In this chapter:

- *Software Prerequisites*
- *Hardware Note*
- *Where to Get the Software*
- *Supported Platforms*
- *Installing Cyrus*
- *Upgrading from Previous Versions of Cyrus IMAP*
- *Components of Cyrus and What They Do*
- *Common Problems*
- *Significant Bugs*

7

Installing the Cyrus IMAP Server

Smooth is the name of the game when it comes to installing the Cyrus server. Aside from the fact that there are several dependencies on other packages, code hacking and Makefile mangling should not be necessary. Cyrus is a very good example of platform-independent open source software that makes no broad assumptions about the technical or psychic ability of the person installing it.

Software Prerequisites

Cyrus *imapd* requires a modest complement of additional freely available software. You need only Tool Command Language (Tcl) Version 7.5 or later and sendmail* 8.7.1 or later. If you're installing Cyrus Version l.6 or later, Cyrus SASL is also required.

If your site has Kerberos, you can compile Cyrus with Kerberos support to add additional security to your IMAP infrastructure. Although your install might go marginally smoother or faster if you have *makedepend* installed, you can also use the dummy shell script included with the package as a workaround for not having *makedepend*.

* Although sendmail enjoys wild popularity as an MTA, you could use any reasonable MTA that permits you to choose Cyrus *deliver* as your delivery agent.

Hardware Note

As with any other service, the hardware demands of Cyrus IMAP can be grouped into demands that are general to any IMAP server and those that are specific to Cyrus IMAP. A good rule is not to just throw hardware, and therefore time and money, at your IMAP server. Find out what kind of appetites your server will have *before* you feed it.

IMAP servers have an appetite for disk I/O, memory, and network bandwidth, in that order. More specifically, Cyrus IMAP has a ravenous appetite for disk I/O bandwidth. The bigger highway you have between your CPU, memory, and disk, the better. To that end, a good Cyrus IMAP machine is one in which user mailboxes are equally split by load between two or more storage channels. The ideal Cyrus IMAP server has two network interfaces and siphons off all non-IMAP overhead traffic (such as networked tape backups) to one interface, leaving the other free to handle IMAP traffic. It also has more than enough memory to handle peak loads.

Chapter 16, *Server Performance Tuning*, covers the details of optimizing an IMAP system, but let's talk briefly about each of these general requirements now. As with any Internet service, memory is of paramount importance. The less paging your system has to do, the better. A good rule of thumb is that your machine should have a bare minimum of 1 MB* of physical memory for each active IMAP process. If you assume one process per user and an overhead of about 20 to 30 housekeeping processes on a black box single-purpose IMAP server, then gauging your memory needs becomes a matter of determining how many users will be online at one time. A good starting point might be to figure that 20 to 30% of ISP customers might be connected at any given time, whereas far more, maybe 80%, of business users will be running IMAP processes simultaneously.

Spreading the load across I/O channels is a straightforward proposition. Typically, mail servers begin their life with all users on a single disk partition that, presumably, is serviced by a single controller. Spreading users' mailboxes out to different partitions served by different controllers permits your disk storage to service more users at once. The Cyrus server's built-in support for a partitioned mailstore simplifies partitioning.

The only system requirement unique to the Cyrus IMAP server is the large *mailboxes* file. One way to accommodate this file would be to hang a solid-state disk

* A Cyrus *imapd* process consumes roughly 400–800 KB of private memory and 1,400–1,600 KB of shared memory. The shared memory usage is counted towards system overhead, and the private memory usage is used to estimate the size of the *imapd* process in terms of capacity planning.

off your mail server and use it exclusively for the *mailboxes* file. Because solid-state disks tend to be more expensive than conventional storage, it might be more prudent to dedicate a partition of RAID storage or just a nothing-fancy disk to the *mailboxes* file. Barring that, anything you can do to isolate the load of constant multiprocess access to the *mailboxes* file from the rest of the system would help. Maybe you can't partition your users' mailboxes into separate storage channels or split your traffic across several network interfaces. You could still put your *mailboxes* file on a dedicated disk with its own I/O channel to increase the load your server can handle.

