NAME

mailfilterex − Mailfilter configuration file examples

SYNOPSIS

$HOME/.mailfilterrc examples

DESCRIPTION

For a description of the rcfile format and its keywords see the **mailfilterrc**(5) man page or get a basic set of options from either the INSTALL file or the doc/ directory of the Mailfilter distribution.

This man page contains several configuration examples and real-life use cases for the Mailfilter program.

EXAMPLES

If not stated otherwise, the following examples assume you are using extended Regular Expressions, compared to Mailfilter's default, basic type. General information on Regular Expressions can be found in the **regex**(7) man page or in any good book on UNIX/POSIX. You could also use slightly modified examples from **procmail**(1) if it is available on your system.

Filtering Domains

To create a very restrictive set of filter rules at least two keywords should be used: ALLOW and DENY. DENY could match all messages coming from an annoying public mail service, while ALLOW matches messages from a good friend who also uses this annoying public mailer.

```
DENY = "^From:.*public-mail\.com"
ALLOW = "^From:.*friend@public-mail\.com"
```

These two lines are enough to block all but your friend's e-mail from the public-mail.com domain.

Case Sensivity

In general case-sensivity is controlled by the REG_CASE keyword. Having Mailfilter treat expressions case-insensitive is almost always more efficient.

```
REG_CASE = "no"
DENY = "^Subject:.*win money"
```

In this example Mailfilter would delete all messages with subject lines like 'WIN MONEY', 'Win Money' or any other mix of capital and non-capital characters. REG_CASE makes filters ignore the case.

A more complex set up can be achieved by additionally using the DENY_CASE keyword.

```
DENY_CASE = "^Subject:.*BUSINESS"
```

In this example only e-mails that have 'BUSINESS' in their subject match the filter, even though in general Mailfilter ignores the case. So in this example all messages with 'business' or 'Business' in their subjects would not be affected by this filter.

Such an option is very useful if you are not interested in commercial bulk mail that offers amazing business opportunities, but in all your business partners who contact you by e-mail.

Defining Friends

The keyword ALLOW can be used to override any spam filters. Similar to the earlier example ALLOW defines a 'friend'.

```
ALLOW = "^Subject:.*mailfilter"
```

Adding this rule to the rcfile would mean all messages that contain anything about Mailfilter in their subject lines can pass the spam filters. But even friends tend to send large e-mails sometimes to share their joy about the latest joke that just made the round in their office. In such cases a limit can be defined that affects particularly 'friends'.

```
MAXSIZE_ALLOW = 500000
```

Setting MAXSIZE_ALLOW to 500000 means no message can be larger than 500 kBytes. (Scanned 'office-jokes' are usually around that size.)

Negative Message Filters

In order to create a very restrictive spam protection it can be more useful sometimes to define which e-mails should not be deleted instantly and consequently get rid of messages that can not be matched to this criterion - rather than vice versa. This can be achieved by using negation. The typical use case is looking at the message tags ‘To:’ or ‘Cc:’ of an e-mail.

```
DENY <> "^(To|Cc):.*my-email@address\.com"
```

Having added such a filter to your personal rule set keeps away a lot of spam that is not directly addressed to your e-mail account. Since this is a very aggressive way of filtering, you are well advised to keep your ‘friends list’ up to date. Also note that the above example, using the logical OR operator, works only with extended Regular Expressions.

Scores

Instead of setting up spam filters, it is also possible to define scores which can be accumulated until a certain threshold is reached. This is very useful to delete advertisements on mailing lists, for instance. High-score marks the threshold:

```
HIGHSCORE = 100
SCORE +100 = "^Subject:.*viagra"
SCORE +100 = "^Content-Type:.*html"
SCORE -100 = "^(To|From):.*my_mailing_list"
```

This simple example is useful to delete mails with a score greater than 100, i.e. if someone sends an HTML mail to my_mailing_list, the message will reach score 0. However, should an HTML mail regarding Viagra reach the list, then the message will classify as spam, because it reached an overall score of 100.

