Sendmail — Care and Feeding

Last Edit
March 24, 1992

Reg Quinton <reggers@julian.uwo.ca>
Computing and Communications Services
The University of Western Ontario
London, Ontario N6A 5B7
Canada

1. History/Goals

Our goal is to make system administrators self sufficient — CCS cannot administer *your* mail system. You have to take responsiblity (but we can help).

Sendmail released with BSD 4.2 circa 1983 by Eric Allman. My first experiences with sendmail were in 1984 with BSD 4.2 on a DEC/Vax 750 called `deepthot`.

The original context:

(1) DARPA network projects — TCP/IP, RFC821/822

(2) MILnet, ARPAnet, Berknet (peer to peer)

(3) UUCP, BITNET, Decnet ...

Current version is 5.64 available by anonymous FTP from `uunet.uu.net` (but most systems come with good versions — 5.61++).

A goal of sendmail is to provide a *flexible* mail router for

(1) TCP/IP peer to peer

(2) Gateway between nets (*eg*. UUCP to ARPAnet)

To be a gateway requires header and address rewriting (*munging*). These days that's minimized since most mail is delivered from sender directly to recipient without having to go through any relays.

Sendmail is a *configurable* system. Predated by delivermail (of BSD 4.1) which required edit/recompile to reconfigure.

`/usr/lib/sendmail`
— the executable image.

`/usr/lib/sendmail.cf`
— the configuration file.

Early distributions had ugly configuration files (but then life was complicated back then).

There is a reputation that you have to be a *guru* to maintain sendmail — this is simply not true!

Alternatives to sendmail? (The usual gripe is “I can’t figure out sendmail.cf”). In my humble opinion alternatives like MMDF, {X, Y, Z}mail, etc. should be **avoided**.

Sendmail is a mature product with few problems. Most Unix sites on the Internet are using sendmail.

Sendmail is also the product CCS supports — choose another at your own risk.

1.1. Where?

Instructions, etc. for supporting a sendmail system are available from `julian.uwo.ca` in `~ftp/nic/sendmail`

```
[3:27pm zebra] ftp julian.uwo.ca
Connected to julian.uwo.ca.
220 julian.uwo.ca FTP server (Version 5.65 ...
Name (julian.uwo.ca:reggers): anonymous
331 Guest login ok, send ident as password.
Password:
230 Guest login ok, access restrictions apply.
ftp> cd nic/sendmail
250 CWD command successful.
ftp>
```

Alternatively ...

```
[3:27pm julian] cd ~ftp/nic/sendmail
[3:28pm julian] ls
Configure.sh            README.aliases
README                  aliases.distribution
sendmail.distribution
```

2. What is Sendmail?

2.1. Read the manual page!

Sendmail delivers/routes all mail on a Unix system. It accepts a number of *arguments* (*eg.* `-bd` means run as a daemon) and messages are usually received/piped as *stdin*.

```
[3:28pm julian] man sendmail
SENDMAIL(8)          EP/IX            SENDMAIL(8

NAME
     sendmail - send mail over the internet

SYNOPSIS
     /usr/lib/sendmail [ flags ] [ address ...

     newaliases

     mailq [ -v ]

DESCRIPTION
     sendmail sends a message to ...
```


2.2. Sendmail is a process

On `julian.uwo.ca` we process about 1,000 — 2,000 E-mail messages a day — all are handle by sendmail.

When you say, *eg.* “`mail reggers`”, your message is piped (as stdin) to an instance of sendmail with `reggers` as an argument (standard fork/exec/argv and pipe notions).

If you look at processes you may catch an instance of sendmail delivering someone’s mail.

```
[3:30pm julian] ps -agx | grep sendmail
 310  ? IW  0:45 /usr/lib/sendmail -bd -q15m
3161 q5 SWN 0:00 /usr/lib/sendmail -em -oi regg
```

You should always see at least one instance of sendmail — the sendmail daemon (this process is started at boot time and persists like most daemons).

The instance with arguments '`-bd -q15m`' is a daemon (`-bd`) that is retrying the queue every 15min (`-q15m`).

