Courier worksheet

Updated for Linux (Fedora Core 2)
July 2004

For scalability, we are going to arrange for exim to deliver all local mail in Maildir format. This creates a subdirectory called "Maildir" in the user's home directory, which in turn contains three subdirectories: `new`, `cur` and `tmp`. Messages are written into `tmp`, moved to `new` when delivery is complete, and moved to `cur` when read. Each message has a long unique filename based on the hostname and the time of day.

Because each message is stored in a separate file, it is much faster for the pop3 daemon to start up every time a user connects. It also allows for safe delivery onto a shared (NFS) disk backend.

Exim does not include any software for retrieving mail from a mailbox, so we need to install additional software. Courier is a mail system which includes a number of packages. In fact it has its own MTA, but we will ignore this (it is still under heavy development, and does not have the flexibility needed for an ISP environment). The components we are interested in are the IMAP/POP3 servers and 'sqwebmail', the webmail server.

You can get the entire courier system as one package (including the MTA), or just the components. We will get the pop3/imap and webmail components separately.

Remember: in the command examples given below, commands shown with the prompt "$" should be run as your normal non-root userid. Only those commands with prompt "#" need to be run as root.

---

1. Reconfigure exim for Maildir local delivery

Edit `/usr/exim/configure`, find the `local_delivery` transport and modify it as follows:

```
local_delivery:
  driver = appendfile
  directory = $home/Maildir
  maildir_format
  maildir_use_size_file
#  file = /var/mail/$local_part
  delivery_date_add
  envelope_to_add
  return_path_add
# group = mail
# mode = 0660
```

Optionally you could add further parameters to this transport which let you impose quotas on your users, for example to limit all users to 10 megabytes of storage each:

```
  maildir_tag = ,S=$message_size
  quota_size_regex = ,S=(\d+)
  quota = 10M
  quota_warn_threshold = 90%
```

(Aside: this quota mechanism relies on users not meddling with the quota information which is stored within their maildir; in other words, users with shell access would be able to bypass their quota if they knew what they were doing)

Remember to HUP your exim daemon. Now test out your new configuration by delivering to some local account on your machine. Try this with something other than root at first:

```
$ /usr/exim/bin/exim -bt localuser
localuser@pcnn.t1.ws.afnog.org
  router = localuser, transport = local_delivery
$ /usr/exim/bin/exim localuser
Here is a test
.
$ cd /home/localuser/Maildir
$ ls
cur     new     tmp
$ ls new
102078119.7969.pcnn.t1.ws.afnog.org,S=426
$ cat new/*
Return-path: <root@pcnn.t1.ws.afnog.org>
...
Here is a test
```

Note: once you have changed to Maildir delivery, you will find that any local Unix MUA (which looks for new messages in /var/mail/username) will no longer see your incoming mail. How to fix this depends on which MUA you are using. Some examples:

mutt
: Edit `/usr/local/etc/Muttrc` and put:
  set spoolfile="~/Maildir/"

pine
: To be confirmed
  [Other commands, 'S' for Setup, 'C' for Config]

elm
: To be confirmed

kmail
: Doesn't appear to access /var/mail or Maildir directly; needs to use POP3/IMAP to retrieve new mail

---

2. Install courier-imap

Homepage: http://www.inter7.com/courierimap/

If you don't have them already, fetch **courier-imap-3.0.6.tar.bz2** and **sqwebmail-4.0.6.tar.bz2** from the **pub/software/courier**, and the **pub/software/sqwebmail** directories on noc.sanog.org. (Grabbing sqwebmail will sae us time later.)

For now, place both files in your own account - *dont use root!*. Courier-imap will not install correctly if your expand out the file and/or compile it initially as root!

Normally when Courier-imap builds it installs itself under `/usr/lib/courier-imap`; this can be overridden using the `--prefix=` option to ./configure, but we will stick with the default for now.

