Chapter 10

The Work of a Courier Administrator

Once authentication is functioning, setting up a POP3/IMAP server is fairly straightforward. Using a few Courier tricks and features will endear you to mail users and make life easier for the administrator.

10.1 Shared Folders

Courier knows two types of shared folders: virtual shared folders and filesystem-based shared folders.

Virtual shared folders allow users to grant other users access to individual IMAP folders through the IMAP protocol. Users do not require shell access to the mail server, as IMAP directly supports shared folders. All well-known clients have this function.

In order to improve performance, Courier requires a configuration file `/etc/courier/shared/index` listing all accounts containing shared IMAP folders; this file requires some configuration. It is possible to automate this using shell scripts. If everything is set up properly, the Courier script `authenumerate` (see section 10.1.1 on page 160) will do the work automatically.

Filesystem-based shared folders use symlinks and a number of "dirty" but efficient tricks to integrate individual IMAP folders or entire maildir structures into multiple accounts. This requires manual adjustments at the filesystem level in order to set the required file permissions.

This type of shared folder is therefore not particularly suitable for use with virtualized accounts or Internet service providers, as these mail users do not usually have shell access. It is better suited to mail servers with a user base of technically experienced individuals who are not afraid of (simple) shell commands, for example, on work servers within small companies. As a quick fix, administrators can use them to implement exceptions, provided they have `root` permissions.

10.1.1 Setting Up Virtual Shared Folders

All good email clients allow mail users to grant access to other mail users' IMAP folders using *access control lists (ACLs)*. Other users are identified by their IMAP login name, which does not have to be identical to their email address.

The `courierimapacl` File

If user `paul@example.com` has granted `tux@example.net` permission to access the folder `Testshare` via his own email client, `paul`'s maildir will contain the file `courierimapacl`:

```
linux:/mail/example.com/paul/Maildir # cd .Testshare
linux:/mail/example.com/paul/Maildir/.Testshare # ls -la
total 24
-rw-r--r--  1 10000 10000    78 18.11.06 11:40:00 PM courierimapacl
drwx------  2 10000 10000  4096 18.10.05 12:56:00 AM courierimapkeywords
-rw-r--r--  1 10000 10000   15 12.01.06 10:39:00 PM courierimapuiddb
drwx------  2 10000 10000 4096 12.01.06 10:39:00 PM cur
-rw-------  1 10000 10000     0 18.10.05 12:56:00 AM maildirfolder
drwx------  2 10000 10000  4096 18.10.05 12:56:00 AM new
drwx------  2 10000 10000  4096 19.11.06 7:48:00 PM tmp
linux:/mail/example.com/paul/Maildir/.Testshare # cat courierimapacl
owner aceilrstwx
administrators aceilrstwx
user=tux@example.net aceilprstwx
```

Table 10.1 shows the *identifiers* that can appear in the first column of this file. By default, two entries for `owner` and `administrators` have to exist in every `courierimapacl` file.

Identifier	Meaning
`owner`	The owner of this folder
`anyone`	Every user
`anonymous`	Every user (identical to `anyone`)
`user=`*`loginid`*	The user *`loginid`*
`group=`*`name`*	All users in group *`name`*
`administrators`	All users in group `administrators` (identical to `group=administrators`)

Table 10.1: Assigning permissions through identifiers in the `courierimapacl` file

The `group` user option (see section 9.12 on page 144) makes it possible to associate individual accounts with groups. In this way, members of a group can be assigned shared rights to folders. This is completely different from the `sharedgroup` user option (see section 9.12 on page 145), which is handled differently.

The third line in the example above assigns `tux@example.net` all possible permissions; Table 2.1 on page 36 shows the meanings of these permissions.

It is possible to remove an individual permission from an identifier (that is, to specify "negative permissions") by prefixing a minus sign; this is useful when generous group or `anyone` permissions allow more permissions to a particular user than desired. Courier evaluates all positive permissions and then subtracts the negative permissions, so the order of these entries within the file is irrelevant.

In the following example, all users receive read and write permissions for the affected directory, but write permission is revoked for `tux@example.net`:

```
linux:/mail/example.com/paul/Maildir/.Testshare # cat courierimapacl
owner aceilrstwx
administrators aceilrstwx
-user=tux@example.net w
anyone=lrw
```

However, Courier now has the problem that the permissions are saved in the IMAP folder, but not in `tux@example.net`'s data. In order to determine the folders that this user can access, Courier would have to search through the maildirs of every user.

The `index` File

This is exactly what Courier does. In order to reduce the work involved, the server processes the `index` file. It checks only the accounts listed in this file to see whether they assign to other users permissions for IMAP folders. The number of directories that need to be searched is thereby drastically reduced.

Courier expects this file as `/etc/courier/shared/index`, or, as for individually compiled Courier installations, `/usr/local/etc/courier/shared/index`. This file contains five fields that must be specified, while a sixth field containing user options is optional:

```
share_name uid gid home_directory maildir options
```

The `index` file is a partial dump of the user database, whether this database is stored in `passwd`, in an SQL database, or in an LDAP directory. Courier currently requires only the fields *`share_name`*, *`home_directory`*, and *`maildir`*. The columns *`uid`* and *`gid`* have to exist but are not evaluated. They set the stage for future extensions.

Every time a user logs in, Courier searches all `courierimapacl` files in the maildir directories mentioned in the `index` file in order to determine the shared folders the user has been given access rights for. Courier uses the *`share_name`* field only as a label that identifies the folders for the other clients. Therefore, the maildir user's actual login ID is not relevant to the contents of this field, although the username of the corresponding account is usually used. The `index` file on the server containing the accounts of `paul@example.com` and `tux@example.com` looks a bit like this:

```
linux:/etc/courier/shared # cat index
tux     1001 500 /home/tux Maildir/
paul    1002 500 /home/paul Maildir/
```

It could also just as well have had the following contents, though `tux` can *not* log on as `cheffe`:

```
linux:/etc/courier/shared # cat index
cheffe  1001 500 /home/tux Maildir/
paul    1002 500 /home/paul Maildir/
```

There are no complications if some of the directories listed in the `index` file do not exist. Courier simply ignores the corresponding entries and does not display those shares. This means that an `index` file can be used simultaneously on more than one server, even though each server contains only some of the maildir directories.

