# Introduction

There are hundreds of Linux applications on the market, each with their own configuration files and help pages. This variety makes Linux vibrant, but it also makes Linux system administration daunting. Fortunately, in most cases, Linux applications use the syslog utility to export all their errors and status messages to files located in the /var/log directory.

This can be invaluable in correlating the timing and causes of related events on your system. It is also important to know that applications frequently don't display errors on the screen, but will usually log them somewhere. Knowing the precise message that accompanies an error can be vital in researching malfunctions in product manuals, online documentation, and Web searches.

syslog, and the logrotate utility that cleans up log files, are both relatively easy to configure but they frequently don't get their fair share of coverage in most texts. I've included syslog here as a dedicated chapter to both emphasize its importance to your Linux knowledge and prepare you with a valuable skill that will help you troubleshoot all the Linux various applications that will be presented throughout the book

# syslog

syslog is a utility for tracking and logging all manner of system messages from the merely informational to the extremely critical. Each system message sent to the syslog server has two descriptive labels associated with it that makes the message easier to handle.

* The first describes the function (facility) of the application that generated it. For example, applications such as mail and cron generate messages with easily identifiable facilities named mail and cron.
* The second describes the degree of severity of the message. There are eight in all and they are listed in Table 5-1:

You can configure syslog's /etc/rsyslog.conf configuration file to place messages of differing severities and facilities in different files. This procedure will be covered next.

## Table 5-1 Syslog Facilities

|  |  |  |
| --- | --- | --- |
| **Severity Level** | **Keyword** | **Description** |
| 0 | emergencies | System unusable |
| 1 | alerts | Immediate action required |
| 2 | critical | Critical condition |
| 3 | errors | Error conditions |
| 4 | warnings | Warning conditions |
| 5 | notifications | Normal but significant conditions |
| 6 | informational | Informational messages |
| 7 | debugging | Debugging messages |

## The /etc/rsyslog.conf File

The files to which syslog writes each type of message received is set in the /etc/rsyslog.conf configuration file. In older versions of Fedora this file was named /etc/syslog.conf.

This file consists of two columns. The first lists the facilities and severities of messages to expect and the second lists the files to which they should be logged. By default, RedHat/Fedora's /etc/rsyslog.conf file is configured to put most of the messages in the file/var/log/messages. Here is a sample:

\*.info;mail.none;authpriv.none;cron.none /var/log/messages

In this case, all messages of severity "info" and above are logged, but none from the mail, cron or authentication facilities/subsystems. You can make this logging even more sensitive by replacing the line above with one that captures all messages from debug severity and above in the /var/log/messages file. This example may be more suitable for troubleshooting.

\*.debug /var/log/messages

In this example, all debug severity messages; except auth, authpriv, news and mail; are logged to the /var/log/debug file in caching mode. Notice how you can spread the configuration syntax across several lines using the slash (\) symbol at the end of each line.

\*.=debug;\

auth,authpriv.none;\

news.none;mail.none -/var/log/debug

Here we see the /var/log/messages file configured in caching mode to receive only info, notice and warning messages except for the auth, authpriv, news and mail facilities.

\*.=info;\*.=notice;\*.=warn;\

auth,authpriv.none;\

cron,daemon.none;\

mail,news.none -/var/log/messages

You can even have certain types of messages sent to the screen of all logged in users. In this example messages of severity emergency and above triggers this type of notification. The file definition is simply replaced by an asterisk to make this occur.

\*.emerg \*

Certain applications will additionally log to their own application specific log files and directories independent of the syslog.conf file. Here are some common examples:

**Files:**

/var/log/maillog  : Mail

/var/log/httpd/access\_log  : Apache web server page access logs

**Directories:**

/var/log

/var/log/samba  : Samba messages

/var/log/mrtg  : MRTG messages

/var/log/httpd  : Apache webserver messages

**Note:** In some older versions of Linux the /etc/rsyslog.conf file was very sensitive to spaces and would recognize only tabs. The use of spaces in the file would cause unpredictable results. Check the formatting of your /etc/rsyslog.conf file to be safe.

