In this chapter:

- *IMAP Configuration File and Directory*
- *Configuring the Authentication Mechanism*
- *Configuring syslog*
- *Configuring the MTA*
- *Getting Cyrus Up and Running*
- *Testing Your Server*

8

Configuring the Cyrus Server

Now that the none-too-exciting work of building Cyrus IMAP is over, let's get down to business and configure it.

IMAP Configuration File and Directory

The Cyrus IMAP server has a single configuration file: */etc/imapd.conf*. No messing about with the source code or the *Makefile* is necessary. Once you've used the appropriate options to make a build of Cyrus that's appropriate for your site, any configuration is done simply by editing *imapd.conf*.

The Server Configuration File: imapd.conf

First, let's create the */etc/imapd.conf* file, which holds all the defining configuration parameters for *imapd*. Each line of *imapd.conf* contains an option and a value for the option, separated by a colon:

```
option: value
```

Blank lines and lines beginning with `#` are treated as comments and are ignored. Boolean options take the values `yes`, `1`, `t`, or `on` to turn the option on, and `no`, `0`, `f`, or `off` to turn the option off.

For the time being, it's best to minimize the number of entries in this file to make debugging your installation easier. The barebones IMAP configuration file must contain at least the following entries:

```
configdirectory: /var/imap
partition-default: /var/spool/imap
admins: cyrus johndoe2
```

See Table 8-1 for the purpose of each of the three options.

A user listed in *admins* should never use her administrative account to read mail!

Users listed as *admins* should never log in to the IMAP server to read mail. *admins* have special privileges that may lead to problems if they open a mailbox using certain IMAP clients. For example, *admins* are able to write to parts of the Cyrus system to which non-privileged users cannot write. Notably, if an administrative user reads mail, he might accidentally create top-level mailboxes (e.g., "Trash" or "Outbox") that other users would see as "public" mailboxes. To be on the safe side, do not even create IMAP mailboxes for your *admins*.

The complete set of options is listed in Table 8-1, and is also documented in the *imapd.conf*(5) manual page. Options that list a default value of "No default" have no default value. Options that list a default value of "None" default to an empty value.

Table 8-1. imapd.conf Options

Option	Default Value	Value
`configdirectory`	No default	Required. The pathname to the IMAP configuration directory. The widely used convention is `/var/imap`.
`defaultpartition`	`default`	The default partition (by partition-name, not path) on which new mailboxes are created.

`partition-`*`name`*	No default	Required. The full path of the partition *name*. At least one `partition-`*`name`* definition is required for the partition specified in the *defaultpartition* option. For example, if the *defaultpartition* is set as follows:
		`defaultpartition: default`
		then the required option is:
		`partition-default: /some/path`

`admins`	None	A list of users, delimited by whitespace, that have administrative rights. *admins* do not have the ability to change the server configuration. Any user can be listed in *admins*, including *cyrus*, but *admins* should never read IMAP mail using the account listed in *admins*.
`srvtab`	`/etc/srvtab`	Kerberos only. The full path of the *srvtab* file containing the IMAP server's private key. This option is only used if the server was compiled with Kerberos authentication.

Table 8-1. imapd.conf Options (continued)

Option	Default Value	Value
`umask`	`077`	Umask value used by programs under */usr/cyrus/bin*. By default, when those programs create files, the files have ownership (*cyrus: mail*) permissions: `-rw-------` If you want your *admins* to be able to read the files without becoming root, then set the value of `umask` to 027 and add your *admins* to group *mail*.
`allowanonymouslogin`	`no`	When set, permits the user *anonymous* to log in with any password. If you plan to provide anonymous access to public mail folders, set this option to `yes`.
`quotawarn`	`90`	Percent of quota usage over which the server sends a warning message to the user. You'll want to set this so that the user receives a warning when he is within 1 MB or so of his quota limit. Ninety percent is a good value for sites with quotas between 10 and 20 MB.
`timeout`	`30`	A value is required for this option. The length of inactivity, in minutes, after which the server logs the session out automatically. If you have large numbers of users logging in over dialups, keep the default—dialup users tend not to log out gracefully. Corporate sites whose users are on the LAN, on the other hand, would probably set the timeout much higher.
`imspservers`	None	List of hostnames of IMSP servers. This feature was never implemented and has been removed from Versions 1.6 and higher.
`defaultacl`	`anyone lrs`	The ACL to set by default on new mailboxes that do not have a parent mailbox. If the mailbox has a parent, it inherits the parent's ACL.
`newsspool`	No default	The pathname of the news spool directory. The `newsspool` option is used only if the option `partition-news` is defined.
`newsprefix`	None	The prefix added to the beginning of a newsgroup name to form the corresponding IMAP mailbox name.