Where to Get the Software

The canonical source of information on where to obtain the Cyrus IMAP server is the Cyrus IMAP Server Home Page at *http://asg.web.cmu.edu/cyrus/download/*. Toward the top of the first page of that site is a link to download the current distribution. You can also go to *ftp://ftp.andrew.cmu.edu/pub/cyrus-mail/* and select the latest version, or an earlier stable release, such as 1.5.19, if you so desire.

As we mentioned earlier, you will need a few other pieces of software, namely sendmail and Tcl, and for Versions 1.6 and higher,* you will need Cyrus SASL (Simple Authentication and Security Layer). Cyrus IMAP has hooks for Kerberos and Sieve, a server-side filtering language, but both are optional.

The current version of sendmail is always available from the Sendmail Current Release web page located at *http://www.sendmail.org/current-release.html*. Do yourself and your users a favor and don't even think about using the version of sendmail that comes with your OS.† Not only is it practically guaranteed not to include desirable new features, but it's an almost sure bet that there are undocumented security problems that are likely to bait any less-than-ethical folks whose talent outstrips their self-restraint.

The canonical source for Tcl is the web site for Scriptics, the company formed by John Ousterhout, the original author of Tcl: *http://www.scriptics.com/*.

---

* As we mentioned in Chapter 6, *Introduction to the Cyrus IMAP Server*, Versions 1.6 and higher require some extra work to use cleartext or shadow password authentication, forcing some sites to stick with 1.5.19 until they can replace cleartext authentication with something else.

† Never, ever, ever use sendmail 8.6. Here's a quote from *ftp://ftp.sendmail.org/pub/sendmail/.message*: "There is NO 8.6.* patch for CA-96.20. 8.6 is not supported, not secure, and should not be run on any network-connected machine." CA-96.20 refers to a CERT Advisory (*http://www.cert.org/advisories/CA-96.20.sendmail_vul.html*) that documents resource starvation and buffer overflow attacks. By now, such attacks are well known and could be used to give arbitrary levels of access to your mail server to unauthorized people with a bit of easily obtainable code.

SASL is required only for Versions 1.6 and later of Cyrus IMAP. If you've opted to stick with Version 1.5.19, then you won't need SASL. Cyrus SASL is available from CMU's FTP site: *ftp://ftp.andrew.cmu.edu/pub/cyrus-mail/cyrus-sasl-1.5.11.tar.gz* (be sure and check for a later version—SASL is being updated every month or so).

If you need to build Kerberos, and United States export laws permit you to do so, you can get the current revision of Kerberos by using the MIT Kerberos Distribution Authorization Form at *http://web.mit.edu/network/kerberos-form.html*.

Supported Platforms

As an open source product, Cyrus runs on any platform you can build it on. You can bet that Cyrus is more easily built on systems that are somewhat commonly used in the industry. Forget about using that ancient PDP-11 you found in the basement.

Because of its resource demands, choose a machine that can:

- Support the amount of memory your Cyrus user base requires
- Reliably support the fastest network media available on your network
- Support robust disk storage and fast I/O

Installing Cyrus

Installing the Cyrus server is literally as easy as 1-2-3. Configuring the server is quite a bit more complicated, but we'll address that in Chapter 8, *Configuring the Cyrus Server*. First, download and unpack the distribution. Second, decide what build-time configuration options you'll use. Finally, compile the software and do your final install.

Versions 1.6 and Up and SASL

As mentioned earlier, the Cyrus SASL library is a prerequisite for Versions 1.6 and up of Cyrus IMAP. Make sure SASL is installed and configured before compiling Cyrus IMAP Version 1.6 or higher. SASL requires some configuration of its own. CMU recommends configuring SASL to use CRAM-MD5 on a single server, or GSSAPI for use on more than one server. You can also configure SASL to use the *pwcheck* daemon for cleartext Unix password authentication.