From the directory where you downloaded to:

```
$ bzip2 -d courier-imap-3.0.6.tar.bz2   [Remember, you cannot be root yet!]
$ tar -xvf /path/to/file/courier-imap-3.0.6.tar
$ cd courier-imap-3.0.6
$ ./configure --with-redhat
... takes a while
$ make
... takes a while
$ make check
... takes a while, check there are no errors displayed
$ su
```

```
Password: <root password>
# make install
# make install-configure
```

To get access to the man pages, edit **/etc/man.config** and add the following line:

```
MANPATH /usr/lib/courier-imap/man
```

Around line 35, or at the end of the first group of "MANPATH" statements.

Test with 'man userdb' and see if the page is displayed.

Note: Consider installation using a Red Hat/Fedora RPM instead. You can see the detailed notice concering this by running "./configure" without the "--with-redhat" option if you wish. In addition, Fedora Core 2 probably won't have "expect" installed, thus you won't be able to change passwords via a webmail interface. You could install "expect" before installing courier-imap to rectify this issue.

---

3. Configure the daemons

Courier can get its user and password information from a variety of places, using authentication modules. Some of these, like mysql and ldap, are only compiled if those pieces of software are already installed (see INSTALL in the source directory for more information).

With current versions of Courier-IMAP, all these authentication methods are built into a single authentication daemon, "authdaemond"

In many cases the only configuration you need to do for the pop3 and imap daemons is to increase the maximum number of concurrent connections from the default of 40, if you have a fairly powerful mailserver:

```
# cd /usr/lib/courier-imap/etc
# vi pop3d
...
MAXDAEMONS=300
...
# vi imapd
...
MAXDAEMONS=300
...
```

However it's recommended that you configure the authdaemond process not to use any authentication mechanisms which you know you don't need. For example, if all your authentication is only via PAM for Unix system passwords, then you can remove all the others (or comment out the line, and add the line just below):

```
# cd /usr/lib/courier-imap/etc
# vi authdaemonrc
...
authmodulelist="authpam"
...
```

The "authpam" modules lets you authenticate against your Unix password file (PAM = Pluggable Authentication Modules). Some configuration of PAM may be needed; the files are `/etc/pam.d/pop3` and `/etc/pam.d/imap` for POP3 and IMAP respectively. If there is a problem with these files, then under FreeBSD-5.x you can fix them like this:

```
# cd /etc/pam.d
# cp other imap
# cp other pop3
```

---

4. Start the daemons

Make sure you have no existing pop3 server enabled in /etc/inetd.conf. You can start the pop3 and/or imap servers using one or both of the following commands, which can go in /etc/rc.local to run at startup time:

```
# /usr/lib/courier-imap/libexec/pop3d.rc start
# /usr/lib/courier-imap/libexec/imapd.rc start

To check they are running:
# ps aux | grep couriertcpd
```

Test using telnet to port 110 (POP3) and 143 (IMAP):

```
# telnet localhost 110
Connected to localhost.
Escape character is '^]'.
+OK Hello there.
user username
+OK Password required.
pass password
+OK logged in.
stat
+OK 26 49857
retr 1
+OK 1073 octets follow.
... message
.
quit
+OK Bye-bye.
Connection closed by foreign host.
```

If you use the same user account you sent a test message to earlier then you should see this message, or older messages, appear on the screen.

Note that every IMAP command must be prefixed by a 'tag' used to associate replies with commands; the tag is an arbitary string. In this example we will just use "a" as the tag.

```
# telnet localhost 143
Connected to localhost.
Escape character is '^]'.
* OK Courier-IMAP ready. Copyright 1998-2001 Double Precision, Inc.  See
COPYING for distribution information.
a login <username> <password>
a OK LOGIN Ok.
a examine inbox
* FLAGS (\Answered \Flagged \Deleted \Seen \Recent)
* OK [PERMANENTFLAGS ()] No permanent flags permitted
* 26 EXISTS
* 0 RECENT
* OK [UIDVALIDITY 989061119] Ok
a OK [READ-ONLY] Ok
a logout
* BYE Courier-IMAP server shutting down
a OK LOGOUT completed
Connection closed by foreign host.
```

