# Table of Contents

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

# Chapter 6. Oak

## 6.1. Overview of Oak

Machines that provide network services, and many network devices such as switches and routers, keep a log of messages about operational conditions. This message log is a useful tool for analyzing a problem after it has occurred. When an interface on a router stops functioning, you can login to the router and examine the error log for information about the problem.

There are drawbacks, however, to a system that requires you to login to a device in order to access its system logs. An outage of even a critical device may go unnoticed if it is off-hours and short in duration. But as an administrator, you need to know that it happened and why it happened so that you can prevent it from recurring. It is also difficult to monitor the log files via a login session if you have a large number of devices in your environment. With dozens of servers or network devices, you will not be able to check the log on each device every day.

For this reason, most devices that keep a local message log are also capable of sending notifications to a remote host using the **syslog** protocol. The syslog protocol was originally used to transport system log messages between Unix workstations but has since grown to become nearly ubiquitous for reporting error messages between devices. Using the syslog protocol, you can configure all of your servers and network devices to send error messages to a single machine, and from that machine, you can monitor the resulting error log. Depending on the device in question, you may even be able to select which kind of messages are forwarded and how severe a message has to be before it is forwarded.

Storing all the log messages on a central server is only half the battle. We would additionally like a system that can summarize and process the messages for us. If a critical problem is detected, an operator should be notified immediately. Less serious problems can be reported in an hourly or daily message. Oak is a program developed at MIT that will process messages and allow you to respond appropriately.

## 6.2. What Oak Can Help You Do

Oak examines a message log in syslog format and allows you to:

- Ignore unimportant messages
- Condense redundant information
- Produce reports of important messages
- Notify operators immediately of critical messages

Note that the term "syslog format" is a bit misleading. There is no standard format for the printed syslog messages themselves, only for the mechanism that transports them between machines. However, printed syslog messages are typically in one of a small number of formats, and Oak takes measures to correctly interpret the format of the message.

The Oak configuration file will specify which messages are important to you and how you wish to be notified in the event they should be received. For example, at MIT, we have Oak configured to send a daily report in email, an hourly report in an instant message[1] to the operational group, and an immediate instant message to the operational group if a critical problem is detected. One of the hourly messages might look like this:

[1] Instant message here simply refers to a text message that is sent directly to the users; it is not delayed as email can be. At MIT, we use the Zephyr protocol and applications for this purpose.

```
Hourly message log

SERVER1.EXAMPLE.COM:
   2: login: ROOT LOGIN console
   1: syslogd: going down on signal 15
   1: saslauthd[___]: Caught signal 15. Cleaning up $
   1: genunix: syncing file systems...
   1: genunix:  done
   1: genunix: ^MSunOS Release 5.9 Version Generic 64-bit
   1: genunix: Copyright 1983-2002 Sun Microsystems,$
   1: Use is subject to license terms.
   ** Too many messages found for host, truncating **

ROUTER.EXAMPLE.COM:
   6: ___:___ %LINEPROTO-5-UPDOWN: \
      Line protocol on Interface Ethernet9/4, change$
   5: ___:___ %LINEPROTO-5-UPDOWN: \
      Line protocol on Interface Ethernet9/4, change$
   2: ___:___ %LINEPROTO-5-UPDOWN: \
      Line protocol on Interface Ethernet9/2, change$
   1: ___:___ %LINEPROTO-5-UPDOWN: \
      Line protocol on Interface Ethernet9/2, change$

SERVER2.EXAMPLE.COM:
   9: named[___]: poll: Invalid argument

SERVER3.EXMPLE.COM:
   11: sshd[___]: ROOT LOGIN as 'root' from CLIENT.EXAMPLE.COM

** Message longer than 25 lines, message has been truncated **
```

The number to the left of each message indicates how many copies of the message were received. Note that in several places, Oak has replaced text with a series of underscores. These are examples of Oak's finding and removing information that may be redundant or unnecessary for reporting. If Oak did not remove the pieces of information to the left of the `LINEPROTO-5-UPDOWN` messages, each one would be reported on a line of its own. This would increase the size of your report and make it more difficult to understand.

