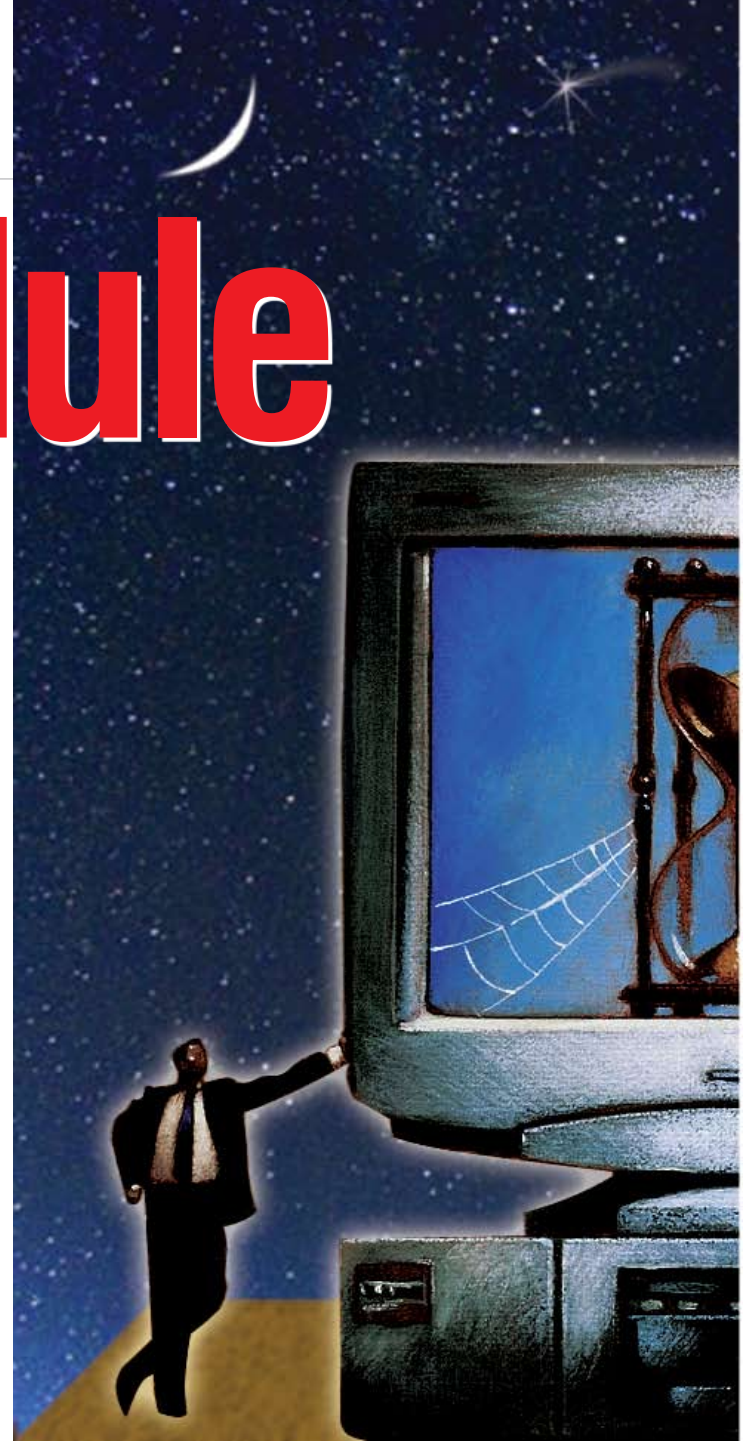
Schedule Your Tasks

It's very easy with cron—so let's get started.



Have you ever wished to run applications on given schedules? Consider the following issue. Some ISPs nowadays provide free Internet usage during off-peak hours, i.e., after midnight. For example, BSNL DataOne broadband users enjoy free downloads between 2 a.m. and 8 a.m. But staying sleepless to accomplish those free downloads seems quite a strain. What if you missed getting up at 8 a.m. and the downloads ate up your bandwidth and money? Scheduling your downloads within free bandwidth hours could be a way out of these issues. But, how do we achieve this?

Linux administrators (or even those geek users) very often need to execute some programs on a regular basis. For example, the admin might need to monitor the disk usage of a system. His best bet in most cases would be *cron*, a handy solution to execute several tasks at a given time schedule. It is a utility written by Brian Kernighan, made available from UNIX version 7. Let us dig into this classic UNIX tool and find out how it can schedule our downloads too.

