the ISP side of the connection to implement *AutoTURN*: an SMTP connection by a client causes the server to initiate a connection back to the client for sending messages queued on the server for the client. This is similar to the ETRN SMTP function.

The source for *serialmail* is available from http://cr.yp.to/software/serialmail-0.75.tar.gz.

serialmail was written by Dan Bernstein, who maintains a web page for it at http://cr.yp.to/serialmail.html.

B.8. *mess822*

mess822 is a library and set of applications for parsing RFC 822 compliant mail messages. The applications include:

- `ofmipd`: a daemon that accepts messages from clients and rewrites From fields based on a database.
- `new-inject`: a `qmail-inject` replacement that supports user-controlled hostname rewriting.
- `iftocc`: a `.qmail` utility for checking whether a message was sent to a specific address.
- `822header`, `822field`, `822date`, and `822received`: extract information from a message.
- `822print`: pretty-prints a message.

The source for *mess822* is available from http://cr.yp.to/software/mess822-0.58.tar.gz.

mess822 was written by Dan Bernstein, who maintains a web page for it at http://cr.yp.to/mess822.html.

B.9. *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* see Mailing List Managers under Advanced Topics.

The source for *ezmlm* is available from http://cr.yp.to/software/ezmlm-0.53.tar.gz.

ezmlm was written by Dan Bernstein, who maintains a web page for it at http://cr.yp.to/ezmlm.html.

Fred Lindberg and Fred B. Ringel have developed an extension to *ezmlm* called *ezmlm-idx*. It adds lots of useful features and is highly recommended. It's now being maintained by Bruce Guenter available from http://www.ezmlm.org/.

B.10. *safecat*

safecat reliably writes a file into a maildir mailbox. It is particularly useful for filing messages in *procmail* recipes. For example, the following recipe files all messages in `Maildir`:

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

safecat was written by Len Budney, who maintains a web page for it at http://jeenyus.net/~budney/linux/software/safecat.html.

B.11. *djbdns*

djbdns is a DNS server written by the author of *qmail*. It includes `tinydns`, a DNS content server, and `dnscache`, a caching DNS server.

The official web page for *djbdns* is http://cr.yp.to/djbdns.html.

B.12. *maildrop*

maildrop is a mail filter similar to *procmail*.

maildrop was written by Sam Varshavchik, who maintains a web page for it at
http://www.courier-mta.org/maildrop/.

B.13. *syncdir*

syncdir is small library that makes the `link()` system call synchronous. This is necessary when using *qmail* with the queue on a filesystem that doesn't perform `link()` synchronously, such as Linux's ext2fs, Reiserfs, SGI's XFS, and BSD FFS with softupdates.

syncdir was written br Bruce Guenter and is available from http://untroubled.org/syncdir/. Installation instructions are available from http://www.ornl.gov/lists/mailing-lists/qmail/2001/12/msg00949.html.