Also notice that Oak truncates the message, both when there are too many messages for a particular host and when the message itself is too long. These are parameters set in the configuration file, and they can be set differently for different reports. The previous example was an instant message and as such was restricted to a relatively small amount of space. The daily email, however, is allowed to use many more lines.

A time-critical message might look like this:

```
**** CRITICAL MESSAGE LOG ****

SERVER4.EXAMPLE.COM:
   ufs: NOTICE: alloc: /var: file system full
```

Here we see a server with a full filesystem, which should be reported to an administrator right away.

## 6.3. Installing Oak

Oak is available from http://web.mit.edu/ktools/. Download the latest version and unpackage it:

```
Solaris% gunzip -c oak-1.3.5.tar.gz | tar xvf -
Solaris% cd oak-1.3.5
```

Then configure and build it:

```
Solaris% ./configure
Solaris% make
```

Because Oak does not make use of any particularly nonstandard libraries, it should build without any problem. When it is complete, you will have a binary called `oak`, which you can install on your system from a root account:

```
  Solaris# make install
```

This will place a copy of the `oak` binary in `/usr/local/bin/`.

## 6.4. Using Oak

Before configuring Oak to notify you of system events, you must first configure your servers and network devices to forward their syslogs to a central server, as described in the next section.

### 6.4.1. Configuring Syslog on Unix Workstations

Every syslog message has four basic parts:

- The system **facility**
- A **severity level**
- A time stamp
- The message content

The system facility refers to different services on a system so that messages can be sorted by the type of service. The valid service types for Solaris are listed in Figure 6.1. These are self-explanatory. The `mail` facility is used for messages about the mail system; the `kern` facility is used for messages from the kernel. There are eight facilities reserved for local admins to use as they please, named `local0` through `local7`. There is also a facility called `mark` which is used internally by syslog.

**Figure 6.1. Facility Types on Solaris.**

| **Facility** |
|---|
| user |
| kern |
| mail |
| daemon |
| auth |
| lpr |

| Facility |
|---|
| |
| news |
| uucp |
| cron |
| local0-7 |

Some of these facilities, like the uucp facility, are a bit out of date, and you will notice that other modern services are not included. There is no `web` facility, for example. Some services, including the Apache Web server, choose to implement their own logging outside the syslog system.

Along with the facility, each syslog message has a level of severity. Valid severity levels are listed in Figure 6.2. The `emerg` severity level is the most severe and `debug` is the least severe. The higher the severity level, the more immediate attention is required.

**Figure 6.2. Severity Level Values.**

| Facility |
|---|
| emerg |
| alert |
| crit |
| err |
| warning |
| notice |
| info |
| debug |

The purpose of designating each message with a facility and severity level is to allow operators to sort syslog messages by priority. As far as we're concerned, we need only to ensure that messages of sufficient importance are forwarded to the central logging machine. If important messages are not sent, Oak cannot alert us to problems. If too many messages are sent, the Oak configuration will become unnecessarily complicated in order to weed out the unnecessary messages.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

The syslog configuration file on most systems is at `/etc/syslog.conf`. An ordinary `syslog.conf` might look like this:

```
*.err;kern.notice;auth.notice       /dev/console
*.err;kern.debug;daemon.notice      /var/adm/messages
*.emerg                             *
auth.info                           /var/log/auth
auth.notice                         /dev/console
mail.info                           /var/log/mailer
daemon.info                         /var/log/daemon
local2.notice                       /var/log/inetd
```

Each line begins with a list of facility/severity levels followed by a number of tab characters and then a file name. Note that on some systems, the separator between the first column and the second column must be tabs, not spaces.

The first column in each line of `syslog.conf` describes a set of messages to match, in the format *facility.severity*. The facility is simply the name of the facility to be used, or an asterisk, which matches all facilities. The severity works differently; the line will match all severities at the named severity level and higher. So the line that begins with `daemon.info` will match all messages in the daemon facility whose severity is `info` up through `emerg`. The only daemon messages not included will be those of severity level `debug` because `debug` is the only severity of less importance than `info`. Note also that you can specify several *facility.severity* tokens separated by semicolons.

