Exim Practical

Objectives

Part 1 is building Exim from source, and installing it.

- Download Exim source and documentation
- Unpack the source and documentation
- Build Exim from the generic distribution
- Install Exim
- Replace Sendmail with Exim

Part 2 is running basic tests. You don't need to modify the Exim configuration to do this.

- Test a standard installation and default configuration
- Inspect and manage the mail queue
- Check relay control
- Process log data

Part 3 involves some simple modification of the runtime configuration.

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

Part 4 sets up your host as a mail relay

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

Part 5 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 used 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.

Common mistakes

In past workshops, these are the most common mistakes that have been made:

- *Doing everything as root.* You only need to be root to install Exim and change its configuration. Otherwise, you should do everything (including building Exim) under your normal login. In the sample commands, the command prompt is shown as **#** for commands that must be run as root, and **$** otherwise.

  In particular, running email tests as root is a bad idea, because root has privileges. You want to test that Exim is working when an ordinary, unprivileged user calls it.

- *Forgetting the dot that terminates a message.* When you type a message directly into Exim, it needs a line containing just a dot to terminate it. Until you type that line, all input is taken as part of the message.
- *Setting PATH incorrectly.* Whenever you change your PATH setting, be sure to check what you have typed carefully before pressing RETURN. If you mess up with PATH, you'll find that lots of commands ‘‘disappear’’.

- *Adding dots to email domains.* You should now have got used to inserting trailing dots in fully qualified domains in DNS zones. Unfortunately, in email configurations, trailing dots are *not* used and will cause problems if you use them.

1. Installing Exim

We are going to install Exim from the generic distribution This way of doing it allows you to make your own choices at build time.

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

Preliminary preparation

▶ If a **sendmail** daemon is running, kill it off. Use **ps** with **grep** to find it. (Modern versions of sendmail may have two or more processes running)

```
# ps auxw | grep sendmail
# /etc/rc.d/init.d/sendmail stop
# chkconfig --del sendmail
```

Create a user and group called *exim.* These will be used for running Exim when it does not need to be root.

```
# useradd -M -u 90 exim
```

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

```
# usermod -G exim username
# grep exim /etc/group           (To check)
```

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

```
# chmod 1777 /var/spool/mail
```

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

Download the source and documentation

▶ As root, 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.taller.nsrc.org
```

Log in as *anonymous.*

```
ftp> cd /pub/software
```

```
ftp> get exim-4.30.tar.bz2
ftp> get exim-html-4.30.tar.bz2
ftp> bye
```

Unzip and untar the source and the HTML documentation:

```
$ cd /usr/exim
$ bunzip2 exim-4.30.tar.bz2
$ tar -xvf exim-4.30.tar
$ bunzip2 exim-html-4.30.tar.bz2
$ tar -xvf exim-html-4.30.tar
```

Check the documentation

Before moving on, make sure you can access 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-4.30/doc/html/index.html
```

There should also be a file called **/usr/exim/exim-4.30/doc/spec.txt**. It contains a copy of the manual in ASCII format which can be searched with a text editor.

Building Exim

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

```
$ cd /usr/exim/exim-4.30
```

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. Find the lines that contain `EXIM_USER` and `SUPPORT_MAILDIR`, and change them to be like this:

```
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 are closer to what an RPM installation might use:
>
> ```
> BIN_DIRECTORY=/usr/local/sbin
> CONFIGURE_FILE=/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 go 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. Easy, wasn't it?

Installing Exim

You need to be root to install Exim:

```
# cd /usr/exim/exim-4.30
# 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 and some other information about which features are included.

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.

We want programs to call exim rather than sendmail, but leave sendmail installed so that if there is any existing mail in sendmail's queue, it can be flushed later. Red Hat lets us choose a new MTA by setting a symlink in the `/etc/alternatives` directory. We redirect the programs 'sendmail' and 'mailq' to point to exim.

```
# cd /etc/alternatives
# rm mta
# ln -s /usr/exim/bin/exim mta
# rm mta-mailq
# ln -s /usr/exim/bin/exim mta-mailq
```

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.

2. Testing Exim

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.

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. **Take great care** when you do this, because messing up your PATH will make many commands ``vanish''. Type this command exactly, taking care with the colon and dollar in the middle:

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

Make sure you substitute a real local user name for *localuser* in what follows. Remember, 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:

```
$ exim -bt junkjunkjunk
```

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 Red Hat, all the default aliases point to *root*. Therefore, you should add a new alias to **/etc/aliases**. Add this line at the end:

```
# Person who should get root's mail
root:          yourname
```

Now try this again:

```
$ exim -bt postmaster
```

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

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

Note: the message is terminated by a line that just contains a dot. Be sure to type it! (Alternatively, you can send ‘‘end of file’’ by pressing CTRL-D.)

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

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.)

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. *Tip*: 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 you get a permission error, make sure that your username is in the ’exim’ group, then logout and login again to become a member of that group.

If the delivery succeeded, you should see two lines in the main log, one containing `<=` for the message arriving, and one containing `=>` for the delivery.

▶ Now go check the local user's mailbox:

```
$ ls -l /var/spool/mail/localuser
$ cat /var/spool/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 -d -odf localuser
<there will be output from Exim here>
This is another test message.
.
```

The `-d` option turns on debugging, which gives a lot more information than `-v`. You need to be an Exim administrator to use `-d`. 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. The speed of this may vary, depending on the state of the network connection. In what follows, replace *user@remote.host* with your home email address.

