Insider Tips: Logrotate

TURNING THE LOGS

Every multi-purpose Linux system produces an enormous amount of log data. To prevent your hard disk from overflowing, a rotating helper application archives logs and gets rid of obsolete data. **BY MARC ANDRÉ SELIG**

You have to be a very special kind of person to actually enjoy evaluating logfiles. But you have to admit that the logfiles in */var/log* give administrators exactly the kind of information they need to discover the origins of mysterious system errors. Log files also provide information on whether services on the system are working.

If you have any trouble installing a daemon or running a script, the logs can tell you what went wrong. Paranoid admins – and that should be all of us – check their systems for unauthorized access and block any traces they find.

You can also use log information to control a computer's behavior. For example, an SMTP server could view an IP address as authorized to send mail if the logfile has a successful login entry for the mailbox. Or a firewall could deploy a packet filter to automatically block an IP address following an obvious attack.

Even if you don't read your own logs, you can still trip over them. The logfiles consume more and more space on your hard disk – or at least on the */var* partition. The gradual consumption of disk space will eventually force your computer to its knees.

Sensible logfile handling – evaluation, archiving, deleting – is a traditional administrative task. But now that Linux has started to conquer the desktop, it is unreasonable to expect users to handle log file administration for themselves. Some fully automated tools were introduced years ago to assist with the task of managing logfiles.

Don't Delete, Rotate!

A Linux system does not simply delete logfiles, but rotates them. It first renames a file such as *XXX.log* to *XXX.log.o* (or something similar). This step is important, as the automatic cleaning program has no way of knowing if a process is currently accessing the logfile. If a process has opened the logfile, it possesses a filehandle for exactly this file and will write to the file regardless of whether another process has renamed, or deleted, the file in the meantime.

After renaming the file, the protocol cleaner needs to recreate new (blank) logfiles and tell the processes concerned that the logfiles have changed. The writing processes then need to close any logfiles they are using and reopen those files, thus receiving the current (blank) version, which they can then populate.

Things get complicated at this juncture, as the central logfile management tool needs to find out which processes write to which logfiles. The fact that many daemons use *syslog* makes this easier to handle, but there are still popular packages – such as Apache – that insist on using their own logfiles for performance or other reasons.

Modular Configuration

To establish this mechanism on a freely-configurable and extensible Linux sys-

Listing 1: logrotate.conf Sample File

```
01 weekly
02 rotate 4
03 create
04 compress
05 delaycompress
06
07 /var/log/wtmp {
08   missingok
09   monthly
10   create 0664 root utmp
11   rotate 1
12 }
13
14 include /etc/logrotate.d
```

Listing 2: /etc/logrotate.d/apache2

```
01 /var/log/apache2/*.log {
02   missingok
03   rotate 52
04   notifempty
05   create 640 root adm
06   sharedscripts
07   postrotate
08     if [ -f /var/run/apache2.
   pid ]; then
09       /etc/init.d/apache2
   restart >/dev/null
10     fi
11   endscript
12 }
```

tem, clever developers came up with the Logrotate package. Logrotate uses a modular configuration like many other centralized packages. The core variables are set by a central configuration file. The file makes a note of how often Logrotate should rotate the logfiles, how long it should keep old logs, whether to compress the files (starting in the second generation, as the logfile that has just been rotated might still be in use!), and so on. The central configuration file is titled */etc/logrotate.conf*.

Additionally, each software package can add an entry to the Logrotate configuration file during the install. Of course, it should not write to the central file – this could lead to nasty inconsistencies; instead, Logrotate uses a configuration directory titled */etc/logrotate.d*.

Packages can store their own miniature Logrotate configuration files under the package name. These files say what to do with rotated logfiles. The configuration entry for a proxy server might be */etc/logrotate.d/squid*, for example.

An Example

Listing 1 gives an example of a minimalist */etc/logrotate.conf*. All it does is set a few defaults: based on this configuration, Logrotate would rotate the logfiles weekly and keep a total of four previous versions. After each rotation, the tool would create a new logfile and compress the pervious week's version.

