

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 responsibility (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.

UWO/Sendmail

2

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!

3

UWO/Sendmail

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
```

<div><div><h2>2. What is Sendmail?</h2><h3>2.1. Read the manual page!</h3><p>Sendmail delivers/routes all mail on a Unix system. It accepts a number of <i>arguments</i> (eg. -bd means run as a daemon) and messages are usually received/piped as <i>stdin</i>.</p><pre>[3:28pm julian] man sendmail SENDMAIL(8) EP/IX SENDMAIL(8)</pre><p>NAME</p><p>sendmail - send mail over the internet</p><p>SYNOPSIS</p><p>/usr/lib/sendmail [flags] [address ...</p><p>newaliases</p><p>mailq [-v]</p><p>DESCRIPTION</p><p>sendmail sends a message to ...</p><div><div>UWO/Sendmail</div><div>6</div></div></div></div> <td><div><div><h2>2.2. Sendmail is a process</h2><p>On julian.uwo.ca we process about 1,000 — 2,000 E-mail messages a day — all are handle by sendmail.</p><p>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).</p><p>If you look at processes you may catch an instance of sendmail delivering someone’s mail.</p><pre>[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</pre><p>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).</p><div><div>UWO/Sendmail</div><div>7</div></div></div></div></td>	<div><div><h2>2.2. Sendmail is a process</h2><p>On julian.uwo.ca we process about 1,000 — 2,000 E-mail messages a day — all are handle by sendmail.</p><p>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).</p><p>If you look at processes you may catch an instance of sendmail delivering someone’s mail.</p><pre>[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</pre><p>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).</p><div><div>UWO/Sendmail</div><div>7</div></div></div></div>
---	--

<p>The instance with arguments ‘-bd -q15m’ is a daemon (-bd) that is retrying the queue every 15min (-q15m).</p> <p>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).</p> <pre>310 ? IW 0:45 /usr/lib/sendmail -bd -q15m</pre> <p>The instance with arguments ‘-em -oi reggers’ has been fork/exec’d from a program like “mail” — delivering mail to ‘reggers’:</p> <pre>3161 q5 SWN 0:00 /usr/lib/sendmail -em -oi reggers</pre>	<h3>2.3. Sendmail Daemon</h3> <p>Sendmail exists as a daemon process started at boot time.</p> <ol style="list-style-type: none">(1) <code>ps -agx grep sendmail</code>(2) <code>grep sendmail /etc/rc*</code>(3) <code>netstat -a grep smtp</code>(4) <code>mconnect julian.uwo.ca</code>(5) <code>telnet julian.uwo.ca 25</code> <p>Sendmail is the process that:</p> <ul style="list-style-type: none">shows up in a process list,is started out of your <code>/etc/rc</code> files,is listening on the SMTP port,you connect to with <code>mconnect</code>,alternatively with <code>telnet</code> to the SMTP port.
<p>UWO/Sendmail</p>	<p>UWO/Sendmail</p>

<p>The sendmail daemon has two principle jobs:</p> <ul style="list-style-type: none">(1) listens to SMTP port for mail arriving via TCP/IP.(2) processes messages spooled (usually) in /usr/spool/mqueue <p>SMTP port assigned in /etc/services as port 25 (that's a "Well Known Service" — WKS records in DNS).</p> <p>Protocol defined in RFC821 "Simple Mail Transfer Protocol", Jon Postel, 1982.</p> <p>SMTP server started by standard fork/exec sequence — child handles connection and parent listens for more.</p> <p>The peer is often another sendmail processing its mail queue.</p>	<div data-bbox="337 604 367 884"><h3>2.4. Sendmail Queue</h3></div> <p>Messages are queued for delivery in a spool directory — usually /usr/spool/mqueue</p> <pre>[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.h (Deferred: Network is unreachable) <marco@cs.athabascau.ca></pre> <p>Alternatively ...</p> <pre>[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</pre> <div data-bbox="1349 216 1372 884"><div>UWO/Sendmail</div><div>10</div></div> <div data-bbox="1349 216 1372 884"><div>UWO/Sendmail</div><div>11</div></div>
---	---

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.