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

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

▶ Now send a message to the remote address:

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

This time, the `-v` option causes Exim 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 -q20m
```

The `-bd` option causes the daemon to listen for incoming SMTP calls, and the `-q20m` option causes it to start a queue runner process every 20 minutes. On a live system, starting the daemon should happen automatically on a reboot, by putting this command in `/etc/rc.local` .

▶ 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**.

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.

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 stays on the queue until a queue runner process notices it.

List the messages on the queue:

```
$ exim -bp
```

Do a manual queue run, with minimal verification 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.)

Checking relay control

To demonstrate that Exim will relay by default via the loopback interface, try the following sequence of SMTP commands. Wait for Exim to respond to each command before typing the next one. Substitute the number of your pc for *nn*:

```
$ telnet 127.0.0.1 25
ehlo localhost
mail from:<localuser@pcnn.taller.nsrc.org>
rcpt to:<localuser@pcnn.taller.nsrc.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 xx.xx.xx.nn 25
ehlo localhost
mail from:<localuser@pcnn.taller.nsrc.org>
rcpt to:<localuser@pcnn.taller.nsrc.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.

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. If you have time, experiment with the options for outputting the statistics as HTML.

3. 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:

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

where *nnnn* is the pid from the previous line.

You can confirm that the daemon has restarted by checking the main log. You are going to be restarting Exim a lot, so make yourself a script to save typing. Use *vi* to create a file called **/usr/local/bin/hupexim**, containing these lines:

```
#!/bin/bash
kill -HUP $(cat /var/spool/exim/exim-daemon.pid)
```

Note that the # character in the first line is part of the file (it's not a prompt). Now make the new file into an executable script:

```
# chmod a+x /usr/local/bin/hupexim
```

If you are using the C-shell (*csh*) you must also run this command:

```
# rehash
```

This causes the internal hash table of the contents of the directories in the PATH variable to be recomputed. This is not necessary if you are using *bash*.

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. You do not have to stick rigidly to these examples; use different domain names or user names if you want to.

Adding more local domains

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

```
domainlist local_domains = @ : testnn.nsrc.org
```

where *nn* is the number of your host. 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, such as LDAP or MySQL, 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*.

Use *vi* to create a file called **/usr/exim/vdomains** that contains a list of domains (as many as you like):

```
vdom1.nsrc.org
vdom2.nsrc.org
...
```

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

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

Note: There is no space following the semicolon. This change makes all the new domains into local domains.

Now we add a new router to handle these domains as virtual domains. Put this router *first*, before all the other routers, immediately after the line ''begin routers'':

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

There must be no space after the semicolon in the ''domains'' line. (Remember to HUP the daemon.)

Create an alias file for the first virtual domain -- use *vi* to make the file

/usr/exim/aliases-vdom1.nsrc.org containing these lines:

```
brian:     b.candler@pobox.com
yourname:  your email address
```

The local parts *brian* and *yourname* should now be valid for the first virtual domain.

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 *brian* aliased to somebody else, and check (with **-bt**) that Exim treats that address differently. **Note:** 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.nsrc.org
```

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? See if you can work out how to do this on your own without looking at the answer below. Do you think that having a router like this is a good idea on a busy host?

Here is a sample router that does this job:

```
unknown_to_postmaster:
  driver = redirect
  data = postmaster
```

It should be placed last, after all the other routers. Test it by sending mail to an unknown user.

4. Relaying from another host

In section 2 above, there is test to demonstrate that relaying is blocked if you connect to your host's IP address.

We are now going to remove this block by changing a line in the configuration to let all the classroom 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 : xx.xx.xx.xx/mm
```

where *xx.xx.xx.xx/mm* is the classroom network. (Don't forget to HUP the daemon.) Then try the telnet test from section 2 again. This time it should accept the request to relay. Ask one of the other students to try relaying through your host -- it should work. If you can, telnet from a host outside the classroom network, and confirm that relaying is still blocked.

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. As you can see, the default list is empty, so no domains match.

▶ Add some domains to this line. For example, add the domain of your home email. In my case, this would be:

```
domainlist relay_to_domains = pobox.com
```

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, waiting for a response between each one:

```
ehlo testhost
mail from:<localuser@pcnn.taller.nsrc.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. Check that it allows relaying to the right domains, and not to any others. End the SMTP session with QUIT.

5. 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 and Philip Hazel's Exim 4 book.

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. For example:

```
$ exim -Mvl 19EdUm-00016A-IA
```

(That is an example message ID; you must use the real one for the message that is on your queue.)

▶ Use **exinext** to see when Exim is next scheduled to deliver to the host that failed:

```
$ exinext remote.domain
```

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

Add a system filter

▶ Use *vi* to create a test system filter file in **/usr/exim/system.filter**, containing these lines:

```
# Exim filter
if $h_subject: is "spam" then save /dev/null endif
```

▶ Arrange for Exim to use the system filter by adding these lines to the configuration (somewhere near the beginning, before the first ‘‘begin’’ line):

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

▶ Now send yourself a message with the subject ‘spam’ and see what happens.

Configure some address rewriting

▶ Find the rewriting section of the configuration (the part that starts with ‘‘begin rewrite’’). Then add this line:

```
othernameotherdomain.com   postmasteryour.domain
```

▶ Now send a message to **othername@otherdomain.com** and see what happens.

▶ You can test rewriting rules with the `-brw` command line option:

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

What next?

If you have got this far in the available time, you are probably starting to understand the basics of Exim pretty well. You can either start reading the documentation, or help out other students who are having problems.