NAME

rsyslogd – reliable and extended syslogd

SYNOPSIS

rsyslogd [**–d**] [**–D**] [**–f** *config file*] [**–i** *pid file*] [**–n**] [**–N** *level*] [**–o** *fullconf*] [**–C**] [**–v**]

DESCRIPTION

Rsyslogd is a system utility providing support for message logging. Support of both internet and unix domain sockets enables this utility to support both local and remote logging.

Note that this version of rsyslog ships with extensive documentation in HTML format. This is provided in the ./doc subdirectory and probably in a separate package if you installed rsyslog via a packaging system. To use rsyslog's advanced features, you **need** to look at the HTML documentation, because the man pages only covers basic aspects of operation. **For details and configuration examples, see the rsyslog.conf (5) man page and the online documentation at https://www.rsyslog.com/doc/**

Rsyslogd(8) is derived from the sysklogd package which in turn is derived from the stock BSD sources.

Rsyslogd provides a kind of logging that many modern programs use. Every logged message contains at least a time and a hostname field, normally a program name field, too, but that depends on how trusty the logging program is. The rsyslog package supports free definition of output formats via templates. It also supports precise timestamps and writing directly to databases. If the database option is used, tools like phpLogCon can be used to view the log data.

While the **rsyslogd** sources have been heavily modified a couple of notes are in order. First of all there has been a systematic attempt to ensure that rsyslogd follows its default, standard BSD behavior. Of course, some configuration file changes are necessary in order to support the template system. However, rsyslogd should be able to use a standard syslog.conf and act like the original syslogd. However, an original syslogd will not work correctly with a rsyslog-enhanced configuration file. At best, it will generate funny looking file names. The second important concept to note is that this version of rsyslogd interacts transparently with the version of syslog found in the standard libraries. If a binary linked to the standard shared libraries fails to function correctly we would like an example of the anomalous behavior.

The main configuration file */etc/rsyslog.conf* or an alternative file, given with the **–f** option, is read at startup. Any lines that begin with the hash mark (''#'') and empty lines are ignored. If an error occurs during parsing the error element is ignored. It is tried to parse the rest of the line.

OPTIONS

–D Runs the Bison config parser in debug mode. This may help when hard to find syntax errors are reported. Please note that the output generated is deeply technical and orignally targeted towards developers.

–d Turns on debug mode. See the DEBUGGING section for more information.

–f *config file*
Specify an alternative configuration file instead of */etc/rsyslog.conf* , which is the default.

–i *pid file*
Specify an alternative pid file instead of the default one. This option must be used if multiple instances of rsyslogd should run on a single machine. To disable writing a pid file, use the reserved name "NONE" (all upper case!), so "-iNONE".

–n Avoid auto-backgrounding. This is needed especially if the **rsyslogd** is started and controlled by **init**(8).

–N level
Do a config check. Do NOT run in regular mode, just check configuration file correctness. This option is meant to verify a config file. To do so, run rsyslogd interactively in foreground,

specifying -f <config-file> and -N level. The level argument modifies behaviour. Currently, 0 is the same as not specifying the -N option at all (so this makes limited sense) and 1 actually activates the code. Later, higher levels will mean more verbosity (this is a forward-compatibility option).

−o fullconf
Generates a consolidated config file *fullconf* that contains all of rsyslog's configuration in a single file. Include files are exploded into that file in exactly the way rsyslog sees them. This option is useful for troubleshooting, especially if problems with the order of action processing is suspected. It may also be used to check for "unexepectedly" included config content.

−C This prevents rsyslogd from changing to the root directory. This is almost never a good idea in production use. This option was introduced in support of the internal testbed.

−v Print version and exit.

SIGNALS

Rsyslogd reacts to a set of signals. You may easily send a signal to **rsyslogd** using the following:

kill -SIGNAL $(cat /var/run/rsyslogd.pid)

Note that -SIGNAL must be replaced with the actual signal you are trying to send, e.g. with HUP. So it then becomes:

kill -HUP $(cat /var/run/rsyslogd.pid)

HUP This lets **rsyslogd** perform close all open files.

TERM , INT , QUIT
Rsyslogd will die.

USR1 Switch debugging on/off. This option can only be used if **rsyslogd** is started with the **−d** debug option.

CHLD Wait for childs if some were born, because of wall'ing messages.