Table 8-1. imapd.conf Options (continued)

Option	Default Value	Value
`autocreatequota`	`0`	The name of this option doesn't describe the option well—`autocreatequota` is dual purpose: the value determines whether or not a user can create his INBOX (and hence, his IMAP account) the first time he attempts to log in to the server. If the value is zero, the user cannot create his own account; if non-zero, the user can create his account. If the value is non-zero and positive, the user's quota is set to the value. If the value is non-zero and negative, the user is given an unlimited quota. Sites that have enterprise-wide usernames could save some work by enabling this option. Sites that want more control over the mailbox environment (such as pre-defined folders with per-folder quotas) should accept the default.
`logtimestamps`	`no`	When set, the server will log the number of seconds since the last command or response in the protocol telemetry logs.
`cleartextloginpause`	`0`	Specifies the number of seconds to wait after a successful cleartext authentication before opening the session. The purpose of this option is mainly to train users to associate a cost with using cleartext authentication. A pause after each login also substantially increases the amount of time it would take to crack any given password with a dictionary attack.
`loginrealms`	None	Kerberos only. List of remote realms whose users may log in using cross-realm authentication. The realms should be delimited by whitespace.
`loginuseacl`	`no`	Kerberos only. When set, an identity that has rights on an INBOX may log in as the owner of the INBOX.
`reject8bit`	`no`	When set, the *deliver* program rejects messages with 8-bit characters in their headers. If not set, then 8-bit characters are changed to the letter X.
`netscapeurl`	`http://andrew2.andrew.cmu.edu/cyrus/imapd`	Specifies the site to contact when Netscape queries the IMAP server for the location of the administration HTTP server. This option must be enabled at compile time. The default site provides only an informational message.

The Configuration Directory

The configuration directory is the repository for information on all components and user data that make up the Cyrus system. The popular convention is to name the configuration directory */var/imap* (that is the convention we will use in our examples). Access to the configuration directory should be restricted to the *cyrus* user and group only. The following commands create the configuration directory and set the correct permissions and ownership:

```
# cd /var
# mkdir imap
# chown cyrus:mail imap
# chmod 0750 imap
```

The Cyrus server is finicky about file permissions and ownership. Be sure the ownership and permissions are set correctly to prevent problems from happening later.

Odds and Ends

Create the supporting files in the configuration directory with the permissions shown. Create an empty *mailboxes* file and the directories that Cyrus will use to store its configuration information:

```
# cd /var/imap
# touch mailboxes
# mkdir user quota proc log msg
# chown cyrus:mail *
```

Create the directory that was defined in *imapd.conf* as `defaultpartition`. The `defaultpartition` directory is where users' mailboxes are stored. Recall that the `defaultpartition` was defined as */var/spool/imap* in the sample *imapd.conf* shown earlier in this chapter.

```
# cd /var/spool
# mkdir imap
# chown cyrus:mail imap
# chmod 750 imap
```

Configuring the Authentication Mechanism

This section shows you how to configure the IMAP server to perform cleartext shadow password and Kerberos authentication. Cleartext authentication without shadow passwords does not require special configuration. The mechanism for

If You're Configuring Cyrus on a Linux System . . .

You would be well served to set the configuration directory, mailstore, user, quota, and mailstore directory for synchronous updates.

```
# cd /var/imap
# chattr +S . user quota
# chattr +S /var/spool/imap
# chattr +S /var/spool/mqueue
```

Doing so allows you to purchase a slightly higher amount of robustness in exchange for a degree of performance (although if your system has more than 5,000 mailboxes, the performance trade-off has been shown to be too high to support synchronous updates). If a file is updated asynchronously, then the cache associated with that file is flushed to disk some time later. If it's set to update synchronously, its cache is flushed immediately. When the cache is flushed immediately, there's less risk of damage to your system if, for example, it were to halt inadvertently.

