NAME

procmailrc − procmail rcfile

SYNOPSIS

$HOME/.procmailrc

DESCRIPTION

For a quick start, see **NOTES** at the end of the **procmail**(1) man page.

The rcfile can contain a mixture of environment variable assignments (some of which have special meanings to procmail), and recipes. In their most simple appearance, the recipes are simply one line regular expressions that are searched for in the header of the arriving mail. The first recipe that matches is used to determine where the mail has to go (usually a file). If processing falls off the end of the rcfile, procmail will deliver the mail to **$DEFAULT**.

There are two kinds of recipes: delivering and non-delivering recipes. If a *delivering recipe* is found to match, procmail considers the mail (you guessed it) delivered and will *cease processing* the rcfile after having successfully executed the action line of the recipe. If a *non-delivering recipe* is found to match, processing of the rcfile will *continue* after the action line of this recipe has been executed.

Delivering recipes are those that cause header and/or body of the mail to be: written into a file, absorbed by a program or forwarded to a mailaddress.

Non-delivering recipes are: those that cause the output of a program or filter to be captured back by procmail or those that start a nesting block.

You can tell procmail to treat a *delivering recipe* as if it were a non-delivering recipe by specifying the ‘c’ flag on such a recipe. This will make procmail generate a *carbon copy* of the mail by delivering it to this recipe, yet continue processing the rcfile.

By using any number of recipes you can presort your mail extremely straightforward into several mailfolders. Bear in mind though that the mail can arrive concurrently in these mailfolders (if several procmail programs happen to run at the same time, not unlikely if a lot of mail arrives). To make sure this does not result in a mess, proper use of lockfiles is highly recommended.

The environment variable **assignments** and **recipes** can be freely intermixed in the rcfile. If any environment variable has a special meaning to procmail, it will be used appropriately the moment it is parsed (i.e., you can change the current directory whenever you want by specifying a new **MAILDIR**, switch lockfiles by specifying a new **LOCKFILE**, change the umask at any time, etc., the possibilities are endless :−).

The assignments and substitutions of these environment variables are handled exactly like in **sh**(1) (that includes all possible quotes and escapes), with the added bonus that blanks around the ’=’ sign are ignored and that, if an environment variable appears without a trailing ’=’, it will be removed from the environment. Any program in backquotes started by procmail will have the entire mail at its stdin.

Comments

A word beginning with # and all the following characters up to a NEWLINE are ignored. This does not apply to condition lines, which cannot be commented.

Recipes

A line starting with ’:’ marks the beginning of a recipe. It has the following format:

:0 [*flags*] [: [*locallockfile*]]
<zero or more conditions (one per line)>
<exactly one action line>

Conditions start with a leading ‘*’, everything after that character is passed on to the internal egrep **literally**, except for leading and trailing whitespace. These regular expressions are **completely** compatible to the normal **egrep**(1) extended regular expressions. See also **Extended regular expressions**.

Conditions are anded; if there are no conditions the result will be true by default.

Flags can be any of the following:

H Egrep the header (default).

B Egrep the body.

D Tell the internal egrep to distinguish between upper and lower case (contrary to the default which is to ignore case).

A This recipe will not be executed unless the conditions on the last preceding recipe (on the current block-nesting level) without the 'A' or 'a' flag matched as well. This allows you to chain actions that depend on a common condition.

a Has the same meaning as the 'A' flag, with the additional condition that the immediately preceding recipe must have been *successfully* completed before this recipe is executed.

E This recipe only executes if the immediately preceding recipe was not executed. Execution of this recipe also disables any immediately following recipes with the 'E' flag. This allows you to specify 'else if' actions.

e This recipe only executes if the immediately preceding recipe *failed* (i.e., the action line was attempted, but resulted in an error).

h Feed the header to the pipe, file or mail destination (default).

b Feed the body to the pipe, file or mail destination (default).

f Consider the pipe as a filter.

c Generate a **carbon copy** of this mail. This only makes sense on *delivering* recipes. The only non-delivering recipe this flag has an effect on is on a nesting block, in order to generate a carbon copy this will **clone** the running procmail process (lockfiles will not be inherited), whereby the clone will proceed as usual and the parent will jump across the block.

w Wait for the filter or program to finish and check its exitcode (normally ignored); if the filter is unsuccessful, then the text will not have been filtered.

