Courier worksheet

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
  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:

```
$ /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 your MUA. For example:

mutt
In /usr/local/etc/Muttrc put:
set spoolfile="~/Maildir/"

---

2. Install courier-imap

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

Compilation requires GNU make ("gmake"), so first we need to check that this package has been installed. If not then fetch and install the package.

```
$ pkg_info | grep gmake
gmake-3.80_1        GNU version of 'make' utility
```

If you don't have them already, fetch **courier-imap-3.0.4.tar.bz2** and **sqwebmail-4.0.4.tar.bz2** from the **pub/src** directory on noc.t1.ws.afnog.org.

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.

```
$ cd
$ tar -xvyf /path/to/file/courier-imap-3.0.4.tar.bz2
$ cd courier-imap-3.0.4
$ ./configure
... takes a while
$ gmake
... takes a while
$ gmake check
... takes a short while, check there are no errors displayed
$ su
Password: <root password>
# gmake install
# gmake install-configure
```

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

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

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

---

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:

```
# 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 auxwww | 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.
```

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
Change:
DEBUG_LOGIN=0
To:
DEBUG_LOGIN=1    # or DEBUG_LOGIN=2, which also logs passwords

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

You also need configure the syslog daemon so that messages at level 'debug' are recorded. In a standard FreeBSD-5.2.1 configuration, you should find that all 'debug' messages go to a file called /var/log/debug.log; check for a line in `/etc/syslog.conf` like

```
*.=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
# killall -HUP syslogd
```

If you run 'tail -f /var/log/*debug.log*' in one window while you make a POP3 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)

---

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
```

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 for testing:

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

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 first: you can do this by installing packages "apache+modssl" and "mm", or building them from the FreeBSD ports collection.

```
# cd /usr/ports/www/apache13-modssl
# make all install
# make clean
# /usr/local/sbin/apachectl start
```

(Note: using the FreeBSD port will mean you must use some different paths than the examples below, e.g. /usr/local/www/data instead of /usr/local/apache/htdocs)

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.

```
$ cd
$ tar -xvyf /path/to/file/sqwebmail-4.0.4.tar.bz2
$ cd sqwebmail-4.0.4
$ ./configure
... takes a while
$ gmake configure-check
(check that it has chosen the right locations for cgi-bin and webmail images)
SqWebMail CGI will be installed in /usr/local/apache/cgi-bin
Images will be installed in /usr/local/apache/htdocs/webmail
URL to the image directory is /webmail
$ gmake
... takes a while
$ gmake check
... takes a short while, check there are no errors displayed
$ su
Password: <root password>
# gmake install-strip
# gmake install-configure
```

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

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: add the following line to `/etc/crontab` to periodically clean out old sessions:

```
0       *       *       *       *       bin     /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.

---

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
```

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
```

---

9. Security and virtual accounts

In the simple examples above, we have been using the system password file to authenticate users. When creating new "E-mail only" accounts on your system, you probably don't want your users to be able to login to Unix using ssh or telnet. To disable this, create their accounts with a nonexistent shell.

```
# pw useradd username -m -s /nonexistent
```

To improve security and scalability further, you may wish to keep all your mail accounts in a completely separate password file or database; these users won't be known to Unix at all. The mail directories and messages have to be owned by *some* Unix user, so we can make them all owned by the 'exim' user.

You have many choices of authentication module: for example an LDAP database, a mysql or postgresql database, or a local dbm file (courier supports a format called 'userdb'). These databases will contain the mail login usernames and passwords, and the directories where the mail will be stored. You'll need to configure Courier to use this new database as a login source, and also configure Exim to read this source to determine whether a user exists and where to deliver mail to.

The following example shows how to do this with authuserdb, which is described in "man makeuserdb" and "man userdb". We'll create two users in the table, and two empty maildirs. We'll make them under directory `/var/vmail01`, which we'll assume is a fast SCSI hard drive. To support many domains we'll make the POP3 login be *user@domain* instead of just a username. We'll have a separate directory for each domain, and also use the first two characters of the username as subdirectories, so that if we have ten thousand users for one domain we don't end up with ten thousand accounts within the same directory.

First we make an empty userdb, and make sure it's not world readable

```
# touch /etc/userdb
# chmod 700 /etc/userdb
```

Next we create some accounts and empty maildirs

```
# /usr/lib/courier-imap/sbin/userdb fred@flintstone.org set uid=90 gid=90 \
          home=/var/vmail01/flintstone.org/f/r/fred
# /usr/lib/courier-imap/sbin/userdbpw -md5 | /usr/lib/courier-imap/sbin/userdb fred set systempw
Password: wibble
Reenter password: wibble
# mkdir -p /var/vmail01/flintstone.org/f/r/fred
# /usr/lib/courier-imap/bin/maildirmake /var/vmail01/flintstone.org/f/r/fred/Maildir
# chown -R exim:exim /var/vmail01/flintstone.org/f/r/fred

