Hentzenwerke Whitepaper Series

Cron Explained

By Whil Hentzen

You can't get far in the Linux world without hearing the words, "Oh, just set up a cronjob to do that for you!" Cron? Cronjob? Sounds uncomfortable, maybe even a bit nasty. This whitepaper describes what cron is and how to use it. It describes what cron, cronjobs and crontabs are, how cron works, how to edit crontabs and how to set up cronjobs, and how to work with cron output via email and log files.

Hentzenwerke Publishing, Inc. • books@hentzenwerke.com • www.hentzenwerke.com

1. Preface

1.1 Copyright

1.2 Revisions

1.2.1 History

Version	Date	Synopsis	Author
1.0.0	2004/6/17	Original	WH
1.0.1	2004/7/7	Corrected grammar and spelling errors.	WH

1.2.2 New version
The newest version of this document will be found at www.hentzenwerke.com.

1.2.3 Feedback and corrections
If you have questions, comments, or corrections about this document, please feel free to email me at 'books@hentzenwerke.com'. I also welcome suggestions for passages you find unclear.

1.3 References and acknowledgements
Thanks to MLUG members Aaron Schrab, Daniel J. Cody, Jerry Davis, Joe Frost, Jonathan C. Detert, Joshua Cowles, Mark Pinkerton, and Thomas Landmann and to Brad Illston's introduction to cron at http://weather.ou.edu/~billston/crontab/.

1.4 Disclaimer

1.5 Prerequisites
This document was written using Fedora Core 1.0 and assumes a beginner's familiarity with use of Linux via the GUI and the Command Window.

2. Cron definition - what is it and what does it do?
Cron is a daemon (Windows folks, think 'Windows Service') that runs continuously in the background on a Linux computer. Its job is to automatically run tasks according to a schedule that's also on the computer. The schedule is contained in an ASCII text file called a crontab ('cron table'). Each user on a computer has their own crontab. There are three elements to getting cron to work: cron has to be installed, cron has to be running, and the crontab itself has to exist.

3. How cron works - the pieces and the plumbing
Cron is a service that starts up automatically during boot up. It is started from the scripts located in the /etc/rc (and thereabouts) directory structure. On my Fedora Core system, the cron daemon (crond) is located in /etc/init.d, and links to it are located in the /etc/rc.d/rcN directory structure. For example, the link for runlevel 5 is

```
/etc/rc.d/rc5.d/S90crond
```

When loaded, cron looks in the /var/spool/cron directory for crontab files that map to accounts in /etc/passwd. Crontabs that are found are loaded into memory and kept there. (Cron also searches a number of other locations that are described in Section 11.)

During system operation, cron examines all stored crontabs once a minute, looking for commands it should execute during that specific minute. Upon successful or unsuccessful completion of a command, cron creates output that is by default emailed to the owner of the crontab. More on how this works and how to change it is covered in Section 9.

During this minute-by-minute check, cron also checks to see if any of the crontabs have been modified (by examining the modtime of the cron's spool directory, which gets changed by the crontab program whenever a crontab is changed), and reloads them if that is the case. As a result, you don't have to stop and restart cron if you change a crontab.

If the computer is down or cron is stopped during a period that there is a task scheduled, that task will not be run once the computer is turned on or cron is restarted. This has implications when your system clock changes, such as daylight savings time. If you have a cronjob scheduled during the hour that is skipped in the spring when daylight savings time causes clocks to be moved from 1 am to 2 am, that job will not run at all. Conversely, if you have a job scheduled during the hour in the fall when clocks are set back, that job will be run twice.

In other words, cronjobs do not get placed in a queue that, if the computer is down for a period of time, can be rescanned in order to run jobs that were skipped during the downtime. Instead, cron simply fires off tasks in realtime.

