Automate system administration tasks by scheduling jobs to run in the future

Weight 4

Linux Professional Institute Certification — 102

Angus Lees <gus@inodes.org> Geoff Robertson <ge@ffrey.com> Nick Urbanik <nicku@nicku.org>

This document Licensed under GPL—see section 7

2005 July
Outline

Context
Objective
Introduction

at command
  Specifying the time for at

The cron System

crontab
  crontab file format
Setting up cron for root

anacron
  anacron configuration:

License Of This Document
1.111.1 Manage users and group accounts and related system files [4]

1.111.2 Tune the user environment and system environment variables [3]

1.111.3 Configure and use system log files to meet administrative and security needs [3]

1.111.4 **Automate system administration tasks by scheduling jobs to run in the future** [4]

1.111.5 Maintain an effective data backup strategy [3]

1.111.6 Maintain system time [4]
Description of Objective
1.111.4 Automate system administration tasks by scheduling jobs to run in the future

Candidate should be able to use cron or anacron to run jobs at regular intervals and to use at to run jobs at a specific time. Task include managing cron and at jobs and configuring user access to cron and at services.
111.4 Scheduling jobs [4]
Key files, terms and utilities

- `crontab`
- `at`
- `atq`
- `/etc/anacrontab`
- `/etc/at.deny`
- `/etc/crontab`
- `/etc/at.allow`
- `/etc/cron.allow`
- `/etc/cron.deny`
- `/var/spool/cron/*`
Basically

- **at** – Run a command once
- **cron** – Run a command periodically
The `at` command

`at` takes a time and a list of commands to run. Any output to STDOUT or STDERR will be mailed to the user running `at`.

```
$ at 2pm
warning: commands will be executed using /bin/sh
at> date
at> ^D
job 3 at 2002-05-08 14:00
```
The at command

The current umask, working directory and environment (except for TERM, DISPLAY and _) are saved and restored before running the job (unlike cron).

The commands to run will be read from STDIN or from a file given with -f.
Example at time specifications

at allows a very flexible time format.

17:36 Run at 5:36pm today or tomorrow.
9pm May 8 Run at 9pm on May 8th.
noon tomorrow Run at 12pm tomorrow.
now + 2 hours Run in 2 hours.

See at(1) for more details.
Queued jobs

`atq` lists a user’s pending jobs.

```
$ atq
  3 2002-05-08 14:00  a g u s
```

```
$ atrm 3 ← removes the queued job.
```

```
$ at -c 3 ← dumps the job on STDOUT.
```
crond is a daemon that reads everyone’s crontab information, spawning new tasks at the appropriate times.

```bash
crontab file
```
Replace your crontab file with `file`.

```bash
crontab -l
```
List your crontab.

```bash
crontab -r
```
Remove your crontab.

```bash
crontab -e
```
Edit your crontab (with `$EDITOR`).
**crontab file format**

**A sample crontab file:**

0 7 1 jan * echo "sleep in, you dont feel so good"
# gratuitous noise
0 17 * * mon,wed,fri wall%meeting in 5 minutes%
0 9-18/2 * * mon-fri $HOME/bin/cron.bihourly

Line based, hash comments, ignored blank lines, etc

- Minute (0-59)
- Hour (0-23)
- Day of month (1-31)
- Month (1-12 or jan-dec)
- Day of week (0-7 or sun-sat)
- Step
- Wildcard
- Ranges
- Lists

See crontab(5) for:

- Environment variables
- Providing STDIN
A few extra issues arise when editing `/etc/crontab` (and similar “system” crontab files):

- Don’t use `crontab -e`, edit `/etc/crontab` directly.
- A new column (after timespec, before command) gives the user to the command run as.
- Distributions often create directories for “common” frequencies. It usually makes much more sense to place a script in there, rather than adding your own crontab lines. Debian, Red Hat runs any scripts in `/etc/cron.{daily, weekly, monthly}` – but these are triggered from normal entries in `/etc/crontab`, so there’s no real mystery here.
- `/etc/cron.d/*` is read in addition to `/etc/crontab` (they also have the extra user field).
Apparently some people turn their machines off.

If your computer is always turned off at night (for example), then daily jobs which are usually scheduled to run in the wee hours, will never be run. This is a problem.

anacron fixes this by running any missed jobs after a reboot (or other times, like AC-on for laptops).
Since anacron can’t use the crontab files, it has its own simplified `/etc/anacrontab`. If you only use the standard `/etc/cron.daily,monthly,weekly`, then no further configuration will be necessary. Otherwise, see `anacrontab(5)`.

Note that the frequency of anacron jobs can only be specified in days.
Topics Covered

Context

Objective

Introduction

at command
   Specifying the time for at

The cron System
   crontab
   crontab file format
   Setting up cron for root

anacron
   anacron configuration: anacrontab

License Of This Document
Automate system administration tasks by scheduling jobs to run in the future.

Angus Lees

Context
Objective
Introduction
at command
The cron System
anacron
License Of This Document

Copyright © 2005, 2003 Angus Lees <gus@inodes.org>, Geoffrey Robertson <ge@ffrey.com> and Nick Urbanik <nicku@nicku.org>. Permission is granted to make and distribute verbatim copies or modified versions of this document provided that this copyright notice and this permission notice are preserved on all copies under the terms of the GNU General Public License as published by the Free Software Foundation—either version 2 of the License or (at your option) any later version.