Courier worksheet

- 0. Reconfigure exim for Maildir delivery
- 1. Install courier-authlib
- 2. Configure and start courier-authlib
- 3. Test courier-authlib
- 4. Install courier-imap
- 5. Configure and start courier-imap
- 6. Test POP3 and IMAP
- 7. POP3 and IMAP over SSL

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.

From the Courier IMAP web site, “The primary advantage of maildirs is that multiple applications can access the same Maildir simultaneously without requiring any kind of locking whatsoever. Maildir is a faster and more efficient way to store mail. It works particularly well over NFS, which has a long history of locking-related woes.

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.The components we are interested in are the IMAP and POP3 servers.

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

---

0. Reconfigure exim for Maildir local delivery

Edit `/usr/local/etc/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
  #user = $local_part
  mode = 0660
  no_mode_fail_narrower
```

Optionally you could add further parameters to this transport which let you impose quotas on your users, for example to limit all users to 100 megabytes of storage each:

```
maildir_tag = ,S=$message_size
quota_size_regex = ,S=(\d+)
quota = 100M
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.

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

Now test out your new configuration by delivering to some local account on your machine. You may want to do these exercises using one terminal logged in as root and another as a user account. Then you can quickly go back and forth using (for example) ALT-CTRL-F1 and ALT-CTRL-F2. Note that "***localuser***" is the *username* that you are using for testing:

```
$ /usr/local/sbin/exim -bt localuser
localuser@pcnn.e0.ws.afnog.org
  router = localuser, transport = local_delivery
$ /usr/local/sbin/exim localuser
Here is a test
.
$ cd /home/localuser/Maildir
$ ls
cur     new     tmp
$ ls new
102078119.7969.pcnn.e0.ws.afnog.org,S=426
$ cat new/*
Return-path: <root@pcnn.e0.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

pine
Not supported by default, patch available.
http://www.math.washington.edu/~chappa/pine/info/maildir.html

kmail
Has direct access to /var/mail or Maildir directly; can also use POP3/IMAP to retrieve new mail.

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

As with most software packages under FreeBSD, you have a choice of installing directly from source, or using the ports system. If you install from source you have the most control over which version is installed and which compilation options are used. However installing from packages is easier, gives you a record of which files where installed where, and installs the files in the "normal" places you'd expect for a FreeBSD system. In particular, the commands get installed in `/usr/local/bin` and `/usr/local/sbin`, which is already in your $PATH. We will use the ports in this lab.

---

1. Install courier-authlib

The courier packages now share a single authentication library, **courier-authlib**. This package is responsible for looking up usernames and passwords - it can retrieve this information from various locations, including Unix system accounts (authpam), SQL databases (authmysql and authpgsql), LDAP databases (authldap), and local file databases (authuserdb). Having a separate package means that the same authentication configuration can now be shared by both POP3/IMAP and Webmail.

```
# portinstall courier-authlib
When prompted for options on the screen, press the down arrow to highlight the option:
      [X] AUTH_USERDB Userdb support
Press <TAB> to highlight OK, and then <ENTER> to continue.
```

Total compile time on your machines will be between around 10 minutes.

2. Configure and start courier-authlib

courier-authlib runs a pool of authentication daemons which perform the actual work; courier-imap and SquirrelMail communicate with these daemons via a socket. So the next thing we need to do is to start the daemons. First you need to edit `/etc/rc.conf`:

```
# vi /etc/rc.conf
add the following line:
courier_authdaemond_enable="YES"
```

Courier-authlib itself has a single configuration file, `/usr/local/etc/authlib/authdaemonrc`. For the purposes of this exercise, we will turn on authentication debugging. In “vi” a shortcut to find something is to press “/” and the text you wish to locate.

```
# cd /usr/local/etc/authlib
# vi authdaemonrc
change this line:
DEBUG_LOGIN=0
to:
DEBUG_LOGIN=1
```

Continue to edit the **authdaemonrc** file. To save resources, you can also configure the authdaemond process not to try 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. Save the original line so that your the following changes look like this:

```
#authmodulelist="authuserdb authvckpw authpam authldap authmysql authpgsql"
authmodulelist="authpam"
```

Now we are ready to start the authentication daemons:

```
# /usr/local/etc/rc.d/courier-authdaemond.sh start
Starting courier_authdaemond.
# ps auxwww | grep authdaemond
root     36787  0.0  0.2  1220  720  p1  S    10:40AM   0:00.00 /usr/local/sbin/courierlc
-pid=/usr/local/var/spool/authdaemon/pid -start /usr/local/libexec/courier-authlib/authda
root     36788  0.0  0.2  1464  880  p1  S    10:40AM   0:00.00 /usr/local/libexec/courie
root     36789  0.0  0.2  1464  880  p1  S    10:40AM   0:00.00 /usr/local/libexec/courie
root     36790  0.0  0.2  1464  880  p1  S    10:40AM   0:00.00 /usr/local/libexec/courie
root     36791  0.0  0.2  1464  880  p1  S    10:40AM   0:00.00 /usr/local/libexec/courie
root     36792  0.0  0.2  1464  880  p1  S    10:40AM   0:00.00 /usr/local/libexec/courie
root     36793  0.0  0.2  1464  880  p1  S    10:40AM   0:00.00 /usr/local/libexec/courie
```

ps shows one courierlogger process, and six authdaemond processes (one master, five workers). If you didn't see "Starting courier_authdaemond" then you made a typing error.

3. Test courier-authlib

You can test the authentication system by itself; the "authtest" command sends requests down the authentication socket, and displays the responses which come back. Test using any Unix login account which already exists on your system.