4. Examples of what cron can do

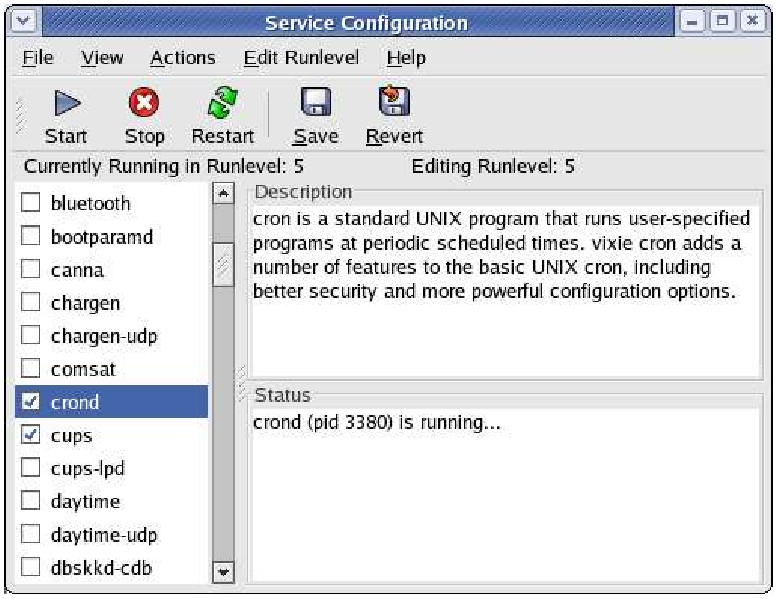
Cron can automate any type of task that you would otherwise do manually. For example, a few simple examples include:

1. Cleaning out temp files
2. Deactivating user accounts that have expired
3. Automatically sending mail as a reminder, such as for a monthly meeting or a weekly status report
4. Grabbing data and merging it into a bigger file

5. The cron daemon - checking if it is installed and running

Before we get into using cron, it would be a good idea to make sure that it's installed and running. After all, it's kind of aggravating when the rest of article doesn't work because for some reason cron isn't installed or running, eh?

On FC1, select the System Settings | Server Settings | Services menu to bring forward the Services dialog as shown in **Figure 1**.



Currently Running in Runlevel: 5 Editing Runlevel: 5

- [ ] bluetooth
- [ ] bootparamd
- [ ] canna
- [ ] chargen
- [ ] chargen-udp
- [ ] comsat
- [✓] crond
- [✓] cups
- [ ] cups-lpd
- [ ] daytime
- [ ] daytime-udp
- [ ] dbskkd-cdb

Description

cron is a standard UNIX program that runs user-specified programs at periodic scheduled times. vixie cron adds a number of features to the basic UNIX cron, including better security and more powerful configuration options.

Status

crond (pid 3380) is running...

Figure 1*. Make sure that cron is installed and running before trying to use it!*

You can check that cron is running in the command window with a command like

```
ps -awx | grep crond
```

If cron is running, you'll see a line in the response that looks something like this:

```
[whil@freedom ~] #ps -awx | grep crond
 3410 ?        S      0:00 crond
 5002 pts/1    S      0:00 grep crond
[whil@freedom ~] #
```

6. All about the crontab file

Now let's talk about the crontab files themselves. Where are they and what's in them?

6.1 Location

Each user, including root, has their own crontab. Where this file is stored depends on the distribution. On FC1, my crontab is stored in /var/spool/cron - the file itself is

```
/var/spool/cron/whil.
```

6.2 Contents - cronjobs

As mentioned, a crontab is simply a text file with jobs listed on separate lines.

A cronjob is a line that contains six fields, separated by spaces. The values in the first five fields determine when the task will be run and the sixth field defines the task itself. The first five fields correspond to minutes, hours, days, months, and weekdays.

The range of values allowed in the various fields are shown in **Table 1**.

