CHAPTER 5

Configuring sendmail and DNS

Many regard configuring sendmail as a daunting task, which seems true if you stare at the long list of raw, supported features and consider the sheer number of configuration options. However, the average administrator simply won't need the majority of those features, and much of the configuration need not be changed. For instance, you can configure sendmail to work with UUCP mail gateways. However, since UUCP is no longer used, you can disregard these options. When you start looking at sendmail from this point of view, it isn't as daunting as you might think at first.

In this chapter, I will cover the basic configuration necessary to bring your MTA listening live on the network for the first time. I will also introduce the primary sendmail configuration files and a few DNS concepts that are necessary for others to route mail to your e-mail server once it is online. Without the proper DNS configuration, e-mail will never make it to your Internet domain.

Introducing the sendmail Configuration Files

The primary configuration files for sendmail are found in the directory `/etc/mail/`, unless you explicitly changed the configuration directory when you installed sendmail from source code. A sample directory listing from our default Fedora Core installation ought to look something like this:

```
[curtis@mail ~]$ cd /etc/mail/
[curtis@mail mail]$ ls -l
```

```
total 188
-rw-r--r--  1 root root   331 May  6 08:35 access
-rw-r-----  1 root root 12288 Aug  8 20:28 access.db
-rw-r--r--  1 root root     0 May  6 08:35 domaintable
-rw-r-----  1 root root 12288 Aug  8 20:28 domaintable.db
-rw-r--r--  1 root root  5588 May  6 08:35 helpfile
-rw-r--r--  1 root root    64 May  6 08:35 local-host-names
-rw-r--r--  1 root root     0 May  6 08:35 mailertable
-rw-r-----  1 root root 12288 Aug  8 20:28 mailertable.db
-rw-r--r--  1 root root  1035 May  6 08:35 Makefile
-rw-r--r--  1 root root 58079 Aug  8 20:28 sendmail.cf
-rw-r--r--  1 root root  7069 May  6 08:35 sendmail.mc
```

```
drwxr-xr-x  2 root root  4096 Aug  8 10:06 spamassassin
-r--r--r--  1 root root 41348 May  6 08:35 submit.cf
-rw-r--r--  1 root root   952 May  6 08:35 submit.mc
-rw-r--r--  1 root root   127 May  6 08:35 trusted-users
-rw-r--r--  1 root root     0 May  6 08:35 virtusertable
-rw-r-----  1 root root 12288 Aug  8 20:28 virtusertable.db
```

In Fedora Core, the sendmail RPM package installs these files, and the sendmail program is preconfigured to listen for and accept mail from the local computer only. This means that sendmail is running and will deliver e-mail locally, but it will not listen on any public network interfaces or receive e-mail from other network nodes. The main configuration files for sendmail are `sendmail.cf` and `submit.cf`; we will discuss these two files and their corresponding files, `sendmail.mc` and `submit.mc`, in depth later in this chapter. All of the other sendmail files in `/etc/mail/` are configuration files included by `sendmail.cf` meant to manipulate other various sendmail features, which are discussed in further detail in Chapter 6. For now, ignore the directory `spamassassin` altogether, if it exists; SpamAssassin and the role of `/etc/mail/spamassassin/` are topics in Chapter 18.

Later in this chapter, when we configure sendmail, you will see that the main `sendmail.cf` configuration file actually depends on a number of other sendmail configuration files, which contain features and operating system–specific configuration directives necessary for sendmail to run correctly on your system. Luckily, even though they are explicitly necessary for reconfiguring sendmail, it is unlikely that you will have to touch any of them at all.

