

Scheduling tasks with the *at* command

Run a task unattended at any time in the future with `at`

Usually, when you press return, a command runs immediately. But what if you wish to have a command run in the middle of the night, or next week? You could log in at those times and run them yourself. However, the `at` command allows you to schedule commands to run unattended, automatically, at any time of any day you specify.

To use the `at` command, the `at daemon` (`atd`) needs to run

A constantly running daemon process runs the jobs scheduled by `at` as and when required. If the `at` daemon doesn't run on a particular machine, then your scheduled jobs cannot run. Find this out by typing:

```
ps auxww | grep atd
```

If you receive no response, then ask your system administrator to install the daemon for you, or install it yourself if you have the access and the ability. Your administrator could have limited your ability to schedule jobs. If so, try asking that he or she removes such restrictions for you.

Type `at`, followed by a time specification

To schedule a job to run at a particular time, you type `at`, followed by a space, followed by the time you wish the job to run. For example:

```
at noon tomorrow
```

or:

```
at 1:30pm 23 January 2005
```

or even:

```
at teatime + 2 days
```

When you press return, the `at>` prompt appears.

`At` allows a large number of time/date specifications

As you can see in the examples above, the `at` command allows a refreshingly flexible approach in how you can specify the date and time you wish a job to appear. If you represent a date, though, you have also to specify a time first. You can specify dates and times like this:

- The time in 24 hour format (23:30) or 12 hour format (11:30pm)
- The time as the words now, noon, midnight or teatime (4pm)
- A US format date (12/09/2005), or UK format date with dots (09.12.2005)
- The month's name, day, and optional year (24 Jan 2005 or 1 February)
- As some time period relative to another time period:
(eg: now + 3 days or teatime + 1 week or 4pm tomorrow)

Type the commands you wish to run after the `at` prompt

When after specifying a valid time/date format, the `at` command prompts you to type the commands you wish to run at that specified time. You could type a collection of commands, and press return after each, just as on the shell, except, of course, they don't run immediately. Or you could specify that `at` runs a pre-prepared *shell-script*, which you have given the execute permission which, in turn, runs a list of commands.

Press `ctrl + d` when you have finished your list of commands

After you have typed your command or commands, and have pressed return, press `ctrl + d` to indicate you have finished. It lets you know that it has scheduled your job, and gives it a job number.

Use the `-f` switch to specify a file with the list of commands

Instead of typing just the time/date specification, and then pressing return, and then issuing the commands you wish `at` to schedule for you, you can specify a file containing that list of commands with the `-f` switch, thus:

```
at 5pm + 3 weeks -f commands.txt
```

Type `atq` to see your user's queue of `at` jobs

To see any jobs for your user still waiting in the queue, type `atq` and press return. It shows a list of pending jobs, similarly to this:

```
10      2005-11-01 12:00 a root
11      2005-12-03 17:45
```

The left-hand column represents the *job ID*, followed by the date and time for the system to run the job, and the user-name for the job to run under.

Type `atrm` followed by the `at` job number to remove an `at` job

To remove a pending job from `at`'s queue, first find its job number with the aforementioned `atq` command, and then run the `atrm` command, with the job number in question as its argument. Eg:

```
atrm 10
```

Run `atq` again to confirm that you have removed it from the queue.

Add users in `/etc/at.deny` whom you wish not to allow `at` scheduling

If you have superuser access, you can create or edit an `at.deny` file in `/etc` which lists any users, one per line, whom you wish not to have the ability to schedule jobs with the `at` command.

Scheduling repeated tasks with the *cron* system

The *at* command lets you schedule one-off tasks

The *at* command, whilst easy and intuitive to use, has a shortcoming in only allowing you to schedule individual future events. If you want to schedule regular, repetitive events, like backing up files or deleting old logs, then it doesn't provide much help.

The *cron* system lets you schedule repeated tasks

Although you can schedule a specific task at a pretty specific time with it, the *cron* system's main functions to provide you with an easy way of setting up repetitive tasks, and in specifying exactly how frequently these repetitions happen.

Your system needs to run the *cron* daemon to schedule tasks

As with the *at* command, the *cron* system requires a *cron daemon* (*crond*) to run. This persistent process checks every minute to see if any tasks need repeating. Your system's administrator has probably installed it, as many common administrative tasks rely on it. However, as with *at*, he or she could have prevented your user's access to its facilities, so make the appropriate request if any of what we work through below fails for you.

You edit your table of *cron* tasks (your *crontab*) with a text editor

Each user has a table of tasks. This table exists as a simple text file, which you can edit with *vi*, for example. However, where the specific location for these files lies on your system depends on its configuration.