You can tell it's a daemon because it has no controlling terminal and it has a low process number (because it was started early).

```
 310  ? IW  0:45 /usr/lib/sendmail -bd -q15m
```

The instance with arguments '`-em -oi reggers`' has been fork/exec'd from a program like "`mail`" — delivering mail to '`reggers`':

```
3161 q5 SWN 0:00 /usr/lib/sendmail -em -oi regg
```


2.3. Sendmail Daemon

Sendmail exists as a daemon process started at boot time.

(1) `ps -agx | grep sendmail`

(2) `grep sendmail /etc/rc*`

(3) `netstat -a | grep smtp`

(4) `mconnect julian.uwo.ca`

(5) `telnet julian.uwo.ca 25`

Sendmail is the process that:
- shows up in a process list,
- is started out of your `/etc/rc` files,
- is listening on the SMTP port,
- you connect to with `mconnect`,
- alternatively with telnet to the SMTP port.

The sendmail daemon has two principle jobs:

(1) listens to SMTP port for mail arriving via TCP/IP.

(2) processes messages spooled (usually) in `/usr/spool/mqueue`

SMTP port assigned in `/etc/services` as port 25 (that's a "Well Known Service" — WKS records in DNS).

Protocol defined in RFC821 "Simple Mail Transfer Protocol", Jon Postel, 1982.

SMTP server started by standard fork/exec sequence — child handles connection and parent listens for more.

The peer is often another sendmail processing its mail queue.

2.4. Sendmail Queue

Messages are queued for delivery in a spool directory — usually `/usr/spool/mqueue`

```
[10:08am julian] mailq
...
AA08790 101 Tue Mar 24 09:54 <VINCE@VAXI.SSCL.U
            (Deferred: Connection timed out ...
            <ASSHUSCR@CMS.AM.CC.READING.AC.UK>
AA07338 396 Tue Mar 24 09:38 <3002_402@uwovax.u
            (Deferred: Connection timed out  ..
            <rosi@sg1.chem.nrc.ca>
AA23045  95 Mon Mar 23 19:50 <magundi@gaul.csd.
            (Deferred: Connection timed out ...
            <rcp@hardy.hdw.csd.harris.com>
AA10804 744 Mon Mar 23 16:56 <PCHIARAM@novell.b
            (Deferred: Network is unreachable)
            <marco@cs.athabascau.ca>
```

Alternatively ....

```
[9:49am julian] ls /usr/spool/mqueue
dfAA00014  dfAA10804  dfAA27661  qfAA00022  qfA
dfAA00022  dfAA18882  dfAA27954  qfAA07180  qfA
dfAA07180  dfAA20949  dfAB15967  qfAA07338  qfA
dfAA07338  dfAA23045  qfAA00014  qfAA10804  qfA
```

3. Sendmail files

`/usr/lib/sendmail`
— the executable. Sendmail must be found here since programs like `Mail`, MH, UUCP, etc. pipe into this program by fork/exec.

`/usr/lib/sendmail.cf`
— the configuration file. Sendmail looks for the configuration file here (on SunOS also `/etc/sendmail.cf`). This is readable text — it's not that tough.

`/usr/lib/sendmail.fc`
— a frozen version of the configuration file. A "quick" image that needn't be parsed. Unreadable. Rebuild when you change the config file by saying

```
[9:49am julian] sendmail -bz
```

Note: the daemon will still be using the old configuration file — kill and restart it too.

`/usr/lib/sendmail.hf`
— the help file (try the HELP command to the SMTP service).

`/usr/spool/mqueue`
— directory of spooled messages (this is configurable). Examine by saying

```
[9:49am julian] mailq
[9:49am julian] sendmail -bp
```

`/usr/lib/aliases`
— a list of aliases for mail forwarding (this is configurable). Readable.

`/usr/lib/aliases.{dir,pag}`
— data base versions of alias file. Aliases are *local* addresses. If you change the aliases file create a new data base:

```
[9:49am julian] newaliases
[9:49am julian] sendmail -bi
```

`/etc/passwd`
— local users data base.

`~user/.forward`
— per user forwarding.

3.1. Sendmail Audit Trail

Sendmail is usually configured (by an Option) to send audit records to `syslogd(8)` (yet another process started at boot time).