# /usr/lib/courier-imap/sbin/userdb wilma@flintstone.org set uid=90 gid=90 \
          home=/var/vmail01/flintstone.org/w/i/wilma
# /usr/lib/courier-imap/sbin/userdbpw -md5 | /usr/lib/courier-imap/sbin/userdb wilma set systempw
Password: boing
Reenter password: boing
# mkdir -p /var/vmail01/flintstone.org/w/i/wilma
# /usr/lib/courier-imap/bin/maildirmake /var/vmail01/flintstone.org/w/i/wilma/Maildir
# chown -R exim:exim /var/vmail01/flintstone.org/w/i/wilma
```

(In real life you'd write a script to automate this process for creating new accounts). The compulsory fields we need to provide are "home" and numeric "uid" and "gid"; these are documented in "man makeuserdb".

Now we check the userdb contents, then convert it into userdb.dat which is the fast indexed version that authuserdb reads. Note that authuserdb requires encrypted passwords.

```
# cat /etc/userdb
fred@flintstone.org     home=/var/vmail01/f/flintstone.org/fred|systempw=$1$96YgsKCe$0oey3dzw0mztdb
wilma@flintstone.org    home=/var/vmail01/f/flintstone.org/wilma|systempw=$1$nXNJyXcB$1mItZjaFmOV/3
# /usr/lib/courier-imap/sbin/makeuserdb
# ls -l /etc/userdb.dat
-rw-r--r--  1 root  wheel  65536 May 14 16:23 /etc/userdb.dat
```

Now you can configure courier-imap and sqwebmail to login using these new accounts. If you leave authpam in the configuration then you can login with both the userdb accounts and the system accounts.

```
# vi /usr/lib/courier-imap/etc/authdaemonrc
...
authmodulelist="authuserdb authpam"
# /usr/lib/courier-imap/libexec/pop3d.rc stop
# /usr/lib/courier-imap/libexec/pop3d.rc start
also restart pop3d-ssl.rc, imapd.rc, imapd-ssl.rc if you are using them

Or for sqwebmail:
# vi /usr/local/share/sqwebmail/authdaemonrc
...
authmodulelist="authuserdb authpam"
# /usr/local/share/sqwebmail/libexec/sqwebmaild.rc stop
```

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

At this point you should be able to login using one of the new accounts, and see the (empty) mail directory.

```
# telnet localhost 110
Trying 127.0.0.1...
Connected to localhost.t1.ws.afnog.org.
Escape character is '^]'.
+OK Hello there.
user fred@flintstone.org
+OK Password required.
pass wibble
+OK logged in.
stat
+OK 0 0
quit
+OK Bye-bye.
Connection closed by foreign host.
```

If this doesn't work, follow the instructions for debugging authentication problems given earlier.

Now all that is necessary is for Exim to know how to deliver messages to these users. There are a couple of ways this can be done; Exim can be configured to read `/etc/userdb.dat` directly, or it can be configured to talk to courier's authdaemond process. Both can be set up using Exim's general-purpose configuration language.

It's a tribute to the flexibility of Exim that this can be done even though Exim does not have any specific features for using userdb or authdaemond, although it does mean that the configuration looks complicated at first glance. If you want to find out in detail how these configurations work, you will need to read the Exim documentation carefully.

Firstly, we need a separate list of which domains which need lookups in the userdb database; for simplicitly we will make this a plain text file which is searched linearly. If this gets large it can be converted into an indexed database.

```
# vi /usr/exim/vdomains
flintstone.org

# vi /usr/exim/configure
Change the local domains setting:
domainlist local_domains = @ : lsearch;/usr/exim/vdomains
```

To read `/etc/userdb.dat` directly, we use the "dbmnz" lookup type. It's made a bit awkward because courier uses a vertical bar to separate fields, whereas Exim's "extract" operator expects fields separated by spaces, but we can use the regular expression substitution operator (sg) to convert this into the form we want.

```
[put this immediately after 'begin routers']
userdb:
  driver = accept
  transport = local_delivery_userdb
  domains = lsearch;/usr/exim/vdomains
  address_data = ${lookup{$local_part@$domain}dbmnz{/etc/userdb.dat}\
     {${sg{$value}{([^=]+)=([^|]+)\\|?}{\$1="\$2" }}}fail}
  # note the space between "\$2" and }}}

# If the address_data lookup succeeds, then we'll go to the transport.
# But if the address_data lookup fails, then we fall through to here; all
# remaining addresses in vdomains need to be bounced.

userdb_bounce:
  driver = redirect
  domains = lsearch;/usr/exim/vdomains
  data = :fail:unknown user
  allow_fail
  fail_verify

