SENDMAIL

INSTALLATION AND OPERATION GUIDE

Eric Allman
University of California, Berkeley
Mammoth Project
eric@CS.Berkeley.EDU

Version 8.58

For Sendmail Version 8.7

Sendmail implements a general purpose internetwork mail routing facility under the UNIX® operating system. It is not tied to any one transport protocol — its function may be likened to a crossbar switch, relaying messages from one domain into another. In the process, it can do a limited amount of message header editing to put the message into a format that is appropriate for the receiving domain. All of this is done under the control of a configuration file.

Due to the requirements of flexibility for *sendmail*, the configuration file can seem somewhat unapproachable. However, there are only a few basic configurations for most sites, for which standard configuration files have been supplied. Most other configurations can be built by adjusting an existing configuration files incrementally.

Sendmail is based on RFC822 (Internet Mail Format Protocol), RFC821 (Simple Mail Transport Protocol), RFC1123 (Internet Host Requirements), and RFC1651 (SMTP Service Extensions). However, since *sendmail* is designed to work in a wider world, in many cases it can be configured to exceed these protocols. These cases are described herein.

Although *sendmail* is intended to run without the need for monitoring, it has a number of features that may be used to monitor or adjust the operation under unusual circumstances. These features are described.

Section one describes how to do a basic *sendmail* installation. Section two explains the day-to-day information you should know to maintain your mail system. If you have a relatively normal site, these two sections should contain sufficient information for you to install *sendmail* and keep it happy. Section three describes some parameters that may be safely tweaked. Section four has information regarding the command line arguments. Section five contains the nitty-gritty information about the configuration file. This section is for masochists and people who must write their own configuration file. Section six describes configuration that can be done at compile time. Section seven gives a brief description of differences in this version of *sendmail*. The appendixes give a brief but detailed explanation of a number of features not described in the rest of the paper.

WARNING: Several major changes were introduced in version 8.7. You should not attempt to use this document for prior versions of *sendmail*.

1. BASIC INSTALLATION

There are two basic steps to installing *sendmail*. The hard part is to build the configuration table. This is a file that *sendmail* reads when it starts up that describes the mailers it knows about, how to parse addresses, how to rewrite the message header, and the settings of various options. Although the configuration table is quite complex, a configuration can usually be built by adjusting an existing off-the-shelf configuration. The second part is actually doing the installation, i.e., creating the necessary files, etc.

The remainder of this section will describe the installation of *sendmail* assuming you can use one of the existing configurations and that the standard installation parameters are acceptable. All pathnames and examples are given from the root of the *sendmail* subtree, normally */usr/src/usr.sbin/sendmail* on 4.4BSD.

If you are loading this off the tape, continue with the next section. If you have a running binary already on your system, you should probably skip to section 1.2.

1.1. Compiling Sendmail

All *sendmail* source is in the *src* subdirectory. If you are running on a 4.4BSD system, compile by typing ‘‘make’’. On other systems, you may have to make some other adjustments. On most systems, you can do the appropriate compilation by typing

```
sh makesendmail
```

This will leave the binary in an appropriately named subdirectory. It works for multiple object versions compiled out of the same directory.

1.1.1. Tweaking the Makefile

Sendmail supports two different formats for the local (on disk) version of databases, notably the *aliases* database. At least one of these should be defined if at all possible.

NDBM — The ‘‘new DBM’’ format, available on nearly all systems around today. This was the preferred format prior to 4.4BSD. It allows such complex things as multiple databases and closing a currently open database.

NEWDB — The new database package from Berkeley. If you have this, use it. It allows long records, multiple open databases, real in-memory caching, and so forth. You can define this in conjunction with one of the other two; if you do, old databases are read, but when a new database is created it will be in NEWDB format. As a nasty hack, if you have NEWDB, NDBM, and NIS defined, and if the alias file name includes the substring ‘‘/yp/’’, *sendmail* will create both new and old versions of the alias file during a *newalias* command. This is required because the Sun NIS/YP system reads the DBM version of the alias file. It’s ugly as sin, but it works.

If neither of these are defined, *sendmail* reads the alias file into memory on every invocation. This can be slow and should be avoided. There are also several methods for remote database access:

NIS — Sun’s Network Information Services (formerly YP).

NISPLUS — Sun’s NIS+ services.

NETINFO — NeXT’s NetInfo service.

HESIOD — Hesiod service (from Athena).

Other compilation flags are set in conf.h and should be predefined for you unless you are porting to a new environment.

1.1.2. Compilation and installation

After making the local system configuration described above, You should be able to compile and install the system. The script ‘‘makesendmail’’ is the best approach on most systems:

sh makesendmail

This will use *uname*(1) to select the correct Makefile for your environment.

You may be able to install using

sh makesendmail install

This should install the binary in /usr/sbin and create links from /usr/bin/newaliases and /usr/bin/mailq to /usr/sbin/sendmail. On 4.4BSD systems it will also format and install man pages.

1.2. Configuration Files

Sendmail cannot operate without a configuration file. The configuration defines the mail delivery mechanisms understood at this site, how to access them, how to forward email to remote mail systems, and a number of tuning parameters. This configuration file is detailed in the later portion of this document.

The *sendmail* configuration can be daunting at first. The world is complex, and the mail configuration reflects that. The distribution includes an m4-based configuration package that hides a lot of the complexity.

These configuration files are simpler than old versions largely because the world has become simpler; in particular, text-based host files are officially eliminated, obviating the need to ‘‘hide’’ hosts behind a registered internet gateway.

These files also assume that most of your neighbors use domain-based UUCP addressing; that is, instead of naming hosts as ‘‘host!user’’ they will use ‘‘host.domain!user’’. The configuration files can be customized to work around this, but it is more complex.

Our configuration files are processed by *m4* to facilitate local customization; the directory *cf* of the *sendmail* distribution directory contains the source files. This directory contains several subdirectories:

cf — Both site-dependent and site-independent descriptions of hosts. These can be literal host names (e.g., ‘‘ucbvax.mc’’) when the hosts are gateways or more general descriptions (such as ‘‘tcpproto.mc’’ as a general description of an SMTP-connected host or ‘‘uucpproto.mc’’ as a general description of a UUCP-connected host). Files ending **.mc** (‘‘Master Configuration’’) are the input descriptions; the output is in the corresponding **.cf** file. The general structure of these files is described below.

domain — Site-dependent subdomain descriptions. These are tied to the way your organization wants to do addressing. For example, **domain/cs.exposed.m4** is our description for hosts in the CS.Berkeley.EDU subdomain that want their individual hostname to be externally visible; **domain/cs.hidden.m4** is the same except that the hostname is hidden (everything looks like it comes from CS.Berkeley.EDU). These are referenced using the DOMAIN **m4** macro in the **.mc** file.

feature — Definitions of specific features that some particular host in your site might want. These are referenced using the FEATURE **m4** macro. An example feature is use_cw_file (which tells *sendmail* to read an /etc/sendmail.cw file on startup to find the set of local names).

hack — Local hacks, referenced using the HACK **m4** macro. Try to avoid these. The point of having them here is to make it clear that they smell.

m4 — Site-independent *m4*(1) include files that have information common to all configuration files. This can be thought of as a ‘‘#include’’ directory.

mailer — Definitions of mailers, referenced using the MAILER **m4** macro. The mailer types that are known in this distribution are fax, local, smtp, uucp, and usenet. For example, to include support for the UUCP-based mailers, use ‘‘MAILER(uucp)’’.

ostype — Definitions describing various operating system environments (such as the location of support files). These are referenced using the OSTYPE **m4** macro.

sh — Shell files used by the **m4** build process. You shouldn't have to mess with these.

siteconfig — Local site configuration information, such as UUCP connectivity. They normally contain lists of site information, for example:

```
SITE(contessa)
SITE(hoptoad)
SITE(nkainc)
SITE(well)
```

They are referenced using the SITECONFIG macro:

```
SITECONFIG(site.config.file, name_of_site, X)
```

where *X* is the macro/class name to use. It can be U (indicating locally connected hosts) or one of W, X, or Y for up to three remote UUCP hubs.

If you are in a new domain (e.g., a company), you will probably want to create a cf/domain file for your domain. This consists primarily of relay definitions: for example, Berkeley's domain definition defines relays for BitNET, CSNET, and UUCP. Of these, only the UUCP relay is particularly specific to Berkeley. All of these are internet-style domain names. Please check to make certain they are reasonable for your domain.

Subdomains at Berkeley are also represented in the cf/domain directory. For example, the domain cs-exposed is the Computer Science subdomain with the local hostname shown to other users; cs-hidden makes users appear to be from the CS.Berkeley.EDU subdomain (with no local host information included). You will probably have to update this directory to be appropriate for your domain.

You will have to use or create **.mc** files in the *cf/cf* subdirectory for your hosts. This is detailed in the cf/README file.

1.3. Details of Installation Files

This subsection describes the files that comprise the *sendmail* installation.

1.3.1. /usr/sbin/sendmail

The binary for *sendmail* is located in /usr/sbin[1]. It should be setuid root. For security reasons, /, /usr, and /usr/sbin should be owned by root, mode 755[2].

1.3.2. /etc/sendmail.cf

This is the configuration file for *sendmail*[3]. This and /etc/sendmail.pid are the only non-library file names compiled into *sendmail*[4].

The configuration file is normally created using the distribution files described above. If you have a particularly unusual system configuration you may need to create a special version. The format of this file is detailed in later sections of this document.

---

[1]This is usually /usr/sbin on 4.4BSD and newer systems; many systems install it in /usr/lib. I understand it is in /usr/ucblib on System V Release 4.

[2]Some vendors ship them owned by bin; this creates a security hole that is not actually related to *sendmail*. Other important directories that should have restrictive ownerships and permissions are /bin, /usr/bin, /etc, /usr/etc, /lib, and /usr/lib.

[3]Actually, the pathname varies depending on the operating system; /etc is the preferred directory. Some older systems install it in **/usr/lib/sendmail.cf**, and I've also seen it in **/usr/ucblib** and **/etc/mail**. If you want to move this file, change *src/conf.h*.

[4]The system libraries can reference other files; in particular, system library subroutines that *sendmail* calls probably reference */etc/passwd* and */etc/resolv.conf*.

1.3.3. /usr/bin/newaliases

The *newaliases* command should just be a link to *sendmail*:

```
rm –f /usr/bin/newaliases
ln –s /usr/sbin/sendmail /usr/bin/newaliases
```

This can be installed in whatever search path you prefer for your system.

1.3.4. /var/spool/mqueue

The directory */var/spool/mqueue* should be created to hold the mail queue. This directory should be mode 700 and owned by root.

The actual path of this directory is defined in the **Q** option of the *sendmail.cf* file.

1.3.5. /etc/aliases*

The system aliases are held in ‘‘/etc/aliases’’. A sample is given in ‘‘lib/aliases’’ which includes some aliases which *must* be defined:

cp lib/aliases /etc/aliases
edit /etc/aliases

You should extend this file with any aliases that are apropos to your system.

Normally *sendmail* looks at a version of these files maintained by the *dbm*(3) or *db*(3) routines. These are stored either in ‘‘/etc/aliases.dir’’ and ‘‘/etc/aliases.pag’’ or ‘‘/etc/aliases.db’’ depending on which database package you are using. These can initially be created as empty files, but they will have to be initialized promptly. These should be mode 644:

```
cp /dev/null /etc/aliases.dir
cp /dev/null /etc/aliases.pag
chmod 644 /etc/aliases.*
newaliases
```

The *db* routines preset the mode reasonably, so this step can be skipped. The actual path of this file is defined in the **A** option of the *sendmail.cf* file.

1.3.6. /etc/rc

It will be necessary to start up the *sendmail* daemon when your system reboots. This daemon performs two functions: it listens on the SMTP socket for connections (to receive mail from a remote system) and it processes the queue periodically to insure that mail gets delivered when hosts come up.

Add the following lines to ‘‘/etc/rc’’ (or ‘‘/etc/rc.local’’ as appropriate) in the area where it is starting up the daemons:

```
if [ –f /usr/sbin/sendmail –a –f /etc/sendmail.cf ]; then
        (cd /var/spool/mqueue; rm –f [lnx]f*)
        /usr/sbin/sendmail –bd –q30m &
        echo –n ’ sendmail’ >/dev/console
fi
```

The ‘‘cd’’ and ‘‘rm’’ commands insure that all lock files have been removed; extraneous lock files may be left around if the system goes down in the middle of processing a message. The line that actually invokes *sendmail* has two flags: ‘‘–bd’’ causes it to listen on the SMTP port, and ‘‘–q30m’’ causes it to run the queue every half hour.

Some people use a more complex startup script, removing zero length qf files and df files for which there is no qf file. For example, see Figure 1 for an example of a complex startup script.

If you are not running a version of UNIX that supports Berkeley TCP/IP, do not include the **–bd** flag.

```
# remove zero length qf files
for qffile in qf*
do
	if [ –r $qffile ]
	then
		if [ ! –s $qffile ]
		then
			echo –n " <zero: $qffile>" > /dev/console
			rm –f $qffile
		fi
	fi
done
# rename tf files to be qf if the qf does not exist
for tffile in tf*
do
	qffile=‘echo $tffile | sed ’s/t/q/’‘
	if [ –r $tffile –a ! –f $qffile ]
	then
		echo –n " <recovering: $tffile>" > /dev/console
		mv $tffile $qffile
	else
		echo –n " <extra: $tffile>" > /dev/console
		rm –f $tffile
	fi
done
# remove df files with no corresponding qf files
for dffile in df*
do
	qffile=‘echo $dffile | sed ’s/d/q/’‘
	if [ –r $dffile –a ! –f $qffile ]
	then
		echo –n " <incomplete: $dffile>" > /dev/console
		mv $dffile ‘echo $dffile | sed ’s/d/D/’‘
	fi
done
# announce files that have been saved during disaster recovery
for xffile in [A-Z]f*
do
	echo –n " <panic: $xffile>" > /dev/console
done
```

Figure 1 — A complex startup script

1.3.7. /usr/lib/sendmail.hf

This is the help file used by the SMTP **HELP** command. It should be copied from ‘‘lib/sendmail.hf’’:

```
cp lib/sendmail.hf /usr/lib
```

The actual path of this file is defined in the **H** option of the *sendmail.cf* file.

1.3.8. /etc/sendmail.st

If you wish to collect statistics about your mail traffic, you should create the file ‘‘/etc/sendmail.st’’:

```
cp /dev/null /etc/sendmail.st
chmod 666 /etc/sendmail.st
```

This file does not grow. It is printed with the program ‘‘mailstats/mailstats.c.’’ The actual path of this file is defined in the **S** option of the *sendmail.cf* file.

1.3.9. /usr/bin/mailq

If *sendmail* is invoked as ‘‘mailq,’’ it will simulate the **−bp** flag (i.e., *sendmail* will print the contents of the mail queue; see below). This should be a link to /usr/sbin/sendmail.

2. NORMAL OPERATIONS

2.1. The System Log

The system log is supported by the *syslogd*(8) program. All messages from *sendmail* are logged under the LOG_MAIL facility[5].

2.1.1. Format

Each line in the system log consists of a timestamp, the name of the machine that generated it (for logging from several machines over the local area network), the word ‘‘sendmail:’’, and a message[6]. Most messages are a sequence of *name*=*value* pairs.

The two most common lines are logged when a message is processed. The first logs the receipt of a message; there will be exactly one of these per message. Some fields may be omitted if they do not contain interesting information. Fields are:

from	The envelope sender address.
size	The size of the message in bytes.
class	The class (i.e., numeric precedence) of the message.
pri	The initial message priority (used for queue sorting).
nrcpts	The number of envelope recipients for this message (after aliasing and forwarding).
msgid	The message id of the message (from the header).
proto	The protocol used to receive this message (e.g., ESMTP or UUCP)
relay	The machine from which it was received.

There is also one line logged per delivery attempt (so there can be several per message if delivery is deferred or there are multiple recipients). Fields are:

to	A comma-separated list of the recipients to this mailer.
ctladdr	The ‘‘controlling user’’, that is, the name of the user whose credentials we use for delivery.
delay	The total delay between the time this message was received and the time it was delivered.
xdelay	The amount of time needed in this delivery attempt (normally indicative of the speed of the connection).

[5]Except on Ultrix, which does not support facilities in the syslog.

[6]This format may vary slightly if your vendor has changed the syntax.

mailer	The name of the mailer used to deliver to this recipient.
relay	The name of the host that actually accepted (or rejected) this recipient.
stat	The delivery status.

Not all fields are present in all messages; for example, the relay is not listed for local deliveries.

2.1.2. Levels

If you have *syslogd* (8) or an equivalent installed, you will be able to do logging. There is a large amount of information that can be logged. The log is arranged as a succession of levels. At the lowest level only extremely strange situations are logged. At the highest level, even the most mundane and uninteresting events are recorded for posterity. As a convention, log levels under ten are considered generally ‘‘useful;’’ log levels above 64 are reserved for debugging purposes. Levels from 11–64 are reserved for verbose information that some sites might want.

A complete description of the log levels is given in section 4.6.

2.2. Dumping State

You can ask *sendmail* to log a dump of the open files and the connection cache by sending it a SIGUSR1 signal. The results are logged at LOG_DEBUG priority.

2.3. The Mail Queue

Sometimes a host cannot handle a message immediately. For example, it may be down or overloaded, causing it to refuse connections. The sending host is then expected to save this message in its mail queue and attempt to deliver it later.

Under normal conditions the mail queue will be processed transparently. However, you may find that manual intervention is sometimes necessary. For example, if a major host is down for a period of time the queue may become clogged. Although *sendmail* ought to recover gracefully when the host comes up, you may find performance unacceptably bad in the meantime.

2.3.1. Printing the queue

The contents of the queue can be printed using the *mailq* command (or by specifying the −**bp** flag to *sendmail*):

```
mailq
```

This will produce a listing of the queue id’s, the size of the message, the date the message entered the queue, and the sender and recipients.

2.3.2. Forcing the queue

Sendmail should run the queue automatically at intervals. The algorithm is to read and sort the queue, and then to attempt to process all jobs in order. When it attempts to run the job, *sendmail* first checks to see if the job is locked. If so, it ignores the job.

There is no attempt to insure that only one queue processor exists at any time, since there is no guarantee that a job cannot take forever to process (however, *sendmail* does include heuristics to try to abort jobs that are taking absurd amounts of time; technically, this violates RFC 821, but is blessed by RFC 1123). Due to the locking algorithm, it is impossible for one job to freeze the entire queue. However, an uncooperative recipient host or a program recipient that never returns can accumulate many processes in your system. Unfortunately, there is no completely general way to solve this.

In some cases, you may find that a major host going down for a couple of days may create a prohibitively large queue. This will result in *sendmail* spending an inordinate amount of time sorting the queue. This situation can be fixed by moving the queue to a temporary place and creating a new queue. The old queue can be run later when the offending host returns to service.

To do this, it is acceptable to move the entire queue directory:

```
cd /var/spool
mv mqueue omqueue; mkdir mqueue; chmod 700 mqueue
```

You should then kill the existing daemon (since it will still be processing in the old queue directory) and create a new daemon.

To run the old mail queue, run the following command:

```
/usr/sbin/sendmail −oQ/var/spool/omqueue −q
```

The **−oQ** flag specifies an alternate queue directory and the **−q** flag says to just run every job in the queue. If you have a tendency toward voyeurism, you can use the **−v** flag to watch what is going on.

When the queue is finally emptied, you can remove the directory:

```
rmdir /var/spool/omqueue
```

2.4. The Service Switch

The implementation of certain system services such as host and user name lookup is controlled by the service switch. If the host operating system supports such a switch *sendmail* will use the native version. Ultrix, Solaris, and DEC OSF/1 are examples of such systems.

