Exim Practical

Objectives

Part 1 is basic set up and basic tests. You don't need to modify the Exim configuration to do this.

- Install Exim from the generic source distribution
- Find the documentation
- Test standard installation and default configuration
- Inspect and manage the mail queue
- Check relay control
- Process log data

Part 2 involves some simple modification of the runtime configuration.

- Modify the runtime configuration to send undeliverable mail to postmaster
- Add some simple virtual domains

Part 3 sets up your host as a mail relay

- Allow relaying from another host
- Allow relaying to another domain

Part 4 is more advanced things to try for those who have time.

- Demonstrate retry mechanisms
- Configure and test address rewriting
- Add a system filter
- Reconfigure for a large installation

This sign is use in the text to mark an action that you need to take if you want to do the things that are suggested in this practical.

1. Installing Exim and testing the default configuration

You do not need to be *root* to build Exim, but you do need to be *root* to install it. In the sample commands below, the command prompt is shown as **#** for commands that must be run as root, and **$** otherwise.

1.1 Preliminary preparation

☞ If a **sendmail** daemon is running, kill it off. Use ***ps*** with ***grep*** to find it.

```
# ps -xa | grep sendmail
# kill process-id
```

Where *process-id* is the process number of the **sendmail** process.

You should previously have created a user and a group called *exim*. These will be used for running Exim when it does not need to be root. If you have not done this, you must do it before trying to build Exim.

You should also have arranged for your personal (non-root) account to be in the **exim** group so that you can be an administrator for Exim.

☞ Ensure that the **/var/mail** directory has the 'sticky' bit set on it. If you don't understand this, don't worry, just do it:

```
# chmod o+wt /var/mail
```

(This is so that the default Exim configuration will work without having to be changed.)

1.2 Install Exim from the generic distribution

This way of doing it allows you to make your own choices at build time.

☞ Make a directory in which to build Exim, say **/usr/exim**, and give yourself access to it:

```
# mkdir /usr/exim
# chown yourname:yourname /usr/exim
```

You can now fetch and build Exim from your own account (not *root*).

☞ Fetch the source of Exim and the HTML documentation from the ftp site on the workshop noc:

```
$ cd /usr/exim
$ ftp noc.ws.afnog.org
```

Log in as *anonymous*.

```
ftp> cd /pub/t1
ftp> get exim-4.04.tar.gz
ftp> get exim-html-4.00.tar.gz
ftp> bye
```

☞ Unzip and untar the source and the HTML documentation:

```
$ cd /usr/exim
$ gunzip exim-4.04.tar.gz
$ tar -xf exim-4.04.tar
$ gunzip exim-html-4.00.tar.gz
$ tar -xf exim-html-4.00.tar
```

☞ Now we can get ready to build Exim. You have to set up two configuration files. Go into the toplevel source directory:

```
$ cd exim-4.04
```

Copy the file **src/EDITME** to **Local/Makefile** and **exim_monitor/EDITME** to **Local/eximon.conf**. You then have to edit **Local/Makefile**, following the instructions inside it:

```
$ cp src/EDITME Local/Makefile
$ cp exim_monitor/EDITME Local/eximon.conf
$ vi Local/Makefile
```

There are lots of instructions inside the file, but you do not have to make many changes. You can leave almost all of the settings at the defaults, but you will need to set `EXIM_USER` to the user for running Exim. You also need to request 'maildir' support for use later in the workshop. That is:

```
EXIM_USER=exim
SUPPORT_MAILDIR=yes
```

(Do not do this at the workshop.) When you build Exim on your own hosts back home, you may want to change `BIN_DIRECTORY` and `CONFIGURE_FILE` from their default values of **/usr/exim/bin** and `/usr/exim/configure`. For example, these settings match what the FreeBSD port uses:

```
BIN_DIRECTORY=/usr/local/sbin
CONFIGURE_FILE=/usr/local/etc/exim/configure
```

However, for this exercise, we assume that you didn't change the default values.

You don't need to edit **Local/eximon.conf** because the default settings will be OK.

☞ Now you can for it!

```
$ make
```

You should see a lot of output while Exim builds, ending with the line:

```
>>> exim binary built
```

When you see that line, you have successfully built Exim.

☞ You need to be root to install Exim:

```
# cd /usr/exim
# make install
```

You should end up with the Exim binaries in **/usr/exim/bin/** and a default configuration file in **/usr/exim/configure**.

☞ Test that Exim has been installed by running:

```
$ /usr/exim/bin/exim -bV
```

which should tell you Exim's version number.

