# Process Management

## Using the renice ratched effect

Do this exercise as ‘root’ to allow you to decrease as well as increase the nice value.

Launch the cpuHog1 processes as in the background:

/usr/local/bin/cpuHog1 &

[1] 41503

Run a command to show the processes current priority and nice value

ps -lp 41503

Renice the process to have a nice value of 10. Check this worked

renice -n 10 41503

ps -lp 41503

Kill the process.

kill 41503

## 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 &

In a separate putty session, run the top command and identify the four process IDs from these processes. 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 four 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%. The figures are approximate.

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 values did you need, in the end?

10,4,3

Finally, kill all the cpuHog processes.

pkill cpuHog

## Scheduling an ‘at’ job.

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 >

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/No.

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.

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?*