## Activating Changes to the syslog Configuration File

Changes to /etc/rsyslog.conf will not take effect until you restart syslog.

Managing the syslog daemon is easy to do, but the procedure differs between Linux distributions. Here are some things to keep in mind.

* Firstly, different Linux distributions use different daemon management systems. Each system has its own set of commands to do similar operations. The most commonly used daemon management systems are SysV and Systemd.
* Secondly, the daemon name needs to be known. In this case the name of the daemon is **rsyslog**.

Armed with this information you can know how to:

* Start your daemons automatically on booting
* Stop, start and restart them later on during troubleshooting or when a configuration file change needs to be applied.

For more details on this, please take a look at the "Managing Daemons" section of Chapter 6 "[Installing Linux Software](http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch06_:_Installing_Linux_Software)"

## How to View New Log Entries as They Happen

If you want to get new log entries to scroll on the screen as they occur, then you can use this command:

[root@bigboy tmp]# tail -f /var/log/messages

Similar commands can be applied to all log files. This is probably one of the best troubleshooting tools available in Linux. Another good command to use apart from tail is grep. grep will help you search for all occurrences of a string in a log file; you can pipe it through the more command so that you only get one screen at a time. Here is an example:

[root@bigboy tmp]# grep string /var/log/messages | more

You can also just use the plain old more command to see one screen at a time of the entire log file without filtering with grep. Here is an example:

[root@bigboy tmp]# more /var/log/messages

## Logging syslog Messages to a Remote Linux Server

Logging your system messages to a remote server is a good security practice. With all servers logging to a central syslog server, it becomes easier to correlate events across your company. It also makes covering up mistakes or malicious activities harder because the purposeful deletion of log files on a server cannot simultaneously occur on your logging server, especially if you restrict the user access to the logging server.

### Configuring the Linux Syslog Server

By default syslog doesn't expect to receive messages from remote clients. Here's how to configure your Linux server to start listening for these messages.

As we saw previously, syslog checks its /etc/rsyslog.conf file to determine the expected names and locations of the log files it should create. It also checks the file /etc/sysconfig/syslog to determine the various modes in which it should operate. Syslog will not listen for remote messages unless the SYSLOGD\_OPTIONS variable in this file has a -r included in it as shown below.

# Options to syslogd

# -m 0 disables 'MARK' messages.

# -r enables logging from remote machines

# -x disables DNS lookups on messages received with -r

# See syslogd(8) for more details

SYSLOGD\_OPTIONS="-m 0 -r"

# Options to klogd

# -2 prints all kernel oops messages twice; once for klogd to decode, and

# once for processing with 'ksymoops'

# -x disables all klogd processing of oops messages entirely

# See klogd(8) for more details

KLOGD\_OPTIONS="-2"

**Note:** In Debian / Ubuntu systems you have to edit the syslog startup script /etc/init.d/sysklogd directly and make the SYSLOGD variable definition become "-r".

# Options for start/restart the daemons

# For remote UDP logging use SYSLOGD="-r"

#

#SYSLOGD="-u syslog"

SYSLOGD="-r"

You will have to restart syslog on the server for the changes to take effect. The server will now start to listen on UDP port 514, which you can verify using either one of the following netstat command variations.

[root@bigboy tmp]# netstat -a | grep syslog

udp 0 0 \*:syslog \*:\*

[root@bigboy tmp]# netstat -an | grep 514

udp 0 0 0.0.0.0:514 0.0.0.0:\*

[root@bigboy tmp]#

### Configuring the Linux Client

The syslog server is now expecting to receive syslog messages. You have to configure your remote Linux client to send messages to it. This is done by editing the /etc/hosts file on the Linux client named smallfry. Here are the steps:

1) Determine the IP address and fully qualified hostname of your remote logging host.

2) Add an entry in the /etc/hosts file in the format:

IP-address fully-qualified-domain-name hostname "loghost"

Example:

192.168.1.100 bigboy.my-site.com bigboy loghost

Now your /etc/hosts file has a nickname of "loghost" for server bigboy.