Courier strictly separates the shared folders available to a user from the user's own IMAP folders. Virtual shared folders are not listed in `INBOX`, but instead in the `#shared.`*`name_of_share`* tree.[1] This prevents mix-ups and name conflicts:

```
#shared.cheffe.Testshare
#shared.cheffe.Holiday
```

But be careful: If the user's login name contains a dot or slash, this destroys the hierarchy of the IMAP namespace when used as a label for folder sharing. Thus, an entry such as

```
linux:/etc/courier/shared # cat index
# Caution: This does *NOT* work
paul.meier  1003 500 /home/paul Maildir/
```

would lead to a shared folder like this:

```
#shared.paul.meier.Testshare
```

The `meier` directory would then be a subfolder of `paul` (instead of `paul.meier` being under `#shared`). Courier tries to solve this problem by replacing each dot and slash with a space (which is permissible in folder names):

```
#shared.paul meier.Testshare
```

The names may no longer be unique in this case. It can also happen that Courier is unable to calculate shared lists properly after such replacements. Large parts of the list will no longer be displayed properly.

Thus, although Courier automatically replaces dots and slashes with spaces in the display name, this replacement can be problematic. Even if the accounts offering shared folders are actually called `paul.meier` or `anna.gerber`, for example, it is advisable to avoid using these two characters in the first column of `index`. Remember that when the `index` file is processed, the important thing is the share name, and not the account names, which can still contain dots.

If you use dots and slashes in your usernames (perhaps because you use complete email addresses for login names), it is a good idea to think of a workaround here and make sure that you have clean labels in the `index` file. `paul.meier` could have the display name `paul-meier`, for example. However, blindly using search-and-replace hacks can run the risk of creating labels that are ambiguous.

[1] Filesystem-based shared folders are shown in `shared` and not in `#shared`.

You can also set the IMAP_SHAREDMUNGENAMES parameter to 1 in the configuration file of the imapd. Courier will then replace an invalid . with a \:, and an invalid / with a \;. This may not look nice, but it is a workable solution and therefore worth testing.

Special characters are allowed in IMAP folder names (and in share names). They have to be UTF-8-coded in this file.

Arranging Shared Files

If you are managing a large number of shared folders, the index file containing the shared folders quickly becomes unmanageable, both for the administrator and for the users, to whom the shared folders are displayed in a lengthy list.

In order to avoid this, you can group shared folders and export them into a separate index file. Each of these groups is assigned a special share name of its own that is displayed as an additional hierarchy level.

The main index file index contains the definitions of the group names and references to the index files corresponding to them:

```
groupname * indexfilename
```

The asterisk in the second column is a predefined special character and indicates to Courier that this line is not a group definition itself, but gives the name of the file which includes these group definitions. A split index file can contain both:

```
linux:/etc/courier/shared # cat index
employees       *    index-employees
interns      *    index-interns
freelancers  *    index-freelancers
bueroorga   1000 1000 /home/bueroorga Maildir
```

The syntax for permissions in the subfiles is the same as for standard permissions:

```
linux:/etc/courier/shared # cat index-employees
tux         1000 1000 /home/bueroorga Maildir
paul        1000 1000 /home/paul Maildir
geeko       1000 1000 /home/geeko Maildir
```

The shared IMAP folders then have the following hierarchy:

```
#shared.bueroorga.folder
#shared.employees.tux.folder
#shared.employees.tux.folder
#shared.employees.geeko.folder
```

These group definitions are not the same thing as *shared groups*, which we will discuss later. Using the latter, you can ensure that users are unable to see shared folders belonging to groups that they do not belong to. In the groups mentioned above, users in the `employees` group are able to see the shares of users in group `interns` or `freelancers`. Arranging these groups affects the way shared folders appear in the IMAP namespace and makes it possible to assign ACL permissions to groups (see section 10.1.1 on page 154).

Self-Contained Share Groups

If a user can view all share names and IMAP folders in the `index` file, regardless of permissions assigned by other users in the `courierimapacl` files, this can compromise security. The user would then be able to infer the account names from the share permissions, which in turn means that a complete list of users (customers?) is freely available. In addition, the user may be able to tell which users belong to which groups.

This may be irrelevant in a company that has a company-wide address book, but it can be a violation of security in a large organization or an ISP.

Unfortunately, the Courier programmers are refusing to deal with this problem. They claim that it would negatively affect performance if Courier has to parse all `courierimapacl` files. This is rather lame; after all the alternative is *not* using these shared folders at all. This problem will not be dealt with unless someone else writes the patch. ...

Luckily, there is a solution. It is not perfect, but it works. Introduce separate *shared groups* with their own index files by entering a group assignment in the user options (see section 9.12 on page 145). Users are then only able to view the permissions for their own shared groups (their *universe*). However, they will still be able to view all share names and all IMAP folders within the shared group.

Shared groups have another advantage. If the `index` file is large, Courier has to search through a correspondingly large number of directories. This solution is not suitable for several thousand accounts or for overworked servers with many logins. If each shared group has an index file, Courier has to search through far fewer directories.

If, for example, you are managing the email accounts for the three domains `example.com`, `example.net`, and `example.org`, and these belong to different companies and organizations, you could group all users of one domain into a shared group. To do this, set a user option such as `sharedgroup=example.com` for every one of these users. Each user can only belong to *one* shared group.

When a user logs in, Courier determines the `sharedgroup` for that user.

After authentication, the server searches for shared folders in the specific index file for that shared group, instead of searching the global `index` file. The index file for the shared group has a predefined filename, consisting of `index` and the value of `sharedgroup`. If the `sharedgroup=example.com` option is specified for a user, the file is named `indexexample.com`; for `sharedgroup=developerteam`, the file is named `indexdeveloperteam`.

The index file of the shared group contains the maildir directories and their share names as described in section 10.1.1 on page 156:

```
linux:/etc/courier/shared # cat indexexample.com
info          1000 1000 /mail/example.com/info Maildir/
accounting    1000 1000 /mail/example.com/accounting Maildir/
paul          1000 1000 /mail/example.com/paul.meier Maildir/
geeko         1000 1000 /mail/example.com/geeko Maildir/
```

Every user can only view shared folders for his or her shared group, so it is possible to assign the same share name in different shared groups. The labels `info`, `accounting`, and `paul` are now also permitted for entries in `example.org` accounts:

```
linux:/etc/courier/shared # cat indexexample.org
info          1000 1000 /mail/example.org/info Maildir/
accounting    1000 1000 /mail/example.org/accounting Maildir/
paul         1000 1000 /mail/example.org/paul Maildir/
```

If you do not define the `sharedgroup` option for an account, Courier IMAP will search the global `index` file.