***Table 1**. Crontab time period values*

Time Period	Values
Minutes	0 – 59
Hour	0 – 23 (0 = midnight)
Day	1 – 31
Month	1 – 12
Weekday	0 – 6 (0 = Sunday)

You can also put an asterisk in a field to indicate that every possible value for that field (such as every minute, every day of the week, etc.) will be used. For example,

```
30 1 * * * task
```

would run 'task' every day at 1:30 in the morning.

There are a number of permutations of what can go in the fields to fine tune when a cronjob is run.

If you want to use more than one instance of a value, separate the values by commas. For example, if you wanted a job to run at 15 and 45 minutes past the hour of every day, you'd use 15,45 in the first field, like so:

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

Similarly,

```
30 22 1,11,21 * * task
```

would run 'task' at 10:30 pm on the 1st, 11th and 21st of every month.

You can use a range of values to span a number of values without typing each of them in. For example,

```
15 5 1,15 * * 'task'
```

would run a job at 5:15 am on the first 15 days of the month.

You can combine multiple instances and ranges together. For example, if you wanted to run a job every minute for the first quarter of the hour, and then every minute during the third quarter of the hour, you would use 1-15, 31-45 in the Minutes field, like so:

```
1-15,31-45 * * * * task
```

You can also follow a value with a /n, where 'n' is a Step Value, in order to run a job each increment of that value. For example, if you wanted to run a job every 10 minutes, you'd use */10 in the Minutes field, like so:

```
*/10 * * * * task
```

And you can combine a Range with a Step Value to run a job on the increment of the step value, but only during the range. To run a job every 3rd minute during the first half of an hour, use 1-30/3 in the Minutes field, like so:

```
1-30/3 * * * * task
```

The job would run on minutes 1,4,7,10,13,16,19,22,25,28.

You may note that it seems that Day of Week and Day of Month could seem to be set up to be mutually exclusive. For example, suppose Day of Month is 1-10 (so the job would only run on the first ten days of the month), but Day of Week is 1 (so the job would run every Monday.) It would seem that "just the first ten days" and "every Monday" are mutually exclusive conditions. Actually, they are mutually exclusive.

In situations like these, the task is executed when either condition is satisfied. In this specific example, the task would be run on the first ten days of the month, and then every Monday after the 10th. For example,

```
59 0 1-7 * 0 task
```

would run 'task' at 12:59 am on the first seven days of the month, and then on every subsequent Sunday after the 7th.

6.3 Advanced contents - environment settings

So far you've seen one type of line in a crontab. There are three other types of lines - blank lines, comment lines (anything preceded by a "#"), and lines that set environment settings.

Cron inherits only a few selected settings in the current user's environment, namely HOME, LOGNAME and SHELL. (See cron's man page for details.) If you want a particular user's cronjobs to know other environment settings, you'll have to set them in that user's crontab.

One very useful setting is the MAILTO environment variable. By default, cron sends a mail message to the local system account of the owner of the crontab. If you wanted cron's email output to be sent to a mail account other than this one, you can set the MAILTO environment variable to that other mail account. For example, if you wanted to send mail to the 'admin' user on this machine, you would place the following line

```
MAILTO=admin
```

in the crontab file. If you wanted to send mail to your 'joebob' account at the 'yourcompany.com' domain, you would place the following line

```
MAILTO=joebob@yourcompany.com
```

in the crontab file.

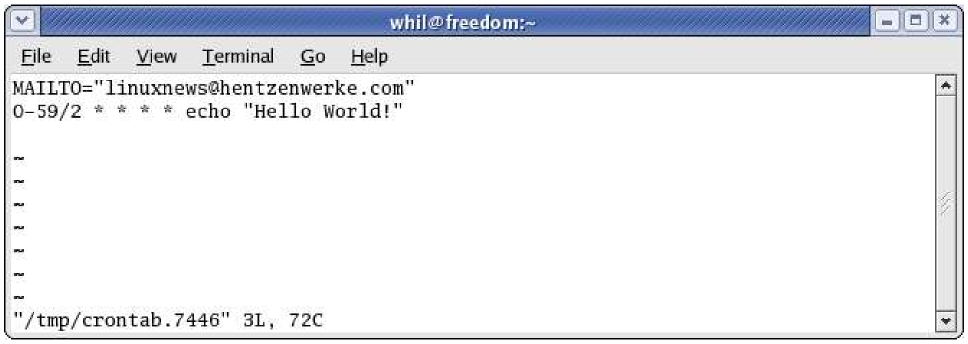
7. Crontab editing requirements and issues

