Switching From procmail To maildrop

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
	2015-07-18		WB

Contents

Last updated 2015-07-18

Available in HTML or PDF. Links to all my articles here. Created with AsciiDoc.

procmail has been around forever, sorting and processing incoming mail in a lot of very useful ways. But it has technical and usage problems that make it ripe for replacement.

1 Choosing maildrop

maildrop stands out among the choices for a procmail replacement. Many of the other choices are strange or obscure or both, and maildrop is the nearest fit in function and compatibility. It's also a popular alternative to procmail, helping to assure future viability. The FreeBSD port is available in */usr/ports/mail/maildrop*.

2 Using maildrop As A Mail Delivery Agent (MDA)

procmail has been around so long that the sendmail option to use a local mail delivery agent is actually named *local_procmail*! But it's just a name, and the feature can use other MDAs. From the Dovecot wiki:

/etc/mail/hostname.mc

```
dnl FEATURE(local_procmail)
FEATURE(`local_procmail', `/usr/local/bin/maildrop', `maildrop -d $u')
```

3 Basic *.mailfilter* Setup

Put a reminder of the maildrop regex options in comments and set a couple of paths to make entry of rules easier.

~/.mailfilter

```
# regex flags, used after the regex: /something/:b
# :h - header
# :b - body
# :D - distinguish between upper and lower case (default is to ignore case)

DEFAULT="$HOME/mail"
```

4 Converting Regular Expressions

maildrop uses the Perl Compatible Regular Expressions library. Most of the time, regexes can be lifted verbatim from *.procmailrc* recipes. Slashes must be escaped in maildrop regexes, but in exchange for that minor inconvenience there are all the simpler and more powerful features of Perl regexes (see `perldoc perlre`).

~/.procmailrc

```
:0
* B ?? http://mail
$DEFAULT/$SPAM
```

becomes

~/.mailfilter

```
if ( /http:\/\/mail/:b )
    to $DEFAULT/$SPAM
```

5 Testing Filter Rules

Test messages can be piped to maildrop in delivery mode to see exactly what it's doing with the *~/.mailfilter* rules. *-V* sets the verbosity level. The steps seem to be "nothing", "too little", and "way too much".

```
% cat testmsg | maildrop -V9 -d
```

6 Filtering Duplicate Messages

Eliminating duplicate messages is similar.

~/.procmailrc

```
# Weed out duplicate messages.
:0 Wh: msgid.lock
| /usr/local/bin/formail -D 8192 $HOME/.msgid.cache
```

~/.mailfilter (from maildropex(7))

```
# weed out duplicate messages
`reformail -D 8192 $HOME/.duplicate.cache`
if ( $RETURNCODE == 0 )
    exit
```

Duplicate messages discarded this way are not logged. Add a manual log command before the exit if logging is desired. Since the clause after the if is not a single statement any more, don't forget the curly brackets. Incidentally, maildrop does not like the first bracket to be on the same line as the if statement. Combine all that to create a log message that's similar to the automatic ones and parsable in the same way.

~/.mailfilter

```
# weed out duplicate messages
`reformail -D 8192 $HOME/.duplicate.cache`
if ( $RETURNCODE == 0 )
{
    log "File: (duplicate)      (${SIZE})\n"
    exit
}
```

7 Folder Filtering

Filtering messages into different folders is, again, quite similar.

~/.procmailrc

```
:0
* ^List-Id:.*freebsd-questions.freebsd.org
$MAILDIR/freebsd-questions
```

~/.mailfilter

```
if (/^List-Id:.*freebsd-questions.freebsd.org/)
    to $MAILDIR/freebsd-questions
```

8 *Maildir* Compatibility

maildrop is compatible with *Maildir* mailboxes. However, if a message is delivered to a *folder* that doesn't exist, maildrop writes it to an *mbox* file instead of creating that folder. The workaround is using maildirmake(1) to create the folder first. maildirmake won't harm an existing folder, so the only downside to calling it before every folder delivery is a usually-invisible *File exists* warning and a little overhead.

9 mailstat

mailstat is a script that comes with procmail to report how many messages have arrived and in what folders. maildrop doesn't have that function, but it's not difficult to parse from a log file.