C. How Internet Mail Works

C.1. How a message gets from point A to point B

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

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 like *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:

```
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 like *Date* and *Message-Id* and modify the values Alice entered (e.g., replace `bob@beta` with "`Bob <bob@beta.example.com>`". Next, the MUA *injects* the message into the mail system. There are two ways to this: 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* method is a de facto standard. With this method, the MUA can put the header and body in a file, separated by a blank line, and pass the file to the `sendmail` program.

4. 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.

5. 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`, to notify her of the problem.

6. 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 *mail delivery agent* (MDA) like `/bin/mail` or `procmail`.

7. 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`.

8. If the delivery succeeds, the message waits in Bob's mailbox until his MUA reads it and displays it.

C.2. More information

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

- Internet mail, by the author of *qmail*. http://cr.yp.to/im.html
- SMTP, by the author of *qmail*. http://cr.yp.to/smtp.html
- Internet mail message header format, by the author of *qmail*. http://cr.yp.to/immhf.html

C.2.1. Internet RFC's

Internet Requests for Comment (RFC's) are the official documentation of the Internet. Most of these are well beyond the commentary stage, and define Internet protocols such as TCP, 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 RFC's is available from the Internet Mail Consortium at http://www.imc.org/mail-standards.html.

D. Architecture

D.1. Modular system architecture

Internet MTA's perform a variety of tasks. Earlier designs like *Sendmail* and *smail* are *monolithic*. In other words, they have one large, complex program that "switches hats": it puts on one hat to be an SMTP server, another to be an SMTP client, another to inject messages locally, another to manage the queue, etc.

qmail is *modular*. Each of these functions is performed by a separate program. As a result, the programs are much smaller, simpler, and less likely to contain functional or security bugs. To further enhance security, *qmail*'s modules run with different privileges, and they don't "trust" each other: they don't assume the other modules always do only what they're supposed to do.

The core modules are:

Modules	**Function**
`qmail-smtpd`	accepts/rejects messages via SMTP
`qmail-inject`	injects messages locally
`qmail-rspawn/qmail-remote`	handles remote deliveries
`qmail-lspawn/qmail-local`	handles local deliveries
`qmail-send`	processes the queue
`qmail-clean`	cleans the queue

There's also a down side to the modular approach. Unlike a monolithic MTA, the interactions between modules are well-defined, and modules only exchange the minimum necessary information with each other. This is generally A Good Thing, but sometimes it makes it hard to do things. For example, the `sendmail` "-v" flag causes *Sendmail* to print a trace of its actions to standard output for debugging purposes. Since the one `sendmail` binary handles injection, queueing, alias processing, `.forward` file processing, and remote forwarding via SMTP, it is able to easily trace the entire delivery until the message is delivered. The equivalent capability in *qmail* doesn't exist, and would require substantial code changes and additional complexity to implement the passing of the "debug" flag from module to module.

D.2. File structure

`/var/qmail` is the root of the *qmail* file structure. This can be changed when *qmail* is being built, but it's a good idea to leave it unchanged so other administrators know where to find things. If you really want to relocate some or all of the *qmail* tree, it's better to do that using symbolic links. See the Create directories subsection of the Installation section for details.

The top-level subdirectories are:

Directory	**Contents**
`alias`	`.qmail` files for system-wide aliases
`bin`	program binaries and scripts
`boot`	startup scripts
`control`	configuration files
`doc`	documentation (except man pages)

`man`	man pages
`queue`	the queue of unsent messages
`users`	the `qmail-users` database files

D.3. Queue structure

The file `INTERNALS` in the build directory discusses the details of queueing more thoroughly. This is a broader overview of structure of the queue.

Subdirectory	Contents
bounce	permanent delivery errors
info*	envelope sender addresses
intd	envelopes under construction by `qmail-queue`
local*	local envelope recipient addresses
lock	lock files
mess*	message files
pid	used by `qmail-queue` to acquire an i-node number
remote*	remote envelope recipient addresses
todo	complete envelopes

Note: Directories marked with an "*" contain a series of *split* subdirectories named "0", "1", ..., up to (*conf-split*-1), where *conf-split* is a compile-time configuration setting contained in the file `conf-split` in the build directory. It defaults to 23. The purpose of splitting these directories is to reduce the number of files in a single directory on very busy servers. *conf-split* must be a prime number.

Files under the `mess` subdirectory are named after their i-node number. What this means is that you can't manually move them using standard UNIX utilities like `mv`, `dump/restore`, and `tar`. There are a couple user-contributed utilities on http://www.qmail.org/ that will rename queue files correctly.

Note: It is not safe to modify queue files while *qmail* is running. If you want to modify the queue, stop *qmail* first, play with the queue *carefully*, then restart *qmail*.

D.4. Pictures

There is a series of files in `/var/qmail/doc` with names starting with `PIC`. These are textual "pictures" of various situations that *qmail* handles. They show the flow of control through the various modules, and are very helpful for debugging and creating complex configurations.

Filename	Scenario
PIC.local2alias	locally-injected message delivered to a local alias
PIC.local2ext	locally-injected message delivered to an extension address
PIC.local2local	locally-injected message delivered to a local user
PIC.local2rem	locally-injected message delivered to a remote address
PIC.local2virt	locally-injected message delivered to an address on a local virtual domain
PIC.nullclient	a message injected on a null client
PIC.relaybad	a failed attempt to use the local host as a relay

PIC.relaygood	a successful attempt to use the local host as a relay
PIC.rem2local	a message received via SMTP for a local user

These files are also available on-line from:

* http://www.qmail.org/man/index.html

If you want *real* pictures of *qmail*, check out Andre Opperman's "big qmail picture" at http://www.nrg4u.com/.

E. Infrequently Asked Questions

These are questions that don't qualify as *frequently* asked, but which are important and not easy to answer.

E.1. How frequently does qmail try to send deferred messages?

Each message has its own retry schedule. The longer a message remains undeliverable, the less frequently *qmail* tries to send it. The retry schedule is not configurable. The following table shows the retry schedule for a message that's undeliverable to a remote recipient until it bounces. Local messages use a similar, but more frequent, schedule.

Delivery Attempt	Seconds	D-HH:MM:SS
1	0	0-00:00:00
2	400	0-00:06:40
3	1600	0-00:26:40
4	3600	0-01:00:00
5	6400	0-01:46:40
6	10000	0-02:46:40
7	14400	0-04:00:00
8	19600	0-05:26:40
9	25600	0-07:06:40
10	32400	0-09:00:00
11	40000	0-11:06:40
12	48400	0-13:26:40
13	57600	0-16:00:00
14	67600	0-18:46:40
15	78400	0-21:46:40
16	90000	1-01:00:00
17	102400	1-04:26:40
18	115600	1-08:06:40
19	129600	1-12:00:00
20	144400	1-16:06:40
21	160000	1-20:26:40
22	176400	2-01:00:00
23	193600	2-05:46:40
24	211600	2-10:46:40
25	230400	2-16:00:00
26	250000	2-21:26:40
27	270400	3-03:06:40
28	291600	3-09:00:00
29	313600	3-15:06:40
30	336400	3-21:26:40

31	360000	4-04:00:00
32	384400	4-10:46:40
33	409600	4-17:46:40
34	435600	5-01:00:00
35	462400	5-08:26:40
36	490000	5-16:06:40
37	518400	6-00:00:00
38	547600	6-08:06:40
39	577600	6-16:26:40
40	608400	7-01:00:00

E.2. Why can't I send mail to a large site with lots of MX's?

If you're getting:

```
deferral: CNAME_lookup_failed_temporarily._(#4.4.3)/
```

The problem might be that *qmail* can't handle large name server query responses. The fix is to install a patch or workaround. See Patches under Advanced Topics.

There's also a question as to why some people *don't* have trouble reaching such systems. Basically, depending on the timing and ordering of queries made to your local nameserver, the size of the response to an `ANY` query for "aol.com" may be larger than the 512 byte limit of a UDP packet, or it may not.

"May not" is likely to happen if the A and MX records time out, but the NS records don't. Since the .COM servers set a 2 day TTL on those, but AOL sets a 1 hour TTL on their records, this will often happen on less busy nameservers. Busier nameservers are more likely to have those records in their cache at any given time, frustrating an unpatched *qmail*'s attempts to check for CNAMEs.

A better test is to send mail to `nosuchuser@large-mx.ckdhr.com`; if it clears your queue and winds up bouncing from ckdhr.com, your MTA can send mail to hosts with MX lists that exceed 512 bytes. (By using a single RRset, with a single TTL, that exceeds 512 bytes, the problem can be seen without depending on the timing and ordering of other queries.)

E.3. What is `QUEUE_EXTRA`?

`QUEUE_EXTRA` is a compile-time configuration variable that specifies an additional recipient that will be added to every delivery. This is used primarily for logging. E.g., the FAQ describes how to use `QUEUE_EXTRA` to keep copies of all incoming and outgoing messages.

To use `QUEUE_EXTRA`, edit `extra.h` specifying the additional recipient in the format `"T`*`recipient`*`\0"`, and the length of the `QUEUE_EXTRA` string in `QUEUE_EXTRALEN` (the "\0" counts as one character). For example:

```
#define QUEUE_EXTRA "Tlog\0"
#define QUEUE_EXTRALEN 5
```

Shut down *qmail* if it's running. If you installed the `qmailctl` script from the Installation section, that can be done by:

```
qmailctl stop
```

If you don't have the `qmailctl` script, you should use your startup/shutdown script or send `qmail-send` a `TERM` signal.

Then rebuild *qmail* using:

