COMPUTER POWER USER

Tips & Tricks

Click To Print

Tips & Tricks
June 2008 • Vol.8 Issue 6

Warm Up To Penguins

Cron: Your Personal Linux Wake-Up Service

As Woody Allen (or perhaps Albert Einstein, depending on your source) once said, “Time is what keeps everything from happening at once.” And on a computer, it’s a very good thing that everything happens at the right time. For example, there are tasks that you might wish to occur once a day (rolling over old log files, for example) or once a week (defragmenting a hard drive). Linux provides two different mechanisms to specify that a given action should be taken at a given time, at and cron.

Before we go any further, you have to understand some basics about Linux program input and output, so we’re going to take a short detour. If you’re already familiar with Linux “pipes,” you can skip ahead a few paragraphs. All programs that run on Linux have access to three data streams: stdin (standard input), stdout (standard output), and stderr. You use these all the time when you’re using Linux; you just don’t realize it. For example, we can use the sort command to sort the lines we type into standard input:

```
james@james-ubuntu-test:~$
sort
alpha
gamma
beta
zeta
delta
^D
alpha
beta
delta
gamma
zeta
james@james-ubuntu-test:~$
```

We type in the unsorted list and press CTRL-D. Any program reading from stdin treats a CTRL-D as “end of file” and will stop reading at that point. (CTRL-D doesn’t actually echo to the terminal, but we’ve included it so you can see where the input stops and the output starts.) The sort command writes the sorted list out to stdout. Suppose we want to make sure that there are no duplicates in the list? We can use another program, uniq, which removes duplicate lines from stdin and output to stdout, to take them out. We can use a special character, “|” (pipe) to take the stdout of the first command and “pipe it” to the stdin of the second, as in:

```
james@james-ubuntu-test:~$ sort |
uniq
delta
gamma
alpha
delta
omega
gamma
```

^D
alpha
delta
gamma
omega

The sort command reads data from stdin (the terminal) and sends the sorted version out on stdout. The uniq command reads from stdin, which is connected via the pipe to the output of sort. It sends its results out on stdout, which in that case will be the terminal. Now that we understand a little about pipes, we can move on to at and cron.

Let's suppose you want to remind all the logged-in users on your system in 30 minutes to go outside for the big pool party. As it happens, Linux comes with a handy command called wall that will write a message that you type into stdin onto the terminal of anyone who is logged in. For example, we'd type the following:

james@james-ubuntu-test:~$ **wall**
I'm going to be having a pool party in 30 minutes.
^D

This is great for sending a message immediately, but we'd really like to send it in 30 minutes telling them that the pool party is *now*. We could wait 30 minutes and use wall, but we want to be outside setting up the tropical fruit drink stations, not poised in front of the computer waiting to send out a message. Luckily, we can use the at command to schedule our message in the future!

james@james-ubuntu-test:~$ **at now + 30 minutes**
warning: commands will be executed using /bin/sh
at> **echo "Big pool party outside!!! Be there or be square!!!" | wall**
at> **^D**<EOT>
job 4 at Sat Mar 15 22:36:00 2008

You can choose which editor crontab uses to edit your cron file.

Let's break down what we did. We started by running the at command, specifying we wanted the commands we specified to run at now plus 30 minutes. The at command prompted us for one or more line of commands to run at that time. We told it to use the echo command (which just echoes its arguments to stdout) to send my pool party announcement to the standard input of the wall command, which will send it to all the users. We needed to put the message in double quotes ("") because it contains exclamation marks, which have special meaning to /bin/sh. Putting them in double quotes hides them from the shell.

So now we know how to schedule a job to run at a future date. The at command is actually pretty smart: It can handle times such as "at 4 p.m. tomorrow" or "at 1 a.m. Wednesday." You can also cancel queued at jobs. Suppose imposing clouds start forming as we're lighting the Tiki torches in preparation for our pool party. To postpone the event and save our guests from the less desirable form of hurricane, we can recall the announcement by issuing the following command:

james@james-ubuntu-test:~$ **atq**
Sat Mar 15 22:36:00 2008 a james
james@james-ubuntu-test:~$ **atrm 4**

We used the atq command to list the at jobs we have queued and saw that the troublesome message is job 4. Then, we used atrm to remove the job, saving ourselves (and our guests) from a soggy fate.

Welcome To Cron

Sometimes you want something to happen periodically, rather than just once. For example, you might want to restart your Web server every Sunday at 2 a.m. to deal with a memory leak that eventually makes the server crash. You obviously don't want to have to be sitting in front of your terminal at 2 a.m. Saturday. So you use cron.

Every user on the system is allowed to have a personal "crontab." A crontab is a text file with a specific format that tells the OS to run certain commands at certain times. The root user has one, as well, and you can look at it by saying:

james@james-ubuntu-test:~$ **sudo crontab -u root –l**

```
# m h dom mon dow command
55 22 * * * /etc/init.d/apache2 restart
```

Breaking this down, the “-u root” means, “Give me the crontab for the root user” and “-l” means, “list it.” We have to use sudo because you can’t edit someone else’s crontab unless you have root privileges. As you can see, there’s a single entry in the crontab, telling the Apache httpd daemon to restart. To know when the command will be run, you need to understand the first five columns of the line. The first column is the minute of the hour, the second is the hour of the day, the third is the day of the month, the fourth is the month of the year, and the fifth is the day of the week. The day of month and month of year start properly at 1, not 0 as some programs do. The day of week starts with 0 for Sunday, but you can also use 7 for Sunday. In place of day of week or month, you can also use the first three letters of the name (e.g., sun, tue, jan, dec).

To edit a crontab, use “-e” instead of “-l”. It will bring up the file (or create a new one) using whatever your default editor is. If you haven’t changed anything, usually this is nano. If you want to use vim instead, just type **export EDITOR=vim** before you type in the crontab command, like this:

```
james@james-ubuntu-test:~$ export EDITOR=vim
james@james-ubuntu-test:~$ crontab –e
no crontab for james - using an empty one
```

You can separate the columns by spaces or tabs or both, as many as you want. Some people like to use tabs to get everything in nice columns. The commands will run as the user who owns the crontab, and any output generated to stdout by the command is bundled up as an email to the user.

There are many more options to crontab, such as using ranges for numbers or telling it not to send emails. To learn them, type **man 5 crontab** into a terminal window. That will load the manual page for the crontab file format. Typing **man crontab** retrieves info on the crontab command. ■

by James Turner

Infinite Loop: The DIY Tank

Gone are the days of hazing pledges or TPing the Quad when you have nothing better to do. For Kettering University student Will Foster, the best way to pass the time was building a half-scale working replica of a Panzer tank. The creation, which cost an estimated $10,000 to build, runs on treads and has a turreted cannon that rotates 360 degrees and is capable of firing golf balls or empty Red Bull cans. Put that in your pipe and smoke it, Blutarsky.

Source:
www.mlive.com/news/index.ssf/2008/04/post_moto_kid_death_story_here.html