GNU Mailman - Installation Manual

Release 2.1

Barry Warsaw

July 19, 2013

barry (at) list dot org

Abstract

This document describes how to install GNU Mailman on a POSIX-based system such as UNIX, MacOSX, or GNU/Linux. It will cover basic installation instructions, as well as guidelines for integrating Mailman with your web and mail servers.

The GNU Mailman website is at http://www.list.org

Contents

1 Installation Requirements

Please note that the information on this page may be out of date. Check for the latest installation information on the Mailman wiki.

GNU Mailman works on most POSIX-based systems such as UNIX, MacOSX, or GNU/Linux. It does not currently work on Windows. You must have a mail server that you can send messages to, and a web server that supports the CGI/1.1 API. Apache makes a fine choice for web server, and mail servers such as Postfix, Exim, Sendmail, and qmail should work just fine.

To install Mailman from source, you will need an ANSI C compiler to build Mailman's security wrappers. The GNU C compiler gcc works well.

You must have the Python interpreter installed somewhere on your system. As of this writing, Python 2.4.4 is recommended, but see the wiki page above for the latest information.

2 Set up your system

Before installing Mailman, you need to prepare your system by adding certain users and groups. You will need to have root privileges to perform the steps in this section.

2.1 Add the group and user

Mailman requires a unique user and group name which will own its files, and under which its processes will run. Mailman's basic security is based on group ownership permissions, so it's important to get this step right[1]. Typically, you will add a new user and a new group, both called `mailman`. The `mailman` user must be a member of the `mailman` group. Mailman will be installed under the `mailman` user and group, with the set-group-id (setgid) bit enabled.

If these names are already in use, you can choose different user and group names, as long as you remember these when you run **configure**. If you choose a different unique user name, you will have to specify this with **configure**'s **--with-username** option, and if you choose a different group name, you will have to specify this with **configure**'s **--with-groupname** option.

On Linux systems, you can use the following commands to create these accounts. Check your system's manual pages for details:

```
% groupadd mailman
% useradd -c''GNU Mailman'' -s /no/shell -d /no/home -g mailman mailman
```

2.2 Create the installation directory

Typically, Mailman is installed into a single directory, which includes both the Mailman source code and the run-time list and archive data. It is possible to split the static program files from the variable data files and install them in separate directories. This section will describe the available options.

The default is to install all of Mailman to '/usr/local/mailman'[2]. You can change this base installation directory (referred to here as *$prefix*) by specifying the directory with the **--prefix configure** option. If you're upgrading from a previous version of Mailman, you may want to use the **--prefix** option unless you move your mailing lists.

> **Warning:** You cannot install Mailman on a filesystem that is mounted with the `nosuid` option. This will break Mailman, which relies on setgid programs for its security. If this describes your environment, simply install Mailman in a location that allows setgid programs.

Make sure the installation directory is set to group `mailman` (or whatever you're going to specify with **--with-groupname**) and has the setgid bit set[3]. You probably also want to guarantee that this directory is readable and executable by everyone. For example, these shell commands will accomplish this:

```
% cd $prefix
% chgrp mailman .
% chmod a+rx,g+ws .
```

[1] You will be able to check and repair your permissions after installation is complete.
[2] This is the default for Mailman 2.1. Earlier versions of Mailman installed everything under '/home/mailman' by default.
[3] BSD users should see the 15.2 section for additional information.

Warning: The installation directory, *$prefix*, cannot be the same directory that the source tarball has been unpacked to and in which you run **configure**, but it can, if you wish, be a subdirectory, e.g., *$prefix/src*.

You are now ready to configure and install the Mailman software.

3 Build and install Mailman

3.1 Run configure

Before you can install Mailman, you must run **configure** to set various installation options your system might need.

Note: Take special note of the **--with-mail-gid** and **--with-cgi-gid** options below. You will probably need to use these.

You should **not** be root while performing the steps in this section. Do them under your own login, or whatever account you typically use to install software. You do not need to do these steps as user `mailman`, but you could. However, make sure that the login used is a member of the `mailman` group as that that group has write permissions to the *$prefix* directory made in the previous step. You must also have permission to create a setgid file in the file system where it resides (NFS and other mounts can be configured to inhibit setgid settings).

If you've installed other GNU software, you should be familiar with the **configure** script. Usually you can just **cd** to the directory you unpacked the Mailman source tarball into, and run **configure** with no arguments:

```
% cd mailman-<version>
% ./configure
% make install
```

The following options allow you to customize your Mailman installation.

--prefix=*dir* Standard GNU configure option which changes the base directory that Mailman is installed into. By default *$prefix* is '/usr/local/mailman'. This directory must already exist, and be set up as described in 2.2.

--exec-prefix=*dir* Standard GNU configure option which lets you specify a different installation directory for architecture dependent binaries.

--with-var-prefix=*dir* Store mutable data under *dir* instead of under the *$prefix* or *$exec_prefix*. Examples of such data include the list archives and list settings database.

--with-python='/path/to/python' Specify an alternative Python interpreter to use for the wrapper programs. The default is to use the interpreter found first on your shell's *$PATH*.

--with-username=*username-or-uid* Specify a different username than `mailman`. The value of this option can be an integer user id or a user name. Be sure your *$prefix* directory is owned by this user.

--with-groupname=*groupname-or-gid* Specify a different groupname than `mailman`. The value of this option can be an integer group id or a group name. Be sure your *$prefix* directory is group-owned by this group.

--with-mail-gid=*group-or-groups* Specify an alternative group for running scripts via the mail wrapper. *group-or-groups* can be a list of one or more integer group ids or symbolic group names. The first value in the list that resolves to an existing group is used. By default, the value is the list `mailman`, `other`, `mail`, and `daemon`.

Note: This is highly system dependent and you must get this right, because the group id is compiled into the mail wrapper program for added security. On systems using **sendmail**, the 'sendmail.cf' configuration file designates the group id of **sendmail** processes using the *DefaultUser* option. (If commented out, it still may be indicating the default...)

Check your mail server's documentation and configuration files to find the right value for this switch.

--with-cgi-gid=*group-or-groups* Specify an alternative group for running scripts via the CGI wrapper. *group-or-groups* can be a list of one or more integer group ids or symbolic group names. The first value in the list that resolves to an existing group is used. By default, the value is the the list `www`, `www-data`, and `nobody`.

Note: The proper value for this is dependent on your web server configuration. You must get this right, because the group id is compiled into the CGI wrapper program for added security, and no Mailman CGI scripts will run if this is incorrect.

If you're using Apache, check the values for the *Group* option in your 'httpd.conf' file.

--with-cgi-ext=*extension* Specify an extension for cgi-bin programs. The CGI wrappers placed in '*$prefix*/cgi-bin' will have this extension (some web servers require an extension). *extension* must include the leading dot.

--with-mailhost=*hostname* Specify the fully qualified host name part for outgoing email. After the installation is complete, this value can be overriden in '*$prefix*/Mailman/mm_cfg.py'.

--with-urlhost=*hostname* Specify the fully qualified host name part of urls. After the installation is complete, this value can be overriden in '*$prefix*/Mailman/mm_cfg.py'.

--with-gcc=no Don't use gcc, even if it is found. In this case, **cc** must be found on your *$PATH*.

