Sanjeevi Machina

<u>fuser command: -</u>

The fuser command is used to find which process is using a file, a directory or a socket. It also gives information about the user owning the process and the type of access. The fuser tool displays the process id(PID) of every process using the specified files or file systems.
Examples: -

1.To Identify which processes are using a particular file system or directory.

```
$ fuser  .
or
$fuser /fsname
or
$fuser /dirname
```

We see that the output consists of process IDs of the processes using fuser but all the PIDs are followed by a **character** 'c'. This indicates the type of access. The type of access can be any one of the following:

c	current directory
e	executable being run
f	open file. f is omitted in default display mode
F	open file for writing. F is omitted in default display mode
r	root directory
m	mmap'ed file or shared library

So 'c' in the output would mean that these processes are using this directory as their current directory.
Use Option -v to display detailed information in the output

```
2.$ fuser -v ./
3.fuser -v -n tcp 5000
4.fuser -v -k socket_serv
5.Interactively Kill Processes using fuser
  $ fuser -v -k -i socket_serv
6.$ fuser -v -k -i ./
7.fuser -k  123/tcp
```

<u>lsof command: -</u>

lsof command stands for List Of Open File. This command provides a list of files that are opened. It gives the information to find out the files which are opened by which process. With one go it lists out all open files in output console. It can be combined with grep command can be used to do advanced searching and listing.
Syntax:

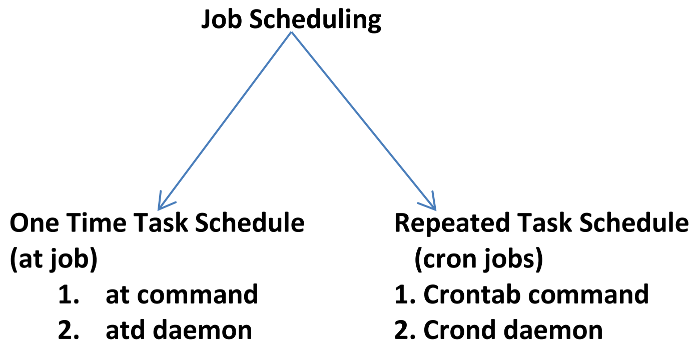
```
#lsof [option][user name]
or
#lsof -p process ID
or
lsof -D directory path
```

Scheduling future tasks (Job Scheduling): -

Scheduling one-time task with at:-

1. Users can schedule jobs for 'atd' daemon using the command line tool 'at'. The atd daemon provides 26 queues a to z, with jobs in alphabetically, later queues getting less system priority (higher nice values)
2. at command read the command to execute the command from standard-in. Below is the syntax of at command:

```
at      TIMESPEC  <filename_in_which_commands_are_entered
atq : to list at jobs
```



1. Scheduling one-time task with at command:- Users can schedule one-time jobs using at command line tool. The atd daemons provides 26 queues a to z, with jobs in alphabetically, later queues getting less system priority (higher nice levels). Synatx:

```
at TIMESPEC < /path/to/fileInwhichCommandsAreEntered
 where TIMESPEC is HHmm  MMDDYYYY
ex:- at 1245  12312019 < myjobs.sh
Note:-
atq : lists all the at jobs
or
at –l
```

To see the job in detail: at –c JobNumber

If the at command is not redirecting the output to any file, the output will be sent as mail to the user.

At acommand also use the blow format:

```
at  -f  /path/to/filenameToExecute   -t   CCYYMMDDhhmm.SS [ increment]
```

where –f: file as input, rather using standard input

-t: Time in the format of CCYYMMDDhhmm.SS

Instead of time format you can also use:

```
noon
midnight
now
teatime
tomorrow
HH:MM (hours:Minutes)
now +5min
teatime tomorrow (teatime is 16:00)
noon +4 d ays
5pm august 3 201 6
```

Examples:-

1. at –f /File/ToExecute   now +1minute
2. at – f /File/ToExecute   now nextminute
3. at -f /path/to/file -t 201909251330.15

the above tell the time 2019the year, Sept 25th 1:30PM 15seconds

To list at jobs: You can use atq or at -l command

Note:-

1. at -c JobNumber will list in detail about the job such as date and time and the program to run at that time.
2. /etc/at.deny and /etc/at.allow is used to restrict user to run the at jobs.
3. To delete at jobs use at –r JobNumber (to delte a particular job) or at –r –u username (to delte all jobs submitted by that respective user)

<u>Cancelling at jobs:-</u>

at –r : Cancels an at job (at –d JobNuber)

atrm JobNum :Cancels at job by jobNumber

atrm username : cancels at jobs by the user (root can use it for any user; users can cancel their own jobs and not other users)

atrm: cancels all at jobs belonging to the user invoking the atrm command.