The second column in `syslog.conf` specifies where the matching messages should be sent. Typically, this is a file name, as are all the examples above with one exception. On Solaris, you can send a syslog notice to a file, to logged in users, or to a remote machine. The syntax for each is listed in Figure 6.3.

### Figure 6.3. Syslog Actions.

| **Action** | **Syntax** |
|---|---|
| Append to file | *filename beginning with slash* |
| Send to logged-in user | *username* |
| Send to logged-in users | *user1, user2, ...* |
| Send to all logged-in users | * |
| Send to a remote host | *@hostname* |

Of particular interest to us is the syntax for sending a message to a remote host. Simply add a line like the following to `syslog.conf` on your servers:

```
*.warning;kern,user,auth.notice        @LOGGER.EXAMPLE.COM
```

This sends all messages of severity level warning or higher, plus kernel, user, and auth messages at the notice level or higher to the host logger.example.com. You may choose to use a more restrictive or less restrictive set of messages to be sent to the logging host, but it is wise to use the same configuration on all your servers. If you receive an error message from one machine, you will expect to receive the same kind of message from another machine encountering the same error.

Once you have added the necessary line to your `syslog.conf`, remembering to use tabs as appropriate, you must send a SIGHUP to the syslog daemon so that it knows to reread the configuration file.[2] This must be done from a root account:

[2] Also note that if you were adding a new file for syslog to log to, you must first create the new file before sending the SIGHUP to syslogd.

```
Solaris# ps -ef | grep syslog
root    211     1  0   Sep 19 ?      0:17 /usr/sbin/syslogd
Solaris# kill -HUP 211
```

You can now send a test syslog message using the `logger` program. By example:

```
Solaris% logger -pwarn "This is a test"
```

This will send a message to syslog at the user.warn level. As configured above, this message will be sent on to the host logger.example.com. Check the logs on that machine for the test message.

Of course, the host logger.example.com must also be configured to place messages it receives into a file. It is this file that Oak will monitor. An appropriate entry may already exist in the `syslog.conf`; if not, you can add one such as:

```
*.notice;kern.debug       /usr/adm/oaklog
```

Remember to create `/usr/adm/oaklog` and send syslogd a SIGHUP as before. For the remaining examples in this chapter, however, we assume messages are being logged to `/var/adm/messages`. Be aware that this is the default location for syslog messages on Solaris, but on Linux, the default is `/var/log/messages`. On either operating system, you should check that the default `syslog.conf` is configured to send all the messages you need to the file you are monitoring.

### 6.4.2. Configuring Syslog on Network Devices

Every device uses a different syntax for configuring remote logging. Cisco IOS uses the `logging` command from configure mode:

```
Router(config)#logging 10.7.21.88
```

This will send log messages to the logging host 10.7.21.88, and by default, the facility will be local0. If you need to change the default logging facility (say it conflicts with a service you already depend on having as local0), use the `logging facility` command:

```
Router(config)#logging facility local3
```

Remember to issue a `write mem` to save your changes.

On Cisco devices running CatOS, you can configure remote logging from enable mode with:

```
switch18> (enable) set logging server 10.7.21.88
switch18> (enable) set logging server enable
```

### 6.4.3. An Introduction to Regular Expressions

The last thing we must understand before configuring Oak is the **regular expression**, an integral part of the Oak configuration language. A regular expression is syntax used to represent a pattern that a text string can either match or not match. For example, the first argument to the `grep` command is a kind of regular expression:

ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

```
Solaris% grep domain /etc/resolv.conf
domain EXAMPLE.COM
```

Grep checks every line of the file `resolve.conf` to see if the string “domain” is present. In this simple case, the regular expression is “domain”; if that text is found on any line of the file, the line is printed to the screen. Here’s a grep command with a slightly more interesting regular expression:

```
Solaris% grep do..in /etc/resolv.conf
domain EXAMPLE.COM
```

A period in a regular expression signifies that *any* character (other than a newline) can take its spot. In this case, the “m” and the “a” fill those roles. If, however, we had tried:

```
Solaris% grep do.in /etc/resolv.conf
```

there would be no matching line. Each period has to be replaced by exactly one character.

