Monitoring directories with iWatch

EYE ON CHANGE

Why wait for cron? iWatch monitors critical files and directories in real-time. This handy Perl script then notifies the user or runs a configurable command when change occurs.

BY OLIVER FROMMEL

If an intruder is loose on your system, it is important to learn about the attack as soon as possible. Several tools in the Linux environment check individual directories, and files send notice of changes. You can run one of these tools as a cronjob or write a script that runs in a loop and regularly performs file checks. In both cases, however, you are forced to either run the tool frequently (at high cost to system resources) or else settle for long intervals between checks, which could potentially open a window for an intruder.

The Linux 2.6.13 kernel introduced a solution to this dilemma. The kernel's Inotify interface provides a means for monitoring files and directories in realtime. Users need only tell the kernel which files they are interested in, and the kernel notifies the user whenever a change occurs.

Because the Inotify interface is implemented as an API, you can build your own tools to fetch and present file access information. Or, if you don't feel like writing your own a tool from scratch, you can use iWatch [1], a Perl script by Cahya Wirawan. iWatch monitors critical directories through the Inotify interface, which notifies the user by email or runs a configurable command when a change occurs.

The iWatch package is easy to install: all you need are the Linux::Inotify2, Event, Mail::Sendmail, and XML::Simple Perl modules.

Figure 1: In a simple scenario, iWatch will log changes to the monitored files at the console.

You can build the Inotify package from the source code, or, as an alternative, you can launch the CPAN interface: *perl -MCPAN -e shell* and then enter *install Linux::Inotify2*.

Inotify Events

iWatch supports two operating modes. In one mode, any information is passed to the application via the command line. In the second mode, iWatch parses an XML configuration file.

In the most simple of cases, you just append the path of the file or directory you wish to monitor at the command line, as in the following, where *test* is the directory that you want to monitor:

```
iwatch /test
```

If you then enter *touch testfile* in another window, iWatch will display the changes (Figure 1). You can see the three Inotify event types that occur on creating a new file here.

By default, iWatch will monitor the *close_write*, *create*, *delete*, *move*, *delete_self*, and *move_self* events. The documentation page on the iWatch project site has a full list of events.

You need to list the events as the parameter for the *-e* flag at the command line. Adding another option, *-r* for recursive, tells iWatch to monitor the specified directory and all of the directory's subdirectories.

The option for specifying an email address is *-m*. The *-c* flag introduces an alternative command to be executed, and the *-s* option tells iWatch to use syslog for event logging.

XML Configuration

As an alternative to the simple command line approach, iWatch can also parse an XML file for settings; This approach is definitely preferred for more complex monitoring jobs. The available XML tags are not formally specified in a DTD, XML schema, or similar; however, the format is quite intuitive.

Table 1 lists the tags along with their attributes. If you do not specify a different XML source, iWatch assumes */etc/iwatch.xml*. You can change the default by using the *-f* switch.

The whole configuration is surrounded by *config* tags. The *guard* tag also resides at this global level; its *email* attribute specifies the notification mail sender. Individual watchlists, which can contain multiple paths, reside between the *config* tags.

You can assign a different email address to each watchlist; this is useful for delegating responsibility for different servers or hard-disk areas.

The *path* tag attributes are particularly interesting. For example, *single* tells iWatch to monitor the directory content only, whereas *recursive* tells it to monitor the subdirectories below the specified path, just like the *-r* switch in command line mode. The *exception* attribute lets you exclude directories from monitoring. Listing 1 shows a simple example of an XML configuration.

Inotify has a couple of restrictions that have nothing to do with the iWatch application. For example, the kernel sets an maximum value for the number of objects to monitor. The proc file for this is */proc/sys/fs/inotify/max_user_watches*, and the default is set to 8192.

Access control is based on normal file permissions; howeve. iWatch does not check whether you actually have the required permissions. Therefore, a normal user could enter the *./iwatch /root/* command without the program saying a thing. What actually happens is that the command is ignored because the tool does not have the required permissions to monitor the directory.

The *-t* switch is fairly new, and therefore, only documented in the changelog; it has the same effect as the *filter* tag in the XML configuration. The argument is a regular expression for the filename, which lets you link a path reference with a filename pattern.

Small but Mighty

iWatch is a practical tool that makes the Inotify interface accessible to users. Instead of using the kernel API to monitor files, all you need is an intuitive XML file to set up granular change monitoring for specific areas of your filesystem. As the iWatch Perl script is easy to read, it lends itself to customization, such as adding a permissions check. ■

Listing 1: example.xml

```
01 <config>
02   <guard email="iwatch@localhost"/>
03   <watchlist>
04     <title>Configuration files</title>
05     <contactpoint email="oliver@localhost"/>
06     <path type="recursive">/etc</path>
07     <path type="exception">/etc/httpd/run</path>
08   </watchlist>
09 </config>
```

Table 1: iWatch Tags

Name	Attribute
config	–
guard	email, name
watchlist	–
title	–
contactpoint	email, name
path	alert, events, exec, syslog, type, filter

INFO

[1] iWatch: *http://iwatch.sourceforge.net*