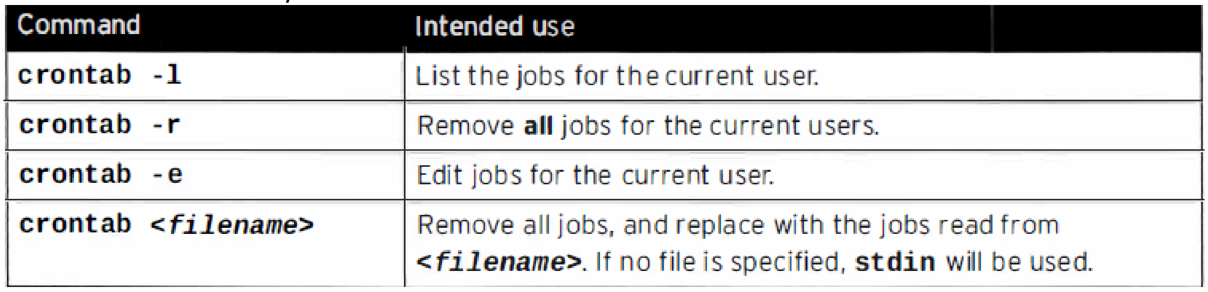
Note:- at –d equivalent to atrm command.

<u>Restricting users to run at command:-</u>

1. /etc/at.deny is used to deny user from executing at jobs (one username per line). This file will be empty by default.
2. /etc/at.allow file is used to allow users (one user per line) to execute at jobs. However, having this file empty, no users are allowed to execute at jobs. However, this rule will not be applied to root user. By default, this file will not be there. Root usercan create it if it is required. The file at.allow will have highest precedence. Ie if the user exists in both at.allow and at.deny, then the user is not allowed to execute at jobs.

Introduction to cron

1. Red Hat Enterprise Linux system's **crond** daemon enable recurring jobs to execute. crond is controlled by multiple le configuration files, one per user (edited with the crontab command), and system wide files. These configuration files give users and administrators fine-grained control over exactly when recurring jobs should be executed.
2. If the commands run from cron job produce any output to either stdout or stderr that is not redirected, the crond daemon will attempt to email that output to the user owning that job sing mail server configured on the system.
3. Normal user can use the crontab command to manager their jobs. This command can be called in four different ways:



Command	Intended use
`crontab -l`	List the jobs for the current user.
`crontab -r`	Remove **all** jobs for the current users.
`crontab -e`	Edit jobs for the current user.
`crontab <filename>`	Remove all jobs, and replace with the jobs read from `<filename>`. If no file is specified, `stdin` will be used.

Note: root can use the option –u <username> to manage the jobs for another user.

4. User's crontabs are found in /var/spool/cron/ directory. Crontab file names are created with username.
5. You must not do vi to edit crontabs. You should use crontab -e to edit crontabs. crontab -e opens vi editor and a temporary file will be created and when you save, the actual file in /var/spool/cron/`uname` will be updated.
6. Note: If at all you edit crontabs with vi editory (which is not best practice), you need to stop and start cron daemon to take effect those newly added tabs. Else newly added tabs will not be

executed by cron daemon. This is not the case when you edit crontabs with crontab -e command.

Crontabs Format:-

Individual jobs consist of six fields detailing when and what should be executed. When all five of the first fields match the current date and time, the command in the last field will be executed. These filed are (in order):

- Minutes
- Hours
- Day-of-Month
- Month
- Day-of-Week
- Command

```
Mins    Hours  DayeOftheMoth    Month    DayOftheWeek /path/to/command/toRun
```

Each field in the cron tab is separated with at least one space (you can also separate with tab)

Filed Name	Possible Values
Mins	00 – 59
Hours	00 – 23
DayeOfTheMonth	01 – 31
Month	01 - 12 (Where 01: Jan and 12: Dec)
DayOftheWeek	0 - 6 (Where 0 : Sunday and 6 : Saturday, 7 also equals to Sunday)
Command	Absolute path to the Command to run in the given schedule

Note:

a) You can specify the range - For Example: Sunday to Wednesday, you can specify 0-3, which means, 0,1,2,3. The same can be followed for Mins, Hours, Month, DayofTheweek fields too.

b) If you want to specify multiple values, they can be separated with comma. Example: Monday, Wednesday and Saturday can be specified: 1,3,6.

c) Use # symbol in the starting line, which is used to comment the crontab entry.

d) To match multiple options, use * as a substitution. Ie * for "don't care" / Always

e) */x to indicate an interval of x, ex:- */7 in the minutes column will run a job exactly every seven minutes.