The MAXSIZE_SCORE keyword can be used to add to the accumulated score for an e-mail. The following will cause all emails not directly addressed to the recipient and greater than 60000 bytes in size to be deleted (a useful way of rejecting many common MS targeted worms and trojans which can clog up your inbox).

```
HIGHSCORE = 100
MAXSIZE_SCORE +50 = 60000
SCORE +50 <> "^(To|Cc):.*my-email@address\.com"
```

This is a less aggressive way of dealing with e-mail sizes than the using the MAXSIZE_DENY keyword. Note that this example (by using the expression (To|Cc):.*my-email@address\.com) works only with extended Regular Expressions.

General Message Size Limits

It is always a good idea to define a very general size limit for e-mails. Mailfilter uses the keyword MAXSIZE_DENY for that purpose.

```
MAXSIZE_DENY = 200000
```

Setting it to 200 kBytes can save you a couple of hours, depending on how much mail you get everyday. Messages bigger than that get deleted on the server, unless they match any of the ALLOW rules. To achieve maximum efficiency it makes sense to use both MAXSIZE_DENY and MAXSIZE_ALLOW. No one should block up your mail box, no ‘friends’, no others.

A rule of thumb is to be twice as tolerant towards friends than you are towards anonymous people.

Dealing with Duplicates

Most people want to download a message only once, even though it might have been sent to two or three of their accounts at the same time. The simple line

```
DEL_DUPLICATES = "yes"
```

will take care of duplicates and makes sure that only one copy of a message has to be delivered.

Normalisation of Message Subjects

Every now and then some clever sales person comes up with the brilliant idea to wrap spam in funny little characters. If you get a message with a subject line similar to this one ‘,L.E-G,A.L; ,C.A-B‘L‘E, .B-O‘X‘’,

then ordinary filters would fail to detect the junk.

```
NORMAL = "yes"
```

Adding this directive to the rcfile tells Mailfilter to ‘normalise’ subject strings, i.e. leave in only the alphanumeric characters and delete the rest. ‘,L.E-G,A.L; ,C.A-B‘L‘E, .B-O‘X‘’ would then become ‘LEGAL CABLE BOX’ which can easily be matched to a spam filter.

Note that Mailfilter first tries to match the original subject string, before it checks on the normalised one.

Control Mechanism

Since Mailfilter deletes e-mails remotely, before they have to be downloaded into the local machine, it is also important to know what is going on while the program is being executed. The least you should do is define a proper level of verbosity and a log file.

```
LOGFILE = "$HOME/logs/mailfilter-‘date +"%h%y"’"
VERBOSE = 3
```

Level three is the default verbosity level. Using it, Mailfilter reports information on deleted messages, runtime errors and dates to the screen and the log file.

You can use ‘command’ to embedd shell skripts into your path names. In the above example it is used to store log files separately for each month and year.

Extended Regular Expressions

For advanced applications, the basic Regular Expressions are typically not sufficient. If you know the syntax and usage of the extended expressions, it is almost always a good idea to set REG_TYPE accordingly.

```
REG_TYPE = "extended"
```

Extended expressions are more flexible, but also more sensitive towards syntax errors and the like. Examples in this man page all use extended type.

NOTES

If you are new to Regular Expressions and new to Mailfilter, you might want to experiment a bit, before you accidently delete messages for real. For such cases Mailfilter provides two keywords. TEST can be used to only simulate the deletion of messages and SHOW_HEADERS stores all the e-mail headers that get examined by the program.

```
TEST = "yes"
SHOW_HEADERS = "$HOME/logs/mailfilter-headers.txt"
```

Use this setup if you are not yet comfortable with the concept of spam filtering. It may help to understand Regular Expressions better and how to use them.

SEE ALSO

mailfilter(1), **mailfilterrc**(5), **procmailrc**(5), **procmailex**(5), **regex**(7)

COPYRIGHT