Cron, at

ON THE DOT

The cron and at utilities help automate processes on a Linux system. You can set up automatic backups or even wake up in the morning with a track from your MP3 collection.

BY HEIKE JURZIK

In Linux, there is no need to take notes as reminders of tasks you need to perform in the future. The at program lets you schedule tasks right now, and cron handles recurring jobs.

A daemon runs in the background to ensure the tasks are performed according to the schedule. The daemon checks for new jobs once a minute. The daemon for at is named *atd*, and the cron daemon is called *cron(d)*.

At Your Service

You can call at in the command line and pass the time to perform a job. Then type more commands in a shell, and quit at by pressing [Ctrl] + [D]:

```
nonumber
$ <B>at 07:00<B>
warning: commands will ➘
be executed using /bin/sh
at> mpg321 -z /home/huhn/mp3/*
at> <EOT>
job 6 at 2006-01-03 07:00
```

Command Line

Although GUIs such as KDE or GNOME are useful for various tasks, if you intend to get the most out of your Linux machine, you will need to revert to the good old command line from time to time. Apart from that, you will probably be confronted with various scenarios where some working knowledge will be extremely useful in finding your way through the command line jungle.

This tells the Mpg321 command-line player to wake you on the dot at 7 am by playing a random selection of songs from the */home/huhn/mp3/* directory – of course, this assumes that your computer is switched on.

Table 1 gives an overview of the most common notations for time. Note that at is persistent; that is, it will keep running after you reboot your machine, and it will remember the schedule for the next six months.

After completing a task, at sends a mail message with the job status to the job owner, no matter whether the job completed successfully or not. To allow this, you need a working mail server configuration (at least for local deliveries). For commands that do not create output by default, such as *rm*, *mv*, or *cp*, you can enforce an email message. To do so, set the *-m* flag, as in *at -m 13:31*.

Displaying and Deleting Jobs

Scheduled at commands are stored in the queue. You can display the queue on screen by calling *at -l* or *atq*:

```
nonumber
$ <B>atq<B>
7     2006-01-03 07:00 a huhnix
11    2006-01-03 12:00 a huhnix
12    2006-06-09 14:24 a huhnix
```

Unfortunately, at is not very talkative at this point; it just tells you the job number at the start of the line, the date and time, the queue name (*a*), and the username. The list does not tell you what jobs are scheduled. Additionally, you only get to see your own jobs as a normal user; only the system administrator gets to see a full list of scheduled jobs.

If you want more details on what the future holds for you, become root, and change to the at job directory below */var/spool*, for example */var/spool/atjobs/* (for Suse Linux) or */var/spool/cron/atjobs/* (for Debian) – the text files tell you exactly what commands will be run.

To delete an at job, give the *at -d* or *atrm* command, specifying the job number:

```
nonumber
$ <B>atrm 7 11<B>
$ <B>atq<B>
12   2006-06-09 14:24 a huhnix
```

Access Privileges

Two files, */etc/at.allow* and */etc/at.deny*, control who is permitted to work with at. Most distributions tend just to have an *at.deny* file with a few “pseudo-user” entries for *lp* (the printer daemon) or *mail* (for the mail daemon). If you create an *at.allow* file as root, you need entries for all users who are permitted to run at jobs – *at.deny* is not parsed in this case.

A Cron for All Seasons

If you are looking for a way of handling regularly recurring tasks, repeatedly running at is not recommended. Instead, you should investigate the other option that Linux gives you. Cron also runs in the background and runs jobs at regular intervals. Again, the cron program just needs the machine to be up, as it “remembers” scheduled jobs when you re-

Listing 1: Debian crontab entries

```
nonumber
17 *    * * *   root    run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || run-parts --report /etc/cron.daily
47 6    * * 7   root    test -x /usr/sbin/anacron || run-parts --report /etc/cron.weekly
52 6    1 * *   root    test -x /usr/sbin/anacron || run-parts --report /etc/cron.monthly
```

boot your machine. In fact, the two programs have even more in common: just like at, cron mails the owner account to confirm that a job has completed successfully.

Individual tasks are referred to as cronjobs, and they are managed in the crontab. This is a table with six columns that defines when a specific job is to be performed. Each command in the crontab occupies a single line. The first five fields describe the time, whereas the sixth field contains the program to be run, including any parameters.

As a normal user, you can create a crontab at the command line by running the crontab program. The command for doing this is *crontab -e*, where the *-e* parameter indicates that you will be editing the table.

As the system administrator you can additionally modify the crontabs of any user by specifying the *-u* parameter and supplying the account name:

```
crontab -u huhn -e
```

By default, this calls the Vi editor – if you prefer a different text editor, just set your *$EDITOR* environmental variable to reflect this, as in:

```
export EDITOR=/usr/bin/kwrite
```

To make this change permanent, add this line to your Bash configuration file, and reparse the configuration file by entering *source ~/.bashrc*.