If the underlying operating system does not support a service switch (e.g., SunOS, HP-UX, BSD) then *sendmail* will provide a stub implementation. The **ServiceSwitchFile** option points to the name of a file that has the service definitions Each line has the name of a service and the possible implementations of that service. For example, the file:

```
hosts    dns files nis
aliases  files nis
```

will ask *sendmail* to look for hosts in the Domain Name System first. If the requested host name is not found, it tries local files, and if that fails it tries NIS. Similarly, when looking for aliases it will try the local files first followed by NIS.

Service switches are not completely integrated. For example, despite the fact that the host entry listed in the above example specifies to look in NIS, on SunOS this won't happen because the system implementation of *gethostbyname* (3) doesn't understand this. If there is enough demand *sendmail* may reimplement *gethostbyname* (3), *gethostbyaddr* (3), *getpwent* (3), and the other system routines that would be necessary to make this work seamlessly.

2.5. The Alias Database

The alias database exists in two forms. One is a text form, maintained in the file */etc/aliases.* The aliases are of the form

```
name: name1, name2, ...
```

Only local names may be aliased; e.g.,

```
eric@prep.ai.MIT.EDU: eric@CS.Berkeley.EDU
```

will not have the desired effect (except on prep.ai.MIT.EDU, and they probably don't want me)[7]. Aliases may be continued by starting any continuation lines with a space or a tab. Blank lines and lines beginning with a sharp sign (‘‘#’’) are comments.

The second form is processed by the *ndbm* (3)[8] or *db* (3) library. This form is in the files */etc/aliases.dir* and */etc/aliases.pag.* This is the form that *sendmail* actually uses to resolve aliases. This technique is used to improve performance.

---

[7] Actually, any mailer that has the ‘A’ mailer flag set will permit aliasing; this is normally limited to the local mailer.

[8] The *gdbm* package probably works as well.

The control of search order is actually set by the service switch. Essentially, the entry

OAswitch:aliases

is always added as the first alias entry; also, the first alias file name without a class (e.g., without ‘‘nis:’’ on the front) will be used as the name of the file for a ‘‘files’’ entry in the aliases switch. For example, if the configuration file contains

OA/etc/aliases

and the service switch contains

aliases nis files nisplus

then aliases will first be searched in the NIS database, then in /etc/aliases, then in the NIS+ database.

You can also use NIS-based alias files. For example, the specification:

OA/etc/aliases
OAnis:mail.aliases@my.nis.domain

will first search the /etc/aliases file and then the map named ‘‘mail.aliases’’ in ‘‘my.nis.domain’’. Warning: if you build your own NIS-based alias files, be sure to provide the **−l** flag to *makedbm*(8) to map upper case letters in the keys to lower case; otherwise, aliases with upper case letters in their names won’t match incoming addresses.

Additional flags can be added after the colon exactly like a **K** line — for example:

OAnis:−N mail.aliases@my.nis.domain

will search the appropriate NIS map and always include null bytes in the key.

2.5.1. Rebuilding the alias database

The DB or DBM version of the database may be rebuilt explicitly by executing the command

newaliases

This is equivalent to giving *sendmail* the **−bi** flag:

/usr/sbin/sendmail −bi

If the **RebuildAliases** (old **D**) option is specified in the configuration, *sendmail* will rebuild the alias database automatically if possible when it is out of date. Auto-rebuild can be dangerous on heavily loaded machines with large alias files; if it might take more than the rebuild timeout (option **AliasWait**, old **a**, which is normally five minutes) to rebuild the database, there is a chance that several processes will start the rebuild process simultaneously.

If you have multiple aliases databases specified, the **−bi** flag rebuilds all the database types it understands (for example, it can rebuild NDBM databases but not NIS databases).

2.5.2. Potential problems

There are a number of problems that can occur with the alias database. They all result from a *sendmail* process accessing the DBM version while it is only partially built. This can happen under two circumstances: One process accesses the database while another process is rebuilding it, or the process rebuilding the database dies (due to being killed or a system crash) before completing the rebuild.

Sendmail has three techniques to try to relieve these problems. First, it ignores interrupts while rebuilding the database; this avoids the problem of someone aborting the process leaving a partially rebuilt database. Second, it locks the database source file during the rebuild — but that may not work over NFS or if the file is unwritable. Third, at the end of the rebuild it adds an alias of the form

@: @

(which is not normally legal). Before *sendmail* will access the database, it checks to insure that this entry exists[9].

2.5.3. List owners

If an error occurs on sending to a certain address, say ‘‘*x*’’, *sendmail* will look for an alias of the form ‘‘owner-*x*’’ to receive the errors. This is typically useful for a mailing list where the submitter of the list has no control over the maintenance of the list itself; in this case the list maintainer would be the owner of the list. For example:

```
unix-wizards: eric@ucbarpa, wnj@monet, nosuchuser,
        sam@matisse
owner-unix-wizards: unix-wizards-request
unix-wizards-request: eric@ucbarpa
```

would cause ‘‘eric@ucbarpa’’ to get the error that will occur when someone sends to unix-wizards due to the inclusion of ‘‘nosuchuser’’ on the list.

List owners also cause the envelope sender address to be modified. The contents of the owner alias are used if they point to a single user, otherwise the name of the alias itself is used. For this reason, and to obey Internet conventions, the ‘‘owner-’’ address normally points at the ‘‘-request’’ address; this causes messages to go out with the typical Internet convention of using ‘‘*list*-request’’ as the return address.

2.6. User Information Database

If you have a version of *sendmail* with the user information database compiled in, and you have specified one or more databases using the **U** option, the databases will be searched for a *user*:maildrop entry. If found, the mail will be sent to the specified address.

2.7. Per-User Forwarding (.forward Files)

As an alternative to the alias database, any user may put a file with the name ‘‘.forward’’ in his or her home directory. If this file exists, *sendmail* redirects mail for that user to the list of addresses listed in the .forward file. For example, if the home directory for user ‘‘mckusick’’ has a .forward file with contents:

```
mckusick@ernie
kirk@calder
```

then any mail arriving for ‘‘mckusick’’ will be redirected to the specified accounts.

Actually, the configuration file defines a sequence of filenames to check. By default, this is the user’s .forward file, but can be defined to be more generally using the **J** option. If you change this, you will have to inform your user base of the change; .forward is pretty well incorporated into the collective subconscious.

2.8. Special Header Lines

Several header lines have special interpretations defined by the configuration file. Others have interpretations built into *sendmail* that cannot be changed without changing the code. These builtins are described here.

2.8.1. Return-Receipt-To:

N.B. *This header line has been preempted by the RET= parameter on the ESMTP transaction as documented by RFC XXX [in preparation]. This header field is deprecated.*

If this header is sent, a message will be sent to any specified addresses when the final delivery is complete, that is, when successfully delivered to a mailer with the **l** flag (local

[9] The **AliasWait** option is required in the configuration for this action to occur. This should normally be specified.

delivery) set in the mailer descriptor[10]. This header can be disabled with the ‘‘noreceipts’’ privacy flag. See the **PrivacyFlags** option.

2.8.2. Errors-To:

If errors occur anywhere during processing, this header will cause error messages to go to the listed addresses. This is intended for mailing lists.

The Errors-To: header was created in the bad old days when UUCP didn’t understand the distinction between an envelope and a header; this was a hack to provide what should now be passed as the envelope sender address. It should go away. It is only used if the **UseErrorsTo** option is set.

The Errors-To: header is official deprecated and will go away in a future release.

2.8.3. Apparently-To:

RFC 822 requires at least one recipient field (To:, Cc:, or Bcc: line) in every message. If a message comes in with no recipients listed in the message then *sendmail* will adjust the header based on the ‘‘NoRecipientAction’’ option. One of the possible actions is to add an ‘‘Apparently-To:’’ header line for any recipients it is aware of. This is not put in as a standard recipient line to warn any recipients that the list is not complete.

The Apparently-To: header is non-standard and is deprecated.

2.8.4. Precedence

The Precedence: header can be used as a crude control of message priority. It tweaks the sort order in the queue and can be configured to change the message timeout values.

2.9. IDENT Protocol Support

Sendmail supports the IDENT protocol as defined in RFC 1413. Although this enhances identification of the author of an email message by doing a ‘‘call back’’ to the originating system to include the owner of a particular TCP connection in the audit trail it is in no sense perfect; a determined forger can easily spoof the IDENT protocol. The following description is excerpted from RFC 1413:

> 6. Security Considerations
>
> The information returned by this protocol is at most as trustworthy as the host providing it OR the organization operating the host. For example, a PC in an open lab has few if any controls on it to prevent a user from having this protocol return any identifier the user wants. Likewise, if the host has been compromised the information returned may be completely erroneous and misleading.
>
> The Identification Protocol is not intended as an authorization or access control protocol. At best, it provides some additional auditing information with respect to TCP connections. At worst, it can provide misleading, incorrect, or maliciously incorrect information.
>
> The use of the information returned by this protocol for other than auditing is strongly discouraged. Specifically, using Identification Protocol information to make access control decisions - either as the primary method (i.e., no other checks) or as an adjunct to other methods may result in a weakening of normal host security.
>
> An Identification server may reveal information about users, entities, objects or processes which might normally be considered private. An Identification server provides service which is a rough analog of the CallerID services provided by some phone companies and many of

[10] Some sites disable this header, and other (non-*sendmail*) systems do not implement it. Do not assume that a failure to get a return receipt means that the mail did not arrive. Also, do not assume that getting a return receipt means that the mail has been read; it just means that the message has been delivered to the recipient’s mailbox. In fact, this header is pretty useless unless you are certain that the recipient mailer implements it.

the same privacy considerations and arguments that apply to the CallerID service apply to Identification. If you wouldn't run a "finger" server due to privacy considerations you may not want to run this protocol.

In some cases your system may not work properly with IDENT support due to a bug in the TCP/IP implementation. The symptoms will be that for some hosts the SMTP connection will be closed almost immediately. If this is true or if you do not want to use IDENT, you should set the IDENT timeout to zero; this will disable the IDENT protocol.

3. ARGUMENTS

The complete list of arguments to *sendmail* is described in detail in Appendix A. Some important arguments are described here.

3.1. Queue Interval

The amount of time between forking a process to run through the queue is defined by the **−q** flag. If you run with delivery mode set to **i** or **b** this can be relatively large, since it will only be relevant when a host that was down comes back up. If you run in **q** mode it should be relatively short, since it defines the maximum amount of time that a message may sit in the queue. (See also the MinQueueAge option.)

RFC 1123 section 5.3.1.1 says that this value should be at least 30 minutes (although that probably doesn't make sense if you use ‘‘queue-only’’ mode).

3.2. Daemon Mode

If you allow incoming mail over an IPC connection, you should have a daemon running. This should be set by your */etc/rc* file using the **−bd** flag. The **−bd** flag and the **−q** flag may be combined in one call:

```
/usr/sbin/sendmail −bd −q30m
```

An alternative approach is to invoke sendmail from *inetd*(8) (use the **−bs** flag to ask sendmail to speak SMTP on its standard input and output). This works and allows you to wrap *sendmail* in a TCP wrapper program, but may be a bit slower since the configuration file has to be re-read on every message that comes in. If you do this, you still need to have a *sendmail* running to flush the queue:

```
/usr/sbin/sendmail −q30m
```

3.3. Forcing the Queue

In some cases you may find that the queue has gotten clogged for some reason. You can force a queue run using the **−q** flag (with no value). It is entertaining to use the **−v** flag (verbose) when this is done to watch what happens:

```
/usr/sbin/sendmail −q −v
```

You can also limit the jobs to those with a particular queue identifier, sender, or recipient using one of the queue modifiers. For example, ‘‘−qRberkeley’’ restricts the queue run to jobs that have the string ‘‘berkeley’’ somewhere in one of the recipient addresses. Similarly, ‘‘−qSstring’’ limits the run to particular senders and ‘‘−qIstring’’ limits it to particular queue identifiers.

3.4. Debugging

There are a fairly large number of debug flags built into *sendmail*. Each debug flag has a number and a level, where higher levels means to print out more information. The convention is that levels greater than nine are ‘‘absurd,’’ i.e., they print out so much information that you wouldn't normally want to see them except for debugging that particular piece of code. Debug flags are set using the **−d** option; the syntax is:

debug-flag:	−**d** debug-list
debug-list:	debug-option [, debug-option]*
debug-option:	debug-range [. debug-level]
debug-range:	integer | integer − integer
debug-level:	integer

where spaces are for reading ease only. For example,

−d12	Set flag 12 to level 1
−d12.3	Set flag 12 to level 3
−d3−17	Set flags 3 through 17 to level 1
−d3−17.4	Set flags 3 through 17 to level 4

For a complete list of the available debug flags you will have to look at the code (they are too dynamic to keep this documentation up to date).

3.5. Changing the Values of Options

Options can be overridden using the **−o** or **−O** command line flags. For example,

```
/usr/sbin/sendmail −oT2m
```

sets the **T** (timeout) option to two minutes for this run only; the equivalent line using the long option name is

```
/usr/sbin/sendmail -OQueueTimeout=2m
```

Some options have security implications. Sendmail allows you to set these, but relinquishes its setuid root permissions thereafter[11].

3.6. Trying a Different Configuration File

An alternative configuration file can be specified using the **−C** flag; for example,

```
/usr/sbin/sendmail −Ctest.cf −oQ/tmp/mqueue
```

uses the configuration file *test.cf* instead of the default */etc/sendmail.cf.* If the **−C** flag has no value it defaults to *sendmail.cf* in the current directory.

Sendmail gives up its setuid root permissions when you use this flag, so it is common to use a publicly writable directory (such as /tmp) as the spool directory (QueueDirectory or Q option) while testing.

3.7. Logging Traffic

Many SMTP implementations do not fully implement the protocol. For example, some personal computer based SMTPs do not understand continuation lines in reply codes. These can be very hard to trace. If you suspect such a problem, you can set traffic logging using the **−X** flag. For example,

```
/usr/sbin/sendmail −X /tmp/traffic −bd
```

will log all traffic in the file */tmp/traffic*.

This logs a lot of data very quickly and should **NEVER** be used during normal operations. After starting up such a daemon, force the errant implementation to send a message to your host. All message traffic in and out of *sendmail*, including the incoming SMTP traffic, will be logged in this file.

---

[11]That is, it sets its effective uid to the real uid; thus, if you are executing as root, as from root's crontab file or during system startup the root permissions will still be honored.

3.8. Testing Configuration Files

When you build a configuration table, you can do a certain amount of testing using the ‘‘test mode’’ of *sendmail*. For example, you could invoke *sendmail* as:

sendmail −bt −Ctest.cf

which would read the configuration file ‘‘test.cf’’ and enter test mode. In this mode, you enter lines of the form:

rwset address

where *rwset* is the rewriting set you want to use and *address* is an address to apply the set to. Test mode shows you the steps it takes as it proceeds, finally showing you the address it ends up with. You may use a comma separated list of rwsets for sequential application of rules to an input. For example:

3,1,21,4 monet:bollard

first applies ruleset three to the input ‘‘monet:bollard.’’ Ruleset one is then applied to the output of ruleset three, followed similarly by rulesets twenty-one and four.

If you need more detail, you can also use the ‘‘−d21’’ flag to turn on more debugging. For example,

sendmail −bt −d21.99

turns on an incredible amount of information; a single word address is probably going to print out several pages worth of information.

You should be warned that internally, *sendmail* applies ruleset 3 to all addresses. In test mode you will have to do that manually. For example, older versions allowed you to use

0 bruce@broadcast.sony.com

This version requires that you use:

3,0 bruce@broadcast.sony.com

As of version 8.7, some other syntaxes are available in test mode:

- .D x value defines macro *x* to have the indicated *value*. This is useful when debugging rules that use the **$&***x* syntax.
- .C c value adds the indicated *value* to class *c*.
- .S ruleset dumps the contents of the indicated ruleset.
- −d debug-spec is equivalent to the command-line flag.

4. TUNING

There are a number of configuration parameters you may want to change, depending on the requirements of your site. Most of these are set using an option in the configuration file. For example, the line ‘‘O Timeout.queuereturn=5d’’ sets option ‘‘Timeout.queuereturn’’ to the value ‘‘5d’’ (five days).

Most of these options have appropriate defaults for most sites. However, sites having very high mail loads may find they need to tune them as appropriate for their mail load. In particular, sites experiencing a large number of small messages, many of which are delivered to many recipients, may find that they need to adjust the parameters dealing with queue priorities.

All versions of *sendmail* prior to 8.7 had single character option names. As of 8.7, options have long (multi-character names). Although old short names are still accepted, most new options do not have short equivalents.

This section only describes the options you are most likely to want to tweak; read section 5 for more details.

4.1. Timeouts

All time intervals are set using a scaled syntax. For example, ‘‘10m’’ represents ten minutes, whereas ‘‘2h30m’’ represents two and a half hours. The full set of scales is:

s	seconds
m	minutes
h	hours
d	days
w	weeks

4.1.1. Queue interval

The argument to the **−q** flag specifies how often a sub-daemon will run the queue. This is typically set to between fifteen minutes and one hour. RFC 1123 section 5.3.1.1 recommends that this be at least 30 minutes.

4.1.2. Read timeouts

Timeouts all have option names ‘‘Timeout.*suboption*’’. The recognized *suboption*s, their default values, and the minimum values allowed by RFC 1123 section 5.3.2 are:

initial — The wait for the initial 220 greeting message [5m, 5m].

helo — The wait for a reply from a HELO or EHLO command [5m, unspecified]. This may require a host name lookup, so five minutes is probably a reasonable minimum.

mail† — The wait for a reply from a MAIL command [10m, 5m].

rcpt† — The wait for a reply from a RCPT command [1h, 5m]. This should be long because it could be pointing at a list that takes a long time to expand (see below).

datainit† — The wait for a reply from a DATA command [5m, 2m].

datablock† — The wait for reading a data block (that is, the body of the message). [1h, 3m]. This should be long because it also applies to programs piping input to *sendmail* which have no guarantee of promptness.

datafinal† — The wait for a reply from the dot terminating a message. [1h, 10m]. If this is shorter than the time actually needed for the receiver to deliver the message, duplicates will be generated. This is discussed in RFC 1047.

rset — The wait for a reply from a RSET command [5m, unspecified].

quit — The wait for a reply from a QUIT command [2m, unspecified].

misc — The wait for a reply from miscellaneous (but short) commands such as NOOP (no-operation) and VERB (go into verbose mode). [2m, unspecified].

command† — In server SMTP, the time to wait for another command. [1h, 5m].

ident — The timeout waiting for a reply to an IDENT query [30s[12], unspecified].

For compatibility with old configuration files, if no *suboption* is specified, all the timeouts marked with † are set to the indicated value.

Many of the RFC 1123 minimum values may well be too short. *Sendmail* was designed to the RFC 822 protocols, which did not specify read timeouts; hence, versions of *sendmail* prior to version 8.1 did not guarantee to reply to messages promptly. In particular, a ‘‘RCPT’’ command specifying a mailing list will expand and verify the entire list; a large list on a slow

[12] On some systems the default is zero to turn the protocol off entirely.

system may easily take more than five minutes[13]. I recommend a one hour timeout — since a communications failure during the RCPT phase is rare, a long timeout is not onerous and may ultimately help reduce network load and duplicated messages.

For example, the lines:

```
O Timeout.command=25m
O Timeout.datablock=3h
```

sets the server SMTP command timeout to 25 minutes and the input data block timeout to three hours.

4.1.3. Message timeouts

After sitting in the queue for a few days, a message will time out. This is to insure that at least the sender is aware of the inability to send a message. The timeout is typically set to five days. It is sometimes considered convenient to also send a warning message if the message is in the queue longer than a few hours (assuming you normally have good connectivity; if your messages normally took several hours to send you wouldn't want to do this because it wouldn't be an unusual event). These timeouts are set using the **Timeout.queuereturn** and **Timeout.queuewarn** options in the configuration file (previously both were set using the **T** option).