A Word About sendmail Security

The sendmail program has to run as the privileged root user for several reasons. One primary reason is that only applications run by the root user may bind to an address and port for sending and receiving connections over the network. Also, root privileges are needed for read/write permissions to all of the queue directories; on a multiuser system, you wouldn't want to make all users' mail queues readable by unprivileged users. The inability of sendmail to run as anything other than the root user has been a major criticism of sendmail over the years, because of the potential security risks of running solely as the privileged root user.

sendmail.cf vs. submit.cf

Beginning with version 8.12 of sendmail, this issue has been addressed. Unfortunately, there is no way around the necessity to run sendmail as the root user to open a network address and port, but the queue problem has been addressed. The solution involves splitting sendmail into two separate processes. One is used for receiving e-mail only, and the other is for e-mail delivery. The sendmail process used for receiving e-mail still runs as the root user and has its own queue (`/var/spool/queue/`) that it, and only it, has access to read from and write to. The second sendmail process runs using a nonprivileged user (typically `smmsp`) and owns another, separate queue (`/var/spool/clientmqueue/`) that it can read from and write to, and that other nonprivileged users can execute in a secure manner through a command-line application. The former receives mail from other e-mail systems over the network and uses the normal `sendmail.cf` configuration file, and the latter is for local mail submission and delivery and uses the `submit.cf` configuration file.

On any system that makes use of this split configuration, like Fedora Core, two different sendmail processes can be seen working in the two modes of operation with a combination of the `ps` and `grep` commands:

```
[curtis@mail ~]$ ps auxw | grep sendmail
```

```
root     13390  0.0  0.6   7952  3176 ?        Ss   11:26   0:00 ➥
sendmail: accepting connections
smmsp    13396  0.0  0.5   6968  2668 ?        Ss   11:26   0:00 ➥
sendmail: Queue runner@01:00:00 for /var/spool/clientmqueue
```

Note the first column (in bold) of the output of the command `ps`, which shows the input from the user running the corresponding process.

This security matter is of relevance to you only if you decide to allow interactive login accounts for your users, from which, they may send e-mail to and receive e-mail from traditional UNIX command-line mail clients like mail, Mutt, or Pine; this is not as common anymore, since most people check their e-mail remotely from their desktop computers. The default behavior is to treat local mail essentially like incoming mail from the public network, passing messages to the local delivery sendmail application. If you install the Fedora Core `sendmail-doc` RPM package, you can read more about this in the file `/usr/share/doc/sendmail/SECURITY`; otherwise, it's available online at `www.sendmail.org` or in the source distribution. Unless you have special requirements, you should not have to change `/etc/mail/submit.cf` at all. However, you will need to make changes to `sendmail.cf`, so let's take a look at sendmail's default configuration and learn how to customize sendmail for your specific needs.

The Default Fedora Core sendmail.cf

Again for security reasons, some Linux operating systems like Fedora Core specifically configure sendmail, so that it will only accept incoming connections from the local computer through the loopback network interface. The IP address for the loopback interface is `127.0.0.1` and is commonly referred to by the generic hostname `localhost`. The reasoning behind this is the general acknowledgment by vendors that not everybody who installs a Linux operating system intends to offer public Internet e-mail services, as is the case for most desktops. Yet the ability to send and receive e-mail, even just locally, is necessary to many of the components on a Linux system. Reducing the number of applications accepting incoming connections over the public network limits the avenues miscreants can take advantage of to break into an improperly configured or secured server application that the user neglects or simply isn't aware of.

■ **Note** The loopback network interface is a virtual interface, emulating a physical network card via software, without any real hardware connected to it. The reason for having such a network interface is to allow a system to privately contact its own local network applications, like sendmail, instead of over the public network.

However, in your case, you need sendmail to accept incoming network connections over the public network. (Otherwise, you probably wouldn't be reading this book!) Changing this default configuration behavior, and much more, is covered in the following section of this chapter.

Configuring sendmail

If you installed Fedora Core, I must reemphasize the need for the separate prerequisite `sendmail-cf` RPM package. If that package is not installed, you will not be able to successfully reconfigure sendmail; refer to Chapter 4 for more details. If you installed sendmail from source code, all the necessary support files should be in place as part of the installation process. For the rest of this chapter, I will assume you are working from a Fedora Core system, using the vendor-supplied sendmail packages. Although the configuration file names should be identical between the two installation methods, the actual locations may vary, and you will need to refer to the documentation that came with the source code you downloaded for exact installation directories.