Generating the `index` File Automatically

If you only wish to permit sharing for selected accounts, it makes sense to manage the corresponding `index` files manually. If, however, you wish to permit all or nearly all users to share folders, you can use the `authenumerate` program to generate the `index` file automatically.

`authenumerate` uses the authlib library, which has access to the complete user database, to generate a dump of all user data. You can then redirect this dump to the index file:

```
linux: # authenumerate > /etc/courier/shared/index
```

Unfortunately, almost no documentation exists for this program. It has two call parameters: `-o` tells the program to output the user options for the accounts in the sixth column. This also includes the `sharedgroup`.

`authenumerate -s` lists only those accounts that are permitted to share folders. If the user option `disableshared` is set to `1` for an account, that account is not listed.

Bear in mind that `authenumerate` uses the commands in parameters `MYSQL_ENUMERATE_CLAUSE`, `PGSQL_ENUMERATE_CLAUSE`, and `LDAP_ENUMERATE_FILTER` (see pages 138, 139, and 141) to read out the user data. This command can limit the accounts to be considered or manipulate the data (especially that in the first column).

Automatically Generating Index Files for Shared Groups

If you use shared groups, you require numerous group-specific index files. The `sharedindexsplit` tool can split a global `index` file accordingly. If the index file contains the user options in the sixth column (thanks to `authenumerate -o`), this column will show which user belongs to which shared group. `sharedindexsplit` then automatically prepares a suitable index file for every shared group. You can run the following shell script, for example, as a regular `cron` job:

```
#!/bin/sh
sysconfdir="/etc/courier"
sbindir="/usr/sbin"

# Remove residues from previous run-throughs
rm -rf $sysconfdir/shared.tmp
mkdir $sysconfdir/shared.tmp || exit 1

# Generate temporary index file containing user options
$sbindir/authenumerate -s -o >$sysconfdir/shared.tmp/.tmplist || exit 1

# Split by sharedgroup
$sbindir/sharedindexsplit $sysconfdir/shared.tmp <$sysconfdir/ \
shared.tmp/.tmplist  || exit 1

# Delete temporary file
rm -f $sysconfdir/shared.tmp/.tmplist

# Move the completed files to $sysconf
$sbindir/sharedindexinstall
```

`sharedindexsplit` can also split the shared folders into different index files according to the first *n* characters if you specify this number as the *second* call parameter. It then ignores the `sharedgroup` user option. This is what the shell script would look like:

```
#!/bin/sh
sysconfdir="/etc/courier"
```

```
sbindir="/usr/sbin"

# Remove residues from previous run-throughs
rm -rf $sysconfdir/shared.tmp
mkdir $sysconfdir/shared.tmp || exit 1

# Generate temporary index file containing user options
$sbindir/authenumerate -s >$sysconfdir/shared.tmp/.tmplist || exit 1

# Split by the first character
$sbindir/sharedindexsplit $sysconfdir/shared.tmp 1 <$sysconfdir/ \
shared.tmp/.tmplist  || exit 1

# Delete temporary file
rm -f $sysconfdir/shared.tmp/.tmplist

# Move the completed files to $sysconf
$sbindir/sharedindexinstall
```

The `sharedindexinstall` shell script provided by Courier simply bundles the temporary files and moves them to `/etc/shared`. Make sure that the correct path is specified in `sysconfdir`:[2]

```
linux: # which sharedindexinstall
/usr/sbin/sharedindexinstall
linux: # cat /usr/sbin/sharedindexinstall
#! /bin/sh
# $Id: sharedindexinstall.in,v 1.1 2004/01/11 02:47:33 mrsam Exp $
#
# Copyright 2004 Double Precision, Inc.
# See COPYING for distribution information.
#
# Sample script to safely update shared folder index files.

prefix="/usr"
sysconfdir="/etc/courier"
[...]
```

Subscribing to Shared Folders

Usually IMAP users have to explicitly subscribe to shared folders for their clients to display them (see section 2.2.4 on page 41).

Some email clients, such as KMail (see Figure 10.1), use the IMAP protocol to ask the server for the correct namespace for personal folders (`INBOX.*`) or virtual shared folders (`#shared.*`) in order to display the directories

[2] The `prefix` variable was previously used, but it is superfluous in the version used here (1.1 from January 11, 2004), as it is not used in the script.

properly. In that case, you will find the corresponding settings in the IMAP account management.

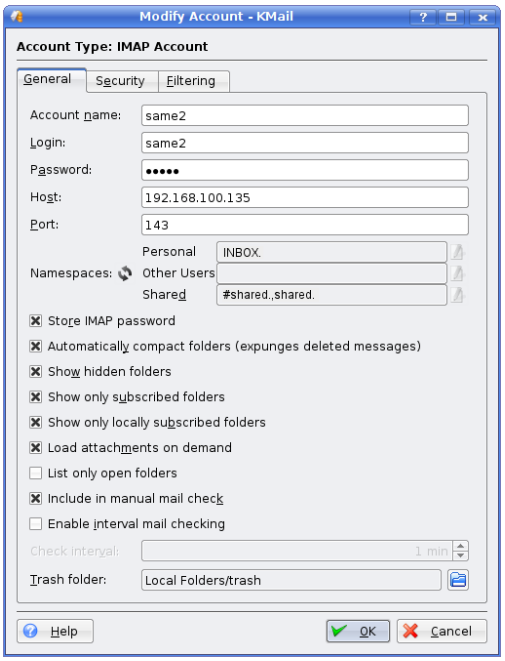

Figure 10.1: KMail automatically queries the IMAP namespace in order to display the folders properly.

10.1.2 Creating Filesystem-Based Shared Folders

If your users are technically experienced and have shell accounts on the server, you can provide filesystem-based shared folders. In this case, the file access permissions in the maildir directories determine each user's access permissions.