1.3 Replace Sendmail with Exim

All the MUAs call **/usr/sbin/sendmail** to pass messages to the MTA. We want them to call Exim instead of Sendmail.

☞ On FreeBSD, there is a file called **/etc/mail/mailer.conf** which selects the MTA. To change the MTA, you must edit this file:

```
# vi /etc/mail/mailer.conf
```

Comment the existing lines, and insert these new lines:

```
sendmail        /usr/exim/bin/exim
send-mail       /usr/exim/bin/exim
mailq           /usr/exim/bin/exim -bp
newaliases      /usr/bin/true
```

☞ Now try that basic test again, but this time using the standard path name:

```
/usr/sbin/sendmail -bV
```

You should get the same output as before, which shows that Exim is now being used instead of Sendmail.

(Do not do this at the workshop.) On operating systems that don't have **/etc/mailer.conf**, you can just make **/usr/sbin/sendmail** point directly to the Exim binary:

```
# mv /usr/sbin/sendmail /usr/sbin/sendmail.sendmail
# ln -s /usr/exim/bin/exim /usr/sbin/sendmail
```

If you are doing a real installation on a live system, you might want to work on the configuration and do lots of testing before removing Sendmail and replacing it with Exim. You can test pretty well everything without disturbing the running MTA if you want to.

1.4 Find the documentation

Before moving on, make sure you know where the Exim documentation is, so that you can look things up if you have problems. If you have a web browser running, point it at:

```
file:/usr/exim/exim-html-4.00/doc/html/index.html
```

There is also a file called **spec.txt** in the **doc** directory in the Exim source tree. It contains a copy of the manual in ASCII format which can be searched with a text editor.

1.5 Test the standard installation and configuration

☞ To save typing, adjust your `PATH` variable so that the command *exim* can be used to run the Exim binary:

```
$ export PATH=/usr/exim/bin:$PATH
```

Substitute a real local user name for *localuser* in what follows. You should not be root when running these tests.

☞ First, check what Exim will do with a local address:

```
$ exim -bt localuser
```

This tests the delivery routing for a local account. See what output you get.

☞ Try with a non-existent local user and see what happens.

☞ Try something that is in **/etc/aliases**:

```
$ exim -bt postmaster
```

Exim will not normally deliver mail to a *root* mailbox (for security reasons) so what people usually do is to make *root* an alias for the sysadmin. In FreeBSD, all the default aliases point to *root*. Therefore, you should add a new alias to **/etc/aliases**:

```
root: yourname
```

Now try this again:

```
$ exim -bt postmaster
```

☞ Now try a real local delivery. You can pass a message directly to Exim without using an MUA:

```
$ exim -odf -v localuser
This is a test message.
.
```

The `-odf` option requests 'foreground' delivery, which means that the ***exim*** command won't return until the delivery is complete. (This avoids your shell prompt getting mixed up with Exim's output.)

The `-v` option turns on user verification output, which shows you copies of Exim's log lines.

☞ Check what is in Exim's logs:

```
cat /var/spool/exim/log/mainlog
cat /var/spool/exim/log/paniclog
```

The panic log should normally be empty, and if nothing has ever been written to it, it will not even exist. On a live system it is helpful to set up a ***cron*** job that mails you a warning if it ever finds a non-empty panic log.

☞ If the delivery succeeded, go and check the local user's mailbox.

```
$ ls -l /var/mail/localuser
$ cat /var/mail/localuser
```

If the delivery didn't succeed, you need to find out why. If the information in the log doesn't help, you can try the delivery again, with debugging turned on:

```
$ exim -odf -d localuser
<there will be output from Exim here>
This is another test message.
.
```

The `-d` option turns on debugging. You need to be an Exim administrator to do this. If you get a 'Permission denied' error, check that you are a member of the Exim group.

☞ If you are logged on as *localuser*, you can use the *mail* command to read the mail in the usual way. You could also try sending a message from the *mail* command.

The next thing is to test whether Exim can send to a remote host. You can try with your own email home address. The speed of this may vary, depending on the state of the network connection.

☞ First, check that Exim can route to the address:

```
$ exim -bt user@remote.host
```

☞ Now send a message to the remote address:

```
$ exim -odf -v user@remote.host
This is a test message.
.
```

The `-v` option causes it to display the SMTP dialogue as well as the log lines. If you can, check that the message arrived safely. If there are problems, see if you can figure out what went wrong and why.

☞ You won't be able to receive messages from a remote host until you start the Exim daemon:

```
$ /usr/exim/bin/exim -bd -q15m
```

The `-bd` option causes the daemon to listen for incoming SMTP calls, while the `-q15m` option causes it to start a queue runner process every 15 minutes. On a live system, starting the daemon should happen automatically on a reboot.