Syslogd is configured by `/etc/syslog.conf`. Ours will log lots of records for mail (anything at the debug level or above):

```
[3:34pm julian] grep mail /etc/syslog.conf
mail.debug              /usr/spool/syslog/mail
```

Every site sending/receiving mail *should* maintain a syslog of mail transactions.

```
[3:34pm julian] tail /usr/spool/syslog/mail
Mar 23 15:34:20 julian sendmail[3462]: AA03462:
     message-id=<9203232024.AA02680@julian.uwo.
Mar 23 15:34:20 julian sendmail[3462]: AA03462:
     from=<operations-owner@uwovax.uwo.ca>,
     size=488, class=0, received from hydra.uwo
Mar 23 15:34:21 julian sendmail[3464]: AA03462:
     to=<sheila@julian.uwo.ca>, delay=00:00:01,
     stat=Sent
```


4. Sendmail Paradigm

Sendmail receives messages, delivers them, all under the control of configuration files.

4.1. Input to sendmail

There are really only two ways for messages to be given to sendmail.

(1) from `mail`, `mh`, `news`, `uucp`, `cron`, etc. These programs fork/exec sendmail with stdin a message and argv a list of recipients and flags.

(2) from other mail daemons as an SMTP service. These programs open a network connection to the SMTP service. The sendmail daemon fork/exec's a child to take care of the connection.

(3) from `/usr/spool/mqueue` (from above). The `df*`, and `cf*` files define messages that need to be delivered (*eg.* when a network connection is down files are spooled). The sendmail daemon fork/exec's a child every so often to handle these (`-q15m`).

4.2. Configuration files

There are lots of configuration files that control how messages are delivered.

(1) `/usr/lib/sendmail.{cf,fc}` defines parsing and delivery rules — tables *how* to munge addresses and get rid of mail.

(2) `/usr/spool/mqueue` is where messages are spooled. The `df*`, and `cf*` files are deferred messages.

(3) `/usr/lib/aliases.{dir,pag}` aliases for local addresses. *eg.* mail for “postmaster” is sent to “colleen”.

(4) `/etc/passwd` defines local addresses.

Note that `aliases` override `passwd` entries — on our system we make sure that all users are entered into the alias file.

4.3. Message delivery

There are really only two ways for sendmail to get rid of messages.

(1) sendmail fork/exec/pipes messages to programs like `uux` (for UUCP) and `binmail` (for local delivery). Exit status determines success or failure.

(2) sendmail talks SMTP over a TCP/IP socket to another mail daemon.

5. Sendmail Configuration File

`/usr/lib/sendmail.cf` is a configuration file. Plain text, readable, edittable. The frozen version (`.fc`) is a compiled version that is easy to load. A supported distribution is available by anonymous ftp:

```
julian.uwo.ca:~ftp/nic/sendmail
```

There are a number of sections to consider. At first blush this looks frightening but it needn't be:

```
# w -- the domain name of this machine
# l -- format of the Unix "From addr date" line
# n -- MAIL-DAEMON's name
# o -- operators (for breaking strings into tok
# q -- default format for sender address
#
De$j Sendmail $v/$$Revision: 1.2 $$ ready at $b
Djlucille.physics.uwo.ca
Dwlucille.physics.uwo.ca
DlFrom $g  $d
DnMAIL-DAEMON
Do.%@!:,|
#Do.:%@!^=/;
Dq$?x$x $.<$g>
#Dq$g$?x ($x)$.
```


5.1. Comments

Any line beginning with a '#' is a comment to aid readablity. Our version should be readable — there certainly are lots of comments.

```
# Options...
# A<file> where is alias file located?
# a<time> wait how long for aliases to be rebui
# B<value> substitute what for white space? (de
# C<number> checkpoint after this many recipien
# c       queue for expensive mailers
# d<mode> delivery mode -
#         i:interactive, b:background, q:queue
# D       rebuild alias data base if required
# e<how>  dispose of errors -
#         p:print, q:status alone, m:mail,
#         w:write, e:mail and exit clean
# F<number> protection mode for queued files
# f       save Unix style From lines
# g<number> set gid on mailers to number
# H<file> location of help file
# i       ignore dotted lines (treat dots as te
# L<number> audit trail logging (0:none, 9:nice
# m       send to me too (even if sender == rec
# o       old style (unix) headers supported
# P<name> Postmaster's name.. Cc: Naks to him t
# Q<dir>  where do I queue?
```

5.2. Options

Any line beginning with an ‘O’ is an option. Most options can be set on the command line as ‘`-ox<value>`’ (but few are). Some have numeric values, others letters, etc.

```
# u<number> mailers run under this uid
# v         verbose mode
# x<number> load average which forces "dq" (def
# X<number> how many SMTP servers (def. 12)
#
OA/usr/local/lib/aliases
Oa10m
OD
Odb
OF0640
Og1
OH/usr/lib/sendmail.hf
OL9
Om
Oo
#OPpostmaster
OQ/usr/spool/mqueue
```

Recall that we said sendmail usually spools in `/usr/spool/mqueue` — the `OQ` option defines the spool area.