Note that this advice applies only to *ext2* filesystems. If you're not sure what type of filesystem you have, use the *mount*(8) command.

authenticating users to the Cyrus server is external to the server itself and eliminates the need for users to have Unix accounts on the server. It's possible to authenticate against an LDAP directory or SQL database, for example. Alternative authentication mechanisms and their associated tools are described in Chapter 18, *IMAP Tools*.

Cleartext Authentication with Shadow Passwords

Cleartext authentication is nothing more than a check against your local *passwd* file. Some flavors of Unix store encrypted passwords right along with usernames and other account information in the local password file. Others store the username and account information in one file and the encrypted passwords in another, separate file (the shadow file). The shadow file has strict access rights—it's owned by root and is readable only by root—and thus protects encrypted passwords from prying eyes.

Cyrus IMAP authentication support changed significantly between the 1.5.19 and the 1.6.22 releases. Versions 1.6 and higher support the *pwcheck* daemon for shadow password authentication from within SASL. There are slight differences in configuring the authentication in Versions 1.5.19 and 1.6.22, which we'll mention later in this section.

Setting up cleartext authentication in Cyrus Version 1.5.19

The *pwcheck* program was introduced in Chapter 7, *Installing the Cyrus IMAP Server*—it is a daemon that the Cyrus server uses to check passwords against the Unix shadow password file. *pwcheck* is required because the Cyrus IMAP server runs as user *cyrus*, and hence does not have read access to the shadow password file (it's readable only by *root*). *pwcheck* creates a named pipe under the directory */var/pwcheck* that the Cyrus server uses to communicate with the *pwcheck* daemon. If you configured your Cyrus server to use *pwcheck*, then you must create that directory and set the permissions appropriately:

```
# mkdir /var/pwcheck
# chown cyrus /var/pwcheck
# chmod 0700 /var/pwcheck
```

pwcheck is meant to be started at system boot time and run as a background process. To start it, create an initialization script for *pwcheck* and arrange for that script to be run at boot time. On a Solaris system, create the file */etc/init.d/pwcheck* shown in Example 8-1.

Example 8-1. pwcheck Startup Script

```
#!/sbin/sh
#
#    Start CMU Cyrus pwcheck daemon

case "$1" in
'start')
        if [ -f /usr/cyrus/bin/pwcheck ]; then
                /usr/cyrus/bin/pwcheck &
        fi
        ;;

'stop')
        pid=`/usr/bin/ps -eo pid,comm | /usr/bin/awk '{ \
            if ($2 == "/usr/cyrus/bin/pwcheck") print $1 }'`
        if test "$pid"
        then
            /usr/bin/kill $pid
        fi
        ;;
*)
        echo "Usage: $0 { start | stop }"
        exit 1
        ;;
esac
exit 0
```

Once */etc/init.d/pwcheck* is in place, arrange for *pwcheck* to be started or stopped when the system is booted or shut down by creating a link from */etc/rc3.d* to the initialization script you created:

```
# cd /etc/rc3.d
# ln -s /etc/init.d/pwcheck S98pwcheck
# cd /etc/rc2.d
# ln -s /etc/init.d/pwcheck K20pwcheck
```

The names of the links to the *pwcheck* script depend on what's already in */etc/rc3.d* and when you want it run in the startup process—any more detail than that is beyond the scope of this book.

Setting up cleartext authentication in Version 1.6.22

Cyrus 1.6.22 uses the SASL framework for authentication. Cyrus SASL 1.5.15 is a separate package that is a prerequisite for Cyrus 1.6.22. Its installation was covered in Chapter 7. On most systems, no special configuration is required. SASL uses the authentication that you specified when you ran the Cyrus configure script. If none is specified, it defaults to the most secure level of authentication available. If you're using cleartext passwords, you should have configured Cyrus with the configure option `--with-auth-unix`.

If you use plain passwords with no shadow, there is no special configuration required. On most systems, SASL will use PAM to authenticate using the plain password. If you do use shadow passwords, there may be some extra work involved. On Linux systems, for example, the permissions on the shadow file have to be changed to allow the *cyrus* user to read it. On Solaris systems, some tweaking of PAM is necessary. Most sites that depend on shadow passwords have opted to either stick with Cyrus 1.5.19 or use *pwcheck* in its most simplistic form with Cyrus 1.6.22.

Kerberos Authentication

If you compiled Cyrus to support Kerberos, you'll need to create a Kerberos key for the Cyrus IMAP server and add the key to the *srvtab* file.