<p>3.1. Sendmail Audit Trail</p> <p>Sendmail is usually configured (by an Option) to send audit records to <code>syslogd(8)</code> (yet another process started at boot time).</p> <p><code>Syslogd</code> is configured by <code>/etc/syslog.conf</code>. Ours will log lots of records for mail (anything at the debug level or above):</p> <pre>[3:34pm julian] grep mail /etc/syslog.conf mail.debug /usr/spool/syslog/mail</pre> <p>Every site sending/receiving mail <i>should</i> maintain a <code>syslog</code> of mail transactions.</p> <pre>[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</pre> <p>UWO/Sendmail</p> <p>14</p>	<p>4. Sendmail Paradigm</p> <p>Sendmail receives messages, delivers them, all under the control of configuration files.</p> <p>4.1. Input to sendmail</p> <p>There are really only two ways for messages to be given to sendmail.</p> <ol style="list-style-type: none"> (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 <code>/usr/spool/mqueue</code> (from above). The <code>df*</code>, and <code>cf*</code> 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 (<code>-q15m</code>). <p>15</p> <p>UWO/Sendmail</p>
--	--

<div data-bbox="337 1522 367 1829" data-label="Section-Header"> <h2>4.2. Configuration files</h2> </div> <div data-bbox="412 1161 472 1829" data-label="Text"> <p>There are lots of configuration files that control how messages are delivered.</p> </div> <div data-bbox="518 1161 967 1816" data-label="List-Group"> <ol style="list-style-type: none"> <li data-bbox="518 1161 613 1816">(1) <code>/usr/lib/sendmail.cf</code> defines parsing and delivery rules — tables <i>how</i> to munge addresses and get rid of mail. <li data-bbox="659 1161 755 1816">(2) <code>/usr/spool/mqueue</code> is where messages are spooled. The <code>df*</code>, and <code>cf*</code> files are deferred messages. <li data-bbox="800 1161 896 1816">(3) <code>/usr/lib/aliases.dir</code> aliases for local addresses. <i>eg.</i> mail for “postmaster” is sent to “colleen”. <li data-bbox="941 1291 967 1816">(4) <code>/etc/passwd</code> defines local addresses. </div> <div data-bbox="1013 1161 1105 1829" data-label="Text"> <p>Note that aliases override <code>passwd</code> entries — on our system we make sure that all users are entered into the alias file.</p> </div> <div data-bbox="1349 1682 1370 1829" data-label="Text"> <p>UWO/Sendmail</p> </div> <div data-bbox="1349 1161 1370 1182" data-label="Text"> <p>16</p> </div>	<div data-bbox="337 596 367 882" data-label="Section-Header"> <h2>4.3. Message delivery</h2> </div> <div data-bbox="412 212 472 882" data-label="Text"> <p>There are really only two ways for sendmail to get rid of messages.</p> </div> <div data-bbox="518 212 719 869" data-label="List-Group"> <ol style="list-style-type: none"> <li data-bbox="518 212 613 869">(1) sendmail <code>fork/exec/pipes</code> messages to programs like <code>uux</code> (for <code>UUCP</code>) and <code>binmail</code> (for local delivery). <code>Exit</code> status determines success or failure. <li data-bbox="659 212 719 869">(2) sendmail talks <code>SMTP</code> over a <code>TCP/IP</code> socket to another mail daemon. </div> <div data-bbox="1349 861 1370 882" data-label="Text"> <p>17</p> </div>	<p>UWO/Sendmail</p>
--	---	---------------------