If user `tux` wants to prevent user `geeko` from viewing his maildir, `tux` has to ensure that only he as the owner of the maildir has read and write permissions:

```
tux@linux:~$ ls -lad Maildir/
drwx------ 6 tux   users 4096 10. Mar 22:30 Maildir/
```

If he changes the file permissions for his maildir directory or for individual IMAP folders in that directory, other users can access them. You can modify Courier so that it offers the shared maildir directories to other users for subscription via IMAP.

A *shareable maildir* is a special maildir with more relaxed access permissions, allowing other users to view it. For this reason, this should not be a user's actual maildir (even though this is technically possible). Instead, a user should create an additional directory in the personal home directory as a *shared maildir*.

Folders in a shareable maildir are called *shared folders*. Other users can subscribe to these folders.

If `tux` wishes to share a folder with colleagues, he uses `maildirmake` to create a separate shareable maildir with open access permissions, without letting other users access his actual maildir. The `-S` parameter tells the program to generate a shareable maildir:

```
tux@linux:~$ maildirmake -S Maildir-Shared
tux@linux:~$ ls -lad Maildir*
drwx------ 7 tux   users 4096 11. Mar 17:31 Maildir
drwxr-xr-x 9 tux   users 4096 11. Mar 17:25 Maildir-Shared
tux@linux:~$
```

The only difference between the maildirs lies in the file permissions. `tux` has now made the shared folder available, and the rest is up to his colleagues.

They can now create a `shared-maildirs` file in their own maildir. This is where they enter the paths to the other available maildirs belonging to other users:

```
geeko@linux:~$ cd Maildir
geeko@linux:~/Maildir$ cat shared-maildir
tux     /home/tux/Maildir-Shared
paul    /home/paul/Maildir2
group   /home/gruppe/Maildir-groupaccess
```

Nothing else needs to be done at file/operating system level. Courier does the rest of the work when somebody subscribes to a folder using the IMAP protocol. When he next logs in, `geeko` can subscribe to a number of additional folders available in the `shared` namespace.[3] In order to tell them apart, Courier completes the short names of the maildirs in the `shared-maildirs` file:

```
shared.tux.*
shared.paul.*
shared.gruppe.*
```

Courier does all of this in the background using symlinks. To do this, it creates an additional folder named `shared-folders` in `geeko`'s maildir;

[3] Virtual shared folders are available from `#shared`.

unlike geeko's normal IMAP folders, this folder does *not* begin with a point. This shared folder contains the three short names as directories:

```
geeko@linux:~/Maildir$ ls -l
drwx------ 2 geeko users 4096 10. Mar 22:35 courierimapkeywords
-rw-r--r-- 1 geeko users  187 10. Mar 22:19 courierimapuiddb
drwx------ 2 geeko users 4096 10. Mar 22:24 cur
drwx------ 2 geeko users 4096 10. Mar 22:19 new
drwx------ 3 geeko users 4096 11. Mar 17:27 shared-folders
-rw-r--r-- 1 geeko users   27 11. Mar 17:26 shared-maildirs
drwx------ 2 geeko users 4096 11. Mar 19:17 tmp
geeko@linux:~/Maildir$ cat shared-maildir
geeko@linux:~/Maildir/shared-folders$ cd shared-folders
geeko@linux:~/Maildir/shared-folders$ ls -l
drwx------ 6 geeko users 4096 11. Mar 17:27 paul
drwx------ 6 geeko users 4096 11. Mar 17:27 tux
drwx------ 6 geeko users 4096 11. Mar 17:27 group
```

If geeko subscribes to tux's Party folder, Courier creates a maildir named tux/Party. This maildir contains the usual maildir directories cur, new, and tmp, and also a symlink to tux's actual maildir:

```
geeko@linux:~/Maildir/shared-folders$ ls -l tux/Party
total 24
drwx------ 2 geeko users 4096 11. Mar 5:30:00 PM courierimapkeywords
-rw-r--r-- 1 geeko users  234 11. Mar 5:30:00 PM courierimapuiddb
drwx------ 2 geeko users 4096 11. Mar 5:30:00 PM cur
drwx------ 2 geeko users 4096 11. Mar 5:30:00 PM new
lrwxrwxrwx 1 geeko users   32 11. Mar 17:27 shared -> /home/tux/Maildir-
Shared/.Party
-rw------- 1 geeko users    1 11. Mar 17:30 shared-timestamp
drwx------ 2 geeko users 4096 11. Mar 5:30:00 PM tmp
```

Whenever geeko logs in, Courier compares the contents of shared/cur to those of cur; for every file in tux's original directory, it creates a symlink in geeko's cur directory:

```
geeko@linux:~/Maildir/shared-folders$ cd tux/Party
geeko@linux:~/Maildir/shared-folders/tux/Party$ ls -l shared/cur
-rw-r--r-- 1 tux   users  3737 11. Mar 17:30 1173630626.M516761P6734V000
0000000000302I000102C8_6.couriertest,S=3737:2,S
-rw-r--r-- 1 tux   users  3795 11. Mar 17:30 1173630626.M900691P6734V000
0000000000302I000102D0_7.couriertest,S=3795:2,S
-rw-r--r-- 1 tux   users  5052 11. Mar 17:30 1173630627.M359308P6734V000
0000000000302I000102D1_8.couriertest,S=5052:2,S
geeko@linux:~/Maildir/shared-folders/tux/Party$ ls -l cur
total 12
lrwxrwxrwx 1 geeko users 111 11. Mar 17:30 1173630626.M516761P6734V00000
00000000302I000102C8_6.couriertest,S=3737:2, -> /home/tux/Maildir-Shared
```

```
/.groupwrite/cur/1173630626.M516761P6734V000000000000302I000102C8_6.cou
riertest,S=3737:2,S
lrwxrwxrwx 1 geeko users 111 11. Mar 17:30 1173630626.M900691P6734V00000
000000000302I000102D0_7.couriertest,S=3795:2, -> /home/tux/Maildir-Shared
/.groupwrite/cur/1173630626.M900691P6734V000000000000302I000102D0_7.cou
riertest,S=3795:2,S
lrwxrwxrwx 1 geeko users 111 11. Mar 17:30 1173630627.M359308P6734V00000
000000000302I000102D1_8.couriertest,S=5052:2, -> /home/tux/Maildir-Shared
/.groupwrite/cur/1173630627.M359308P6734V000000000000302I000102D1_8.cou
riertest,S=5052:2,S
```

Once this has been done, `geeko` can treat this IMAP folder like a personal inbox, as long as he has the required file permissions. If `tux` grants only read permissions, `geeko` does not have write permissions and may therefore neither delete nor add emails.

