Internet Mail
and
The Exim Mail Transfer Agent

Philip Hazel

University of Cambridge Computing Service

Main topics

- How Internet mail and SMTP work
- Problems of large installations
- Overview of Exim
- Some specific Exim examples
- Building and installing Exim
- Now try it out! Practical work...

Mail agents

- MUA = Mail User Agent
- Interacts directly with the end user

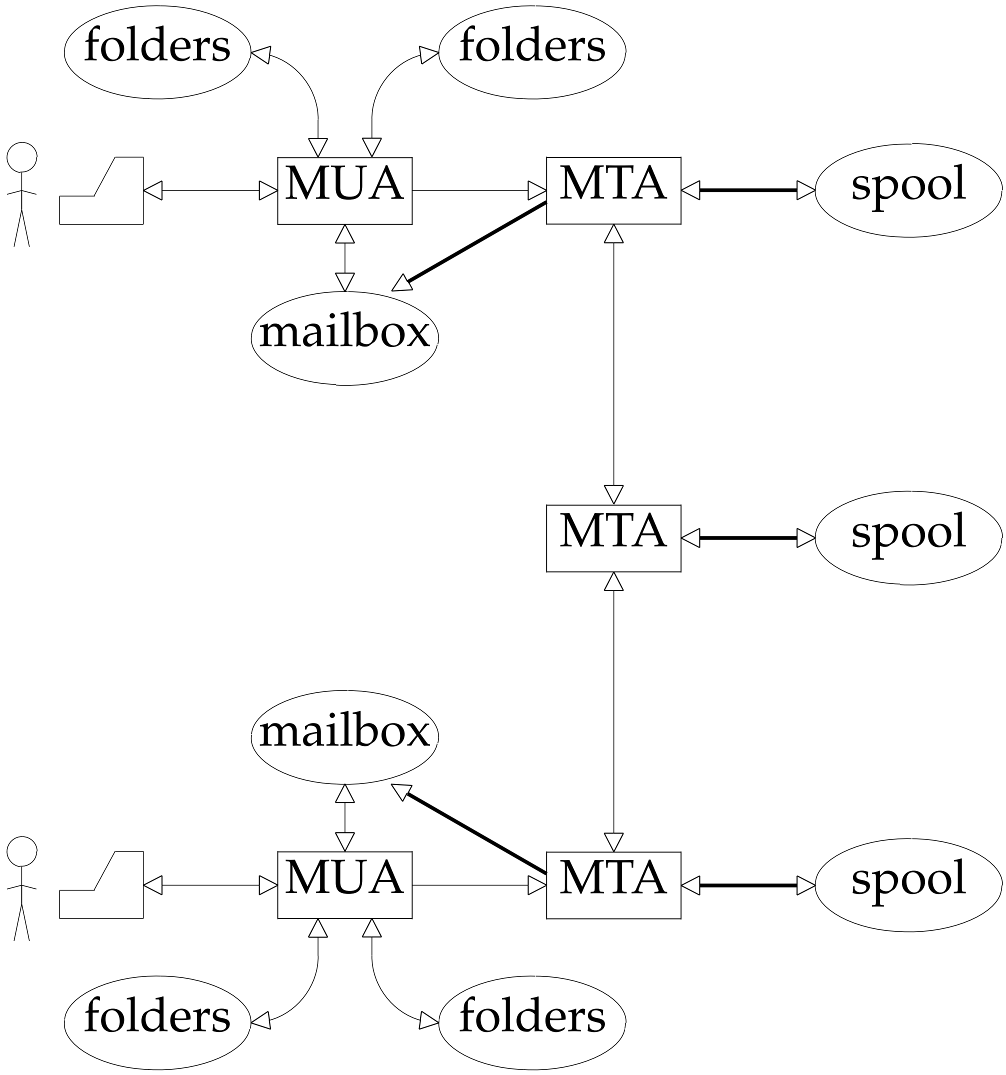
  Pine, MH, Elm, mail, Eudora, Marcel,
  Mailstrom, Mulberry, Pegasus, Simeon,
  Netscape, ...
- Multiple MUAs on one system – end user choice

- MTA = Mail Transfer Agent
- Receives and delivers messages

  Sendmail, Smail, PP, MMDF, Charon,
  Exim, qmail, Postfix, ...
- One MTA per system – sysadmin choice

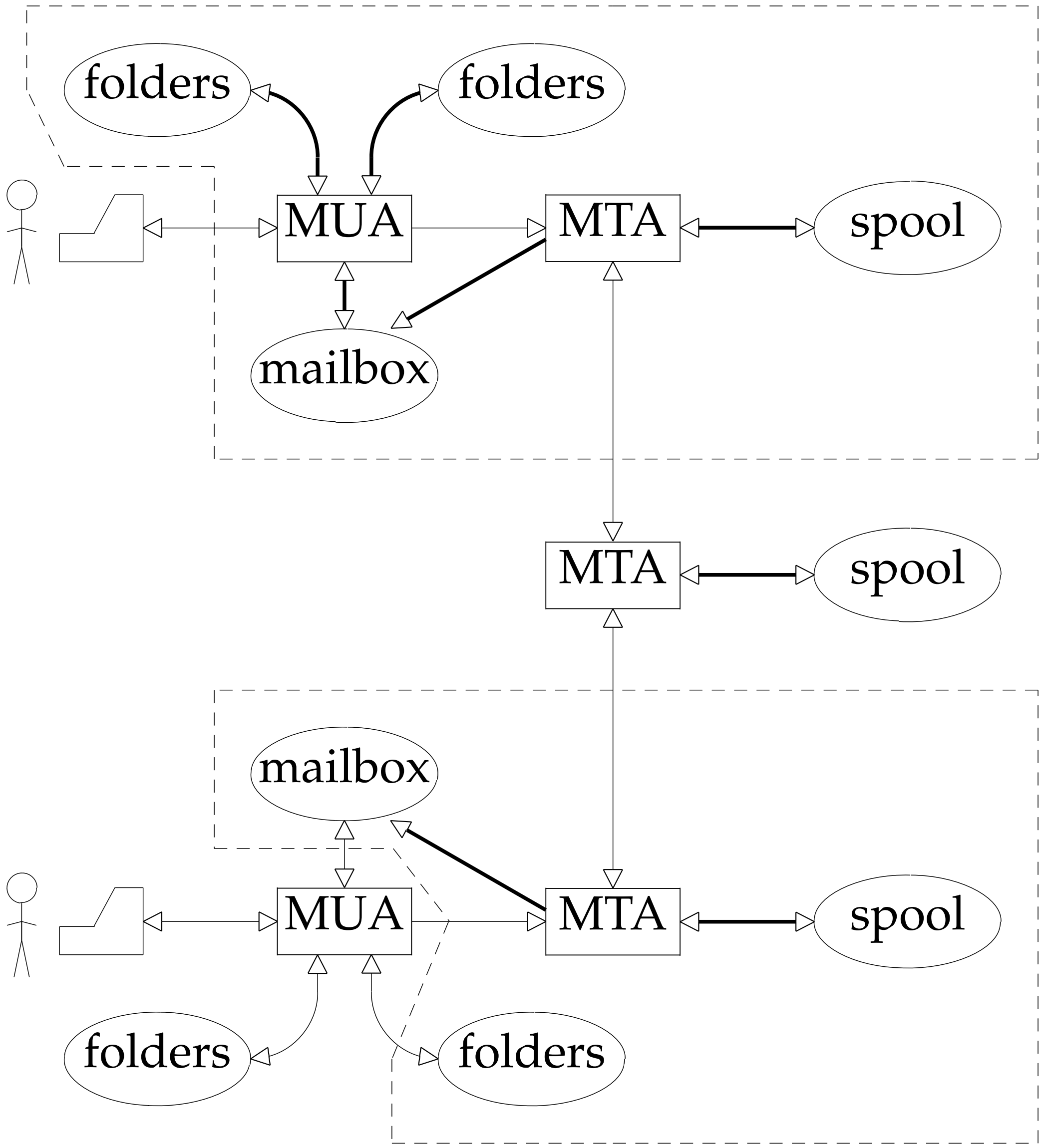
```
From: Philip Hazel <ph10@cus.cam.ac.uk>
To: Julius Caesar <julius@ancient-rome.net>
cc: Mark Anthony <MarkA@cleo.co.uk>
Subject: How Internet mail works

Julius,
  I'm going to be running a course on ...
```