The following example creates a key for the hostname *rooster*. *rooster*'s Kerberos realm is *THEMULLETS.NET*.

```
# ksrvutil -f /etc/srvtab add
Name: imap
Instance: rooster
Realm: THEMULLETS.NET
Version number: <Return>
New principal: imap.rooster@THEMULLETS.NET; version 0
Is this correct? (y/n) [y] y
Password: xxxxxxxx
```

```
Verifying, please re-enter Password: xxxxxxxx
Key successfully added.
Would you like to add another key? (y/n) [y] n
```

Finally, give ownership of */etc/srvtab* to the *cyrus* user:

```
# chown cyrus /etc/srvtab
```

Configuring syslog

The Cyrus server uses the BSD 4.3 variant of *syslog*. BSD 4.3 *syslog* separates log messages into both facilities and severity levels. Before configuring *syslog* on your system to log messages from the Cyrus server, you will need to determine whether your *syslog* is the BSD 4.3 variant. Run the command:

```
% man syslog
```

Look at the definition of the `openlog()` function in the synopsis:

```
void openlog(const char *ident, int logopt, int facility);
```

The `openlog()` function takes either two or three arguments. If the `openlog()` function takes three arguments, then your *syslog* is a BSD 4.3 variant and is compatible with Cyrus *syslog* function calls. To configure *syslog*, edit */etc/syslog.conf* to include a line similar to the following line, which tells *syslog* to log debug level messages to the *local6* facility and write them to the log file */var/log/imapd.log*:

```
local6.debug                /var/log/imapd.log
```

Then create an empty *imapd.log* file and restart *syslog*:

```
# touch /var/log/imapd.log
# /etc/init.d/syslog stop; /etc/init.d/syslog start
```

If your system's `openlog()` function takes only two arguments, then it's not the BSD 4.3 variant and you must use the *syslogd* and *syslog.conf* that are provided with the Cyrus distribution. Make backup copies of your system's *syslogd* and *syslog.conf*, then change directory to the top level of your Cyrus source distribution:

```
(Stop syslog)
# cd syslog
# cp syslogd /etc/syslogd
# cp syslog.conf /etc/syslog.conf
(Start syslog)
```

Configuring the MTA

Unlike the UW server, Cyrus IMAP's mailstore format ties it intrinsically to the local mail transport agent or, more accurately, to the local mail delivery agent. In this book, we're presuming that you've chosen *sendmail* as your MTA. *sendmail* can serve just about any size user base. If, however, you elect to use another MTA, be

mindful of the fact that you'll have to configure it to use the Cyrus *deliver* program as a delivery agent, and use this section as a rough guide.

The deliver MDA

The Cyrus *deliver* program is the mail delivery agent that drops mail messages into users' mailboxes. *deliver* takes a mail message on standard input and delivers it to the specified mailboxes. *deliver*'s configuration options are set in */etc/imapd.conf*.

deliver uses the options listed in Table 8-2 when invoked to deliver mail. Other options are described in the *deliver*(8) manual page.

Table 8-2. deliver Options

Option	Description

`-m` *`mailbox`*	Deliver a message to the Cyrus mailbox *`mailbox`*.
	To deliver to a specific mailbox, for example *user.johndoe.lists*, use
	`deliver -m user.johndoe.lists`
	You must have *p* access rights on the specified mailbox; if you don't, then delivery fails and returns the message:[a]
	`user.johndoe.lists: Mailbox does not exist`
	If a mailbox is specified with the –m argument and a username argument is given, then *deliver* will attempt delivery to the specified mailbox under the mailbox hierarchy belonging to the username. For example, the command:
	`/usr/cyrus/bin/deliver -m lists johndoe`
	delivers a message to *user.johndoe.lists*. Again, the user invoking *deliver* must have *p* access rights (access rights are described in Chapter 9, *Cyrus System Administration*) on the specified mailbox.
	If a mailbox and list of usernames are specified, *deliver* will attempt to deliver the message to the mailbox for each username. For example, the command:
	`/usr/cyrus/bin/deliver –m lists johndoe msmith kjones`
	delivers a message to *user.johndoe.lists*, *user.msmith.lists*, and *user.kjones.lists*.

