Cron Jobs

Cron is a very useful tool for scheduling periodic tasks on any Unix-like operating system. Often these scheduled tasks are named **Cron Jobs**.

Some useful use-cases are:

- periodic system backups
- cleaning up temporary files
- removing or minimizing log files
- monitoring system resources such as CPU usage, disk space, ...
- running maintenance tasks
- ....

Basically each user can create cron jobs on a system with tasks specific for that user. The "system tasks" are scheduled separately by the root user.

Crontab

To manage cron jobs, one can use the `crontab` (cron table) utility. `crontab` is the program used to install, deinstall or list the tables used to drive the `cron` daemon.

Each user can have their own cron table. Commands defined in any given crontab are executed under the user who owns that particular crontab.

You can execute crontab if your name appears in the file `/usr/lib/cron/cron.allow`. If that file does not exist, you can use crontab if your name does not appear in the file `/usr/lib/cron/cron.deny`. If only `cron.deny` exists and is empty, all users can use crontab. If neither file exists, only the root user can use crontab. The allow/deny files consist of one user name per line.

The files of `crontab` can be found in `/var/spool/cron/crontabs`. However, they are not intended to be edited directly but rather managed via the crontab utility.

To **list** your current scheduled tasks, one can use the `-l` option:

```
crontab -l
```

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
```

```
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

# Send a message via MQTT
* * * * * echo "Another minute has passed ... $(date)" >> /tmp/timing.log
```

To edit the current user's cron jobs, use the `-e` option:

```
crontab -e
```

Each line of a crontab file represents a job, and looks like this:

```
# ┌───────────── minute (0 - 59)
# │ ┌───────────── hour (0 - 23)
# │ │ ┌───────────── day of the month (1 - 31)
# │ │ │ ┌───────────── month (1 - 12)
# │ │ │ │ ┌───────────── day of the week (0 - 6) (Sunday to Saturday;
# │ │ │ │ │                                   7 is also Sunday on some
systems)
# │ │ │ │ │
# │ │ │ │ │
# * * * * * <command to execute>
```

The syntax of each line expects a cron expression made of **five fields which represent the time to execute the command**, followed by a shell command to execute. Basically a cron job is executed if the current time matches the expression.

Each value column can have either of the following:

- an asterisk `*` which matches every discrete possible value for that column
- a list of comma separated values like `3,5`.
- a range of numbers, where the start and end value are seperated using a hyphen `-`. The specified range is inclusive. For example `2-4` matches `2`, `3` and `4`.
- Step values can also be used in conjunction with ranges. Following a range with `/<number>` specifies skips of the number's value through the range. For example, `0-23/6` can be used in the hours field to specify command execution every six hours, matching `0`, `6`, `12` and `18` hours.

- Steps are also permitted after an asterisk `*`, so if you want to say *every two hours*, one can use `*/2`.

Some examples

Cron Time Expression	Description
`0 * * * *`	Every hour at minute `0`
`* * * * *`	Every minute
`* 0 * * */2`	Every other day of the week at midnight
`30 16 * 1-6,9-12 1-5`	At 16:30 on every day-of-week from Monday through Friday in every month from January through June and every month from September through December.
`0 20 * * 6,0`	At 20:00 on Saturday and Sunday.

The specification of days can be made in two fields: `month day` and `weekday`. If both are specified in an entry, they are cumulative, meaning both of the entries will get executed .

Some other rules are:

- Blank lines, leading spaces, and tabs are ignored.
- Lines whose first non-white space character is a pound-sign (#) are comments, and are not processed.
- Comments are not allowed on the same line as the cron commands, since they are considered a part of the command.

Non-standard

While not standard in every cron implementation, some support some extra functionality.

Instead of the first five fields, one of eight special strings may appear:

String	Description
`@reboot`	Run once, at startup.
`@yearly` or `@annually`	Run once a year, same as `0 0 1 1 *`.
`@monthly`	Run once a month, same as `0 0 1 * *`.
`@weekly`	Run once a week, same as `0 0 * * 0`.
`@daily` or `@midnight`	Run once a day, same as `0 0 * * *`.
`@hourly`	Run once an hour, same as `0 * * * *`.

Note that startup, as far as `@reboot` is concerned, is the time when the cron daemon starts up. In particular, it may be before some system daemons, or other facilities, were startup. This is due to the boot order sequence of the machine.