Important Points to note while using crontabs:

- You can edit crontabs with : crontab -e command
- You can list crontabs with : crontab -l command
- You list other users crontabs(being as root only) : crontab -l userName
  ex:- crontab -l oraAdmin
  The above command shows crontabs for OraAdmin user. However, Ordinary user can not execute this command. Only root can do this.
- You can remove all crontab entries with crontab -r command.
- When used crontab –r command the complete crontab file will deleted in /var/spool/cron/ directory. You cannot get it back one you delete.
- The crond daemon (/usr/sbin/crond) is responsible to execute crontabs for all users.

- Crond daemon visits crontabs of users once for each 60 seconds and if any entries that matches with system's current time, those jobs will be executed.
- The output of the program is sent to user via mail, if the output is not redirected to any file.
- You can check if cron job is success or not by looking at log file /var/log/cron.

Restricting Users to run cron:

All users by default have the privilege to set up scheduled jobs to be monitored by cron. This is because the file /etc/cron.deny, which denies privileges to users, is empty. As the administrator, you can restrict access to cron by adding user names to this text file.
There is another file that also restricts user's privileges - /etc/cron.allow. To use this file, you should remove cron.deny and create the cron.allow to list the users that are allow to use cron. If cron.allow exists and is empty, NO user will be able to use cron (even root user too). If both cron.allow and cron.deny exist, then cron.allow is the file is used. If neither cron.allow or cron.deny exist, then only root can use cron.

Exercises:-

Assume there is a program called /home/apple/myprg.sh. Schedule this program under mentioned timings:
1. Every Tuesday at 02.00 PM
```
00  14  *  *  2 /home/apple/myprg.sh
```
2. Every Night at 2.00
```
00  02  *  *  * /home/apple/myprg.sh
```
3. Every night at Midnight
```
00  00  *  *  * /home/apple/myprg.sh
```
4. On 7th, 14th and 21st day at 4.20 PM
```
20  16  7,14,21  * * /home/apple/myprg.sh
```
5. Monday through Friday at 4.20AM
```
20  04  *  *  1-5 /home/apple/myprg.sh
```
6.1st Jan, 2nd Feb, 1st July and 2nd Dec 9.30 AM
```
a)30 09 1 1,7 * /home/apple/myprg
b)30 09 2 2,12 * /home/apple/myprg
```
7.1st Jan, 1st July, and 1st Dec 4.20 PM
```
20 16 1 1,7,12 * /home/apple/myprg
```

System Cron jobs:-

Apart from user cron jobs, there are also system cron jobs. System cron jobs are not defined using the crontab command, but are instead configured in a set of configuration files. The main difference in these configuration files is an extra field located between the day-of week filed and the command field, specifying under which user a job should be run.

The /etc/crontab has useful syntax diagram in the included comments

```
# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

The system cron jobs are defined in two locations: /etc/crontab and /etc/cron.d/*. There are also predefined jobs that run every hour, day, week and month. These jobs will execute all scripts placed in /etc/cron.hourly/, /etc/cron.daily/, etc/cron.weekly/ and /etc/cron.monthly/ respectively. These directories contain executable scripts and not cron configuration files.

The /etc/cron.hourly/* scripts are executed using the run-parts command from a job defined in /etc/cron.d/0hoursly. The daily, weekly, and monthly jobs are also executed using the run-parts command but from a different configuration file: /etc/crontab.

Anacrontrab:-
In order to make sure that important jobs will always be run, and not skipped accidentally because the system was turned off or hibernating when the job should have been executed, /etc/anacrontab is used to process the job once the system is up. In RHEL6, this was handled by anacron daemon and from RHEL7 onwards this file is parsed by the regular crond daemon.

The syntax of /etc/anacrontab is different from the other cron configuration files. It contains exactly four field per line:
period in days : Once per how many days this job should run
delay in minutes : The amount time the cron daemon should wait before starting this job
job-identifier: This is the name of the file in /var/spool/anacron/ that will be used to check if this job has run. When cron starts a job from /etc/anacrontab, it will update the timestamp on this file. The same timestamp is used to check when a job has last run in the past.
command: Command to be executed.

```
#period in days   delay in minutes   job-identifier   command
1                  5                 cron.daily       nice run-parts /etc/cron.daily
7                 25                 cron.weekly      nice run-parts /etc/cron.weekly
@monthly          45                 cron.monthly     nice run-parts /etc/cron.monthly
```