☞ Use telnet to check that the daemon is accepting SMTP calls:

```
$ telnet localhost 25
```

You should see an Exim greeting message. Use QUIT to exit.

☞ Now check that a remote host can send a message to your host, and see how Exim logs what happens. If that succeeds, you have a working basic installation correctly installed.

☞ Try sending to an invalid address from a remote host, and see what error message you get, and how Exim logs this case. Look in both **mainlog** and **rejectlog**.

1.6 Starting the Exim Monitor

You need to have an X-windows session running to run the monitor.

☞ Start the monitor:

```
$ /usr/exim/bin/eximon
```

The upper window shows a ‘tail’ of the main log; the lower window shows the messages that are waiting in the queue. Expect both to be empty to start with. Send a few messages and watch what the monitor displays.

1.7 Queue management tests

There are several command line options (and equivalent menu items in the monitor) for doing things to messages.

☞ To put a message on the queue without its being delivered, run

```
$ exim -odq address1 address2 ...
Test message.
.
```

The message will stay there until a queue runner process notices it.

☞ List the messages on the queue:

```
$ exim -bp
```

☞ Do a manual queue run, with minimal output:

```
$ exim -v -q
```

(Without `-v` you won’t see any output at all on the terminal, but there will be entries in the log.)

1.8 Checking relay control

☞ To demonstrate that Exim will relay by default via the loopback interface, try the following sequence of SMTP commands. Substitute the number of your host for *nn*:

```
$ telnet 127.0.0.1 25
ehlo localhost
mail from:<localuser@hostnn.t1.ws.afnog.org>
rcpt to:<localuser@hostnn.t1.ws.afnog.org>
rcpt to:<user@some.remote.domain>
```

You should get an OK response to all the SMTP commands. Type ‘quit’ to end the SMTP session without actually sending a message.

☞ Now try the same thing, but use your host’s IP address instead of 127.0.0.1.

```
$ telnet 80.248.72.nn 25
ehlo localhost
mail from:<localuser@hostnn.t1.ws.afnog.org>
rcpt to:<localuser@hostnn.t1.ws.afnog.org>
rcpt to:<user@some.remote.domain>
```

In this case, you should get the error message

```
550 relay not permitted
```

for the second `RCPT` command, which is the one that is trying to relay. The first `RCPT` command should be accepted, because it specifies a local delivery. You could also try telnetting from an external host and running the same check.

1.9 Processing log data

☞ Run **exigrep** to extract all information about a certain message, or a certain user's messages, or messages for a certain domain. For example:

```
$ exigrep localuser /var/spool/exim/log/mainlog
```

That extracts all the log information for all messages that have any log line containing '*localuser*'. It's a Perl pattern match, so you can use Perl regular expressions.

☞ To extract simple statistics from a log, run

```
$ eximstats /var/spool/exim/log/mainlog | more
```

There are options for selecting which bits you don't want. Details are in the manual.

2. Tests that involve changing the configuration

To change Exim's runtime configuration, you must edit **/usr/exim/configure** and then HUP the Exim daemon (as root). The daemon stores its process id (pid) in a file, in order to make this easy. This command restarts the daemon:

```
# kill -HUP ‘cat /var/spool/exim/exim-daemon.pid‘
```

You can confirm that the daemon has restarted by checking the main log. You are doing to be restarting Exim a lot, so make yourself a script to save typing:

```
# cat > /usr/local/bin/hupexim
#! /bin/sh
kill -HUP ‘cat /var/spool/exim/exim-daemon.pid‘
^D
# chmod a+x /usr/local/bin/hupexim
```

Now you can restart Exim just by running:

```
# hupexim
```

The following sections contain some suggestions for configuration modifications that you can try, just to get a feel for how the configuration file works.

2.1 Adding more local domains

☞ Edit the configuration, and change the **local_domains** setting so that it looks like this:

```
local_domains = @ : testnn.afnog.org
```

Remember to HUP the daemon afterwards. Now you have a new local domain. Try sending it some mail:

```
$ mail yourname@testnn.afnog.org
```

Check that it arrives in your mailbox.

> **Note:** The domains that we are adding now can only be used from your own host, because there are no DNS records for them. When you are adding domains to a production host, you must of course also add MX records for them.

If you want to add a lot of domains, or if you want to keep changing them, it is easier to keep the list of domains in a file instead of in the Exim configuration. (You can also keep them in several different kinds of database, but we don't cover that in this workshop.) We are now going to add some domains like this, and then make them into *virtual domains*.