Unpack the Distribution

Assuming you've already downloaded Cyrus using a command similar in effect to:

```
% lynx -dump ftp://ftp.andrew.cmu.edu/pub/cyrus-mail/cyrus-imapd-v1.5.19.tar.gz \
  > cyrus-imapd-v1.5.19.tar.gz
```

uncompress it, untar it, and you've got your source tree for Cyrus:

```
% ls -Fs
total 1048                        1048 cyrus-imapd-v1.5.19.tar.gz
% gunzip cyrus-imapd-v1.5.19.tar.gz
% ls -s
total 4688                     4688 cyrus-imapd-v1.5.19.tar
% tar xf cyrus-imapd-v1.5.19.tar
% ls -FaCs
total 4694                        2 cyrus-imapd-v1.5.19
   2 ./                        4688 cyrus-imapd-v1.5.19.tar
   2 ../
```

Setting Up Build-Time Configuration Options

Your first task is to decide on what options to use with the *configure* script. *configure*'s options are shown in Table 7-1. If you have a somewhat commonly configured host that uses shadow passwords, has Tcl installed, and doesn't use Kerberos, you might be able to get away with something like:

```
% ./configure --with-login=unix_pwcheck --without-krb
```

That instructs *configure* to prepare an appropriate *Makefile* for a host that isn't Kerberized, uses shadow passwords, and has Tcl somewhere under */usr/local* or in the compiler's default link path.

Table 7-1. configure Options

Option	Purpose
`--help`	Dump a list of configure options.
`--quiet`, `--silent`, `--q`	Omit status messages listing what checks are being made.
`--version`	Show what version of Autoconf is being used.
`--prefix=`*`path`*	Install architecture-dependent files in *path* (*/usr/local* is default).
`--with-cyrus-prefix=`*`path`*	Set the Cyrus install directory to *path*. Try to leave this alone. It defaults to */usr/cyrus* and Cyrus troubleshooting becomes much more difficult if non-default values are used in the install without a very compelling reason.
`--cache-file=`*`file`*	Set an alternate configure cache file *file*. Defaults to *config.cache*. Not necessary under most circumstances.

Table 7-1. configure Options (continued)

Option	Purpose
`--srcdir=`*`dir`*	Set the source directory to *dir*. Almost never necessary. Don't use unless there's good reason to suspect that Autoconf can't find your sources or you're working from multiple source trees with one copy of configure and Autoconf.
`--with-cyrus-user=`*`user`*	Set an alternate Cyrus *user*. The default is the user *cyrus*. Using other than *cyrus* is not a good idea, especially if sendmail is your MTA. The Cyrus *m4* prototype used in creating the *sendmail.cf* doesn't react well to *userid:group* combinations other than *cyrus:mail*. As with the Cyrus prefix, don't change this unless you've got a compelling reason to do so.
`--with-cyrus-group=`*`group`*	Set an alternate cyrus *group*. The default is the group *mail*. See `--with-cyrus-user`.
`--with-statedir=`*`path`*	Set an alternate server runtime state directory to *path*. Defaults to */var/cyrus*.
`--with-login=`*`method`*	How should Cyrus authenticate users? For Version 1.5.x, use *unix* for cleartext local passwd authentication, *unix_pwcheck* for shadow passwd authentication, and *krb* or *krb_pwcheck* for Kerberos. Versions 1.6.x have the option of *unix* (local passwd) or *krb* (Kerberos).
`--with-pwcheck=`*`method`*	(Version 1.5.x only) What method should the *pwcheck* daemon use to authenticate? This option is only used if the `--with-login` option value ended in `_pwcheck`. *pwcheck* figures it out on its own, so this option is probably unnecessary for you. Default is *getpsnam* if *getpsnam* exists; otherwise, it is *getpwnam*.
`--with-auth=`*`method`*	Authorization method used by the login method. Defaults to the authentication method with the same name as the login method. Usually not necessary to specify anything other than the default unless you're using Kerberos.
`--with-inn=path`	Path to INN for Usenet news integration.
`--with-krb=`*`path`* `--without-krb`	Path to Kerberos V4 libraries. This support requires the DES library and is subject to US export laws. If you don't have Kerberos, use the `--without-krb` switch.
`--with-lock=`*`lock-method`*	Force the use of *lock-method*. *lock-method* is either `fcntl` or `flock`.
`--with-tcl=`*`path`* `--without-tcl`	Path to the Tcl library and header files. Looks in */usr/local/* and your compiler's default link path by default. Usually not necessary to set this option.
`--disable-cyradm`	If you don't have Tcl, don't have time to build it, or simply don't like it, you can use this option to disable the Cyrus administrative client and use alternative methods like IMAP scripting to accomplish the same goals.