The regular expressions used in `grep` are somewhat limited unless we use special options with the program. The regular expressions in Oak are more full featured. The most common features are listed below, but a full listing of features is available in the regex man page.

Unless a character is otherwise designated for a special purpose the regular expression will match that character exactly. That is, an “e” in a regular expression simply means that an “e” must be present in the text.

### The . Character

As described above, a single period will match any character except a newline character.

### The + and * Modifiers

A character followed by an asterisk means the character can be present zero or more times. For example, the regular expression:

```
fo*bar
```

will match “foobar” as well as “fbar” and “fooooobar.” Likewise, the expression:

```
foo.*bar
```

will match all of “foobar,” “fooqbar,” and “fooqqqbar.”

A character followed by a plus sign means the character must be present one or more times. So the regular expression:

```
foo.+bar
```

will match “fooabar” and “fooaaabar” but *not* “foobar.”

### The [ ] Operator

When a number of characters are enclosed in square brackets, the regular expression will match on any one of those characters. The expression:

```
foo[123]bar
```

will match on “foo1bar,” “foo2bar,” but not “foo4bar.” This expression can be combined with the previous + and * modifiers, so that:

```
foo[123]+bar
```

will match any string that starts with “foo,” ends with “bar,” and contains one or more of the characters “1,” “2,” or “3” in the middle, such as “foo1332bar.”

If the first character inside the square brackets is a circumflex, the character in the text must be anything *other* than those listed. Thus:

```
foo[^123]bar
```

will match “foo5bar,” but not “foo1bar.”

Additionally, a hyphen used within square brackets can denote a range of characters. The regular expression:

```
[0-9]+
```

is extremely useful because it matches a series of one or more digits.

### The ^ and $ and Anchors

The ^ and $ characters are called anchors because they force the expression to be interpreted at a particular place on the text line. The circumflex denotes the beginning of the line. When placed at the beginning of a regular expression, it indicates that the next character must be the first character of the line being matched. Using grep as an example again:

```
Solaris% grep omain /etc/resolv.conf
domain EXAMPLE.COM
Solaris% grep ^omain /etc/resolv.conf
Solaris% grep ^domain /etc/resolv.conf
domain EXAMPLE.COM
```

Similarly, a dollar sign denotes the end of a line. So:

```
foobar$
```

will match a line that ends with “foobar” but not a line that ends with “foobarbaz.”

### Quoting with \

Any special character preceded by a backslash indicates the actual character should be matched. Using \.+ matches one or more periods, not one or more of any character except newline.

The backslash itself is no exception; if you wish to match on a real backslash, use \\ in your regular expression.

### Substitution with ( )

Ordinarily parentheses can be used in a regular expression to grab a section of text for later use. In Perl, for example, the regular expression:

```
foo(.+)bar
```

will match the text foo52bar, and furthermore, the string “52” will be stored in a variable that can be used later.

Oak also allows parentheses to be used in regular expressions but for a different purpose. Anything found in parentheses is replaced with underscores. This is how you will inform Oak which information in a message is redundant or private and should not be included in notifications. For example:

```
^sendmail\[(.+)\]: (.+): SYSERR.*: Cannot open btree database .+
```

We see two sections enclosed in parentheses. The second one is the sendmail queue ID number. If this message is a problem that will occur on every piece of mail processed, we do not wish to see a different log message for every piece of mail, just a single message indicating the problem. When we tell Oak that the sendmail queue ID is unimportant, it will condense many messages into one.

## 6.4.4. Configuring Oak

The Oak configuration is centered on the idea of a message **queue**. Each queue is defined to take a certain action at a specified time interval. One queue may send an email every morning. Another may send an instant message each hour. After the queues are defined, you will define regular expressions that control which messages are sent to which queues. For convenience, there is a built-in “trash” queue for messages that can be discarded.