`-e`	Enable duplicate delivery suppression.
`-q`	Force delivery of a message when the specified mailbox is over quota.
`-F` *`flag`*	Set the flag *`flag`* on the delivered message. *`flag`* can take the values `\seen`, `\answered`, `\flagged`, `\draft`, or `\deleted`.
`-a` *`authID`*	Specify the authorization ID *`authID`* of the sender. If no value for *`authID`* is given, defaults to `anonymous`. *`authID`* is a way that person A could allow person B to use his authorized privileges without sharing his password (person B would use her own password).
`-r` *`address`*	Insert a *Return-Path:* header containing *`address`* at the top of the message.
`-f` *`address`*	Identical to *–r* argument.

[a] The error message is misinformational—it is returned whether or not the mailbox actually exists if you do not have *p* rights on the mailbox.

To manually deliver the message contained in the file *39.* to *johndoe*'s mailbox from the Unix shell prompt, use the command:

```
% /usr/cyrus/bin/deliver -m user.johndoe johndoe < 39.
```

You must specify both the mailbox name (*user.johndoe*) using the *–m* argument, and the username that the mailbox belongs to (*johndoe*). If you do not specify the username, *deliver* will assume that you want to deliver the mail to the mailbox belonging to the user running the program from the command line. For example, if you are logged in as *smith* and you run the previous example command from your shell prompt, *deliver* will attempt to find a mailbox called *user.johndoe* in your (*smith*'s) mailbox hierarchy, and will fail. If you specify the username of the recipient but not the mailbox name, then *deliver* will attempt to deliver the message to the mailbox *user.username* by default.

Some delivery agents are configured to generate a Unix-style *From** header. *deliver* does not handle the *From* header. If you try to use *deliver* on the command line to deliver a message that contains a *From* header, *deliver* will fail. For example, if the file *39.* contains the line:

```
>From johndoe@localhost Fri Jul 16 11:31:13 1999
```

The *deliver* command will return the message:

```
johndoe: Message contains invalid header
```

As we'll see in the next section, *deliver* should be configured in your *sendmail* configuration *not* to generate the *From* header. The message should minimally contain *To:*, *From:*, *Subject:*, and *Date:* headers. If those headers are missing, the message will be delivered, but it is difficult to say what the corresponding fields in the mail client will contain when the recipient reads his mail. *deliver* will deliver a message that contains no header at all if the message begins with a blank line.

The sendmail Configuration File

sendmail versions newer than 8.7 include support for Cyrus and include a prototype M4 macro file that can be used to build a basic *sendmail.cf* configuration file. This section provides basic instructions for building a *sendmail.cf* file to support Cyrus. The instructions assume that *sendmail* is already installed and that the source distribution is available.

---

* Not to be confused with the RFC 822 *From:* header, we are referring here to the "MTA" *From* header, which has no colon delimiter and is appended to the message by the MTA (*sendmail*).

Build the sendmail configuration file

Change directory to the top level of the *sendmail* source tree. An *ls* should show most or all of the following files:

```
% ls -CF
FAQ             RELEASE_NOTES   doc/            makemap/        smrsh/
KNOWNBUGS       cf/             mail.local/     praliases/      src/
READ_ME         contrib/        mailstats/      rmail/          test/
```

The M4 macros are located in the *cf/cf* subdirectory. Under that directory, you will see a file named *cyrusproto.mc*:

```
% cd cf/cf
% ls cyrusproto.mc
cyrusproto.mc
```

cyrusproto.mc is an M4 macro script used to build a *sendmail* configuration file. Before building your configuration file, you'll need to edit the macro script to specify your operating system version. If you have a domain-specific *sendmail* configuration, you should also include a statement to define your domain. The OSTYPE variable is used to specify the operating system type. The supported values of OSTYPE can be found in the *cf/ostype* directory in the *sendmail* source distribution. For a Solaris 2.x system in the *unt.edu* domain, the following lines would have to be added to *cyrusproto.mc*:

```
OSTYPE(solaris2.ml)
DOMAIN('UNT.EDU')
```

To build a bare-bones configuration file that supports Cyrus, use *m4*:

```
% m4 ../m4/cf.m4 cyrusproto.mc > cyrusproto.cf
```

The *cyrusproto.cf* is a modified version of the *sendmail.cf* that uses Cyrus deliver as the MDA. The MDA specification, as it appears in *cyrusproto.cf*, is shown in Example 8-2.