3.2 Make and install

Once you've run **configure**, you can simply run **make**, then **make install** to build and install Mailman.

4 Check your installation

After you've run **make install**, you should check that your installation has all the correct permissions and group ownerships by running the **check_perms** script. First change to the installation (i.e. *$prefix*) directory, then run the **bin/check_perms** program. Don't try to run bin/check_perms from the source directory; it will only run from the installation directory.

If this reports no problems, then it's very likely ¡wink¿ that your installation is set up correctly. If it reports problems, then you can either fix them manually, re-run the installation, or use **bin/check_perms** to fix the problems (probably the easiest solution):

- You need to become the user that did the installation, and that owns all the files in *$prefix*, or root.
- Run **bin/check_perms -f**
- Repeat previous step until no more errors are reported!

Warning: If you're running Mailman on a shared multiuser system, and you have mailing lists with private archives, you may want to hide the private archive directory from other users on your system. In that case, you should drop the other execute permission (o-x) from the 'archives/private' directory. However, the web server process must be able to follow the symbolic link in public directory, otherwise your public Pipermail archives will not work. To set this up, become root and run the following commands:

```
# cd <prefix>/archives
# chown <web-server-user> private
# chmod o-x private
```

You need to know what user your web server runs as. It may be `www`, `apache`, `httpd` or `nobody`, depending on your server's configuration.

5 Set up your web server

Congratulations! You've installed the Mailman software. To get everything running you need to hook Mailman up to both your web server and your mail system.

If you plan on running your mail and web servers on different machines, sharing Mailman installations via NFS, be sure that the clocks on those two machines are synchronized closely. You might take a look at the file 'Mailman/LockFile.py'; the constant *CLOCK_SLOP* helps the locking mechanism compensate for clock skew in this type of environment.

This section describes some of the things you need to do to connect Mailman's web interface to your web server. The instructions here are somewhat geared toward the Apache web server, so you should consult your web server documentation for details.

You must configure your web server to enable CGI script permission in the '*$prefix*/cgi-bin' to run CGI scripts. The line you should add might look something like the following, with the real absolute directory substituted for *$prefix*, of course:

```
Exec        /mailman/*      $prefix/cgi-bin/*
```

or:

```
ScriptAlias /mailman/       $prefix/cgi-bin/
```

> **Warning:** You want to be very sure that the user id under which your CGI scripts run is **not** in the `mailman` group you created above, otherwise private archives will be accessible to anyone.

Copy the Mailman, Python, and GNU logos to a location accessible to your web server. E.g. with Apache, you've usually got an 'icons' directory that you can drop the images into. For example:

```
% cp $prefix/icons/*.{jpg,png} /path/to/apache/icons
```

You then want to add a line to your '*$prefix*/Mailman/mm_cfg.py' file which sets the base URL for the logos. For example:

```
IMAGE_LOGOS = '/images/'
```

The default value for *IMAGE_LOGOS* is '/icons/'. Read the comment in 'Defaults.py.in' for details.

Configure your web server to point to the Pipermail public mailing list archives. For example, in Apache:

```
Alias   /pipermail/     $varprefix/archives/public/
```

where *$varprefix* is usually *$prefix* unless you've used the **--with-var-prefix** option to **configure**. Also be sure to configure your web server to follow symbolic links in this directory, otherwise public Pipermail archives won't be accessible. For Apache users, consult the *FollowSymLinks* option.

If you're going to be supporting internationalized public archives, you will probably want to turn off any default charset directive for the Pipermail directory, otherwise your multilingual archive pages won't show up correctly. Here's an example for Apache, based on the standard installation directories:

```
<Directory "/usr/local/mailman/archives/public/">
    AddDefaultCharset Off
</Directory>
```

Now restart your web server.

6 Set up your mail server

This section describes some of the things you need to do to connect Mailman's email interface to your mail server. The instructions here are different for each mail server; if your mail server is not described in the following subsections, try to generalize from the existing documentation, and consider contributing documentation updates to the Mailman developers.

Under rare circumstances or due to mis-configuration, mail to the owner(s) of the 'mailman' site-list (see section 8) can bounce. In order to prevent a mail loop this mail is sent with envelope from mailman-loop which is normally aliased as

```
mailman-loop: $varprefix/data/owner-bounces.mbox
```

but which can be aliased to any, always deliverable, local address or file. If you are using the Postfix MTA integrated as described in section 6.1, this alias will be generated automatically. In all other cases, you should install this alias along with your normal system aliases.

6.1 Using the Postfix mail server

Mailman should work pretty much out of the box with a standard Postfix installation. It has been tested with various Postfix versions up to and including Postfix 2.1.5.

In order to support Mailman's optional VERP delivery, you will want to disable `luser_relay` (the default) and you will want to set `recipient_delimiter` for extended address semantics. You should comment out any `luser_relay` value in your 'main.cf' and just go with the defaults. Also, add this to your 'main.cf' file:

```
recipient_delimiter = +
```

Using '+' as the delimiter works well with the default values for *VERP_FORMAT* and *VERP_REGEXP* in 'Defaults.py'.

When attempting to deliver a message to a non-existent local address, Postfix may return a 450 error code. Since this is a transient error code, Mailman will continue to attempt to deliver the message for *DELIVERY_RETRY_PERIOD* – 5 days by default. You might want to set Postfix up so that it returns permanent error codes for non-existent local users by adding the following to your 'main.cf' file:

```
unknown_local_recipient_reject_code = 550
```

Finally, if you are using Postfix-style virtual domains, read the section on virtual domain support below.

Integrating Postfix and Mailman

You can integrate Postfix and Mailman such that when new lists are created, or lists are removed, Postfix's alias database will be automatically updated. The following are the steps you need to take to make this work.

In the description below, we assume that you've installed Mailman in the default location, i.e. '/usr/local/mailman'. If that's not the case, adjust the instructions according to your use of **configure**'s **--prefix** and **--with-var-prefix** options.

Note: If you are using virtual domains and you want Mailman to honor your virtual domains, read the 6.1 section below first!

- Add this to the bottom of the '*$prefix*/Mailman/mm_cfg.py' file:

```
MTA = 'Postfix'
```

  The MTA variable names a module in the 'Mailman/MTA' directory which contains the mail server-specific functions to be executed when a list is created or removed.

- Look at the 'Defaults.py' file for the variables *POSTFIX_ALIAS_CMD* and *POSTFIX_MAP_CMD* command. Make sure these point to your **postalias** and **postmap** programs respectively. Remember that if you need to make changes, do it in 'mm_cfg.py'.

- Run the **bin/genaliases** script to initialize your 'aliases' file.

```
% cd /usr/local/mailman
% bin/genaliases
```

  Make sure that the owner of the 'data/aliases' and 'data/aliases.db' file is `mailman`, that the group owner for those files is `mailman`, or whatever user and group you used in the configure command, and that both files are group writable:

```
% su
% chown mailman:mailman data/aliases*
% chmod g+w data/aliases*
```

- Hack your Postfix's 'main.cf' file to include the following path in your *alias_maps* variable:

```
/usr/local/mailman/data/aliases
```

  Note that there should be no trailing `.db`. Do not include this in your *alias_database* variable. This is because you do not want Postfix's **newaliases** command to modify Mailman's 'aliases.db' file, but you do want Postfix to consult 'aliases.db' when looking for local addresses.

