NAME

logrotate - rotates, compresses, and mails system logs

SYNOPSIS

logrotate [**−−force**] [**−−debug**] [**−−state** *file*] [**−−skip-state-lock**] [**−−verbose**] [**−−log** *file*] [**−−mail** *command*] *config_file* [*config_file2 ...*]

DESCRIPTION

logrotate is designed to ease administration of systems that generate large numbers of log files. It allows automatic rotation, compression, removal, and mailing of log files. Each log file may be handled daily, weekly, monthly, or when it grows too large.

Normally, **logrotate** is run as a daily cron job. It will not modify a log more than once in one day unless the criterion for that log is based on the log's size and **logrotate** is being run more than once each day, or unless the **−f** or **−−force** option is used.

Any number of config files may be given on the command line. Later config files may override the options given in earlier files, so the order in which the **logrotate** config files are listed is important. Normally, a single config file which includes any other config files which are needed should be used. See below for more information on how to use the **include** directive to accomplish this. If a directory is given on the command line, every file in that directory is used as a config file.

If no command line arguments are given, **logrotate** will print version and copyright information, along with a short usage summary. If any errors occur while rotating logs, **logrotate** will exit with non-zero status, although the state file will be updated.

OPTIONS

−f, **−−force**

Tells **logrotate** to force the rotation, even if it doesn't think this is necessary. Sometimes this is useful after adding new entries to a **logrotate** config file, or if old log files have been removed by hand, as the new files will be created, and logging will continue correctly.

−d, **−−debug**

Turn on debug mode, which means that no changes are made to the logs and the **logrotate** state file is not updated. Only debug messages are printed.

−s, **−−state** *statefile*

Tells **logrotate** to use an alternate state file. This is useful if **logrotate** is being run as a different user for various sets of log files. To prevent parallel execution **logrotate** by default acquires a lock on the state file, if it cannot be acquired **logrotate** will exit with value 3. The default state file is */var/lib/logrotate/status*. If*/de v/null* is given as the state file, then **logrotate** will not try to write the state file.

−−skip-state-lock

Do not lock the state file, for example if locking is unsupported or prohibited.

−v, **−−verbose**

Turns on verbose mode, for example to display messages during rotation.

−l, **−−log** *file*

Tells **logrotate** to log verbose output into the log_file. The verbose output logged to that file is the same as when running **logrotate** with **-v** switch. The log file is overwritten on every **logrotate** execution.

−m, **−−mail** *command*
Tells **logrotate** which command to use when mailing logs. This command should accept the following arguments:

1) the subject of the message given with '-s subject'
2) the recipient.

The command must then read a message on standard input and mail it to the recipient. The default mail command is */usr/bin/mail*.

−−usage
Prints a short usage message.

−?, **−−help**
Prints help message.

−−version
Display version information.

CONFIGURATION FILE

logrotate reads everything about the log files it should be handling from the series of configuration files specified on the command line. Each configuration file can set global options (local definitions override global ones, and later definitions override earlier ones) and specify logfiles to rotate. Global options do not affect preceding include directives. A simple configuration file looks like this:

```
# sample logrotate configuration file
compress

/var/log/messages {
   rotate 5
   weekly
   postrotate
      /usr/bin/killall −HUP syslogd
   endscript
}

"/var/log/httpd/access.log" /var/log/httpd/error.log {
   rotate 5
   mail recipient@example.org
   size 100k
   sharedscripts
   postrotate
      /usr/bin/killall −HUP httpd
   endscript
}

/var/log/news/* {
   monthly
   rotate 2
   olddir /var/log/news/old
   missingok
   sharedscripts
   postrotate
```

```
        kill -HUP $(cat /var/run/inn.pid)
    endscript
    nocompress
}

~/log/*.log {}
```

The first few lines set global options; in the example, logs are compressed after they are rotated. Note that comments may appear anywhere in the config file as long as the first non-whitespace character on the line is a **#**.

Values are separated from directives by whitespace and/or an optional =. Numbers must be specified in a format understood by **strtoul**(3).

The next section of the config file defines how to handle the log file */var/log/messages*. The log will go through five weekly rotations before being removed. After the log file has been rotated (but before the old version of the log has been compressed), the command */usr/bin/killall -HUP syslogd* will be executed.