Check in the log files for records of successful and failed logins:

```
$ tail /var/log/maillog
May 14 15:01:06 noc pop3d: LOGIN, user=inst, ip=[::1]
May 14 15:01:09 noc pop3d: LOGOUT, user=inst, ip=[::1], top=0, retr=0
```

NOTE: You won't be able to login if the Maildir does not exist. Hence you need to have delivered at least one message to the user, to create their mailbox, before they can login. Alternatively, you can use the courier utility 'maildirmake' to create an empty maildir structure:

```
$ cd
$ /usr/lib/courier-imap/bin/maildirmake Maildir
```

---

5. Debugging authentication problems

You can configure courier-imap to log extended information when processing a login, which is especially useful if you are finding that logins are failing. Full details of how to do this are in the file `authlib/README.authdebug.html` within the source directory, but in summary what you need to do is:

```
# vi /usr/lib/courier-imap/etc/pop3d
or
# vi /usr/lib/courier-imap/etc/imapd
Change:
DEBUG_LOGIN=0
To:
DEBUG_LOGIN=1    # or DEBUG_LOGIN=2, which also logs passwords
```

```
If you changed the pop3d file do:
```

```
# /usr/lib/courier-imap/libexec/pop3d.rc stop
# /usr/lib/courier-imap/libexec/pop3d.rc start
```

```
If you changed the imapd file do:
```

```
# /usr/lib/courier-imap/libexec/imapd.rc stop
# /usr/lib/courier-imap/libexec/imapd.rc start
```

You also need configure the syslog daemon so that messages at level 'debug' are recorded. In a standard Linux configuration, can add the following line to the file `/etc/syslog.conf` to send all debug messages to a particular file:

```
*.=debug                                        /var/log/debug.log
```

If you prefer, you can configure /var/log/maillog so that all mail information (both debugging and normal) goes to this file:

```
# vi /etc/syslog.conf
Change:
mail.info                                       /var/log/maillog
To:
mail.debug                                      /var/log/maillog
```

```
Next, restart the system logger service (as root):
```

```
# service syslog restart
```

If you did "su" instead of "su -" to become root, then this will fail. You would need to specify the path "service", which is `/sbin/service`.

If you run 'tail -f /var/log/*debug.log*' in one window while you make a POP3 or IMAP login in another window, you should see much more information about the decision-making process - the login being passed to each of the authentication modules in turn, and whether each module decided to ACCEPT, REJECT (pass to next module), or TEMPFAIL (abort, don't try any further modules). Don't forget to reset your DEBUG_LOGIN level once you are finished.

---

6. pop3 and imap over SSL

If you wish, you can enable pop3 over SSL (port 995) and imap over SSL (port 993). The advantage is that, for clients which support it, the traffic is encrypted. The disadvantage is higher CPU load on your server for the encryption of data.

To run SSL you will need a certificate. For testing purposes you can use a 'self-signed' certificate, the following scripts will generate them for you:

```
# /usr/lib/courier-imap/sbin/mkpop3dcert
# /usr/lib/courier-imap/sbin/mkimapdcert
```

Then you start the servers:

```
# /usr/lib/courier-imap/libexec/pop3d-ssl.rc start
# /usr/lib/courier-imap/libexec/imapd-ssl.rc start
```

Don't forget to add these to /etc/rc.local if you want to have them start automatically when you reboot your system.

You can't use a regular telnet to test it, because all your communication needs to be encrypted, but openssl has an SSL client you can use to make an encrypted connection (to POP) for testing:

```
# openssl s_client -connect localhost:995
```

And, as seen earlier, you can test your POP connection like so:

```
Connected to localhost.
Escape character is '^]'.
+OK Hello there.
user username
+OK Password required.
pass password
+OK logged in.
stat
+OK 26 49857
retr 1
+OK 1073 octets follow.
... message
.
quit
+OK Bye-bye.
Connection closed by foreign host.
```

You can make an encrypted connection to IMAP by doing:

```
# openssl s_client -connect localhost:995
```

And, then running the same test as earlier.