W Has the same meaning as the 'w' flag, but will suppress any 'Program failure' message.

i Ignore any write errors on this recipe (i.e., usually due to an early closed pipe).

r Raw mode, do not try to ensure the mail ends with an empty line, write it out as is.

There are some special conditions you can use that are not straight regular expressions. To select them, the condition must start with:

! Invert the condition.

$ Evaluate the remainder of this condition according to **sh**(1) substitution rules inside double quotes, skip leading whitespace, then reparse it.

? Use the exitcode of the specified program.

< Check if the total length of the mail is shorter than the specified (in decimal) number of bytes.

> Analogous to '<'.

variablename *??*

Match the remainder of this condition against the value of this environment variable (which cannot be a pseudo variable). A special case is if variablename is equal to 'B', 'H', 'HB' or 'BH'; this merely overrides the default header/body search area defined by the initial flags on this recipe.

\ To quote any of the above at the start of the line.

Local lockfile

If you put a second (trailing) ':' on the first recipe line, then procmail will use a *locallockfile* (for this recipe only). You can optionally specify the locallockfile to use; if you don't however, procmail will use the destination filename (or the filename following the first '>>') and will append $LOCKEXT to it.

Recipe action line

The action line can start with the following characters:

! Forwards to all the specified mail addresses.

| Starts the specified program, possibly in $SHELL if any of the characters $SHELLMETAS are spotted. You can optionally prepend this pipe symbol with *variable*=, which will cause stdout of the program to be captured in the environment *variable* (procmail will **not** terminate processing the rcfile at this point). If you specify just this pipe symbol, without any program, then procmail will pipe the mail to stdout.

{ Followed by at least one space, tab or newline will mark the start of a nesting block. Everything up till the next closing brace will depend on the conditions specified for this recipe. Unlimited nesting is permitted. The closing brace exists merely to delimit the block, it will *not* cause procmail to terminate in any way. If the end of a block is reached processing will continue as usual after the block. On a nesting block, the flags 'H' and 'B' only affect the conditions leading up to the block, the flags 'h' and 'b' have no effect whatsoever.

Anything else will be taken as a mailbox name (either a filename or a directory, absolute or relative to the current directory (see MAILDIR)). If it is a (possibly yet nonexistent) filename, the mail will be appended to it.

If it is a directory, the mail will be delivered to a newly created, guaranteed to be unique file named $MSGPREFIX* in the specified directory. If the mailbox name ends in "/.", then this directory is presumed to be an MH folder; i.e., procmail will use the next number it finds available. If the mailbox name ends in "/", then this directory is presumed to be a maildir folder; i.e., procmail will deliver the message to a file in a subdirectory named "tmp" and rename it to be inside a subdirectory named "new". If the mailbox is specified to be an MH folder or maildir folder, procmail will create the necessary directories if they don't exist, rather than treat the mailbox as a non-existent filename. When procmail is delivering to directories, you can specify multiple directories to deliver to (procmail will do so utilising hardlinks).

Environment variable defaults

LOGNAME, HOME and SHELL	Your (the recipient's) defaults
PATH	$HOME/bin:/bin:/usr/bin:/sbin:/usr/sbin:/usr/local/bin :/usr/X11R6/bin (Except during the processing of an /etc/procmailrc file, when it will be set to '/bin:/usr/bin'.)
SHELLMETAS	&|<>~;?*[
SHELLFLAGS	-c
ORGMAIL	/var/spool/mail/$LOGNAME (Unless **−m** has been specified, in which case it is unset)
MAILDIR	$HOME (Unless the name of the first successfully opened rcfile starts with './' or if **−m** has been specified, in which case it defaults to '.')

DEFAULT	$ORGMAIL
MSGPREFIX	msg.
SENDMAIL	/usr/sbin/sendmail
SENDMAILFLAGS	-oi
HOST	The current hostname
COMSAT	no (If an rcfile is specified on the command line)
PROCMAIL_VERSION	3.22
LOCKEXT	.lock

Other cleared or preset environment variables are IFS, ENV and PWD.

For security reasons, upon startup procmail will wipe out all environment variables that are suspected of modifying the behavior of the runtime linker.