  You probably want to use a `hash:` style database for this entry. Here's an example:

```
alias_maps = hash:/etc/postfix/aliases,
    hash:/usr/local/mailman/data/aliases
```

- When you configure Mailman, use the **--with-mail-gid=mailman** switch; this will be the default if you configured Mailman after adding the `mailman` owner. Because the owner of the 'aliases.db' file is `mailman`, Postfix will execute Mailman's wrapper program as uid and gid `mailman`.

That's it! One caveat: when you add or remove a list, the 'aliases.db' file will updated, but it will not automatically run **postfix reload**. This is because you need to be root to run this and suid-root scripts are not secure. The only effect of this is that it will take about a minute for Postfix to notice the change to the 'aliases.db' file and update its tables.

Virtual domains

Note: This section describes how to integrate Mailman with Postfix for automatic generation of Postfix virtual_alias_-maps for Mailman list addresses. Mailman's support of virtual domains is limited in that list names must be globally unique within a single Mailman instance, i.e., two lists may not have the same name even if they are in different domains.

Postfix 2.0 supports "virtual alias domains", essentially what used to be called "Postfix-style virtual domains" in earlier Postfix versions. To make virtual alias domains work with Mailman, you need to do some setup in both Postfix and Mailman. Mailman will write all virtual alias mappings to a file called, by default, '/usr/local/mailman/data/virtual-mailman'. It will also use **postmap** to create the **virtual-mailman.db** file that Postfix will actually use.

First, you need to set up the Postfix virtual alias domains as described in the Postfix documentation (see Postfix's `virtual(5)` manpage). Note that it's your responsibility to include the `virtual-alias.domain anything` line as described manpage (in recent Postfix this is not required if the domain is included in virtual_alias_domains in main.cf); Mailman will not include this line in 'virtual-mailman'. You are highly encouraged to make sure your virtual alias domains are working properly before integrating with Mailman.

Next, add a path to Postfix's *virtual_alias_maps* variable, pointing to the virtual-mailman file, e.g.:

```
virtual_alias_maps = <your normal virtual alias files>,
    hash:/usr/local/mailman/data/virtual-mailman
```

assuming you've installed Mailman in the default location. If you're using an older version of Postfix which doesn't have the *virtual_alias_maps* variable, use the *virtual_maps* variable instead.

Next, in your 'mm_cfg.py' file, you will want to set the variable *POSTFIX_STYLE_VIRTUAL_DOMAINS* to the list of virtual domains that Mailman should update. This may not be all of the virtual alias domains that your Postfix installation supports! The values in this list will be matched against the *host_name* attribute of mailing lists objects, and must be an exact match.

Here's an example. Say that Postfix is configured to handle the virtual domains `dom1.ain`, `dom2.ain`, and `dom3.ain`, and further that in your 'main.cf' file you've got the following settings:

```
myhostname = mail.dom1.ain
mydomain = dom1.ain
mydestination = $myhostname, localhost.$mydomain
virtual_alias_maps =
    hash:/some/path/to/virtual-dom1,
    hash:/some/path/to/virtual-dom2,
    hash:/some/path/to/virtual-dom2
```

If in your 'virtual-dom1' file, you've got the following lines:

```
dom1.ain  IGNORE
@dom1.ain @mail.dom1.ain
```

this tells Postfix to deliver anything addressed to `dom1.ain` to the same mailbox at `mail.dom1.com`, its default destination.

In this case you would not include `dom1.ain` in *POSTFIX_STYLE_VIRTUAL_DOMAINS* because otherwise Mailman will write entries for mailing lists in the dom1.ain domain as

```
mylist@dom1.ain         mylist
mylist-request@dom1.ain mylist-request
# and so on...
```

The more specific entries trump your more general entries, thus breaking the delivery of any `dom1.ain` mailing list.

However, you would include `dom2.ain` and `dom3.ain` in 'mm_cfg.py':

```
POSTFIX_STYLE_VIRTUAL_DOMAINS = ['dom2.ain', 'dom3.ain']
```

Now, any list that Mailman creates in either of those two domains, will have the correct entries written to '/usr/local/mailman/data/virtual-mailman'.

As above with the 'data/aliases*' files, you want to make sure that both 'data/virtual-mailman' and 'data/virtual-mailman.db' are user and group owned by `mailman`.

An alternative approach

Fil fil@rezo.net has an alternative approach based on virtual maps and regular expressions, as described at:

- (French) http://listes.rezo.net/comment.php
- (English) http://listes.rezo.net/how.php

This is a good (and simpler) alternative if you don't mind exposing an additional hostname in the domain part of the addresses people will use to contact your list. I.e. if people should use `mylist@lists.dom.ain` instead of `mylist@dom.ain`.

6.2 Using the Exim mail server

Note: This section is derived from Nigel Metheringham's "HOWTO - Using Exim and Mailman together", which covers Mailman 2.0.x and Exim 3. It has been updated to cover Mailman 2.1 and Exim 4. The updated document is here: http://www.exim.org/howto/mailman21.html and is recommended over the information in the subsections below if you are using Exim 4.

There is no Mailman configuration needed other than the standard options detailed in the Mailman install documentation. The Exim configuration is transparent to Mailman. The user and group settings for Mailman must match those in the config fragments given below.

Exim configuration

The Exim configuration is built so that a list created within Mailman automatically appears to Exim without the need for defining any additional aliases.

The drawback of this configuration is that it will work poorly on systems supporting lists in several different mail domains. While Mailman handles virtual domains, it does not yet support having two distinct lists with the same name in different virtual domains, using the same Mailman installation. This will eventually change. (But see below for a variation on this scheme that should accommodate virtual domains better.)

The configuration file excerpts below are for use in an already functional Exim configuration, which accepts mail for the domain in which the list resides. If this domain is separate from the others handled by your Exim configuration, then you'll need to:

- add the list domain, "my.list.domain" to *local_domains*
- add a "domains=my.list.domain" option to the director (router) for the list
- (optional) exclude that domain from your other directors (routers)

Note: The instructions in this document should work with either Exim 3 or Exim 4. In Exim 3, you must have a *local_domains* configuration setting; in Exim 4, you most likely have a *local_domains* domainlist. If you don't, you probably know what you're doing and can adjust accordingly. Similarly, in Exim 4 the concept of "directors" has disappeared – there are only routers now. So if you're using Exim 4, whenever this document says "director", read "router".

Whether you are using Exim 3 or Exim 4, you will need to add some macros to the main section of your Exim config file. You will also need to define one new transport. With Exim 3, you'll need to add a new director; with Exim 4, a new router plays the same role.

Finally, the configuration supplied here should allow co-habiting Mailman 2.0 and 2.1 installations, with the proviso that you'll probably want to use `mm21` in place of `mailman` – e.g., *MM21_HOME*, *mm21_transport*, etc.

