Automate System Administration Tasks

Information

These notes were originally written in the year 2000 as part of a set of LPI Exam 101 training materials. The LPI training course at Bromley College was subsequently discontinued and some of the sections of the notes modified and incorporated into our one-day System Administration Courses. The remainder of the notes have now been made publicly available on the linuxtraining.org.uk website.

If you are a beginner please do not be put off of training courses by these notes, as they are rather technical. On the other hand if you are a more experienced Linux user we hope you find the coverage of this topic refreshingly clear.

For full details of our current Linux training please visit the site:
http://ce.bromley.ac.uk/linux

If you have reached this page from a search engine and wish to see the full contents list for the published notes please visit the site:
http://www.linuxtraining.org.uk

We hope you find these notes useful, but please remember that they apply to the 2.2 kernel. I will update them when I have the time.

Clive Gould - 21st December 2004

Automate System Administration Tasks

Objective 4

Automate system administration tasks by scheduling jobs to run in the future: Use cron to run jobs at regular intervals, use at to run jobs at a specific time, manage cron and at jobs, configure user access to cron and at services

Maintain crontab Files for Individual Users - crontab

The **crond** process allows jobs to be run on a regular, recurrent basis. The crond daemon is started from the rc files when the system boots. The **crontab** command is used to create a job for crond to run. The crond process checks all the stored cron jobs each minute looking for jobs that are due to be run. By default, when executing jobs any output is mailed to the owner of the crontab, not the standard output.

The syntax for the crontab command is shown below:

```
crontab option(s)
```

Common options used with crontab are:

Option	***Explanation***
-e	Used to edit the current users crontab using the default editor (vi)
-l	Lists the current users crontab
-r	Removes the current users crontab
- u *USERNAME*	Specifies the *USERNAME* of the user whose crontab is to be edited, displayed, or deleted. Used only by the root in conjunction with the options -e or -l or -r

Automate System Administration Tasks

To create a crontab job you invoke crontab with the -e switch, which opens vi onto a blank crontab file. This is illustrated below:

```
[clive@redhat clive]$ crontab -e

~
~
~
~
~
~
~
~
"/tmp/crontab.16966" 0 lines, 0 characters
```

It is now necessary to enter details of when you would like the job run and what command(s) you would like run. You will need to enter the **i** command to put vi into insert mode. The format used by crontab consists of six fields:

```
minute hour dayofmonth month dayofweek command
```

Each field is separated from the next by a space. The contents of the fields are explained in the table below:

Field	***Contents***
minute	Specify 00 to 59
hour	Specify 00 to 23
dayofmonth	Specify 1 to 31
month	Specify 1 to 12, or name such as jan, feb etc.
dayofweek	Specify 0 to 6 where 0 is Sunday, or name such as mon, tue etc
command	The command to be executed. Can also be a shell script. Note that only bash commands can be used in crontab file entries.

If you wish to leave a field blank you must enter an asterisk (*) in that field. You can also specify ranges of days and times using a hyphen (e.g. 08-18) and non inclusive values using commas to separate each value (e.g. mon,fri). Several format examples are shown below:

```
00 08-18 * * mon-fri who >> home/clive/userlist
```

The above example would append the list of system users every hour between 8 a.m. and 6 p.m., Monday to Friday, to the contents of the a file, userlist.

```
00 00 * * sat tar -czvf /backup/usr.tar.gz /home/*
```

The above example would perform a full backup of the contents of all users home directories every Friday night at midnight.

When you have finished entering your data into vi, press the [**Esc**] key to return to command mode and save your file using the editor command **:wq**.

```
00 08-18 * * mon-fri who >> /home/clive/userlist
~
~
~
~
~
"/tmp/crontab.17029" 1 line, 30 characters

crontab: installing new crontab
[clive@redhat clive]$
```

The crontab command saves your job under your username in the directory **/var/spool/cron** which is regularly checked by crond. You should always use the crontab command to edit this file as it read/write only by root and not intended to be edited directly.

If you have many users running cron jobs, your system could get overloaded. You can prevent particular users from scheduling cron jobs by using the files **/etc/cron.allow** and **/etc/cron.deny**. If the /etc/cron.allow file exists, then you must be listed in it, in order to be allowed to use this command. If the allow file does not exist but the /etc/cron.deny file exists, then you must not be listed in the deny file in order to use this command. If neither of these files exists, then all users are able to schedule cron jobs.