Environment

Before you get lost in the multitude of environment variables, keep in mind that all of them have reasonable defaults.

MAILDIR Current directory while procmail is executing (that means that all paths are relative to $MAILDIR).

DEFAULT Default **mailbox** file (if not told otherwise, procmail will dump mail in this mailbox). Procmail will automatically use $DEFAULT$LOCKEXT as lockfile prior to writing to this mailbox. You do not need to set this variable, since it already points to the standard system mailbox.

LOGFILE This file will also contain any error or diagnostic messages from procmail (normally none :−) or any other programs started by procmail. If this file is not specified, any diagnostics or error messages will be mailed back to the sender. See also **LOGABSTRACT**.

VERBOSE You can turn on *extended diagnostics* by setting this variable to ‘yes’ or ‘on’, to turn it off again set it to ‘no’ or ‘off’.

LOGABSTRACT Just before procmail exits it logs an abstract of the delivered message in $LOGFILE showing the ‘From ’ and ‘Subject:’ fields of the header, what folder it finally went to and how long (in bytes) the message was. By setting this variable to ‘no’, generation of this abstract is suppressed. If you set it to ‘all’, procmail will log an abstract for every successful *delivering recipe* it processes.

LOG Anything assigned to this variable will be appended to $LOGFILE.

ORGMAIL Usually the system mailbox (**OR**i**G**inal **MAIL**box). If, for some obscure reason (like ‘**filesystem full**’) the mail could not be delivered, then this mailbox will be the last resort. If procmail fails to save the mail in here (deep, deep trouble :−), then the mail will bounce back to the sender.

LOCKFILE Global semaphore file. If this file already exists, procmail will wait until it has gone before proceeding, and will create it itself (cleaning it up when ready, of course). If more than one *lockfile* are specified, then the previous one will be removed before trying to create the new one. The use of a global lockfile is discouraged, whenever possible use locallockfiles (on a per recipe basis) instead.

LOCKEXT Default extension that is appended to a destination file to determine what local *lockfile* to use (only if turned on, on a per-recipe basis).

LOCKSLEEP Number of seconds procmail will sleep before retrying on a *lockfile* (if it already existed); if not specified, it defaults to 8 seconds.

LOCKTIMEOUT Number of seconds that have to have passed since a *lockfile* was last modified/created before procmail decides that this must be an erroneously leftover lockfile that can be removed by force now. If zero, then no timeout will be used and procmail will wait forever until the lockfile is removed; if not specified, it defaults to 1024 seconds. This variable is useful to prevent indefinite hangups of **sendmail**/procmail. Procmail is immune to clock skew across machines.

TIMEOUT Number of seconds that have to have passed before procmail decides that some child it started must be hanging. The offending program will receive a TERMINATE signal from procmail, and processing of the rcfile will continue. If zero, then no timeout will be used and procmail will wait forever until the child has terminated; if not specified, it defaults to 960 seconds.

MSGPREFIX Filename prefix that is used when delivering to a directory (not used when delivering to a maildir or an MH directory).

HOST If this is not the *hostname* of the machine, processing of the current *rcfile* will immediately cease. If other rcfiles were specified on the command line, processing will continue with the next one. If all rcfiles are exhausted, the program will terminate, but will not generate an error (i.e., to the mailer it will seem that the mail has been delivered).

UMASK The name says it all (if it doesn't, then forget about this one :−). Anything assigned to UMASK is taken as an **octal** number. If not specified, the umask defaults to 077. If the umask permits o+x, all the mailboxes procmail delivers to directly will receive an o+x mode change. This can be used to check if new mail arrived.

SHELLMETAS If any of the characters in SHELLMETAS appears in the line specifying a filter or program, the line will be fed to $SHELL instead of being executed directly.

SHELLFLAGS Any invocation of $SHELL will be like:
"$SHELL" "$SHELLFLAGS" "$*";

SENDMAIL If you're not using the *forwarding* facility don't worry about this one. It specifies the program being called to forward any mail.
It gets invoked as: "$SENDMAIL" $SENDMAILFLAGS "$@";

NORESRETRY Number of retries that are to be made if any '**process table full**', '**file table full**', '**out of memory**' or '**out of swap space**' error should occur. If this number is negative, then procmail will retry indefinitely; if not specified, it defaults to 4 times. The retries occur with a $SUSPEND second interval. The idea behind this is that if, e.g., the *swap space* has been exhausted or the *process table* is full, usually several other programs will either detect this as well and abort or crash 8−), thereby freeing valuable *resources* for procmail.