The typical Oak configuration follows this order:

- Set global options
- Define queues
- Define regular expressions for critical messages
- Define regular expressions for trash messages
- Define regular expressions for summarizing other messages
- Define a catch-all regular expression for everything else

It is important to understand that when a message is processed by Oak, it will be matched against the list of regular expressions in order from top to bottom, and when a match is found, the message will not be checked against any further expressions. This explains the ordering above. First, we look for critical messages. If the message doesn’t match any critical messages, check if it should be thrown away, and if not, try to summarize it.

### Global Options

Oak has 10 global options available, listed in Figure 6.4. Each line in the configuration file that sets a global option begins with the key word “set.”

**Figure 6.4. Facility Types on Solaris.**

| **Option** | **Function** |
|---|---|
| set infile *file* | Define the file to be monitored |
| set nukepid | Automatically remove process IDs |
| set no nuke pid | Do not automatically remove PIDs |
| set nukeciscoid | Remove log IDs from cisco syslogs |
| set no nukeciscoid | Do not remove Cisco log IDs |
| set nukesmqid | Remove Sendmail queue IDs |
| set no nukesmqid | Do not remove Sendmail queue IDs |
| set ignorehosts *host* [*host* ... ] | Ignore logs from the listed hosts |
| set onlyhosts *host* [*host* ... ] | Process logs only from the listed hosts |
| set replacestr *string* | Replace text with *string* instead of underscores |

The `infile` option defaults to `/var/adm/messages`, but it is good practice to define it explicitly. It is also the case that the `nukepid`, `nukeciscoid` and `nukesmqid` options are all on by default. The beginning of a Oak configuration file might look like:

```
set infile /var/adm/messages
set nukepid
set nukeciscoid
set nukesmqid
```

### Defining Queues

Every queue definition begins with a line in the form "define queue *queuename*" and is followed by options for that queue. For example:

```
define queue network-gazette
 prescan
 action mail admin@example.com devnull@example.com "Daily Report"
   action-limits 1000 100 100 100
 fire 09:00
 header Daily message log
```

This defines a queue called "network-gazette" that sends an email message every day at 9:00 a.m.

The action command defines what action the queue should take when it's ready to send a message. There are currently three built-in options: mail, zephyr, and exec. Mail is for email, and zephyr is an instant messaging system in use at MIT and a number of other universities. The exec option can be used to run any external program. This can be a program that pages your operations staff or sends some other kind of immediate message. A queue can have as many actions as you like; simply list each one on a separate line, each beginning with the action command.

The arguments to the mail action are *to from subject*. In the example above, mail is sent to admin@example.com from the address devnull@example.com with the subject line Daily Report.

When the exec action command is used, the first argument is the name of the program to be run, and the following arguments are arguments to be passed to that program. The messages in the queue are sent to the standard input of the program being executed.

After an action statement, you may define action-limits specifying the limitaions on the size of messages sent through the action. The four arguments, in order, are:

- Maximum number of lines
- Maximum number of characters on a line
- Maximum number of hosts to report on
- Maximum number of logs per host

If no action limits are specified, the message size will be set to default values coded into Oak.

Next, the fire statement defines how often Oak should report messages for this queue. A number in the form *hh:mm*, using a 24-hour clock, will report every day at the given time. A number in the form **num*h|m|s, will repeat at regular intervals. For example, *25m will be triggered once every 25 minutes, and *1h will be triggered once an hour. The time can also be the string now, which instructs Oak to report immediately on messages placed in this queue. More information about using now follows. The header command simply specifies a header to be prepended to the outgoing message.

There are two special commands that can be included in the definition of a queue. One is used above: It is the prescan command. This instructs Oak that upon startup, any messages already in the logfile should be included in the first notification. If, in this example, we had to kill and restart the Oak daemon, we would still like earlier messages in the log to be included in the next morning's email. However, we do not want the instant messaging queue to send an IM including all the errors that took place already today, so the prescan option is not defined for the IM queue.