Table 7-1. configure Options (continued)

Option	Purpose
`--with-notify=`*`method`* `--without-notify`	This looks like an encouraging hook to pull in an arbitrary notification method, but currently really only specifies if Zephyr notification is to be used for new mail notification or not. Defaults to *zephyr* if the *zephyr* libraries are found, or to *no* (no new mail notification) if they're not found. *configure* can determine the default from the libraries in the search path. The vast majority of sites will want to accept the default.
`--with-zephyr=`*`path`* `--without-zephyr`	Zephyr is an instant messaging protocol (*ftp://athena-dist.mit.edu/pub/ATHENA/zephyr/*). Because most sites will configure Cyrus `--without-notify`, those sites will set this option to `--without-zephyr`.
`--disable-privacy`	If you are using Kerberos and want just authentication but not content to be encrypted, use this switch.
`--disable-sieve`	(Version 1.6.x only) Disable support for the Sieve server-side filtering language.[a]

[a] See Chapter 15, *Server-Side Mail Filtering*, for more information on Sieve.

Depending on the version of Cyrus you're building, the *configure* script may accept options in addition to those listed in Table 7-1 (use the command *configure –help* for a complete listing). Your system may require unusual options that are not defined in the *configure* script. In that case, you can set the options in your environment before running *make all*. Options set in the environment override options set by *configure*. The options most commonly set in the environment are listed in Table 7-2.

Table 7-2. Options Commonly Set in the Environment

Option	Defines
CC	C compiler program. Defaults to *gcc* if *gcc* is found in the PATH; otherwise, defaults to *cc*.
CFLAGS	C compiler options.
CPPFLAGS	Extra directories to search for header files. The default is none.
DEFS	Miscellaneous options, such as platform options (–DSVR4).
LIBS	Libraries to link with (e.g., –lposix).
LDFLAGS	Linker options. Defaults to none.

To set an option in the environment, run *configure* as shown in the following examples:

```
$ CC=gcc CFLAGS=-O2 LIBS=-lsocket ./configure  (Bourne shell variants)
% env CPPFLAGS=-I/usr/local/lib ./configure    (C shell variants)
```

You can also override the make variables CFLAGS and LDFLAGS after *configure* has been run when you run *make all*; for example:

```
% make all CFLAGS=-O LDFLAGS=-s
```

Now that you've selected a (hopefully modest) number of switches to use with *configure*, your *configure* run should look something like this:

```
% ./configure --with-login=unix_pwcheck --without-krb
loading cache ./config.cache
checking for makedepend... makedepend
checking for gcc... gcc
checking whether the C compiler (gcc  ) works... yes
checking whether the C compiler (gcc  ) is a cross-compiler... no
checking whether we are using GNU C... yes
checking whether gcc accepts -g... yes
(Many lines skipped...)
updating cache ./config.cache
creating ./config.status
creating Makefile
creating man/Makefile
creating lib/Makefile
creating imap/Makefile
creating imap/feedcyrus
creating imtest/Makefile
creating pwcheck/Makefile
creating cyradm/Makefile
creating et/Makefile
%
```

Compile and Install the Software

Configure has now built your *Makefile*. We're going to assume that you don't have *makedepend*, because the included dummy *makedepend* script works just fine—it makes things so easy that you needn't worry about installing *makedepend*. Should you have problems using the dummy *makedepend* script, Cyrus also includes its own *makedepend* in the *makedepend* subdirectory of the source distribution.