SUSPEND Number of seconds that procmail will pause if it has to wait for something that is currently unavailable (memory, fork, etc.); if not specified, it will default to 16 seconds. See also: **LOCKSLEEP**.

LINEBUF Length of the internal line buffers, cannot be set smaller than 128. All lines read from the *rcfile* should not exceed $LINEBUF characters before and after expansion. If not specified, it defaults to 2048. This limit, of course, does *not* apply to the mail itself, which can have arbitrary line lengths, or could be a binary file for that matter. See also PROCMAIL_OVERFLOW.

DELIVERED If set to 'yes' procmail will pretend (to the mail agent) the mail has been delivered. If mail cannot be delivered after having met this assignment (set to 'yes'), the mail will be lost (i.e., it will not bounce).

TRAP When procmail terminates of its own accord and not because it received a signal, it will execute the contents of this variable. A copy of the mail can be read from stdin. Any output produced by this command will be appended to $LOGFILE. Possible uses for TRAP are: removal of temporary files, logging customised abstracts, etc. See also **EXITCODE** and **LOGABSTRACT**.

EXITCODE By default, procmail returns an exitcode of zero (success) if it successfully delivered the message or if the **HOST** variable was misset and there were no more rcfiles on the command line; otherwise it returns failure. Before doing so, procmail examines the value of this variable. If it is set to a positive numeric value, procmail will instead use that value as its exitcode. If this variable is set but empty and **TRAP** is set, procmail will set the exitcode to whatever the **TRAP** program returns. If this variable is not set, procmail will set it shortly before calling up the **TRAP** program.

LASTFOLDER This variable is assigned to by procmail whenever it is delivering to a folder or program. It always contains the name of the last file (or program) procmail delivered to. If the last delivery was to several directory folders together then $LASTFOLDER will contain the hardlinked filenames as a space separated list.

MATCH This variable is assigned to by procmail whenever it is told to extract text from a matching regular expression. It will contain all text matching the regular expression past the '\' token.

SHIFT Assigning a positive value to this variable has the same effect as the 'shift' command in **sh**(1). This command is most useful to extract extra arguments passed to procmail when acting as a generic mailfilter.

INCLUDERC Names an rcfile (relative to the current directory) which will be included here as if it were part of the current rcfile. Nesting is permitted and only limited by systems resources (memory and file descriptors). As no checking is done on the permissions or ownership of the rcfile, users of **INCLUDERC** should make sure that only trusted users have write access to the included rcfile or the directory it is in. Command line assignments to **INCLUDERC** have no effect.

SWITCHRC Names an rcfile (relative to the current directory) to which processing will be switched. If the named rcfile doesn't exist or is not a normal file or /dev/null then an error will be logged and processing will continue in the current rcfile. Otherwise, processing of the current rcfile will be aborted and the named rcfile started. Unsetting **SWITCHRC** aborts processing of the current rcfile as if it had ended at the assignment. As with **INCLUDERC**, no checking is done on the permissions or ownership of the rcfile and command line assignments have no effect.

PROCMAIL_VERSION
The version number of the running procmail binary.

PROCMAIL_OVERFLOW
This variable will be set to a non-empty value if procmail detects a buffer overflow. See the **BUGS** section below for other details of operation when overflow occurs.

COMSAT **Comsat**(8)/**biff**(1) notification is on by default, it can be turned off by setting this variable to 'no'. Alternatively the biff-service can be customised by setting it to either 'service@', '@hostname', or 'service@hostname'. When not specified it defaults to biff@localhost.

DROPPRIVS If set to 'yes' procmail will drop all privileges it might have had (suid or sgid). This is only useful if you want to guarantee that the bottom half of the /etc/procmailrc file is executed on behalf of the recipient.

Extended regular expressions

The following tokens are known to both the procmail internal egrep and the standard **egrep**(1) (beware that some egrep implementations include other non-standard extensions):

^ Start of a line.

$ End of a line.

. Any character except a newline.

a* Any sequence of zero or more a's.

a+ Any sequence of one or more a's.

a? Either zero or one a.