The next section defines the parameters for both */var/log/httpd/access.log* and */var/log/httpd/error.log*. Each is rotated whenever it grows over 100 kilobytes in size, and the old logs files are mailed (uncompressed) to recipient@example.org after going through 5 rotations, rather than being removed. The**shar edscripts** means that the **postrotate** script will only be run once (after the old logs have been compressed), not once for each log which is rotated. Note that log file names may be enclosed in quotes (and that quotes are required if the name contains spaces). Normal shell quoting rules apply, with **'**, **"**, and \ characters supported.

The next section defines the parameters for all of the files in */var/log/news*. Each file is rotated on a monthly basis.

The last section uses tilde expansion to rotate log files in the home directory of the current user. This is only available, if your glob library supports tilde expansion. GNU glob does support this.

Please use wildcards with caution. If you specify *, **logrotate** will rotate all files, including previously rotated ones. A way around this is to use the **olddir** directive or a more exact wildcard (such as *.log).

Please note, by default when using **systemd**(1), the option *ProtectSystem=full* is set in the *logrotate.service* file. This prevents **logrotate** from modifying logs in */etc* and */usr*.

Here is more information on the directives which may be included in a **logrotate** configuration file:

CONFIGURATION FILE DIRECTIVES

These directives may be included in a **logrotate** configuration file:

Rotation

rotate *count*

Log files are rotated *count* times before being removed or mailed to the address specified in a **mail** directive. If*count* is 0, old v ersions are removed rather than rotated. If *count* is -1, old logs are not removed at all, except they are affected by **maxage** (use with caution, may waste performance and disk space). Default is 0.

olddir *directory*
Logs are moved into *directory* for rotation. The *directory* must be on the same physical device as the log file being rotated, unless **copy**, **copytruncate** or **renamecopy** option is used. The *directory* is assumed to be relative to the directory holding the log file unless an absolute path name is specified. When this option is used all old versions of the log end up in *directory*. This option may be overridden by the **noolddir** option.

noolddir
Logs are rotated in the directory they normally reside in (this overrides the **olddir** option).

su *user group*
Rotate log files set under this user and group instead of using default user/group (usually root). *user* specifies the user used for rotation and *group* specifies the group used for rotation (see the section **USER AND GROUP** for details). If the user/group you specify here does not have sufficient privilege to make files with the ownership you've specified in a **create** directive, it will cause an error. If**logr otate** runs with root privileges, it is recommended to use the **su** directive to rotate files in directories that are directly or indirectly in control of non-privileged users.

Frequency

hourly Log files are rotated every hour. Note that usually**logr otate** is configured to be run by cron daily (or by *logrotate.timer* when using **systemd**(1)). You have to change this configuration and run **logrotate** hourly to be able to really rotate logs hourly.

daily Log files are rotated every day.

weekly [*weekday*]
Log files are rotated once each *weekday*, or if the date is advanced by at least 7 days since the last rotation (while ignoring the exact time). The *weekday* interpretation is following: 0 means Sunday, 1 means Monday, ..., 6 means Saturday; the special value 7 means each 7 days, irrespectively of weekday. Defaults to 0 if the *weekday* argument is omitted.

monthly
Log files are rotated the first time **logrotate** is run in a month (this is normally on the first day of the month).

yearly Log files are rotated if the current year is not the same as the last rotation.

size *size*
Log files are rotated only if they grow bigger than *size* bytes. If *size* is followed by *k*, the size is assumed to be in kilobytes. If *M* is used, the size is in megabytes, and if *G* is used, the size is in gigabytes. So *size 100*, *size 100k*, *size 100M* and *size 100G* are all valid. This option is mutually exclusive with the time interval options, and it causes log files to be rotated without regard for the last rotation time, if specified after the time criteria (the last specified option takes the precedence).