You use the `crontab` command to manipulate your table of tasks

You use, appropriately enough, the eponymous *crontab* command to examine, remove and edit your user's *crontab*. Appropriate switches tell the system which action you require. You could search for where *cron* actually stores this file, but the *crontab* command finds it for you, and does safety checks when you try to save a new or edited *crontab* file.

Use `crontab -l` to list your table of tasks

Use the `-l` (lowercase L) switch on the *crontab* command to list the contents of your *crontab* file. If you have none installed, the command responds with nothing. Otherwise, it lists it. If you have a long list, pipe the command's output through a pager program, like *less*:

```
crontab -l | less
```

Use `crontab -r` to remove your table of tasks

You can remove your *crontab* file completely, wiping it and all memory of any tasks allocated in it. To do so, use the `-r` switch on the `crontab` command. Use this switch with care!

Use `crontab -e` to edit your table of tasks

To edit your *crontab* – to add, modify or remove individual repetitive tasks – use the `-e` switch on the `crontab` command. On pressing return, it loads your *crontab* into your preferred text editor (usually *vi*). Once you save and exit the editor, *cron* notes your changes immediately. If you merely exit without saving, then *cron* ignores your changes, of course.

The *cron* system Emails you the output of any run tasks

When you schedule a task, what happens if it outputs something? When you run it at the shell, then you can see immediately any standard or errored output. But if *cron* runs your command, then it mails you with any and all of the command's output. If you do not want it to do this, then redirect its output `> /dev/null` (as discussed previously).

Your *crontab* requires a specific format

The *crontab* file requires a precise format for you to specify what task you want to repeat when. If you make any syntax errors, then the *cron* system warns you when you try to save the file.

Begin any comments, which `crond` ignores, with `#`

When editing your *crontab*, you can place notes to yourself (comments) anywhere, so long as the first character in a line appears as a hash (`#`); the *cron* interpreter skips over such lines. You can use comments to remind yourself what specific listed scheduled tasks do, for example.

Specify an Email address for `crond` to use with `MAILTO=`

By default, *cron* tries to send the aforementioned Emails to your *username@your machine*. You can override this by having, on a line in the file, `MAILTO=` followed immediately by your preferred Email address. Eg:

`MAILTO=fred@bl oggs. co. uk`

Specify one scheduled task per line

If not a comment, blank line or a setting like MAILTO, then a line in the *crontab* probably specifies a task. One line represents one task. Remember that the presence of a carriage return determines a line, so even if your terminal's width cannot show a full line, and it appears to wrap, it could nevertheless represent just one task's configuration.

The task specification contains six “columns”

Each task configuration line contains six “columns”. By columns (or fields), we merely mean entities separated by spaces or tabs. You can use as many spaces as you wish between columns, to make them stand out.

The first five “columns” represent when to run the task

The first five columns represent precisely when and how often you want tasks to repeat. Each column represents a different division of time, and can contain numbers or appropriate symbols. For example, the first column represents the minute of an hour. So a 5 in this column represents 5 minutes past an hour.

The sixth “column” contains the command to run for the task

The final, sixth “column” determines the actual command-line task you wish to run at the times and frequencies specified to its left. You could include a standard *Unix*-like command here. For more elaborate tasks, refer to the exact path of a pre-prepared shell script. For example:

```
/home/fred/scripts/backup.sh
```

Put a * in a column to specify every time-period

In the first five “when” columns, the first from the left represents minutes of the day and the second hours. So the following means “run at 07h15 every day” (assuming further columns contain appropriate entries):

```
15 7
```

But what if you want something to recur, say, at 15 minutes past every hour? Simply place an asterisk (*) within the second, hour, column:

```
15 *
```

If you want *cron* instead repeatedly to run a task every minute of every hour of the day, the following suffices - * in both columns:

```
* *
```

Note that a proper task line has to contain the other columns too. We show the first two here in isolation just for explication.

The “when” columns: minutes hours days-of-month months week-days

We explain here what the five “when” columns indicate, from left to right, and give some examples:

- Column 1 – *Minutes*. You specify which minute in the hour you wish the task to run. For example, a 15 here means at a quarter past an hour.
- Column 2 – *Hours*. You specify which hour (out of the 24) in the day you wish the task to run. 20 as the value to represents 8pm, for example.
- Column 3 – *Day of the month*. You specify which day in the month you wish this command to run. 31 here, therefore, represents the maximum. A 1 means, of course, the first of the month. A * here means that you want the command to run *every* day of the month.
- Column 4 – *The month*. You specify which month in the year you wish the command to run. 1 stands for January, and 12 for December, of course. You can also use the full or abbreviated month name. A * here means that you wish to run this task *every* month of the year.
- Column 5 – *The day of the week*. You specify which day of the week (from Monday to Sunday) you wish the command to run. 1 represents Monday and 6 represents Saturday. You can use either 7 or 0 to represent Sunday. Or you can use the full or abbreviated day name. A * here means that you wish to run this task *every* day of the week.