Throughout this chapter, I've said that the default sendmail configuration file is `sendmail.cf`. However, as you might have noted when we looked at the directory listing of `/etc/mail/`, for every file ending with a `.cf` file extension, a file with the same name ending with the `.mc` file extension also exists. When configuring sendmail, you never edit the `.cf` file directly. Indeed, if you just browse `sendmail.cf`, I think you'll quickly see why! To simplify the configuration process, we edit the `.mc` files and use those simpler files to produce the final, more complex `sendmail.cf`. This is done with the macro processor called GNU m4.

Introducing the m4 Macro Processor

Generally speaking, a *macro* is simply a representation, or abbreviation, of a set of commands or strings of text. Macros are used in popular word processors all the time. Executing one macro command can initiate several keystrokes or commands, easing repetitive tasks.

The m4 macro processor is an application that does just as the name implies—it processes macros. To be more precise, m4 copies its input, expands macros as it reads the input, and then presents the results as its output. The original is untouched and a new file with the macros expanded is generated. If the input is modified, and m4 is run again, the output is expanded accordingly.

m4 is *stream-based*, meaning is has no notion of lines; it simply sees its input as a continuous line of characters. Because of this, you will see the special m4 keyword `dnl` throughout the `.mc` files. It stands for "delete through newline." m4 ignores the characters following `dnl` up to, and including, the newline character, effectively removing the string of characters from the output. This can be used to remove redundant empty lines and make the resulting output more readable. It can also be used to insert comments in the input that won't show up in the output. I think a simple example can illustrate this point fairly well.

Type the following text into a temporary text file called something like `test.m4` (a great chance to practice those vi skills!); the file should begin and end with empty lines:

```
See? dnl

dnl
dnl Ignore everything after a "dnl"
dnl

m4 dnl foo dnl
isn't so bad!
```

Now, from the command line in the same directory as the file you just created, use the command `m4` to process the input text by typing the following:

```
[curtis@mail ~]$ m4 test.m4
```

```
See?

m4 isn't so bad!
```

It almost seems like magic!

But the real power is in macros. Recall that a macro is simply a short, meaningful representation for a set of elaborate, complex strings of characters or commands. For instance, if you find that you are typing `/usr/share/sendmail-cf/` a lot, why not define a macro to replace every occurrence of the string `cf` with the longer string `/usr/share/sendmail-cf/`? There are some built-in macros that m4 automatically recognizes and processes, but you can create your own macros with the m4 directive `define`. For example, `define(`foo',`bar')` creates the macro named `foo` with the value `bar`. Every instance of `foo` in the input will be replaced by `bar` in the output. Let's edit `test.m4` and change it accordingly (changes are noted in bold):

```
See? dnl

dnl
dnl Ignore everything after a "dnl"
dnl
define(`bad', `hard')
m4 dnl foo dnl
isn't so bad!
```

Note that the syntax of the `define` directive is a backtick (`` ` ``), a text string, and a single quote (`'`) enclosed in parenthesis.

Process the new input text with m4, and compare the output to the previous example:

```
[curtis@mail ~]$ m4 test.m4
```

```
See?

m4 isn't so hard!
```

Even though `sendmail.cf` is the primary sendmail configuration file, you don't edit it directly. Instead, edit `sendmail.mc`, and use the m4 macro processor to generate a new `sendmail.cf`. Let's look at how to do that next.

Editing sendmail.mc

The first thing I suggest, before making any major changes to any configuration file at all, is to back up the original. So, let's back up `sendmail.mc`:

```
[curtis@mail ~]$ cd /etc/mail/
[curtis@mail mail]$ sudo cp sendmail.mc sendmail.mc.orig
[curtis@mail mail]$ ls -l sendmail.mc sendmail.mc.orig
```

```
-rw-r--r--  1 root root 7069 May  6 08:35 sendmail.mc
-rw-r--r--  1 root root 7069 Aug 17 20:07 sendmail.mc.orig
```

Now it's time to start editing `sendmail.mc`. Open the sendmail macro configuration file in your text editor of choice with sudo. (Aren't you glad you practiced hard with vi?) The first thing you should notice is that this file is heavily commented, so if you ever have to make changes to an existing option not covered in this book, basic information is at your fingertips. I won't go through the entire file line by line. Rather, we'll walk through the few options that are relevant to the goals of this book and to getting sendmail ready to accept SMTP connections from the public network.