Access permissions are defined by the user ID and group ID at system level. The owner of the maildir can only grant read and write permissions for his or her own user group or for all users on the server. It is not possible to grant permissions to individual accounts.

Do not forget that clients may still have to subscribe to these newly available directories.

10.2 Quotas

When asked about the capabilities an IMAP server should have, most administrators name quotas as one of the most important. This is understandable: There is the constant fear that an ever-increasing volume of data will become unmanageable, and too many users treat the technology and infrastructure carelessly. These users often forget that the "couple of GBs" that are unimportant on their personal computer can quickly add up to huge amounts of data on company networks or ISP servers with hundreds or thousands of users; this data is almost unmanageable, and the storage costs in secure environments can be considerable.

On the other hand, it is worth considering whether and how quotas are worth using; sometimes, they do more harm than good. In most cases, the servers have more than enough memory. Quotas are set up to restrict individuals. If everyone *did* use this much space, there would *not* be enough space. This means that emails are preemptively blocked, even though the server could still process them.

Low quota limits also introduce vulnerability to denial-of-service attacks. An account with a restricted quota is flooded on Friday evening, and the problem is only detected on Monday morning, and by then all subsequent legitimate emails (including those from paying customers) have been denied.

As long as the email server has a lot of free memory, lack of quotas is not a problem. Also, with or without quotas, MTAs such as Postfix monitor the free memory and refuse emails before the hard disk is full. Even when quotas have been set, it is still necessary to monitor the server and the free memory: Quotas are usually so generous that the sum of quotas for *all* users exceeds the size of the hard disk, and so offer no protection from filling up server memory.

You can still retain control over disk usage even if you do not use quotas. Simply write a small shell script that determines the inbox size for individual users. All you need to do is enter `du -s *` in the maildir's main directory; `| sort` will show the mailboxes with the largest memory consumption. Now you can give a verbal warning and request the user in person to tidy up the directory, or you can automatically send a warning message to the user. This method is often more successful than locking the account, which can create some bad feelings between you and the user.

Another interesting idea is to block SMTP *sending* if a user exceeds his or her quota. You can use the monitoring scripts mentioned to set up this block. While classic quota bounces affect innocent senders, the SMTP block affects the owner of the account. If a user exceeds the quota, he or she is unable to send out new emails.

Not only is this sanction very effective, it also makes sure that users will notice much more quickly when they exceed quotas. If you use traditional quota bounces, users will often only find out that they have exceeded the quota once the senders of (unreceived) emails complain.

The most sensible course is to combine the two quota sanctions: prevent emails from being sent, and only stop accepting new emails once the next limit is reached. This prevents dead accounts from accumulating emails for years.

In the free email sector, quotas can prevent misuse of mailboxes. It makes sense to balance out the benefits and drawbacks: It can be embarrassing for companies if emails are returned to external senders because the employees' mailboxes are full.

10.2.1 Quotas for Courier

Courier permits quotas on two levels. At file level, the server uses the quota capability of the Linux kernel and requires user accounts with an individual user ID. The advantage here is that quotas have nothing to do with the email system; instead, they are enforced because the email server can no longer save new emails, which in turn causes the required bounce.

The maildir++ format permits quotas that Courier can evaluate. Quota files saved in this format also have to be taken into account by the saving Mail

Delivery Agent (MDA); `maildrop`, `procmail`, `local`, and `deliverquota` do this. There are a few MDAs that require an extra patch for these quotas. One of them is the MDA `virtual`, designed by Postfix for MySQL and LDAP users.[4]

Quotas at File Level

As long as you work with real shell accounts rather than with virtual accounts, you can use standard filesystem quotas. As emails are stored under the user ID of the user, they count towards that user's quota. If the quota is exceeded, the system prevents the mail server from saving additional emails.

The mail server usually views such write errors as temporary errors. This means that emails are not lost, but instead remain in the mail server's mail queue; the mail server continues to attempt to deliver them locally to the hard disk. Emails are only returned to the sender as undeliverable if the maximum queuing time has been exceeded. For Postfix, the default of `maximum_queue_lifetime` is five days.

The procedure for setting up quotas at file level differs slightly for different kernel versions, both regarding the names of the commands and the files to be modified. The following version should apply to 2.4.*x* and 2.6.*x*.

First, you have to use the options `usrquota` and `grpquota` to tell the `mount` command that it should activate quotas for the corresponding partition at the user or group level. To do this, complete the appropriate `/etc/fstab` entry:

```
/dev/sda5  /var/maildir  ext3  defaults,usrquota    1    1
```

Please note that the quotas have to be supported by the filesystem. This is the case for Ext2, Ext3, and ReiserFS. After this, you have to remount the partition:

```
linux: # mount -o remount /var/maildir
```

Next, you generate the quota database again so that existing files on all partitions are included if they were mounted using the option `usrquota` or `grpquota`:

```
linux: # quotacheck -avug
```

The top level of the quota partition (in this case, `/var/maildir`) should contain the files `aquota.user` and/or `aquota.group`. For older kernel versions, these files are named `quota.user` and `quota.group`.

[4] See `http://vda.sourceforge.net`.

Even the best mount options are useless if quotas are deactivated in the kernel. The commands `quotaon` and `quotaoff` switch this function on and off:

```
linux: # quotaon
```

Use the `edquota` command followed by a username to specify the quotas for that user:

```
linux: # edquota geeko
Disk quotas for user geeko (uid 1000):

Filesystem    blocks    soft    hard    inodes    soft    hard
/dev/sda5         74       0       0        23       0       0
```

The second column shows the blocks on this partition currently used by `geeko`. You can not modify this value, as it is determined by the files stored. The next two columns show the soft and hard limits for the user in blocks. The user may briefly exceed the soft limit, but the hard limit is an absolute restriction.

Like free data blocks, the number of available inodes is limited, so you can also specify quotas for inodes. Column five shows the current consumption, and columns six and seven can be edited and show the soft and hard quotas for inodes.