3) The next thing you need to do is edit your /etc/rsyslog.conf file to make the syslog messages get sent to your new loghost nickname.

\*.debug @loghost

\*.debug /var/log/messages

You have now configured all debug messages and higher to be logged to both server bigboy ("loghost") and the local file /var/log/messages. Remember to restart syslog to get the remote logging started.

You can now test to make sure that the syslog server is receiving the messages with a simple test such as restarting the lpd printer daemon and making sure the remote server sees the messages.

**Linux Client**

[root@smallfry tmp]# systemctl restart lpd.service

**Linux Server**

[root@bigboy tmp]# tail /var/log/messages

...

...

Apr 11 22:09:35 smallfry lpd: lpd shutdown succeeded

Apr 11 22:09:39 smallfry lpd: lpd startup succeeded

...

...

[root@bigboy tmp]#

## Syslog Configuration and Cisco Network Devices

syslog reserves facilities "local0" through "local7" for log messages received from remote servers and network devices. Routers, switches, firewalls and load balancers each logging with a different facility can each have their own log files for easy troubleshooting. Appendix 4 has examples of how to configure syslog to do this with Cisco devices using separate log files for the routers, switches, PIX firewalls, CSS load balancers and LocalDirectors.

# Logrotate

The Linux utility logrotate renames and reuses system error log files on a periodic basis so that they don't occupy excessive disk space.

## The /etc/logrotate.conf File

This is logrotate's general configuration file in which you can specify the frequency with which the files are reused.

* You can specify either a weekly or daily rotation parameter. In the case below the weekly option is commented out with a #, allowing for daily updates.
* The rotate parameter specifies the number of copies of log files logrotate will maintain. In the case below the 4 copy option is commented out with a #, while allowing 7 copies.
* The create parameter creates a new log file after each rotation

Therefore, our sample configuration file will create daily archives of all the logfiles and store them for seven days. The files will have the following names with, logfile being current active version:

logfile

logfile.0

logfile.1

logfile.2

logfile.3

logfile.4

logfile.5

logfile.6

## Sample Contents of /etc/logrotate.conf

# rotate log files weekly

#weekly

# rotate log files daily

daily

# keep 4 weeks worth of backlogs

#rotate 4

# keep 7 days worth of backlogs

rotate 7

# create new (empty) log files after rotating old ones

create

## The /etc/logrotate.d Directory

Most Linux applications that use syslog will put an additional configuration file in this directory to specify the names of the log files to be rotated. It is a good practice to verify that all new applications that you want to use the syslog log have configuration files in this directory. Here are some sample files that define the specific files to be rotated for each application.

Here is an example of a custom file located in this directory that rotates files with the .tgz extension which are located in the/data/backups directory. The parameters in this file will override the global defaults in the /etc/logrotate.conf file. In this case, the rotated files won't be compressed, they'll be held for 30 days only if they are not empty, and they will be given file permissions of 600 for userroot.

/data/backups/\*.tgz {

daily

rotate 30

nocompress

missingok

notifempty

create 0600 root root

}

**Note:** In Debian / Ubuntu systems the /etc/cron.daily/sysklogd script reads the /etc/rsyslog.conf file and rotates any log files it finds configured there. This eliminates the need to create log rotation configuration files for the common system log files in the/etc/logrotate.d directory. As the script resides in the /etc/cron.daily directory it automatically runs every 24 hours. In Fedora / Redhat systems this script is replaced by the /etc/cron.daily/logrotate daily script which does not use the contents of the syslog configuration file, relying mostly on the contents of the /etc/logrotate.d directory.

## Activating logrotate

The above logrotate settings in the previous section will not take effect until you issue the following command:

[root@bigboy tmp]# logrotate -f

If you want logrotate to reload only a specific configuration file, and not all of them, then issue the logrotate command with just that filename as the argument like this:

[root@bigboy tmp]# logrotate -f /etc/logrotate.d/syslog

## Compressing Your Log Files

On busy Web sites the size of your log files can become quite large. Compression can be activated by editing the logrotate.conf file and adding the compress option.