☞ Create a file that contains a list of domains:

```
$ cat > /usr/exim/vdomains
vdom1.afnog.org
vdom2.afnog.org
^D
```

☞ Edit **/usr/exim/configure** to change the local domains setting:

```
local_domains = @ : testnn.afnog.org : \
                  lsearch;/usr/exim/vdomains
```

☞ You must also add a new router to handle these domains. Put this router *first*, before all the other routers:

```
virtual_domains:
  driver = redirect
  domains = lsearch;/usr/exim/vdomains
  data = ${lookup{$local_part}lsearch{/usr/exim/aliases-$domain}}
  no_more
```

(Remember to HUP the daemon.)

☞ Create an alias file for the first virtual domain:

```
$ cat >/usr/exim/aliases-vdom1.afnog.org
philip:    ph10@cam.ac.uk
yourname:  your email address
```

☞ Test that Exim recognizes the virtual addresses:

```
$ exim -bt philip@vdom1.afnog.org
```

Please don't actually send test mail to that address! I get too much junk already!

☞ Now create a different alias file for the second virtual domain, with 'philip' aliased to somebody else, and check (with **-bt**) that Exim treats that address differently.

It is always important to test that incorrect addresses are handled the way you want. So you need to run this test:

```
$ exim -bt unknown@vdom1.afnog.org
```

2.2 Catching undeliverable mail

☞ Add a **redirect** router that sends all undeliverable mail in your domain to the postmaster. Where in the list of routers should this go? Do you think that having a router like this is a good idea?

3. Allowing relaying from another host

In section 1.8 above, there is test to demonstrate that relaying is blocked if you connect to your host's IP address.

☞ Unblock it by changing a line in the configuration to let all the T1 hosts relay through your host. Change this line:

```
hostlist   relay_from_hosts = 127.0.0.1
```

to

```
hostlist   relay_from_hosts = 127.0.0.1 : 80.248.72.0/25
```

(Don't forget to HUP the daemon.) Then try the telnet test from section 1.8 again. This time it should accept the request to relay.

3.1 Allowing relaying to specific domains

The default configuration contains the line

```
domainlist relay_to_domains =
```

This defines domains to which your host will relay, wherever the message comes from.

☞ Add some domains to this line. For example, add the domain of your home email.

Now we need to test that Exim will indeed relay to those domains (but not to others) from a host that does not match **relay_from_hosts**. Exim has a testing facility that lets you simulate an SMTP call from a remote host. Run it like this:

```
$ exim -bh 192.168.1.1
```

You will see some debugging output, and then an SMTP greeting line. Now type SMTP commands:

```
ehlo testhost
mail from:<localuser@hostnn.t1.ws.afnog.org>
rcpt to:<user@your.home.domain>
rcpt to:<user@some.other.domain>
```

You will see the tests that Exim is making as it runs the ACL after each RCPT command.

4. More advanced configuration

These are ideas for things to do for those who have time. Don't worry if you do not get to this part. Not everything here is covered in the lectures, but it is all in the manual.

4.1 Demonstrate retry mechanisms

The easiest way to demonstrate what happens when Exim cannot deliver a message is to force connections to remote hosts to fail.

☞ Edit the configuration, and change the **remote_smtp** transport to be like this:

```
remote_smtp:
  driver = smtp
  port = 3456
```

(Remember to HUP the daemon.) This makes Exim try port 3456 instead of the SMTP port (25) when delivering, causing the remote host to refuse the connection (assuming you've chosen an unused port!)

☞ Send a message to a remote address and see what happens.

☞ Start a queue run

```
$ exim -q
```

and see what happens and what gets logged. Have a look at the message's own **msglog** file, which you can do from the monitor or by using the `-Mvl` option.

☞ Use **exinext** to see when it is next scheduled to deliver:

```
$ exinext remote.domain
```

☞ Remember to remove the setting of `port` when you have finished playing with retries (and HUP the daemon).

4.2 Add a system filter

☞ Add a system filter file to the configuration:

```
system_filter = /usr/exim/system.filter
system_filter_file_transport = address_file
```

Create this test filter file:

```
$ cat >/usr/exim/system.filter
# Exim filter
if $h_subject: is "spam" then save /dev/null endif
^D
```

☞ Now sent yourself a message with the subject 'spam' and see what happens.

4.3 Configure some address rewriting

☞ Add to the rewriting section of the configuration

```
othername@otherdomain.com   postmaster@your.domain
```

Then send a message to **othername@otherdomain.com** and see what happens. You can also test rewriting rules with `-brw`:

```
$ exim -brw othername@otherdomain.com
```

Have a look at the chapter on rewriting and experiment with other forms of rule.

* * *