In addition to the files in **/var/spool/cron**, crontab also checks the **/etc/crontab** file regularly. The contents of a typical /etc/crontab file are illustrated below:

```
[root@redhat /root]# cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

The **run-parts** script executes each script in the designated directory on a regular basis.

The typical contents of the **/etc/cron.hourly**, **/etc/cron.daily**, **/etc/cron.weekly**, and **/etc/cron/monthly** directories are illustrated below:

```
cron.hourly:
drwxr-xr-x   2 root     root         4096 Nov 15  1999 .
drwxr-xr-x  39 root     root         4096 Jun  2 13:00 ..
-rwxr-xr-x   1 root     root           65 Aug 30  1999 inn-cron-nntpsend

cron.daily:
drwxr-xr-x   2 root     root         4096 Nov 15  1999 .
drwxr-xr-x  39 root     root         4096 Jun  2 13:00 ..
-rwxr-xr-x   1 root     root           77 Aug 30  1999 inn-cron-expire
-rwxr-xr-x   1 root     root           53 Aug 30  1999 inn-cron-rnews
-rwxr-xr-x   1 root     root           51 Jun 16  1999 logrotate
-rwxr-xr-x   1 root     root          390 Sep 14  1999 makewhatis.cron
-rwxr-xr-x   1 root     root          102 Sep 20  1999 slocate.cron
-rwxr-xr-x   1 root     root           97 Jul 21  1999 slrnpull-expire
-rwxr-xr-x   1 root     root          103 Sep 25  1999 tetex.cron
-rwxr-xr-x   1 root     root          104 Aug 30  1999 tmpwatch

cron.weekly:
drwxr-xr-x   2 root     root         4096 Nov 15  1999 .
drwxr-xr-x  39 root     root         4096 Jun  2 13:00 ..
-rwxr-xr-x   1 root     root          387 Sep 14  1999 makewhatis.cron

cron.monthly:
drwxr-xr-x   2 root     root         4096 Aug 27  1999 .
drwxr-xr-x  39 root     root         4096 Jun  2 13:00 ..
```

Automate System Administration Tasks

Each of the above files is a script which performs a periodic routine task such as rotating logs, keeping the whatis database up to date, etc.

You can make your own shell script run at a particular time, by creating the script as root, making it executable with 0700 permissions and putting it in the appropriate cron directory. Remember that the default output for any messages generated when scripts are run by cron is mail. If you wish output to be sent to the screen instead, the output needs to be redirected.

Queue Jobs for Later Execution - at

The **at** command can be used when you have job that you want to run once at a specific time in the future. Jobs created by at are actually run by the atd daemon which is usually started by the rc scripts when the system boots.

The syntax for the at command is shown below:

```
at option(s) time
```

Common options used with at are:

Option	*Explanation*
-b	Schedules a job to run as soon as the average system load falls below 1.5. (/proc/loadavg) This option is the same as the **batch** command.
-d *JOB_NUMBER*	Deletes the at job with *JOB_NUMBER*. This option is the same as the **atrm** command.
-f *FILENAME*	Reads a job from a specified command file, instead of from the terminal at the at> prompt.
-l	Lists all the jobs belonging to the current user. This option is the same as the **atq** command.
-m	Mails a user when the job completes.

Automate System Administration Tasks

The time argument may be expressed in a variety of formats. The time format consists of two fields, a time of day field plus an optional qualifier field. These are explained in the table below:

Time of Day	*Syntax*
24 hour clock	`HH:MM`
12 hour clock	`HH AM or HH PM`
keyword	`midnight, noon, teatime (4p.m.), now`

Note: If the time has already passed today the next day is assumed.

Qualifier	*Syntax*

Increments	`+COUNT TIME_UNITS`
	`Where TIME_UNITS can be minutes, hours, days or weeks`

Keywords	`today, tomorrow`

Date	`Month day year(optional) or`
	`MMDDYY or MM/DD/YY or DD.MM.YY`

Examples of time arguments are shown below:

Example	*Explanation*
`23:15 tomorrow`	Run the job at 11.15 pm tomorrow night
`4pm +3 days`	Run the job at 4 pm in three days time
`10am Jul 31`	Run the job at 10 am on 31st July in the current year
`now +15 minutes`	Run the job in fifteen minutes

When you enter the at command followed by any options and the time argument (if appropriate), the **at>** prompt will be displayed. At the prompt you can type a number of lines of commands you want to run at the time specified and press [Ctrl]+[D] to save the job, which adds it to the queue of jobs waiting to be executed.

Running at jobs uses system resources and if many users run long, complex jobs this could slow your system down appreciably.

The root can always use the at command. For other users, permission to use at is determined by the files **/etc/at.allow** and **/etc/at.deny**.

If the allow file exists, then you must be listed in it to be allowed to use the at command. If the allow file does not exist, but the deny file exists, then you must not be listed in the deny file to be able to use the at command. If neither of these files exists only the root is able to schedule at jobs.

By default the at.deny file exists, but is empty which means that all users can schedule at jobs. You can prevent users from being able to schedule at jobs by simply deleting this file.

An example of using the at command is illustrated below:

```
[clive@redhat clive]$ at now + 5 minutes
at> who > /home/clive/userlist
at> <EOT>
warning: commands will be executed using /bin/sh
job 14 at 2000-09-04 15:11
```

The above example will redirect a list of logged on users into the file /home/clive/userlist in 5 minutes time.

```
[clive@redhat clive]$ at -l
14       2000-09-04 15:11 a

[clive@redhat clive]$ at -d 14
```

The above example lists at jobs for the current user and then deletes job 14.