#

# File: /etc/logrotate.conf

#

# Activate log compression

compress

The log files will then start to become archived with the gzip utility, each file having a .gz extension.

[root@bigboy tmp]# ls /var/log/messages\*

/var/log/messages /var/log/messages.1.gz /var/log/messages.2.gz

/var/log/messages.3.gz /var/log/messages.4.gz /var/log/messages.5.gz

/var/log/messages.6.gz /var/log/messages.7.gz

[root@bigboy tmp]#

Viewing the contents of the files still remains easy because the zcat command can quickly output their contents to the screen. Use the command with the compressed file's name as the argument as seen below.

[root@bigboy tmp]# zcat /var/log/messages.1.gz

...

...

Nov 15 04:08:02 bigboy httpd: httpd shutdown succeeded

Nov 15 04:08:04 bigboy httpd: httpd startup succeeded

Nov 15 04:08:05 bigboy sendmail[6003]: iACFMLHZ023165: to=<tvaughan@clematis4spiders.info>,

delay=2+20:45:44, xdelay=00:00:02, mailer=esmtp, pri=6388168,

relay=www.clematis4spiders.info. [222.134.66.34], dsn=4.0.0,

stat=Deferred: Connection refused by www.clematis4spiders.info.

[root@bigboy tmp]#

# syslog-ng

The more recent syslog-ng application combines the features of logrotate and syslog to create a much more customizable and feature rich product. This can be easily seen in the discussion of its configuration file that follows.

## The /etc/syslog-ng/syslog-ng.conf file

The main configuration file for syslog-ng is the /etc/syslog-ng/sylog-ng.conf file but only rudimentary help on its keywords can be found using the Linux man pages.

[root@bigboy tmp]# man syslog-ng.conf

Don’t worry, we’ll soon explore how much more flexible syslog-ng can be when compared to regular syslog.

### Simple Server Side Configuration for Remote Clients

Figure 5-1 has a sample syslog-ng.conf file and outlines some key features. The options section that covers global characteristics is fully commented, but it is the source, destination and log sections that define the true strength of the customizability of syslog-ng.

#### Figure 5-1 A Sample syslog-ng.conf File

options {

# Number of syslog lines stored in memory before being written to files

sync (0);

# Syslog-ng uses queues

log\_fifo\_size (1000);

# Create log directories as needed

create\_dirs (yes);

# Make the group "logs" own the log files and directories

group (logs);

dir\_group (logs);

# Set the file and directory permissions

perm (0640);

dir\_perm (0750);

# Check client hostnames for valid DNS characters

check\_hostname (yes);

# Specify whether to trust hostname in the log message.

# If "yes", then it is left unchanged, if "no" the server replaces

# it with client's DNS lookup value.

keep\_hostname (yes);

# Use DNS fully qualified domain names (FQDN)

# for the names of log file folders

use\_fqdn (yes);

use\_dns (yes);

# Cache DNS entries for up to 1000 hosts for 12 hours

dns\_cache (yes);

dns\_cache\_size (1000);

dns\_cache\_expire (43200);

};

# Define all the sources of localhost generated syslog

# messages and label it "d\_localhost"

source s\_localhost {

pipe ("/proc/kmsg" log\_prefix("kernel: "));

unix-stream ("/dev/log");

internal();

};

# Define all the sources of network generated syslog

# messages and label it "d\_network"

source s\_network {

tcp(max-connections(5000));

udp();

};

# Define the destination "d\_localhost" log directory

destination d\_localhost {

file ("/var/log/syslog-ng/$YEAR.$MONTH.$DAY/localhost/$FACILITY.log");

};

# Define the destination "d\_network" log directory

destination d\_network {

file ("/var/log/syslog-ng/$YEAR.$MONTH.$DAY/$HOST/$FACILITY.log");

};

# Any logs that match the "s\_localhost" source should be logged

# in the "d\_localhost" directory

log { source(s\_localhost);

destination(d\_localhost);

};

# Any logs that match the "s\_network" source should be logged

# in the "d\_network" directory

log { source(s\_network);

destination(d\_network);

};

In our example, the first set of sources is labeled s\_localhost. It includes all system messages sent to the Linux /dev/log device, which is one of syslog's data sources, all messages that syslog-ng views as being of an internal nature and additionally inserts the prefix "kernel" to all messages it intercepts on their way to the /proc/kmsg kernel message file.