Main configuration settings

First, you need to add some macros to the top of your Exim config file. These just make the director (router) and transport below a bit cleaner. Obviously, you'll need to edit these based on how you configured and installed Mailman.

```
# Home dir for your Mailman installation -- aka Mailman's prefix
# directory.
MAILMAN_HOME=/usr/local/mailman
MAILMAN_WRAP=MAILMAN_HOME/mail/mailman

# User and group for Mailman, should match your --with-mail-gid
# switch to Mailman's configure script.
MAILMAN_USER=mailman
MAILMAN_GROUP=mailman
```

Transport for Exim 3

Add this to the transports section of your Exim config file, i.e. somewhere between the first and second "end" line:

```
mailman_transport:
  driver = pipe
  command = MAILMAN_WRAP \
            '${if def:local_part_suffix \
                  {${sg{$local_part_suffix}{-(\\w+)(\\+.*)?}{\$1}}} \
                  {post}}' \
            $local_part
  current_directory = MAILMAN_HOME
  home_directory = MAILMAN_HOME
  user = MAILMAN_USER
  group = MAILMAN_GROUP
```

Director for Exim 3

If you're using Exim 3, you'll need to add the following director to your config file (directors go between the second and third "end" lines). Also, don't forget that order matters – e.g. you can make Mailman lists take precedence over system aliases by putting this director in front of your aliasfile director, or vice-versa.

```
# Handle all addresses related to a list 'foo': the posting address.
# Automatically detects list existence by looking
# for lists/$local_part/config.pck under MAILMAN_HOME.
mailman_director:
  driver = smartuser
  require_files = MAILMAN_HOME/lists/$local_part/config.pck
  suffix_optional
  suffix = -bounces : -bounces+* : \
           -confirm+* : -join : -leave : \
           -owner : -request : -admin
  transport = mailman_transport
```

Router for Exim 4

In Exim 4, there's no such thing as directors – you need to add a new router instead. Also, the canonical order of the configuration file was changed so routers come before transports, so the router for Exim 4 comes first here. Put this router somewhere after the "begin routers" line of your config file, and remember that order matters.

```
mailman_router:
  driver = accept
  require_files = MAILMAN_HOME/lists/$local_part/config.pck
  local_part_suffix_optional
  local_part_suffix = -admin : -bounces : -bounces+* : \
                      -confirm : -confirm+* : \
                      -join : -leave : \
                      -owner : -request : \
                      -subscribe : -unsubscribe
  transport = mailman_transport
```

Transports for Exim 4

The transport for Exim 4 is the same as for Exim 3 (see 6.2; just copy the transport given above to somewhere under the "begin transports" line of your Exim config file.

Additional notes

Exim should be configured to allow reasonable volume – e.g. don't set *max_recipients* down to a silly value – and with normal degrees of security – specifically, be sure to allow relaying from 127.0.0.1, but pretty much nothing else. Parallel deliveries and other tweaks can also be used if you like; experiment with your setup to see what works. Delay warning messages should be switched off or configured to only happen for non-list mail, unless you like receiving tons of mail when some random host is down.

Problems

- Mailman will send as many `MAIL FROM`/`RCPT TO` as it needs. It may result in more than 10 or 100 messages sent in one connection, which will exceed the default value of Exim's *smtp_accept_queue_per_connection* value. This is bad because it will cause Exim to switch into queue mode and severely delay delivery of your list messages. The way to fix this is to set Mailman's *SMTP_MAX_SESSIONS_PER_CONNECTION* (in '*$prefix*/Mailman/mm_cfg.py') to a smaller value than Exim's *smtp_accept_queue_per_connection*.
- Mailman should ignore Exim delay warning messages, even though Exim should never send this to list messages. Mailman 2.1's general bounce detection and VERP support should greatly improve the bounce detector's hit rates.
- List existence is determined by the existence of a 'config.pck' file for a list. If you delete lists by foul means, be aware of this.
- If you are getting Exim or Mailman complaining about user ids when you send mail to a list, check that the *MAILMAN_USER* and *MAILMAN_GROUP* match those of Mailman itself (i.e. what were used in the **configure** script). Also make sure you do not have aliases in the main alias file for the list.

Receiver Verification

Exim's receiver verification feature is very useful – it lets Exim reject unrouteable addresses at SMTP time. However, this is most useful for externally-originating mail that is addressed to mail in one of your local domains. For Mailman list traffic, mail originates on your server, and is addressed to random external domains that are not under your control. Furthermore, each message is addressed to many recipients – up to 500 if you use Mailman's default configuration and don't tweak *SMTP_MAX_RCPTS*.

Doing receiver verification on Mailman list traffic is a recipe for trouble. In particular, Exim will attempt to route every recipient addresses in outgoing Mailman list posts. Even though this requires nothing more than a few DNS lookups for each address, it can still introduce significant delays. Therefore, you should disable recipient verification for Mailman traffic.

Under Exim 3, put this in your main configuration section:

```
receiver_verify_hosts = !127.0.0.1
```

Under Exim 4, this is probably already taken care of for you by the default recipient verification ACL statement (in the `RCPT TO` ACL):

```
accept  domains       = +local_domains
        endpass
        message       = unknown user
        verify        = recipient
```

which only does recipient verification on addresses in your domain. (That's not exactly the same as doing recipient verification only on messages coming from non-127.0.0.1 hosts, but it should do the trick for Mailman.)

SMTP Callback

Exim's SMTP callback feature is an even more powerful way to detect bogus sender addresses than normal sender verification. Unfortunately, lots of servers send bounce messages with a bogus address in the header, and there are plenty that send bounces with bogus envelope senders (even though they're supposed to just use an empty envelope sender for bounces).

In order to ensure that Mailman can disable/remove bouncing addresses, you generally want to receive bounces for Mailman lists, even if those bounces are themselves not bounceable. Thus, you might want to disable SMTP callback on bounce messages.

With Exim 4, you can accomplish this using something like the following in your `RCPT TO` ACL:

```
# Accept bounces to lists even if callbacks or other checks would fail
warn     message      = X-WhitelistedRCPT-nohdrfromcallback: Yes
         condition    = \
         ${if and {{match{$local_part}{(.*)-bounces\+.*}} \
                   {exists {MAILMAN_HOME/lists/$1/config.pck}}} \
              {yes}{no}}

accept   condition    = \
         ${if and {{match{$local_part}{(.*)-bounces\+.*}} \
                   {exists {MAILMAN_HOME/lists/$1/config.pck}}} \
              {yes}{no}}

# Now, check sender address with SMTP callback.
deny   !verify = sender/callout=90s
```

If you also do SMTP callbacks on header addresses, you'll want something like this in your `DATA` ACL:

```
deny   !condition = $header_X-WhitelistedRCPT-nohdrfromcallback:
       !verify = header_sender/callout=90s
```

Doing VERP with Exim and Mailman

VERP will send one email, with a separate envelope sender (return path), for each of your subscribers – read the information in '*$prefix*/Mailman/Defaults.py' for the options that start with VERP. In a nutshell, all you need to do to enable VERP with Exim is to add these lines to '*$prefix*/Mailman/mm_cfg.py':

```
VERP_PASSWORD_REMINDERS = Yes
VERP_PERSONALIZED_DELIVERIES = Yes
VERP_DELIVERY_INTERVAL = Yes
VERP_CONFIRMATIONS = Yes
```

(The director (router) above is smart enough to deal with VERP bounces.)

Virtual Domains

One approach to handling virtual domains is to use a separate Mailman installation for each virtual domain. Currently, this is the only way to have lists with the same name in different virtual domains handled by the same machine.