Since these options are global, and since you can not know *a priori* how long another host outside your domain will be down, a five day timeout is recommended. This allows a recipient to fix the problem even if it occurs at the beginning of a long weekend. RFC 1123 section 5.3.1.1 says that this parameter should be ‘‘at least 4–5 days’’.

The **Timeout.queuewarn** value can be piggybacked on the **T** option by indicating a time after which a warning message should be sent; the two timeouts are separated by a slash. For example, the line

```
OT5d/4h
```

causes email to fail after five days, but a warning message will be sent after four hours. This should be large enough that the message will have been tried several times.

4.2. Forking During Queue Runs

By setting the **ForkEachJob** (**Y**) option, *sendmail* will fork before each individual message while running the queue. This will prevent *sendmail* from consuming large amounts of memory, so it may be useful in memory-poor environments. However, if the **ForkEachJob** option is not set, *sendmail* will keep track of hosts that are down during a queue run, which can improve performance dramatically.

If the **ForkEachJob** option is set, *sendmail* can not use connection caching.

4.3. Queue Priorities

Every message is assigned a priority when it is first instantiated, consisting of the message size (in bytes) offset by the message class (which is determined from the Precedence: header) times the ‘‘work class factor’’ and the number of recipients times the ‘‘work recipient factor.’’ The priority is used to order the queue. Higher numbers for the priority mean that the message will be processed later when running the queue.

The message size is included so that large messages are penalized relative to small messages. The message class allows users to send ‘‘high priority’’ messages by including a ‘‘Precedence:’’ field in their message; the value of this field is looked up in the **P** lines of the configuration file. Since the number of recipients affects the amount of load a message presents to the system, this is also included into the priority.

[13]This verification includes looking up every address with the name server; this involves network delays, and can in some cases can be considerable.

The recipient and class factors can be set in the configuration file using the **RecipientFactor** (**y**) and **ClassFactor** (**z**) options respectively. They default to 30000 (for the recipient factor) and 1800 (for the class factor). The initial priority is:

$$pri = msgsize - (class \times \textbf{ClassFactor}) + (nrcpt \times \textbf{RecipientFactor})$$

(Remember, higher values for this parameter actually mean that the job will be treated with lower priority.)

The priority of a job can also be adjusted each time it is processed (that is, each time an attempt is made to deliver it) using the ‘‘work time factor,’’ set by the **RetryFactor** (**Z**) option. This is added to the priority, so it normally decreases the precedence of the job, on the grounds that jobs that have failed many times will tend to fail again in the future. The **RetryFactor** option defaults to 90000.

4.4. Load Limiting

Sendmail can be asked to queue (but not deliver) mail if the system load average gets too high using the **QueueLA** (**x**) option. When the load average exceeds the value of the **QueueLA** option, the delivery mode is set to **q** (queue only) if the **QueueFactor** (**q**) option divided by the difference in the current load average and the **QueueLA** option plus one exceeds the priority of the message — that is, the message is queued iff:

$$pri > \frac{\textbf{QueueFactor}}{LA - \textbf{QueueLA} + 1}$$

The **QueueFactor** option defaults to 600000, so each point of load average is worth 600000 priority points (as described above).

For drastic cases, the **RefuseLA** (**X**) option defines a load average at which *sendmail* will refuse to accept network connections. Locally generated mail (including incoming UUCP mail) is still accepted.

4.5. Delivery Mode

There are a number of delivery modes that *sendmail* can operate in, set by the **DeliveryMode** (**d**) configuration option. These modes specify how quickly mail will be delivered. Legal modes are:

i	deliver interactively (synchronously)
b	deliver in background (asynchronously)
q	queue only (don’t deliver)

There are tradeoffs. Mode ‘‘i’’ gives the sender the quickest feedback, but may slow down some mailers and is hardly ever necessary. Mode ‘‘q’’ puts the minimum load on your machine, but means that delivery may be delayed for up to the queue interval. Mode ‘‘b’’ delivers promptly but can cause large numbers of processes if you have a mailer that takes a long time to deliver a message. Mode ‘‘b’’ is the usual default.

If you run in mode ‘‘q’’ (queue only) or ‘‘b’’ (deliver in background) *sendmail* will not expand aliases and follow .forward files upon initial receipt of the mail. This speeds up the response to RCPT commands. Mode ‘‘i’’ cannot be used by the SMTP server.

4.6. Log Level

The level of logging can be set for *sendmail*. The default using a standard configuration table is level 9. The levels are as follows:

0	No logging.
1	Serious system failures and potential security problems.
2	Lost communications (network problems) and protocol failures.
3	Other serious failures.

4 Minor failures.

5 Message collection statistics.

6 Creation of error messages, VRFY and EXPN commands.

7 Delivery failures (host or user unknown, etc.).

8 Successful deliveries and alias database rebuilds.

9 Messages being deferred (due to a host being down, etc.).

10 Database expansion (alias, forward, and userdb lookups).

20 Logs attempts to run locked queue files. These are not errors, but can be useful to note if your queue appears to be clogged.

30 Lost locks (only if using lockf instead of flock).

Additionally, values above 64 are reserved for extremely verbose debugging output. No normal site would ever set these.

4.7. File Modes

The modes used for files depend on what functionality you want and the level of security you require.

4.7.1. To suid or not to suid?

Sendmail can safely be made setuid to root. At the point where it is about to *exec* (2) a mailer, it checks to see if the userid is zero; if so, it resets the userid and groupid to a default (set by the **u** and **g** options). (This can be overridden by setting the **S** flag to the mailer for mailers that are trusted and must be called as root.) However, this will cause mail processing to be accounted (using *sa* (8)) to root rather than to the user sending the mail.

If you don't make *sendmail* setuid to root, it will still run but you lose a lot of functionality and a lot of privacy, since you'll have to make the queue directory world readable. You could also make *sendmail* setuid to some pseudo-user (e.g., create a user called ‘‘sendmail’’ and make *sendmail* setuid to that) which will fix the privacy problems but not the functionality issues. Also, this isn't a guarantee of security: for example, root occasionally sends mail, and the daemon often runs as root.

4.7.2. Should my alias database be writable?

At Berkeley we have the alias database (/etc/aliases*) mode 644. While this is not as flexible as if the database were more 666, it avoids potential security problems with a globally writable database.

The database that *sendmail* actually used is represented by the two files *aliases.dir* and *aliases.pag* (both in /etc) (or *aliases.db* if you are running with the new Berkeley database primitives). The mode on these files should match the mode on /etc/aliases. If *aliases* is writable and the DBM files (*aliases.dir* and *aliases.pag*) are not, users will be unable to reflect their desired changes through to the actual database. However, if *aliases* is read-only and the DBM files are writable, a slightly sophisticated user can arrange to steal mail anyway.

If your DBM files are not writable by the world or you do not have auto-rebuild enabled (with the **AutoRebuildAliases** option), then you must be careful to reconstruct the alias database each time you change the text version:

```
newaliases
```

If this step is ignored or forgotten any intended changes will also be ignored or forgotten.

4.8. Connection Caching

When processing the queue, *sendmail* will try to keep the last few open connections open to avoid startup and shutdown costs. This only applies to IPC connections.

When trying to open a connection the cache is first searched. If an open connection is found, it is probed to see if it is still active by sending a NOOP command. It is not an error if this fails; instead, the connection is closed and reopened.

Two parameters control the connection cache. The **ConnectionCacheSize** (**k**) option defines the number of simultaneous open connections that will be permitted. If it is set to zero, connections will be closed as quickly as possible. The default is one. This should be set as appropriate for your system size; it will limit the amount of system resources that *sendmail* will use during queue runs. Never set this higher than 4.

The **ConnectionCacheTimeout** (**K**) option specifies the maximum time that any cached connection will be permitted to idle. When the idle time exceeds this value the connection is closed. This number should be small (under ten minutes) to prevent you from grabbing too many resources from other hosts. The default is five minutes.

4.9. Name Server Access

Control of host address lookups is set by the **hosts** service entry in your service switch file. If you are on a system that has built-in service switch support (e.g., Ultrix, Solaris, or DEC OSF/1) then your system is probably configured properly already. Otherwise, *sendmail* will consult the file **/etc/service.switch**, which should be created. *Sendmail* only uses two entries: **hosts** and **aliases**.

However, some systems (such as SunOS) will do DNS lookups regardless of the setting of the service switch entry. In particular, the system routine *gethostbyname*(3) is used to look up host names, and many vendor versions try some combination of DNS, NIS, and file lookup in /etc/hosts without consulting a service switch. *Sendmail* makes no attempt to work around this problem, and the DNS lookup will be done anyway. If you do not have a nameserver configured at all, such as at a UUCP-only site, *sendmail* will get a ‘‘connection refused’’ message when it tries to connect to the name server. If the **hosts** switch entry has the service ‘‘dns’’ listed somewhere in the list, *sendmail* will interpret this to mean a temporary failure and will queue the mail for later processing; otherwise, it ignores the name server data.

The same technique is used to decide whether to do MX lookups. If you want MX support, you *must* have ‘‘dns’’ listed as a service in the **hosts** switch entry.

The **ResolverOptions** (**I**) option allows you to tweak name server options. The command line takes a series of flags as documented in *resolver*(3) (with the leading ‘‘RES_’’ deleted). Each can be preceded by an optional ‘+’ or ‘−’. For example, the line

    O ResolverOptions=+AAONLY −DNSRCH

turns on the AAONLY (accept authoritative answers only) and turns off the DNSRCH (search the domain path) options. Most resolver libraries default DNSRCH, DEFNAMES, and RECURSE flags on and all others off. You can also include ‘‘HasWildcardMX’’ to specify that there is a wildcard MX record matching your domain; this turns off MX matching when canonifying names, which can lead to inappropriate canonifications.

Version level 1 configurations turn DNSRCH and DEFNAMES off when doing delivery lookups, but leave them on everywhere else. Version 8 of *sendmail* ignores them when doing canonification lookups (that is, when using $[... $]), and always does the search. If you don’t want to do automatic name extension, don’t call $[... $].

The search rules for $[... $] are somewhat different than usual. If the name being looked up has at least one dot, it always tries the unmodified name first. If that fails, it tries the reduced search path, and lastly tries the unmodified name (but only for names without a dot, since names with a dot have already been tried). This allows names such as ‘‘utc.CS’’ to match the site in Czechoslovakia rather than the site in your local Computer Science department. It also prefers A and CNAME records over MX records — that is, if it finds an MX record it makes note of it, but keeps looking. This way, if you have a wildcard MX record matching your domain, it will not assume that all names match.

To completely turn off all name server access on systems without service switch support (such as SunOS) you will have to recompile with −DNAMED_BIND=0 and remove −lresolv from the list of libraries to be searched when linking.

4.10. Moving the Per-User Forward Files

Some sites mount each user's home directory from a local disk on their workstation, so that local access is fast. However, the result is that .forward file lookups are slow. In some cases, mail can even be delivered on machines inappropriately because of a file server being down. The performance can be especially bad if you run the automounter.

The **ForwardPath** (**J**) option allows you to set a path of forward files. For example, the config file line

O ForwardPath=/var/forward/$u:$z/.forward.$w

would first look for a file with the same name as the user's login in /var/forward; if that is not found (or is inaccessible) the file ''.forward.*machinename*'' in the user's home directory is searched. A truly perverse site could also search by sender by using $r, $s, or $f.

If you create a directory such as /var/forward, it should be mode 1777 (that is, the sticky bit should be set). Users should create the files mode 644.

4.11. Free Space

On systems that have one of the system calls in the *statfs*(2) family (including *statvfs* and *ustat*), you can specify a minimum number of free blocks on the queue filesystem using the **MinFreeBlocks** (**b**) option. If there are fewer than the indicated number of blocks free on the filesystem on which the queue is mounted the SMTP server will reject mail with the 452 error code. This invites the SMTP client to try again later.

Beware of setting this option too high; it can cause rejection of email when that mail would be processed without difficulty.

4.12. Maximum Message Size

To avoid overflowing your system with a large message, the **MaxMessageSize** option can be set to set an absolute limit on the size of any one message. This will be advertised in the ESMTP dialogue and checked during message collection.

4.13. Privacy Flags

The **PrivacyOptions** (**p**) option allows you to set certain ''privacy'' flags. Actually, many of them don't give you any extra privacy, rather just insisting that client SMTP servers use the HELO command before using certain commands or adding extra headers to indicate possible spoof attempts.

The option takes a series of flag names; the final privacy is the inclusive or of those flags. For example:

O PrivacyOptions=needmailhelo, noexpn

insists that the HELO or EHLO command be used before a MAIL command is accepted and disables the EXPN command.

The flags are detailed in section 5.1.6.

4.14. Send to Me Too

Normally, *sendmail* deletes the (envelope) sender from any list expansions. For example, if ''matt'' sends to a list that contains ''matt'' as one of the members he won't get a copy of the message. If the **–m** (me too) command line flag, or if the **MeToo** (**m**) option is set in the configuration file, this behaviour is suppressed. Some sites like to run the SMTP daemon with **–m**.

5. THE WHOLE SCOOP ON THE CONFIGURATION FILE

This section describes the configuration file in detail.

There is one point that should be made clear immediately: the syntax of the configuration file is designed to be reasonably easy to parse, since this is done every time *sendmail* starts up, rather than easy for a human to read or write. On the ‘‘future project’’ list is a configuration-file compiler.

The configuration file is organized as a series of lines, each of which begins with a single character defining the semantics for the rest of the line. Lines beginning with a space or a tab are continuation lines (although the semantics are not well defined in many places). Blank lines and lines beginning with a sharp symbol (‘#’) are comments.

5.1. R and S — Rewriting Rules

The core of address parsing are the rewriting rules. These are an ordered production system. *Sendmail* scans through the set of rewriting rules looking for a match on the left hand side (LHS) of the rule. When a rule matches, the address is replaced by the right hand side (RHS) of the rule.

There are several sets of rewriting rules. Some of the rewriting sets are used internally and must have specific semantics. Other rewriting sets do not have specifically assigned semantics, and may be referenced by the mailer definitions or by other rewriting sets.

The syntax of these two commands are:

S*n*

Sets the current ruleset being collected to *n*. If you begin a ruleset more than once it deletes the old definition.

R*lhs rhs comments*

The fields must be separated by at least one tab character; there may be embedded spaces in the fields. The *lhs* is a pattern that is applied to the input. If it matches, the input is rewritten to the *rhs*. The *comments* are ignored.

Macro expansions of the form **$***x* are performed when the configuration file is read. Expansions of the form **$&***x* are performed at run time using a somewhat less general algorithm. This for is intended only for referencing internally defined macros such as **$h** that are changed at runtime.

5.1.1. The left hand side

The left hand side of rewriting rules contains a pattern. Normal words are simply matched directly. Metasyntax is introduced using a dollar sign. The metasymbols are:

$*	Match zero or more tokens
$+	Match one or more tokens
$–	Match exactly one token
$=*x*	Match any phrase in class *x*
$~*x*	Match any word not in class *x*

If any of these match, they are assigned to the symbol **$***n* for replacement on the right hand side, where *n* is the index in the LHS. For example, if the LHS:

$–:$+

is applied to the input:

UCBARPA:eric

the rule will match, and the values passed to the RHS will be:

$1	UCBARPA
$2	eric

Additionally, the LHS can include **$@** to match zero tokens. This is *not* bound to a **$***n* on the RHS, and is normally only used when it stands alone in order to match the null input.

5.1.2. The right hand side

When the left hand side of a rewriting rule matches, the input is deleted and replaced by the right hand side. Tokens are copied directly from the RHS unless they begin with a dollar sign. Metasymbols are:

$*n*	Substitute indefinite token *n* from LHS
$[***name***$]	Canonicalize *name*
$(*map key* **$@***arguments* **$:***default* **$)**	Generalized keyed mapping function
$>*n*	‘‘Call’’ ruleset *n*
$#*mailer*	Resolve to *mailer*
$@*host*	Specify *host*
$:*user*	Specify *user*

The **$***n* syntax substitutes the corresponding value from a **$+**, **$–**, **$***, **$=**, or **$˜** match on the LHS. It may be used anywhere.

A host name enclosed between **$[** and **$]** is looked up in the host database(s) and replaced by the canonical name[14]. For example, ‘‘$[ftp$]’’ might become ‘‘ftp.CS.Berkeley.EDU’’ and ‘‘$[[128.32.130.2]$]’’ would become ‘‘vangogh.CS.Berkeley.EDU.’’ *Sendmail* recognizes it’s numeric IP address without calling the name server and replaces it with it’s canonical name.

The **$(** ... **$)** syntax is a more general form of lookup; it uses a named map instead of an implicit map. If no lookup is found, the indicated *default* is inserted; if no default is specified and no lookup matches, the value is left unchanged. The *arguments* are passed to the map for possible use.

The **$>***n* syntax causes the remainder of the line to be substituted as usual and then passed as the argument to ruleset *n*. The final value of ruleset *n* then becomes the substitution for this rule. The **$>** syntax can only be used at the beginning of the right hand side; it can be only be preceded by **$@** or **$:**.

The **$#** syntax should *only* be used in ruleset zero or a subroutine of ruleset zero. It causes evaluation of the ruleset to terminate immediately, and signals to *sendmail* that the address has completely resolved. The complete syntax is:

$#*mailer* **$@***host* **$:***user*

This specifies the {mailer, host, user} 3-tuple necessary to direct the mailer. If the mailer is local the host part may be omitted[15]. The *mailer* must be a single word, but the *host* and *user* may be multi-part. If the *mailer* is the builtin IPC mailer, the *host* may be a colon-separated list of hosts that are searched in order for the first working address (exactly like MX records). The *user* is later rewritten by the mailer-specific envelope rewriting set and assigned to the **$u** macro. As a special case, if the value to **$#** is ‘‘local’’ and the first character of the **$:** value is ‘‘@’’, the ‘‘@’’ is stripped off, and a flag is set in the address descriptor that causes sendmail to not do ruleset 5 processing.

Normally, a rule that matches is retried, that is, the rule loops until it fails. A RHS may also be preceded by a **$@** or a **$:** to change this behavior. A **$@** prefix causes the ruleset to return with the remainder of the RHS as the value. A **$:** prefix causes the rule to terminate immediately, but the ruleset to continue; this can be used to avoid continued application of a rule. The prefix is stripped before continuing.

The **$@** and **$:** prefixes may precede a **$>** spec; for example:

```
R$+		$: $>7 $1
```

matches anything, passes that to ruleset seven, and continues; the **$:** is necessary to avoid an infinite loop.

---

[14]This is actually completely equivalent to $(host *hostname*$). In particular, a **$:** default can be used.

[15]You may want to use it for special ‘‘per user’’ extensions. For example, in the address ‘‘jgm+foo@CMU.EDU’’; the ‘‘+foo’’ part is not part of the user name, and is passed to the local mailer for local use.

Substitution occurs in the order described, that is, parameters from the LHS are substituted, hostnames are canonicalized, ‘‘subroutines’’ are called, and finally **$#**, **$@**, and **$:** are processed.

5.1.3. Semantics of rewriting rule sets

There are five rewriting sets that have specific semantics. Four of these are related as depicted by figure 1.

Ruleset three should turn the address into ‘‘canonical form.’’ This form should have the basic syntax:

local-part@host-domain-spec

Ruleset three is applied by *sendmail* before doing anything with any address.

If no ‘‘@’’ sign is specified, then the host-domain-spec *may* be appended (box ‘‘D’’ in Figure 1) from the sender address (if the **C** flag is set in the mailer definition corresponding to the *sending* mailer).