Simply modify the appropriate columns to change the quotas for a user. `0` means that no quotas have been set. The following example sets a soft limit of 5,000KB and a hard limit of 7,500KB, as the default block size is currently 1,024 bytes.

```
Disk quotas for user geeko (uid 1001):
Filesystem    blocks    soft    hard    inodes    soft    hard
/dev/hda5         74    5000    7500        23       0       0
```

Enter `repquota` to read out all quotas:

```
linux: # repquota -a
*** Report for user quotas on device /dev/sda5
Block grace time: 7days; Inode grace time: 7days
                        Block limits                File limits
User            used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root       -- 2443081       0       0          99647     0     0
lp         --      55       0       0             18     0     0
mail       --       1       0       0              1     0     0
news       --       1       0       0              6     0     0
uucp       --       1       0       0              2     0     0
```

```
games       --    6993       0       0            179     0     0
man         --    1302       0       0            999     0     0
at          --       1       0       0              3     0     0
wwwrun      --       1       0       0              1     0     0
postfix     --       2       0       0             39     0     0
ntp         --      17       0       0              5     0     0

mdnsd       --       1       0       0              6     0     0
messagebus  --       1       0       0              1     0     0
haldaemon   --       1       0       0              1     0     0
nobody      --       1       0       0              1     0     0
geeko       --      74    5000    7500             23     0     0
```

If your users have shell access to the mail server, they can use the `quota` command to find out the current status:

```
user@linux:~$ quota
Disk quotas for user peer (uid 1001):
     Filesystem  blocks   quota   limit   grace   files   quota   limit
grace
      /dev/hda5         74    5000    7500        23       0        0
```

Quotas through maildir++

Filesystem quotas are often unnecessary, as the maildir format enhanced by Courier contains its own quotas. In addition to the three subdirectories `cur`, `new`, and `tmp`, which are contained in a generic `maildir` directory created by `maildirmake`, there are some Courier-specific files (maildir++ extensions), some of which are used for quotas.

You can use the `maildirmake` call parameter `-q` to activate quotas in an *existing* maildir directory. You can specify a maximum mailbox size in bytes and/or a maximum number of emails: The command `maildirmake -q 10000000S,1000C` */path/to/*`Maildir` sets a quota of approximately 10MB or 10,000,000 bytes (`S` is short for *size*) and permits up to 1,000 messages (`C` is short for *count*). The quotas are triggered if one of the two limits is exceeded. As usual, Courier manages the settings in small ASCII files.

```
linux:/home/tux # su tux -c "maildirmake Maildir"
linux:/home/tux # ls -la Maildir
total 20
drwx------  5 tux  users  4096 Jul 28 22:11 .
drwxr-xr-x  8 tux  users  4096 Jul 28 22:11 ..
drwx------  2 tux  users  4096 Jul 28 22:11 cur
drwx------  2 tux  users  4096 Jul 28 22:11 new
drwx------  2 tux  users  4096 Jul 28 22:11 tmp
linux:/home/tux # maildirmake -q 10000000S,1000C Maildir
linux:/home/tux # ls -la Maildir
total 24
```

```
drwx------  5 tux  users  4096 Jul 28 10:12:00 PM .
drwxr-xr-x  8 tux  users  4096 Jul 28 22:11 ..
drwx------  2 tux  users  4096 Jul 28 22:11 cur
-rw-r--r--  1 tux  users    36 Jul 28 22:12 maildirsize
drwx------  2 tux  users  4096 Jul 28 22:11 new
drwx------  2 tux  users  4096 Jul 28 10:12:00 PM tmp
```

You can also create the `maildirsize` file manually for empty maildir directories and edit it at a later stage to change the quota size:

```
linux:/home/tux # cat Maildir/maildirsize
10000000S.1000C
           0            0
```

The `maildirsize` file will look different if the maildir contained emails when the quotas were activated. In this case, `maildirmake` measures the occupied memory and logs this information in the `maildirsize` file.

When new messages are written to the maildir directory, the `maildirsize` file keeps a quota log: The first column contains the changes in occupied memory, and the second column contains the number of new and deleted messages. Courier does not add up the existing values; instead, each software component adds one log line showing how much space and how many emails it has used. Negative values signify that a message has been deleted and the corresponding amount of storage volume has been freed up.

If `tux` has received a few messages in the meantime, the `maildirsize` file would look like this:

```
linux:/home/tux # cat Maildir/maildirsize
10000000S.1000C
           0            0
         523            1
       37909            1
        2039            1
       12976            1
       -2039           -1
```

If a software component wants to check a user's quotas, it reads the valid quota settings from the first line and adds up the values from all other lines. The result shows the occupied memory and the number of files used.

This list naturally increases over time, which means that using this method to calculate quotas would take far too much time once several thousand messages have been received. For this reason, Courier runs through all maildirs from time to time (usually every 15 minutes) and recalculates the `maildirsize` file based on the stored emails. Here is the result after such a file cleanup, when the maildir showed 51,408 bytes in 3 files for `tux`:

```
linux:/home/tux # cat Maildir/maildirsize
10000000S.1000C
       51408            3
```

This quota monitoring method assumes that software components with write access to the maildir understand the `maildirsize` file, process it, and log new emails in it. This means that mail software not only has to be compatible with maildir (which is standard), but also with the maildir++ extensions.

Naturally, Courier is capable of this, but the IMAP server is not the only server with write access to maildirs. The MTA also saves new emails, and if you use shell accounts, even the local email program may access the maildir directly instead of via IMAP—`pine`, KMail, and other clients can do this. If the `maildirsize` file is not processed during these interactions, quotas can be exceeded without attracting notice.

In principle, this is not a problem, as Courier automatically cleans up the `maildirsize` file, recalculates the quotas using all emails, and collects all the correct data.

One problem remains: The IMAP server is not the most important component for quota monitoring. It can be used to upload emails that increase the data volume, but IMAP sessions usually ensure that messages are deleted and memory is freed up.

The MTA (Postfix, QMail, Exim) is far more important. After all, when quotas are exceeded it is the MTA that has to refuse emails for the account. For this reason, the MTA must recognize and evaluate `maildirsize`, or else the quotas will not be effective. Courier does calculate everything, but who will activate the emergency brake?

10.2.2 Quotas and the MDA

Nearly all MTAs contain their own separate programs that save emails in the filesystem. These programs are Mail Delivery Agents (MDAs). Postfix preferentially uses `local` or `virtual`; the best-known free MDAs are `procmail` (popular for its filters) and `maildrop` (the Courier project MDA). These MDAs are interchangeable.