File selection

missingok
If the log file is missing, go on to the next one without issuing an error message. See also **nomissingok**.

nomissingok
If a log file does not exist, issue an error. This is the default.

ifempty
Rotate the log file even if it is empty, overriding the **notifempty** option (**ifempty** is the default).

notifempty
Do not rotate the log if it is empty (this overrides the **ifempty** option).

minage *count*
Do not rotate logs which are less than <count> days old.

maxage *count*
Remove rotated logs older than <count> days. The age is only checked if the logfile is to be rotated. **rotate −1** does not hinder removal. The files are mailed to the configured address if**mail-last** and **mail** are configured.

minsize *size*
Log files are rotated when they grow bigger than *size* bytes, but not before the additionally specified time interval (**daily**, **weekly**, **monthly**, or **yearly**). The related**size** option is similar e xcept that it is mutually exclusive with the time interval options, and it causes log files to be rotated without regard for the last rotation time, if specified after the time criteria (the last specified option takes the precedence). When **minsize** is used, both the size and timestamp of a log file are considered.

maxsize *size*
Log files are rotated when they grow bigger than *size* bytes even before the additionally specified time interval (**daily**, **weekly**, **monthly**, or **yearly**). The related**size** option is similar e xcept that it is mutually exclusive with the time interval options, and it causes log files to be rotated without regard for the last rotation time, if specified after the time criteria (the last specified option takes the precedence). When**maxsize** is used, both the size and timestamp of a log file are considered.

tabooext [+] *list*
The current taboo extension list is changed (see the **include** directive for information on the taboo extensions). If a + precedes the list of extensions, the current taboo extension list is augmented, otherwise it is replaced. At startup, the taboo extension list *,v*, *.cfsaved*, *.disabled*, *.dpkg−bak*, *.dpkg−del*, *.dpkg−dist*, *.dpkg−new*, *.dpkg−old*, *.rhn−cfg−tmp−**, *.rpmnew*, *.rpmorig*, *.rpmsave*, *.swp*, *.ucf−dist*, *.ucf−new*, *.ucf−old*, *~*

taboopat [+] *list*
The current taboo glob pattern list is changed (see the **include** directive for information on the taboo extensions and patterns). If a + precedes the list of patterns, the current taboo pattern list is augmented, otherwise it is replaced. At startup, the taboo pattern list is empty.

Files and Folders

create *mode owner group*, **create** *owner group*
Immediately after rotation (before the **postrotate** script is run) the log file is created (with the same name as the log file just rotated). *mode* specifies the mode for the log file in octal (the same as **chmod**(2)), *owner* specifies the user who will own the log file, and *group* specifies the group the log file will belong to (see the section **USER AND GROUP** for details). Any of the log file attributes may be omitted, in which case those attributes for the new file will use the same values as

the original log file for the omitted attributes. This option can be disabled using the**nocr eate** option.

nocreate
New log files are not created (this overrides the **create** option).

createolddir *mode owner group*
If the directory specified by **olddir** directive does not exist, it is created. *mode* specifies the mode for the **olddir** directory in octal (the same as **chmod**(2)), *owner* specifies the user who will own the **olddir** directory, and *group* specifies the group the **olddir** directory will belong to (see the section **USER AND GROUP**
for details). This option can be disabled using the **nocreateolddir** option.

nocreateolddir
olddir directory is not created by **logrotate** when it does not exist.

copy Make a copy of the log file, but don't change the original at all. This option can be used, for instance, to make a snapshot of the current log file, or when some other utility needs to truncate or parse the file. When this option is used, the **create** option will have no effect, as the old log file stays in place. The **copy** option allows storing rotated log files on the different devices using **olddir** directive.

nocopy Do not copy the original log file and leave it in place. (this overrides the **copy** option).

copytruncate
Truncate the original log file to zero size in place after creating a copy, instead of moving the old log file and optionally creating a new one. It can be used when some program cannot be told to close its logfile and thus might continue writing (appending) to the previous log file forever. Note that there is a very small time slice between copying the file and truncating it, so some logging data might be lost. When this option is used, the **create** option will have no effect, as the old log file stays in place. The **copytruncate** option allows storing rotated log files on the different devices using **olddir** directive. The**copytruncate** option implies **nor enamecopy**.

nocopytruncate
Do not truncate the original log file in place after creating a copy (this overrides the **copytruncate** option).

renamecopy
Log file is renamed to temporary filename in the same directory by adding ".tmp" extension to it. After that, **postrotate** script is run and log file is copied from temporary filename to final filename. In the end, temporary filename is removed. The**r enamecopy** option allows storing rotated log files on the different devices using **olddir** directive. The**r enamecopy** option implies **nocopytruncate**.