Example 8-2. Cyrus Mailer Specification

```
###################################################
###   Cyrus Mailer specification                ###
###################################################

#####  @(#)cyrus.m4     8.4 (Carnegie Mellon) 9/2/96  #####

Mcyrus,     P=/usr/cyrus/bin/deliver, F=lsDFMnPqA5@W, S=10, R=20/40,
            U=cyrus:mail,
            A=deliver -m $h -- $u
```

The specification translates as follows (consult the sendmail* book for a more detailed understanding):

`Mcyrus`
: The name of this mailer definition is "cyrus."

`P=/usr/cyrus/bin/deliver`
: Path to the *deliver* program.

`F=lsDFMnPqA5@W`
: The list of delivery flags that tell *deliver* how to behave. In particular, the *n* flag tells the mailer not to include the Unix-style *From* header.

`S=10`
: Use ruleset 10 to process both the envelope and header sender addresses.

`R=20/40`
: Use ruleset 20 to process the envelope recipient address and ruleset 40 to process the header recipient address.

`U=cyrus:mail`
: The user and group to become when running the cyrus mailer. *deliver* must always run as user *cyrus*.

`A=deliver-m $h -- $u`
: The *deliver* program and its arguments, as described in Table 8-2.

Copy the *cyrusproto.cf* file you just created into the directory where you normally keep your *sendmail* configuration file (usually */etc/mail*), then restart *sendmail*:

```
% su -
# cp /etc/mail/sendmail.cf /etc/mail/sendmail.cf.bak
# cp cyrusproto.cf /etc/mail/sendmail.cf
```

The Cyrus installation document tells you to add the user *daemon* to the mail group in */etc/group*, but that is unnecessary if you're running a modern incarnation of *sendmail* (i.e., the recommended Version 8.7.1 or better). *sendmail* runs as *root*, so the groups it belongs to are irrelevant as far as *deliver* is concerned:

```
# /etc/init.d/sendmail stop; /etc/init.d/sendmail start
```

---

* *sendmail*, by Bryan Costales with Eric Allman (O'Reilly).

Testing the sendmail configuration

First and most important: you must set up a Cyrus test account to which mail can be delivered. Use the commands below to set up a basic test account called "debug." If you're using Unix authentication, be sure to put the user in the local password file:

```
% cyradm -user cyrus localhost imap
localhost password:
localhost> cm user.debug
```

Next, use *sendmail* on the command line to deliver a test message to the test account:

```
% su - debug
Password: xxxxxxxx
Sun Microsystems Inc.   SunOS 5.7       Generic October 1998
$ echo "Subject: Testing 1 2 3" | /usr/lib/sendmail -v debug
debug... Connecting to cyrus...
debug... Sent
```

The message should appear in the *debug* user's mailbox. You can check this without using a mail client by looking at the contents of debug's top level mailbox:

```
$ cd /var/spool/cyrus/user/debug
$ ls -ltr
total 18
-rw-------   1 cyrus    mail         135 Jun 19 09:54 cyrus.header
-rw-------   1 cyrus    mail          53 Jul 17 12:19 cyrus.seen
-rw-------   1 cyrus    mail          96 Jul 17 20:08 cyrus.index
-rw-------   1 cyrus    mail         488 Jul 17 20:08 cyrus.cache
-rw-------   1 cyrus    mail         289 Jul 17 20:08 1.
$ cat 1.
Return-Path: <debug>
Received: (from debug@localhost)
        by localhost (8.9.1/8.9.1) id UAA25991
        for debug; Sat, 17 Jul 1999 20:08:36 -0500 (CDT)
Date: Sat, 17 Jul 1999 20:08:36 -0500 (CDT)
From: debug
Message-Id: <199907180108.UAA25991@localhost>

Testing 1 2 3
```

Duplicate delivery suppression and the delivered database maintenance

deliver can be invoked by *sendmail* with the *–e* option enabled. The *–e* option suppresses delivery of messages that have a *Message-ID:* header identical to a message that has already been delivered to a given mailbox. Information on mail deliveries is maintained in the *delivered* database (see Chapter 6, *Introduction to the Cyrus IMAP Server*, for a description). The *delivered* database should be pruned periodically to keep it from growing too large. To keep the *delivered* database

clean, run *deliver* with the *–E* argument every day or so. Create a *crontab* entry to run as the *cyrus* user.* The *crontab* entry would look like this:

```
0 2 * * * /usr/cyrus/bin/deliver -E 3
```

The *–E 3* argument, for example, tells *deliver* to prune the *delivered* database of entries older than 3 days. If the database is small, you may decide to prune it less often. There is a cost, albeit a small one, in using duplicate delivery suppression—when it's turned on, every delivery accesses the *delivered* database. The *delivered* database may also be problematic during Cyrus upgrades; depending on the new version of Cyrus being installed, there may be a rebuild required of the *delivered* database. Some sites opt out of suppressing duplicate deliveries at all. The *delivered* database is really only required in Versions 1.6 and higher to support Sieve filtering.