5.3. Strings

Any line beginning with a ‘D’ is a string definition. Strings are set to single letters and referred to with the `$j` notation.

```
# Required macros (Defined strings)
# e -- the SMTP service ready message
# j -- the mail-domain name of this machine
# w -- the domain name of this machine
# l -- format of the Unix "From addr date" line
# n -- MAIL-DAEMON's name
# o -- operators (for breaking strings into tok
# q -- default format for sender address
#
De$j Sendmail $v/$$Revision: 1.2 $$ ready at $b
Djlucille.physics.uwo.ca
Dwlucille.physics.uwo.ca
DlFrom $g  $d
DnMAIL-DAEMON
Do.%@!:,|
#Do.:%@!^=/;
Dq$?x$x $.<$g>
#Dq$g$?x ($x)$.
```

Note, there are lots of reserved strings — `$a` is the ARPA style date. Don’t try to define these.

5.4. Classes

Any line beginning with a “C” is a class definition (a set of tokens).

```
# Hackneyed domains I reach thru a local relay
CRbitnet cdn uucp
# major relay host
DRmail-relay.uwo.ca
```

Classes are set to single letters and referenced by the `$=Y` notation. The only class we use is for the set of pseudo domains not supported in the DNS. We get mail to them by punting to `mail-relay.uwo.ca`.

Note distinction between `$R` (the string) and `$=R` (the class). In the example we define a class of pseudo domains

```
CRbitnet cdn uucp
```

and a string for the name of a gateway machine

```
DRmail-relay.uwo.ca
```


5.5. Headers

Any line beginning with a “H” is a header definition — these are usually fine.

```
#############################
# Format of headers
#
H?R?Received: $?sfrom $s $.by $j$?r with $r$.;
        (id $i) $b
H?M?Resent-Message-Id: <$t.$i@$j>
H?M?Message-Id: <$t.$i@$j>
H?D?Resent-Date: $a
H?D?Date: $a
H?F?Resent-From: $q
H?F?From: $q
H?x?Full-Name: $x
HSubject:
# HPosted-Date: $a
# H?l?Received-Date: $b
```

There’s an interesting conditional construction:

```
$?sfrom $s $.
```

This means: if the string ‘`s`’ is defined then substitute ‘`from $s`’.

5.6. Mailer

Any line beginning with a “M” is a mailer definition. This defines how sendmail should communicate with different programs, *eg*. does the program accept multiple recipients on the same line?

```
#
# Mailer specifications...
#
Mlocal, P=/usr/local/lib/binmail,
        F=rlsXRDFMmnP, A=mail -d $u
Mprog,  P=/bin/sh,   F=lsDFMeuP,
        A=sh -c $u
Mether, P=[IPC], F=msDFMuCX,
        A=IPC $h, E=\r\n
```

The IPC mailer (ie. SMTP over TCP/IP) is built in — sendmail takes care of the Inter Process Communcation without the aid of any delivery program.

Sendmail uses a program to deliver mail locally — that is not built in.

We, optionally, support a mailer for routing over a UUCP link.

5.7. Rules and Rule Sets

Any line beginning with an “S” defines a numbered rule set. *eg*. S0 the message delivery rule set and S3 the preamble rule set.

```
################################################
# Re-writing rules are trivial, I either know a
# the MX world or, alternatively, know of somec
# does. Two routes are supported to the smart g
# smtp/ip or uux/uucp.
#
S0
R|$*@$j        $#prog$:$1              local pipe
R$*@$j         $#local$:$1             local pers
R$*@$*.$=R     $#ether$@$R$:$1@$2.$3   to gateway
R$*@$*         $#ether$@$2$:$1@$2      via mx met
```

Certain rule sets are reserved (*eg. all of* S0, S1, S2, S3 and S4 have special roles).