So now you have an idea of what goes in a crontab. How do you actually GET that info into the file? You can't just open up the crontab file in /var/spool/ with your favorite editor and blast away at the contents. Rather, there are two specific ways to work with the crontab. The first is to use the crontab command to access the system's default editor to create or edit the crontab file. The second is to use your editor of choice to create or edit a temporary crontab file, and then use a variation of the crontab command to update the master crontab with your temporary crontab.

The rest of this section covers these various choices. Section 8 goes into actually creating, editing and deleting a crontab.

7.1 A 30 second tutorial on using vi

Many systems are configured with vi as the default editor, as shown in **Figure 2**. Vi isn't at all intuitive to use, so here's a 30 second vi tutorial that will show you how to edit and save a file.



```
MAILTO="linuxnews@hentzenwerke.com"
0-59/2 * * * * echo "Hello World!"

~
~
~
~
~
~
~
"/tmp/crontab.7446" 3L, 72C
```

***Figure 2**. Editing a crontab file in vi.*

Vi has two modes, insert mode (when you're typing into the file) and command mode (when you're issuing commands to vi to do something such as save or quit.) When you start up vi, you're in command mode. In order to issue a command, type ":", the letter of the command, and then Enter. For example,

1. To quit vi: type :q and press Enter.
2. To save the current file: type :w and press Enter.
3. To switch from command to insert mode: type any of the following letters: a i o c s, in either upper or lowercase.
4. To switch from insert to command mode: press Escape.

7.2 Changing the system editor to be your favorite editor

What if you don't like the default system editor? Or what if you don't even know what it is? You can type "SET" at a command prompt to determine what your environment variables are. You might see variables named "EDITOR" and "VISUAL" in the list. If you don't, the system editor is probably vi.

In order to change the system editor to something you prefer, add a line like:

```
EDITOR=/usr/bin/<name of editor>
```

to your ~/.bashrc file. For example, on my Fedora Core system, I used

```
EDITOR=/bin/kedit
```

to switch the system editor to kedit.

7.3 Using your favorite editor without changing the system editor

If you're bound and determined to use your own favorite editor but don't want to mess with changing the default editor (say you're working on someone else's system), you can do that too.

First, use your own favorite editor to create the crontab file, and place it somewhere innocuous, like your home directory, resulting in a file like

```
/home/{username}/crontab
```

Then issue the crontab command, followed by the name of the file you just created, like so:

```
crontab /home/{username}/crontab
```

Crontab will create (or overwrite) the official crontab for you.

As you get more sophisticated, you'll may find that even this isn't enough. For example, one thing some folks do is endeavor to keep their customizations all in one place that can be backed up easily. Since the crontab file is located in /var/spool, a directory that isn't usually backed up, it's possible for you to lose your custom crontab file upon system reinstall.

A way around this is to keep your crontab file in your home directory, say, in ~/.crontab, and then point cron to your own file, like so:

```
crontab /home/{username}/.crontab
```

8. Crontab file maintenance

Now that you're ready to use your editor of choice, it's time to actually work with your crontab.

8.1 Create a crontab

Open a command window and type

```
crontab -e
```

If there isn't already a crontab file, you'll get a blank screen, ready for you to enter your cronjob.

