# Process Management

## Using the renice ratched effect

This exercise can be done as ‘instructor’. There is no need switch to ‘root’.

Launch the cpuHog1 processes as in the background:

/usr/local/bin/cpuHog1 &

[1] 41503

As instructor, renice this to ….

Note : pgrep and pkill practice

[instructor@ml-refvm-422293 ~]$ /usr/local/bin/cpuHog1 &

[1] 11750

[instructor@ml-refvm-422293 ~]$ pgrep cpuHog1

11750

[instructor@ml-refvm-422293 ~]$ pkill cpuHog1

[instructor@ml-refvm-422293 ~]$ pgrep cpuHog1

[1]+ Terminated /usr/local/bin/cpuHog1

## Balancing CPU Usage

This exercise can be done as ‘instructor’. There is no need switch to ‘root’.

Launch four processes as follows:

/usr/local/bin/cpuHog1 &

/usr/local/bin/cpuHog2 &

/usr/local/bin/cpuHog3 &

/usr/local/bin/cpuHog4 &

In a separate putty session, run the top command and identify the two process IDs. You may note that between them they consume more or less all the CPU. Look at the %CPU column which tells that, roughly, 50%, or maybe 49.5% of the CPU resource is being consumed by each process.

The renice command can be used to change the ‘nice’ value associated with a process.

The command

renice –n 9 1360

would assign a nice value of ‘9’ to process ID 1360.

The two process are processing important business information. However, for financial reasons, management want cpuHog1 a full 100% of a core to itelf, cpuHog2 to get 50%, cpuHog3 to register 40% and cpuHog4 10%.

Use the renice command to ‘balance’ the cpu usage so that instead of a 1:1:1:1 ratio of you get the resource allocation management require. The figures are approximate.

What nice value did you need, in the end?

3

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND

12240 instruc+ 20 0 113284 1180 996 R 97.4 0.0 1:09.76 cpuHog1

11850 instruc+ 23 3 113284 1176 996 R 50.3 0.0 15:01.35 cpuHog2

12163 instruc+ 24 4 113284 1180 996 R 38.4 0.0 3:08.40 cpuHog3

11875 instruc+ 29 9 113284 1180 996 R 12.3 0.0 8:03.61 cpuHog4

Finally, kill both processes.

## Scheduling an ‘at’ job.

This exercise can be done as ‘instructor’. There is no need switch to ‘root’.

Note the time by running the date command.

Run the command

date +%Y%m%d$H$M

This gives a timestamp.

Use the at command to get the script /usr/local/bin/atJob to run three minutes in the future. Run atq, to see your schedules at jobs. Wait three minutes and list the content of /tmp. Do you see a file called atJob.<timestamp>, where <timestamp> is a few minutes later than the one you saw earlier?

Yes.

## Scheduling a cron job

To launch the ‘at’ job, run the command:

at –f /usr/local/bin/atJob now + 3 minutes

This exercise can be done as ‘instructor’. There is no need switch to ‘root’

The objective is to put an entry in your cron table which will run 5 minutes in the future.

Run the date command. What is the cron entry which would define this time. In other words if you wanted to run a command at the time and date you have just seen, what would the crontable entry be?

<A set of five figures of the approximate form 13 11 9 5 2 >

(NB: can use [https://crontab.guru/#\*\_\*\_\*\_\*\_\*](https://crontab.guru/#*_*_*_*_*) to better understand cron)

Check your answer by running /usr/local/bin/cronNow.

Your answer and the output should be in close agreement. (The minute column might be a little different). Were they?

Yes.

Work out the cron entry that corresponds to the current time and date, as per a cron table entry, but five minutes in the future. Remember these figures.

To edit/create a cron table entry you need to set the EDITOR environment variable.

EDITOR=vi

export EDITOR

Now we can launch ‘crontab –e’ which will allow you to edite your cron table.

crontab –e

You are now running the vi editor. Type ‘o’ to open a new line.

Type the figures you just calculated then /usr/local/bin/cronJob

The entry will look something like

13 11 9 5 2 /usr/local/bin/cronJob

Quit by pressing the ‘escape key’, top left! Then ‘:wq’. Thus

‘Esc’:wq

Wait five minutes and check /tmp. You should see a file named something like

cronJob.201805091123

Obviously the time stamp will be different, but should match the time you planned.

[instructor@ml-refvm-422293 ~]$ ls /tmp

atJob.202207201424

atJob.202207202105

cronJob.20220720214002

crontab.PaNUGn

hiLog

lost+found

systemd-private-d7f1c70b33394f38ba4e684b572276ff-chronyd.service-Bafu0j

systemd-private-d7f1c70b33394f38ba4e684b572276ff-httpd.service-I3w7Dw

Run the command

cal 5 2018

This shows the calendar for May 2018. You can do this for many future years.

When will your crontab entry next run? User the cal command to find out.

*Run the script /usr/local/bin/whenNextCron. Do you agree?*

[instructor@ml-refvm-422293 ~]$ /usr/local/bin/whenNextCron

The next time your cron job will run is in 2033*.*

[instructor@ml-refvm-422293 ~]$ cal 7 2033

July 2033

Mo Tu We Th Fr Sa Su

1 2 3

4 5 6 7 8 9 10

11 12 13 14 15 16 17

18 19 20 21 22 23 24

25 26 27 28 29 30 31

yes