Some are required, *eg*. S0 is required (for obvious reasons). All else can be empty.

5.8. Rules

Any line beginning with an “R” is a rewriting rule. A rule on how to munge an address. All rules look like

R*pattern*<tab>*rewrite-rule*<tab>*comment*

For example, the preamble rule set:

```
S3
Rlocal!$*      $:$1         to make rmail work
R<>            $@$n@$j      turn into magic toker
R$*<$+>$*      $2           basic RFC821/822 pars
R@$*:$*        $:$2         around here (req)
R$- at $-      $1@$2        old fashioned, and du
R$*@$-         $@$1@$2.$U   qualify
R$*@$w         $@$1@$j      use preffered domain
R$*@$*         $@$1@$2      looks fine
R$*            $@$1@$j
```

In our sendmail configuration the function of S3 is to turn all addresses into strings that look like ‘`user@do.ma.in`’.

5.8.1. Rule Set Flow

All addresses are munged by a sequence of ‘Rewriting Rule Sets’. The diagram shows the various sequences. For example, a *resolved address* is an address that has been munged by S3 then by S0.

```
                  -> S0 -> a 'resolved address'
                 /
                /      -> S1 -> S ->
               /      /             \
address -> S3 -> D -                 -> S4 -> mes
                      \             /
                       -> S2 -> R ->
```

All addresses are munged through S3, then follow the arrows as appropriate.

The paths are for envelope addresses (S3 to S0) and message header addresses (S3, D, S1, S, and S4 for Sender addresses vs S3, D, S2, R, and S4 for Recipient addresses).

(1) Recipient addresses then go through S3 then S0 where they're 'resolved' to a delivery program and arg list; ie. `$#Mailer$@...$:...`

(2) Rule D is implicit (ie. no written set here) and confusing — if delivery to the sender is through a mailer with a C flag then any unqualified addresses get his domain.

(3) All Sender addresses go through S1, a mailer specific rule set 'S' (see mailer line), through S4, and into the message header.

(4) All Recipient addresses go through S2, a mailer specific rule set 'R' (see mailer line), through S4, and into the message header.

Rule set D is useful since many sites don't full qualify their mail — ie. by '`user`' they mean '`@my-domain`' and not '`@your-domain`'.

Rule sets S and R are useful when you communicate with non-compliant sites — like dumb old UUCP.

5.8.2. Rule Semantics

While the input string matches the left hand pattern apply the right hand rewrite rule. Else procede to next rule in the set.

```
R$* at $-    $1@$2.BITNET    yech, profs
```

In the sendmail.cf we support you'll find:

(1) S1, S2, S4 are empty — the important work is done in S0 and S3.

(2) S3 turns all addresses into `user@do.ma.in` (note that the < and > are dropped).

(3) S0, given input as `user@do.ma.in`, decides delivery mechanism by pattern matching on "`do.ma.in`". Is it local? Is it a pseudo-domain (like UUCP and BITNET)? Is it an MX domain?

(4) Rules sets for recipient S5 and sender S6 rewriting are defined as mailer specific rules — the only given one is for UUCP.

5.8.3. Pattern Semantics

Strings are broken in “tokens“ according to the special characters of “`$o`”.

```
Do.%@!:
```

For example, the address

```
a362%uwoccl.bitnet@cunyvm.cuny.edu
```

will be broken into a sequence of tokens for pattern matching and rewriting.

The tokens in this example are: ‘`a362`’ (a string), ‘`%`’ (a single character), ‘`uwoccl`’, ‘`.`’, ‘`bitnet`’, ‘`@`’, ‘`cunyvm`’, ‘`.`’, ‘`cuny`’, ‘`.`’, and finally ‘`edu`’.

Some tokens are single characters, others are character strings.

