Linux Lessons: Tips and Tricks from Users

“It's On with Cron”

by Pete Choppin

Cron is a program that enables you to execute a command, or a script with a sequence of commands, at a specified date, time or at set intervals. The commands or scripts that you want cron to run are defined in a file called crontab, and every user has their own independent crontab file. Cron is a system program that is running all the time and is similar to the Windows scheduler (although it is much more powerful).

There are many different uses for cron, from checking your broadband connection overnight to maintain a stream or hefty download, or simply checking up to see what's going on in your digitized social space. For example, you may want to check the output of the command 'w' every hour to see who's on.

This tutorial will explain how to use cron and crontab and contains some basic working examples that should hopefully illustrate how it functions.

Breaking Down Cron

There is a special format for entering crontabs:

Minute Hour Day Month Day Task

Minute = Minute of the hour, 00 to 59. * Will indicate every minute (details later)
Hour = Hour of the day in 24-hour format, 00 to 23. * Will indicate every hour (details later)
Day = Day of the month, 1 to 31. * Will indicate every day (details later)
Month = Month of the year, 1 to 12. * Will indicate every month (details later)
Day = Day of the week, 3 chars - sun, mon, tue, or numeric (0=sun, 1=mon etc).... * Will indicate every day (details later)
Task = The command you want to execute

Note: Each of the above must be separated by at least one space.

The common way to use crontab is via the crontab command.

crontab –e

This command 'edits' the crontab. When you use this command, you will be able to enter the commands that you wish to run. My version of Linux uses the text editor VI to open the crontab file and edit the contents. (See last week's Linux Lessons for an explanation of text editors.)

Here is an example of a crontab command:

0 1 24 5 0 w

That will run a command at 1:00AM on Monday, May 24th. Now this gets a bit cryptic, so to make it a little better, this would also work:

0 1 24 may mon w

But what if you want it to run every hour, regardless of date? An "*" means that that field doesn't matter, or do the command no matter what is in those fields. So to run our 'w' command every hour, the command would be this:

0 * * * * w

Which means that it runs every day, every hour at the 0 minute mark, meaning the beginning of the hour.

A bunch of different variations of fields can be generated like this. For instance, say you wanted a command to run every two hours. You could specify the "hour" field as this: */2
Which would run at 2,4,6,8, etc. You can also use commas to specify more than one time. For instance, say you want to run it at half past the hour, and a quarter of. You could specify the minute field as this: 30,45

If you use a dash between two values, it will include everything in between them. An example of this would be to run a command every day for the first week of a month. The day of the month field would be this: 1-7
So to have the command run every two hours, at half past and a quarter of, and run for the first seven days of a month. We would have this:

30,45 */2 1-7 * * w >/dev/null 2>&1

Alternative to the Command Line

So you still don't like using the command line? There is an alternative—the Gnome Schedule (KDE has a similar utility called Kron). This is a graphic utility that simply edits the crontab for you.

Simply set up your schedule and enter the command or script you want to run and click Add. Your task is then added to the crontab.

Stop Bugging Me

You may notice that we added the ">/dev/null 2>&1" string at the end of the command above. The default cron job will always send an e-mail to the root account whenever a command is executed. This can get very annoying. If you don't want to receive e-mails every day notifying you about your job's execution, place this at the end of every instance of every crontab

command. It sends the notification into a null holding area—essentially, the message goes nowhere.

Back It Up

Many administrators, and even some users, have a crontab that has multiple schedules, all relying on one crontab file. It is a good idea to create a backup copy of this file in case you lose it or overwrite something you did not intend. Here is a very simple way to create a backup of your crontab:

```
# crontab –l > crontab.txt
```

This command creates a list of the scheduled tasks in the crontab and then appends this to a file called crontab.txt. If you need to retrieve your crontab you simply open the crontab.txt file.

There are literally tons of things that can be done with cron. Hopefully this gives you a brief overview of some of the possibilities. You can combine complicated scripts into a single cron job and set up multiple schedules to run them, or simply have one task that you want to automate. It just depends on what your needs are and how automated you want to get. The possibilities are endless.