Ruleset zero is applied after ruleset three to addresses that are going to actually specify recipients. It must resolve to a *{mailer, host, user}* triple. The *mailer* must be defined in the mailer definitions from the configuration file. The *host* is defined into the **$h** macro for use in the argv expansion of the specified mailer.

Rulesets one and two are applied to all sender and recipient addresses respectively. They are applied before any specification in the mailer definition. They must never resolve.

Ruleset four is applied to all addresses in the message. It is typically used to translate internal to external form.

5.1.4. IPC mailers

Some special processing occurs if the ruleset zero resolves to an IPC mailer (that is, a mailer that has ‘‘[IPC]’’ listed as the Path in the **M** configuration line. The host name passed after ‘‘$@’’ has MX expansion performed; this looks the name up in DNS to find alternate delivery sites.

The host name can also be provided as a dotted quad in square brackets; for example:

[128.32.149.78]

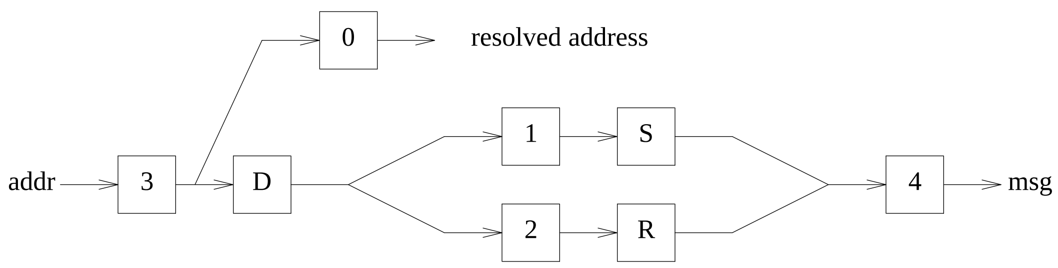

Figure 1 — Rewriting set semantics
D — sender domain addition
S — mailer-specific sender rewriting
R — mailer-specific recipient rewriting

This causes direct conversion of the numeric value to a TCP/IP host address.

The host name passed in after the ‘‘$@’’ may also be a colon-separated list of hosts. Each is separately MX expanded and the results are concatenated to make (essentially) one long MX list. The intent here is to create ‘‘fake’’ MX records that are not published in DNS for private internal networks.

As a final special case, the host name can be passed in as a text string in square brackets:

[ucbvax.berkeley.edu]

This form avoids the MX mapping. **N.B.:** *This is intended only for situations where you have a network firewall or other host that will do special processing for all your mail, so that your MX record points to a gateway machine; this machine could then do direct delivery to machines within your local domain. Use of this feature directly violates RFC 1123 section 5.3.5: it should not be used lightly.*

5.2. D — Define Macro

Macros are named with a single character or with a word in {braces}. Single character names may be selected from the entire ASCII set, but user-defined macros should be selected from the set of upper case letters only. Lower case letters and special symbols are used internally. Long names beginning with a lower case letter or a punctuation character are reserved for use by sendmail, so user-defined long macro names should begin with an upper case letter.

The syntax for macro definitions is:

D*x val*

where *x* is the name of the macro (which may be a single character or a word in braces) and *val* is the value it should have. There should be no spaces given that do not actually belong in the macro value.

Macros are interpolated using the construct **$***x*, where *x* is the name of the macro to be interpolated. This interpolation is done when the configuration file is read, except in **M** lines. The special construct **$&***x* can be used in **R** lines to get deferred interpolation.

Conditionals can be specified using the syntax:

$?x text1 $| text2 $.

This interpolates *text1* if the macro **$x** is set, and *text2* otherwise. The ‘‘else’’ (**$|**) clause may be omitted.

Lower case macro names are reserved to have special semantics, used to pass information in or out of *sendmail*, and special characters are reserved to provide conditionals, etc. Upper case names (that is, **$A** through **$Z**) are specifically reserved for configuration file authors.

The following macros are defined and/or used internally by *sendmail* for interpolation into argv’s for mailers or for other contexts. The ones marked † are information passed into sendmail[16], the ones marked ‡ are information passed both in and out of sendmail, and the unmarked macros are passed out of sendmail but are not otherwise used internally. These macros are:

$a The origination date in RFC 822 format. This is extracted from the Date: line.

$b The current date in RFC 822 format.

$c The hop count. This is a count of the number of Received: lines plus the value of the **−h** command line flag.

$d The current date in UNIX (ctime) format.

$e† The SMTP entry message. This is printed out when SMTP starts up. The first word must be the **$j** macro as specified by RFC821. Defaults to ‘‘$j Sendmail $v ready at $b’’. Commonly redefined to include the configuration version number, e.g., ‘‘$j Sendmail $v/$Z ready at $b’’

[16] As of version 8.6, all of these macros have reasonable defaults. Previous versions required that they be defined.

$f The envelope sender (from) address.

$g The sender address relative to the recipient. For example, if **$f** is ‘‘foo’’, **$g** will be ‘‘host!foo’’, ‘‘foo@host.domain’’, or whatever is appropriate for the receiving mailer.

$h The recipient host. This is set in ruleset 0 from the $# field of a parsed address.

$i The queue id, e.g., ‘‘HAA12345’’.

$j‡ The ‘‘official’’ domain name for this site. This is fully qualified if the full qualification can be found. It *must* be redefined to be the fully qualified domain name if your system is not configured so that information can find it automatically.

$k The UUCP node name (from the uname system call).

$l† The format of the UNIX from line. Unless you have changed the UNIX mailbox format, you should not change the default, which is ‘‘From $g $d’’.

$m The domain part of the *gethostname* return value. Under normal circumstances, **$j** is equivalent to **$w.$m**.

$n† The name of the daemon (for error messages). Defaults to ‘‘MAILER-DAEMON’’.

$o† The set of ‘‘operators’’ in addresses. A list of characters which will be considered tokens and which will separate tokens when doing parsing. For example, if ‘‘@’’ were in the **$o** macro, then the input ‘‘a@b’’ would be scanned as three tokens: ‘‘a,’’ ‘‘@,’’ and ‘‘b.’’ Defaults to ‘‘.:@[]’’, which is the minimum set necessary to do RFC 822 parsing; a richer set of operators is ‘‘.:%@!/[]’’, which adds support for UUCP, the %-hack, and X.400 addresses.

$p Sendmail’s process id.

$q† Default format of sender address. The **$q** macro specifies how an address should appear in a message when it is defaulted. Defaults to ‘‘<$g>’’. It is commonly redefined to be ‘‘$?x$x <$g>$|g.’’ or ‘‘g?x ($x)$.’’, corresponding to the following two formats:

Eric Allman <eric@CS.Berkeley.EDU>
eric@CS.Berkeley.EDU (Eric Allman)

Sendmail properly quotes names that have special characters if the first form is used.

$r Protocol used to receive the message. Set from the **−p** command line flag or by the SMTP server code.

$s Sender’s host name. Set from the **−p** command line flag or by the SMTP server code.

$t A numeric representation of the current time.

$u The recipient user.

$v The version number of the *sendmail* binary.

$w‡ The hostname of this site. This is the root name of this host (but see below for caveats).

$x The full name of the sender.

$z The home directory of the recipient.

$_ The validated sender address.

There are three types of dates that can be used. The **$a** and **$b** macros are in RFC 822 format; **$a** is the time as extracted from the ‘‘Date:’’ line of the message (if there was one), and **$b** is the current date and time (used for postmarks). If no ‘‘Date:’’ line is found in the incoming message, **$a** is set to the current time also. The **$d** macro is equivalent to the **$b** macro in UNIX (ctime) format.

The macros **$w**, **$j**, and **$m** are set to the identity of this host. *Sendmail* tries to find the fully qualified name of the host if at all possible; it does this by calling *gethostname*(2) to get the current hostname and then passing that to *gethostbyname*(3) which is supposed to return the canonical version of that host name.[17] Assuming this is successful, **$j** is set to the fully qualified name and **$m** is set to the domain part of the name (everything after the first dot). The **$w** macro is set to the first word (everything before the first dot) if you have a level 5 or higher configuration file; otherwise, it is set to the same value as **$j**. If the canonification is not successful, it is imperative that the config file set **$j** to the fully qualified domain name[18].

The **$f** macro is the id of the sender as originally determined; when mailing to a specific host the **$g** macro is set to the address of the sender *relative to the recipient.* For example, if I send to ''bollard@matisse.CS.Berkeley.EDU'' from the machine ''vangogh.CS.Berkeley.EDU'' the **$f** macro will be ''eric'' and the **$g** macro will be ''eric@vangogh.CS.Berkeley.EDU.''

The **$x** macro is set to the full name of the sender. This can be determined in several ways. It can be passed as flag to *sendmail*. It can be defined in the NAME environment variable. The third choice is the value of the ''Full-Name:'' line in the header if it exists, and the fourth choice is the comment field of a ''From:'' line. If all of these fail, and if the message is being originated locally, the full name is looked up in the */etc/passwd* file.

When sending, the **$h**, **$u**, and **$z** macros get set to the host, user, and home directory (if local) of the recipient. The first two are set from the **$@** and **$:** part of the rewriting rules, respectively.

The **$p** and **$t** macros are used to create unique strings (e.g., for the ''Message-Id:'' field). The **$i** macro is set to the queue id on this host; if put into the timestamp line it can be extremely useful for tracking messages. The **$v** macro is set to be the version number of *sendmail*; this is normally put in timestamps and has been proven extremely useful for debugging.

The **$c** field is set to the ''hop count,'' i.e., the number of times this message has been processed. This can be determined by the **−h** flag on the command line or by counting the timestamps in the message.

The **$r** and **$s** fields are set to the protocol used to communicate with *sendmail* and the sending hostname. They can be set together using the **−p** command line flag or separately using the **−M** or −**oM** flags.

The **$_** is set to a validated sender host name. If the sender is running an RFC 1413 compliant IDENT server and the receiver has the IDENT protocol turned on, it will include the user name on that host.

5.3. C and F — Define Classes

Classes of phrases may be defined to match on the left hand side of rewriting rules, where a ''phrase'' is a sequence of characters that do not contain space characters. For example a class of all local names for this site might be created so that attempts to send to oneself can be eliminated. These can either be defined directly in the configuration file or read in from another file. Classes are named as a single letter or a word in {braces}. Class names beginning with lower case letters and special characters are reserved for system use. Classes defined in config files may be given names from the set of upper case letters for short names or beginning with an upper case letter for long names.

The syntax is:

> **C***c phrase1 phrase2...*
> **F***c file*

The first form defines the class *c* to match any of the named words. It is permissible to split them among multiple lines; for example, the two forms:

> CHmonet ucbmonet

---

[17]For example, on some systems *gethostname* might return ''foo'' which would be mapped to ''foo.bar.com'' by *gethostbyname*.

[18]Older versions of sendmail didn't pre-define **$j** at all, so up until 8.6, config files *always* had to define **$j**.

and

CHmonet
CHucbmonet

are equivalent. The ‘‘F’’ form reads the elements of the class *c* from the named *file*.

Elements of classes can be accessed in rules using **$=** or **$~**. The **$~** (match entries not in class) only matches a single word; multi-word entries in the class are ignored in this context.

The class **$=w** is set to be the set of all names this host is known by. This can be used to match local hostnames.

The class **$=k** is set to be the same as **$k**, that is, the UUCP node name.

The class **$=m** is set to the set of domains by which this host is known, initially just **$m**.

The class **$=t** is set to the set of trusted users by the **T** configuration line. If you want to read trusted users from a file use **Ft***/file/name*.

The class **$=n** can be set to the set of MIME body types that can never be eight to seven bit encoded. It defaults to ‘‘message/rfc822’’, ‘‘message/partial’’, ‘‘message/external-body’’, and ‘‘multipart/signed’’.

Sendmail can be compiled to allow a *scanf*(3) string on the **F** line. This lets you do simplistic parsing of text files. For example, to read all the user names in your system */etc/passwd* file into a class, use

```
FL/etc/passwd %[^:]
```

which reads every line up to the first colon.

5.4. M — Define Mailer

Programs and interfaces to mailers are defined in this line. The format is:

M*name*, {*field*=*value* }*

where *name* is the name of the mailer (used internally only) and the ‘‘field=name’’ pairs define attributes of the mailer. Fields are:

Path	The pathname of the mailer
Flags	Special flags for this mailer
Sender	Rewriting set(s) for sender addresses
Recipient	Rewriting set(s) for recipient addresses
Argv	An argument vector to pass to this mailer
Eol	The end-of-line string for this mailer
Maxsize	The maximum message length to this mailer
Linelimit	The maximum line length in the message body
Directory	The working directory for the mailer
Userid	The default user and group id to run as
Nice	The nice(2) increment for the mailer
Charset	The default character set for 8-bit characters
Type	The MTS type information (used for error messages)

Only the first character of the field name is checked.

The following flags may be set in the mailer description. Any other flags may be used freely to conditionally assign headers to messages destined for particular mailers. Flags marked with † are not interpreted by the *sendmail* binary; these are the conventionally used to correlate to the flags portion of the **H** line. Flags marked with ‡ apply to the mailers for the sender address rather than the usual recipient mailers.

a Run Extended SMTP (ESMTP) protocol (defined in RFCs 1651, 1652, and 1653). This flag defaults on if the SMTP greeting message includes the word ‘‘ESMTP’’.

A Look up the user part of the address in the alias database. Normally this is only set for local mailers.

b Force a blank line on the end of a message. This is intended to work around some stupid versions of /bin/mail that require a blank line, but do not provide it themselves. It would not normally be used on network mail.

c Do not include comments in addresses. This should only be used if you have to work around a remote mailer that gets confused by comments. This strips addresses of the form ‘‘Phrase <address>’’ or ‘‘address (Comment)’’ down to just ‘‘address’’.

C‡ If mail is *received* from a mailer with this flag set, any addresses in the header that do not have an at sign (‘‘@’’) after being rewritten by ruleset three will have the ‘‘@domain’’ clause from the sender envelope address tacked on. This allows mail with headers of the form:

    From: usera@hosta
    To: userb@hostb, userc

to be rewritten as:

    From: usera@hosta
    To: userb@hostb, userc@hosta

automatically. However, it doesn’t really work reliably.

D† This mailer wants a ‘‘Date:’’ header line.

e This mailer is expensive to connect to, so try to avoid connecting normally; any necessary connection will occur during a queue run.

E Escape lines beginning with ‘‘From’’ in the message with a ‘>’ sign.

f The mailer wants a **–f** *from* flag, but only if this is a network forward operation (i.e., the mailer will give an error if the executing user does not have special permissions).

F† This mailer wants a ‘‘From:’’ header line.

g Normally, *sendmail* sends internally generated email (e.g., error messages) using the null return address as required by RFC 1123. However, some mailers don’t accept a null return address. If necessary, you can set the **g** flag to prevent *sendmail* from obeying the standards; error messages will be sent as from the MAILER-DAEMON (actually, the value of the **$n** macro).

h Upper case should be preserved in host names for this mailer.

I This mailer will be speaking SMTP to another *sendmail* — as such it can use special protocol features. This option is not required (i.e., if this option is omitted the transmission will still operate successfully, although perhaps not as efficiently as possible).

k Normally when *sendmail* connects to a host via SMTP, it checks to make sure that this isn’t accidently the same host name as might happen if *sendmail* is misconfigured or if a long-haul network interface is set in loopback mode. This flag disables the loopback check. It should only be used under very unusual circumstances.

K Currently unimplemented. Reserved for chunking.

l This mailer is local (i.e., final delivery will be performed).

L Limit the line lengths as specified in RFC821. This deprecated option should be replaced by the **L=** mail declaration. For historic reasons, the **L** flag also sets the **7** flag.

m This mailer can send to multiple users on the same host in one transaction. When a **$u** macro occurs in the *argv* part of the mailer definition, that field will be repeated as necessary for all qualifying users.

M† This mailer wants a ‘‘Message-Id:’’ header line.

n Do not insert a UNIX-style ‘‘From’’ line on the front of the message.

o Always run as the owner of the recipient mailbox. Normally *sendmail* runs as the sender for locally generated mail or as ‘‘daemon’’ (actually, the user specified in the **u** option) when delivering network mail. The normal behaviour is required by most local mailers, which will not allow the envelope sender address to be set unless the mailer is running as daemon. This flag is ignored if the **S** flag is set.

p Use the route-addr style reverse-path in the SMTP ‘‘MAIL FROM:’’ command rather than just the return address; although this is required in RFC821 section 3.1, many hosts do not process reverse-paths properly. Reverse-paths are officially discouraged by RFC 1123.

P† This mailer wants a ‘‘Return-Path:’’ line.

r Same as **f**, but sends a –**r** flag.

s Strip quote characters (" and \) off of the address before calling the mailer.

S Don’t reset the userid before calling the mailer. This would be used in a secure environment where *sendmail* ran as root. This could be used to avoid forged addresses. If the **U=** field is also specified, this flag causes the user id to always be set to that user and group (instead of leaving it as root).

u Upper case should be preserved in user names for this mailer.

U This mailer wants UUCP-style ‘‘From’’ lines with the ugly ‘‘remote from <host>’’ on the end.

w The user must have a valid account on this machine, i.e., getpwnam must succeed. If not, the mail is bounced. This is required to get ‘‘.forward’’ capability.

x† This mailer wants a ‘‘Full-Name:’’ header line.

X This mailer want to use the hidden dot algorithm as specified in RFC821; basically, any line beginning with a dot will have an extra dot prepended (to be stripped at the other end). This insures that lines in the message containing a dot will not terminate the message prematurely.

5 If no aliases are found for this address, pass the address through ruleset 5 for possible alternate resolution. This is intended to forward the mail to an alternate delivery spot.

7 Strip all output to seven bits. This is the default if the **L** flag is set. Note that clearing this option is not sufficient to get full eight bit data passed through *sendmail*. If the **7** option is set, this is essentially always set, since the eighth bit was stripped on input. Note that this option will only impact messages that didn’t have 8→7 bit MIME conversions performed.

8 If set, it is acceptable to send eight bit data to this mailer; the usual attempt to do 8→7 bit MIME conversions will be bypassed.

: Check addresses to see if they begin ‘‘:include:’’; if they do, convert them to the ‘‘*include*’’ mailer.

| Check addresses to see if they begin with a ‘|’; if they do, convert them to the ‘‘prog’’ mailer.

/ Check addresses to see if they begin with a ‘/’; if they do, convert them to the ‘‘*file*’’ mailer.

@ Look up addresses in the user database.

Configuration files prior to level 6 assume the ‘A’, ‘w’, ‘5’, ‘:’, ‘|’, ‘/’, and ‘@’ options on the mailer named ‘‘local’’.

The mailer with the special name ‘‘error’’ can be used to generate a user error. The (optional) host field is an exit status to be returned, and the user field is a message to be printed. The exit status may be numeric or one of the values USAGE, NOUSER, NOHOST, UNAVAILABLE, SOFTWARE, TEMPFAIL, PROTOCOL, or CONFIG to return the corresponding EX_ exit code. For example, the entry:

```
$#error $@ NOHOST $: Host unknown in this domain
```

on the RHS of a rule will cause the specified error to be generated and the ‘‘Host unknown’’ exit status to be returned if the LHS matches. This mailer is only functional in rulesets zero or five.

The mailer named ‘‘local’’ *must* be defined in every configuration file. This is used to deliver local mail, and is treated specially in several ways. Additionally, three other mailers named ‘‘prog’’, ‘‘*file*’’, and ‘‘*include*’’ may be defined to tune the delivery of messages to programs, files, and :include: lists respectively. They default to:

```
Mprog, P=/bin/sh, F=lsD, A=sh –c $u
M*file*, P=/dev/null, F=lsDFMPEu, A=FILE
M*include*, P=/dev/null, F=su, A=INCLUDE
```

The Sender and Recipient rewriting sets may either be a simple ruleset id or may be two ids separated by a slash; if so, the first rewriting set is applied to envelope addresses and the second is applied to headers.