Then enter the line of six fields - the five values for the schedule and the name of the program or command that you want to run. While you might be able to get away with just using the name of the program itself, it's a good idea to enter the entire path, so that there's no confusion.

For example, if you were going to remove the temp files from your own home directory, you could

```
30 1 * * * rm /home/{username}/temp/*
```

but it would be safer to use

```
30 1 * * * /bin/rm /home/{username}/temp/*
```

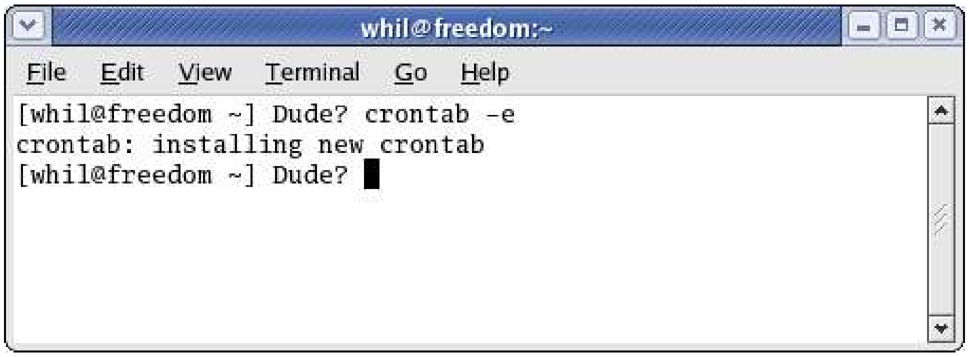
Note that in any case, you can't use relative pathing. You must use 'absolute paths' when referring to the entry to be executed. You can't refer to

```
~/myfile.txt
```

Rather, use

```
/home/{username}/myfile.txt
```

Note that in vi, you'll save the file with :w, then exit vi with :q. Once you do so, you should see a confirmation that your crontab file has been successfully updated, as shown in **Figure 3**.



```
[whil@freedom ~] Dude? crontab -e
crontab: installing new crontab
[whil@freedom ~] Dude?
```

Figure 3. *Crontab will echo a successful update in the command window.*

If you just close your command window after :w, crontab won't successfully update the master crontab file.

8.2 Examine an existing crontab

If you just want to see what's in a crontab without editing it (and possibly messing it up), use the -l (the letter 'ell') parameter:

```
crontab -l
```

8.3 Edit an existing crontab

To edit an existing crontab, use the same command that you used when creating it.

```
crontab -e
```

Since the crontab is always in the same location, you don't have to worry about creating a second crontab in a second place.

8.3.1 Add a new job

To add a new job, simply put the job on a new line.

8.3.2 Remove an existing job

To remove a job, delete the line containing the job. You can close up the line or leave it blank, but don't put another string in it's place (unless it's a comment - see next section).

8.3.3 Temporarily remove an existing job

To temporarily stop a job from executing, comment it out by putting a # in front of the line, like so:

```
#30 1 * * * task
```

8.4 Delete a crontab

To delete the crontab file completely, use the command

```
crontab -r
```

This deletes the file. Cron will still run, but it won't find anything to do, so it will be bored.

8.5 Editing the systemwide crontab

When you want to edit the systemwide crontab, simply login as root user and issue the command 'crontab -e'. The '-e' switch tells crontab to edit the default systemwide file.

8.6 Editing a user's crontab as admin

You can also use the '-u username' switch to edit specific users crontab files, which is useful if you're a system administrator trying to track down errant processes that users are executing via the cron daemon.

9. Working with cron results

So how do you know if your cron job ran or not? You have several choices.

Cron doesn't ordinarily generate output by itself; if you have, say, a backup cronjob, cron will take care of its business without ever peeping. So you aren't automatically notified when a cronjob runs (or fails to run.) The exception is when a task generates output itself. For example, if your cronjob was simply an 'echo' command (uninteresting but useful for discussion's sake), the cronjob would generate output.