- Message format is defined by RFC 822
- Message consists of
  Header lines
  A blank line
  Body lines
- A basic body is unstructured
- Later RFCs (MIME, 2045) add additional headers which define structure for the body.
- This supports attachments of various kinds and in various encodings


folders
folders
MUA
MTA
spool
mailbox
MTA
spool
mailbox
MUA
MTA
spool
folders
folders

How a typical message is transferred

- User composes message using MUA
- MUA passes message to MTA
  Usually MUA keeps a copy
- MTA stores message on its spool
- MTA transfers message to another MTA
  SMTP (RFC 821) over TCP/IP is used
- Not necessarily the final MTA
  Firewalls
  Corporate mail hubs
  Backup MTAs
- Message reaches final MTA
- Final MTA puts it in a user mailbox
- Recipient MUA displays it to user


folders
folders
MUA
MTA
spool
mailbox
MTA
spool
mailbox
MUA
MTA
spool
folders
folders

MUA Protocols

- Embedded MUA just reads files to access
  mailboxes
- Embedded MUA uses inter-process call
  to send to MTA – may use file, pipe, or
  internal SMTP over a pipe.
- MTA knows the identity of the user
  Normally inserts **Sender:** header if this differs
  from **From:**
- Freestanding MUA uses IMAP/POP to read mail
- Freestanding MUA uses SMTP to send mail
  MTA cannot easily distinguish local/remote
  No authentication
  MUA can be pointed at any MTA whatsoever
  Need for network blocking

```
MAIL FROM:<ph10@cus.cam.ac.uk>
RCPT TO:<julius@ancient-rome.net>
```

```
Received: from taurus.cus.cam.ac.uk ([131.111.8.48]
        ident=cusexim)
        by mauve.csi.cam.ac.uk with esmtp (Exim 2.05 #3)
        id 101qxX-0001lX-00;
        Sun, 17 Jan 1999 11:50:39 +0000
Received: from ph10 (helo=localhost)
        by taurus.cus.cam.ac.uk with local-smtp
        (Exim 2.10 #5) id 101qin-0005PB-00;
        Sun, 17 Jan 1999 11:50:25 +0000
From: Philip Hazel <ph10@cus.cam.ac.uk>
To: Julius Caesar <julius@ancient-rome.net>
cc: Mark Anthony <MarkA@cleo.co.uk>
Subject: How Internet mail works
Date: Sun, 17 Jan 1999 11:29:24 +0000 (BST)
Message-ID: <Pine.SOL.3.96.990117111343.19032A-100000@
  taurus.cus.cam.ac.uk>
MIME-Version: 1.0
Content-Type: TEXT/PLAIN; charset=US-ASCII

Julius,
  I'm going to be running a course on ...
```

- Envelope (RFC 821) fields may differ from header (RFC 822) fields
- MTAs are (mainly) concerned with envelopes
- Error ('bounce') messages have null senders

An SMTP session

```
telnet relay.ancient-rome.net 25
220 relay.ancient-rome.net ESMTP Exim ...
EHLO taurus.cus.cam.ac.uk
250-relay.ancient-rome.net ...
250-SIZE 10485760
250-PIPELINING
250 HELP
MAIL FROM:<ph10@cam.ac.uk>
250 <ph10@cam.ac.uk> is syntactically correct
RCPT TO:<julius@ancient-rome.net>
250 <julius@ancient-rome.net> verified
DATA
354 Enter message, ending with "." ...
Received: ...
From: ...
To: ...
etc...
.
250 OK id=10sPdR-00034H-00
QUIT
221 relay.ancient-rome.net closing connection
```

2*xx* OK
3*xx* send more data
4*xx* temporary failure
5*xx* permanent failure

Email forgery

- It is trivial to forge unencrypted email
- Encryption protects only the body
- This is an inevitable consequence when the sender and recipient are independent
- It is less trivial to forge really well!
- Unsolicited junk ('spam') usually contains some forged headers
- Be alert for forgery when investigating

Use of the DNS for email

- Two DNS record types are used for routing mail
- Mail Exchange (MX) records map mail domains to host names, and provide a list of hosts with preferences:

```
hermes.cam.ac.uk
      MX  5   green.csi.cam.ac.uk
      MX  7   ppsw3.cam.ac.uk
      MX  7   ppsw4.cam.ac.uk
```

- Address (A) records map host names to IP addresses:

```
green.csi.cam.ac.uk  A  131.111.8.57
ppsw3.csi.cam.ac.uk  A  131.111.8.38
ppsw4.csi.cam.ac.uk  A  131.111.8.44
```

- Backwards compatibility rule:

  If no MX records found, look for an A record, and if found, treat it as MX with 0 preference.