In the pattern matching the meta notations are:

(1) `$*` matches any sequence of tokens

```
Rlocal!$*        $:$1
```

eg. local!julian.uwo.ca!reggers

(2) `$+` matches any non-trivial sequence of tokens

```
R$*<$+>$*        $:$2
```

eg. Reg Quinton <reggers@uwovax.uwo.ca>

(3) `$−` matches any single token.

```
R$*@$-           $:$1@$2.$U
```

eg. reggers@uwovax

(4) `$x` matches the **string** defined by ‘`Dx`’

```
R$*@$j          $#local$:$1
```

eg. `reggers@julian.uwo.ca`

(5) `$=x` matches a **token** in the class defined by ‘`Cx`’

```
R$*@$*.$=R      $#ether$@$R$:$1@$2.$3
```

eg. `a362@uwocc1.BITNET`

5.8.4. Rewriting Semantics

A rewriting rule is only applied if the string matches the pattern. The rule is applied until it fails, then on to the next in the rule set. In the rewriting the meta notations are:

(1) `$n` is *nth* patten match in the string (count only the $ matches).

```
R@$*:$*         $:$2
```

(2) `$:rule`... apply the rule and procede to the next in rule set.

```
R<>             $:$n
```

(3) `$@rule`... apply the rule and exit the rule set.

```
R$-             $@$1@$j
```

(4) `$#mailer`... deliver to mailer (only in S0!)

```
R$-                 $#local$:$1
```

Within a `#Mailer` rule `$:` names the user, `$@` names the host. For example

```
R$*@$*.$=R      $#ether$@$R$:$1@$2.$3
```

The mailer is `ether`, the host `$R`, and the user is `$1@$2.$3`. This punts pseudo domains to a gateway machine.

```
R$*@$*          $#ether$@$2$:$1@$2
```

Again, the mailer is `ether`, but the host `$2`, and the user is `$1@$2`. This sends real domains to the machine listed in the MX record.

6. Sendmail Testing

You can test whether an address is deliverable or how a list expands:

```
[10:43am julian] sendmail -bv reggers@uwovax.uw
reggers@uwovax.uwo.ca... deliverable
```

You can test rule sets in a verbose mode:

```
[10:43am julian] sendmail -bt
ADDRESS TEST MODE
Enter <ruleset> <address>
> 0 reggers@uwovax.uwo.ca
rewrite: ruleset  3   input:
     "reggers" "@" "uwovax" "." "uwo" "." "ca"
rewrite: ruleset  3 returns:
     "reggers" "@" "uwovax" "." "uwo" "." "ca"
rewrite: ruleset  0   input:
     "reggers" "@" "uwovax" "." "uwo" "." "ca"
rewrite: ruleset  0 returns:
     "^V" "ether" "^W" "uwovax" "." "uwo" "." "
     "^X" "reggers" "@" "uwovax" "." "uwo" "."
```

Note that Rule Set 3 is always applied. Note also the token sequences displayed as input and output to th the rule sets.

You can test a new configuration file:

```
[10:43am julian] sendmail -bt -Csendmail.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
> 0 reggers@uwovax.uwo.ca
```

Make sure you test things before you install them. Testing with rule set 0 is most common.

```
[10:43am julian] sendmail -bt
ADDRESS TEST MODE
Enter <ruleset> <address>
> 0 reggers@uwovax.uwo.ca
rewrite: ruleset  3   input:
     "reggers" "@" "uwovax" "." "uwo" "." "ca"
rewrite: ruleset  3 returns:
     "reggers" "@" "uwovax" "." "uwo" "." "ca"
rewrite: ruleset  0   input:
     "reggers" "@" "uwovax" "." "uwo" "." "ca"
rewrite: ruleset  0 returns:
     "^V" "ether" "^W" "uwovax" "." "uwo" "." "
     "^X" "reggers" "@" "uwovax" "." "uwo" "."
```

The mailer is specified with ^V, the host with ^W, the user with ^X. Each as a sequence of tokens.

7. Address Rewriting

We're trying to turn all addresses into user@do.ma.in. Here's what you have to deal with:

`user`
Easy, tack on local domain (`$j`).

`user@do.ma.in`
Easy, assume correct.

`<@site,@site:user@domain>`
Easy, use the last part. But some sites don't qualify the trailing domain (that's their problem). This predates the Domain Name System and shouldn't be used anymore.

`host!host!user`
Easy, write as `user@host.UUCP` (but requires that someone support UUCP maps). Should only appear on julian.uwo.ca (and on uucp neighbors).

`host!do.ma.in!user`
Easy, write as `user@do.ma.in`

`user@host`
: This is hard — should we assume local domain, or UUCP, or BITNET, or what? No matter what you chose, you will be wrong in **some** instances. We assume local domain.