In this case, the *MAILMAN_HOME* and *MAILMAN_WRAP* macros are useless – you can remove them. Change your director (router) to something like this:

```
require_files = /virtual/${domain}/mailman/lists/${lc:$local_part}/config.pck
```

and change your transport like this:

```
command = /virtual/${domain}/mailman/mail/mailman \
          ${if def:local_part_suffix \
               {${sg{$local_part_suffix}{-(\\w+)(\\+.*)?}{\$1}}}
               {post}} \
            $local_part
current_directory = /virtual/${domain}/mailman
home_directory = /virtual/${domain}/mailman
```

List Verification

This is how a set of address tests for the Exim lists look on a working system. The list in question is quixote-users@mems-exchange.org, and these commands were run on the `mems-exchange.org` mail server ("% " indicates the Unix shell prompt):

```
% exim -bt quixote-users
quixote-users@mems-exchange.org
  router = mailman_main_router, transport = mailman_transport

% exim -bt quixote-users-request
quixote-users-request@mems-exchange.org
  router = mailman_router, transport = mailman_transport

% exim -bt quixote-users-bounces
quixote-users-bounces@mems-exchange.org
  router = mailman_router, transport = mailman_transport

% exim -bt quixote-users-bounces+luser=example.com
quixote-users-bounces+luser=example.com@mems-exchange.org
  router = mailman_router, transport = mailman_transport
```

If your **exim -bt** output looks something like this, that's a start: at least it means Exim will pass the right messages to the right Mailman commands. It by no means guarantees that your Exim/Mailman installation is functioning perfectly, though!

Document History

Originally written by Nigel Metheringham postmaster@exim.org. Updated by Marc Merlin marc_soft@merlins.org for Mailman 2.1, Exim 4. Overhauled/reformatted/clarified/simplified by Greg Ward gward@python.net.

6.3 Using the Sendmail mail server

Warning: You may be tempted to set the *DELIVERY_MODULE* configuration variable in 'mm_cfg.py' to `'Sendmail'` when using the Sendmail mail server. **Don't**. The 'Sendmail.py' module is misnamed – it's really a command line based message handoff scheme as opposed to the SMTP scheme used in 'SMTPDirect.py' (the default). 'Sendmail.py' has known security holes and is provided as a proof-of-concept only[a]. If you are having problems using 'SMTPDirect.py' fix those instead of using 'Sendmail.py', or you may open your system up to security exploits.

[a] In fact, in later versions of Mailman, this module is explicitly sabotaged. You have to know what you're doing in order to re-enable it.

Sendmail "smrsh" compatibility

Many newer versions of Sendmail come with a restricted execution utility called "smrsh", which limits the executables that Sendmail will allow to be used as mail programs. You need to explicitly allow Mailman's wrapper program to be used with smrsh or Mailman will not work. If mail is not getting delivered to Mailman's wrapper program and you're getting an "operating system error" in your mail syslog, this could be your problem.

One good way of enabling this is:

- Find out where your Sendmail executes its smrsh wrapper

```
% grep smrsh /etc/mail/sendmail.cf
```

- Figure out where smrsh expects symlinks for allowable mail programs. At the very beginning of the following output you will see a full path to some directory, e.g. '/var/adm/sm.bin' or similar:

```
% strings $path_to_smrsh | less
```

- cd into '/var/adm/sm.bin', or where ever it happens to reside on your system – alternatives include '/etc/smrsh', '/var/smrsh' and '/usr/local/smrsh'.

```
% cd /var/adm/sm.bin
```

- Create a symbolic link to Mailman's wrapper program:

```
% ln -s /usr/local/mailman/mail/mailman mailman
```

Integrating Sendmail and Mailman

David Champion has contributed a recipe for more closely integrating Sendmail and Mailman, such that Sendmail will automatically recognize and deliver to new mailing lists as they are created, without having to manually edit alias tables.

In the 'contrib' directory of Mailman's source distribution, you will find four files:

- 'mm-handler.readme' - an explanation of how to set everything up
- 'mm-handler' - the mail delivery agent (MDA)
- 'mailman.mc' - a toy configuration file sample
- 'virtusertable' - a sample for RFC 2142 address exceptions

Performance notes

One of the surest performance killers for Sendmail users is when Sendmail is configured to synchronously verify the recipient's host via DNS. If it does this for messages posted to it from Mailman, you will get horrible performance. Since Mailman usually connects via `localhost` (i.e. 127.0.0.1) to the SMTP port of Sendmail, you should be sure to configure Sendmail to **not** do DNS verification synchronously for localhost connections.

6.4 Using the Qmail mail server

There are some issues that users of the qmail mail transport agent have encountered. None of the core maintainers use qmail, so all of this information has been contributed by the Mailman user community, especially Martin Preishuber and Christian Tismer, with notes by Balazs Nagy (BN) and Norbert Bollow (NB).

- You might need to set the mail-gid user to either `qmail`, `mailman`, or `nofiles` by using the **--with-mail-gid configure** option.

  BN: it highly depends on your mail storing policy. For example if you use the simple '~alias/.qmail-*' files, you can use **'id -g alias'**. But if you use '/var/qmail/users', the specified mail gid can be used.

  If you are going to be directing virtual domains directly to the `mailman` user (using "virtualdomains" on a list-only domain, for example), you will have to use **--with-mail-gid**=*gid of mailman user's group*. This is incompatible with having list aliases in '~alias', unless that alias simply forwards to `mailman-listname*`.

- If there is a user `mailman` on your system, the alias `mailman-owner` will work only in '~mailman'. You have to do a **touch .qmail-owner** in '~mailman' directory to create this alias.

  NB: An alternative, IMHO better solution is to **chown root ~mailman**, that will stop qmail from considering `mailman` to be a user to whom mail can be delivered. (See "man 8 qmail-getpw".)

- In a related issue, if you have any users with the same name as one of your mailing lists, you will have problems if list names contain '-' in them. Putting '.qmail' redirections into the user's home directory doesn't work because the Mailman wrappers will not get spawned with the proper GID. The solution is to put the following lines in the '/var/qmail/users/assign' file:

```
+zope-:alias:112:11:/var/qmail/alias:-:zope-:
.
```

where in this case the listname is e.g. `zope-users`.

NB: Alternatively, you could host the lists on a virtual domain, and use the '/var/qmail/control/virtualdomains' file to put the `mailman` user in charge of this virtual domain.

- *BN:*If inbound messages are delivered by another user than `mailman`, it's necessary to allow it to access '˜mailman'. Be sure that '˜mailman' has group writing access and setgid bit is set. Then put the delivering user to `mailman` group, and you can deny access to '˜mailman' to others. Be sure that you can do the same with the WWW service.

  By the way the best thing is to make a virtual mail server to handle all of the mail. *NB:* E.g. make an additional "A" DNS record for the virtual mailserver pointing to your IP address, add the line `lists.kva.hu:mailman` to '/var/qmail/control/virtualdomains' and a `lists.kva.hu` line to '/var/qmail/control/rcpthosts' file. Don't forget to HUP the qmail-send after modifying "virtualdomains". Then every mail to lists.kva.hu will arrive to mail.kva.hu's mailman user.