norenamecopy
Do not rename and copy the original log file (this overrides the **renamecopy** option).

shred Delete log files using **shred** −u instead of unlink(). This should ensure that logs are not readable after their scheduled deletion; this is off by default. See also**noshr ed**.

noshred
Do not use **shred** when deleting old log files. See also **shred**.

shredcycles *count*
Asks GNU **shred**(1) to overwrite log files **count** times before deletion. Without this option, **shred**'s default will be used.

allowhardlink
Rotate files with multiple hard links; this is off by default. The target file might get emptied, e.g. with **shred** or **copytruncate**. Use with caution, especially when the log files are rotated as root.

noallowhardlink
Do not rotate files with multiple hard links. See also **allowhardlink**.

Compression

compress
Old versions of log files are compressed with **gzip**(1) by default. See also**nocompr ess**.

nocompress
Old versions of log files are not compressed. See also **compress**.

compresscmd
Specifies which command to use to compress log files. The default is **gzip**(1). See also**compr ess**.

uncompresscmd
Specifies which command to use to uncompress log files. The default is **gunzip**(1).

compressext
Specifies which extension to use on compressed logfiles, if compression is enabled. The default follows that of the configured compression command.

compressoptions
Command line options may be passed to the compression program, if one is in use. The default, for **gzip**(1), is "−6" (biased towards high compression at the expense of speed). If you use a different compression command, you may need to change the **compressoptions** to match.

delaycompress
Postpone compression of the previous log file to the next rotation cycle. This only has effect when used in combination with **compress**. It can be used when some program cannot be told to close its logfile and thus might continue writing to the previous log file for some time.

nodelaycompress
Do not postpone compression of the previous log file to the next rotation cycle (this overrides the **delaycompress** option).

Filenames

extension *ext*
Log files with *ext* extension can keep it after the rotation. If compression is used, the compression extension (normally *.gz*) appears after *ext*. For example you have a logfile named mylog.foo and want to rotate it to mylog.1.foo.gz instead of mylog.foo.1.gz.

addextension *ext*
Log files are given the final extension *ext* after rotation. If the original file already ends with *ext*, the extension is not duplicated, but merely moved to the end, that is both **filename** and **filename***ext* would get rotated to filename.1*ext*. If compression is used, the compression extension (normally **.gz**) appears after *ext*.

start *count*
This is the number to use as the base for rotation. For example, if you specify 0, the logs will be created with a .0 extension as they are rotated from the original log files. If you specify 9, log files will be created with a .9, skipping 0–8. Files will still be rotated the number of times specified with the **rotate** directive.

dateext
Archive old versions of log files adding a date extension like YYYYMMDD instead of simply adding a number. The extension may be configured using the **dateformat** and **dateyesterday** options.

nodateext
Do not archive old versions of log files with date extension (this overrides the **dateext** option).

dateformat *format_string*
Specify the extension for **dateext** using the notation similar to **strftime**(3) function. Only %Y %m %d %H %M %S %V and %s specifiers are allowed. The default value is −%Y%m%d except hourly, which uses −%Y%m%d%H as default value. Note that also the character separating log name from the extension is part of the dateformat string. The system clock must be set past Sep 9th 2001 for %s to work correctly. Note that the datestamps generated by this format must be lexically sortable (that is first the year, then the month then the day. For example 2001/12/01 is ok, but 01/12/2001 is not, since 01/11/2002 would sort lower while it is later). This is because when using the **rotate** option, **logrotate** sorts all rotated filenames to find out which logfiles are older and should be removed.

dateyesterday
Use yesterday's instead of today's date to create the **dateext** extension, so that the rotated log file has a date in its name that is the same as the timestamps within it.

datehourago
Use hour ago instead of current date to create the **dateext** extension, so that the rotated log file has a hour in its name that is the same as the timestamps within it. Useful with rotate **hourly**.

Mail

mail *address*
When a log is rotated out of existence, it is mailed to *address*. If no mail should be generated by a particular log, the **nomail** directive may be used.

nomail Do not mail old log files to any address.

mailfirst
When using the **mail** command, mail the just-rotated file, instead of the about-to-expire file.

maillast

When using the **mail** command, mail the about-to-expire file, instead of the just-rotated file (this is the default).