- MX records were invented for gateways, but are heavily used for generic mail addresses.

DNS mysteries

- Sometimes primary and secondary nameservers get out of step
- When mystified, check for server disagreement

```
host -t ns ioe.ac.uk
ioe.ac.uk      NS      mentor.ioe.ac.uk
ioe.ac.uk      NS      ns0.ja.net

host mentor.ioe.ac.uk mentor.ioe.ac.uk
mentor.ioe.ac.uk        A       144.82.31.3

host mentor.ioe.ac.uk ns0.ja.net
mentor.ioe.ac.uk has no A record at ns0.ja.net
  (Authoritative answer)
```

Common DNS errors

- MX records point to aliases instead of canonical names; this should work, but is inefficient
- MX records point to non-existent hosts
- MX records contain an IP address instead of a host name
- MX records do not contain preference values
- Some broken nameservers give a server error when asked for a non-existent MX record

Routing a message

- Process local addresses
  Alias lists
  Forwarding files
- Recognize special remote addresses
  e.g. local client hosts
- Look up MX records for remote addresses
- If self in list, ignore all MX >= self
- For each MX record, get IP address(es)
- Perform local delivery
- Try to send to each remote host until one succeeds
- If all fail, try again later if not a hard failure
- Time out after deferring too many times
- Addresses are sorted to avoid sending multiple copies

Checking incoming mail (1)

Recipients

- Some MTAs check local recipients during
  the SMTP transaction
  ==> error detected by *sending* MTA
- Some accept message without checking, and
  look at all recipients later
  ==> error detected by *receiving* MTA

Senders

- Not all MTAs check senders, which is a pity
- The result is lots of crud on the net
  (a) Mis-configured mailers
  (b) Un-registered domains
  (c) Mis-configured nameservers
  (d) Forgers
- Sometimes bad envelope with good address in
  the **From:** header inside
  Common when coming via other mail systems

Checking incoming mail (2)

Policy controls

- Block known miscreant hosts and networks

  Realtime Blocking List (RBL)
  http://maps.vix.com/rbl

  Dial-up List (DUL)
  http://maps.vix.com/dul

  Open Relay Behaviour-modification System
  http://www.orbs.org

- Block known miscreant senders
- Relay control
- Refuse malformed messages
- Recognize junk mail
  Discard, or
  Annotate

Problems of large installations

- Linear password file – search is inefficient

  FreeBSD has **passwd.db**
  Or use NIS or other DB – must be efficient
  POP can be expensive on password lookups
  IMAP is much less expensive
  With non-login users, still need searchable list

- Too many mailbox files in one directory

  Linux degrades at around 1000 (default fs)
  Solaris degrades at around 10,000
  FreeBSD degrades at ?
  Split into multiple directory levels
  e.g. **/var/mail/9/78/username**

- Simultaneous deliveries to same mailbox

  Use maildir format (separate file per message)

- Nameserver delays

  Ensure local (on LAN) nameserver
  Plenty of memory

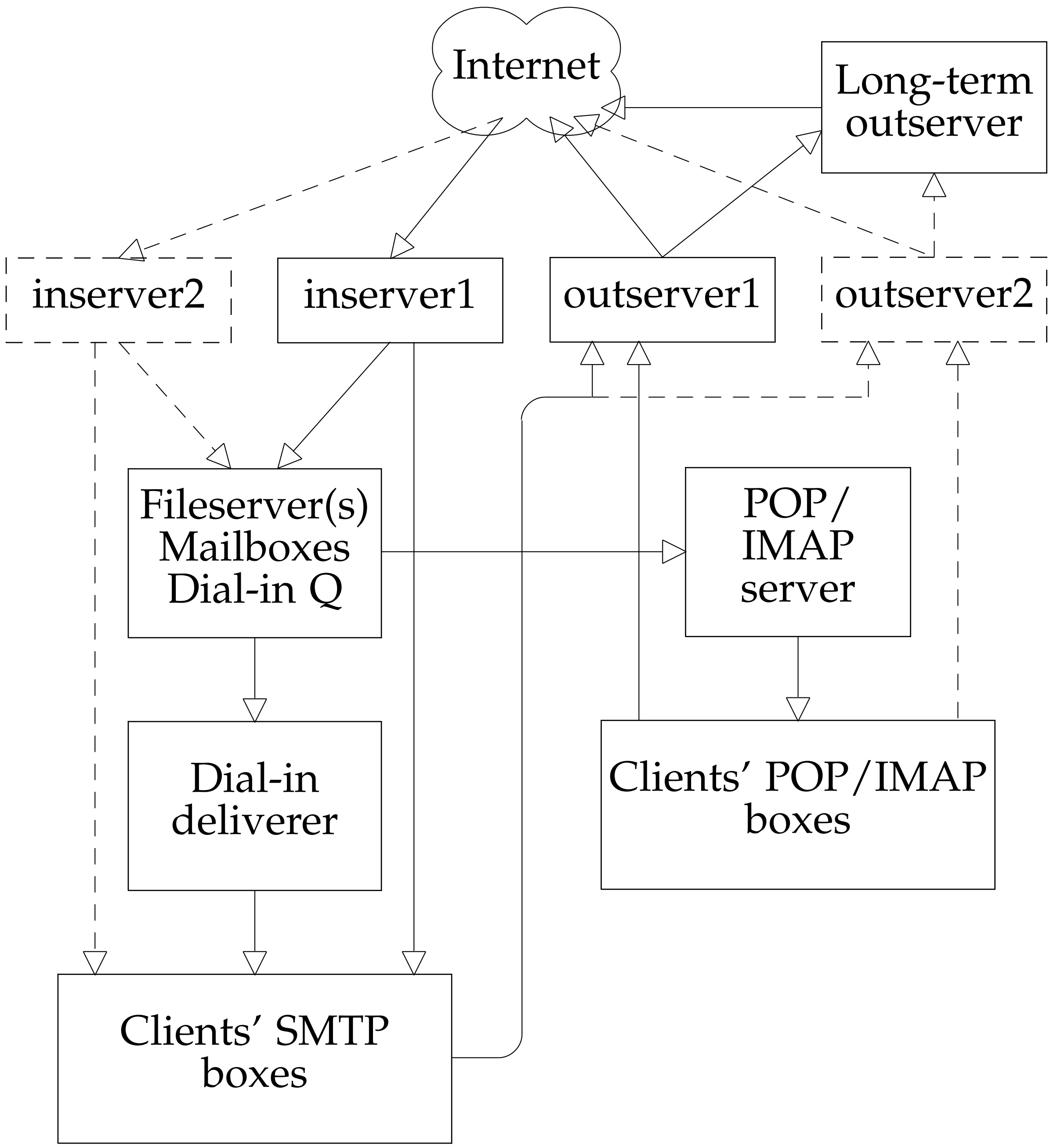
- Messages waiting for dial-up hosts

  Get MTA to deliver into local files
  Control of files by space and time
  Use other program for ultimate delivery
  Can be kicked off by `ETRN` command