  Then make your aliases:

  ```
  .qmail              => mailman@...'s letters
  .qmail-owner        => mailman-owner's letters
  ```

  For list aliases, you can either create them manually:

  ```
  .qmail-list         => posts to the 'list' list
  .qmail-list-admin   => posts to the 'list's owner
  .qmail-list-request => requests to 'list'
  etc
  ```

  or for automatic list alias handling (when using the lists.kva.hu virtual as above), see 'contrib/qmail-to-mailman.py' in the Mailman source distribution. Modify the '˜mailman/.qmail-default' to include:

  ```
  |preline /path/to/python /path/to/qmail-to-mailman.py
  ```

  and new lists will automatically be picked up.

- You have to make sure that the localhost can relay. If you start qmail via inetd and tcpenv, you need some line the following in your '/etc/hosts.allow' file:

  ```
  tcp-env: 127. 10.205.200. : setenv RELAYCLIENT
  ```

  where 10.205.200. is your IP address block. If you use tcpserver, then you need something like the following in your '/etc/tcp.smtp' file:

  ```
  10.205.200.:allow,RELAYCLIENT=""
  127.:allow,RELAYCLIENT=""
  ```

- *BN:* Bigger '/var/qmail/control/concurrencyremote' values work better sending outbound messages, within reason. Unless you know your system can handle it (many if not most cannot) this should not be set to a value greater than 120.
- More information about setting up qmail and relaying can be found in the qmail documentation.

BN: Last but not least, here's a little script to generate aliases to your lists (if for some reason you can/will not have them automatically picked up using 'contrib/qmail-to-mailman.py'):

This script is for the Mailman 2.0 series:

```
#!/bin/sh
if [ $# = 1 ]; then
    i=$1
    echo Making links to $i in the current directory...
    echo "|preline /home/mailman/mail/mailman post $i" > .qmail-$i
    echo "|preline /home/mailman/mail/mailman mailowner $i" > .qmail-$i-admin
    echo "|preline /home/mailman/mail/mailman mailowner $i" > .qmail-$i-owner
    echo "|preline /home/mailman/mail/mailman mailowner $i" > .qmail-owner-$i
    echo "|preline /home/mailman/mail/mailman mailcmd $i" > .qmail-$i-request
fi
```

Note: This is for a new Mailman 2.1 installation. Users upgrading from Mailman 2.0 would most likely change '/usr/local/mailman' to '/home/mailman'. If in doubt, refer to the **--prefix** option passed to **configure** during compile time.

```
#!/bin/sh
if [ $# = 1 ]; then
    i=$1
    echo Making links to $i in the current directory...
    echo "|preline /usr/local/mailman/mail/mailman post $i" > .qmail-$i
    echo "|preline /usr/local/mailman/mail/mailman admin $i" > .qmail-$i-admin
    echo "|preline /usr/local/mailman/mail/mailman bounces $i" > .qmail-$i-bounces
    # The following line is for VERP
    # echo "|preline /usr/local/mailman/mail/mailman bounces $i" > .qmail-$i-bounces-default
    echo "|preline /usr/local/mailman/mail/mailman confirm $i" > .qmail-$i-confirm
    echo "|preline /usr/local/mailman/mail/mailman join $i" > .qmail-$i-join
    echo "|preline /usr/local/mailman/mail/mailman leave $i" > .qmail-$i-leave
    echo "|preline /usr/local/mailman/mail/mailman owner $i" > .qmail-$i-owner
    echo "|preline /usr/local/mailman/mail/mailman request $i" > .qmail-$i-request
    echo "|preline /usr/local/mailman/mail/mailman subscribe $i" > .qmail-$i-subscribe
    echo "|preline /usr/local/mailman/mail/mailman unsubscribe $i" > .qmail-$i-unsubscribe
fi
```

Information on VERP

You will note in the alias generating script for 2.1 above, there is a line for VERP that has been commented out. If you are interested in VERP there are two options. The first option is to allow Mailman to do the VERP formatting. To activate this, uncomment that line and add the following lines to your 'mm_cfg.py' file:

```
VERP_FORMAT = '%(bounces)s-+%(mailbox)s=%(host)s'
VERP_REGEXP = r'^(?P<bounces>.*?)-\+(?P<mailbox>[^=]+)=(?P<host>[^@]+)@.*$'
```

The second option is a patch on SourceForge located at:

http://sourceforge.net/tracker/?func=detail&atid=300103&aid=645513&group_id=103

This patch currently needs more testing and might best be suitable for developers or people well familiar with qmail. Having said that, this patch is the more qmail-friendly approach resulting in large performance gains.

Virtual mail server

As mentioned in the 6.4 section for a virtual mail server, a patch under testing is located at:

http://sf.net/tracker/index.php?func=detail&aid=621257&group_id=103&atid=300103

Again, this patch is for people familiar with their qmail installation.

More information

You might be interested in some information on modifying footers that Norbert Bollow has written about Mailman and qmail, available here:

http://mailman.cis.to/qmail-verh/

7 Review your site defaults

Mailman has a large number of site-wide configuration options which you should now review and change according to your needs. Some of the options control how Mailman interacts with your environment, and other options select defaults for newly created lists[4]. There are system tuning parameters and integration options.

The full set of site-wide defaults lives in the '*$prefix*/Mailman/Defaults.py' file, however you should **never** modify this file! Instead, change the 'mm_cfg.py' file in that same directory. You only need to add values to 'mm_cfg.py' that are different than the defaults in 'Defaults.py', and future Mailman upgrades are guaranteed never to touch your 'mm_cfg.py' file.

The 'Defaults.py' file is documented extensively, so the options are not described here. The 'Defaults.py' and 'mm_cfg.py' are both Python files so valid Python syntax must be maintained or your Mailman installation will break.

Note: Do **not** change the *HOME_DIR* or *MAILMAN_DIR* variables. These are set automatically by the **configure** script, and you will break your Mailman installation by if you change these.

You should make any changes to 'mm_cfg.py' using the account you installed Mailman under in the 3 section.

8 Create a site-wide mailing list

After you have completed the integration of Mailman and your mail server, you need to create a "site-wide" mailing list. This is the one that password reminders will appear to come from, and it is required for proper Mailman operation. Usually this should be a list called `mailman`, but if you need to change this, be sure to change the *MAILMAN_SITE_LIST* variable in 'mm_cfg.py'. You can create the site list with this command, following the prompts:

```
% bin/newlist mailman
```

[4] In general, changing the list defaults described in this section will not affect any already created lists. To make changes after a list has been created, use the web interface or the command line scripts, such as **bin/withlist** and **bin/config_list**.

Now configure your site list. There is a convenient template for a generic site list in the installation directory, under 'data/sitelist.cfg' which can help you with this. You should review the configuration options in the template, but note that any options not named in the 'sitelist.cfg' file won't be changed.

The template can be applied to your site list by running:

```
% bin/config_list -i data/sitelist.cfg mailman
```

After applying the 'sitelist.cfg' options, be sure you review the site list's configuration via the admin pages.

You should also subscribe yourself to the site list.

9 Set up cron

Several Mailman features occur on a regular schedule, so you must set up **cron** to run the right programs at the right time[5].