The other special command is the locking command, which tells Oak to suppress repeated notifications from this queue for a certain period of time. For example:

```
define queue network-now
     action exec /usr/local/bin/page network-admins
       action-limits 25 30 100 10
     fire now
     locking 30m
     header **** CRITICAL MESSAGE LOG ****
```

This is a queue that fires immediately. It is used to send a message to the pagers of the network administrators when a critical message arrives. Because many such messages may be logged, and because you do not want the administrators to be paged repeatedly, this locking statement will cause Oak to suppress pages matching a given line for 30 minutes. If a different message comes in, Oak will page about it, so use a queue like this carefully; make sure that redundant information is removed from log messages or you will be bombarded with notifications.

Notice that the prescan option is not set for this queue; when starting up Oak, we wish for only newly arriving messages to page the operational staff.

For completeness we will also define the following queue used in later examples:

```
define queue network-zephyr
     action zwrite network-admins oak *
       action-limits 25 100 100 10
     fire *1hr
     header Hourly message log
```

This queue sends an instant message every hour, if there is something to report.

### Defining Regular Expressions

Each regular expression begins with the keyword on followed by the regular expression and then a newline. On the next line we list the queues that should receive any matching messages. Usually, we start with the critical messages section first:

```
on ^sendmail\[(.+)\]: (.+): SYSERR.*: (.+): cannot fork:
     queues network-now network-zephyr network-gazette
on ^sendmail\[(.+)\]: WorkList for .+ maxed out at .+
     queues network-now network-zephyr network-gazette
on ^unix: WARNING: Sorry, no swap space to grow stack for pid (.+)
     queues network-now network-zephyr network-gazette
on ^(.+): %SYS-2-MALLOCFAIL: Memory allocation of (.+) bytes \
   failed from (.+), pool Processor, alignment (.+)
     queues network-now network-zephyr network-gazette
on file system full
     queues network-now network-zephyr network-gazette
```

The line matching MALLOCFAIL is split for readability but must be on only one line in the config and does not include the backslash.

Let’s examine the first regular expression above. It matches messages from sendmail when it complains about being unable to fork. The part of the regular expression just after the string sendmail is used to ignore the process ID of the sendmail program. Because we have automatic process ID removal enabled, this isn’t strictly necessary, but we can do it anyway. Note that we have chosen to match on any character between real brackets (the backslashes are necessary to indicate the brackets are present in the syslog message and are not used as special regular expression characters). We could have been more exacting and required the characters between the brackets to be digits, but that would have failed to match the underscores that Oak will automatically substitute for the process ID. The (.+) instructs Oak to replace the characters between the brackets because it is redundant information.

When Oak encounters a message that fits the criteria of this first regular expression, it will replace any of the sections within parentheses, and then add the message to the “network-now,” “network-zephyr,” and “network-gazette” queues, as defined on the next line. This way, it is sent out for immediate notification but is also included in the hourly and daily reports.

After all the critical messages are out of the way, you may list messages that never need to be viewed by putting them into the trash queue. Here are a few examples of messages we do not care to see. The last one discards root login messages, which can be a nuisance if administrators are often logging into servers. You may choose to configure this differently, or you may wish to ignore only login messages from particular users or machines:

```
on ^(.+):(.*)%SYS-5-CONFIG_I: Configured from console
     queues trash
on ^imapd\[(.+)\]: PROTERR: Connection reset by peer
     queues trash
on ^eklogind\[(.+)\]: ROOT login by (.+) \((.+)\)
     queues trash
```

Note that you do not need to define a trash queue; it exists for you by default. You may wonder if there is any point to using substitutions in these regular expressions if the messages are just going to be thrown away. They have been included only because the redundancy information was easy to encode at the time, and if we should want to move the messages out of the trash queue at a later point, we will already have the appropriate formatting.