If you were running the service commercially you would be better to get a proper certificate signed by a recognised CA, rather than using a self-signed certificate.

Actually, even POP3 over port 110 and IMAP over port 143 also support encryption, using the commands STLS / STARTTLS; this allows the client to connect on the normal port and then 'upgrade' the connection from normal to encrypted. Only more modern clients support this mode of operation.

---

7. sqwebmail

Homepage: http://www.inter7.com/sqwebmail/

webmail is a very useful service to offer your clients - although you may need to be careful of the extra CPU load and bandwidth it might use.

Unlike many other webmail solutions, which use POP3 or IMAP to talk to the mail store, sqwebmail reads and writes Maildir directories directly. This makes it efficient in the case where POP/IMAP and webmail run on the same box, or where there is an NFS-shared mailstore. It's also written in compiled C, not a scripting language like PHP.

sqwebmail is feature-rich, very customisable through HTML templates and stylesheets, supports multiple languages, and is simple to install (it runs as a single CGI). Note however that it is still under very active development and hence subject to change quite frequently.

If you don't have it, install and test Apache and mod_ssl first. You should have both of these already installed at this point.

Also, if you install 'ispell' and/or 'gnupg' before compiling sqwebmail, then these will be detected and usable from the web interface.

Sqwebmail installs in `/usr/local/share/sqwebmail` by default. Again, you can change this using the `--prefix=` option on the ./configure command line.

As your own account, not root, do:

```
$ cd /location/of/sqwebmail
$ bzip2 -d sqwebmail-4.0.6.tar.bz2
```

```
$ tar -xvf /path/to/file/sqwebmail-4.0.6.tar
$ cd sqwebmail-4.0.6
$ ./configure     [Will warn about GnuPG version on Fedora Core 2]
... takes a while
$ make configure-check
(check that it has chosen the right locations for cgi-bin and webmail images)
SqWebMail CGI will be installed in /var/www/cgi-bin, and Images will be installed in
/var/www/html/webmail. URL to the image directory is /webmail.
$ make
... takes a while
$ make check
... takes a short while, check there are no errors displayed
$ su
Password: <root password>
# make install-strip
# make install-configure
```

It will automatically install the CGI as /var/www/cgi-bin/sqwebmail. It uses the same types of authentication modules as Courier-imap (configured in `/usr/local/share/sqwebmail/authdaemonrc`)

Hint: So, you need to edit /usr/local/share/sqwebmail/authdaemonrc and make sure that "authmodulelist" used the appropriate authentication module for your system. If you run in to problems look in /var/log/maillog for possible error messages concerning authentication.

Fedora Core 2 PAM Authentication Issue: Finally, when you log in on sqwebmail as we have it configured here the authorization daemon for sqwebmail (authdaemond) will attempt to verify your username and password using PAM and. PAM will see sqwebmail as a new service called "webmail" so it is necessary to tell PAM what requirement are needed for someone to authenticate. You can either read up on PAM for Fedora Core 2 (Red Hat) here:

> http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/ref-guide/ch-pam.html

or you can simply borrow an already preconfigured PAM service file that we are using with courier-imap for pop authentication. To do this type:

```
# cp /etc/pam.d/pop3 /etc/pam.d/webmail
```

If you cannot get sqwebmail to authenticate, then turn on debugging output for sqwebmail and look for something like this in your /var/log/debug.log file:

```
Jul 26 09:28:41 localhost authdaemond.ldap: pam_service=webmail, pam_username=nessus
Jul 26 09:28:41 localhost authdaemond.ldap: pam_authenticate failed, result 7
Jul 26 09:28:41 localhost authdaemond.ldap: authpam: TEMPFAIL - no more modules will be tried
Jul 26 09:28:42 localhost sqwebmaild: authdaemon: TEMPFAIL - no more modules will be tried
```

This indicates you do not have a PAM service file called "webmail" in /etc/pam.d, so you need to do what was indicated above.