Well Structured

Crontab lines are not allowed to contain line breaks. Comments in the file are indicated by a pound sign at the start of the line. The six fields contain the following information in this order:

- Minute: Values from *0* to *59* are possible, as is the * wildcard
- Hour: Values from *0* to *23* or *
- Day: Values from *1* to *31* or *
- Month: *1* through *12*, *Jan* through *Dec*, *jan* through *dec* or *
- Weekday: *0* bis *7* (where both *0* and *7* mean "Sunday"), *Sun* through *Sat*, *sun* through *sat* or *

advertisement

- Command: the command to run including options; alternatively, this can be the name of a script with more commands

If you would like your computer to wake you at seven every morning, just add an entry like this one to your crontab:

```
0 7 * * * mpg321 -z ⮐
/home/huhn/mp3/*
```

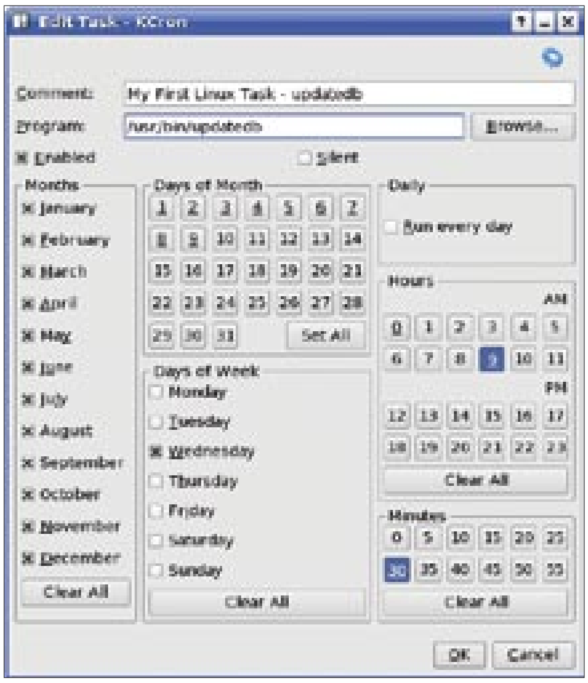
The values in the individual fields can be comma-separated. To stop your musical alarm clock from ringing on Saturdays and Sundays, add the following to the fifth field for the weekday

```
0 7 * * 1,2,3,4,5 mpg321 -z ⮐
/home/huhn/mp3/*
```

You can also specify a range using dashes (*1-5*). But using weekday names makes entries more easily readable:

```
0 7 * * mon-fri ...
```

A combination of times can also be useful. If the fourth field (for the month) has values of *1-4,7,10-12*, this means "January through April, July, October through December". A slash followed by a number also defines regular periods of time; for example, */2* in the second column means "every two hours" and *1-6/2* means "1,3,5".


Edit Task - KCron

Comment: My First Linux Task - updatedb
Program: /usr/bin/updatedb
[✓ Enabled] Silent
Months: [✓ January] [✓ February] [✓ March] [✓ April] [✓ May] [✓ June] [✓ July] [✓ August] [✓ September] [✓ October] [✓ November] [✓ December]
Days of Week: Monday, Tuesday, [✓ Wednesday], Thursday, Friday, Saturday, Sunday
Daily: Run every day
Hours: [✓ 9]
Minutes: [✓ 30]

Figure 1: Kcron provides a convenient GUI for managing cron settings.

Table 1: at Time Formats

Format	Meaning
16:16	16:16 hours today.
07:00pm	19:00 hours today (if you do not specify *am* or *pm*, *am* is assumed).
now	Right now
tomorrow	Tomorrow
today	Today
now + 10min	In ten minutes time; you can also specify *hours*, *days*, *weeks*, and *months*.
noon tomorrow	At 12:00 the next day; at also understands *teatime* (= 4:00 pm) or *midnight*.
6/9/06	June 9, 2006; alternative notations for the date are e.g. *6.9.06* and *6906*.

The cron tables for users are stored in the */var* directory. Every distribution has a different approach to sorting the tables. For example, Debian stores them in */var/spool/cron/crontabs/* and sorts by username; Suse Linux uses the */var/spool/cron/tabs/* directory instead.

As you do not have read permission for this directory as a normal user, you can display your own cron table at the command line by running the crontab program:

```
$ <B>crontab -l<B>
10 8 * *  mon-fri mpg321 -z ⮐
/home/huhn/mp3/*
```

To delete individual entries, launch the editor again by entering *crontab -e*; if you intend to delete the whole table, you can run *crontab -r* instead.

Global Cron Tables

Cron not only handles user-specific lists, it also helps the root user with system administration tasks. Working as root, take a look at the */etc/crontab* file; this tells you which jobs cron handles for you. Depending on the distribution, the global crontab can vary; Debian has the entries shown in Listing 1, for example.

In contrast to normal user crontabs, the global contrab has a seventh field which contains the name of the user with whose privileges the command will run – this is typically *root*. This list tells us that the cron daemon runs the *run-parts --report /etc/cron.hourly* with root privileges, once an hour at 17 minutes past the hour, and at 6:52 am on the first day of each month cron runs *run-parts --report /etc/cron.monthly*. Cron takes care of the daily chores (the executable scripts in */etc/cron.daily*) at 6:25 am, including the *logrotate* script, which rotates, compresses, and sorts logfiles.

If you do not run your computer 24x7, it can make sense to modify these entries and specify times when you know your computer will be up:

```
25 17   * * *   root    ⮐
test -x /usr/sbin/anacron || ⮐
run-parts --report ⮐
/etc/cron.daily
```

GUI-Based Helpers for Cron

You'll find numerous GUI-based tools to help you create a cron table. Gnome users will discover Gcrontab, a well-organized, and easy-to-use program that lets you put together a schedule with a few mouse clicks. The KDE KCron tool is even easier to use, however, it does not display the entries it creates in cron syntax, and thus it does not help you become more familiar with cron (Figure 1). But in the end (and as in most cases), the command line gives you more flexibility, and you will be much faster typing entries than clicking and pointing. ■

THE AUTHOR

Heike Jurzik studied German, Computer Science and English at the University of Cologne, Germany. She discovered Linux in 1996 and has been fascinated with the scope of the Linux command line ever since. In her leisure time you might find Heike hanging out at Irish folk sessions or visiting Ireland.