Next, you can include all the messages that should be in the regular reports but need to have redundant information removed. Here is an excerpt from this part of the configuration:

```
on ^sendmail\[(.+)\]: (.+): SYSERR.*: Cannot open btree databas
     queues network-zephyr network-gazette
on ^sendmail\[(.+)\]: (.+): (.+): SMTP DATA-2 protocol error: 5
     queues network-zephyr network-gazette
on ^sendmail\[(.+)\]: (.+): SYSERR(.*): (.+) config error: mail
     queues network-zephyr network-gazette
```

Finally, at the very end, you may include a statement catching anything that has not already been matched:

```
on .*
     queues network-zephyr network-gazette
```

If a line like this is not included at the end, messages that do not match any of the earlier lines will simply be ignored.

Of course, the above style of organizing the Oak configuration is optional. You do not have to place critical messages first, followed by trash, other messages, and then the catch-all. You can use any method you like as long as you keep in mind the rule that Oak will match on the first line it finds and that if it does not find a maching line, it will ignore the message.

## Running Oak

Oak runs as a daemon and is invoked simply as `oak -c` *configfile*. If necessary, you can kill the program at any point and restart it. Remember that only queues defined with the `prescan` option will pick up messages already in the system log. If you change the Oak configuration file, you will need to stop and restart the program in order for it to notice the change.

If your system is set up to rotate log files, you do not need to take any special action to make Oak recognize a new file. Oak will automatically detect if the filename originally used no longer points to the file that Oak is actively monitoring. When this happens, Oak will open the new file and begin monitoring it instead, without any operator intervention.

Oak does not necessarily need to be run with root privileges. It does need to have access to read the syslog file it monitors, however. On some systems, the log file is readable only by root, and on others, it is readable by everyone. Check that the account you will run Oak from has access to read the syslog file.

## A Small Sample Configuration

Here is a small, sample Oak configuration. It is just enough to get you started working with the program.

```
# global options
set infile /var/adm/messages

# define queues
define queue testqueue
     action mail admin@example.com devnull@example.com Report
       action-limits 1000 100 100 100
     fire *5m
     header 5 Minute Test Report

# critical messages
on ^sendmail\[(.+)\]: (.+): SYSERR.*: (.+): cannot fork:
     queues testqueue

# trash
on ^(.+):(.*)%SYS-5-CONFIG_I: Configured from console
     queues trash
```

```
# other
on ^(.+): %SYS-3-CPUHOG: Task ran for (.+) msec \((.+)\),
     queues network-zephyr network-gazette
on .*
     queues network-zephyr network-gazette
```

## 6.5. Maintaining Oak

The bulk of the maintenance required for Oak is creating and updating the configuration file. As you change other components of your system, such as upgrading router software or deploying new servers, you can expect to see new and different syslog messages. The Oak configuration must be updated to reflect these changes. The risk of a stale configuration is that notifications will become long and less compact, which may make it easy to overlook important warning messages.

It is good practice to update the configuration every few weeks by looking over earlier reports and determining which information can be safely ignored or made more compact.

## 6.6. References and Further Study

There are a number of other free programs available that perform system log monitoring. For example, the `logwatch` program, available from http://www.logwatch.org/, is free and now comes installed with many Linux systems. Unlike Oak, it allows you to write customized filters that condense information into any kind of message you desire. However, the notification options are limited and the configuration is a bit tricky.

A very popular log monitoring tool is Swatch, available from http://swatch.sourceforge.net/. Swatch is similar to Oak in that it uses a simple configuration language based on regular expressions, and each regular expression can trigger a notification. The advantage Swatch has over Oak is that it is much more widely used and has been around a lot longer, making it a more stable and reliable tool. Oak allows for much more flexibility in notification options than Swatch does, and it has a few more features, but as a younger piece of software, it is not nearly as tried and true. If you encounter bugs or other problems with Oak, send a detailed description of the problem to bug-ktools@mit.edu.

The remote syslog protocol that allows syslogs to be sent between machines is described in RFC 3164. The options for the `syslogd` program and the `syslog.conf` file can be found in the Unix man pages for `syslogd` and `syslog.conf`, respectively. Regular expression syntax is described in the Unix man pages `regex` and `regexp`, and there are books available on regular expressions, such as *Mastering Regular Expressions* (O'Reilly and Associates, 2002) by Jeffrey Friedl.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003