- Exim is ultimately limited by disc I/O

  High performance discs
  Expand 'sideways' to multiple parallel servers
  Separate incoming and outgoing

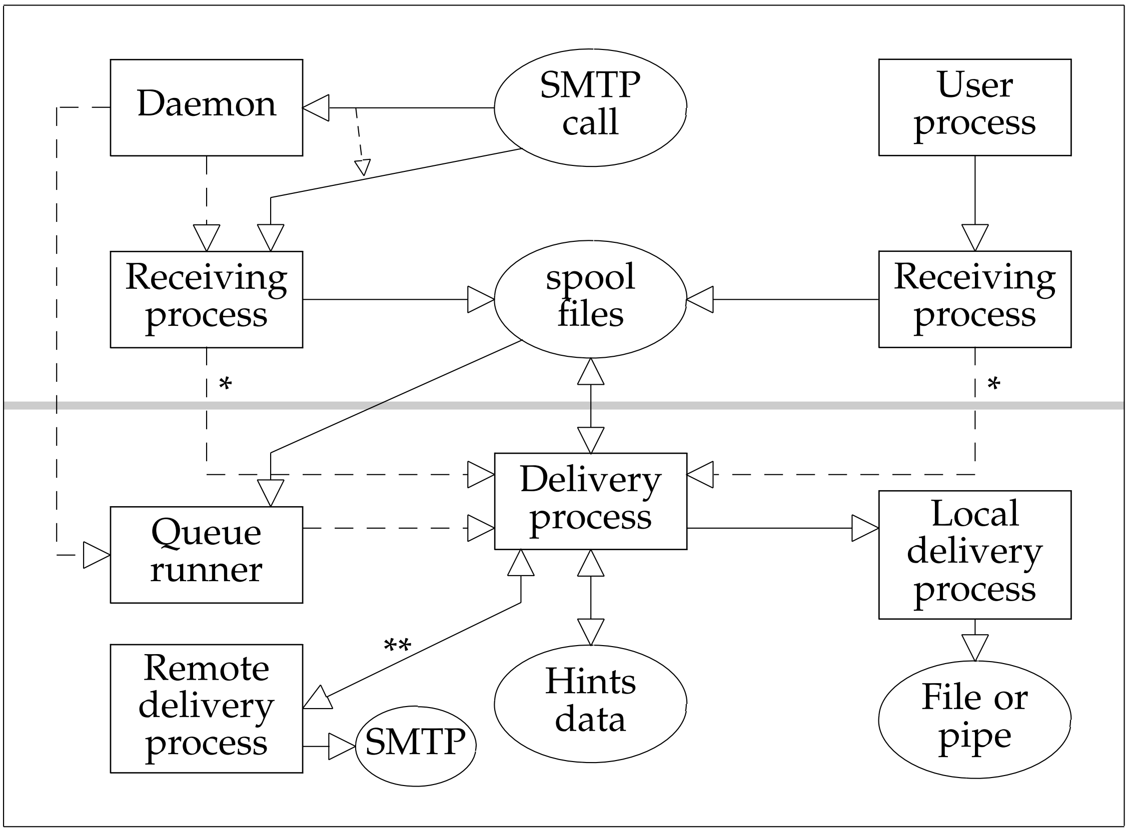
Separating mail functions



Exim design overview

- Immediate delivery on reception (usually)
- One pool of waiting messages
- There is no single central controlling process
  Optional daemon for listening and
  starting 'queue-runner' processes
- Multiple processes with decentralized control
  No long-lived processes (daemon excepted)
- Interaction is via shared files
  Spool directory
  Hints data
  Log files

How Exim processes interact



*   At times of high load, automatic delivery may be suspended

** Only when parallel remote delivery is configured, and there are multiple remote recipients

The Daemon process

- Daemon listens for incoming SMTP connections
  Forks a receiving process for each one
  Maximum number of simultaneous calls
  Can reserve slots for local hosts
- Daemon starts queue-runner processes periodically
  Maximum number of simultaneous ones
- Running a daemon is optional
  Can use **inetd** and **cron** instead
  Lose maximum number control
- Use SIGHUP to restart after configuration change

Receiving processes

- Accept local or SMTP Messages
- Each message is written to the spool area
- Checks on envelope and host are performed for SMTP input (as configured)
- Normally, a delivery process is started

Queue runner processes

- Scan pool of waiting messages in random order
- Start a delivery process for each (unlocked) one
- Wait for completion before moving on
- Several queue-runners may be active at one time

Delivery processes

- Each message delivery attempt happens in a separate process
- Retry times are checked for each address and host
- Local deliveries are done first, each one in yet another process, **setuid** to the user
- Local deliveries write to files or pipes
- Remote deliveries (over SMTP) can be done in parallel if configured, for multiple addresses
- Error messages are generated after delivery failures

Frozen messages

- Configuration error or serious delivery problem
  e.g. non-absolute path for alias or mailbox file
- Explicit freezing in system filter file
- Undeliverable message that cannot be returned to its sender – usually a 'bounce' message
- Bounce failures can be ignored, or kept for a short time only
- Postmaster can be mailed whenever a message is frozen
- Messages can be thawed manually
- Auto-thaw can be set to retry at a given interval