Like a regular syslog server which listens for client messages on UDP port 514, syslog-ng also listens on TCP port 514. The second set of sources is labeled s\_network and includes all syslog messages obtained from UDP sources and limits TCP syslog connections to 5000. Limiting the number of connections to help regulate system load is a good practice in the event that some syslog client begins to inundate your server with messages.

Our example also has two destinations for syslog messages, one named d\_localhost, the other, d\_network. These examples show the flexibility of syslog-ng in using variables. The $YEAR, $MONTH and $DAY variables map to the current year, month and day in YYYY, MM and DD format respectively. Therefore the example:

/var/log/syslog-ng/$YEAR.$MONTH.$DAY/$HOST/$FACILITY.log

refers to a directory called /var/log/syslog-ng/2005.07.09 when messages arrive on July 9, 2005. The $HOST variable refers to the hostname of the syslog client and will map to the client's IP address if DNS services are deactivated in the options section of the syslog-ng.conf file. Similarly the $FACILITY variable refers to the facility of the syslog messages that arrive from that host.

### Using syslog-ng in Large Data Centers

Figure 5-2 has a sample syslog-ng.conf file snippet that defines some additional features that may be of interest in a data center environment.

#### Figure 5-2 More Specialized syslog-ng.conf Configuration

options {

# Number of syslog lines stored in memory before being written to files

sync (100);

};

# Define all the sources of network generated syslog

# messages and label it "s\_network\_1"

source s\_network\_1 {

udp(ip(192.168.1.201) port(514));

};

# Define all the sources of network generated syslog

# messages and label it "s\_network\_2"

source s\_network\_2 {

udp(ip(192.168.1.202) port(514));

};

# Define the destination "d\_network\_1" log directory

destination d\_network\_1 {

file ("/var/log/syslog-ng/servers/$YEAR.$MONTH.$DAY/$HOST/$FACILITY.log");

};

# Define the destination "d\_network\_2" log directory

destination d\_network\_2 {

file ("/var/log/syslog-ng/network/$YEAR.$MONTH.$DAY/$HOST/$FACILITY.log");

};

# Define the destination "d\_network\_2B" log directory

destination d\_network\_2B {

file ("/var/log/syslog-ng/network/all/network.log");

};

# Any logs that match the "s\_network\_1" source should be logged

# in the "d\_network\_1" directory

log { source(s\_network\_1);

destination(d\_network\_1);

};

# Any logs that match the "s\_network\_2" source should be logged

# in the "d\_network\_2" directory

log { source(s\_network\_2);

destination(d\_network\_2);

};

# Any logs that match the "s\_network\_2" source should be logged

# in the "d\_network\_2B" directory also

log { source(s\_network\_2);

destination(d\_network\_2B);

};

In this case we have configured syslog to:

1. Listen on IP address 192.168.1.201 as defined in the source s\_network\_1. Messages arriving at this address will be logged to a subdirectory of /var/log/syslog-ng/servers/ arranged by date as specified by destination d\_network\_1. As you can guess, this address and directory be used by all servers in the data center.
2. Listen on IP address 192.168.1.202 as defined in the source s\_network\_2. Messages arriving at this address will be logged to a subdirectory of /var/log/syslog-ng/network/ arranged by date as specified by d\_network\_2. This will be the IP address and directory to which network devices would log.
3. Listen on IP address 192.168.1.202 as defined in the source s\_network\_2. Messages arriving at this address will also be logged to file /var/log/syslog-ng/all/debug.log as part of destination d\_network\_2B.This will be a single file to which all network devices would log. Server failures are usually isolated to single servers whereas network failures are more likely to be cascading involving many devices. The advantage of searching a single file is that it makes it easier to determine the exact sequence of events.
4. As there could be many devices logging to the syslog-ng server, the sync option is set to write data to disk only after receiving 100 syslog messages. Constant receipt of syslog messages can have a significant impact on your system’s disk performance. This option allows you to queue the messages in memory for less frequent disk updates.

