IT 1100 : Introduction to Operating Systems

Cron Jobs

Cron Jobs

The examples on this page have been taken from the following website:

https://help.ubuntu.com/community/CronHowto”

What is Cron?

Cron is the name of program that enables unix users to execute commands or scripts (groups of commands) automatically at a specified time/date.

Why use Cron?

- Any task that you need to be automated.
- Any task that you need to be run at certain times every day, week, month, or year.
  - i.e. Backups, log file rotation, clean things up, email newsletters

Special cron directories

Look at the following:

- `/etc/cron.hourly`
- `/etc/cron.daily` (we might have some entries in here)
- `/etc/cron.monthly`
- `/etc/cron.weekly`

View the `/etc/crontab` file to see when the above programs are run.

Type `sudo crontab -l` to view the sudo cron jobs. Yes it is an “el” for list.

Type `crontab -l` to view your personal cron jobs

Editing Your Cron Jobs

To edit your local cron jobs and add entries to your personal crontab file:

- `crontab -e`

To edit the system cron jobs (ones that require sudo permissions or are run for all users) and add entries to the system crontab file:

- `sudo crontab -e`

There will be a bunch of explanation text at the top of the file- You can remove it or leave it, but it is recommended that you leave at least the last line `m h dom mon dow command` to help you remember the format for your cron jobs.

When you save and close the crontab file it will let you know if you made any mistakes. The crontab will be installed and begin running if there are no errors.

Writing a Cron Job in the Crontab file

Cron jobs in the crontab file are written in the following format:

m h dom mon dow command

minute (0-59), hour (0-23, 0 = midnight), day (1-31), month (1-12), weekday (0-6, 0 = Sunday), command

An **asterisk (*)** can be used to represent every instance or basically saying - we don't care what the value is.

Comma-seperated (,) values and **dash (-)** ranges can be used represent multiple run dates and times.

It is recommended that you use the **full path** to the desired commands as shown in the following examples. This is a safe-guard to prevent unwanted behavior in case multiple instances or aliases of a command exist or are created at some point.

An **</num>** will allow the cron job to run at regular intervals of time, such as every 5 minutes or every other month. This format can be used on any of the values: minute, hour, day, month, and weekday

---

Examples

```
01 04 1 1 1 /usr/bin/somedirectory/somecommand
01 04 1 1 1 /bin/echo "Happy January" >> /home/s/smorgan/jan.txt
01 04 1 1 1 echo "Happy January" >> jan.txt
```

The above example will run /usr/bin/somedirectory/somecommand at 4:01am on January 1st plus every Monday in January.

```
01 04 * * * /usr/bin/somedirectory/somecommand
01 04 * * * /usr/bin/updatedb
01 04 * * * updatedb
```

The above example will run /usr/bin/somedirectory/somecommand at 4:01am on every day of every month.

```
01,31 04,05 1-15 1,6 * /usr/bin/somedirectory/somecommand
01,31 04,05 1-15 1,6 * /bin/tar -cf /backups/home-backup.tar /home
01,31 04,05 1-15 1,6 * tar -cf /backups/home-backup.tar /home
```

The above example will run /usr/bin/somedirectory/somecommand at 01 and 31 past the hours of 4:00am and 5:00am on the 1st through the 15th of every January and June.

```
0,10,20,30,40,50 * * * * /usr/bin/somedirectory/somecommand
0,10,20,30,40,50 * * * * /bin/cp /var/log/syslog /home/s/smorgan/syslog-copy.txt
0,10,20,30,40,50 * * * * cp /var/log/syslog syslog-copy.txt
```

The above example will run every 10 minutes. It is the same as the following command which runs on minutes divisible by 10: 0, 10, 20, 30, etc.

```
*/10 * * * * /usr/bin/somedirectory/somecommand
*/10 * * * * /bin/cp /var/log/syslog /home/s/smorgan/syslog-copy.txt
*/10 * * * * cp /var/log/syslog syslog-copy.txt
```

Nested Commands

To run a command inside of another command we can use $( ). What this does is tells bash to evaluate the command or values inside of the parenthesis before evaluating the main command. Other scripting languages such as php use this same format.

The following will run the date command before it runs the echo command. Thus displaying the date inside of the quote.

- `echo "This is the date $(date)"`

Backticks do the same thing.

- `echo 'This is the date `date`'`

Whichever method you use to nest your commands the output is the same: `This is the date Wed Mar 8 15:06:49 MST 2017`

If we want to we can even redirect this echo to a file.

- `echo 'This is the date `date`' >> date.txt`
- `cat date.txt`

Now that we have a working command we just need to put it in a cronjob. When will the following cron job run? What will it do?

- `00 06 */5 */2 * echo "This is the date $(date)" >> date.txt`

Textbook Time

There is no textbook reading for this section.
