Linux Certification Study Group

A set of LaTeX presentations

Angus Lees

gus@inodes.org

Granville College of TAFE

Sydney, Australia

111.4 Scheduling jobs [4]

Objective

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. Tasks 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`
`/etc/anacrontab`	`atq`
`/etc/crontab`	`/etc/at.deny`
`/etc/cron.allow`	`/etc/at.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 gus
```

`$ atrm 3 ↩` removes the queued job.

`$ at -c 3 ↩` dumps the job on STDOUT.

Queued jobs

`atq` lists a user's pending jobs.

```
$ atq ↩
3 2002-05-08 14:00 a gus
```

`$ atrm 3 ↩` removes the queued job.

`$ at -c 3 ↩` dumps the job on STDOUT.

`crontab`

`cron` is a daemon that reads everyone's `crontab` information, spawning new tasks at the appropriate times.

`crontab` ***`file`***	Replace your crontab file with *file*.
`crontab -l`	List your crontab.
`crontab -r`	Remove your crontab.
`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
```

`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

`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
```

- Minute (0-59)
- Hour (0-23)

`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
```

- Day of month (1-31)
- Month (1-12 or jan-dec)
- Day of week (0-7 or sun-sat)

`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
```

- Step
- Wildcard
- Ranges
- Lists

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
```

See crontab(5) for:

- Environment variables
- Providing STDIN

cron from root

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 (for example) 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.
- *(Debian specific?)* `/etc/cron.d/*` is read in addition to `/etc/crontab` (they also have the extra user field).

anacron

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).

anacron

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.