The six basic macros and groups of macros follow:

- `VERSIONID`: Macro to insert optional versioning information into `sendmail.cf`
- `OSTYPE`: Macro to define the operating system sendmail is running on
- `FEATURE`: Macro to turn on special built-in features, like procmail delivery
- *Local macro definitions*: Macros defined to customize features
- `DAEMON_OPTIONS`: Macro to define options fed to the sendmail daemon, such as the network port to listen on
- `MAILER`: Macro used to define the protocols or program used to receive and deliver e-mail

In general, these rules, or groups of rules, should appear in `sendmail.mc` in the order listed, the exception being any local macro definitions that affect a particular `FEATURE` definition. In that case, the local macro definition should appear before the `FEATURE` definition that it modifies.

Defining the VERSIONID Macro

This macro is actually optional, but I highly recommend using it. It simply inserts simple versioning information into `sendmail.cf`. It can be used to keep track of different revisions to `sendmail.mc`. You can use a revision control system, like CVS or Subversion. Neither of these are covered in this book, but more information can be found in Garrett Rooney's excellent book *Practical Subversion* (Apress, 2004). If you use a revision control system, version tags can be entered here. Otherwise, you can just change this by hand and rebuild `sendmail.cf` each time you make a change. Defining `VERSIONID` is useful, so you can tell whether `sendmail.cf` was successfully rebuilt after making your changes by comparing the version strings before and after. If multiple administrators will be making changes to the sendmail configuration files, this also might be a useful way to track who made changes last. The default set by the Fedora Core team is:

```
VERSIONID(`setup for Red Hat Linux')dnl
```

Using some combination of the date in ISO 8601 format and the administrator's initials might be a useful way of tracking changes and times of changes, so let's change the default setting to something a bit more informative for future reference. I suggest something like the following:

```
VERSIONID(`Config last revised 2005-08-17 cjs')dnl
```

At any rate, the `VERSIONID` can be set to whatever you prefer, or left out completely; it has no bearing on the operation or customization of sendmail itself. This information is not given out publicly, so it is only meaningful to you to track your configuration changes. However, the next macro to be defined is not optional and is crucial to the operation of sendmail.

Defining the OSTYPE Macro

The `OSTYPE` macro defines the operating system environment in which you are configuring sendmail. Defining the operating system is necessary so that sendmail knows how to set certain macro values, like the path to the aliases file, the location of the mail queues, or how sendmail should interact with the operating system. Sendmail supports a vast list of UNIX and UNIX-like BSD and Linux operating systems, each with various idiosyncrasies that need to be taken into account.

If this macro is not defined, `sendmail.mc` will not compile or generate a `sendmail.cf` configuration file at all. The sendmail source comes with many operating system environments already configured and available to be included in your local sendmail configuration. The complete list of available operating system environments can be found in the `ostype` directory of the sendmail source or in `/usr/share/sendmail-cf/ostype/` on a Fedora Core system with the `sendmail-cf` RPM package installed.

It is absolutely necessary to define this macro before any `MAILER` macro definitions (which we will discuss a bit later in this chapter) and to define only one `OSTYPE` in your configuration.

The default set by the Fedora Core development team is:

```
OSTYPE(`linux')dnl
```

and should remain defined as such, since you are, presumably, configuring sendmail for a Linux operating system environment.

The next several definitions you'll come across in the default Fedora Core `sendmail.mc` are various options, features, and local macro definitions used to customize your sendmail configuration.