Logging is off by default, so add a line to *.mailfilter* to create the log file. maildrop creates this file if it's not already present.

~/.mailfilter

```
logfile "$HOME/.mailfilter.log"
```

This Ruby script produces output similar to the old mailstat. It could be more elegant; suggestions welcome.

mailstat

```
#!/usr/bin/env ruby

# mailstat replacement in Ruby
# Warren Block
# Last update 2015-07-18
# no warranties expressed or implied, use at your own risk
# thanks to Alex Stangl for pointing out problems

logname = ENV['LOGNAME']
logfile = "/home/#{logname}/.mailfilter.log"
tmpfile = "#{logfile}.tmp"

exit unless File.exists?(logfile)

abort "**couldn't rename '#{logfile}' to '#{tmpfile}'" unless File.rename(logfile, tmpfile)

Encoding.default_external = Encoding::ASCII_8BIT if defined?(Encoding)

lines = File.new(tmpfile).readlines
lines.select!{|line| line =~ /^File:/}

folders = Hash.new

lines.each do |line|
    line =~ /^File: (\S+?) +\((\d+)\)/
    next unless $1 and $2
    name = File.basename($1)
    size = $2.to_i
    name = "(*Inbox)" if [logname, "Maildir"].include?(name)
    name = $1 if name =~ /^\.(.*)/
    folders[name] = {"size" => 0, "count" => 0} unless folders.has_key?(name)
    folders[name]["size"]  += size
    folders[name]["count"] += 1
end

puts "  Total  Number  Folder"
puts "  -----  ------  ------"
folders.keys.sort.each do |key|
    printf("%7d %7d  %s\n", folders[key]["size"], folders[key]["count"], key)
end

File.delete(tmpfile) if File.exists?(tmpfile)
```

10 Done!

That's it. Converting the filter rules is the hardest part, and it turns out to be fairly easy.

A A Complete Sample *.mailfilter*

This full example does a lot of different things. It can be configured for either *mbox*, *Maildir*, or *mh* mailboxes. For a *Maildir*, it handles prepending a dot to folder names.

Creation of directories for mailing lists is automated, just subscribe. When the first message is received from the new list, the directory is automatically created.

Some sample spam filtering is included. Many of these rules were gathered from various sources over the years, and I'd give credit if I knew the source. Some are my own. Please test on sample mail before unleashing it on the real thing.

`xfilter` is used to run `reformail`, adding a header to messages that are caught by spam detection rules. This makes it much easier to tell which rule actually caught the message.