When using the dummy script, you should always run *make clean* before you run *makedepend* and *make all*:

```
% make clean
### Making clean in /usr/local/cyrus/cyrus-imapd-v1.5.19/man
rm -f *.o Makefile.bak
### Done with /usr/local/cyrus/cyrus-imapd-v1.5.19/man
### Making clean in /usr/local/cyrus/cyrus-imapd-v1.5.19/et
rm -f compile_et compile_et.o error_table.o
rm -f libcom_err.a
 (Many lines deleted...)
### Making clean in /usr/local/cyrus/cyrus-imapd-v1.5.19/pwcheck
rm -f *.o Makefile.bak pwcheck
### Done with /usr/local/cyrus/cyrus-imapd-v1.5.19/pwcheck
```

Next, run *makedepend* followed by *make all*. Your output should be similar to this:

```
% make depend
### Making depend in /usr/local/cyrus/cyrus-imapd-v1.5.19/man
### Making depend in /usr/local/cyrus/cyrus-imapd-v1.5.19/et
./config_script ./compile_et.sh nawk sed > compile_et
ed -s Makefile < eddep
rm eddep
echo '' >> Makefile
echo '# IF YOU PUT STUFF HERE IT WILL GO AWAY' >> Makefile
echo '# see makedepend above' >> Makefile
```

Run *make all* to build the executables:

```
% make all CFLAGS=-O
```

As your *make all* sputters out rather uneventfully, hopefully with no error messages, you can move on to the next step and create the *cyrus* account and add the account to the *mail* group. The procedure for doing so varies somewhat from system to system, depending on the OS and where your users live (NIS, DCE, */etc/passwd*, LDAP, etc.).

Once you've ensured that the *cyrus* account exists and that it has been added to the mail group, log in as *root* and do a *make install* to complete the install process and prepare your system for configuration. Your *make install* output should look something like this:

```
# make install
./install-sh -d /usr/local/bin
./install-sh -d /usr/local/lib
(Many lines skipped ...)
### Making install in /usr/local/cyrus/cyrus-imapd-v1.5.19/imtest
.././install-sh -c -s -m 755 imtest /usr/local/bin
### Making install in /usr/local/cyrus/cyrus-imapd-v1.5.19/cyradm
.././install-sh -c -s -m 755 cyradm /usr/local/bin
### Making install in /usr/local/cyrus/cyrus-imapd-v1.5.19/pwcheck
.././install-sh -c -s -m 755 pwcheck /usr/cyrus/bin
```

Upgrading from Previous Versions of Cyrus IMAP

Fresh installation of Cyrus may be all well trodden territory for you. If you're upgrading from a previous version of Cyrus IMAP, however, there are a few cleanup tasks to take care of after your upgrade. In addition to the notes that follow, be sure to read the contents of the *doc* directory carefully, with particular attention to the notes on upgrading.

- If you're upgrading from Version 1.5.2 or earlier, delete your *delivered* database and your PTS cache database (if you use the AFS filesystem), if they exist. If you've previously enabled duplicate delivery suppression, the delivered

database (*delivered.db*) is in your configuration directory. If you use AFS PTS group support, the PTS cache is in */var/ptclient*. Deleting both these files has no deleterious effects except that, in the case of the delivered database, some duplicate messages may be delivered.

- If you're upgrading from Version 1.4 or earlier, you must run *reconstruct –r* after installing the software and before putting it into production, so that the indexes for all your system's mailboxes are rebuilt.
- If you're upgrading from Version 1.3 or earlier, you'll also have to run a *reconstruct –m* to rebuild your mailboxes file.
- Finally, if you run *imspd* and *imapd* on the same machine and use AFS ACLs, you'll need to be running at least Version 1.5a5 of the IMSP server, because the format of the PTS cache for IMSP has changed.