If your version of crontab supports the **-u** option, you must be root to do this next step. Add '*$prefix*/cron/crontab.in' as a crontab entry by executing these commands:

```
% cd $prefix/cron
% crontab -u mailman crontab.in
```

If you used the **--with-username** option, use that user name instead of `mailman` for the **-u** argument value. If your crontab does not support the **-u** option, try these commands:

```
% cd $prefix/cron
% su - mailman
% crontab crontab.in
```

Warning: If you accepted the defaults for the **--with-username** option and for the name of the site list, and one of the cron jobs ever encounters an error, the cron daemon will mail the error output to the 'mailman' user and it will most likely be delivered to the 'mailman' site list and possibly not be accepted. For this reason it is a good idea to insert

```
MAILTO=user@example.com
```

or

```
MAILTO=mailman-owner
```

at the beginning of crontab.in before installing it to cause this output to be mailed to a real user or to the owner of the site list or to configure the site list (see section 8) to accept this mail.

[5]Note that if you're upgrading from a previous version of Mailman, you'll want to install the new crontab, but be careful if you're running multiple Mailman installations on your site! Changing the crontab could mess with other parallel Mailman installations.

10 Start the Mailman qrunner

Mailman depends on a process called the "qrunner" to delivery all email messages it sees. You must start the qrunner by executing the following command from the *$prefix* directory:

```
% bin/mailmanctl start
```

You probably want to start Mailman every time you reboot your system. Exactly how to do this depends on your operating system. If your OS supports the **chkconfig** command (e.g. RedHat and Mandrake Linuxes) you can do the following (as root, from the Mailman install directory):

```
% cp scripts/mailman /etc/init.d/mailman
% chkconfig --add mailman
```

Note that '/etc/init.d' may be '/etc/rc.d/init.d' on some systems.

On Gentoo Linux, you can do the following:

```
% cp scripts/mailman /etc/init.d/mailman
% rc-update add mailman default
```

On Debian, you probably want to use:

```
% update-rc.d mailman defaults
```

For UNIXes that don't support **chkconfig**, you might try the following set of commands:

```
% cp scripts/mailman /etc/init.d/mailman
% cp misc/mailman /etc/init.d
% cd /etc/rc.d/rc0.d
% ln -s ../init.d/mailman K12mailman
% cd ../rc1.d
% ln -s ../init.d/mailman K12mailman
% cd ../rc2.d
% ln -s ../init.d/mailman S98mailman
% cd ../rc3.d
% ln -s ../init.d/mailman S98mailman
% cd ../rc4.d
% ln -s ../init.d/mailman S98mailman
% cd ../rc5.d
% ln -s ../init.d/mailman S98mailman
% cd ../rc6.d
% ln -s ../init.d/mailman K12mailman
```

11 Check the hostname settings

You should check the values for *DEFAULT_EMAIL_HOST* and *DEFAULT_URL_HOST* in 'Defaults.py'. Make any necessary changes in the 'mm_cfg.py' file, **not** in the 'Defaults.py' file. If you change either of these two values, you'll want to add the following afterwards in the 'mm_cfg.py' file:

```
add_virtualhost(DEFAULT_URL_HOST, DEFAULT_EMAIL_HOST)
```

You will want to run the **bin/fix_url.py** to change the domain of any existing lists.

12 Create the site password

There are two site-wide passwords that you can create from the command line, using the **bin/mmsitepass** script. The first is the "site password" which can be used anywhere a password is required in the system. The site password will get you into the administration page for any list, and it can be used to log in as any user. Think `root` for a Unix system, so pick this password wisely!

The second password is a site-wide "list creator" password. You can use this to delegate the ability to create new mailing lists without providing all the privileges of the site password. Of course, the owner of the site password can also create new mailing lists, but the list creator password is limited to just that special role.

To set the site password, use this command:

```
% $prefix/bin/mmsitepass <your-site-password>
```

To set the list creator password, use this command:

```
% $prefix/bin/mmsitepass -c <list-creator-password>
```

It is okay not to set a list creator password, but you probably do want a site password.

13 Create your first mailing list

For more detailed information about using Mailman, including creating and configuring mailing lists, see the Mailman List Adminstration Manual. These instructions provide a quick guide to creating your first mailing list via the web interface:

- Start by visiting the url `http://my.dom.ain/mailman/create`.
- Fill out the form as described in the on-screen instructions, and in the "List creator's password" field, type the password you entered in section 7. Type your own email address for the "Initial list owner address", and select "Yes" to notify the list administrator.
- Click on the "Create List" button.
- Check your email for a message from Mailman informing you that your new mailing list was created.
- Now visit the list's administration page, either by following the link on the confirmation web page or clicking on the link from the email Mailman just sent you. Typically the url will be something like `http://my.dom.ain/mailman/admin/mylist`.

- Type in the list's password and click on "Let me in..."
- Click on "Membership Management" and then on "Mass Subscription".
- Enter your email address in the big text field, and click on "Submit Your Changes".
- Now go to your email and send a message to `mylist@my.dom.ain`. Within a minute or two you should see your message reflected back to you via Mailman.

Congratulations! You've just set up and tested your first Mailman mailing list. If you had any problems along the way, please see the 14 section.

14 Troubleshooting

If you encounter problems with running Mailman, first check the question and answer section below. If your problem is not covered there, check the online help, including the FAQ and the community FAQ wiki.

Also check for errors in your syslog files, your mail and web server log files and in Mailman's '*$prefix*/logs/error' file. If you're still having problems, you should send a message to the mailman-users@python.org mailing list[6]; see http://mail.python.org/mailman/listinfo/mailman-users for more information.

Be sure to including information on your operating system, which version of Python you're using, and which version of Mailman you're installing.

Here is a list of some common questions and answers:

- **Problem:** All Mailman web pages give a 404 File not found error.

  Solution: Your web server has not been set up properly for handling Mailman's CGI programs. Make sure you have:

  1. configured the web server to give permissions to '*$prefix*/cgi-bin'
  2. restarted the web server properly.

  Consult your web server's documentation for instructions on how to do check these issues.

- **Problem:** All Mailman web pages give an "Internal Server Error".

  Solution: The likely problem is that you are using the wrong user or group for the CGI scripts. Check your web server's log files. If you see a line like

  ```
  Attempt to exec script with invalid gid 51, expected 99
  ```

  you will need to reinstall Mailman, specifying the proper CGI group id, as described in the 3 section.

- **Problem:** I send mail to the list, and get back mail saying the list is not found!

  Solution: You probably didn't add the necessary aliases to the system alias database, or you didn't properly integrate Mailman with your mail server. Perhaps you didn't update the alias database, or your system requires you to run **newaliases** explicitly. Refer to your server specific instructions in the 6 section.

- **Problem:** I send mail to the list, and get back mail saying, "unknown mailer error".

  Solution: The likely problem is that you are using the wrong user or group id for the mail wrappers. Check your mail server's log files; if you see a line like

[6] You must subscribe to this mailing list in order to post to it, but the mailing list's archives are publicly visible.

```
Attempt to exec script with invalid gid 51, expected 99
```

you will need to reinstall Mailman, specifying the proper mail group id as described in the 3 section.

- **Problem:** I use Postfix as my mail server and the mail wrapper programs are logging complaints about the wrong GID.

  Solution: Make sure the '*$prefix*/data/aliases.db' file is user owned by `mailman` (or whatever user name you used in the **configure** command). If this file is not user owned by `mailman`, Postfix will not run the mail programs as the correct user.

- **Problem:** I use Sendmail as my mail server, and when I send mail to the list, I get back mail saying, "sh: mailman not available for sendmail programs".