```
make setup check
```

Populate `~alias/.qmail-log` with whatever logging you want. E.g., to log Message-ID's:

```
| awk '/^$/ { exit } /^[mM][eE][sS][sS][aA][gG][eE]-/ { print }'
```

Finally, restart *qmail*.

F. Error Messages

qmail error messages and what they mean.

See RFC 1893 for an explanation of the error codes in parentheses.

This appendix is incomplete.

G. Gotchas

These "gotchas" frequently cause problems for *qmail* newbies.

G.1. *qmail* doesn't deliver mail to superusers.

To prevent the possibility of `qmail-local` running commands as a privileged user, *qmail* ignores all users whose UID is 0. 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 `~alias/.qmail-root`.

G.2. *qmail* doesn't deliver mail to users who don't own their home directory.

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

G.3. *qmail* doesn't deliver mail to users whose usernames contain uppercase letters.

qmail converts the entire "local part"--everything left of 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.

G.4. *qmail* replaces dots (.) in extension addresses with colons (:).

Another security feature. The purpose is prevent extension addresses from backing up the file tree using "..". By replacing them with colons, *qmail* ensures that all `.qmail` files for a user are under their home directory. Documented in the `dot-qmail` man page.

G.5. *qmail* converts uppercase characters in extension addresses to lowercase.

This is another result of the fact that *qmail* lowercases the entire local part of addresses. Documented in the `dot-qmail` man page.

G.6. *qmail* doesn't use `/etc/hosts`.

qmail **never** uses `/etc/hosts` to determine the 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, though. Hosts in control files can be specified by IP address by enclosing them in square brackets ([]), e.g.:

```
[10.1.2.219]
```

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

G.7. *qmail* doesn't log SMTP activity.

For a number of 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. The procedure is documented in the FAQ at http://cr.yp.to/qmail/faq/servers.html#recordio.

G.8. *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. An undeliverable message will only be returned to the originator after it spends `queuelifetime` in the queue.

G.9. *qmail* is slow if `/var/qmail/queue/lock/trigger` is gone/has the wrong permissions/is a regular file.

`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 a half hour or so.

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

```
# 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 (says it's a named pipe), the mode (especially world writable), and the owner/group.

G.10. DNS or IDENT lookups can make SMTP slow.

If `qmail-smtpd` is slow to respond to connections, the problem is probably due to DNS reverse lookups or IDENT lookups. If you're starting `qmail-smtpd` with `tcpserver`, remove the "-h", "-p", and "-r" options and add "-H", "-P", "-R", and "-l *hostname*".

See the `tcpserver` documentation at http://cr.yp.to/ucspi-tcp/tcpserver.html for an explanation of these options.

G.11. Carriage Return/Linefeed (CRLF) line breaks don't work.

`qmail-inject` and other local injection mechanisms like `sendmail` don't work right when messages are injected with DOS-style carriage return/linefeed (CRLF) line breaks. Unlike *Sendmail*, *qmail* requires locally-injected messages to use Unix newlines (LF only). This is a common problem with PHP scripts.

G.12. `qmail-send` or `tcpserver` stop working if logs back up.

If you're logging to a supervised log service, as described in section 2, and the log service fails for any reason: disk full, typo in the `run` script, log directory configuration error, etc., the pipeline will eventually fill up, causing the service to block, or hang. Fix the problem (see Troubleshooting) and everything will return to normal.

G.13. `qmail-smtpd` doesn't validate the local part of an address.

If `example.com` is listed in `control/rcpthosts`, mail to *`anything`*`@example.com` will be accepted during the SMTP session. If *`anything`* isn't a valid user or alias, *qmail* will send a bounce message to the envelope sender address.

Some simpleminded relaying tests assume that if a message is accepted, it will be delivered. That's wrong. If someone claims that your system is an open relay, demand to see a copy of message relayed through it--including the complete header, especially the Received fields--then compare them with your logs.

See the Rejecting Invalid Recipients During SMTP Dialogue section for information about ways to add recipient validation to *qmail*.

G.14. Firewalls can block remote access to your SMTP/POP3/IMAP server.