Exim spool files

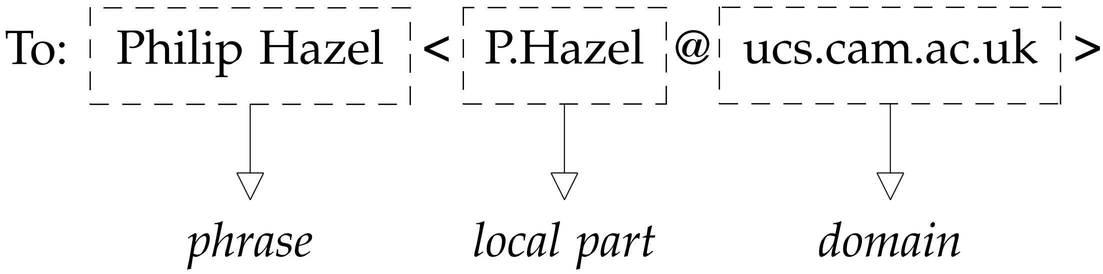
Commonly kept in **/var/spool/exim**

- Messages: **input** directory, two files per message
  -H file contains header, envelope and status
  -D file contains body
- **input** can be subdivided into 62 sub-directories
  to improve performance (single-char names)
- -J files in **input** are delivery journals
- -J file is deleted once -H is updated
- Message logs: **msglog** directory, for information
  Removed when message complete
- Pid file for the daemon: **exim-daemon.pid**
- Log files: **log** directory (optionally)
- Hints databases: **db** directory

Exim log files

- Main log: **log/mainlog**, normally rotated daily
  Log level controls the verbosity
- Panic log: **log/paniclog**, should normally be empty
- Reject log: **log/rejectlog**, records policy rejections
  Some duplication with **mainlog**, but includes additional information

Address handling



Two types of address handling:

- Depends only on domain
- Depends on local part and domain

Two kinds of domain:

- Local domain – normally uses local part
  Handled by *director*
- Remote domain – normally no use of local part
  Handled by *router*

Local domains

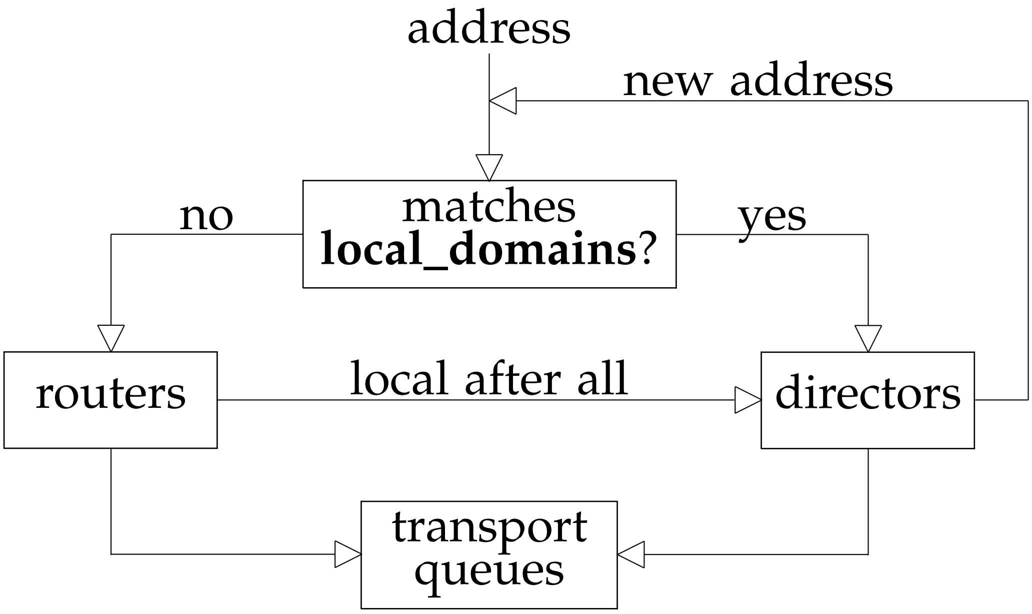
Defined in the main configuration:

```
local_domains = cus.cam.ac.uk

local_domains = "ursa.cus.cam.ac.uk:\
                 taurus.cus.cam.ac.uk:\
                 cus.cam.ac.uk"

local_domains = *.cus.cam.ac.uk

local_domains = "\
  dbm;/usr/exim/local_domains"
```

Detected by routers:

- Expansion of abbreviated name
  e.g. cus => cus.cam.ac.uk
- MXed to the local host (if so configured)

Processing an address



All directing and routing is done before transporting

- Local transports (to files or pipes) are run first
- Remote (SMTP) transports afterwards
- Delivery log lines record which director or router, and which transport, were used
- Routing and directing is done from scratch at each delivery attempt
- **Exception**: the **once** option for mailing lists

Configuration file

Main configuration

Transports configuration

Directors configuration

Routers configuration

Retry configuration

Rewrite configuration

Authenticators configuration

Transports configuration

Delivery to remote host:

```
remote_smtp:
  driver = smtp
```

Delivery to local mailbox:

```
local_delivery:
  driver = appendfile
  file = /var/mail/${local_part}
# file = /home/${local_part}/inbox
# file = ${home}/inbox
  delivery_date_add
  envelope_to_add
  return_path_add
# group = mail
# mode = 0660
```

Deliveries caused by **.forward** files:

```
address_pipe:
  driver = pipe
  return_output

address_file:
  driver = appendfile
  delivery_date_add
  envelope_to_add
  return_path_add

address_reply:
  driver = autoreply
```

Directors configuration

Handle system aliases:

```
system_aliases:
  driver = aliasfile
  file = /etc/aliases
  search_type = lsearch
# user = exim
  file_transport = address_file
  pipe_transport = address_pipe
```

Handle users' **.forward** files:

```
userforward:
  driver = forwardfile
  file = .forward
  no_verify
  no_expn
  check_ancestor
# filter
  file_transport = address_file
  pipe_transport = address_pipe
  reply_transport = address_reply
```

Set up local deliveries:

```
localuser:
  driver = localuser
  transport = local_delivery
```

Routers configuration

Route via the DNS:

```
lookuphost:
  driver = lookuphost
  transport = remote_smtp
```

Route IP literal addresses:

```
literal:
  driver = ipliteral
  transport = remote_smtp
```

- IP literal addresses are of the form

  `user@[192.168.5.3]`
- Not generally relevant in today's Internet
- Many sites lock them out.