Some examples of valid *when* columns

Each of the following represents a valid *cron* “when” construction:

```
30 00 * * * rm /home/fred/junk.txt
```

This means “*Remove the junk.txt file at half past midnight every day of every month – for every day of the week*”.

```
15 * 12 25 * /home/fred/scripts/jingle.sh
```

This means “*At a quarter past the hour – each hour of every Christmas (whichever day of the week it happens to fall) – run fred's jingle.sh*”

```
MAILTO=fred@bloggs.co.uk
* * * * * ls -la /etc
```

This means “*For every minute, of every hour, of every day, of every month – and any day of the week run the ls -la command on the /etc directory*”. The preceding MAILTO line ensures that the system sends the command's output to Fred's appropriate mailbox.

```
00 23 * * 07 /home/fred/scripts/work_tomorrow.sh 2> /dev/null
```

This means “*At exactly 11pm, any and every Sunday, run the work_tomorrow.sh script. Forward via Email all output except any resulting errors*”. If the 2> confuses you, read over the earlier “Pipes and Redirections” section.

You can specify lists of times/dates separated by commas (,)

If you want a task to repeat at different periods, you don't have to respecify it on a separate line: instead, you can include the different values, separated by commas, in the appropriate column. For example, to run something at quarter past and a quarter to every hour of the day, use the following:

```
15,45 * * * *
```

Or tell the system that you wish to run something at the midnight of every Monday in February, March and August:

```
00 00 * 2, 3, 8 1
```

Or specify that you want something run every Saturday at and Sunday in January at 10am like this:

```
00 10 * 1 6, 7
```

Or the same, like this:

```
00 10 * Jan Sat, Sun
```

You can specify ranges with a dash between the start - end value

Rather than specify every option with commas, you can specify a range by placing the first value to the left of a dash, and the second to the right of the dash. So, to specify that you want a task to run on the hour between 9am and 5pm, every day except weekends:

```
00 9-17 * * 1-5
```

You can specify more than one range, separated by commas. The following means "At midnight every day from February to April, in July, and from September to December":

```
00 00 * 2-4, 7, 9-12 *
```

You can step through ranges by dividing them with a slash (/)

If you want, say, to run something every 5 minutes, you could specify it like this:

```
0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55 * * * *
```

But what a pain! Fortunately, you can specify "every divided by 5", which means the same:

```
*/5 * * * *
```

The following runs something every second month, on the 10th of that month, at 32 minutes past every third hour (ie: eight times that day):

```
32 */3 10 */2 *
```

You can use some **@** shortcuts for the time specifications

You can replace all of the five when-columns explained above with any of the following shortcuts, all beginning with the at (@) sign:

- @hourly - run once an hour, every hour, on the hour
- @daily - run once every day at midnight
- @midnight - the same as @daily.
- @weekly - run at midnight every Sunday of every week
- @monthly - run at midnight on the first day of every month.
- @yearly - run at midnight on the first of January
- @annually - the same as @yearly
- @reboot - run just once, when *cron* starts up after a server reboot

The *cron* system uses additional directories

Depending on your system's configuration, you could find the actual *crontab* files that the *crontab* command accesses within the `/var/spool/cron/crontabs/` directory. Also depending on how your administrator or distribution has configured your system, in addition to the tasks specified in each user's *crontab*, the *cron* system runs commands it finds in specifically allocated directories. See, for example, if your `etc/` directory contains the following directories: `cron.daily/`, `cron.hourly/`, `cron.monthly/` and `cron.weekly/`. If so, the *superuser* can place shell-scripts within these directories, or symbolic links to shell-scripts, and *cron* then runs them at the frequency the directory's name suggests.

Further reading

Summaries of popular administrative commands

<http://linux.juxta.com/command.html>

Popular *Unix* performance-monitoring tools for *GNU/Linux*

<http://www.informit.com/articles/article.asp?p=29666&seqNum=1>

A discussion on using *cron* to schedule tasks

<http://www.uwsg.iu.edu/usail/automation/cron.html>

A concise summary of much of GNU/Linux and its day to day operation

Linux in a Nutshell, 4th Edition

Siever, Figgins and Weber 2003, O'Reilly Media, ISBN: 0596004826

A book distilling tons of tricks and tips in managing and using *GNU/Linux*

Linux Cookbook

Carla Schroder 2004, O'Reilly Media, ISBN: 0596006403