Consider some of the scenarios when you can use *cron*:

- If you run a website that deals with a large number of images and you want to create thumbnails automatically during a given time period every day or week.
- If you want to keep track of your back-ups synchronised easily without much pain and effort.
- Most importantly, if you want to run file downloads and torrents in a specified time.
- If scheduling of automatic system updates is required.

To define *cron* in a classical way, it is a daemon that runs in the background as a service. In order to create scheduled jobs, we use the command *crontab*.

How to create a scheduled job

Open a terminal and enter the following command:

```
crontab -l
```

It will list the currently installed crontable:

To edit the list of jobs in *cron*, you can run:

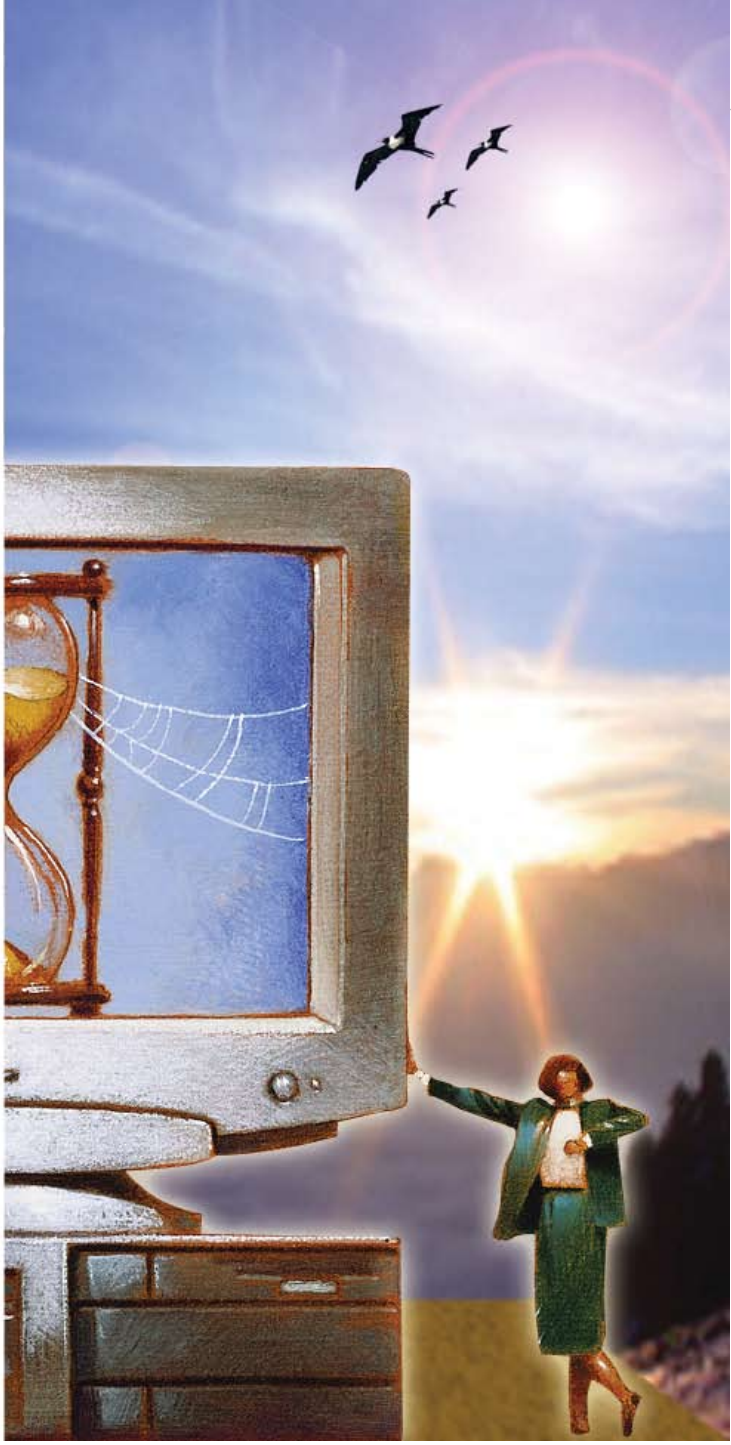
```
crontab -e
```

It will open the default text editor to let you manipulate the jobs. After you are done making changes, save and exit the editor. It will immediately activate all your *cron* jobs.