[^-a-d] Any character which is **not** either a dash, a, b, c, d or newline.

de|abc Either the sequence 'de' or 'abc'.

(abc)* Zero or more times the sequence 'abc'.

\. Matches a single dot; use \ to quote any of the magic characters to get rid of their special meaning. See also $\ variable substitution.

These were only samples, of course, any more complex combination is valid as well.

The following token meanings are special procmail extensions:

^ or **$** Match a newline (for multiline matches).

^^ Anchor the expression at the very start of the search area, or if encountered at the end of the expression, anchor it at the very end of the search area.

\< or \> Match the character before or after a word. They are merely a shorthand for '[^a-zA-Z0-9_]', but can also match newlines. Since they match actual characters, they are only suitable to delimit words, not to delimit inter-word space.

\/ Splits the expression in two parts. Everything matching the right part will be assigned to the MATCH environment variable.

EXAMPLES

Look in the **procmailex**(5) man page.

CAVEATS

Continued lines in an action line that specifies a program always have to end in a backslash, even if the underlying shell would not need or want the backslash to indicate continuation. This is due to the two pass parsing process needed (first procmail, then the shell (or not, depending on **SHELLMETAS**)).

Don't put comments on the regular expression condition lines in a recipe, these lines are fed to the internal egrep *literally* (except for continuation backslashes at the end of a line).

Leading whitespace on continued regular expression condition lines is usually ignored (so that they can be indented), but **not** on continued condition lines that are evaluated according to the **sh**(1) substitution rules inside double quotes.

Watch out for deadlocks when doing unhealthy things like forwarding mail to your own account. Deadlocks can be broken by proper use of **LOCKTIMEOUT**.

Any default values that procmail has for some environment variables will **always** override the ones that were already defined. If you really want to override the defaults, you either have to put them in the **rcfile** or on the command line as arguments.

The /etc/procmailrc file cannot change the PATH setting seen by user rcfiles as the value is reset when procmail finishes the /etc/procmailrc file. While future enhancements are expected in this area, recompiling procmail with the desired value is currently the only correct solution.

Environment variables set **inside** the shell-interpreted-'|' action part of a recipe will **not** retain their value after the recipe has finished since they are set in a subshell of procmail. To make sure the value of an environment variable is retained you have to put the assignment to the variable before the leading '|' of a recipe, so that it can capture stdout of the program.

If you specify only a 'h' or a 'b' flag on a delivering recipe, and the recipe matches, then, unless the 'c' flag is present as well, the body respectively the header of the mail will be silently lost.

SEE ALSO

procmail(1), **procmailsc**(5), **procmailex**(5), **sh**(1), **csh**(1), **mail**(1), **mailx**(1), **binmail**(1), **uucp**(1), **aliases**(5), **sendmail**(8), **egrep**(1), **regexp**(5), **grep**(1), **biff**(1), **comsat**(8), **lockfile**(1), **formail**(1)

BUGS

The only substitutions of environment variables that can be handled by procmail itself are of the type $name, ${name}, ${name:-text}, ${name:+text}, ${name-text}, ${name+text}, $\name, $#, $n, $$, $?, $_, $− and $=; whereby $\name will be substituted by the all-magic-regular-expression-characters-disarmed equivalent of $name, $_ by the name of the current rcfile, $− by $LASTFOLDER and $= will contain the score of the last recipe. Furthermore, the result of $\name substitution will never be split on whitespace. When the **−a** or **−m** options are used, $# will expand to the number of arguments so specified and "$@" (the quotes are required) will expand to the specified arguments. However, "$@" will only be expanded when used in the argument list to a program, and then only one such occurrence will be expanded.

Unquoted variable expansions performed by procmail are always split on space, tab, and newline characters; the IFS variable is not used internally.

Procmail does not support the expansion of '~'.

A line buffer of length $LINEBUF is used when processing the *rcfile*, any expansions that don't fit within this limit will be truncated and PROCMAIL_OVERFLOW will be set. If the overflowing line is a condition or an action line, then it will be considered failed and procmail will continue processing. If it is a variable assignment or recipe start line then procmail will abort the entire rcfile.

If the global lockfile has a *relative* path, and the current directory is not the same as when the global lockfile was created, then the global lockfile will not be removed if procmail exits at that point (remedy: use *absolute* paths to specify global lockfiles).

