NAME

sysklogd − Linux system logging utilities.

SYNOPSIS

syslogd [**−a** *socket*] [**−d**] [**−f** *config file*] [**−h**] [**−l** *hostlist*] [**−m** *interval*] [**−n**] [**−p** *socket*] [**−r**] [**−s** *domainlist*] [**−v**] [**−x**]

DESCRIPTION

Sysklogd provides two system utilities which provide support for system logging and kernel message trapping. Support of both internet and unix domain sockets enables this utility package to support both local and remote logging.

System logging is provided by a version of **syslogd**(8) derived from the stock BSD sources. Support for kernel logging is provided by the **klogd**(8) utility which allows kernel logging to be conducted in either a standalone fashion or as a client of syslogd.

Syslogd 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.

While the **syslogd** sources have been heavily modified a couple of notes are in order. First of all there has been a systematic attempt to insure that syslogd follows its default, standard BSD behavior. The second important concept to note is that this version of syslogd 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/syslog.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 whole line is ignored.

OPTIONS

−a *socket*

Using this argument you can specify additional sockets from that **syslogd** has to listen to. This is needed if you’re going to let some daemon run within a chroot() environment. You can use up to 19 additional sockets. If your environment needs even more, you have to increase the symbol **MAXFUNIX** within the syslogd.c source file. An example for a chroot() daemon is described by the people from OpenBSD at http://www.psionic.com/papers/dns.html.

−d Turns on debug mode. Using this the daemon will not proceed a **fork**(2) to set itself in the background, but opposite to that stay in the foreground and write much debug information on the current tty. See the DEBUGGING section for more information.

−f *config file*

Specify an alternative configuration file instead of */etc/syslog.conf* , which is the default.

−h By default syslogd will not forward messages it receives from remote hosts. Specifying this switch on the command line will cause the log daemon to forward any remote messages it receives to forwarding hosts which have been defined.

−l *hostlist*

Specify a hostname that should be logged only with its simple hostname and not the fqdn. Multiple hosts may be specified using the colon (‘‘:’’) separator.

−m *interval*

The **syslogd** logs a mark timestamp regularly. The default *interval* between two -- *MARK* -- lines is 20 minutes. This can be changed with this option. Setting the *interval* to zero turns it off entirely.

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

−p *socket*
You can specify an alternative unix domain socket instead of */dev/log*.

−r This option will enable the facility to receive message from the network using an internet domain socket with the syslog service (see **services**(5)). The default is to not receive any messages from the network.

This option is introduced in version 1.3 of the sysklogd package. Please note that the default behavior is the opposite of how older versions behave, so you might have to turn this on.

−s *domainlist*
Specify a domainname that should be stripped off before logging. Multiple domains may be specified using the colon (":") separator. Please be advised that no sub-domains may be specified but only entire domains. For example if **−s north.de** is specified and the host logging resolves to satu.infodrom.north.de no domain would be cut, you will have to specify two domains like: **−s north.de:infodrom.north.de**.

−v Print version and exit.

−x Disable name lookups when receiving remote messages. This avoids deadlocks when the nameserver is running on the same machine that runs the syslog daemon.

SIGNALS

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

kill -SIGNAL `cat /var/run/syslogd.pid`

SIGHUP
This lets **syslogd** perform a re-initialization. All open files are closed, the configuration file (default is */etc/syslog.conf*) will be reread and the **syslog**(3) facility is started again.

SIGTERM
The **syslogd** will die.

SIGINT, **SIGQUIT**
If debugging is enabled these are ignored, otherwise **syslogd** will die.

SIGUSR1
Switch debugging on/off. This option can only be used if **syslogd** is started with the **−d** debug option.

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

CONFIGURATION FILE SYNTAX DIFFERENCES

Syslogd uses a slightly different syntax for its configuration file than the original BSD sources. Originally all messages of a specific priority and above were forwarded to the log file.

For example the following line caused ALL output from daemons using the daemon facilities (debug is the lowest priority, so every higher will also match) to go into */usr/adm/daemons*:

```
# Sample syslog.conf
daemon.debug                    /usr/adm/daemons
```

Under the new scheme this behavior remains the same. The difference is the addition of four new specifiers, the asterisk (*****) wildcard, the equation sign (**=**), the exclamation mark (**!**), and the minus sign (**-**).

The ***** specifies that all messages for the specified facility are to be directed to the destination. Note that this behavior is degenerate with specifying a priority level of debug. Users have indicated that the asterisk notation is more intuitive.