Looking at sendmail Features, Local Macro Definitions, and Options

The next few macros define various options that affect how sendmail handles mail. The Fedora Core `sendmail.mc` sets some standard settings, and we will not make any changes to those. Some of the options are for advanced functionality that does not need to be altered under most circumstances. Several options are commented out and define how `SMTP AUTH` is configured; we will ignore these options until Chapter 21.

Various special features can be turned on with the `FEATURE` macro. For example, the following line in the default Fedora Core `sendmail.mc` file:

```
FEATURE(`smrsh',`/usr/sbin/smrsh')dnl
```

defines a special security feature for mail delivery, which we will specifically cover in more depth in Chapter 13.

Following FEATURE definitions are local macro definitions. Again these are simply ways of changing the default behavior of sendmail. If a local macro influences a feature, it should generally occur before the FEATURE definition itself. One local macro definition in the default Fedora Core sendmail.mc that we need to change is the following line:

```
DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```

This configuration line tells sendmail to only listen on the localhost loopback IP interface (127.0.0.1) on port smtp (a setting we discussed earlier in this chapter). Port smtp is a reference to a services string as defined in the file /etc/services. So, in this case, smtp is looked up in /etc/services and replaced with 25 when sendmail starts; port 25 is the well-known, standard port for communication over SMTP.

■ **Note** The file /etc/services is a text file that maps standard service names to port numbers. The service names and appropriate ports were historically defined by RFC 1700 (www.ietf.org/rfc/rfc1700.txt), but RFC 1700 has since been replaced by the Internet Assigned Numbers Authority (IANA) web database found at www.iana.org/assignments/port-numbers. For most well-known ports like SMTP, these assignments did not change, so it's not necessary to update /etc/services.

Because the whole intent of our work here is to run an Internet e-mail server, we want sendmail to listen for incoming connections from the public network. So, go ahead and comment out the previous line by adding dnl to the beginning of it. It should look something like the following line when you're done:

```
dnl DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```

Now insert a new line directly below the option you just commented out, and add the following line:

```
dnl #
dnl # Make sendmail listen on our public Internet IP address.
DAEMON_OPTIONS(`Port=smtp,Addr=192.168.69.4, Name=MTA-Public')dnl
dnl #
```

Replace the IP Address 192.168.69.4 with the public IP address of your server. Save your changes to sendmail.mc without exiting your text editor. Now let's explore the final primary class of sendmail configuration macros.

Defining MAILER Macros for E-mail Delivery

The MAILER macros should be the very last set of macro definitions. Different mailers are defined to tell sendmail what protocols to use to send and receive e-mail. For example, the local mailer is used to deliver e-mail sent to users hosted on the local server and is always automatically included. As we discussed early in this chapter, the standard protocol for the

exchange of Internet e-mail is SMTP, so you should also define the SMTP mailer. Other mailers, like the UUCP mailer, exist for historical purposes but are very rarely used anymore.

For the purposes of this book, we only need the local and SMTP mailers, which are defined by the following:

```
MAILER(smtp)dnl
MAILER(procmail)dnl
```

These happen to be the defaults in the Fedora Core `sendmail.mc` and should be left as such.

Our first attempt at customizing sendmail is complete. The next thing to do is save your changes, if you haven't done so already, exit your text editor, and compile the `sendmail.cf` with the m4 macro processor.

Compiling sendmail.mc

Now it's time to compile your customized sendmail macro configuration file. It's necessary to recreate a new `sendmail.cf` every time you make changes to `sendmail.mc`. And remember, doing so on a Fedora Core system requires the `sendmail-cf` RPM package to be installed. First, make a backup of your current `sendmail.cf`, in case you need to fall back on a configuration that you know is good should some problem arise. I usually name backup files with the date and my initials, so it's clear when the file was backed up and by whom. Go ahead and back up `sendmail.cf` now, just to be safe:

```
[curtis@mail ~]$ cd /etc/mail/
[curtis@mail mail]$ sudo cp sendmail.cf sendmail.cf.20050817.cjs
[curtis@mail mail]$ ls -l sendmail.cf*
```

```
-rw-r--r--  1 root root 58079 Aug  8 20:28 sendmail.cf
-rw-r--r--  1 root root 58079 Aug 17 14:34 sendmail.cf.20050817.cjs
```

Caution When rebuilding a new `sendmail.cf` from a modified `sendmail.mc`, the existing `sendmail.cf` is overwritten, so I strongly advise you to make backups of important files before proceeding.