[put this anywhere after 'begin transports']
local_delivery_userdb:
  driver = appendfile
  directory = ${extract{home}{$address_data}}/${extract{mail}{$address_data}{$value}{Maildir}}/
  maildir_format
  maildir_use_size_file
  delivery_date_add
  envelope_to_add
  return_path_add
  user = exim
  group = exim
  maildir_tag = ,S=$message_size
  quota_size_regex = ,S=(\d+)
  quota = ${if match{${extract{quota}{$address_data}}}{([0-9]+)S}{$1}{}}
  quota_filecount = ${if match{${extract{quota}{$address_data}}}{([0-9]+)C}{$1}{}}
  quota_warn_threshold = 90%
```

Once you've done this, test using

```
# /usr/exim/bin/exim -bt brian@flintstone.org
brian@flintstone.org is undeliverable:
  unknown user
# /usr/exim/bin/exim -bt fred@flintstone.org
fred@flintstone.org
  router = userdb, transport = local_delivery_userdb
# /usr/exim/bin/exim -v fred@flintstone.org
Subject: test

hello
.
LOG: MAIN
  <= root@noc.t1.ws.afnog.org U=root P=local S=302
LOG: MAIN
  => fred <fred@flintstone.org> R=userdb T=local_delivery_userdb
LOG: MAIN
  Completed
```

It's even possible for Exim to send a request to courier's authdaemond process to perform the lookup, which has the advantage that it will work for *any* courier authentication module or combination of modules. However there are similar difficulties with parsing the response properly, and the debug output you get from courier is not as good as Exim produces.

```
[put this immediately after 'begin routers']

# We use manualroute with empty route_data as a dummy router, just to
# set address_data to the value read from the socket.

authdaemon_lookup:
  driver = manualroute
  route_data =
  domains = lsearch;/usr/exim/vdomains
  address_data = ${sg{${readsocket{/usr/lib/courier-imap/var/authdaemon/socket}\
        {PRE . exim $local_part@$domain\n}}}{([^=]+)=([^\n]+)\n}{\$1="\$2" }}
  # note the space between "\$2" and }}

# Next, if the response contains HOME= then we know this address is valid
# and we can send it to the local delivery transport

courier:
  driver = accept
  transport = local_delivery_courier
  domains = lsearch;/usr/exim/vdomains
  condition = ${extract{HOME}{$address_data}{1}{0}}

# Otherwise, the address was bad. If it contains FAIL then it's a permanant
# failure, otherwise it's a temporary failure

courier_bounce:
  driver = redirect
  domains = lsearch;/usr/exim/vdomains
  data = ${if match{$address_data}{FAIL}{:fail:unknown user}fail}
  allow_fail
  fail_verify

[transports]
local_delivery_courier:
  driver = appendfile
  directory = ${extract{HOME}{$address_data}}/${extract{MAILDIR}{$address_data}{$value}{Maildir}}/
  maildir_format
  maildir_use_size_file
  delivery_date_add
  envelope_to_add
  return_path_add
  user = exim
  group = exim
  maildir_tag = ,S=$message_size
  quota_size_regex = ,S=(\d+)
  quota = ${if match{${extract{QUOTA}{$address_data}}}{([0-9]+)S}{$1}{}}
  quota_filecount = ${if match{${extract{QUOTA}{$address_data}}}{([0-9]+)C}{$1}{}}
  quota_warn_threshold = 90%
```

Finally, if you are building a box where you know that *all* the mail will be owned by the 'exim' user, then in fact the daemons no longer need to run with root privileges - they can run as the 'exim' user only, as they never have to change to any other userid. This in theory should make your system more secure.

There are some steps you can take to do this for each daemon:

Exim
: Set the 'deliver_drop_privilege' option in `/usr/exim/configure`

Courier-imap

Edit /usr/lib/courier-imap/etc/pop3d and imapd and add "-user=exim" to couriertcpd options, i.e.

```
...
TCPDOPTS="-nodnslookup -noidentlookup -user=exim"
...
```

Sqwebmail

In principle you can run the sqwebmaild.rc startup script as user 'exim', using su:

```
# su exim -c "/usr/local/share/sqwebmail/libexec/sqwebmaild.rc start"
```

However, for this to work, you first need to sort out some permissions issues: you need to recompile and reinstall sqwebmail with

```
./configure --with-cacheowner=exim
```

and change the entry in /etc/crontab to run cleancache.pl as user 'exim' not user 'bin'. You also need to change ownerships of the configuration files:

```
# cd /usr/local/share/sqwebmail
# chown exim:exim sqwebmaild authdaemonrc
# chown -R exim:exim /usr/local/share/sqwebmail/var
```

and also set the PIDFILE parameter in the sqwebmaild configuration file to point to a writable directory.

```
PIDFILE=/usr/local/share/sqwebmail/var/sqwebmaild.pid
```