The = wildcard is used to restrict logging to the specified priority class. This allows, for example, routing only debug messages to a particular logging source.

For example the following line in *syslog.conf* would direct debug messages from all sources to the */usr/adm/debug* file.

```
# Sample syslog.conf
*.=debug                        /usr/adm/debug
```

The **!** is used to exclude logging of the specified priorities. This affects all (!) possibilities of specifying priorities.

For example the following lines would log all messages of the facility mail except those with the priority info to the */usr/adm/mail* file. And all messages from news.info (including) to news.crit (excluding) would be logged to the */usr/adm/news* file.

```
# Sample syslog.conf
mail.*;mail.!=info              /usr/adm/mail
news.info;news.!crit        /usr/adm/news
```

You may use it intuitively as an exception specifier. The above mentioned interpretation is simply inverted. Doing that you may use

```
    mail.none
or
    mail.!*
or
    mail.!debug
```

to skip every message that comes with a mail facility. There is much room to play with it. :-)

The - may only be used to prefix a filename if you want to omit sync'ing the file after every write to it.

This may take some acclimatization for those individuals used to the pure BSD behavior but testers have indicated that this syntax is somewhat more flexible than the BSD behavior. Note that these changes should not affect standard **syslog.conf**(5) files. You must specifically modify the configuration files to obtain the enhanced behavior.

SUPPORT FOR REMOTE LOGGING

These modifications provide network support to the syslogd facility. Network support means that messages can be forwarded from one node running syslogd to another node running syslogd where they will be actually logged to a disk file.

To enable this you have to specify the **−r** option on the command line. The default behavior is that **syslogd** won't listen to the network.

The strategy is to have syslogd listen on a unix domain socket for locally generated log messages. This behavior will allow syslogd to inter-operate with the syslog found in the standard C library. At the same time syslogd listens on the standard syslog port for messages forwarded from other hosts. To have this work correctly the **services**(5) files (typically found in */etc*) must have the following entry:

```
syslog      514/udp
```

If this entry is missing **syslogd** neither can receive remote messages nor send them, because the UDP port cant be opened. Instead **syslogd** will die immediately, blowing out an error message.

To cause messages to be forwarded to another host replace the normal file line in the *syslog.conf* file with the name of the host to which the messages is to be sent prepended with an @.

For example, to forward **ALL** messages to a remote host use the following *syslog.conf* entry:

```
# Sample syslogd configuration file to
# messages to a remote host forward all.
*.*                     @hostname
```

To forward all **kernel** messages to a remote host the configuration file would be as follows:

```
# Sample configuration file to forward all kernel
# messages to a remote host.
kern.*          @hostname
```

If the remote hostname cannot be resolved at startup, because the name-server might not be accessible (it may be started after syslogd) you don't have to worry. **Syslogd** will retry to resolve the name ten times and then complain. Another possibility to avoid this is to place the hostname in */etc/hosts*.

With normal **syslogd**s you would get syslog-loops if you send out messages that were received from a remote host to the same host (or more complicated to a third host that sends it back to the first one, and so on). In my domain (Infodrom Oldenburg) we accidently got one and our disks filled up with the same single message. :-(

To avoid this in further times no messages that were received from a remote host are sent out to another (or the same) remote host anymore. If there are scenarios where this doesn't make sense, please drop me (Joey) a line.

If the remote host is located in the same domain as the host, **syslogd** is running on, only the simple hostname will be logged instead of the whole fqdn.

In a local network you may provide a central log server to have all the important information kept on one machine. If the network consists of different domains you don't have to complain about logging fully qualified names instead of simple hostnames. You may want to use the strip-domain feature **−s** of this server. You can tell the **syslogd** to strip off several domains other than the one the server is located in and only log simple hostnames.

Using the **−l** option there's also a possibility to define single hosts as local machines. This, too, results in logging only their simple hostnames and not the fqdns.

The UDP socket used to forward messages to remote hosts or to receive messages from them is only opened when it is needed. In releases prior to 1.3-23 it was opened every time but not opened for reading or forwarding respectively.

OUTPUT TO NAMED PIPES (FIFOs)

This version of syslogd has support for logging output to named pipes (fifos). A fifo or named pipe can be used as a destination for log messages by prepending a pipy symbol (''|'') to the name of the file. This is handy for debugging. Note that the fifo must be created with the mkfifo command before syslogd is started.