  Solution: Your system uses the Sendmail restricted shell (smrsh). You need to configure smrsh by creating a symbolic link from the mail wrapper ('*$prefix*/mail/mailman') to the directory identifying executables allowed to run under smrsh.

  Some common names for this directory are '/var/admin/sm.bin', '/usr/admin/sm.bin' or '/etc/smrsh'.

  Note that on Debian Linux, the system makes '/usr/lib/sm.bin', which is wrong, you will need to create the directory '/usr/admin/sm.bin' and add the link there. Note further any aliases **newaliases** spits out will need to be adjusted to point to the secure link to the wrapper.

- **Problem:** I messed up when I called **configure**. How do I clean things up and re-install?

  Solution:

```
% make clean
% ./configure --with-the-right-options
% make install
```

15 Platform and operating system notes

Generally, Mailman runs on any POSIX-based system, such as Solaris, the various BSD variants, Linux systems, MacOSX, and other generic UNIX systems. It doesn't run on Windows. For the most part, the generic instructions given in this document should be sufficient to get Mailman working on any supported platform. Some operating systems have additional recommended installation or configuration instructions.

15.1 GNU/Linux issues

Linux seems to be the most popular platform for running Mailman. Here are some hints on getting Mailman to run on Linux:

- If you are getting errors with hard link creations and/or you are using a special secure kernel (securelinux/openwall/grsecurity), see the file 'contrib/README.check_perms_grsecurity' in the Mailman source distribution.

  Note that if you are using Linux Mandrake in secure mode, you are probably concerned by this.

- Apparently Mandrake 9.0 changed the permissions on gcc, so if you build as the `mailman` user, you need to be sure `mailman` is in the `cctools` group.

- If you installed Python from your Linux distribution's package manager (e.g. .rpms for Redhat-derived systems or .deb for Debian), you must install the "development" package of Python, or you may not get everything you need.

  For example, using Python 2.2 on Debian, you will need to install the `python2.2-dev` package. On Redhat, you probably need the `python2-devel` package.

  If you install Python from source, you should be fine.

  One symptom of this problem, although for unknown reasons, is that you might get an error such as this during your install:

  ```
  Traceback (most recent call last):
    File "bin/update", line 44, in ?
      import paths
  ImportError: No module named paths
  make: *** [update] Error 1
  ```

  If this happens, install the Python development package and try **configure** and **make install** again. Or install the latest version of Python from source, available from http://www.python.org.

  This problem can manifest itself in other Linux distributions in different ways, although usually it appears as `ImportErrors`.

15.2 BSD issues

Vivek Khera writes that some BSDs do nightly security scans for setuid file changes. setgid directories also come up on the scan when they change. Also, the setgid bit is not necessary on BSD systems because group ownership is automatically inherited on files created in directories. On other UNIXes, this only happens when the directory has the setgid bit turned on.

To install without turning on the setgid bit on directories, simply pass in the *DIRSETGID* variable to **make**, after you've run **configure**:

```
% make DIRSETGID=: install
```

This disables the **chmod g+s** command on installed directories.

15.3 MacOSX issues

Many people run Mailman on MacOSX. Here are some pointers that have been collected on getting Mailman to run on MacOSX.

- Jaguar (MacOSX 10.2) comes with Python 2.2. While this isn't the very latest stable version of Python, it ought to be sufficient to run Mailman 2.1.
- David B. O'Donnell has a web page describing his configuration of Mailman 2.0.13 and Postfix on MacOSX Server.

  http://www.afp548.com/Articles/mail/python-mailman.html
- Kathleen Webb posted her experiences in getting Mailman running on Jaguar using Sendmail.

  http://mail.python.org/pipermail/mailman-users/2002-October/022944.html

- Panther server (MacOSX 10.3) comes with Mailman; Your operating system should contain documentation that will help you, and Apple has a tech document about a problem you might encounter running Mailman on Mac OS X Server 10.3:

  http://docs.info.apple.com/article.html?artnum=107889

Terry Allen provides the following detailed instructions on running Mailman on the 'client' version of OSX, or in earlier versions of OSX:

Mac OSX 10.3 and onwards has the basics for a successful Mailman installation. Users of earlier versions of Mac OSX contains Sendmail and those users should look at the Sendmail installation section for tips. You should follow the basic installation steps as described earlier in this manual, substituting as appropriate, the steps outlined in this section.

By default, Mac OSX 10.3 'client' version does not have a fully functional version of Postfix. Setting up a working MTA such as Postfix is beyond the scope of this guide and you should refer to http://www.postfix.org for tips on getting Postfix running. An easy way to set Postfix up is to install and run Postfix Enabler, a stand-alone tool for configuring Postfix on Mac OSX, available from http://www.roadstead.com/weblog/Tutorials/PostfixEnabler.html.

Likewise, Mac OSX 'client' version from 10.1 onwards includes a working Apache webserver. This is switched on using the System Preferences control panel under the 'Sharing tab'. A useful tool for configuring the Apache on Mac OSX is Webmin, which can be obtained from http://www.webmin.com.

Webmin can also perform configuration for other system tasks, including Postfix, adding jobs to your crontab, adding user and groups, plus adding startup and shutdown jobs.

In a stock installation of OSX, the requirement for Mailman is to have Python installed. Python is not installed by default, so it is advised that you install the developer's tools package, which may have been provided with your system. It can also be downloaded from the Apple developer site at http://connect.apple.com. Not only is the developer tools package an essential requirement for installing Mailman, but it will come in handy at a later date should you need other tools. The developer's tools are also know by the name XCode tools.

As a minimum, the Python version should be 2.2, but 2.3 is recommended.

If you wish to add a user and group using the command line in OSX instead of via Webmin or another GUI interface, open your terminal application and follow the commands as indicated below - do not type the comments following the '#' since they are just notes:

```
sudo tcsh
niutil -create / /users/mailman
niutil -createprop / /users/mailman name mailman
# Note that xxx is a free user ID number on your system
niutil -createprop / /users/mailman uid xxx
niutil -createprop / /users/mailman home /usr/local/mailman
mkdir -p /usr/local/mailman
niutil -createprop / /users/mailman shell /bin/tcsh
passwd mailman
# To prevent malicious hacking, supply a secure password here
niutil -create / /groups/mailman
niutil -createprop / /groups/mailman name mailman
# Note that xxx is a free group ID number on your system
niutil -createprop / /groups/mailman gid xxx
niutil -createprop / /groups/mailman passwd '*'
niutil -createprop / /groups/mailman users 'mailman'
chown mailman:mailman /usr/local/mailman
cd /usr/local/mailman
chmod a+rx,g+ws .
exit
su mailman
```

For setting up Apache on OSX to handle Mailman, the steps are almost identical and the configuration file on a stock Mac OSX Client version is stored in the nearly standard location of '/etc/httpd/httpd.conf'.

The AFP548.com site has a time-saving automated startup item creator for Mailman, which can be found at http://www.afp548.com/Software/MailmanStartup.tar.gz

To install it, copy it into your '/Library/StartupItems' directory. As the root or superuser, from the terminal, enter the following:

```
gunzip MailmanStartup.tar.gz
tar xvf MailmanStartup.tar
```

It will create the startup item for you so that when you reboot, Mailman will start up.