The Directory is actually a colon-separated path of directories to try. For example, the definition ''D=$z:/'' first tries to execute in the recipient's home directory; if that is not available, it tries to execute in the root of the filesystem. This is intended to be used only on the ''prog'' mailer, since some shells (such as *csh*) refuse to execute if they cannot read the home directory. Since the queue directory is not normally readable by unprivileged users *csh* scripts as recipients can fail.

The Userid specifies the default user and group id to run as, overriding the **DefaultUser** option (q.v.). If the **S** mailer flag is also specified, this is the user and group to run as in all circumstances. This may be given as *user:group* to set both the user and group id; either may be an integer or a symbolic name to be looked up in the *passwd* and *group* files respectively. If only a symbolic user name is specified, the group id in the *passwd* file for that user is used as the group id.

The Charset field is used when converting a message to MIME; this is the character set used in the Content-Type: header. If this is not set, the **DefaultCharset** option is used, and if that is not set, the value ''unknown-8bit'' is used. **WARNING:** this field applies to the sender's mailer, not the recipient's mailer. For example, if the envelope sender address lists an address on the local network and the recipient is on an external network, the character set will be set from the Charset= field for the local network mailer, not that of the external network mailer.

The Type= field sets the type information used in MIME error messages as defined by RFC XXX (not yet published). It is actually three values separated by slashes: the MTA-type (that is, the description of how hosts are named), the address type (the description of e-mail addresses), and the diagnostic type (the description of error diagnostic codes). Each of these must be a registered value or begin with ''X–''. The default is ''dns/rfc822/smtp''.

5.5. H — Define Header

The format of the header lines that *sendmail* inserts into the message are defined by the **H** line. The syntax of this line is:

H[**?***mflags***?**]*hname***:** *htemplate*

Continuation lines in this spec are reflected directly into the outgoing message. The *htemplate* is macro expanded before insertion into the message. If the *mflags* (surrounded by question marks) are specified, at least one of the specified flags must be stated in the mailer definition for this header to be automatically output. If one of these headers is in the input it is reflected to the output regardless of these flags.

Some headers have special semantics that will be described later.

5.6. O — Set Option

There are a number of global options that can be set from a configuration file. Options are represented by full words; some are also representable as single characters for back compatibility. The syntax of this line is:

O *option***=***value*

This sets option *option* to be *value*. Note that there *must* be a space between the letter 'O' and the name of the option. An older version is:

O*o value*

where the option *o* is a single character. Depending on the option, *value* may be a string, an integer, a boolean (with legal values ''t'', ''T'', ''f'', or ''F''; the default is TRUE), or a time interval.

The options supported (with the old, one character names in brackets) are:

AliasFile=*spec, spec, ...*
: [A] Specify possible alias file(s). Each *spec* should be in the format ‘‘*class***:** *file*’’ where *class***:** is optional and defaults to ‘‘implicit’’. Depending on how *sendmail* is compiled, valid classes are ‘‘implicit’’ (search through a compiled-in list of alias file types, for back compatibility), ‘‘hash’’ (if NEWDB is specified), ‘‘dbm’’ (if NDBM is specified), ‘‘stab’’ (internal symbol table — not normally used unless you have no other database lookup), or ‘‘nis’’ (if NIS is specified). If a list of *spec*s are provided, *sendmail* searches them in order.

AliasWait=*timeout*
: [a] If set, wait up to *timeout* (units default to minutes) for an ‘‘@:@’’ entry to exist in the alias database before starting up. If it does not appear in the *timeout* interval rebuild the database (if the **AutoRebuildAliases** option is also set) or issue a warning.

AutoRebuildAliases
: [D] If set, rebuild the alias database if necessary and possible. If this option is not set, *sendmail* will never rebuild the alias database unless explicitly requested using −**bi**. Not recommended — can cause thrashing.

BlankSub=*c*
: [B] Set the blank substitution character to *c*. Unquoted spaces in addresses are replaced by this character. Defaults to space (i.e., no change is made).

CheckAliases
: [n] Validate the RHS of aliases when rebuilding the alias database.

CheckpointInterval=*N*
: [C] Checkpoints the queue every *N* (default 10) addresses sent. If your system crashes during delivery to a large list, this prevents retransmission to any but the last recipients.

ClassFactor=*fact*
: [z] The indicated *fact*or is multiplied by the message class (determined by the Precedence: field in the user header and the **P** lines in the configuration file) and subtracted from the priority. Thus, messages with a higher Priority: will be favored. Defaults to 1800.

ColonOkInAddr
: [no short name] If set, colons are acceptable in e-mail addresses (e.g., ‘‘host:user’’). If not set, colons indicate the beginning of a RFC 822 group construct (‘‘groupname: member1, member2, ... memberN;’’). Doubled colons are always acceptable (‘‘nodename::user’’) and proper route-addr nesting is understood (‘‘<@relay:user@host>’’). Furthermore, this option defaults on if the configuration version level is less than 6 (for back compatibility). However, it must be off for full compatibility with RFC 822.

ConnectionCacheSize=*N*
: [k] The maximum number of open connections that will be cached at a time. The default is one. This delays closing the current connection until either this invocation of *sendmail* needs to connect to another host or it terminates. Setting it to zero defaults to the old behavior, that is, connections are closed immediately. Since this consumes file descriptors, the connection cache should be kept small: 4 is probably a practical maximum.

ConnectionCacheTimeout=*timeout*
: [K] The maximum amount of time a cached connection will be permitted to idle without activity. If this time is exceeded, the connection is immediately closed. This value should be small (on the order of ten minutes). Before *sendmail* uses a cached connection, it always sends a RSET command to check the connection; if this fails, it reopens the connection. This keeps your end from failing if the other end times out. The point of this option is to be a good network neighbor and avoid using up excessive resources on the other end. The default is five minutes.

DaemonPortOptions=*options*
: [O] Set server SMTP options. The options are *key=value* pairs. Known keys are:

Port	Name/number of listening port (defaults to "smtp")
Addr	Address mask (defaults INADDR_ANY)
Family	Address family (defaults to INET)
Listen	Size of listen queue (defaults to 10)
SndBufSize	Size of TCP send buffer
RcvBufSize	Size of TCP receive buffer

The *Addr*ess mask may be a numeric address in dot notation or a network name.

DefaultCharSet=*charset*
[no short name] When a message that has 8-bit characters but is not in MIME format is converted to MIME (see the EightBitMode option) a character set must be included in the Content-Type: header. This character set is normally set from the Charset= field of the mailer descriptor. If that is not set, the value of this option is used. If this option is not set, the value ‘‘unknown-8bit’’ is used.

DefaultUser=*user:group*
[u] Set the default userid for mailers to *user:group*. If *group* is omitted and *user* is a user name (as opposed to a numeric user id) the default group listed in the /etc/passwd file for that user is used as the default group. Both *user* and *group* may be numeric. Mailers without the *S* flag in the mailer definition will run as this user. Defaults to 1:1. The value can also be given as a symbolic user name.[19]

DeliveryMode=*x* [d] Deliver in mode *x*. Legal modes are:

i	Deliver interactively (synchronously)
b	Deliver in background (asynchronously)
q	Just queue the message (deliver during queue run)

Defaults to ‘‘b’’ if no option is specified, ‘‘i’’ if it is specified but given no argument (i.e., ‘‘Od’’ is equivalent to ‘‘Odi’’). The **–v** command line flag sets this to **i**.

DialDelay=*sleeptime*
[no short name] Dial-on-demand network connections can see timeouts if a connection is opened before the call is set up. If this is set to an interval and a connection times out on the first connection being attempted *sendmail* will sleep for this amount of time and try again. This should give your system time to establish the connection to your service provider. Units default to seconds, so ‘‘DialDelay=5’’ uses a five second delay. Defaults to zero (no retry).

DontExpandCnames
[no short name] The standards say that all host addresses used in a mail message must be fully canonical. For example, if your host is named ‘‘Cruft.Foo.ORG’’ and also has an alias of ‘‘FTP.Foo.ORG’’, the former name must be used at all times. This is enforced during host name canonification ($[... $] lookups). If this option is set, the protocols are ignored and the ‘‘wrong’’ thing is done. However, the IETF is moving toward changing this standard, so the behaviour may become acceptable. Please note that hosts downstream may still rewrite the address to be the true canonical name however.

DontPruneRoutes [R] Normally, *sendmail* tries to eliminate any unnecessary explicit routes when sending an error message (as discussed in RFC 1123 § 5.2.6). For example, when sending an error message to

<@known1,@known2,@known3:user@unknown>

sendmail will strip off the ‘‘@known1,@known2’’ in order to make the route as direct as possible. However, if the **R** option is set, this will be disabled, and the mail will be sent to the first address in the route, even if later addresses are

[19] The old **g** option has been combined into the **DefaultUser** option.

known. This may be useful if you are caught behind a firewall.

EightBitMode=*action*
: [8] Set handling of eight-bit data. There are two kinds of eight-bit data: that declared as such using the **BODY=8BITMIME** ESMTP declaration or the **–B8BITMIME** command line flag, and undeclared 8-bit data, that is, input that just happens to be eight bits. There are three basic operations that can happen: undeclared 8-bit data can be automatically converted to 8BITMIME, undeclared 8-bit data can be passed as-is without conversion to MIME (‘‘just send 8’’), and declared 8-bit data can be converted to 7-bits for transmission to a non-8BITMIME mailer. The possible *action*s are:

    s Reject undeclared 8-bit data (‘‘strict’’)
    m Convert undeclared 8-bit data to MIME (‘‘mime’’)
    p Pass undeclared 8-bit data (‘‘pass’’)

  In all cases properly declared 8BITMIME data will be converted to 7BIT as needed.

ErrorHeader=*file-or-message*
: [E] Prepend error messages with the indicated message. If it begins with a slash, it is assumed to be the pathname of a file containing a message (this is the recommended setting). Otherwise, it is a literal message. The error file might contain the name, email address, and/or phone number of a local postmaster who could provide assistance in to end users. If the option is missing or null, or if it names a file which does not exist or which is not readable, no message is printed.

ErrorMode=*x*
: [e] Dispose of errors using mode *x*. The values for *x* are:

    p Print error messages (default)
    q No messages, just give exit status
    m Mail back errors
    w Write back errors (mail if user not logged in)
    e Mail back errors and give zero exit stat always

FallbackMXhost=*fallbackhost*
: [V] If specified, the *fallbackhost* acts like a very low priority MX on every host. This is intended to be used by sites with poor network connectivity.

ForkEachJob
: [Y] If set, deliver each job that is run from the queue in a separate process. Use this option if you are short of memory, since the default tends to consume considerable amounts of memory while the queue is being processed.

ForwardPath=*path*
: [J] Set the path for searching for users’ .forward files. The default is ‘‘$z/.forward’’. Some sites that use the automounter may prefer to change this to ‘‘/var/forward/$u’’ to search a file with the same name as the user in a system directory. It can also be set to a sequence of paths separated by colons; *sendmail* stops at the first file it can successfully and safely open. For example, ‘‘/var/forward/$u:$z/.forward’’ will search first in /var/forward/*username* and then in ˜*username*/.forward (but only if the first file does not exist).

HelpFile=*file*
: [H] Specify the help file for SMTP.

HoldExpensive
: [c] If an outgoing mailer is marked as being expensive, don’t connect immediately. This requires that queueing be compiled in, since it will depend on a queue run process to actually send the mail.

IgnoreDots
: [i] Ignore dots in incoming messages. This is always disabled (that is, dots are always accepted) when reading SMTP mail.

LogLevel=*n*
: [L] Set the default log level to *n*. Defaults to 9.

M*x value*
: [no long version] Set the macro *x* to *value*. This is intended only for use from the command line. The **–M** flag is preferred.

MatchGECOS [G] Allow fuzzy matching on the GECOS field. If this flag is set, and the usual user name lookups fail (that is, there is no alias with this name and a *getpwnam* fails), sequentially search the password file for a matching entry in the GECOS field. This also requires that MATCHGECOS be turned on during compilation. This option is not recommended.

MaxHopCount=*N*
[h] The maximum hop count. Messages that have been processed more than *N* times are assumed to be in a loop and are rejected. Defaults to 25.

MaxHostStatAge=*age*
[no short name] Not yet implemented. This option specifies how long host status information will be retained. For example, if a host is found to be down, connections to that host will not be retried for this interval. The units default to minutes.

MaxQueueRunSize=*N*
[no short name] The maximum number of jobs that will be processed in a single queue run. If not set, there is no limit on the size. If you have very large queues or a very short queue run interval this could be unstable. However, since the first *N* jobs in queue directory order are run (rather than the *N* highest priority jobs) this should be set as high as possible to avoid ‘‘losing’’ jobs that happen to fall late in the queue directory.

MeToo [m] Send to me too, even if I am in an alias expansion.

MaxMessageSize=*N*
[no short name] Specify the maximum message size to be advertised in the ESMTP EHLO response. Messages larger than this will be rejected.

MinFreeBlocks=*N*
[b] Insist on at least *N* blocks free on the filesystem that holds the queue files before accepting email via SMTP. If there is insufficient space *sendmail* gives a 452 response to the MAIL command. This invites the sender to try again later.

MinQueueAge=age
[no short name] Don’t process any queued jobs that have been in the queue less than the indicated time interval. This is intended to allow you to get responsiveness by processing the queue fairly frequently without thrashing your system by trying jobs too often. The default units are minutes.

ResolverOptions=*options*
[I] Set resolver options. Values can be set using +*flag* and cleared using –*flag*; the *flag*s can be ‘‘debug’’, ‘‘aaonly’’, ‘‘usevc’’, ‘‘primary’’, ‘‘igntc’’, ‘‘recurse’’, ‘‘defnames’’, ‘‘stayopen’’, or ‘‘dnsrch’’. The string ‘‘HasWildcardMX’’ (without a + or –) can be specified to turn off matching against MX records when doing name canonifications. **N.B.** Prior to 8.7, this option indicated that the name server be responding in order to accept addresses. This has been replaced by checking to see if the ‘‘dns’’ method is listed in the service switch entry for the ‘‘hosts’’ service.

NoRecipientAction
[no short name] The action to take when you receive a message that has no valid recipient headers (To:, Cc:, Bcc:). It can be **None** to pass the message on unmodified, which violates the protocol, **Add-To** to add a To: header with any recipients it can find in the envelope (which might expose Bcc: recipients), **Add-Apparently-To** to add an Apparently-To: header (this is only for back-compatibility and is officially deprecated), **Add-To-Undisclosed** to add a header ‘‘To: undisclosed-recipients:;’’ to make the header legal without disclosing anything, or **Add-Bcc** to add an empty Bcc: header.

OldStyleHeaders [o] Assume that the headers may be in old format, i.e., spaces delimit names. This actually turns on an adaptive algorithm: if any recipient address contains a comma, parenthesis, or angle bracket, it will be assumed that commas already exist. If this flag is not on, only commas delimit names. Headers are always output with commas between the names. Defaults to off.

PostmasterCopy=*postmaster*
: [P] If set, copies of error messages will be sent to the named *postmaster*. Only the header of the failed message is sent. Since most errors are user problems, this is probably not a good idea on large sites, and arguably contains all sorts of privacy violations, but it seems to be popular with certain operating systems vendors. Defaults to no postmaster copies.

PrivacyOptions= *opt,opt,...*
: [p] Set the privacy *opt*ions. ''Privacy'' is really a misnomer; many of these are just a way of insisting on stricter adherence to the SMTP protocol. The *opt*ions can be selected from:

public	Allow open access
needmailhelo	Insist on HELO or EHLO command before MAIL
needexpnhelo	Insist on HELO or EHLO command before EXPN
noexpn	Disallow EXPN entirely
needvrfyhelo	Insist on HELO or EHLO command before VRFY
novrfy	Disallow VRFY entirely
restrictmailq	Restrict mailq command
restrictqrun	Restrict −q command line flag
noreceipts	Ignore Return-Receipt-To: header
goaway	Disallow essentially all SMTP status queries
authwarnings	Put X-Authentication-Warning: headers in messages

The ''goaway'' pseudo-flag sets all flags except ''restrictmailq'' and ''restrictqrun''. If mailq is restricted, only people in the same group as the queue directory can print the queue. If queue runs are restricted, only root and the owner of the queue directory can run the queue. Authentication Warnings add warnings about various conditions that may indicate attempts to spoof the mail system, such as using an non-standard queue directory.

QueueDirectory=*dir*
: [Q] Use the named *dir* as the queue directory.

QueueFactor=*factor*
: [q] Use *factor* as the multiplier in the map function to decide when to just queue up jobs rather than run them. This value is divided by the difference between the current load average and the load average limit (**QueueLA** option) to determine the maximum message priority that will be sent. Defaults to 600000.

QueueLA=*LA*
: [x] When the system load average exceeds *LA*, just queue messages (i.e., don't try to send them). Defaults to 8.

QueueSortOrder=*algorithm*
: [no short name] Sets the *algorithm* used for sorting the queue. Only the first character of the value is used. Legal values are ''host'' (to order by the name of the first host name of the first recipient) and ''priority'' (to order strictly by message priority). Host ordering makes better use of the connection cache, but may tend to process low priority messages that go to a single host over high priority messages that go to several hosts; it probably shouldn't be used on slow network links. Priority ordering is the default.

Timeout.*type*= *timeout*
: [r; subsumes old T option as well] Set timeout values. The actual timeout is indicated by the *type*. The recognized timeouts and their default values, and their minimum values specified in RFC 1123 section 5.3.2 are:

initial	wait for initial greeting message [5m, 5m]
helo	reply to HELO or EHLO command [5m, none]
mail	reply to MAIL command [10m, 5m]
rcpt	reply to RCPT command [1h, 5m]
datainit	reply to DATA command [5m, 2m]
datablock	data block read [1h, 3m]
datafinal	reply to final ‘‘.’’ in data [1h, 10m]
rset	reply to RSET command [5m, none]
quit	reply to QUIT command [2m, none]
misc	reply to NOOP and VERB commands [2m, none]
ident	IDENT protocol timeout [30s, none]
fileopen†	timeout on opening .forward and :include: files [60s, none]
command†	command read [1h, 5m]
queuereturn†	how long until a message is returned [5d, 5d]
queuewarn†	how long until a warning is sent [none, none]

All but those marked with a dagger (†) apply to client SMTP. If the message is submitted using the NOTIFY SMTP extension, warning messages will only be sent if NOTIFY=DELAY is specified. The queuereturn and queuewarn timeouts can be further qualified with a tag based on the Precedence: field in the message; they must be one of ‘‘urgent’’ (indicating a positive non-zero precedence) ‘‘normal’’ (indicating a zero precedence), or ‘‘non-urgent’’ (indicating negative precedences). For example, setting ‘‘Timeout.queuewarn.urgent=1h’’ sets the warning timeout for urgent messages only to one hour. The default if no precedence is indicated is to set the timeout for all precedences.

RecipientFactor=*fact*
[y] The indicated *fact*or is added to the priority (thus *lowering* the priority of the job) for each recipient, i.e., this value penalizes jobs with large numbers of recipients. Defaults to 30000.

RefuseLA=*LA* [X] When the system load average exceeds *LA*, refuse incoming SMTP connections. Defaults to 12.

RetryFactor=*fact* [Z] The *fact*or is added to the priority every time a job is processed. Thus, each time a job is processed, its priority will be decreased by the indicated value. In most environments this should be positive, since hosts that are down are all too often down for a long time. Defaults to 90000.

SaveFromLine [f] Save Unix-style ‘‘From’’ lines at the front of headers. Normally they are assumed redundant and discarded.

SendMIMEErrors
[j] If set, send error messages in MIME format (see RFC1521 and RFC1344 for details).