Additional config files

include *file_or_directory*

Reads the file given as an argument as if it was included inline where the **include** directive appears. If a directory is given, most of the files in that directory are read in alphabetic order before processing of the including file continues. The only files which are ignored are files which are not regular files (such as directories and named pipes) and files whose names end with one of the taboo extensions or patterns, as specified by the **tabooext** or **taboopat** directives, respectively. The given path may start with ~/ to make it relative to the home directory of the executing user. For security reasons configuration files must not be group-writable nor world-writable.

Scripts

sharedscripts

Normally, **prerotate** and **postrotate** scripts are run for each log which is rotated and the absolute path to the log file is passed as first argument to the script. That means a single script may be run multiple times for log file entries which match multiple files (such as the */var/log/news/** example). If**shar edscripts** is specified, the scripts are only run once, no matter how many logs match the wildcarded pattern, and whole pattern is passed to them. However, if none of the logs in the pattern require rotating, the scripts will not be run at all. If the scripts exit with error (or any log fails to rotate), the remaining actions will not be executed for any logs. This option overrides the **nosharedscripts** option.

nosharedscripts

Run **prerotate** and **postrotate** scripts for every log file which is rotated (this is the default, and overrides the **sharedscripts** option). The absolute path to the log file is passed as first argument to the script. The absolute path to the final rotated log file is passed as the second argument to the **postrotate** script. If the scripts exit with error, the remaining actions will not be executed for the affected log only.

firstaction

script

endscript

The *script* is executed once before all log files that match the wildcarded pattern are rotated, before the prerotate script is run and only if at least one log will actually be rotated. These directives may only appear inside a log file definition. The whole pattern is passed to the script as its first argument. If the script exits with an error, no further processing is done. See also **lastaction** and the **SCRIPTS** section.

lastaction

script

endscript

The *script* is executed once after all log files that match the wildcarded pattern are rotated, after the postrotate script is run and only if at least one log is rotated. These directives may only appear inside a log file definition. The whole pattern is passed to the script as its first argument. If the script exits with an error, just an error message is shown (as this is the last action). See also **firstaction** and the **SCRIPTS** section.

prerotate

script
endscript

The *script* is executed before the log file is rotated and only if the log will actually be rotated. These directives may only appear inside a log file definition. Normally, the absolute path to the log file is passed as the first argument to the script. If **sharedscripts** is specified, the whole pattern is passed to the script. See also **postrotate** and the **SCRIPTS** section. See **sharedscripts** and **nosharedscripts** for error handling.

postrotate
script
endscript

The *script* is executed after the log file is rotated. These directives may only appear inside a log file definition. Normally, the absolute path to the log file is passed as the first argument to the script and the absolute path to the final rotated log file is passed as the second argument to the script. If**shar edscripts** is specified, the whole pattern is passed as the first argument to the script, and the second argument is omitted. See also **prerotate** and the **SCRIPTS** section. See **sharedscripts** and **nosharedscripts** for error handling.

preremove
script
endscript

The *script* is executed once just before removal of a log file. **logr otate** will pass the name of file which is soon to be removed as the first argument to the script. See also **firstaction** and the **SCRIPTS** section.

SCRIPTS

The lines between the starting keyword (e.g. **prerotate**) and **endscript** (both of which must appear on lines by themselves) are executed (using **/bin/sh**). The script inherits some traits from the**logr otate** process, including stderr, stdout, the current directory, the environment, and the umask. Scripts are run as the invoking user and group, irrespective of any **su** directive. If the**−−log** flag w as specified, file descriptor 3 is the log file. The current working directory is unspecified.

USER AND GROUP

User and group identifiers are resolved first by trying the textual representation and, in case it fails, afterwards by the numeric value.

FILES

/var/lib/logrotate/status	Default state file.
/etc/logrotate.conf	Configuration options.

SEE ALSO

chmod(2), **gunzip**(1), **gzip**(1), **mail**(1), **shred**(1), **strftime**(3), **strtoul**(3), <https://github.com/logrotate/logrotate>

AUTHORS

Erik Troan, Preston Brown, Jan Kaluza.

<https://github.com/logrotate/logrotate>