~/.mailfilter

```
# WB
# .mailfilter for maildrop
# Last update 2011-11-06
# no warranties expressed or implied, use at your own risk

# regex flags, used after the regex: /something/:b
# :h - header
# :b - body
# :D - distinguish between upper and lower case (default is to ignore case)

# configure for mbox, maildir, or mh
TYPE="maildir"


logfile "$HOME/.mailfilter.log"

ECHO="/bin/echo"
MAIL="/usr/bin/mail"
MAILDIRMAKE="/usr/local/bin/maildirmake"
REFORMAIL="/usr/local/bin/reformail"

SPAMHEADER="X-WB-spam:"


MBOX=($TYPE =~ /mbox/)
MAILDIR=($TYPE =~ /maildir/)
MH=($TYPE =~ /mh/)

# use mbox by default
DEFAULT="$HOME/mail"
FOLDERS="$DEFAULT/"
SPAM="${FOLDERS}spam"
```

```
if ( $MAILDIR )
{
    DEFAULT="$HOME/Maildir"
    FOLDERS="$DEFAULT/."
    SPAM="${FOLDERS}spam"
}

if ( $MH )
{
    DEFAULT="| $STORE +inbox"
    FOLDERS="| $STORE +"
}


# filter out duplicate messages
`${REFORMAIL} -D 8192 $HOME/.duplicate.cache`
if ( $RETURNCODE == 0 )
{
    log "File: (duplicate)     ($SIZE)\n"
    exit
}


# sort miscellaneous messages

# let messages from relatives through
if ( /^From:.*crazyrelatives@example.com/ )
    to $DEFAULT


# sort other mail into folders

if ( /^From:.*example.net/
    to ${FOLDERS}example


# handle mailing list messages automatically
if ( /^List-Id:.*<([0-9A-Za-z_\-]+)\.+/ )
{
    LISTNAME="$MATCH1"

    # don't create a folder for Mailman status messages, just deliver them
    if ( $LISTNAME =~ /Mailman/ )
        to $DEFAULT

    if ( $MAILDIR )
    {
        `${MAILDIRMAKE} -f "$LISTNAME" "$DEFAULT"`
        if ( $RETURNCODE == 0 )
        {
            # notify the user when new folders are created
            NEWFOLDERMSG="$LISTNAME list folder created"
            `${ECHO} "$NEWFOLDERMSG" | ${MAIL} -s "$NEWFOLDERMSG" $LOGNAME`
        }
    }
    to ${FOLDERS}$LISTNAME
}


# spam filtering

# make sure the spam directory exists
```

```
if ( $MAILDIR )
    `${MAILDIRMAKE} -f spam "$DEFAULT"`

if ( ! /^((Resent-|Apparently-)?(To|Cc|Bcc)):.*${LOGNAME}/ )
{
    xfilter "${REFORMAIL} -a'$SPAMHEADER mail not addressed to me'"
    to $SPAM
}

if ( $SIZE > 2097152 )
{
    xfilter "${REFORMAIL} -a'$SPAMHEADER size too large'"
    to $SPAM
}

# attachments are in the body, so :b flag
if ( /^Content-type: (audio|application)/:b \
     && /name=.*\.(bat|com|docx|exe|hta|pif|scr|shs|vb[es]|ws[fh]|zip)/:b )
{
    xfilter "${REFORMAIL} -a'$SPAMHEADER potential virus attachment'"
    to $SPAM
}

if ( /^Received:.*-0[67]00 (E[DS]T)/ || \
     /^Received:.*-6000 \(E[DS]T\)/ )
{
    xfilter "${REFORMAIL} -a'$SPAMHEADER fake time stamp'"
    to $SPAM
}

if ( /^(To|From|Reply-To):.*@public.(com|net)/ )
{
    xfilter "${REFORMAIL} -a'$SPAMHEADER public.com or public.net'"
    to $SPAM
}

if ( /^(To|From|Reply-To|Subject):.*(windows-1251)/ )
{
    xfilter "${REFORMAIL} -a'$SPAMHEADER non-English character sets'"
    to $SPAM
}

if ( /^(\/?)http:..((.+(\@|\%40))?)[0-9](\.?)[0-9](\.?)[0-9](\.?)[0-9](\.?)[0-9](\.?) ↩
    [0-9](\.?)[0-9](\.?).*/:b )
{
    xfilter "${REFORMAIL} -a'$SPAMHEADER decimal IP or IP-only URLs in body'"
    to $SPAM
}

if ( /^Message-ID:.*<\ *>/ )
{
    xfilter "${REFORMAIL} -a'$SPAMHEADER empty Message-ID'"
    to $SPAM
}

if ( /^X-Advertise?ment/ || /^X-ADV/ )
{
    xfilter "${REFORMAIL} -a'$SPAMHEADER X-Advertisement header'"
    to $SPAM
}

if ( /^X-Uidl:/ )
```

```
{
    xfilter "${REFORMAIL} -a'$SPAMHEADER phony POP3 UIDL headers'"
    to $SPAM
}

if ( /http:\/\/mail/:b )
{
    xfilter "${REFORMAIL} -a'$SPAMHEADER snowshoe spam with URL like http://mail'"
    to $SPAM
}

if ( /<script src\=/:b || /javascript:window\.open/:b )
{
    xfilter "${REFORMAIL} -a'$SPAMHEADER Javascript in body'"
    to $SPAM
}

# characters with the high bit set in more than five lines
# add one for each time the regex matches
# if the result is "> 5", the condition is true
# skip A0 because it shows up in ordinary text.
if ( /[\xA1-\xFF]+/:b,1,1 > 5 )
{
    xfilter "${REFORMAIL} -a'$SPAMHEADER high-bit characters in more than five lines'"
    to $SPAM
}
```