Next, to actually rebuild `sendmail.cf` on a Fedora Core system using the `sendmail` and `sendmail-cf` RPM package installation, use the GNU `make` command to perform the rebuild process in one easy step:

```
[curtis@mail mail]$ sudo make -C /etc/mail
```

```
make: Entering directory `/etc/mail'
make: Leaving directory `/etc/mail'
```

The two lines of output from `make` are normal. If you don't get any errors, you successfully rebuilt a new `sendmail.cf` based on your customizations of `sendmail.mc`!

Manually Starting and Stopping sendmail

Now, let's start the newly configured sendmail daemon. This can be achieved with the init script, typically `/etc/init.d/sendmail` or `/etc/rc.d/init.d/sendmail`, depending on which flavor of Linux operating system you've chosen. For example, to start sendmail with the init script, you might try the following command:

```
[curtis@mail mail]$ sudo /etc/init.d/sendmail start
```

or to stop sendmail, use this command:

```
[curtis@mail mail]$ sudo /etc/init.d/sendmail stop
```

On Fedora Core, you can use the shortcut command called `service`, which makes it a little easier to use the init scripts without having to guess or type the whole path to the init script:

```
[curtis@mail mail]$ sudo service sendmail start
```

```
Starting sendmail:                                         [  OK  ]
Starting sm-client:                                        [  OK  ]
```

or to stop sendmail, you can use the following command:

```
[curtis@mail mail]$ sudo service sendmail stop
```

```
Shutting down sendmail:                                    [  OK  ]
Shutting down sm-client:                                   [  OK  ]
```

When you start sendmail, you should see log entries like the following ones indicating successful process start-up in `/var/log/maillog`:

```
Aug 26 22:43:14 mail sendmail[32624]: starting daemon (8.13.4): ➥
SMTP+queueing@01:00:00
Aug 26 22:43:14 mail sm-msp-queue[32630]: starting daemon (8.13.4): ➥
queueing@01:00:00
```

The first line is the sendmail program listening and accepting incoming connections over the public network, and the second line indicates the local client security sendmail program.

Starting sendmail Automatically

On a Fedora Core system using sendmail installed from the RPM package, you can use the command `chkconfig` to configure an application to automatically start up and shut down. By default, sendmail comes preconfigured to always start up when your system boots. You can see this by passing the `--list` argument to `chkconfig`:

```
[curtis@mail ~]$ sudo chkconfig --list sendmail

sendmail        0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

As you can see, sendmail is already configured to automatically start in the appropriate run levels. For a more complete introduction to Linux run levels and Fedora Core init scripts, refer to Chapter 3.

Now that sendmail is started, make sure your system firewall has been configured to pass network traffic to sendmail.

Opening Your Firewall to sendmail

We've already discussed the Linux firewall and opened a hole for the SSH server. We've got to do the same thing for the sendmail server, too. SMTP typically listens on the well-known port TCP 25. On a Fedora Core system, add a line like the following one to `/etc/sysconfig/iptables`:

```
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 25 -j ACCEPT
```

Don't forget to restart the firewall after you've made your changes! Now, you're all set. Since you've successfully reconfigured your sendmail installation and have it running and ready to send and receive e-mail, there's one more thing to configure, so other MTAs can find your server to deliver e-mail for your domain—the DNS.

Configuring DNS for Successful E-mail Delivery

Other Internet e-mail servers have to know how to deliver e-mail destined for your Internet domain to your e-mail server. Everyone is at least familiar with the Internet DNS, even if they don't know it. DNS is the Internet protocol used to translate domain names that are easily recallable for humans, like `www.apress.com`, to their computer-routable IP addresses.

The most common DNS server application is the Internet Systems Consortium's Berkeley Internet Name Domain (BIND) application (`www.isc.org/index.pl?/sw/bind/`). It's possible to run your own BIND server, though the details of doing so are well outside the scope of this book. If you're unfamiliar with BIND and would like to learn more, consider picking up Ron Aitchison's excellent book *Pro DNS and BIND* (Apress, 2005).

The first thing to do is make sure your server itself is configured with a resolvable Internet domain name. This domain name must be assigned to an IP address that is controlled by you or your ISP. This is the same IP address you assigned your server during the installation process, but now you've got to publish a name for that IP address. The domain name that is mapped to an IP address is called a DNS `A` record and looks like this for a BIND configuration:

```
mail.example.com    IN    A    192.168.2.4
```

When a DNS lookup is done on the domain name of your server, it should resolve to the IP address of your server. This is called a *reverse lookup*. It is controlled by the DNS `PTR` record, and looks like this for a BIND configuration:

```
192.168.2.4    IN    PTR    mail.example.com
```

Well, the same system is used for routing e-mail from a client's MTA to another domain's MTA. Specifically, the DNS mail exchange (MX) record type is used to define one or more mail destinations for any given Internet domain. When an e-mail client or MTA looks up the mail destination for a domain, the authoritative DNS server for that domain returns an ordered list.

Introducing the DNS MX Record

MX, short for *mail exchange*, is no more than an Internet host that either forwards e-mail for a domain to a final destination or is the final destination itself. A domain can have more than one MX record if you have more than one mail exchange for your domain, but for the purposes of this book, we'll assume that you have only one mail exchange for simplicity's sake. An example MX record follows:

```
example.com.    IN  MX  10    mail.example.com.
```

The first field is the domain for which this MX record is defined; the second field defines the class to be Internet; the third field indicates this is an MX record; the fourth field defines the preference; and the last field defines the name of the mail exchange. The mail exchange should be defined by the appropriate BIND `A` record, mapping the name to a valid IP address. Strictly speaking, a DNS MX record should not point to a `CNAME`, or alias, record, but pointing to a `CNAME` is common, so most MTAs will accept it.

Testing Your DNS Configuration

To test to make sure your DNS configuration is correct, whether your DNS is hosted by you or your ISP, on most Linux systems, including a Fedora Core system, you find the command `host`. The `host` command can be used to make various DNS queries from the command line.

To make sure that the hostname of your e-mail system resolves to the correct IP address, run the following command from the command line:

```
[curtis@mail ~]$ host mail.example.com
```

```
mail.example.com has address 192.168.2.4
```

To make sure that the reverse lookup on your IP address resolves to the correct hostname, run the following command from the command line:

```
[curtis@mail ~]$ host 192.168.2.4
```

```
4.2.168.192.in-addr.arpa domain name pointer mail.example.com.
```

And finally, to check that your MX record for your e-mail domain is set up correctly, run the following command from the command line:

```
[curtis@mail ~]$ host -t mx example.com
```

```
example.com mail is handled by 10 mail.example.com.
```

Summary

In this chapter, you took a whirlwind tour of the sendmail configuration files. I showed you how sendmail has addressed some local security problems by running a separate sendmail application specially configured for local e-mail submission only. Typically configured with individual configuration files, `sendmail.cf` and `submit.cf`, running sendmail as two separation applications helps achieve the privilege separation discussed in Chapter 3. Next, I introduced the GNU m4 macro processor and showed how you could use it to simplify sendmail configuration. By editing the `sendmail.mc` macro configuration file and running one command, we were able to generate a brand new `sendmail.cf` customized for your specific installation. Without proper DNS MX records, nobody would know where to deliver e-mail destined for your domain. Although a complete discussion of DNS is outside the scope of this book, I gave an example of an MX record for the BIND application, a DNS server almost as venerable as the sendmail program. In Chapter 6, we will continue to build on the foundation we've built so far to further fine-tune and customize sendmail to meet your specific needs.