Getting Cyrus Up and Running

Edit */etc/services*, if necessary, to contain the following line:

```
imap    143/tcp
```

Edit */etc/inetd.conf* to include the line:

```
imap stream tcp nowait cyrus /usr/cyrus/bin/imapd imapd
```

Once the files have been edited, restart *inetd*. On most Unix systems, find the process ID, then send a HUP signal to the process using the *kill* command:

```
# ps -ef | grep inetd
root 13005    1  0 22:50:57 ? 0.00    0:00 /usr/sbin/inetd -s
# kill -HUP 13005
```

If your Unix supports the *pkill* command, then you can save a step by using the command:

```
# /usr/bin/pkill -HUP -x inetd
```

Testing Your Server

You've configured your server, in time to just tell everyone it's in production and leave on your vacation for the Bahamas. On second thought, maybe it would be a good idea to test it first.

---

* It is imperative that the *crontab* entry belong to the *cyrus* user, *not* to *root* or some other user listed under *admins* in the IMAP configuration file.

Testing a Cyrus Installation on the Same Machine as Your Production Server

If you are currently running an IMAP server and wish to test a Cyrus installation without disabling the current IMAP server, specify an alternate name and port in */etc/services*, such as:

```
imaptest 243/tcp
```

Add the following line to */etc/inetd.conf*:

```
imaptest stream tcp nowait cyrus /usr/cyrus/bin/imapd imapd
```

After restarting *inetd*, the *imaptest* server will be running on port 243. Switching from the old to the new server is a simple matter of changing the name of the *imaptest* service *in /etc/inetd.conf* to *imap* and restarting *inetd*.

Caution is advised: the test server should never modify files that belong to the production server. Configure your test server's *imapd.conf* to use a different set of configuration files and a different mailstore than your production server uses.

Check That the Server Is Running

Most simple test first. Let's check to see if the IMAP listen is being serviced on the right port and if the Cyrus server is on the other end of that listen. As a normal (i.e., non-administrative) user, *telnet* to the IMAP port on your machine:

```
% telnet localhost imap
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
* OK localhost Cyrus IMAP4 v1.5.19 server ready
. logout
* BYE LOGOUT received
. OK Completed
Connection closed by foreign host.
```

The command *. logout* closes the connection. If you see a message that begins with *. OK*, then the server is running. Any other message, or no message at all, indicates a problem.

Testing Cleartext Password Authentication

If you use cleartext password authentication, take advantage of the *imtest* program to test authentication.* Run the *imtest* program as user *cyrus* or another existing

* You could *telnet* to the IMAP port and log in using your cleartext password, but some people are (rightfully) nervous about seeing passwords in cleartext on their display. *imtest* does not echo the password.

IMAP account on your system (at this point you probably have not created user accounts yet and have only the *cyrus* account). The following example is run from the *cyrus* account:

```
% /usr/local/bin/imtest -p localhost imap
* OK localhost Cyrus IMAP4 v1.5.19 server ready
Password:
. LOGIN cyrus {L+}
X
. OK User logged in
. logout
```

The reply message *. OK User logged in* indicates that authentication is working. The message *. NO Login incorrect* can indicate any of the following problems:

- The password entered was incorrect.
- The user running *imtest* does not have an IMAP account on the system.
- *pwcheck* should be running and is not.
- There is an error in permissions or ownership somewhere, and it needs correction.

Type the command *. logout* to close the connection and quit.

Testing Kerberos Authentication

If your server uses Kerberos authentication, you can also use the *imtest* command to make sure Kerberos authentication is working. As *cyrus* or another existing user on your system, enter the command:

```
% /usr/local/bin/imtest -k localhost imap
```

If the output ends with the message:

```
. OK User logged in (no protection)
```

then Kerberos authentication is working. Any other message indicates a failure. More specific error messages are logged to the *imapd.log* file—check there for hints about what the source of the problem might be. To end the test and close the connection, type the command *. logout.*