Listing 1 specifies the logfile for Logrotate to manage, */var/log/wtmp*. Logrotate only creates this file once a month, handing the permissions for the file to *root* and the *utmp* group. The *rotate 1* keyword in line 11 tells Logrotate to keep just one previous version.

logrotate.conf does not specify how to handle the other logfiles; instead this is handled by configuration snippets from */etc/logrotate.d*, which the include directive in line 14 enables. Include ignores any backup files created by editors, files from version control systems, fragments left over by package managers (for example, **.rpmsave* or **.dpkg-old*), and files with the extension *.disabled*.

Special Treatment for Apache

The */etc/logrotate.d* file includes the file *apache2*, with an example for Apache (Listing 2) containing special options for rotating Apache logs. This listing starts with the path to the logfiles – Apache is set up to store logfiles below */var/log/apache2*.

The default settings from Listing 1 apply to all the files in */var/log/apache2*; first: Logrotate rotates the files on a weekly basis, packs the files after another week, and creates a new file for each file it rotates. Line 5 in Listing 2 assigns specific access privileges to the new and empty logs to prevent normal users from viewing them.

One assumption the example makes is that web server logs are very important for system operations and billing. We need the system to keep the logs for a whole year, creating 52 generations at a rate of one generation per week. If Apache fails to launch for some unknown reason, which would lead to a missing log, the program will not rotate, as stipulated by the *notifempty* keyword.

Listing 2 also contains instructions on how to let Apache know when files have been rotated. The message is sent after the event – as indicated by the *postrotate* keyword (the keyword for a message prior to rotation is *prerotate*.) The keyword is followed by a few lines of shell script, which close with the *endscript* keyword. Lines 8 through 10 say: if Apache is running – that is, if the PID file exists – the matching Init script will handle the relaunch.

The *sharedscript* keyword in line 6 tells Logrotate to call this info script once only, even when rotating multiple logfiles. It launches once rotation has been completed for all logfiles.

Sophisticated

The settings in Listings 1 and 2 give you examples of how to configure the standard daemons on common Linux distributions. But *logrotate* has more tricks up its sleeve – Listing 3 configures a packet filter, for example. This setup assumes that the packet filter logs will be stored in *ipfilter-bulk.log*. The file could be enormous on a typical server, and its sheer size would prevent manual evaluation. A script (which is not printed here for reasons of space) thus parses the file for relevant, critical information and writes this information to the *ipfilter-high.log* file.

Listing 3 needs to treat these two files quite differently. It rotates *ipfilter-bulk.log* whenever the file reaches a size of 20 Mbytes, but normally no more than once a day. Logrotate compresses the old logfile using *bzip2*. As this is a syslog-based logfile, the next step is to notify the daemon of the logfile changes; the *postrotate* directive in the appropriate Init script is used to do this, as in the previous Apache example.

ipfilter-high.log needs a different kind of treatment. As it might contain time-critical information, *logrotate* rotates this file once a day – and additionally mails the administrator. If the administrator happens to be slacking (although it is not typical of administrators to do so), this gives the admin fair warning and enough time to take care of the logs. *logrotate* normally dispatches the oldest logfile, but the *mailfirst* directive tells it to send the current file just after rotation.

Critical data is not simply discarded after a few weeks but kept for forensic investigation – in our case, 730 days, or two years if you prefer. To prevent the old logfiles from cluttering up the */var/log* directory, *logrotate* moves them into a separate directory below */var/log/ipfilter-old.d*. ■

Listing 3: Two Different Logs on a Packet Filter

```
01 /var/log/ipfilter-bulk.log {
02   size 20M
03   rotate 10
04   compress
05   compresscmd bzip2
06   compressext bz2
07   postrotate
08     /etc/init.d/sysklogd
   reload >/dev/null
09   endscript
10 }
11
12 /var/log/ipfilter-high.log {
13   daily
14   rotate 730
15   olddir /var/log/
   ipfilter-old.d
16   nocompress
17   mail <I>ich<I>@<I>meine.
   domain.de<I>
18   mailfirst
19 }
```