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 authlib, 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.

---

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 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/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, pop3/imap and webmail 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.

```
# cd /usr/ports/security/courier-authlib/

# make

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.

# make install
# make clean         (optional step - deletes temporary files in 'work' subdir)
```

Total compile time on your machines will be between 10 and 15 minutes.

2. Configure and start courier-authlib

courier-authlib runs a pool of authentication daemons which perform the actual work; courier-imap and sqwebmail 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.

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

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 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/courierlogger
-pid=/usr/local/var/spool/authdaemon/pid -start /usr/local/libexec/courier-authlib/authdaemond
root     36788  0.0  0.2  1464  880  p1  S    10:40AM   0:00.00 /usr/local/libexec/courier-authl
root     36789  0.0  0.2  1464  880  p1  S    10:40AM   0:00.00 /usr/local/libexec/courier-authl
root     36790  0.0  0.2  1464  880  p1  S    10:40AM   0:00.00 /usr/local/libexec/courier-authl
root     36791  0.0  0.2  1464  880  p1  S    10:40AM   0:00.00 /usr/local/libexec/courier-authl
root     36792  0.0  0.2  1464  880  p1  S    10:40AM   0:00.00 /usr/local/libexec/courier-authl
root     36793  0.0  0.2  1464  880  p1  S    10:40AM   0:00.00 /usr/local/libexec/courier-authl
```

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.

```
# authtest brian              -- find an account called 'brian'
# authtest brian foo          -- 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:

```
# cd /usr/ports/mail/courier-imap
# make
```

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

```
# make install
# make clean          (optional step)
```

Compilation will take 10 to 15 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. 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:993
# openssl s_client -connect localhost:995
```

See the previous exercise 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.

---

8. Install 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.

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, but you should already have Apache from your exercises done on previously:

```
# cd /usr/ports/www/apache13-modssl
# make all install clean
# vi /etc/rc.conf
apache_enable="YES"
# /usr/local/etc/rc.d/apache.sh start
```

Check your Apache install is working by pointing a web browser at http://localhost/

Now install sqwebmail:

```
# cd /usr/ports/mail/sqwebmail
# make WITH_CHARSET=all
```

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

```
# make install
# make clean       (optional step)
```

This will take about 15 minutes to compile on your machines.

The option "WITH_CHARSET=all" allows sqwebmail to view messages in a wide range of character sets. This increases the size of the binary by about one megabyte with the extra translation tables which are included.

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
```

Sqwebmail comes in two parts: a small CGI stub which sends HTTP requests down a socket; and a pool of daemons which perform the actual work. The CGI stub is installed in `/usr/local/www/cgi-bin-dist` by default, and there are some graphics installed in `/usr/local/www/data-dist/sqwebmail/*`. These locations will work for a default Apache install, but if you have changed the normal Apache configuration (e.g. DocumentRoot) then you may need to copy these somewhere else.

9. Configure and start sqwebmail

sqwebmail's main configuration file is `/usr/local/etc/sqwebmail/sqwebmaild` - however you almost certainly don't need to change it.

As usual, you will need to enable the sqwebmail daemon in `/etc/rc.conf`, and then call its startup script.

```
# vi /etc/rc.conf
...
sqwebmaild_enable="YES"
```

```
# /usr/local/etc/rc.d/sqwebmail-sqwebmaild.sh start
Starting sqwebmaild.
```

10. Test sqwebmail

If everything is working correctly, you should be able to point a web browser at http://localhost/cgi-bin/sqwebmail/sqwebmail and be presented with a login screen, where you can enter a username and password and login.

In addition consider connecting securely at this point:

> https://localhost/cgi-bin/sqwebmail/sqwebmail

If this does not work:

- Check your Apache logs - `/var/log/httpd-access.log` and `/var/log/httpd-error.log`
- Check your mail log - `/var/log/maillog`
- Check your debug log - `/var/log/debug.log`

Further documentation for sqwebmail can be found at http://www.courier-mta.org/sqwebmail/ and installed in `/usr/local/share/doc/sqwebmail/`

11. Optional extra exercises

11.1 Give your neighbour a mail account on your system. Let them check that they can collect mail using POP3,

IMAP and Webmail.

11.2 When users send mail via sqwebmail, we would like their IP address and login name to be recorded in one of the Received: headers to provide a security audit. This can be done by modifying the script `'sendit.sh'` which sqwebmail uses to send all outgoing mail.

```
# vi /usr/local/share/sqwebmail/sendit.sh
change:
exec /usr/sbin/sendmail -oi -t $DSN -f "$1"
to:
exec /usr/sbin/sendmail -oi -t -f "$1" -oMr "$SERVER_PROTOCOL" -oMa "$REMOTE_ADDR" -oMt "$2"
# /usr/local/etc/rc.d/sqwebmail-sqwebmaild.sh restart
```

After making this change, compose a mail via the sqwebmail interface. When it is delivered, check the full headers and look for the bottom Received: header which should record the source of the mail. You'll need to click on the mail icon with the magnifying glass to see full headers:

```
Received: from [192.168.0.1] (ident=fred@flintstone.org)
        by billdog.local.linnet.org with HTTP/1.1 (Exim 4.43 (FreeBSD))
        id 1Cln2K-0000Pd-EP
        for wilma@flintstone.org; Tue, 04 Jan 2005 11:39:40 +0000
```

11.3 A number of behaviours of courier-imap and sqwebmail can be changed by means of "account options". These can be set globally, and overridden for individual accounts (although not for Unix system accounts). Try the following:

```
# vi /usr/local/etc/authlib/authdaemonrc
change
DEFAULTOPTIONS=""
to
DEFAULTOPTIONS="wbnodsn=1,wbnochangingfrom=1,disableshared=1"
# /usr/local/etc/rc.d/courier-authdaemond.sh restart
```

You can see account options for an account using "`authtest` *`username`*", and list all accounts together with their options using "`authenumerate -o`"

The available options are:

`disableshared=1`	Disable shared folder functionality (hides the 'key' icon in sqwebmail). Shared folders need additional setting up, and only work for systems with virtual accounts.

`disablepop3=1`
`disableimap=1`
`disablewebmail=1`
	Disable these types of access from this account

`wbnochangingfrom=1`	Webmail: disable ability for users to set the From: header on outgoing mail
`wbnochangepass=1`	Webmail: disable ability for users to change their passwords
`wbusexsender=1`	Webmail: add an X-Sender: header to outgoing mail
`wbnoimages=1`	Webmail: use a text-only interface
`wbnodsn=1`	Webmail: disable the "return receipt" functionality (Exim does not support this so we must disable it anyway)