Local transport for a large installation

```
local_delivery:
  driver = appendfile
  directory = /var/mail/\
    ${nhash_64:$local_part}/\
    $local_part
# directory = /var/mail/\
#   ${nhash_8_512:$local_part}/\
#   $local_part
  maildir_format
  delivery_date_add
  envelope_to_add
  return_path_add
# group = mail
# mode = 0660
```

If user name is *afnog*

- Directory `/var/mail/5/afnog` is created
- (`/var/mail/4/389/afnog` in 2nd example)
- Subdirectories `tmp`, `new`, and `cur`
- Each message is a separate file
- Simultaneous deliveries can occur
- No locking needed
- Daemons and MUAs must support maildir

Retry algorithms

- Fixed or increasing time intervals
- Change of rule as time passes
- Predication on specific errors

Default retry rule:

```
*   *  F,2h,15m;  G,16h,1h,1.5;   F,4d,8h;
```

For all domains

For all errors

Every 15 minutes for 2 hours

Start at 1 hour interval; increase by 1.5 until 16 hours

Every 8 hours up to 4 days

Use **-brt** to check retry rules:

```
exim -brt aol.com
Retry rule: aol.com F,2h,15m; F,4d,30m;
```

Delivery errors

Host errors: not related to message or recipients

- Always temporary
- Host gets delayed, for all messages
  No message is sent to it till its retry time arrives
- Retry rule selected by host or domain

Message errors: related to message, but not recipient

- Message gets delayed to that host
- Rule selected by host or domain
- Does not affect other messages to the host

Recipient errors: specific to a recipient

- Recipient gets delayed in all messages
  but only in queue runs
- Rule selected by domain (remote deliveries)
  or full address (local deliveries)

Longstop check: total message queue time

- Bounce if more than host's retry period

Address rewriting

- Configured rewriting is done on reception
- Rewriting should be used with care
- It is not intended as a routing mechanism
- Host name to corporate domain

```
*@*.plc.co.uk  $local_part@plc.co.uk
```

- Login name to real name

```
*@plc.co.uk   ${lookup{$local_part}\
               dbm{/etc/realnames}\
               {$value}fail}@$domain bcfrF
```

- Use quotes if replacement contains white space

SMTP Authentication

- Extension to SMTP protocol
- Server advertises AUTH mechanisms in response to EHLO
- Challenge/response sequence

  C: AUTH <*mechanism*> [<*data*>]
  S: <*challenge*>
  C: <*response*>
  ...
  S: 235 Authentication successful

- All data is encoded as base-64 strings

Control of incoming mail

- Verification
  Checking that an envelope makes sense before accepting a message
- Policy control:
  Blocking mail from sources you don't like
- Relay control
  Controlling what goes *through* your host

Verification

Recipient verification (**receiver_verify**)
Sender verification (**sender_verify**)

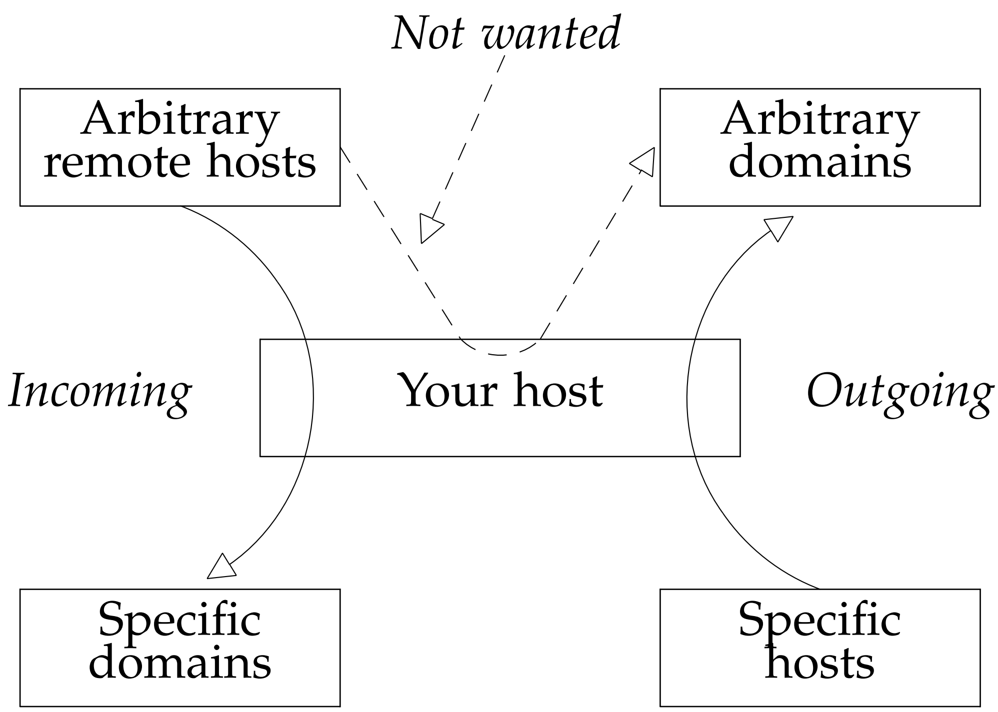
- Check envelope at SMTP time
- Exceptions by host, network, or sender
- Run directors and routers in verify mode
  Local parts checked for local domains
  Domain check only for the rest
- Failures result in SMTP error codes
- Logging in the reject log

Sender verification failures reject

- After DATA the first time (logging headers)
- After MAIL the second time
- After all RCPTs thereafter

Also possible to insist on valid header syntax
and valid EHLO/HELO data

Relay control



Incoming is controlled by **relay_domains**

```
relay_domains = "*.mydomain.ex: \
  cdb;/customer/domain/list"
```

- **relay_domains_include_local_mx** adds any domain MX'ed to your host
- RCPT is accepted if domain matches
- Otherwise treated as outgoing relay

Relay control (2)

Outgoing relaying is controlled by

host_accept_relay

To accept from all but one host on the local net:

```
host_accept_relay = "\
  !192.153.213.99 : 192.153.213.0/24"
```

Relaying from specific sender addresses can be controlled by

sender_address_relay
relay_match_host_or_sender

The latter converts the default 'and' of the conditions into an 'or' combination – but is not recommended because it is easy to forge.

When SMTP authentication is in use

host_accept_auth_relay

allows relaying from specific authenticated hosts.