<div data-bbox="337 1415 367 1829" data-label="Section-Header"> <h2>5. Sendmail Configuration File</h2> </div> <div data-bbox="412 1159 540 1829" data-label="Text"> <p><code>/usr/lib/sendmail.cf</code> is a configuration file. Plain text, readable, editable. The frozen version (<code>.fc</code>) is a compiled version that is easy to load. A supported distribution is available by anonymous ftp:</p> </div> <div data-bbox="586 1236 613 1749" data-label="Text"> <pre>julian.uwo.ca:~ftp/nic/sendmail</pre> </div> <div data-bbox="665 1159 727 1829" data-label="Text"> <p>There are a number of sections to consider. At first blush this looks frightening but it needn't be:</p> </div> <div data-bbox="768 1050 1260 1829" data-label="Text"> <pre># 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 tokens) # q -- default format for sender address # De\$ 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) \$.</pre> </div> <div data-bbox="1349 1682 1370 1829" data-label="Text"> <p>UWO/Sendmail</p> </div> <div data-bbox="1349 1159 1370 1182" data-label="Page-Footer"> <p>18</p> </div>	<div data-bbox="337 680 367 884" data-label="Section-Header"> <h2>5.1. Comments</h2> </div> <div data-bbox="412 216 505 884" data-label="Text"> <p>Any line beginning with a '#' is a comment to aid readability. Our version should be readable — there certainly are lots of comments.</p> </div> <div data-bbox="550 102 1274 884" data-label="Text"> <pre># Options... # A<file> where is alias file located? # a<time> wait how long for aliases to be rebuilt # B<value> substitute what for white space? (default) # C<number> checkpoint after this many recipients # 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 cleanly # 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 text) # L<number> audit trail logging (0:none, 9:nice) # m send to me too (even if sender == recipient) # o old style (unix) headers supported # P<name> Postmaster's name.. Cc: Naks to him to # Q<dir> where do I queue?</pre> </div> <div data-bbox="1349 216 1370 363" data-label="Text"> <p>UWO/Sendmail</p> </div> <div data-bbox="1349 861 1370 884" data-label="Page-Footer"> <p>19</p> </div>
---	--

<h3>5.2. Options</h3> <p>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.</p> <pre># 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</pre> <p>Recall that we said sendmail usually spools in /usr/spool/mqueue — the OQ option defines the spool area.</p>	<h3>5.3. Strings</h3> <p>Any line beginning with a ‘D’ is a string definition. Strings are set to single letters and referred to with the \$ j notation.</p> <pre># 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 \$h Djlucille.physics.uwo.ca Dwlucille.physics.uwo.ca DlFrom \$g \$d DnMAIL-DAEMON Do.%@!;, #Do.:%@!^=;/; Dq\$x\$x \$.<\$g> #Dq\$g\$x (\$x)\$.</pre> <p>Note, there are lots of reserved strings — \$a is the ARPA style date. Don’t try to define these.</p>
<p>UWO/Sendmail</p> <p>20</p>	<p>UWO/Sendmail</p> <p>21</p>

<div data-bbox="337 1669 365 1829" data-label="Section-Header"> <h2>5.4. Classes</h2> </div> <div data-bbox="412 1060 472 1829" data-label="Text"> <p>Any line beginning with a “C” is a class definition (a set of tokens).</p> </div> <div data-bbox="514 1060 639 1829" data-label="Text"> <pre># Hackneyed domains I reach thru a local relay CRbitnet cdn uucp # major relay host DRmail-relay.uwo.ca</pre> </div> <div data-bbox="682 1060 807 1829" data-label="Text"> <p>Classes are set to single letters and referenced by the <code>\$=Y</code> 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 <code>mail-relay.uwo.ca</code>.</p> </div> <div data-bbox="849 1060 909 1829" data-label="Text"> <p>Note distinction between <code>\$R</code> (the string) and <code>\$=R</code> (the class). In the example we define a class of pseudo domains</p> </div> <div data-bbox="951 1060 1107 1829" data-label="Text"> <pre>CRbitnet cdn uucp and a string for the name of a gateway machine DRmail-relay.uwo.ca</pre> </div>	<div data-bbox="337 711 365 884" data-label="Section-Header"> <h2>5.5. Headers</h2> </div> <div data-bbox="412 212 472 884" data-label="Text"> <p>Any line beginning with a “H” is a header definition — these are usually fine.