If an rcfile has a *relative* path and when the rcfile is first opened **MAILDIR** contains a relative path, and if

at one point procmail is instructed to clone itself and the current directory has changed since the rcfile was opened, then procmail will not be able to clone itself (remedy: use an *absolute* path to reference the rcfile or make sure MAILDIR contains an absolute path as the rcfile is opened).

A locallockfile on the recipe that marks the start of a non-forking nested block does not work as expected.

When capturing stdout from a recipe into an environment variable, exactly one trailing newline will be stripped.

Some non-optimal and non-obvious regexps set MATCH to an incorrect value. The regexp can be made to work by removing one or more unneeded '*', '+', or '?' operator on the left-hand side of the \/ token.

MISCELLANEOUS

If the regular expression contains '**^TO_**' it will be substituted by '**(^((Original-)?(Resent-)?(To|Cc|Bcc)|(X-Envelope|Apparently(-Resent)?)-To):(.*[^-a-zA-Z0-9_.])?)**', which should catch all destination specifications containing a specific *address*.

If the regular expression contains '**^TO**' it will be substituted by '**(^((Original-)?(Resent-)?(To|Cc|Bcc)|(X-Envelope|Apparently(-Resent)?)-To):(.*[^a-zA-Z])?)**', which should catch all destination specifications containing a specific *word*.

If the regular expression contains '**^FROM_DAEMON**' it will be substituted by '**(^(Mailing-List:|Precedence:.*(junk|bulk|list)|To: Multiple recipients of |(((Resent-)?(From|Sender)|X-Envelope-From):|>?From)([^>]*[^(.%@a-z0-9])?(Post(ma?(st(e?r)?|n)|office)|(send)?Mail(er)?|daemon|m(mdf|ajordomo)|n?uucp|LIST(SERV|proc)|NETSERV|o(wner|ps)|r(e(quest|sponse)|oot)|b(ounce|bs\.smtp)|echo|mirror|s(erv(ices?|er)|mtp(error)?|ystem)|A(dmin(istrator)?|MMGR|utoanswer))(([^).!:a-z0-9][-_a-z0-9]*)?[%@>\t][^<)]*(\(.*\).*)?)?$([^>]|$)))**', which should catch mails coming from most daemons (how's that for a regular expression :−).

If the regular expression contains '**^FROM_MAILER**' it will be substituted by '**(^(((Resent-)?(From|Sender)|X-Envelope-From):|>?From)([^>]*[^(.%@a-z0-9])?(Post(ma(st(er)?|n)|office)|(send)?Mail(er)?|daemon|mmdf|n?uucp|ops|r(esponse|oot)|(bbs\.)?smtp(error)?|s(erv(ices?|er)|ystem)|A(dmin(istrator)?|MMGR))(([^).!:a-z0-9][-_a-z0-9]*)?[%@>\t][^<)]*(\(.*\).*)?)?$([^>]|$))**' (a stripped down version of '**^FROM_DAEMON**'), which should catch mails coming from most mailer-daemons.

When assigning boolean values to variables like VERBOSE, DELIVERED or COMSAT, procmail accepts as true every string starting with: a non-zero value, 'on', 'y', 't' or 'e'. False is every string starting with: a zero value, 'off', 'n', 'f' or 'd'.

If the action line of a recipe specifies a program, a sole backslash-newline pair in it on an otherwise empty line will be converted into a newline.

The regular expression engine built into procmail does not support named character classes.

NOTES

Since unquoted leading whitespace is generally ignored in the rcfile you can indent everything to taste.

The leading '|' on the action line to specify a program or filter is stripped before checking for $SHELLMETAS.

Files included with the INCLUDERC directive containing only environment variable assignments can be shared with sh.

The current behavior of assignments on the command line to **INCLUDERC** and **SWITCHRC** is not guaranteed, has been changed once already, and may be changed again or removed in future releases.

For *really* complicated processing you can even consider calling **procmail** recursively.

In the old days, the ':0' that marks the beginning of a recipe, had to be changed to ':n', whereby 'n' denotes the number of conditions that follow.

AUTHORS

Stephen R. van den Berg
<srb@cuci.nl>
Philip A. Guenther
<guenther@sendmail.com>