ServiceSwitchFile=*filename*
[no short name] If your host operating system has a service switch abstraction (e.g., /etc/nsswitch.conf on Solaris or /etc/svc.conf on Ultrix and DEC OSF/1) that service will be consulted and this option is ignored. Otherwise, this is the name of a file that provides the list of methods used to implement particular services. The syntax is a series of lines, each of which is a sequence of words. The first word is the service name, and following words are service types. The services that *sendmail* consults directly are ‘‘aliases’’ and ‘‘hosts.’’ Service types can be ‘‘dns’’, ‘‘nis’’, ‘‘nisplus’’, or ‘‘files’’ (with the caveat that the appropriate support must be compiled in before the service can be referenced). If ServiceSwitchFile is not specified, it defaults to /etc/service.switch. If that file does not exist, the default switch is:

aliases	files
hosts	dns nis files

The default file is ''/etc/service.switch''.

SevenBitInput [7] Strip input to seven bits for compatibility with old systems. This shouldn't be necessary.

StatusFile=*file* [S] Log summary statistics in the named *file*. If not set, no summary statistics are saved. This file does not grow in size. It can be printed using the *mailstats*(8) program.

SuperSafe [s] Be super-safe when running things, i.e., always instantiate the queue file, even if you are going to attempt immediate delivery. *Sendmail* always instantiates the queue file before returning control the client under any circumstances. This should really *always* be set.

TempFileMode=*mode*
[F] The file mode for queue files. It is interpreted in octal by default. Defaults to 0600.

TimeZoneSpec=*tzinfo*
[t] Set the local time zone info to *tzinfo* — for example, ''PST8PDT''. Actually, if this is not set, the TZ environment variable is cleared (so the system default is used); if set but null, the user's TZ variable is used, and if set and non-null the TZ variable is set to this value.

TryNullMXList [w] If this system is the ''best'' (that is, lowest preference) MX for a given host, its configuration rules should normally detect this situation and treat that condition specially by forwarding the mail to a UUCP feed, treating it as local, or whatever. However, in some cases (such as Internet firewalls) you may want to try to connect directly to that host as though it had no MX records at all. Setting this option causes *sendmail* to try this. The downside is that errors in your configuration are likely to be diagnosed as ''host unknown'' or ''message timed out'' instead of something more meaningful. This option is disrecommended.

UseErrorsTo [l] If there is an ''Errors-To:'' header, send error messages to the addresses listed there. They normally go to the envelope sender. Use of this option causes *sendmail* to violate RFC 1123. This option is disrecommended and deprecated.

UserDatabaseSpec=*udbspec*
[U] The user database specification.

Verbose [v] Run in verbose mode. If this is set, *sendmail* adjusts options **HoldExpensive** (old **c**) and **DeliveryMode** (old **d**) so that all mail is delivered completely in a single job so that you can see the entire delivery process. Option **Verbose** should *never* be set in the configuration file; it is intended for command line use only.

All options can be specified on the command line using the –O or –o flag, but most will cause *sendmail* to relinquish its setuid permissions. The options that will not cause this are MinFreeBlocks [b], DeliveryMode [d], ErrorMode [e], IgnoreDots [i], LogLevel [L], MeToo [m], OldStyleHeaders [o], PrivacyOptions [p], Timeouts [r], SuperSafe [s], Verbose [v], CheckpointInterval [C], and SevenBitInput [7]. Also, M (define macro) when defining the r or s macros is also considered ''safe''.

5.7. P — Precedence Definitions

Values for the ''Precedence:'' field may be defined using the **P** control line. The syntax of this field is:

P*name***=***num*

When the *name* is found in a ''Precedence:'' field, the message class is set to *num*. Higher numbers mean higher precedence. Numbers less than zero have the special property that if an error occurs during processing the body of the message will not be returned; this is expected to be used for ''bulk'' mail such as through mailing lists. The default precedence is zero. For example, our list of precedences is:

```
Pfirst-class=0
Pspecial-delivery=100
Plist=−30
Pbulk=−60
Pjunk=−100
```

People writing mailing list exploders are encouraged to use ‘‘Precedence: list’’. Older versions of *sendmail* (which discarded all error returns for negative precedences) didn’t recognize this name, giving it a default precedence of zero. This allows list maintainers to see error returns on both old and new versions of *sendmail*.

5.8. V — Configuration Version Level

To provide compatibility with old configuration files, the **V** line has been added to define some very basic semantics of the configuration file. These are not intended to be long term supports; rather, they describe compatibility features which will probably be removed in future releases.

N.B.: these version *levels* have nothing to do with the version *number* on the files. For example, as of this writing version 8 config files (specifically, 8.7) used version level 6 configurations.

‘‘Old’’ configuration files are defined as version level one. Version level two files make the following changes:

(1) Host name canonification ($[... $]) appends a dot if the name is recognized; this gives the config file a way of finding out if anything matched. (Actually, this just initializes the ‘‘host’’ map with the ‘‘−a.’’ flag — you can reset it to anything you prefer by declaring the map explicitly.)

(2) Default host name extension is consistent throughout processing; version level one configurations turned off domain extension (that is, adding the local domain name) during certain points in processing. Version level two configurations are expected to include a trailing dot to indicate that the name is already canonical.

(3) Local names that are not aliases are passed through a new distinguished ruleset five; this can be used to append a local relay. This behaviour can be prevented by resolving the local name with an initial ‘@’. That is, something that resolves to a local mailer and a user name of ‘‘vikki’’ will be passed through ruleset five, but a user name of ‘‘@vikki’’ will have the ‘@’ stripped, will not be passed through ruleset five, but will otherwise be treated the same as the prior example. The expectation is that this might be used to implement a policy where mail sent to ‘‘vikki’’ was handled by a central hub, but mail sent to ‘‘vikki@localhost’’ was delivered directly.

Version level three files allow # initiated comments on all lines. Exceptions are backslash escaped # marks and the $# syntax.

Version level four configurations are completely equivalent to level three for historical reasons.

Version level five configuration files change the default definition of **$w** to be just the first component of the hostname.

Version level six configuration files change many of the local processing options (such as aliasing and matching the beginning of the address for ‘|’ characters) to be mailer flags; this allows fine-grained control over the special local processing. Level six configuration files may also use long option names. The **ColonOkInAddr** option (to allow colons in the local-part of addresses) defaults **on** for lower numbered configuration files; the configuration file requires some additional intelligence to properly handle the RFC 822 group construct.

The **V** line may have an optional */vendor* to indicate that this configuration file uses modifications specific to a particular vendor[20]. You may use ‘‘/Berkeley’’ to emphasize that this configuration file uses the Berkeley dialect of *sendmail*.

5.9. K — Key File Declaration

Special maps can be defined using the line:

Kmapname mapclass arguments

The *mapname* is the handle by which this map is referenced in the rewriting rules. The *mapclass* is the name of a type of map; these are compiled in to *sendmail*. The *arguments* are interpreted depending on the class; typically, there would be a single argument naming the file containing the map.

Maps are referenced using the syntax:

$(*map key* $@ *arguments* $: *default* $)

where either or both of the *arguments* or *default* portion may be omitted. The *$@ arguments* may appear more than once. The indicated *key* and *arguments* are passed to the appropriate mapping function. If it returns a value, it replaces the input. If it does not return a value and the *default* is specified, the *default* replaces the input. Otherwise, the input is unchanged.

During replacement of either a map value or default the string ‘‘%*n*’’ (where *n* is a digit) is replaced by the corresponding *argument*. Argument zero is always the database key. For example, the rule

```
R$– ! $+        $: $(uucp $1 $@ $2 $: %1 @ %0 . UUCP $)
```

Looks up the UUCP name in a (user defined) UUCP map; if not found it turns it into ‘‘.UUCP’’ form. The database might contain records like:

```
decvax          %1@%0.DEC.COM
research        %1@%0.ATT.COM
```

The built in map with both name and class ‘‘host’’ is the host name canonicalization lookup. Thus, the syntax:

$(host *hostname*$)

is equivalent to:

$[*hostname*$]

There are many defined classes.

dbm	Database lookups using the ndbm(3) library. *Sendmail* must be compiled with **DBM** defined.
btree	Database lookups using the btree interface to the Berkeley db(3) library. *Sendmail* must be compiled with **NEWDB** defined.
hash	Database lookups using the hash interface to the Berkeley db(3) library. *Sendmail* must be compiled with **NEWDB** defined.
nis	NIS lookups. *Sendmail* must be compiled with **NIS** defined.
nisplus	NIS+ lookups. *Sendmail* must be compiled with **NISPLUS** defined. The argument is the name of the table to use for lookups, and the **–k** and **–v** flags may be used to set the key and value columns respectively.
hesiod	Hesiod lookups. *Sendmail* must be compiled with **HESIOD** defined.
netinfo	NeXT NetInfo lookups. *Sendmail* must be compiled with **NETINFO** defined.

[20]And of course, vendors are encouraged to add themselves to the list of recognized vendors by editing the routine *setvendor* in *conf.c*. Please send e-mail to sendmail@CS.Berkeley.EDU to register your vendor dialect.

text — Text file lookups. The format of the text file is defined by the −**k** (key field number), −**v** (value field number), and −**z** (field delimiter) flags.

stab — Internal symbol table lookups. Used internally for aliasing.

implicit — Really should be called ‘‘alias’’ — this is used to get the default lookups for alias files, and is the default if no class is specified for alias files.

user — Looks up users using *getpwnam*(3). The −**v** flag can be used to specify the name of the field to return (although this is normally used only to check the existence of a user).

host — Canonifies host domain names. Given a host name it calls the name server to find the canonical name for that host.

sequence — The arguments on the ‘K’ line are a list of maps; the resulting map searches the argument maps in order until it finds a match for the indicated key. For example, if the key definition is:

```
Kmap1 ...
Kmap2 ...
Kseqmap sequence map1 map2
```

then a lookup against ‘‘seqmap’’ first does a lookup in map1. If that is found, it returns immediately. Otherwise, the same key is used for map2.

switch — Much like the ‘‘sequence’’ map except that the order of maps is determined by the service switch. The argument is the name of the service to be looked up; the values from the service switch are appended to the service name to create new map names. For example, consider the key definition:

```
Kali switch aliases
```

together with the service switch entry:

```
aliases		nis files
```

This causes a query against the map ‘‘ali’’ to search maps named ‘‘aliases.nis’’ and ‘‘aliases.files’’ in that order.

dequote — Strip double quotes (") from a name. It does not strip backslashes, and will not strip quotes if the resulting string would contain unscannable syntax (that is, basic errors like unbalanced angle brackets; more sophisticated errors such as unknown hosts are not checked). The intent is for use when trying to accept mail from systems such as DECnet that routinely quote odd syntax such as

```
"49ers::ubell"
```

A typical usage is probably something like:

```
Kdequote dequote

...

R$−		$: $(dequote $1 $)
R$− $+		$: $>3 $1 $2
```

Care must be taken to prevent unexpected results; for example,

```
"|someprogram < input > output"
```

will have quotes stripped, but the result is probably not what you had in mind. Fortunately these cases are rare.

Most of these accept as arguments the same optional flags and a filename (or a mapname for NIS; the filename is the root of the database path, so that ‘‘.db’’ or some other extension appropriate for the database type will be added to get the actual database name). Known flags are:

−o — Indicates that this map is optional — that is, if it cannot be opened, no error is produced, and *sendmail* will behave as if the map existed but was empty.

−N, −O	If neither **−N** or **−O** are specified, *sendmail* uses an adaptive algorithm to decide whether or not to look for null bytes on the end of keys. It starts by trying both; if it finds any key with a null byte it never tries again without a null byte and vice versa. If **−N** is specified it never tries without a null byte and if **−O** is specified it never tries with a null byte. Setting one of these can speed matches but are never necessary. If both **−N** and **−O** are specified, *sendmail* will never try any matches at all — that is, everything will appear to fail.
−a*x*	Append the string *x* on successful matches. For example, the default *host* map appends a dot on successful matches.
−f	Do not fold upper to lower case before looking up the key.
−m	Match only (without replacing the value). If you only care about the existence of a key and not the value (as you might when searching the NIS map ‘‘hosts.byname’’ for example), this flag prevents the map from substituting the value. However, The −a argument is still appended on a match, and the default is still taken if the match fails.
−k*keycol*	The key column name (for NIS+) or number (for text lookups).
−v*valcol*	The value column name (for NIS+) or number (for text lookups).
−z*delim*	The column delimiter (for text lookups). It can be a single character or one of the special strings ‘‘\n’’ or ‘‘\t’’ to indicate newline or tab respectively. If omitted entirely, the column separator is any sequence of whitespace.
−s*spacesub*	For the dequote map only, the character to use to replace space characters after a successful dequote.

The *dbm* map appends the strings ‘‘.pag’’ and ‘‘.dir’’ to the given filename; the two *db*-based maps append ‘‘.db’’. For example, the map specification

    Kuucp dbm −o −N /usr/lib/uucpmap

specifies an optional map named ‘‘uucp’’ of class ‘‘dbm’’; it always has null bytes at the end of every string, and the data is located in /usr/lib/uucpmap.{dir,pag}.

The program *makemap*(8) can be used to build any of the three database-oriented maps. It takes the following flags:

−f	Do not fold upper to lower case in the map.
−N	Include null bytes in keys.
−o	Append to an existing (old) file.
−r	Allow replacement of existing keys; normally, re-inserting an existing key is an error.
−v	Print what is happening.

The *sendmail* daemon does not have to be restarted to read the new maps as long as you change them in place; file locking is used so that the maps won’t be read while they are being updated.[21]

New classes can be added in the routine **setupmaps** in file **conf.c**.

5.10. The User Database

If you have a version of *sendmail* with the user database package compiled in, the handling of sender and recipient addresses is modified.

The location of this database is controlled with the **UserDatabaseSpec** option.

---

[21] That is, don’t create new maps and then use *mv*(1) to move them into place. Since the maps are already open the new maps will never be seen.

5.10.1. Structure of the user database

The database is a sorted (BTree-based) structure. User records are stored with the key:

*user-name***:***field-name*

The sorted database format ensures that user records are clustered together. Meta-information is always stored with a leading colon.

Field names define both the syntax and semantics of the value. Defined fields include:

maildrop	The delivery address for this user. There may be multiple values of this record. In particular, mailing lists will have one *maildrop* record for each user on the list.
mailname	The outgoing mailname for this user. For each outgoing name, there should be an appropriate *maildrop* record for that name to allow return mail. See also *:default:mailname*.
mailsender	Changes any mail sent to this address to have the indicated envelope sender. This is intended for mailing lists, and will normally be the name of an appropriate -request address. It is very similar to the owner-*list* syntax in the alias file.
fullname	The full name of the user.
office-address	The office address for this user.
office-phone	The office phone number for this user.
office-fax	The office FAX number for this user.
home-address	The home address for this user.
home-phone	The home phone number for this user.
home-fax	The home FAX number for this user.
project	A (short) description of the project this person is affiliated with. In the University this is often just the name of their graduate advisor.
plan	A pointer to a file from which plan information can be gathered.

As of this writing, only a few of these fields are actually being used by *sendmail*: *maildrop* and *mailname*. A *finger* program that uses the other fields is planned.

5.10.2. User database semantics

When the rewriting rules submit an address to the local mailer, the user name is passed through the alias file. If no alias is found (or if the alias points back to the same address), the name (with ‘‘:maildrop’’ appended) is then used as a key in the user database. If no match occurs (or if the maildrop points at the same address), forwarding is tried.

If the first token of the user name returned by ruleset 0 is an ‘‘@’’ sign, the user database lookup is skipped. The intent is that the user database will act as a set of defaults for a cluster (in our case, the Computer Science Division); mail sent to a specific machine should ignore these defaults.

When mail is sent, the name of the sending user is looked up in the database. If that user has a ‘‘mailname’’ record, the value of that record is used as their outgoing name. For example, I might have a record:

```
eric:mailname          Eric.Allman@CS.Berkeley.EDU
```

This would cause my outgoing mail to be sent as Eric.Allman.

If a ‘‘maildrop’’ is found for the user, but no corresponding ‘‘mailname’’ record exists, the record ‘‘:default:mailname’’ is consulted. If present, this is the name of a host to override the local host. For example, in our case we would set it to ‘‘CS.Berkeley.EDU’’. The effect is that anyone known in the database gets their outgoing mail stamped as ‘‘user@CS.Berkeley.EDU’’, but people not listed in the database use the local hostname.

5.10.3. Creating the database[22]

The user database is built from a text file using the *makemap* utility (in the distribution in the makemap subdirectory). The text file is a series of lines corresponding to userdb records; each line has a key and a value separated by white space. The key is always in the format described above — for example:

eric:maildrop

This file is normally installed in a system directory; for example, it might be called */etc/userdb*. To make the database version of the map, run the program:

makemap btree /etc/userdb.db < /etc/userdb

Then create a config file that uses this. For example, using the V8 M4 configuration, include the following line in your .mc file:

define(`confUSERDB_SPEC´, /etc/userdb.db)

6. OTHER CONFIGURATION

There are some configuration changes that can be made by recompiling *sendmail*. This section describes what changes can be made and what has to be modified to make them. In most cases this should be unnecessary unless you are porting *sendmail* to a new environment.

6.1. Parameters in src/Makefile

These parameters are intended to describe the compilation environment, not site policy, and should normally be defined in src/Makefile.

NDBM — If set, the new version of the DBM library that allows multiple databases will be used. If neither NDBM nor NEWDB are set, a much less efficient method of alias lookup is used.

NEWDB — If set, use the new database package from Berkeley (from 4.4BSD). This package is substantially faster than DBM or NDBM. If NEWDB and NDBM are both set, *sendmail* will read DBM files, but will create and use NEWDB files.

NIS — Include support for NIS. If set together with *both* NEWDB and NDBM, *sendmail* will create both DBM and NEWDB files if and only if an alias file includes the substring ‘‘/yp/’’ in the name. This is intended for compatibility with Sun Microsystems’ *mkalias* program used on YP masters.

NISPLUS — Compile in support for NIS+.

NETINFO — Compile in support for NetInfo (NeXT stations).

HESIOD — Compile in support for Hesiod.

_PATH_SENDMAILCF — The pathname of the sendmail.cf file.

_PATH_SENDMAILPID — The pathname of the sendmail.pid file.

There are also several compilation flags to indicate the environment such as ‘‘_AIX3’’ and ‘‘_SCO_unix_’’. See the READ_ME file for the latest scoop on these flags.

6.2. Parameters in src/conf.h

Parameters and compilation options are defined in conf.h. Most of these need not normally be tweaked; common parameters are all in sendmail.cf. However, the sizes of certain primitive vectors, etc., are included in this file. The numbers following the parameters are their default value.

[22]These instructions are known to be incomplete. A future version of the user database is planned including things such as finger service — and good documentation.

MAXLINE [1024] The maximum line length of any input line. If message lines exceed this length they will still be processed correctly; however, header lines, configuration file lines, alias lines, etc., must fit within this limit.

MAXNAME [256] The maximum length of any name, such as a host or a user name.

MAXPV [40] The maximum number of parameters to any mailer. This limits the number of recipients that may be passed in one transaction. It can be set to any arbitrary number above about 10, since *sendmail* will break up a delivery into smaller batches as needed. A higher number may reduce load on your system, however.

MAXATOM [100] The maximum number of atoms (tokens) in a single address. For example, the address ‘‘eric@CS.Berkeley.EDU’’ is seven atoms.

MAXMAILERS [25]
The maximum number of mailers that may be defined in the configuration file.

MAXRWSETS [200]
The maximum number of rewriting sets that may be defined. The first half of these are reserved for numeric specification (e.g., ‘‘S92’’), while the upper half are reserved for auto-numbering (e.g., ‘‘Sfoo’’). Thus, with a value of 200 an attempt to use ‘‘S99’’ will succeed, but ‘‘S100’’ will fail.