</p> </div> <div data-bbox="514 119 1006 884" data-label="Text"> <pre>##### # 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</pre> </div> <div data-bbox="1049 357 1075 884" data-label="Text"> <p>There’s an interesting conditional construction:</p> </div> <div data-bbox="1117 669 1141 884" data-label="Text"> <pre>\$?sfrom \$s \$.</pre> </div> <div data-bbox="1183 212 1242 884" data-label="Text"> <p>This means: if the string ‘s’ is defined then substitute ‘from \$s’.</p> </div>
<div data-bbox="1349 1682 1372 1829" data-label="Page-Footer"> <p>UWO/Sendmail</p> </div>	<div data-bbox="1349 1161 1372 1184" data-label="Page-Footer"> <p>22</p> </div> <div data-bbox="1349 216 1372 363" data-label="Page-Footer"> <p>UWO/Sendmail</p> </div>

<div data-bbox="337 1675 363 1829" data-label="Section-Header"> <h2>5.6. Mailer</h2> </div> <div data-bbox="412 1161 537 1829" data-label="Text"> <p>Any line beginning with a “M” is a mailer definition. This defines how sendmail should communicate with different programs, <i>eg.</i> does the program accept multiple recipients on the same line?</p> </div> <div data-bbox="581 1245 872 1829" data-label="Text"> <pre># # Mailer specifications... # Mlocal, P=/usr/local/lib/binmail, F=rlsXRDFMnP, 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</pre> </div> <div data-bbox="920 1161 1015 1829" data-label="Text"> <p>The IPC mailer (ie. SMTP over TCP/IP) is built in — sendmail takes care of the Inter Process Communication without the aid of any delivery program.</p> </div> <div data-bbox="1062 1161 1120 1829" data-label="Text"> <p>Sendmail uses a program to deliver mail locally — that is not built in.</p> </div> <div data-bbox="1167 1161 1226 1829" data-label="Text"> <p>We, optionally, support a mailer for routing over a UUCP link.</p> </div> <div data-bbox="1349 1682 1370 1829" data-label="Text"> <p>UWO/Sendmail</p> </div> <div data-bbox="1349 1161 1370 1184" data-label="Text"> <p>24</p> </div>	<div data-bbox="337 558 363 884" data-label="Section-Header"> <h2>5.7. Rules and Rule Sets</h2> </div> <div data-bbox="412 216 506 884" data-label="Text"> <p>Any line beginning with an “S” defines a numbered rule set. <i>eg.</i> S0 the message delivery rule set and S3 the preamble rule set.</p> </div> <div data-bbox="550 102 708 884" data-label="Text"> <pre>##### # Re-writing rules are trivial, I either know a # the MX world or, alternatively, know of some # does. Two routes are supported to the smart # smtp/ip or uux/uucp. #</pre> </div> <div data-bbox="751 102 907 884" data-label="Text"> <pre>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</pre> </div> <div data-bbox="989 216 1047 884" data-label="Text"> <p>Certain rule sets are reserved (<i>eg. all of</i> S0, S1, S2, S3 and S4 have special roles).</p> </div> <div data-bbox="1094 216 1153 884" data-label="Text"> <p>Some are required, <i>eg.</i> S0 is required (for obvious reasons). All else can be empty.</p> </div> <div data-bbox="1349 861 1370 884" data-label="Text"> <p>25</p> </div> <div data-bbox="1349 216 1370 363" data-label="Text"> <p>UWO/Sendmail</p> </div>
--	---

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

Rpattern<tab>rewrite-rule<tab>comment

For example, the preamble rule set:

S3

Rlocal!\$*

\$:\$1

to make rmail work

R<>

@\$n@\$j

turn into magic token

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 preferred domain

R\$*@\$*

@\$1@\$2

looks fine

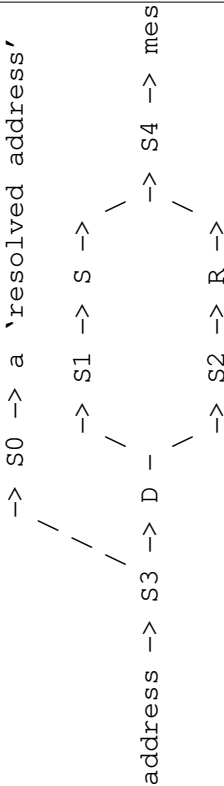
R\$*

@\$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.