Now that you have an understanding of how to configure syslog-ng it’s time to see how you install it.

## Installing and Starting syslog-ng

You can install syslog-ng using standard Linux procedures.The syslog-ng and rsyslog packages cannot be installed at the same time. You have to uninstall one in order for the other to work. Here’s how you can install syslog-ng using RPM package files.

1. Uninstall rsyslog using the rpm command. There are some other RPMs that rely on rsyslog so you will have to do this while ignoring any dependencies with the –nodeps flag.

[root@bigboy tmp]# rpm -e --nodeps rsyslog

2. Install syslog-ng using yum.

[root@bigboy tmp]# yum -y install syslog-ng

3. Start the new syslog-ng daemon immediately and make sure it will start on the next reboot.

Systems using sysvinit:

[root@bigboy tmp]# chkconfig syslog-ng on

[root@bigboy tmp]# service syslog-ng start

Starting syslog-ng: [ OK ]

[root@bigboy tmp]#

Systems using systemd:

[root@bigboy tmp]# systemctl enable syslog-ng.service

[root@bigboy tmp]# systemctl start syslog-ng.service

Starting syslog-ng: [ OK ]

[root@bigboy tmp]#

Your new syslog-ng package is now up and running and ready to go!

## Configuring syslog-ng Clients

Clients logging to the syslog-ng server don't need to have syslog-ng installed on them, a regular syslog client configuration will suffice.

If you are running syslog-ng on clients, then you’ll need to modify your configuration file. Let’s look at Example 5-1 – Syslog-ng Sample Client Configuration.

### Example 5-1 - Syslog-ng Sample Client Configuration

source s\_sys {

file ("/proc/kmsg" log\_prefix("kernel: "));

unix-stream ("/dev/log");

internal();

};

destination loghost {

udp("loghost.linuxhomenetworking.com");

};

filter notdebug {

level(info...emerg);

};

log {

source(local);

filter(notdebug);

destination(loghost);

};

The s\_sys source comes default in many syslong-ng.conf files, we have just added some additional parameters to make it work. Here the destination syslog logging server is defined as loghost.linuxhomenetworking.com. We have also added a filter to the log section to make sure only the most urgent messages, info level and above (not debug), get logged to the remote server. After restarting syslong-ng on your client, your syslog server will start receiving messages.

# Simple syslog Security

One of the shortcomings of a syslog server is that it doesn't filter out messages from undesirable sources. It is therefore wise to implement the use of TCP wrappers or a firewall to limit the acceptable sources of messages when your server isn't located on a secure network. This will help to limit the effectiveness of syslog based denial of service attacks aimed at filling up your server's hard disk or taxing other system resources that could eventually cause the server to crash.

Remember that regular syslog servers listen on UDP port 514 and syslog-ng servers rely on port 514 for both UDP and TCP. Please refer to Chapter 14, "[Linux Firewalls Using iptables](http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch14_:_Linux_Firewalls_Using_iptables)", on Linux firewalls for details on how to configure the Linux iptables firewall application and Appendix I, "[Miscellaneous Linux Topics](http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ap01_:_Miscellaneous_Linux_Topics)", for further information on configuring TCP wrappers.

# Conclusion

In the next chapter we cover the installation of Linux applications, and the use of syslog will become increasingly important especially in the troubleshooting of Linux-based firewalls which can be configured to ignore and then log all undesirable packets; the Apache Web server which logs all application programming errors generated by some of the popular scripting languages such as PERL and PHP; and finally, Linux mail whose configuration files are probably the most frequently edited system documents of all and which correspondingly suffer from the most mistakes.

This syslog chapter should make you more confident to learn more about these applications via experimentation because you'll at least know where to look at the first sign of trouble.