Not all MDAs support quotas, as this function is not part of the original maildir definition, but is part of the extended maildir++. The free MDAs `local`, `procmail`, and `maildrop` are able to use quotas, whereas `virtual` requires the VDA patch (discussed in the next section).

Check whether the MDA on your MTA can use maildir++ quotas before you use them. You may have to replace the MDA. It is not really important which program saves the file onto the hard disk in the maildir storage format.

Adding Maildir Quotas to `virtual`

Conflicts arise if you need the special capabilities of your original MDA. The Postfix MDA `virtual`, for example, allows user data including maildir paths to be stored in MySQL databases or LDAP directories. `virtual`, however, is not able to use quotas, and often it cannot be replaced with `maildrop`.

Now there is a patch for `virtual`, the VDA patch.[5] You will, however, have to compile Postfix and the `virtual` module yourself. If you do not want to build your own Postfix, you can add a patched `virtual` program to your production system. You can compile this program elsewhere and operate it in *parallel* to the unpatched version in the distribution package, so as not to interfere with the update mechanisms of your distribution. Name the patched version `virtual-quota`, for example, and copy it to the Postfix modules (usually to `/usr/lib/postfix`). Now change the module call in `/etc/postfix/master.cf` by replacing the name of the MDA in the line

```
virtual   unix  -       n       n       -       -       virtual
```

with this:

```
virtual   unix  -       n       n       -       -       virtual-quotas
```

The `deliverquota` MDA

The `deliverquota` MDA from the Courier project naturally supports maildir quotas. It expects the email at the standard input; for the call parameter, it requires the absolute or relative path to the directory where the email is to be saved:

```
linux: # cat mail_file | deliverquota /home/tux/Maildir
```

If `deliverquota` serves as an auxiliary program for the MTA, it does not make sense to specify an absolute path to the maildir where the email will be saved, as emails could then only be delivered to one single inbox. Check instead which variable(s) your MTA offers, or use a tilde as placeholder for the home directory.

If you want to have quotas set when the maildir is accessed, `deliverquota` can carry out this task. Simply enter the required quota definition as the last parameter after the maildir path:

```
linux: # deliverquota -c -w 90 ~/Maildir 10000000S,1000C
```

As shown here, the program knows the following call options:

[5] See `http://vda.sourceforge.net/`.

- If you add `c` (*create*), it will automatically generate any missing maildir directories, and it will even generate parent directories if necessary. This makes a call to `maildirmake` for new users unnecessary, as the first email received will trigger the creation of the maildir.
- `-w` *`percent`* will tell `deliverquota` to deliver a quota warning to a user's mailbox as soon as that user has used up more than *`percent`* of his or her quota limit. You can store this warning as a complete RFC-822 email in `/etc/courier/quotawarnmsg`, including the mail header and the body. Courier will only update the message ID and the date:

```
linux: # cat /etc/courier/quotawarnmsg
From: Postmaster <postmaster@tux.local>
Reply-To: support@tux.local
To: You;
Subject: Warning: Email quotas exceeded!
Mime-Version: 1.0
Content-Type: text/plain; charset=iso-8859-1
Content-Transfer-Encoding: 7bit

Dear user,

Your mailbox is more than 90% full. Please clean up your mailbox, or
you may be unable to receive new messages.
```

  Bear in mind that only 7-bit characters are permitted in the mail header. This means that special characters in the `subject` line are invalid unless they are coded.

`deliverquota` has no interest in your user database; it does not search for the path to the maildir of the recipient, but instead insists on correct call parameters. This has the advantage that `deliverquota` is fairly easy to use as an MDA on different systems, as almost every MTA contains a part in its configuration specifying which program is to save the email. For Postfix, this information is located in the `mailbox_command` parameter in `main.cf`. This parameter is usually empty:

```
linux: # postconf mailbox_command
mailbox_command =
```

At this point we could use `deliverquota` and the call parameters:

```
linux: # postconf -e "mailbox_command = deliverquota -c -w 90 ~/Maildir"
```

As Postfix accepts the user ID corresponding to the mailbox before saving the emails, we can use the tilde (`~`) to refer to the home directory and thereby specify the maildir path.

Now use a few short test emails to check whether `deliverquota` and your MTA work together properly. Check the following things:

- New emails are recorded in `maildirsize`
- The length of new emails is already coded in their filename by parameter `S=`

10.3 Building an IMAP Proxy with Courier

Section 3.2 on page 50 discussed whether IMAP proxies are suitable and when. Courier IMAP users have the advantage that they can easily reconfigure their existing IMAP server into a proxy, as all essential questions such as authentication or access to the email storage have already been resolved.

In order to decide where to transfer the client's IMAP connection, Courier evaluates the `mailhost` attribute (see section 9.12 on page 144) when a user logs in. This attribute has to contain the name of the IMAP server that physically contains the IMAP mailbox.

Three parameters are required in the `/etc/courier/imapd` file for the proxy setup:

`IMAP_PROXY=1` or `POP3_PROXY=1`
: These parameters activate the proxy function. Courier searches for the `mailhost` attribute. `IMAP_PROXY=0`, and `POP3_PROXY=0` deactivate the proxy function; the `mailhost` attribute may exist, but it is ignored.

`PROXY_HOSTNAME=`*`hostname`*
: If Courier IMAP finds its own hostname in the `mailhost` attribute, it may not transfer the connection (to itself), as this would cause an endless loop. In this case, Courier functions as a normal IMAP server and accesses its local filesystem.

  Courier uses the `gethostname()` function to determine its own hostname; `uname -n` will also return this hostname.

  If you prefer to specify the name of the IMAP server manually, you can use `PROXY_HOSTNAME` to do this. This is necessary if the Unix hostname differs from the name stored in the LDAP and Courier is unable to recognize itself.

`IMAP_PROXY_FOREIGN=0`
: If Courier transfers the IMAP connection to another Courier instance, this setting specifies that Courier does not need to use the IMAP command `CAPABILITY` to determine the capabilities of the other IMAP server, as it knows its own capabilities. If Courier is to transfer the IMAP connection to some other IMAP software, you should set this parameter to `1`.

10.4 Push Instead of Pull: The IDLE Command

It is a burden for the mail server if a connected email client searches all IMAP folders for new messages every few minutes. As a large number of clients are constantly connected to the server, this creates constant background activity, and the searches through email folders quickly turn into a basic I/O burden on the hard disk.