Next run the startup script; this starts a pool of sqwebmaild processes (which perform the actual webmail work) and authdaemond processes. The 'sqwebmail' CGI is actually just a small stub program which takes the CGI request and squirts it down a socket to the pool of sqwebmaild processes, and therefore cannot work if these daemons have not been started.

```
# /usr/local/share/sqwebmail/libexec/sqwebmaild.rc start
```

You need to put that command in `/etc/rc.local` so that it starts automatically when the machine is next rebooted.

One other change is required: you need a daily cron script to clean out old sessions. In the directory /etc/cron.daily create a file called "sqwebmail" and placce the following lines in it:

```
#!/bin/sh

/usr/local/share/sqwebmail/cleancache.pl
```

Then, using a web browser, go to `http://`*`yourmachine`*`/cgi-bin/sqwebmail` to log in, or https:// if you have SSL running. You should be able to login, read and send mail.

The INSTALL file in the source tarball has more information on other configuration changes you can make.

If you wish to turn on authentication debugging, edit the file `/usr/local/share/sqwebmail/sqwebmaild` and set the DEBUG_LOGIN parameter, just the same as courier-imap.

Remember, when you are all done with this your /etc/rc.local file should have the following startup commands included:

```
/usr/lib/courier-imap/libexec/pop3d.rc start
/usr/lib/courier-imap/libexec/pop3d-ssl.rc start
/usr/lib/courier-imap/libexec/imapd.rc start
/usr/lib/courier-imap/libexec/imapd-ssl.rc start
/usr/local/share/sqwebmail/libexec/sqwebmaild.rc start
```

You'll probably end up creating init scripts in /etc/rc.d/init.d over time (or finding them) if you use courier-imap and sqwebmail on a permanent basis.

---

8. Maildir++ and quotas

Courier implements an extension to Maildir called Maildir++ which efficiently keeps track of quota status for each mailbox, even when the user has thousands of stored messages, and also allows the Maildir to contain sub-folders. For this to work, all software which accesses the Maildir must adhere to this extension.

As of version 4.31, Exim now has built-in support for Maildir++. To enable this you just need to set option 'maildir_use_size_file' on the transport, which we have already done.

Once Exim has delivered a message to the maildir, it creates a file called "maildirsize" which contains information about the quota and the number and size of messages within the maildir. This file is detected and read by courier-imap and sqwebmail. You therefore don't need to configure those programs with quota information; they will pick up the value which Exim has set.

As an alternative to using Exim's built-in Maildir++ code, you can get Exim to call the external 'deliverquota' program (supplied as part of courier-imap) whenever it wants to deliver a message. Just for reference, this is how it is done - but don't do this in the exercise unless you have plenty of spare time. Using Exim's built-in Maildir++ code is now normally the preferred option, because it is more efficient than spawning an external program, but this demonstrates how simple it is to configure Exim to use a different local delivery agent.

Modify the 'localuser' router:

```
localuser:
  driver = accept
  check_local_user
  transport = local_delivery_courier
  cannot_route_message = Unknown user
```

Add a new entry in the 'transports' section:

```
local_delivery_courier:
  driver = pipe
  command = /usr/lib/courier-imap/bin/deliverquota -c -w 90 \
        $home/Maildir 10000000S
  return_fail_output

  delivery_date_add
  envelope_to_add
  return_path_add
```

In this case, note that exim's "quota" and "quota_warn_threshold" settings are ignored. Instead, we pass the quota on the command line ("10000000S", where the trailing 'S' means 'bytes', and "-w 90" for a warning when the mailbox gets to 90% full)

If you want to deliver a warning message when the mailbox gets nearly full, you also need to create a file containing the warning message:

```
# cd /usr/lib/courier-imap/etc
# cp quotawarnmsg.example quotawarnmsg
# vi quotawarnmsg
```

You should edit /usr/lib/courier-imap/etc/quotawarnmsg to contain text that is meaningful to your site.

Last update July 26, 2004 by Hervey Allen
Materials by Brian Candler