Policy controls (hosts)

Control by host name

```
host_reject = !xx.yy.zz : *.yy.zz : !*.zz
```

Wildcard names require DNS reverse lookups
DNS lookups can give temporary errors
`+allow_unknown` accepts unknown hosts

Control by IP address

```
host_reject = "!131.111.8.1 : \
   131.111.8.0/24 :\
  !131.111.0.0/16"
```

Error response given at connection time
Alternatively, reject at RCPT time by using

host_reject_recipients

Policy controls (hosts 2)

Use of RBL and/or DUL

```
rbl_domains = "\
  rbl.maps.vix.com:dul.maps.vix.com"
rbl_reject_recipients = false
rbl_warn_header
```

Can use **/warn** or **/reject** for individual control

```
rbl_domains = "\
  rbl.maps.vix.com/warn:\
  dul.maps.vix.com/reject"
```

Error messages for policy rejections can be tailored by setting **prohibition_message**, e.g.

```
prohibition_message = "\
  Contact address: postmaster@xxx.yyy"
```

Testing policy controls

The **-bh** option runs a fake SMTP session

```
exim -bh 192.203.178.4
>>> host in host_lookup? yes (end of list)
>>> looking up host name for 192.203.178.4
>>> IP address lookup yielded dul.crynwr.com
>>> host in host_reject? no (option unset)
>>> host in host_reject_recipients? no
    (option unset)
>>> host in rbl_hosts? yes (*)
>>> RBL lookup for
    4.178.203.192.dul.maps.vix.com succeeded
>>> => that means it is black listed ...
>>> See <http://maps.vix.com/dul/>
LOG: recipients refused from dul.crynwr.com
    [192.203.178.4] (RBL dul.maps.vix.com)
220 xoanon.csi.cam.ac.uk ESMTP Exim 3.02 ...
```

Message filtering

A user’s filter

- Run during directing, whenever a message is addressed to the user and not yet delivered
- True forwarding
- Alternate mailboxes
- Deliveries to pipes

The system filter

- Run once per message, at every delivery start
- Additional filtering commands: **freeze**, **fail**
- Can see all recipients
- Can add header lines
- Can set ‘scores’ for user filters

Filter file example

```
# Exim filter

# Don't touch error messages

if error_message then finish endif

# Throw away known junk
if
  $h_subject: contains "Make money" or
  $h_precedence: is "junk" or
  $h_sender: matches ^\\d{8}@ or
  $message_body contains "this is spam"
then
  seen finish
endif

# Auto-reply if next line commented

finish

if personal
   alias ph10@cam.ac.uk
then
   mail
   subject "Re: $h_subject"
   file $home/Auto-Reply/MESSAGE
   log  $home/Auto-Reply/LOG
   once $home/Auto-Reply/ONCE
endif
```

Virtual domains (1)

- Straightforward cases are just aliasing
- Include in **local_domains**; use a director

```
virtual_domains:
  driver = aliasfile
  domains = cdb;/etc/virtuals
  file = /etc/aliases/$domain
  search_type = lsearch
  qualify_preserve_domain
  no_more
```

- New addresses can be remote, or mailboxes on the local host
- Alias files can be shared between domains

```
file = "${lookup{$domain}cdb\
  {/etc/virtuals}{$value}}"
```

- Or

```
file = $domain_data
```

- Use (e.g.) `lsearch*` for defaults
- Exceptions can be handled with **:fail:**

```
*:             admin
except:        :fail: Unknown user
admin:         theboss@domain1
user1:         abc@domain2
user2:         xyz@domain3
```

Virtual domains (2)

- Use **include_domain** to mix domains in one file

```
virtual_domains:
  driver = aliasfile
  domains = cdb;/etc/virtuals
  file = /etc/virtual.aliases
  search_type = cdb
  include_domain
  qualify_preserve_domain
  no_more
```

- Use (e.g.) `lsearch*@` for defaults

```
*:             lmn@domain6
abc@virt1:     xyz@domain1
*@virt1:       abc@virt1
abc@virt2:     abc@domain2
xyz@virt3:     pqr@domain3
```

- Could omit **domains** and **no_more** if subsequent directors have a **domains** setting

Virtual domains (3)

- Common postmaster address

```
postmaster:
  driver = smartuser
  local_parts = postmaster
  new_address = postmaster@your.domain
```

- Place before to override all domains
- Place after if some domains have postmasters
- If any have defaults and no postmaster, use

```
postmaster:   :unknown:
```

- Excepts **postmaster** from the default

Saving mail for onward delivery (1)

- Avoids contaminating Exim's queue
- Allows time and space control
- Several ways of preserving the envelope
- (1) BSMTP uses SMTP commands

```
MAIL FROM:<sender@sender.domain>
RCPT TO:<1st@recipient.domain>
RCPT TO:<2nd@recipient.domain>
DATA
The message, with lines that start with
a dot escaped by inserting another dot.
.
```

- Can be copied verbatim to an SMTP channel
- Several variants
  `bsmtp = one` => one recipient per message
  `bsmtp = domain` => one copy for each domain
  `bsmtp = all` => one copy only
- Subject to other contraints
  Must not refer to **$local_part**
  Domain batching if **$domain** used
  Must have same error addresses, hosts, header additions/removals, etc.
- (2) Use **return_path_add** and **envelope_to_add**
- (3) 'Mailstore' format puts envelope in separate file
- Requires one file per message

Saving mail for onward delivery (2)

- BSMTP example

```
bsmtp_router:
  driver = domainlist
  transport = bsmtp_transport
  route_list = "*.bsmtp.domains \
    ${lookup{$domain}cdb{/etc/bhosts}\
    ${value}fail}"
```

- This transport files messages by domain (host not used)

```
bsmtp_transport:
  driver = appendfile
  bsmtp = domain
  file = /bsmtp/$domain
  user = mail
```

- This transport files messages by host

```
bsmtp_transport:
  driver = appendfile
  bsmtp = all
  file = /bsmtp/$host
  user = mail
```

- Use **directory** instead of **file** for one-file-per-message

```
directory = /bsmtp/$host
maildir_format
```

- Maildir separates incoming from completed by using two directories, **tmp** and **new**

Mailboxes without accounts (1)

- Validity of local parts can be checked in several ways
- Use of aliasfile

```
no_account:
  driver = aliasfile
  file = /etc/no_account.db
  search_type = dbm
  transport = no_account
```

  Setting `transport` changes the behaviour

- Use of smartuser

```
no_account:
  driver = smartuser
  local_parts = "\
    dbm;/etc/no_account.db"
  transport = no_account
```

- In both cases, the data from the lookup is not used by the director

Mailboxes without accounts (2)

- For local delivery, a user/group is needed
- For a message store where files are not individually owned, transport can be simple

```
no_account:
  driver = appendfile
  file = /var/mail/$local_part
  user = mail
```

- Otherwise, make use of per-mailbox data in a file or a database

```
user1: uid=1234 gid=1023
       home=/home/user1
       mailbox=/home/user1/inbox
```

- Multiple file lookups in the transport will use caching

```
user = "\
  ${lookup{$local_part:uid}dbm{..."
```