9.1 Output via mail

Output from a cronjob is ordinarily sent as an email to the owner of the cronjob.

Naturally, for cron to send email, there will have to be an email program on your machine. Most Linux machines have one installed and running. Fedora Core machines typically have sendmail installed while SuSE machines often have postfix installed. The default mail program for many Linux distributions is sendmail; you can tell if your machine has it (or another mail program) installed and running through the Services applet, as shown in Figure 1 earlier.

If you want to be notified whether a job runs or not, regardless of whether the cronjob itself generates output, you can include the mail command in the task to send the cronjob owner (or someone else) email. For example,

```
30 1 * * * task | mail -s"cronjob 'task' just ran" whil
```

will send an email message with the subject line of

```
cronjob 'task' just ran
```

to the 'whil' account on the machine.

Note also that if the mail command doesn't run properly, the cronjob now generates output (an error message) that will be sent to the owner of the cronjob.

9.1.1 How cron's default mailing behavior works

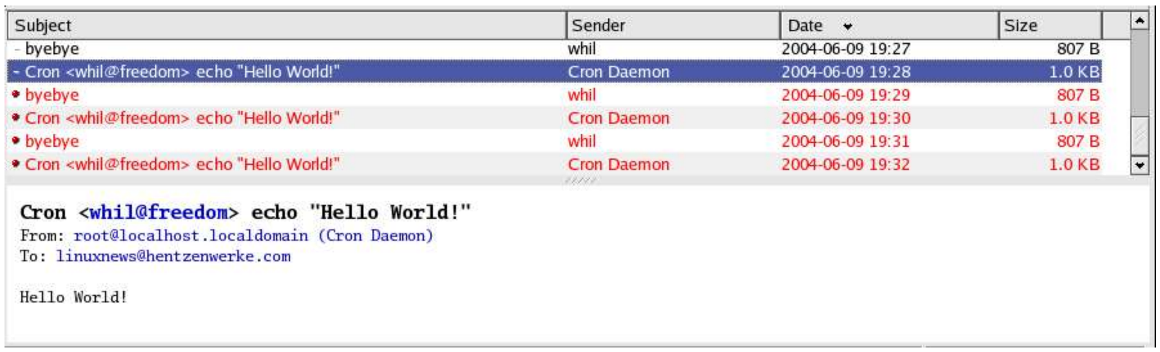
Cron will use the mail program to send a message to the email account for that program for the user for whom the cron job is running. Any output is mailed to the owner of the crontab (or to the user or email address named in the MAILTO environment variable in the crontab, if there is such a line, as described in section 4.3).

Once you have set up a cron job and it has run at least once, you can check the mail file to see if it has gotten larger and/or its timestamp has changed. The mail file for a user is typically something like /var/mail/{username}.

For example, a cronjob task of

```
  echo "Hello World!"
```

will result in an email containing the outut of the echo command to the user of the crontab, from "Cron Daemon". Such a cronjob, incorporated with a MAILTO=linuxnews@hentzenwerke.com environment line in the crontab, results in the email that is shown in the KMail client in **Figure 4**.



Subject	Sender	Date	Size
byebye	whil	2004-06-09 19:27	807 B
Cron <whil@freedom> echo "Hello World!"	Cron Daemon	2004-06-09 19:28	1.0 KB
byebye	whil	2004-06-09 19:29	807 B
Cron <whil@freedom> echo "Hello World!"	Cron Daemon	2004-06-09 19:30	1.0 KB
byebye	whil	2004-06-09 19:31	807 B
Cron <whil@freedom> echo "Hello World!"	Cron Daemon	2004-06-09 19:32	1.0 KB

```
Cron <whil@freedom> echo "Hello World!"
From: root@localhost.localdomain (Cron Daemon)
To: linuxnews@hentzenwerke.com

Hello World!
```