MAXPRIORITIES [25]
The maximum number of values for the ‘‘Precedence:’’ field that may be defined (using the **P** line in sendmail.cf).

MAXUSERENVIRON [40]
The maximum number of items in the user environment that will be passed to subordinate mailers.

MAXMXHOSTS [20]
The maximum number of MX records we will accept for any single host.

A number of other compilation options exist. These specify whether or not specific code should be compiled in. Ones marked with † are 0/1 valued.

DEBUG If set, debugging information is compiled in. To actually get the debugging output, the −**d** flag must be used. **WE STRONGLY RECOMMEND THAT THIS BE LEFT ON.** Some people, believing that it was a security hole (it was, once) have turned it off and thus crippled debuggers.

NETINET If set, support for Internet protocol networking is compiled in. Previous versions of *sendmail* referred to this as DAEMON; this old usage is now incorrect.

NETISO If set, support for ISO protocol networking is compiled in (it may be appropriate to #define this in the Makefile instead of conf.h).

LOG If set, the *syslog* routine in use at some sites is used. This makes an informational log record for each message processed, and makes a higher priority log record for internal system errors. **STRONGLY RECOMMENDED —** if you want no logging, turn it off in the configuration file.

MATCHGECOS Compile in the code to do ‘‘fuzzy matching’’ on the GECOS field in /etc/passwd. This also requires that the **MatchGECOS** option be turned on.

NAMED_BIND† Compile in code to use the Berkeley Internet Name Domain (BIND) server to resolve TCP/IP host names.

NOTUNIX If you are using a non-UNIX mail format, you can set this flag to turn off special processing of UNIX-style ‘‘From ’’ lines.

QUEUE This flag should be set to compile in the queueing code. If this is not set, mailers must accept the mail immediately or it will be returned to the sender.

SETPROCTITLE If defined, *sendmail* will change its *argv* array to indicate its current status. This can be used in conjunction with the *ps* command to find out just what it’s up to.

SMTP If set, the code to handle user and server SMTP will be compiled in. This is only necessary if your machine has some mailer that speaks SMTP (this means most machines everywhere).

UGLYUUCP If you have a UUCP host adjacent to you which is not running a reasonable version of *rmail*, you will have to set this flag to include the ‘‘remote from sysname’’ info on the from line. Otherwise, UUCP gets confused about where the mail came from.

USERDB Include the **experimental** Berkeley user information database package. This adds a new level of local name expansion between aliasing and forwarding. It also uses the NEWDB package. This may change in future releases.

The following options are normally turned on in per-operating-system clauses in conf.h.

IDENTPROTO† Compile in the IDENT protocol as defined in RFC 1413. This defaults on for all systems except Ultrix, which apparently has the interesting ‘‘feature’’ that when it receives a ‘‘host unreachable’’ message it closes all open connections to that host. Since some firewall gateways send this error code when you access an unauthorized port (such as 113, used by IDENT), Ultrix cannot receive email from such hosts.

SYSTEM5 Set all of the compilation parameters appropriate for System V.

LOCKF Use System V **lockf** instead of Berkeley **flock**. Due to the highly unusual semantics of locks across forks in **lockf**, this should never be used unless absolutely necessary. Set by default if SYSTEM5 is set.

SYS5TZ Use System V time zone semantics.

HASINITGROUPS Set this if your system has the *initgroups()* call (if you have multiple group support). This is the default if SYSTEM5 is *not* defined or if you are on HPUX.

HASUNAME Set this if you have the *uname*(2) system call (or corresponding library routine). Set by default if SYSTEM5 is set.

HASSTATFS Set this if you have the *statfs*(2) system call. This will allow you to give a temporary failure message to incoming SMTP email when you are low on disk space. It is set by default on 4.4BSD and OSF/1 systems.

HASUSTAT Set if you have the *ustat*(2) system call. This is an alternative implementation of disk space control. You should only set one of HASSTATFS or HASUSTAT; the first is preferred.

LA_TYPE The load average type. Details are described below.

The are several built-in ways of computing the load average. *Sendmail* tries to auto-configure them based on imperfect guesses; you can select one using the *cc* option **–DLA_TYPE=***type*, where *type* is:

LA_INT The kernel stores the load average in the kernel as an array of long integers. The actual values are scaled by a factor FSCALE (default 256).

LA_SHORT The kernel stores the load average in the kernel as an array of short integers. The actual values are scaled by a factor FSCALE (default 256).

LA_FLOAT The kernel stores the load average in the kernel as an array of double precision floats.

LA_MACH Use MACH-style load averages.

LA_SUBR Call the *getloadavg* routine to get the load average as an array of doubles.

LA_ZERO Always return zero as the load average. This is the fallback case.

If type LA_INT, LA_SHORT, or LA_FLOAT is specified, you may also need to specify _PATH_UNIX (the path to your system binary) and LA_AVENRUN (the name of the variable containing the load average in the kernel; usually ‘‘_avenrun’’ or ‘‘avenrun’’).

6.3. Configuration in src/conf.c

The following changes can be made in conf.c.

6.3.1. Built-in Header Semantics

Not all header semantics are defined in the configuration file. Header lines that should only be included by certain mailers (as well as other more obscure semantics) must be specified in the *HdrInfo* table in *conf.c*. This table contains the header name (which should be in all lower case) and a set of header control flags (described below), The flags are:

H_ACHECK — Normally when the check is made to see if a header line is compatible with a mailer, *sendmail* will not delete an existing line. If this flag is set, *sendmail* will delete even existing header lines. That is, if this bit is set and the mailer does not have flag bits set that intersect with the required mailer flags in the header definition in sendmail.cf, the header line is *always* deleted.

H_EOH — If this header field is set, treat it like a blank line, i.e., it will signal the end of the header and the beginning of the message text.

H_FORCE — Add this header entry even if one existed in the message before. If a header entry does not have this bit set, *sendmail* will not add another header line if a header line of this name already existed. This would normally be used to stamp the message by everyone who handled it.

H_TRACE — If set, this is a timestamp (trace) field. If the number of trace fields in a message exceeds a preset amount the message is returned on the assumption that it has an aliasing loop.

H_RCPT — If set, this field contains recipient addresses. This is used by the −**t** flag to determine who to send to when it is collecting recipients from the message.

H_FROM — This flag indicates that this field specifies a sender. The order of these fields in the *HdrInfo* table specifies *sendmail*’s preference for which field to return error messages to.

H_RECEIPTTO — This header has return-receipt information.

H_ERRORSTO — Addresses in this header should receive error messages.

H_CTE — This header is a Content-Transfer-Encoding header.

H_CTYPE — This header is a Content-Type header.

H_STRIPVAL — Strip the value from the header (for Bcc:).

Let’s look at a sample *HdrInfo* specification:

```
struct hdrinfo          HdrInfo[] =
{
        /* originator fields, most to least significant  */
    "resent-sender",    H_FROM,
    "resent-from",      H_FROM,
    "sender",           H_FROM,
    "from",             H_FROM,
    "full-name",        H_ACHECK,
    "return-receipt-to", H_FROM|H_RECEIPTTO,
    "errors-to",        H_FROM|H_ERRORSTO,
        /* destination fields */
    "to",               H_RCPT,
    "resent-to",        H_RCPT,
    "cc",               H_RCPT,
    "bcc",              H_RCPT|H_STRIPVAL,
        /* message identification and control */
    "message",          H_EOH,
    "text",             H_EOH,
        /* trace fields */
    "received",         H_TRACE|H_FORCE,
        /* miscellaneous fields */
    "content-transfer-encoding",H_CTE,
    "content-type",     H_CTYPE,

    NULL,               0,
};
```

This structure indicates that the ‘‘To:’’, ‘‘Resent-To:’’, and ‘‘Cc:’’ fields all specify recipient addresses. Any ‘‘Full-Name:’’ field will be deleted unless the required mailer flag (indicated in the configuration file) is specified. The ‘‘Message:’’ and ‘‘Text:’’ fields will terminate the header; these are used by random dissenters around the network world. The ‘‘Received:’’ field will always be added, and can be used to trace messages.

There are a number of important points here. First, header fields are not added automatically just because they are in the *HdrInfo* structure; they must be specified in the configuration file in order to be added to the message. Any header fields mentioned in the configuration file but not mentioned in the *HdrInfo* structure have default processing performed; that is, they are added unless they were in the message already. Second, the *HdrInfo* structure only specifies cliched processing; certain headers are processed specially by ad hoc code regardless of the status specified in *HdrInfo*. For example, the ‘‘Sender:’’ and ‘‘From:’’ fields are always scanned on ARPANET mail to determine the sender[23]; this is used to perform the ‘‘return to sender’’ function. The ‘‘From:’’ and ‘‘Full-Name:’’ fields are used to determine the full name of the sender if possible; this is stored in the macro **$x** and used in a number of ways.

6.3.2. Restricting Use of Email

If it is necessary to restrict mail through a relay, the *checkcompat* routine can be modified. This routine is called for every recipient address. It returns an exit status indicating the status of the message. The status EX_OK accepts the address, EX_TEMPFAIL queues the message for a later try, and other values (commonly EX_UNAVAILABLE) reject the message. It is up to *checkcompat* to print an error message (using *usrerr*) if the message is rejected. For example, *checkcompat* could read:

[23]Actually, this is no longer true in SMTP; this information is contained in the envelope. The older ARPANET protocols did not completely distinguish envelope from header.

```
int
checkcompat(to, e)
        register ADDRESS *to;
        register ENVELOPE *e;
{
        register STAB *s;

        s = stab("private", ST_MAILER, ST_FIND);
        if (s != NULL && e->e_from.q_mailer != LocalMailer &&
            to->q_mailer == s->s_mailer)
        {
                usrerr("No private net mail allowed through this machine");
                return (EX_UNAVAILABLE);
        }
        if (MsgSize > 50000 && bitnset(M_LOCALMAILER, to->q_mailer))
        {
                usrerr("Message too large for non-local delivery");
                e->e_flags |= EF_NORETURN;
                return (EX_UNAVAILABLE);
        }
        return (EX_OK);
}
```

This would reject messages greater than 50000 bytes unless they were local. The *EF_NORETURN* flag can be set in *e→e_flags* to suppress the return of the actual body of the message in the error return. The actual use of this routine is highly dependent on the implementation, and use should be limited.

6.3.3. Load Average Computation

The routine *getla* should return an approximation of the current system load average as an integer. There are several versions included on compilation flags as described above.

6.3.4. New Database Map Classes

New key maps can be added by creating a class initialization function and a lookup function. These are then added to the routine *setupmaps.*

The initialization function is called as

*xxx*_map_init(MAP *map, char *mapname, char *args)

The *map* is an internal data structure. The *mapname* is the name of the map (used for error messages). The *args* is a pointer to the rest of the configuration file line; flags and filenames can be extracted from this line. The initialization function must return TRUE if it successfully opened the map, FALSE otherwise.

The lookup function is called as

*xxx*_map_lookup(MAP *map, char buf[], int bufsize, char **av, int *statp)

The *map* defines the map internally. The parameters *buf* and *bufsize* have the input key. This may be (and often is) used destructively. The *av* is a list of arguments passed in from the rewrite line. The lookup function should return a pointer to the new value. IF the map lookup fails, **statp* should be set to an exit status code; in particular, it should be set to EX_TEMPFAIL if recovery is to be attempted by the higher level code.

6.3.5. Queueing Function

The routine *shouldqueue* is called to decide if a message should be queued or processed immediately. Typically this compares the message priority to the current load average. The default definition is:

```
bool
shouldqueue(pri, ctime)
	long pri;
	time_t ctime;
{
	if (CurrentLA < QueueLA)
		return (FALSE);
	if (CurrentLA >= RefuseLA)
		return (TRUE);
	return (pri > (QueueFactor / (CurrentLA − QueueLA + 1)));
}
```

If the current load average (global variable *CurrentLA*, which is set before this function is called) is less than the low threshold load average (option **x**, variable *QueueLA*), *shouldqueue* returns FALSE immediately (that is, it should *not* queue). If the current load average exceeds the high threshold load average (option **X**, variable *RefuseLA*), *shouldqueue* returns TRUE immediately. Otherwise, it computes the function based on the message priority, the queue factor (option **q**, global variable *QueueFactor*), and the current and threshold load averages.

An implementation wishing to take the actual age of the message into account can also use the *ctime* parameter, which is the time that the message was first submitted to *sendmail*. Note that the *pri* parameter is already weighted by the number of times the message has been tried (although this tends to lower the priority of the message with time); the expectation is that the *ctime* would be used as an ‘‘escape clause’’ to ensure that messages are eventually processed.

6.3.6. Refusing Incoming SMTP Connections

The function *refuseconnections* returns TRUE if incoming SMTP connections should be refused. The current implementation is based exclusively on the current load average and the refuse load average option (option **X**, global variable *RefuseLA*):

```
bool
refuseconnections()
{
	return (CurrentLA >= RefuseLA);
}
```

A more clever implementation could look at more system resources.

6.3.7. Load Average Computation

The routine *getla* returns the current load average (as a rounded integer). The distribution includes several possible implementations. If you are porting to a new environment you may need to add some new tweaks.[24]

6.4. Configuration in src/daemon.c

The file *src/daemon.c* contains a number of routines that are dependent on the local networking environment. The version supplied assumes you have BSD style sockets.

In previous releases, we recommended that you modify the routine *maphostname* if you wanted to generalize **$[... $]** lookups. We now recommend that you create a new keyed map instead.

[24] If you do, please send updates to sendmail@CS.Berkeley.EDU.

7. CHANGES IN VERSION 8

The following summarizes changes since the last commonly available version of *sendmail* (5.67). For a detailed list, consult the file RELEASE_NOTES in the root directory of the *sendmail* distribution.

7.1. Connection Caching

Instead of closing SMTP connections immediately, those connections are cached for possible future use. The advent of MX records made this effective for mailing lists; in addition, substantial performance improvements can be expected for queue processing.

7.2. MX Piggybacking

If two hosts with different names in a single message happen to have the same set of MX hosts, they can be sent in the same transaction. Version 8 notices this and tries to batch the messages.

7.3. RFC 1123 Compliance

A number of changes have been made to make *sendmail* ‘‘conditionally compliant’’ (that is, *sendmail* satisfies all of the ‘‘MUST’’ clauses and most but not all of the ‘‘SHOULD’’ clauses in RFC 1123).

The major areas of change are (numbers are RFC 1123 section numbers):

5.2.7 Response to RCPT command is fast.

5.2.8 Numeric IP addresses are logged in Received: lines.

5.2.17 Self domain literal is properly handled.

5.3.2 Better control over individual timeouts.

5.3.3 Error messages are sent as ‘‘From:<>’’.

5.3.3 Error messages are never sent to ‘‘<>’’.

5.3.3 Route-addrs are pruned.

The areas in which *sendmail* is not ‘‘unconditionally compliant’’ are:

5.2.6 *Sendmail* does do header munging.

5.2.10 *Sendmail* doesn’t always use the exact SMTP message text as listed in RFC 821.

5.3.1.1 *Sendmail* doesn’t guarantee only one connect for each host in queue runs.

5.3.1.1 *Sendmail* doesn’t always provide adequate concurrency limits.

7.4. Extended SMTP Support

Version 8 includes both sending and receiving support for Extended SMTP support as defined by RFC 1651 (basic) and RFC 1653 (SIZE); and limited support for RFC 1652 (BODY).

7.5. Eight-Bit Clean

Previous versions of *sendmail* used the 0200 bit for quoting. This version avoids that use. However, for compatibility with RFC 822, you can set option ‘7’ to get seven bit stripping.

Individual mailers can still produce seven bit output using the ‘7’ mailer flag.

7.6. User Database

The user database is an as-yet experimental attempt to provide unified large-site name support. We are installing it at Berkeley; future versions may show significant modifications.

7.7. Improved BIND Support

The BIND support, particularly for MX records, had a number of annoying ‘‘features’’ which have been removed in this release. In particular, these more tightly bind (pun intended) the name server to *sendmail*, so that the name server resolution rules are incorporated directly into **sendmail**.

7.8. Keyed Files

Generalized keyed files is an idea taken directly from IDA *sendmail* (albeit with a completely different implementation). They can be useful on large sites.

Version 8 also understands YP.

7.9. Multi-Word Classes

Classes can now be multiple words. For example,

CShofmann.CS.Berkeley.EDU

allows you to match the entire string ‘‘hofmann.CS.Berkeley.EDU’’ using the single construct ‘‘$=S’’.

7.10. Deferred Macro Expansion

The **$&***x* construct has been adopted from IDA.

7.11. IDENT Protocol Support

The IDENT protocol as defined in RFC 1413 is supported.

7.12. Parsing Bug Fixes

A number of small bugs having to do with things like backslash-escaped quotes inside of comments have been fixed.

7.13. Separate Envelope/Header Processing

Since the From: line is passed in separately from the envelope sender, these have both been made visible; the **$g** macro is set to the envelope sender during processing of mailer argument vectors and the header sender during processing of headers.

It is also possible to specify separate per-mailer envelope and header processing. The **S**enderRWSet and **R**ecipientRWset arguments for mailers can be specified as *envelope/header* to give different rewritings for envelope versus header addresses.

7.14. Owner-List Propagates to Envelope

When an alias has an associated owner−list name, that alias is used to change the envelope sender address. This will cause downstream errors to be returned to that owner.

7.15. Dynamic Header Allocation

The fixed size limit on header lines has been eliminated.

7.16. New Command Line Flags

The **−B** flag has been added to pass in body type information.

The **−p** flag has been added to pass in protocol information.

The **−X** flag has been added to allow logging of all protocol in and out of *sendmail* for debugging.

The **−O** flag simplies setting long-form options.

7.17. Enhanced Command Line Flags

The **−q** flag can limit limit a queue run to specific recipients, senders, or queue ids using −**qR***substring*, −**qS***substring*, **or** −**qI***substring* **respectively.**

7.18. New and Old Configuration Line Types

The **K** line has been added to declare database maps.

The **V** line has been added to declare the configuration version level.

The **M** line has a ‘‘D=’’ field that lets you change into a temporary directory while that mailer is running. It also has a ‘‘U=’’ field to allow you to set the user and group id to be used when running the mailer.

7.19. New Options

Several new options have been added, many to support new features, others to allow tuning that was previously available only by recompiling. They are described in detail in Section 5.1.5. Briefly,

b Insist on a minimum number of disk blocks.

C Set checkpoint interval.

E Default error message.

G Enable GECOS matching.

h Maximum hop count.

j Send errors in MIME-encapsulated format.

J Forward file path.

k Connection cache size

K Connection cache lifetime.

l Enable Errors-To: header. These headers violate RFC 1123; this option is included to provide back compatibility with old versions of *sendmail*.

O Set incoming SMTP daemon options, such as an alternate SMTP port.

p Privacy options.

R Don’t prune route-addrs.

U User database spec.

V Fallback ‘‘MX’’ host.

w ‘‘Best MX’’ handling technique.

7 Do not run eight bit clean.

8 Eight bit data handling mode.

7.20. Extended Options

The **r** (read timeout), **I** (use BIND), and **T** (queue timeout) options have been extended to pass in more information.

7.21. New Mailer Flags

Several new mailer flags have been added.

a Try to use ESMTP when creating a connection. If this is not set, *sendmail* will still try if the other end hints that it knows about ESMTP in its greeting message; this flag says to try even if it doesn’t hint. If the EHLO (extended hello) command fails, *sendmail* falls back to old SMTP.

A Try the user part of addresses for this mailer as aliases.

b Ensure that there is a blank line at the end of all messages.

c Strip all comments from addresses; this should only be used as a last resort when dealing with cranky mailers.

g Never use the null sender as the envelope sender, even when running SMTP. Although this violates RFC 1123, it may be necessary when you must deal with some obnoxious old hosts.

k Turn off the loopback check in the HELO protocol; doing this may cause mailer loops.

o Always run the mailer as the recipient of the message.

w This user should have a passwd file entry.