```
# rehash                        -- to ensure your shell sees this new command
# authtest inst                 -- find an account called 'brian'
# authtest inst <password>      -- check 'brian' has password 'foo'
# authenumerate                 -- list all accounts
```

Try it also with a non-existent username, and with both the right password and a wrong password for an account, to confirm that passwords are being validated properly.

Because we enabled login debugging, you should find that each authentication request generates detailled information in `/var/log/debug.log` showing how the request is passed to each module in turn. Have a look in this file to confirm:

```
# less /var/log/debug.log
```

Further documentation for courier-authlib can be found on the web at http://www.courier-mta.org/authlib/, and is also installed in:

```
/usr/local/share/doc/courier-authlib/
```

---

4. Install courier-imap

Using ports, building courier-imap is straightforward:

```
# portinstall courier-imap

[When prompted for options on the screen, press <TAB> to highlight OK, and then <ENTER> to continue
```

Compilation will take around 10 minutes on your machines.

5. Configure and start courier-imap

You can choose to run POP3, IMAP, or both. There is a configuration file for each one:

```
/usr/local/etc/courier-imap/pop3d
/usr/local/etc/courier-imap/imapd
```

The default configuration is acceptable in most cases. However for a large server you may wish to increase the maximum number of concurrent connections from the default of 40, if you have fairly powerful hardware:

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

Then, you need to enable the daemon(s) which you wish to run in `/etc/rc.conf`

```
# vi /etc/rc.conf
add the following line(s):
courier_imap_pop3d_enable="YES"
courier_imap_imapd_enable="YES"
```

And then run the startup script(s):

```
# /usr/local/etc/rc.d/courier-imap-pop3d.sh start
Starting courier_imap_pop3d.
# /usr/local/etc/rc.d/courier-imap-imapd.sh start
Starting courier_imap_imapd.
```

6. Test POP3 and IMAP

Test using telnet: POP3 and IMAP are both text-based layer 7 protocols and you can drive them by hand.

```
# telnet localhost 110
Connected to localhost.ws.afnog.org
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.
```

```
# telnet localhost 143
Connected to localhost.ws.afnog.org.
Escape character is '^]'.
* OK [CAPABILITY IMAP4rev1 UIDPLUS CHILDREN NAMESPACE THREAD=ORDEREDSUBJECT
THREAD=REFERENCES SORT QUOTA IDLE ACL ACL2=UNION STARTTLS] Courier-IMAP ready.
Copyright 1998-2005 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
* OK [READ-ONLY] Ok
a logout
* BYE Courier-IMAP server shutting down
a OK LOGOUT completed
Connection closed by foreign host.
```

NOTE: The daemons will fail to login if the mail directory does not exist, although current versions do now provide an error message. Hence you need to have delivered at least one message to the user, to create their mailbox, before they can login (or use the 'maildirmake' command to create it). Look for logging messages in `/var/log/maillog` and `/var/log/debug.log`.

---

7. pop3 and imap over SSL

If you wish, you can choose to allow pop3 over SSL (port 995) and imap over SSL (port 993). The advantage is that, for clients which support it, the traffic is encrypted. In this day and age this is almost a requirement for any sane mail server. Allowing connections to pop or imap over unencrypted ports is no longer a viable solution. 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 `pop3d.cnf` and `imapd.cnf` files contain the
parameters for the Snakeoil certificate. You may edit this for you environment, but note that it is not proper certificate signed by a recognised CA.
Run the following scripts which will generate them for you:

```
# cd /usr/local/etc/courier-imap
# cp pop3d.cnf.dist pop3d.cnf
# cp imapd.cnf.dist imapd.cnf
# mkpop3dcert
```

```
# mkimapdcert
```

Next, enable the SSL daemons in /etc/rc.conf:

```
# vi /etc/rc.conf
courier_imap_pop3d_ssl_enable="YES"       # pop3 over ssl, port 995
courier_imap_imapd_ssl_enable="YES"       # imap over ssl, port 993
```

Then you start the servers:

```
# /usr/local/etc/rc.d/courier-imap-pop3d-ssl.sh start
Starting courier_imap_pop3d_ssl.
# /usr/local/etc/rc.d/courier-imap-imapd-ssl.sh start
Starting courier_imap_imapd_ssl.
```

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               # pop3 over ssl, port 995
# openssl s_client -connect localhost:993               # imap over ssl, port 993
```

See the previous exercise (#6) for the commands to use with each connection.

If you were running the service commercially you might want to consider a certificate signed by a recognised CA, rather than using a self-signed certificate.