The following configuration file routes debug messages from the kernel to a fifo:

```
# Sample configuration to route kernel debugging
# messages ONLY to /usr/adm/debug which is a
# named pipe.
kern.=debug                     |/usr/adm/debug
```

INSTALLATION CONCERNS

There is probably one important consideration when installing this version of syslogd. This version of syslogd is dependent on proper formatting of messages by the syslog function. The functioning of the syslog

function in the shared libraries changed somewhere in the region of libc.so.4.[2-4].n. The specific change was to null-terminate the message before transmitting it to the */dev/log* socket. Proper functioning of this version of syslogd is dependent on null-termination of the message.

This problem will typically manifest itself if old statically linked binaries are being used on the system. Binaries using old versions of the syslog function will cause empty lines to be logged followed by the message with the first character in the message removed. Relinking these binaries to newer versions of the shared libraries will correct this problem.

Both the **syslogd**(8) and the **klogd**(8) can either be run from **init**(8) or started as part of the rc.* sequence. If it is started from init the option *−n* must be set, otherwise you'll get tons of syslog daemons started. This is because **init**(8) depends on the process ID.

SECURITY THREATS

There is the potential for the syslogd daemon to be used as a conduit for a denial of service attack. Thanks go to John Morrison (jmorriso@rflab.ee.ubc.ca) for alerting me to this potential. A rogue program(mer) could very easily flood the syslogd 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 syslogd to be run as a non-root process. **ALSO NOTE** that this will prevent usage of remote logging since syslogd will be unable to bind to the 514/UDP socket.
4. Disabling inet domain sockets will limit risk to the local machine.
5. Use step 4 and if the problem persists and is not secondary to a rogue program/daemon get a 3.5 ft (approx. 1 meter) length of sucker rod* and have a chat with the user in question.

   Sucker rod def. — 3/4, 7/8 or 1in. hardened steel rod, male threaded on each end. Primary use in the oil industry in Western North Dakota and other locations to pump 'suck' oil from oil wells. Secondary uses are for the construction of cattle feed lots and for dealing with the occasional recalcitrant or belligerent individual.

DEBUGGING

When debugging is turned on using **−d** option then **syslogd** will be very verbose by writing much of what it does on stdout. Whenever the configuration file is reread and re-parsed you'll see a tabular, corresponding to the internal data structure. This tabular consists of four fields:

number
: This field contains a serial number starting by zero. This number represents the position in the internal data structure (i.e. the array). If one number is left out then there might be an error in the corresponding line in */etc/syslog.conf*.

pattern
: This field is tricky and represents the internal structure exactly. Every column stands for a facility (refer to **syslog**(3)). As you can see, there are still some facilities left free for former use, only the left most are used. Every field in a column represents the priorities (refer to **syslog**(3)).

action
: This field describes the particular action that takes place whenever a message is received that matches the pattern. Refer to the **syslog.conf**(5) manpage for all possible actions.

arguments

This field shows additional arguments to the actions in the last field. For file-logging this is the filename for the logfile; for user-logging this is a list of users; for remote logging this is the hostname of the machine to log to; for console-logging this is the used console; for tty-logging this is the specified tty; wall has no additional arguments.

FILES

/etc/syslog.conf

Configuration file for **syslogd**. See **syslog.conf**(5) for exact information.

/dev/log

The Unix domain socket to from where local syslog messages are read.

/var/run/syslogd.pid

The file containing the process id of **syslogd**.

BUGS

If an error occurs in one line the whole rule is ignored.

Syslogd doesn't change the filemode of opened logfiles at any stage of process. If a file is created it is world readable. If you want to avoid this, you have to create it and change permissions on your own. This could be done in combination with rotating logfiles using the **savelog**(8) program that is shipped in the **smail** 3.x distribution. Remember that it might be a security hole if everybody is able to read auth.* messages as these might contain passwords.

SEE ALSO

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

COLLABORATORS

Syslogd is taken from BSD sources, Greg Wettstein (greg@wind.enjellic.com) performed the port to Linux, Martin Schulze (joey@linux.de) fixed some bugs and added several new features. **Klogd** was originally written by Steve Lord (lord@cray.com), Greg Wettstein made major improvements.

Dr. Greg Wettstein
Enjellic Systems Development
Oncology Research Division Computing Facility
Roger Maris Cancer Center
Fargo, ND
greg@wind.enjellic.com

Stephen Tweedie
Department of Computer Science
Edinburgh University, Scotland
sct@dcs.ed.ac.uk

Juha Virtanen
jiivee@hut.fi

Shane Alderton
shane@ion.apana.org.au

Martin Schulze
Infodrom Oldenburg
joey@linux.de