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@cunyvum.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’, ‘@’, ‘cunyvum’, ‘.’, ‘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.\$3U
eg. reggers@uwovax

<div data-bbox="337 1308 367 1814">(4) \$x matches the string defined by 'Dx'</div> <div data-bbox="407 1291 435 1740">R\$*@\$j \$#local\$:\$1</div> <div data-bbox="475 1341 503 1740">eg. reggers@julian.uwo.ca</div> <div data-bbox="544 1161 604 1785">(5) \$=x matches a token in the class defined by 'Cx'</div> <div data-bbox="644 1096 672 1711">R\$*@\$*.\$=R \$#ether\$@\$R\$:\$1@\$2.\$3</div> <div data-bbox="712 1362 740 1711">eg. a362@uwoccl1.BITNET</div>	<div data-bbox="337 522 367 882">5.8.4. Rewriting Semantics</div> <div data-bbox="407 212 508 882">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:</div> <div data-bbox="548 212 613 867">(1) \$n is <i>nth</i> patten match in the string (count only the \$ matches).</div> <div data-bbox="654 462 682 795">R@\$*:\$* \$:\$2</div> <div data-bbox="722 212 787 867">(2) \$:rule... apply the rule and procede to the next in rule set.</div> <div data-bbox="828 459 855 795">R<> \$:\$n</div> <div data-bbox="896 281 930 867">(3) \$@rule... apply the rule and exit the rule set.</div> <div data-bbox="971 413 998 795">R\$- \$@\$1@\$j</div> <div data-bbox="1349 1682 1372 1829">UWO/Sendmail</div> <div data-bbox="1349 1161 1372 1184">32</div> <div data-bbox="1349 861 1372 882">33</div> <div data-bbox="1349 212 1372 359">UWO/Sendmail</div>
---	---

<p>(4) \$#mailer... deliver to mailer (only in S0!)</p> <p style="padding-left: 100px;">R\$- \$#local\$: \$1</p> <p>Within a #Mailer rule \$: names the user, \$# names the host. For example</p> <p style="padding-left: 100px;">R\$*@\$. \$. \$=R \$#ether\$@\$R\$: \$1@\$2. \$3</p> <p>The mailer is ether, the host \$R, and the user is \$1@\$2. \$3. This punts pseudo domains to a gateway machine.</p> <p style="padding-left: 100px;">R\$*@\$. \$#ether\$@\$2\$: \$1@\$2</p> <p>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.</p>	<div> <div>6. Sendmail Testing</div> <div> <p>You can test whether an address is deliverable or how a list expands:</p> <pre>[10:43am julian] sendmail -bv reggers@uwovax.uwo.ca... deliverable</pre> <p>You can test rule sets in a verbose mode:</p> <pre>[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: ^Y "ether" ^W "uwovax" "." "uwo" "." ^X "reggers" "@" "uwovax" "." "uwo" "."</pre> <p>Note that Rule Set 3 is always applied. Note also the token sequences displayed as input and output to th the rule sets.</p> </div> </div> <div> <div>UWO/Sendmail</div> <div>34</div> </div> <div> <div>UWO/Sendmail</div> <div>35</div> </div>
---	---

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 exchanger), and other good stuff (the `IN` records are Internet information, other records are possible — 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.

<p>Alternatively ...</p> <pre> [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 > </pre> <p>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.</p> <p>You can mail to sites not on the Internet — by mailing to MX gateways which can forward things on.</p>	<p>Note that some mail addresses mention systems that are not listed in the DNS</p> <pre> [11:28am julian] nslookup Default Server: julian.uwo.ca Address: 129.100.2.12 > set type=any > uwoccl.bitnet Server: julian.uwo.ca Address: 129.100.2.12 *** julian.uwo.ca can't find uwoccl.bitnet: Non-existent domain > watmath.uucp Server: julian.uwo.ca Address: 129.100.2.12 *** julian.uwo.ca can't find watmath.uucp: Non-existent domain </pre>
<p>UWO/Sendmail</p>	<p>41</p> <p>UWO/Sendmail</p>

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

reggers%uwoccl.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).