If you've installed an SMTP, POP3, or IMAP server, and you can connect to it from the local host or a host on the local network, but not from a remote host, a firewall might be the problem.

The first place to look is on the server itself. Red Hat Linux, for example, blocks SMTP in the default configuration using `iptables`. Other packet filtering mechanisms such as `ipchains` may also be responsible.

It's also possible that your Internet Service Provider (ISP) blocks certain ports to prevent spamming or enforce their Terms of Service (TOS). Contact your ISP's tech support *after* ensuring that packet filtering isn't responsible and that running a server doesn't violate your TOS.

G.15. `qmail-inject` sets `From` field to `anonymous` if `USER` and `LOGNAME` aren't set.

If a message sent via `qmail-inject` doesn't contain a `From` field, `qmail-inject` looks for environment variables to tell it which user is sending the message. The variables it looks for, in order, are: `QMAILUSER`, `MAILUSER`, `USER`, and `LOGNAME`,

Normal user login sessions usually set one or both of `USER` and `LOGNAME`, but some batch jobs, such as those started by `cron` might not have either of these set.

To cause your `cron` jobs to have a valid `From` field, set one these environment variables before sending any mail messages.

G.16. `qmail-send` doesn't always exit immediately when killed.

Sending `qmail-send` a `TERM` signal doesn't cause it to exit immediately if there are deliveries in progress. `qmail-send` will wait for all `qmail-local` and `qmail-remote` processes to finish before it exits so it can record the results of these deliveries. Because of this, "`qmailctl restart`" or "`qmailctl stop`" might report that `qmail-send` has been stopped, even though it's still running. Always run "`qmailctl stat`" to verify that the stop or restart has actually completed.

Also note that `qmail-send` makes a pass through the queue before exiting, so with very large queues this can cause a noticeable delay.

G.17. Delivering to `/dev/null` doesn't throw messages away.

A delivery instruction like:

```
/dev/null
```

Causes *qmail* to think that `/dev/null` is an mbox mailbox, but since `/dev/null` is a special file, *qmail* can't deliver to it successfully.

The best was to throw messages away is to create a `.qmail` file that contains no valid delivery instructions *but* isn't empty. (Empty `.qmail` files are treated as if they contain the default delivery instructions specified in `defaultdelivery` or on the `qmail-start` command line.) This is accomplished by populating the file with nothing but comments.

For example, a `.qmail` file containing only:

```
#
```

or:

```
# throw messages away undelivered
```

will efficiently throw messages away without delivering them.

G.18. Modifying the queue while `qmail-send` is running is *dangerous*.

Modifying any of the files or directories under `/var/qmail/queue` while `qmail-send` is running without knowing **exactly** what you're doing is likely to result in a corrupt queue--e.g., messages in an undefined state, bizarre error messages in the logs, duplicate deliveries, bogus bounces, etc. Once this happens, you'll have to find and run a queue checking utility (there are a couple listed on qmail.org) or create a new, empty queue.

If you want to modify the queue, stop *qmail* first, play with the queue carefully, then restart *qmail*. Note that corruption is still possible with `qmail-send` stopped, so you still have to know what you're doing.

H. Frequently Asked Questions about Life with qmail

H.1. What version is Life with qmail?

This is LWQ version 2007-11-30.

H.2. Who owns Life with qmail?

Life with qmail is Copyright 1999-2007 David E. Sill

http://Web.InfoAve.Net/~dsill/dave/

H.3. How is Life with qmail licensed?

Life with qmail is covered by the OpenContent License, version 1.0. See http://www.opencontent.org/opl.shtml for the full license. Basically, you can copy, redistribute, or modify Life with qmail provided that modified versions, if redistributed, are also covered by the OpenContent License.

H.4. How can I be notified when new releases of LWQ are made available?

Join the `lwq-announce` mailing list by sending a message to lwq-announce-subscribe@sws1.ctd.ornl.gov.

H.5. Where can LWQ contributors and fans talk about it?

Join the `lwq` mailing list by sending a message to lwq-subscribe@sws1.ctd.ornl.gov.

H.6. Has Life with qmail been translated to *language*?