Or alternatively, open the terminal and enter the following command:

```
gedit joblist.cron
```

This will launch the *gedit* text editor (replace *gedit* with *kwrite* if you use KDE) after creating a file called *joblist.cron*. Enter the following line in the file:



```
00 * * * * date >>/tmp/hourlytime.log
```

Save and exit. Now, back in the terminal, run the following commands:

```
crontab joblist.cron
```

…followed by:

```
crontab -l
```

This will now list the following:

```
00 * * * * date >> /tmp/hourlytime.log
```

Figure 1: crontab syntax

I'm sure you must be wondering how to read the above line. There is a special syntax in which *cron* jobs are to be entered. Figure 1 explains it.

Environment settings

You might have thought that since *cron* runs on a shell, it will use the same environment settings as the parent shell. But that is not true. We have to specify separate environment variables by adding them to *crontab*. For example:

```
DISPLAY=:0
00 10 * * * /usr/bin/gedit
```

I recommend that you add the DISPLAY environment variable line on your *crontab* entry if you are scheduling to run applications that need X (GUI); else, it will not work.

Writing good *cron* jobs

The *cron* job line consists of the following five parameters separated by spaces:

1. Minute (from 0-59)
2. Hour (from 0 to 23)
3. Day of month (from 1 to 31)
4. Month (from 1 to 12)
5. Day of week (from 0 to 6) (0=Sunday)
6. Command

In most cases, you may find the asterisk (*) given as the field. If the hour field is given as * and the minute field as 0, it means that the command will run every hour at 0 minutes, i.e., * means every minute, or hour, or day of month, or month, or day of week.

Take a look at the following sample *cron* job entry:

```
00 02 * * * /usr/bin/ktorrent
```

What does the entry mean?

- 00 – 0th minute
- 02 – 2am
- * – any day of month
- * – any month
- * – any weekday

The above *cron* job can be translated to structured

English as follows:

Execute */usr/bin/ktorrent* at the 0th minute of 2 a.m., on any day of the month, any month, any day of the week.

The *cron* syntax allows you to specify the parameter for each field with hyphens to specify the range, i.e., for minute files you can specify 0-10. It also permits you to use comma separators to specify multiple parameters for the same field—i.e., 0,3,6 0-5 * * * command is a valid *cron* job. The syntax also allows you to add comments in the *crontab* entry.

```
# Open gedit at 10 am
00 10 * * * gedit /home/slynux/sample.txt
```

I'm sure you now have some idea on how to write *cron* jobs. Hope you make the best of the unlimited bandwidth that's only made available at certain times of the day.

More *crontab* recipes

To start downloads at 2 a.m. and stop them at 8 a.m., execute the following:

00 02 * * * cd /home/slynux/distros/; wget -c http://example.com/ubuntu.iso
00 08 * * * killall wget -s 9

To execute *diskusage.sh* every 30th minute, repeatedly, use the following command:

```
00/30 * * * * /usr/bin/diskusage.sh
```

To update the *locate* command search database every Sunday between 8 a.m. and 8 p.m., add the *crontab* as the root user, since *updatedb* needs higher write privileges:

```
* 8-20 * * 0 updatedb
```

To schedule system updates at 12 a.m. (here, too, set *crontab* as the root for obvious reasons), use the following command:

```
00 0 * * * apt-get dist-upgrade -y
```

…if you use a Debian-based distro.

Or:

```
00 0 * * * yum upgrade -y
```

…if you have a Fedora-based one.

To shut down your machine at 10 p.m. (install *crontab* as the root user):

```
00 22 * * * halt
```

To remove *crontab* for a specific user:

```
# crontab -u username -r
```

Logging outputs from commands that you run

You can record the progress of the commands run by redirecting the standard output to some log file. It will be very useful to trace or debug something unusual that interrupted the *cron* job. Write your *cron* job as follows:

```
00 00 * * * command >> /var/log/cronjob.log
# cat /var/log/cronjob.log  # For viewing log file.
```

cron is a utility that gives you an awesome user experience. There are also GUI implementations for it. But it is always fun to do everything on the command line, as it powers you to unleash the ultimate potential of the GNU/Linux system. There is also a utility called *at* for temporary job scheduling. Have a look at *man at*. That's it for now. Have fun with *crontab,* and happy hacking! END

By: Sarath Lakshman *is an 18 year old hacker and free software enthusiast from Kerala. He loves working on the awesome GNU/Linux environment and he contributes to the PiTiVi video editor project. He is also the developer of SLYNUX, a distro for newbies. He is currently studying at Model Engineering College, Cochin. He blogs at www.sarathlakshman.info*