Some options for large installations

- `split_spool_directory`

  Messages held in 62 subdirectories on the spool

- `remote_max_parallel = 20`

  Parallel remote deliveries (default none)
  Important for mailing lists

- `smtp_accept_max = 80`

  Maximum incoming (default 20)

- `smtp_accept_backlog = 50`

  TCP/IP stack's queue (default 5)

- `fallback_hosts =...`

  Can be used to move queue to secondary server

Building from FreeBSD port

```
cd /usr/ports/mail/exim
make -k deinstall
make clean
make
make install
make clean
```

Configuration choices are made for you

- An Exim user/group is not defined
  Only root can administer Exim
  Runs as root when receiving messages
  Runs as root when doing remote deliveries
- Only **lsearch** and **dbm** lookups are included
- Maildir support is not included
- Binaries, scripts in **/usr/local/sbin**
- Config in **/usr/local/etc/exim**
- Log cycling **cron** job not created

Building from generic distribution

Preparation

- Create a user and group for Exim
  e.g. username = groupname = `exim`,
  uid = gid = 42
- Add sysadmins to the group
- Choose location of files

spool	`/var/spool/exim`
logs	`/var/spool/exim/%slog`
	`/var/log/exim/exim_%slog`
config	`/etc/exim.conf`

- Fetch tarball, gunzip and de-tar
- You should get a directory `exim-3.14` (e.g.)

Build-time configuration

- `cd` to source directory
- `mkdir Local`
- Copy `/src/EDITME` to `Local/Makefile`
- Edit it according to instructions inside
- You can normally re-use for the next release
- You should not need to edit other files

Building from generic distribution (2)

Mandatory (example values)

```
BIN_DIRECTORY=/usr/exim/bin
CONFIGURE_FILE=/usr/exim/configure
```

Recommended

```
EXIM_UID=42
EXIM_GID=42
LOG_FILE_PATH=/var/log/exim_%slog
SPOOL_DIRECTORY=/var/spool/exim
SPOOL_MODE=0640
```

Drivers

Defaults normally taken (see next slide)

Optional

```
LOG_MAX=28
EXIM_MONITOR=eximon.bin
```

Optional modules

```
LOOKUP_DBM=yes
LOOKUP_LSEARCH=yes
LOOKUP_CDB=yes
LOOKUP_MYSQL=yes
```

etc.

```
LOOKUP_INCLUDE=-I /usr/local/mysql/include
LOOKUP_LIBS=-L/usr/local/lib -lmysqlclient
```

System-related

```
CC=gcc
CFLAGS=-O2 -Wall
```

and similar

Building from generic distribution (3)

Example **Local/Makefile** (comments removed)

```
BIN_DIRECTORY=/opt/exim/bin                  M
CFLAGS=-O2 -Wall -I/opt/local/include
CONFIGURE_FILE=/opt/exim/configure           M

DIRECTOR_ALIASFILE=yes                       D
DIRECTOR_FORWARDFILE=yes                     D
DIRECTOR_LOCALUSER=yes                       D
DIRECTOR_SMARTUSER=yes                       D

EXICYCLOG_MAX=28
EXIM_GID=42                                  R
EXIM_UID=42                                  R

EXIM_MONITOR=eximon.bin                      R

LOOKUP_CDB=yes
LOOKUP_DBM=yes                               D
LOOKUP_LSEARCH=yes                           D
LOOKUP_MYSQL=yes
LOOKUP_INCLUDE=-I/opt/local/mysql/include
LOOKUP_LIBS=-L/opt/local/mysql/lib

ROUTER_DOMAINLIST=yes                        D
ROUTER_LOOKUPHOST=yes                        D

SPOOL_DIRECTORY=/var/spool/exim              R
SPOOL_MODE=0640                              R

TRANSPORT_APPENDFILE=yes                     D
TRANSPORT_AUTOREPLY=yes                      D
TRANSPORT_PIPE=yes                           D
TRANSPORT_SMTP=yes                           D
```

Building from generic distribution (4)

Exim Monitor

- Additional configuration in `Local/eximon.conf`
- Edit `exim_monitor/EDITME`
- Can be an empty file

The Build

- `make` creates build directory
  e.g. `build-SunOS5-5.8-sparc`
- Links to source files
- Creates `Makefile` in build directory
- Builds...
- `make install` must be run as root
- Installs binaries and scripts
- Installs default configuration if none
- Does not replace `sendmail`

Test

- Fire up `eximon` if possible
- `exim -bt` *address*
- `exim -d` *address*
  message
  `.`
- Check `mainlog`, `paniclog`

Replacing Sendmail with Exim

Removing `sendmail`

- Kill `sendmail` daemon
- Replace `/usr/lib/sendmail` with link to Exim binary (`/usr/sbin/sendmail` on FreeBSD)

```
lrwxrwxrwx 1 root other 18 Apr 14 14:33
  /usr/lib/sendmail -> /usr/exim/bin/exim
```

- Restart `sendmail` daemon without `-bd`, to clear off anything on its queue, if necessary.

```
/usr/lib/sendmail.sendmail -q10m
```

- Start up Exim daemon

```
exim -bd -q15m
```

- Set up **cron** job to rotate logs:

```
1   0 * * *   /usr/exim/bin/exicyclog
```

Converting Sendmail alias files

Sendmail alias files can contain four types of item

```
user1:            newuser1@newdomain1
user2@domain2:    newuser2@newdomain2
@domain3:         newuser3@newdomain3
%1@domain4:       @newdomain4
```

The first is a traditional alias, handled by Exim by default.

The second can be handled by setting `include_domain` on an **aliasfile** director. You need two directors if you are mixing both kinds in the same file:

```
qualified_aliases:
  driver = aliasfile
  file = /etc/aliases
  search_type = lsearch
  include_domain

unqualified_aliases:
  driver = aliasfile
  file = /etc/aliases
  search_type = lsearch
```

The third is a default alias for the domain. This needs to be changed to `*@domain3` and then it can be handled by

```
search_type = lsearch*
```

Converting Sendmail alias files (2)

The fourth is a change of domain, keeping the same local part. Exim can handle this in one of two ways:

- Make the domain non-local, and use a **domainlist** router:

```
route_domain4:
  driver = domainlist
  route_list = domain4 newdomain4
```

- Keep the domain local, and use a **smartuser** director:

```
alias_domain4:
  driver = smartuser
  domains = domain4
  new_address = $local_part@newdomain4
```