***Figure 4**. A sample mail message from the Cron Daemon to the cronjob's owner.*

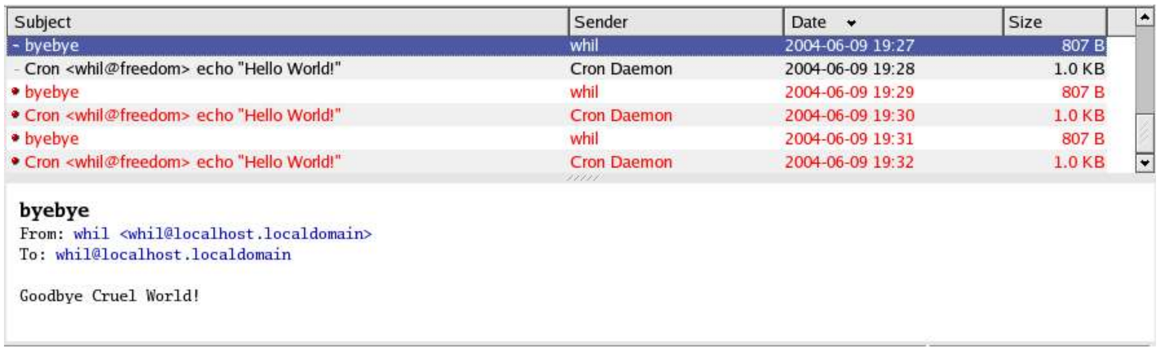
Note that cron determines the subject line of the email, and identifies itself as the sender ("Cron Daemon").

9.1.2 Changing cron's default mailing behavior

As mentioned, you can use the mail program, and identify the user as well, like so:

```
  echo "Goodbye Cruel World!" | mail -s"byebye" whil
```

The result, as received by the KMail client, is shown in **Figure 5**.



Subject	Sender	Date	Size
- byebye	whil	2004-06-09 19:27	807 B
Cron <whil@freedom> echo "Hello World!"	Cron Daemon	2004-06-09 19:28	1.0 KB
byebye	whil	2004-06-09 19:29	807 B
Cron <whil@freedom> echo "Hello World!"	Cron Daemon	2004-06-09 19:30	1.0 KB
byebye	whil	2004-06-09 19:31	807 B
Cron <whil@freedom> echo "Hello World!"	Cron Daemon	2004-06-09 19:32	1.0 KB

```
byebye
From: whil <whil@localhost.localdomain>
To: whil@localhost.localdomain

Goodbye Cruel World!
```

Figure 5. *Using the mail program to email a user after a cronjob runs.*

In order to see the difference in these two approaches, the following crontab alternates a regular cronjob that emails the owner of the cronjob with a cronjob that pipes the output to the local user, 'whil'. (I created a .forward file in /home/whil that contains 'linuxnews@hentzenwerke.com' so that I could read the piped email in the KMail email client that reads the linuxnews email account.)

```
  MAILTO="linuxnews@hentzenwerke.com"
  0-59/2 * * * * echo "Hello World!"
  1-59/2 * * * * echo "Goodbye Cruel World!" | mail -s"byebye" whil
```

This crontab uses a combination of range values and the /2 step value for both cronjobs in order to accomplish the alternation. The first cronjob will execute on minutes 00, 02, 04, 06, and so on each hour, while the second cronjob will execute on minutes 01, 03, 05, 07, and so on.

9.1.3 Suppressing mail

You may not want to get mail on routine cronjobs, as they could fill up your mailbox pretty quickly. (Consider setting up a cronjob that emails a big file to a soon-to-be-former friend every minute, for example.)

You can suppress the sending of mail by including the following string at the end of a crontab line:

```
>/dev/null 2>&1
```

like so

```
30 * * * * echo "Hello World" >/dev/null 2>&1
```

Finally, you can have cron write the results to a log file instead of using mail, or even have cron do both.

