

Lab 7: Cron

In this lab, you will learn how to use the **cron** facility, and setup user **crontab**'s to schedule periodic jobs. Perform the indicated steps, and write the answers to questions in the appropriate spaces on this hand-out. Turn-in one completed handout per team and be sure to include everyone's name

Boot the System

Prepare the system for the current lab.

Step 1. Boot your system into single-user mode.

Step 2. Add the lines below to the end of the file `/etc/sysconfig/network-scripts/ifcfg-eth0`

```
BOOTPROTO=dhcp
ONBOOT=yes
```

Step 3. Bring the system into multi-user mode.

Step 4. Test that the system has internet access by ping'ing some internet host.

Scheduling a Periodic Job

Learn how to use cron and to create a crontab to schedule a job to run periodically..

Step 5. Log in as root

Step 6. Create a crontab entry for root by running the command below. It will bring up a **vi** editor session allowing you to enter one or more crontab entries for the current user.

```
# crontab -e
```

Step 7. Add a crontab entry that will run the **uptime** command every minute for today only, and use redirection to capture the cumulative output into a log file in `/var/log`. The command below can be used as the *command* field in the crontab entry. After entering your crontab entry, save the file and quit the editor session.

```
uptime >> /var/log/uptime.log
```

Step 8. Users can view the contents of their crontab files using the **-l** option of the **crontab** utility. View the contents of **root**'s crontab entry:

```
# crontab -l
```

Step 9. The crontab utility maintains its own copy of a user's crontab file. Examine the contents of the `/var/spool/cron` directory.

Q1. What is the list of files contained in `/var/spool/cron`? _____

Q2. How do the contents of the file **root** compare with the output from Step 8 above? _____

Step 10. Wait a minute or two and examine the contents of the log file where the output of **uptime** is being stored from Step 7 above.

Q3. How many times has the **cron** job run already? _____

Step 11. Replace the crontab entries with a new entry that will shutdown the system automatically at 10:30pm every Monday evening. The command to use is:

```
shutdown -h now
```

Step 12. Logout as the **root** user and login as user **student**.

Step 13. Create a crontab entry for user **student** using **crontab -e** and add a command to mail a list of users currently logged onto the system every 1/2 hour, but only on Monday, Wednesday, and Friday's. The *command* field of the crontab entry is:

```
who | mail student
```

Step 14. Run the command that will list the contents of user **student**'s crontab file.

Q4. What are the contents? _____

Step 15. Create a new file **~/my-crontab** using your favorite editor. Add a crontab entry to remove any files from the **/tmp** directory that are 3 days or older. Have it run every hour. Save the file, and then install the crontab using the command below. Hint: See page 160 of your textbook (page 155 for the green Linux book).

```
$ crontab ~/my-crontab
```

Q5. What was your crontab entry? _____

Step 16. List the contents of **student**'s crontab entry.

Q6. What happened to the entry you installed in Step 13? _____

Step 17. Delete **student**'s crontab entry with:

```
$ crontab -r
```

Q7. What command will validate that **student**'s crontab entry no longer exists? _____

Q8. When a command in a user's crontab runs, what will be the UID of the process? _____

Controlling Usage of crontab
Use cron.deny and cron.allow to control the usage of the crontab utility.

Step 18. As **root**, create the file **/etc/cron.deny** and add the single line **student** to the file.

Step 19. Switch users to **student** and try to edit and schedule a crontab entry using **crontab -e**.

Q9. What happens? _____

Step 20. As **root**, remove the file **/etc/cron.deny** and create the file **/etc/cron.allow**, adding the single line **root** to the file.

Step 21. Become **student** again - try to edit and schedule a crontab entry using **crontab -e**.

Q10. What happens? _____

The cron Log File
Examine and learn about the log file created and maintained by the cron daemon

Step 22. Examine the file **/var/log/cron**. This file is a log of the actions performed by **cron** or **crontab**. Try to figure out and understand each entry. The entries in the file are in reverse order, with the newest entries last.

Q11. Which users are allowed to examine this file? _____

Q12. Which commands have been run by **cron** so far? _____

Q13. Where did **cron** get these crontab entries to run (identify the exact files)? _____

System crontabs

Learn about the system crontab entries

Step 23. Examine the file `/etc/crontab` and the contents of the directory `/etc/cron.d`.

Q14. Describe the contents of these files: _____

Step 24. Examine the entries more carefully this time, comparing each entry with the format specified in the lecture notes.

Q15. What is different in these entries? _____

Q16. Explain what the additional field means. _____

Q17. What is **run-parts**? _____

Q18. What are **cron.weekly**, **cron.daily**, etc. as referred to in `/etc/crontab`? _____

Q19. What programs are run each day? _____

Q20. What are the permissions of the **crontab** program? _____

Q21. What are the permissions of the `/var/spool/cron` directory? _____

The find Command

Learn about and practice using the find utility, a command used frequently for system administrative duties.

Step 25. The **find** utility is used frequently in system administration. Explore the various options available in **find** by reviewing its manual page.

Q22. What option(s) are used to prevent crossing into NFS filesystems? _____

Q23. What option will find empty files? _____

Q24. What option will find only regular files (not links, devices, etc.)? _____

Q25. What option is used to execute a command for each found path? _____

Step 26. Write a **find** command line that will find all **setuid** shell scripts in local filesystems (not NFS filesystems).

Q26. What is the command? _____

Step 27. Hard! See if you can write a **find** command that will remove the files **core** or **core.#** (where **#** is any number of digits) from local filesystems (not NFS filesystems) that have not been accessed in the past 5 days. Be certain your command will not remove the entry `/dev/core`, which is a legitimate device entry. Hint: if you get stumped on this one, ask for help.