5 Try ruleset 5 if no local aliases.

7 Strip all output to 7 bits.

: Check for :include: files.

| Check for |program addresses.

/ Check for /file addresses.

@ Check this user against the user database.

7.22. Long Option Names

All options can be specified using long names, and some new options can only be specified with long names.

7.23. New Pre-Defined Macros

The following macros are pre-defined:

$k The UUCP node name, nominally from *uname*(2) call.

$m The domain part of our full hostname.

$_ The RFC 1413-provided sender address.

7.24. New LHS Token

Version 8 allows **$@** on the Left Hand Side of an ‘‘R’’ line to match zero tokens. This is intended to be used to match the null input.

7.25. Bigger Defaults

Version 8 allows up to 100 rulesets instead of 30. It is recommended that rulesets 0–9 be reserved for *sendmail*’s dedicated use in future releases.

The total number of MX records that can be used has been raised to 20.

The number of queued messages that can be handled at one time has been raised from 600 to 1000.

7.26. Different Default Tuning Parameters

Version 8 has changed the default parameters for tuning queue costs to make the number of recipients more important than the size of the message (for small messages). This is reasonable if you are connected with reasonably fast links.

7.27. Auto-Quoting in Addresses

Previously, the ‘‘Full Name <email address>’’ syntax would generate incorrect protocol output if ‘‘Full Name’’ had special characters such as dot. This version puts quotes around such names.

7.28. Symbolic Names On Error Mailer

Several names have been built in to the $@ portion of the $#error mailer.

7.29. SMTP VRFY Doesn't Expand

Previous versions of *sendmail* treated VRFY and EXPN the same. In this version, VRFY doesn't expand aliases or follow .forward files. EXPN still does.

As an optimization, if you run with your default delivery mode being queue-only or deliver-in-background, the RCPT command will also not chase aliases and .forward files. It will chase them when it processes the queue.

7.30. [IPC] Mailers Allow Multiple Hosts

When an address resolves to a mailer that has ‘‘[IPC]’’ as its ‘‘Path’’, the $@ part (host name) can be a colon-separated list of hosts instead of a single hostname. This asks *sendmail* to search the list for the first entry that is available exactly as though it were an MX record. The intent is to route internal traffic through internal networks without publishing an MX record to the net. MX expansion is still done on the individual items.

7.31. Aliases Extended

The implementation has been merged with maps. Among other things, this supports NIS-based aliases.

7.32. Portability and Security Enhancements

A number of internal changes have been made to enhance portability.

Several fixes have been made to increase the paranoia factor.

7.33. Miscellaneous Changes

Sendmail writes a */etc/sendmail.pid* file with the current process id of the SMTP daemon.

Two people using the same program in their .forward file are considered different so that duplicate elimination doesn't delete one of them.

The *mailstats* program prints mailer names and gets the location of the *sendmail.st* file from */etc/sendmail.cf*.

Many minor bugs have been fixed, such as handling of backslashes inside of quotes.

A hook (ruleset 5) has been added to allow rewriting of local addresses after aliasing.

8. ACKNOWLEDGEMENTS

I've worked on *sendmail* for many years, and many employers have been remarkably patient about letting me work on a large project that was not part of my official job. This includes time on the INGRES Project at Berkeley, at Britton Lee, and again on the Mammoth Project at Berkeley.

Much of the second wave of improvements should be credited to Bryan Costales of ICSI. As he passed me drafts of his book on *sendmail* I was inspired to start working on things again. Bryan was also available to bounce ideas off of.

Many, many people contributed chunks of code and ideas to *sendmail*. It has proven to be a group network effort. Version 8 in particular was a group project. The following people made notable contributions:

John Beck, Hewlett-Packard
Keith Bostic, CSRG, University of California, Berkeley
Andrew Cheng, Sun Microsystems
Michael J. Corrigan, University of California, San Diego
Bryan Costales, International Computer Science Institute
Pär (Pell) Emanuelsson
Craig Everhart, Transarc Corporation
Tom Ivar Helbekkmo, Norwegian School of Economics
Allan E. Johannesen, WPI
Jonathan Kamens, OpenVision Technologies, Inc.

Takahiro Kanbe, Fuji Xerox Information Systems Co., Ltd.
Brian Kantor, University of California, San Diego
Murray S. Kucherawy, HookUp Communication Corp.
Bruce Lilly, Sony U.S.
Karl London
Motonori Nakamura, Ritsumeikan University & Kyoto University
John Gardiner Myers, Carnegie Mellon University
Neil Rickert, Northern Illinois University
Eric Schnoebelen, Convex Computer Corp.
Eric Wassenaar, National Institute for Nuclear and High Energy Physics, Amsterdam
Christophe Wolfhugel, Pasteur Institute & Herve Schauer Consultants (Paris)

I apologize for anyone I have omitted, misspelled, misattributed, or otherwise missed. At this point, I suspect that at least a hundred people have contributed code, and many more have contributed ideas, comments, and encouragement. I've tried to list them in the RELEASE_NOTES in the distribution directory. I appreciate their contribution as well.

Special thanks are reserved for Michael Corrigan and Christophe Wolfhugel, who besides being wonderful guinea pigs and contributors have also consented to be added to the ‘‘sendmail@CS.Berkeley.EDU’’ list and, by answering the bulk of the questions sent to that list, have freed me up to do other work.

APPENDIX A

COMMAND LINE FLAGS

Arguments must be presented with flags before addresses. The flags are:

−b*x*	Set operation mode to *x*. Operation modes are: m Deliver mail (default) s Speak SMTP on input side a† ‘‘Arpanet’’ mode (get envelope sender information from header) d Run as a daemon t Run in test mode v Just verify addresses, don’t collect or deliver i Initialize the alias database p Print the mail queue
−B*type*	Indicate body type.
−C*file*	Use a different configuration file. *Sendmail* runs as the invoking user (rather than root) when this flag is specified.
−d*level*	Set debugging level.
−f *addr*	The sender’s machine address is *addr*.
−F*name*	Sets the full name of this user to *name*.
−h *cnt*	Sets the ‘‘hop count’’ to *cnt*. This represents the number of times this message has been processed by *sendmail* (to the extent that it is supported by the underlying networks). *Cnt* is incremented during processing, and if it reaches MAXHOP (currently 30) *sendmail* throws away the message with an error.
−n	Don’t do aliasing or forwarding.
−r *addr*	An obsolete form of **−f**.
−o*x value*	Set option *x* to the specified *value*. These options are described in Appendix B.
−O*option*=*value*	Set *option* to the specified *value* (for long form option names).
−M*x value*	*Set macro x to the specified value.*
−p*protocol*	Set the sending protocol. Programs are encouraged to set this. The protocol field can be in the form *protocol***:***host* to set both the sending protocol and sending host. For example, ‘‘−pUUCP:uunet’’ sets the sending protocol to UUCP and the sending host to uunet. (Some existing programs use −oM to set the r and s macros; this is equivalent to using −p.)
−q*time*	Try to process the queued up mail. If the time is given, a *sendmail* will run through the queue at the specified interval to deliver queued mail; otherwise, it only runs once.
−q*Xstring*	Run the queue once, limiting the jobs to those matching *Xstring*. The key letter *X* can be **I** to limit based on queue identifier, **R** to limit based on recipient, or **S** to limit based on sender. A particular queued job is accepted if one of the corresponding addresses contains the indicated *string*.
−t	Read the header for ‘‘To:’’, ‘‘Cc:’’, and ‘‘Bcc:’’ lines, and send to everyone listed in those lists. The ‘‘Bcc:’’ line will be deleted before sending. Any addresses in the

†Deprecated.

argument vector will be deleted from the send list.

−X *logfile* Log all traffic in and out of *sendmail* in the indicated *logfile* for debugging mailer problems. This produces a lot of data very quickly and should be used sparingly.

There are a number of options that may be specified as primitive flags. These are the e, i, m, and v options. Also, the f option may be specified as the −**s** flag.

APPENDIX B

QUEUE FILE FORMATS

This appendix describes the format of the queue files. These files live in the directory defined by the **Q** option in the *sendmail.cf* file, usually */var/spool/mqueue* or */usr/spool/mqueue*.

All queue files have the name *x* **f***AAA99999* where *AAA99999* is the *id* for this message and the *x* is a type. The first letter of the id encodes the hour of the day that the message was received by the system (with A being the hour between midnight and 1:00AM). All files with the same id collectively define one message.

The types are:

d The data file. The message body (excluding the header) is kept in this file.

q The queue control file. This file contains the information necessary to process the job.

t A temporary file. These are an image of the **qf** file when it is being rebuilt. It should be renamed to a **qf** file very quickly.

x A transcript file, existing during the life of a session showing everything that happens during that session.

The **qf** file is structured as a series of lines each beginning with a code letter. The lines are as follows:

V The version number of the queue file format, used to allow new *sendmail* binaries to read queue files created by older versions. Defaults to version zero. Must be the first line of the file if present.

H A header definition. There may be any number of these lines. The order is important: they represent the order in the final message. These use the same syntax as header definitions in the configuration file.

C The controlling address. The syntax is ‘‘localuser:aliasname’’. Recipient addresses following this line will be flagged so that deliveries will be run as the *localuser* (a user name from the /etc/passwd file); *aliasname* is the name of the alias that expanded to this address (used for printing messages).

Q The ‘‘original recipient’’, specified by the ORCPT= field in an ESMTP transaction. Used exclusively for Delivery Status Notifications. It applies only to the immediately following ‘R’ line.

R A recipient address. This will normally be completely aliased, but is actually realiased when the job is processed. There will be one line for each recipient. Version 1 qf files also include a leading colon-terminated list of flags, which can be ‘S’ to return a message on successful final delivery, ‘F’ to return a message on failure, ‘D’ to return a message if the message is delayed, ‘B’ to indicate that the body should be returned, ‘N’ to suppress returning the body, and ‘P’ to declare this as a ‘‘primary’’ (command line or SMTP-session) address.

S The sender address. There may only be one of these lines.

T The job creation time. This is used to compute when to time out the job.

P The current message priority. This is used to order the queue. Higher numbers mean lower priorities. The priority changes as the message sits in the queue. The initial priority depends on the message class and the size of the message.

M A message. This line is printed by the *mailq* command, and is generally used to store status information. It can contain any text.

F Flag bits, represented as one letter per flag. Defined flag bits are **r** indicating that this is a response message and **w** indicating that a warning message has been sent announcing that the mail has been delayed.

N The total number of delivery attempts.

K The time (as seconds since January 1, 1970) of the last delivery attempt.

I The i-number of the data file; this can be used to recover your mail queue after a disastrous disk crash.

$ A macro definition. The values of certain macros (as of this writing, only **$r** and **$s**) are passed through to the queue run phase.

B The body type. The remainder of the line is a text string defining the body type. If this field is missing, the body type is assumed to be ‘‘undefined’’ and no special processing is attempted. Legal values are ‘‘7BIT’’ and ‘‘8BITMIME’’.

O The original MTS value (from the ESMTP transaction). For Deliver Status Notifications only.

Z The original envelope id (from the ESMTP transaction). For Deliver Status Notifications only.

As an example, the following is a queue file sent to ‘‘eric@mammoth.Berkeley.EDU’’ and ‘‘bostic@okeeffe.CS.Berkeley.EDU’’[1]:

```
P835771
T404261372
Seric
Ceric:sendmail@vangogh.CS.Berkeley.EDU
Reric@mammoth.Berkeley.EDU
Rbostic@okeeffe.CS.Berkeley.EDU
H?P?return-path: <owner-sendmail@vangogh.CS.Berkeley.EDU>
Hreceived: by vangogh.CS.Berkeley.EDU (5.108/2.7) id AAA06703;
        Fri, 17 Jul 92 00:28:55 -0700
Hreceived: from mail.CS.Berkeley.EDU by vangogh.CS.Berkeley.EDU (5.108/2.7)
        id AAA06698; Fri, 17 Jul 92 00:28:54 -0700
Hreceived: from [128.32.31.21] by mail.CS.Berkeley.EDU (5.96/2.5)
        id AA22777; Fri, 17 Jul 92 03:29:14 -0400
Hreceived: by foo.bar.baz.de (5.57/Ultrix3.0-C)
        id AA22757; Fri, 17 Jul 92 09:31:25 GMT
H?F?from: eric@foo.bar.baz.de (Eric Allman)
H?x?full-name: Eric Allman
Hmessage-id: <9207170931.AA22757@foo.bar.baz.de>
HTo: sendmail@vangogh.CS.Berkeley.EDU
Hsubject: this is an example message
```

This shows the person who sent the message, the submission time (in seconds since January 1, 1970), the message priority, the message class, the recipients, and the headers for the message.

[1]This example is contrived and probably inaccurate for your environment. Glance over it to get an idea; nothing can replace looking at what your own system generates.

APPENDIX C

SUMMARY OF SUPPORT FILES

This is a summary of the support files that *sendmail* creates or generates. Many of these can be changed by editing the sendmail.cf file; check there to find the actual pathnames.

/usr/sbin/sendmail
: The binary of *sendmail*.

/usr/bin/newaliases
: A link to /usr/sbin/sendmail; causes the alias database to be rebuilt. Running this program is completely equivalent to giving *sendmail* the −**bi** flag.

/usr/bin/mailq
: Prints a listing of the mail queue. This program is equivalent to using the −**bp** flag to *sendmail*.

/etc/sendmail.cf
: The configuration file, in textual form.

/usr/lib/sendmail.hf
: The SMTP help file.

/etc/sendmail.st
: A statistics file; need not be present.

/etc/sendmail.pid
: Created in daemon mode; it contains the process id of the current SMTP daemon. If you use this in scripts; use ‘‘head −1’’ to get just the first line; later versions of *sendmail* may add information to subsequent lines.

/etc/aliases
: The textual version of the alias file.

/etc/aliases.{pag,dir}
: The alias file in *dbm* (3) format.

/var/spool/mqueue
: The directory in which the mail queue and temporary files reside.

/var/spool/mqueue/qf*
: Control (queue) files for messages.

/var/spool/mqueue/df*
: Data files.

/var/spool/mqueue/tf*
: Temporary versions of the qf files, used during queue file rebuild.

/var/spool/mqueue/xf*
: A transcript of the current session.

This page intentionally left blank;
replace it with a blank sheet for double-sided output.

TABLE OF CONTENTS

1. BASIC INSTALLATION 7
  1.1. Compiling Sendmail 7
    1.1.1. Tweaking the Makefile 7
    1.1.2. Compilation and installation 7
  1.2. Configuration Files 8
  1.3. Details of Installation Files 9
    1.3.1. /usr/sbin/sendmail 9
    1.3.2. /etc/sendmail.cf 9
    1.3.3. /usr/bin/newaliases 10
    1.3.4. /var/spool/mqueue 10
    1.3.5. /etc/aliases* 10
    1.3.6. /etc/rc 10
    1.3.7. /usr/lib/sendmail.hf 11
    1.3.8. /etc/sendmail.st 12
    1.3.9. /usr/bin/mailq 12
2. NORMAL OPERATIONS 12
  2.1. The System Log 12
    2.1.1. Format 12
    2.1.2. Levels 13
  2.2. Dumping State 13
  2.3. The Mail Queue 13
    2.3.1. Printing the queue 13
    2.3.2. Forcing the queue 13
  2.4. The Service Switch 14
  2.5. The Alias Database 14
    2.5.1. Rebuilding the alias database 15
    2.5.2. Potential problems 15
    2.5.3. List owners 16
  2.6. User Information Database 16
  2.7. Per-User Forwarding (.forward Files) 16
  2.8. Special Header Lines 16
    2.8.1. Return-Receipt-To: 16
    2.8.2. Errors-To: 17
    2.8.3. Apparently-To: 17
    2.8.4. Precedence 17
  2.9. IDENT Protocol Support 17
3. ARGUMENTS 18
  3.1. Queue Interval 18
  3.2. Daemon Mode 18
  3.3. Forcing the Queue 18
  3.4. Debugging 18
  3.5. Changing the Values of Options 19
  3.6. Trying a Different Configuration File 19
  3.7. Logging Traffic 19
  3.8. Testing Configuration Files 20

4. TUNING 20
4.1. Timeouts 21
4.1.1. Queue interval 21
4.1.2. Read timeouts 21
4.1.3. Message timeouts 22
4.2. Forking During Queue Runs 22
4.3. Queue Priorities 22
4.4. Load Limiting 23
4.5. Delivery Mode 23
4.6. Log Level 23
4.7. File Modes 24
4.7.1. To suid or not to suid? 24
4.7.2. Should my alias database be writable? 24
4.8. Connection Caching 24
4.9. Name Server Access 25
4.10. Moving the Per-User Forward Files 26
4.11. Free Space 26
4.12. Maximum Message Size 26
4.13. Privacy Flags 26
4.14. Send to Me Too 26
5. THE WHOLE SCOOP ON THE CONFIGURATION FILE 26
5.1. R and S — Rewriting Rules 27
5.1.1. The left hand side 27
5.1.2. The right hand side 27
5.1.3. Semantics of rewriting rule sets 29
5.1.4. IPC mailers 29
5.2. D — Define Macro 30
5.3. C and F — Define Classes 32
5.4. M — Define Mailer 33
5.5. H — Define Header 36
5.6. O — Set Option 36
5.7. P — Precedence Definitions 43
5.8. V — Configuration Version Level 44
5.9. K — Key File Declaration 45
5.10. The User Database 47
5.10.1. Structure of the user database 48
5.10.2. User database semantics 48
5.10.3. Creating the database[22] 49
6. OTHER CONFIGURATION 49
6.1. Parameters in src/Makefile 49
6.2. Parameters in src/conf.h 49
6.3. Configuration in src/conf.c 52
6.3.1. Built-in Header Semantics 52
6.3.2. Restricting Use of Email 53
6.3.3. Load Average Computation 54
6.3.4. New Database Map Classes 54
6.3.5. Queueing Function 54

6.3.6. Refusing Incoming SMTP Connections 55
6.3.7. Load Average Computation 55
6.4. Configuration in src/daemon.c 55
7. CHANGES IN VERSION 8 56
7.1. Connection Caching 56
7.2. MX Piggybacking 56
7.3. RFC 1123 Compliance 56
7.4. Extended SMTP Support 56
7.5. Eight-Bit Clean 56
7.6. User Database 56
7.7. Improved BIND Support 56
7.8. Keyed Files 57
7.9. Multi-Word Classes 57
7.10. Deferred Macro Expansion 57
7.11. IDENT Protocol Support 57
7.12. Parsing Bug Fixes 57
7.13. Separate Envelope/Header Processing 57
7.14. Owner-List Propagates to Envelope 57
7.15. Dynamic Header Allocation 57
7.16. New Command Line Flags 57
7.17. Enhanced Command Line Flags 57
7.18. New and Old Configuration Line Types 58
7.19. New Options 58
7.20. Extended Options 58
7.21. New Mailer Flags 58
7.22. Long Option Names 59
7.23. New Pre-Defined Macros 59
7.24. New LHS Token 59
7.25. Bigger Defaults 59
7.26. Different Default Tuning Parameters 59
7.27. Auto-Quoting in Addresses 59
7.28. Symbolic Names On Error Mailer 59
7.29. SMTP VRFY Doesn't Expand 60
7.30. [IPC] Mailers Allow Multiple Hosts 60
7.31. Aliases Extended 60
7.32. Portability and Security Enhancements 60
7.33. Miscellaneous Changes 60
8. ACKNOWLEDGEMENTS 60
Appendix A. COMMAND LINE FLAGS 62
Appendix B. QUEUE FILE FORMATS 64
Appendix C. SUMMARY OF SUPPORT FILES 66