SECURITY THREATS

There is the potential for the rsyslogd daemon to be used as a conduit for a denial of service attack. A rogue program(mer) could very easily flood the rsyslogd daemon with syslog messages resulting in the log files consuming all the remaining space on the filesystem. Activating logging over the inet domain sockets will of course expose a system to risks outside of programs or individuals on the local machine.

There are a number of methods of protecting a machine:

1. Implement kernel firewalling to limit which hosts or networks have access to the 514/UDP socket.
2. Logging can be directed to an isolated or non-root filesystem which, if filled, will not impair the machine.
3. The ext2 filesystem can be used which can be configured to limit a certain percentage of a filesystem to usage by root only. **NOTE** that this will require rsyslogd to be run as a non-root process. **ALSO NOTE** that this will prevent usage of remote logging on the default port since rsyslogd will be unable to bind to the 514/UDP socket.
4. Disabling inet domain sockets will limit risk to the local machine.

Message replay and spoofing

If remote logging is enabled, messages can easily be spoofed and replayed. As the messages are transmitted in clear-text, an attacker might use the information obtained from the packets for malicious things. Also, an attacker might replay recorded messages or spoof a sender's IP address, which could lead to a wrong perception of system activity. These can be prevented by using GSS-API authentication and encryption. Be sure to think about syslog network security before enabling it.

DEBUGGING

When debugging is turned on using the **–d** option, **rsyslogd** produces debugging information according to the **RSYSLOG_DEBUG** environment variable and the signals received. When run in foreground, the information is written to stdout. An additional output file can be specified using the **RSYSLOG_DEBUGLOG** environment variable.

FILES

/etc/rsyslog.conf
: Configuration file for **rsyslogd**. See **rsyslog.conf**(5) for exact information.

/dev/log
: The Unix domain socket to from where local syslog messages are read.

/var/run/rsyslogd.pid
: The file containing the process id of **rsyslogd**.

prefix/lib/rsyslog
: Default directory for **rsyslogd** modules. The *prefix* is specified during compilation (e.g. /usr/local).

ENVIRONMENT

RSYSLOG_DEBUG
: Controls runtime debug support. It contains an option string with the following options possible (all are case insensitive):

Debug
: Turns on debugging and prevents forking. This is processed earlier in the startup than command line options (i.e. -d) and as such enables earlier debugging output. Mutually exclusive with DebugOnDemand.

DebugOnDemand
: Enables debugging but turns off debug output. The output can be toggled by sending SIGUSR1. Mutually exclusive with Debug.

LogFuncFlow
: Print out the logical flow of functions (entering and exiting them)

FileTrace
: Specifies which files to trace LogFuncFlow. If not set (the default), a LogFuncFlow trace is provided for all files. Set to limit it to the files specified.FileTrace may be specified multiple times, one file each (e.g. export RSYSLOG_DEBUG="LogFuncFlow FileTrace=vm.c FileTrace=expr.c"

PrintFuncDB
: Print the content of the debug function database whenever debug information is printed (e.g. abort case)!

PrintAllDebugInfoOnExit
: Print all debug information immediately before rsyslogd exits (currently not implemented!)

PrintMutexAction
: Print mutex action as it happens. Useful for finding deadlocks and such.

NoLogTimeStamp
: Do not prefix log lines with a timestamp (default is to do that).

NoStdOut
: Do not emit debug messages to stdout. If RSYSLOG_DEBUGLOG is not set, this means no messages will be displayed at all.

Help
: Display a very short list of commands - hopefully a life saver if you can't access the documentation...

RSYSLOG_DEBUGLOG
: If set, writes (almost) all debug message to the specified log file in addition to stdout.

RSYSLOG_MODDIR
: Provides the default directory in which loadable modules reside.

BUGS

Please review the file BUGS for up-to-date information on known bugs and annoyances.

Further Information

Please visit **https://www.rsyslog.com/doc/** for additional information, tutorials and a support forum.

SEE ALSO

rsyslog.conf(5), **logger**(1), **syslog**(2), **syslog**(3), **services**(5), **savelog**(8)

COLLABORATORS

rsyslogd is derived from sysklogd sources, which in turn was taken from the BSD sources. Special thanks to Greg Wettstein (greg@wind.enjellic.com) and Martin Schulze (joey@linux.de) for the fine sysklogd package.

Rainer Gerhards
Adiscon GmbH
Grossrinderfeld, Germany
rgerhards@adiscon.com