This explains why one occasionally sees addresses of the form `user@host.uwo.ca` when the host isn't in the local domain

If everyone behaved, ie. used fully qualified registered domain names, we would have no problems.

8. MX sendmail

A domain is listed in the Domain Name Service. Records detail things like `A` (address), `WKS` (well known services), `HINFO` (machine type), `MX` (mail exchangeer), and other good stuff (the `IN` records are Internet information, other records are possbile — like Decnet, IPX, etc.).

From the cogsci DNS data:

```
cogsci.uwo.ca.   IN A 129.100.6.10
     IN MX 0     cogsci.uwo.ca.
     IN MX 10    julian.uwo.ca.
     IN HINFO    Sun3/160 "Sun UNIX"
```

The MX records order the hosts that should be tried — if you have mail for `user@cogsci.uwo.ca` try to connect to `cogsci.uwo.ca`, if that fails try `julian.uwo.ca`, and if that fails spool and try again later.

Note that mail for cogsci.uwo.ca is delivered **directly** to them. It is **not** store and forward through some gateway. Very different from the old days of BITNET, UUCP, MLNET, and other store and forward nets.

Alternatively ...

```
[11:26am julian] nslookup
Default Server:  julian.uwo.ca
Address:  129.100.2.12

> set type=any
> lri.uwo.ca
Server:  julian.uwo.ca
Address:  129.100.2.12

lri.uwo.ca      preference = 0,
     mail exchanger = julian.uwo.ca
lri.uwo.ca      preference = 10,
     mail exchanger = hydra.uwo.ca
lri.uwo.ca      CPU=SGI OS=Unix
julian.uwo.ca   inet address = 129.100.2.12
hydra.uwo.ca    inet address = 129.100.2.13
>
```

Not all domains listed in the DNS have an IP address (*eg.* `*.dec.com`). If a domain is addressable by E-mail they either have an IP address or a MX record pointing at someone who does.

You can mail to sites not on the Internet — by mailing to MX gateways which can forward things on.

Note that **some** mail addresses mention systems that are not listed in the DNS

```
[11:28am julian] nslookup
Default Server:  julian.uwo.ca
Address:  129.100.2.12

> set type=any
> uwocc1.bitnet
Server:  julian.uwo.ca
Address:  129.100.2.12
*** julian.uwo.ca
    can't find uwocc1.bitnet: Non-existent doma

> watmath.uucp
Server:  julian.uwo.ca
Address:  129.100.2.12
*** julian.uwo.ca
    can't find watmath.uucp: Non-existent domai
```

Strictly speaking, pseudo-domains (like bitnet and uucp) should be hidden behind some real domain.

```
reggers%uwocc1.bitnet@hydra.uwo.ca
```

But we support *some* with a rule

```
CRbitnet uucp cdn
```
etc .....
```
R$*@$*.$=R      $#ether$@$R$:$1@$2.$3
```

You can mail to `user@host.bitnet` because your sendmail system has been configured to route that pseudo domain to a gateway machine.

8.1. MX routing

Consider the rule in S0:

```
R$*@$*          $#ether$@$R$:$1@$2
```

This punts, via SMTP, everything to a gateway (usually `mail-relay.uwo.ca`). This requires that the gateway understand MX routing (or punt on to some who does).

This also requires a cooperating gateway to do your work (recall our goal is to make you self sufficient).

```
R$*@$*          $#ether$@$2$:$1@$2
```

This punts, via SMTP, to the recipient domain. For non-MX sendmails this would require that you list all 250,000 domains in `/etc/hosts`. For MX sendmail the DNS is queried and MX records are honored.

An MX version of sendmail makes rule set S0 simple — an address is either local, a pseudo-domain, or else a real domain.

Fortunately the number of pseudo domains has decreased over the years.

9. Rsendmail — NFS environment

The typical NFS environment around Western is a hidden domain with shared:

(1) `/etc/passwd`
(2) `/usr/spool/mail`

Having configured sendmail on the server machine how do users submit mail? Two alternatives:

(1) Configure a simple sendmail.cf for your clients that punts to the server.
(2) Construct a simple sendmail for your clients that rsh's sendmail on the server.

```
julian.uwo.ca:~ftp/nic/rsendmail
```

This has some minor problems, but is in use at many sites (especially within CCS).