Components of Cyrus and What They Do

Once you've built Cyrus, you'll have the following programs.

imapd(8)
: This is the IMAP server itself. It is invoked by *inetd*(8). In the next chapter, we'll cover how to set up *inetd* to run *imapd*, and ensure that it functions correctly.

deliver(8)
: This is the Cyrus mail delivery agent. *deliver* takes mail on standard input and delivers it to one or more Cyrus format mailboxes. It is also the pruning utility for the duplicate delivery database.

reconstruct(8)
: The *reconstruct* utility rebuilds the indexes of any number of Cyrus mailboxes and reconstructs the global mailboxes database. *reconstruct* is at the core of any conversion system from other mailbox formats to Cyrus, as well as most disaster recovery strategies. Once you've rebuilt a quota-restricted mailbox with *reconstruct*, it's a good idea to run *quota –f* to repair the appropriate quota root files.

cyradm(1)
: This is Cyrus's Tcl-based administrative control and scripting language. The interactive and scripting commands are different, and much if not all of what you can do in *cyradm* is also possible by spoofing the IMAP protocol from other languages if Tcl isn't your cup of tea.

pwcheck(8)

Cyrus *imapd* runs as user *cyrus*, so authentication using shadow passwords isn't directly available. *pwcheck* allows you to use shadow passwords on a non-Kerberized Cyrus system. *pwcheck* is a fairly robust hack that creates a for-*cyrus*-use-only named pipe that the root-owned *pwcheck* daemon uses to check username/password pairs.

quota(8)

This utility reports on the quota root limits and usage of given mailbox prefixes. Given with the *–f* switch, it also fixes quota inconsistencies before doing so.

fud(8)

fud is an experimental workaround to permit remote users with certain finger clients to display information about when a targeted user last read their mail, when mail last arrived at their mailbox, and how many messages are recent for that user. If you're not sure if your finger client supports the *fud* service, then it probably does not. LDAP and ACAP capabilities will likely supersede *fud* in the long run.

pop3d(8)

This daemon presents the Cyrus mailstore, or at least the inboxes, using the POP3 protocol. Good for sites transitioning from POP3 to IMAP4, so you can change the backend more or less transparently, then announce sometime later that IMAP is also available to your users.

rmnews(8)

Coordinates Usenet news article deletion between INN and IMAP.

syncnews(8)

Synchronizes IMAP's and INN's ideas about which newsgroups are active. Groups showing up as IMAP mailboxes but not in the *active* file are deleted. Groups showing up in the Usenet *active* file but not as IMAP mailboxes are created on the IMAP side.

arbitron(8)

Reports on mailbox readership statistics. Of negligible use with private mailboxes, but could be quite useful for sites that make use of shared mailboxes and want to determine who has read which mailboxes and when.

collectnews(8)

Used by INN to update the appropriate IMAP mailboxes when Usenet news arrives.

Common Problems

A number of common installation problems and their solutions are described in the Cyrus Installation FAQ (*http://asg.web.cmu.edu/cyrus/imapd/install-FAQ.html*). We cannot encourage you enough to visit that site if you run into problems during your installation! It has ceased to be a surprise when the same installation question comes up repeatedly on the list, even though the answer is available on the Installation FAQ.

Significant Bugs

- In Versions up to 1.5.19, duplicate delivery suppression displays misleading errors in the *imapd.log* that might cause you to believe a message wasn't delivered at all, when it actually was. This is fixed in 1.5.19 and later.
- There were a number of bugs that are fixed in Version 1.5.14. It is highly recommended that you upgrade if you're running a version older than 1.5.14.
- The IMAP RENAME command doesn't behave hierarchically in Cyrus releases prior to 2.0, so it's been temporarily advertised in the "capability" output as "X-NON-HIERARCHICAL-RENAME" in the interim before it's fixed. Hierarchical rename is fixed in Cyrus 2.0.
- *deliver* has a minor standard I/O problem in which text between a NUL and the end of line of a message will be lost. No fix is available as of the 1.5.19 release.

So much for building Cyrus. Let's move on to the next chapter and the configuration of your Cyrus server.