9.2 Directing output to a log file

If you wish to disable the email (and output to a log file) then, at the end of each of the cron job lines you wish to not be notified on, place the string:

```
  > {logfile path and name}
```

or

```
  >> {logfile path and name}
```

at the end of the job. One > means overwrite the log file while two (>>) means to append the current output to the end of the existing log file.

Note that these are additive with the string that suppresses the mail. In other words, you can configure cron to just send the output to mail, or just to a log file, or to both (sort of a belt and suspenders approach).

This command sends the output to a txt file:

```
30 * * * * echo "Hello!" > /home/{username}/cronlogs/cronlogfile.txt
```

This sends the output to a txt file and suppresses the mail as well:

```
30 * * * * echo "Hello!" > /home/{username}/cronlogs/cronlogfile.txt >/dev/null 2>&1
```

10. Tutorial

Now that you've got all this 'book learning' out of the way, let's do one. We're going to create a cron job that deletes files from the /home/{username}/temp directory every five minutes every day.

First, create a dummy file in /home/{username}/temp.

```
cd ~
touch temp/yo.txt
```

Remove it manually just to make sure you've got the command correct (and then put it back again).

```
rm /home/{username}/temp/*
touch temp/yo.txt
```

Now edit your crontab file.

```
crontab -e
```

Place a cronjob in it.

```
0-59/5 * * * * rm /home/{username}/temp/*
```

Save the crontab file and then look at it to make sure it's set up properly.

```
crontab -l
```

Now we wait. Within five minutes, you should see two things happen. First, the yo.txt file in /home/{username}/temp should be gone. Second, your mail file, /var/mail/{username} should be updated with a new message.

Finally, go back into your crontab and either delete the line altogether, or change the time fields to once a day. For example,

```
30 4 * * *  rm /home/{username}/temp/*
```

will run the job to delete the files every morning at 4:30.

In order to have the mail sent to you at a different mail address, change your crontab to read like so:

```
  MAILTO=youraddress@yourcompany.com
  30 4 * * *  rm /home/{username}/temp/*
```

11. Super-duper advanced stuff

Cron doesn't just run jobs for individual users; it also handles systemwide jobs that are run on hourly, daily, weekly and monthly intervals. The /etc/crontab file contains the specification of how these systemwide cronjobs are run. For example, the default Fedora Core /etc/crontab file looks like this:

```
 SHELL=/bin/bash
 PATH=/sbin:/bin:/usr/sbin:/usr/bin
 MAILTO=root
 HOME=/

 # run-parts
 01 * * * * root run-parts /etc/cron.hourly
 02 4 * * * root run-parts /etc/cron.daily
 22 4 * * 0 root run-parts /etc/cron.weekly
 42 4 1 * * root run-parts /etc/cron.monthly
```

The directories listed in each cronjob contain programs and/or scripts that are to be run on a regular basis. The five datetime values define when the jobs are to be run. The /etc/cron.daily directory, for example, includes (but is not limited to) the following scripts:

```
00-logwatch
logrotate
rpm
locate.cron
```

while the /etc/cron.hourly directory includes a 'diskcheck' script (actually, a Python program.)

There are two more possible directories in this group. /etc/cron.allow contains a file list of users allowed to use crontab while /etc/cron.deny contains a file list of users denied to use crontab. These don't always exist.

You can also put your own files in these directories, and they'll be run along with the rest of the files, according to the date and time specified in /etc/crontab.

12. Where to go for more information

This free whitepaper is published and distributed by Hentzenwerke Publishing, Inc. We have the largest lists of "Moving to Linux", OpenOffice.org, and Visual FoxPro books on the planet.

We also have oodles of free whitepapers on our website and more are being added regularly. Our Preferred Customer mailing list gets bi-monthly announcements of new whitepapers (and gets discounts on our books, first crack at special deals, and other stuff as we think of it.)

Click on "Your Account" at www.hentzenwerke.com to get on our Preferred Customer list.