Maybe. LWQ has been translated into a few languages. See http://lifewithqmail.org/trans.html for more information about LWQ translations.

H.7. Is Life with qmail available in PostScript, PDF, plain text, or any other format beside HTML?

Yes, alternative formats can be found at http://lifewithqmail.org/.

H.8. I used Life with qmail and it crashed my system/erased my hard disk/ruined my love life/killed my dog/etc.

I'm sorry. Really sorry. But Life with qmail comes with no warranty. See the OpenContent License mentioned above. I didn't get paid to write it, I just wanted to contribute something useful to the *qmail* community.

Actually, this isn't a FAQ. In fact, I hope it's a NAQ (Never Asked Question).

H.9. How can I contribute to LWQ?

Please send corrections, suggestions, complaints, etc. to lwq@sill.org.

If you'd like to make a larger contribution, such as a new subsection or appendix, that's great! You might want to check with me first to make sure the topic is something I want to cover in LWQ and that nobody else is already working on it.

Another way to support LWQ is to shop at my bookstore, in association with Amazon.com, using this link: http://www.amazon.com/exec/obidos/redirect-home/davesill.

Thanks for your support!

H.10. What's changed in this version of LWQ?

- Updated for netqmail 1.06.
- Added info for Wheeler book.
- Fixed various typos.
- Fixed some dead links.

H.10.1. What changed in the 2006-01-02 version of LWQ?

- Lots of minor tweaks, fixed dead links, etc.
- /dev/null gotcha added. Thanks to Payal Rathod.
- Expanded `nosuid` warning. Thanks to Tony Hansmann.
- Added link to Cazabon's nodefaultrbl patch. Thanks to Jeremy Kister.
- Added mkdirs to qmail-pop3d instructions. Thanks to Larry Weldon.
- Added note about errno patch for checkpasswd. Thanks to tlad.
- Added smarthost relay subsection. Thanks to the Silver Dirk.
- Improved Qmail-Scanner wording. Thanks to Jason Haar.
- Added a section for Yenigul's book. Thanks to Ismail Yenigul.
- Added abuse@ alias. Thanks to Charles Cazabon.
- Added ucspi-tls section. Thanks to Scott Gifford.
- Added queue modification gotcha.

H.10.2. What changed in the 2004-06-30 version of LWQ?

- Incorporated new *qmail* logo. Thanks to Michael Kadrie.
- Fixed various typos.
- Updated Macintosh OS X note.
- Updated ClamAV link.
- Added recipient validation section to Advanced Topics.
- Fixed "setuid/exe required for queue filesystem" wording.

H.10.3. What changed in the 2004-03-28 version of LWQ?

- Updated info for Levine book.
- Fixed typo in "What changed in the 2004-03-01 version of LWQ?".
- Added comment in G.16 about `qmail-send` scanning the queue before exiting.

H.10.4. What changed in the 2004-03-01 version of LWQ?

- G.16 said "KILL", should have been "TERM".
- Added softlimit-too-low example error message.
- Fixed E.2, CNAME lookup problem.

H.10.5. What changed in the 2004-01-26 version of LWQ?

- Updated for netqmail 1.05.
- Fixed minor typos.

H.10.6. What changed in the 2003-11-10 version of LWQ?

- Fixed minor typos.

H.10.7. What changed in the 2003-10-30 version of LWQ?

- Adjusted installation for *netqmail* distribution.
- Updated Binc IMAP entry.

H.10.8. What changed in the 2003-08-16 version of LWQ?

- Added this change log section. Thanks to Jerry Asher for requesting it.
- Added note about noexec and nosuid to system requirements section.
- Added link to inst_check and examples to the testing section.
- Added link to Mate Wierdl's errno patches to the patch section.
- Added OS X note to system requirements section.
- Added Cyrus entry to IMAP server section. Thanks to Rick Updegrove.
- Added Dovecot entry to IMAP server section.
- Reworded part of the entension section. Thanks to Adrian Ho.
- Reformatted the POP run scripts for easier copying and pasting. Thanks to Woody Preston.
- Fixed a typo in the section on Chuck Foster's DNS patch. Thanks to Josh Parreco.
- Fixed a couple typos in the QMTP section. Thanks to Marek Gutkowski.