The IDLE command in the IMAP protocol deals with this problem: The server actively informs the client of changes in the email directories. Unfortunately, not all email clients support it, even though it has a number of benefits:

- The client is informed immediately when a new message is received, instead of finding it during the next routine check made every few minutes.
- It is less work for the server to monitor file changes than for it to let the email client carry out regular searches.
- There is less data traffic (this is why IMAP clients for cell phones in particular support the IDLE command).

As new emails are written to the maildir directories by the MTAs/MDAs, Courier is not immediately aware of new emails. For this purpose it can use a *file alteration monitor* such as *FAM*.[6] A prerequisite is that Courier was linked to the FAM during compilation. This is the case for SuSE, for example; SuSE also installs and starts FAM automatically.

Gamin is a second project that claims to be better than the traditional FAM.[7]

Red Hat has already replaced FAM with Gamin. In principle, Courier should work with both tools, as they use identical APIs, according to the Gamin programmers. FAM and Gamin run in the background as daemons; other programs can register directories and files with them, and they will then monitor them for changes and signal when changes occur.

If your distribution allows FAM/Gamin to be used for Courier, you can activate the IDLE command in the IMAP_ENHANCEDIDLE=1 configuration in the imapd file. You should check the following three items:

- Courier has to announce during IMAP login that it supports IDLE. This *capability* should be entered in /etc/courier/imap:

```
IMAP_CAPABILITY="IMAP4rev1 UIDPLUS CHILDREN NAMESPACE THREAD=ORDEREDSU
BJECT THREAD=REFERENCES SORT QUOTA IDLE AUTH=CRAM-MD5 AUTH=CRAM-SHA1"
```

[6] See http://oss.sgi.com/projects/fam/.
[7] See http://www.gnome.org/~veillard/gamin.

- The famd or gam_server needs to have been started and must be visible in the process list:

```
linux: # ps ax | grep fam
 2869 ?        Ss    45:21 /usr/sbin/famd -t 240 -T 0 -l -L
```

- You should also check for communication problems between Courier and FAM/Gamin; they will be visible from the following type of entry in your mail log file:

```
May 28 17:43:10 kjidder couriertcpd: Error: Input/output error
May 28 5:43:11 PM kjidder couriertcpd: Check for proper operation and
configuration
May 28 5:43:11 PM kjidder couriertcpd: of the File Access Monitor daem
on (famd).
```

If you do not have performance problems on your IMAP server, you will not usually need to make any further settings. However, on high-load servers it is advisable to configure FAM to use fewer resources. The following famd call parameters are helpful here:

- The FAM daemon terminates by default after five seconds if no client is connected to it. This is not suitable when using it with IMAP servers. Specify -T 0; this means that FAM will terminate after 0 seconds (i. e., never).

- If FAM has to monitor an NFS filesystem, it can *not* receive information on file changes from the local Linux kernel. Instead, it uses RPC to connect to the FAM daemon of the NFS server, which then monitors and transfers any changes locally.

  If FAM has not been activated there, the local FAM will check for changes via NFS every few seconds. You can specify this interval using -t *secs*. The default value is 6 seconds, which can result in heavy background activity on busy NFS servers. Usually this parameter is irrelevant, as an FAM daemon typically operates on the NFS server. If you also specify the option -l, FAM no longer runs queries via NFS and therefore silently stops operating when the FAM daemon on the NFS server is not running.

- -L ensures that FAM accepts only local client queries. This is suitable for the IMAP server, but you should never set -L on the NFS server.

You will usually find these configurations in /etc/famd.conf. The idle_timeout configuration parameter corresponds to -T, nfs_polling_interval fulfils the same function as -t, no_polling corresponds to -l, and local_only is identical to -L.

Be careful if you are using SuSE: This distribution contains a /etc/famd.conf, but any entries you make have no effect (at least for OpenSuSE 10.2).

Instead, the call parameters are generated from `/etc/sysconfig/fam`, for some strange reason.

10.5 Sending Emails via the IMAP Server

Did you know that IMAP servers can send emails? This function is deactivated by default in Courier. To activate it, simply define a specific outbox folder in `/etc/courier/imapd`:

```
OUTBOX=.Outbox
```

This corresponds to `INBOX.Outbox`; that is, to a subfolder under the inbox, as Courier does not permit directories parallel to the `INBOX` (see section 8.1 on page 110). You should probably use a name other than `Outbox`, as even users unaware of the special meaning of this directory could set up an `outbox` folder, which would then behave in an unexpected manner.

Once you have defined the `OUTBOX` variable, Courier IMAP uses SMTP to send all emails saved in this directory by the client. Courier can add a special X header to specify from which account the email was sent:

```
HEADERFROM=X-IMAP-Sender
```

This results in the following header entry in the sent email:

```
X-IMAP-Sender: tux@linux.local
```

This prevents emails from being sent from an undetectable sender, as the IMAP login name of the sender is added to each outgoing email. You have to judge for yourself whether this is desirable.

For each email in the outbox folder, Courier calls `/usr/bin/sendmail` and transfers the entire email to it. This `sendmail` program contains all well-known MTAs—not only Sendmail, but also Postfix, QMail, and Exim. For this reason, the sending of emails is often controlled from PHP on web servers.

The tool reads the sender and the recipient from the mail header; then the local MTA (installed on the IMAP server) sends the email in the usual fashion.

By default you can only ever send one single email; this is to prevent ricochets when a copying action goes wrong. Imagine if you accidentally moved 20,000 messages from your trash folder to the outbox... You can switch off this safety feature if you add the following line to `imapd`:

```
OUTBOX_MULTIPLE_SEND=1
```

This method of sending emails from the IMAP server sounds unusual; indeed, we have only ever used it for test purposes. However, it is not uninteresting. More and more providers, WLAN hotspots, and universities are locking the SMTP port for sending emails. The IMAP outbox solves this problem, as IMAP connections are frequently possible.

In order to use the folder, you should configure the email client so that it temporarily saves emails in the outbox instead of sending them immediately via SMTP. At the same time, you set up a correctly named outbox folder in your IMAP account.

If you can configure your email client to use this new outbox folder as the outbox, the rest will happen automatically. If this does not work, you unfortunately have to drag and drop the messages you wish to send into the outbox folder.