OPERATING SYSTEMS
CCGC-5000

btn

Module - 08

Agenda – Module 8

Authentic information is available from the given resources in course outline and URL's mentioned from this slides, and this presentation is only supportive document to be read with the given resources and corrected accordingly if required..

- Programs, daemon, process, jobs
- Parent – Child Process
- Process commands & Utilities
- File name with date of creation
- Archiving & Compression
- Scheduling, at,batch, cron,anacron
- Configuration and control files of Scheduling
- Managing Service Units
- Logging & Log rotation
- Journal & Tuning

Refer Course Book – Chapter 3, 8, 12

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/ch-automating_system_tasks

Programs, Process, Daemon, Job

- **Program**
  - a ‘passive entity’ which exists in the secondary storage persistently even if the machine reboots
- **Daemon**
  - A program that after being spawned, either at boot or by a command from a shell, disconnects itself from the terminal that started it and runs in the background.
- **Processes**
  - A process is termed as an ‘active entity’ since it is always stored in the main memory and disappears if the machine is power cycled
  - Several process may be associated with a same program
- **Job**
  - is a process that is not running in the foreground and is accessible only at the terminal with which it is associated, typically background or suspended processes.

Parent – Child process

- Processes are used by the operating system to effectively and efficiently handle system activities.
- A process is created every time you run a UNIX command or program from the command line or by scheduling or by startup programs/processes when the system boots up
- ***Parent process*** *starts new process* and the newly created process is the ***child process***.
- A process has **one parent process**, but can have **many child processes**.
- The operating system kernel identifies each process by its **process identifier.**
- Process 1, known as **systemd,** is the ancestor of every other process in the system.
- **Zombie process** or **defunct process** is a process that has completed execution but still has an entry in the process table
- An **orphan process** is a computer process remains running itself, whose parent process has finished or terminated.
- These processes may be viewed and monitored using various native tools such as **ps** *(process status)* and **top** *(table of processes)*

Process Management

- For example, when you execute a simple command like ls, a single process is created to list the contents of the working directory.
- Command **ps** report a snapshot of current processes which can be used with various options to provide output as per requirement.

```
[user1@rhel ~]$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 Oct25 ?        00:00:06 /usr/lib/systemd/systemd --s
root           2       0  0 Oct25 ?        00:00:00 [kthreadd]
root           3       2  0 Oct25 ?        00:00:00 [rcu_gp]
root           4       2  0 Oct25 ?        00:00:00 [rcu_par_gp]
```

```
[user1@rhel ~]$ ps -ax
    PID TTY      STAT   TIME COMMAND
      1 ?        Ss     0:06 /usr/lib/systemd/systemd --switched-root --system --deserial
      2 ?        S      0:00 [kthreadd]
      3 ?        I<     0:00 [rcu_gp]
      4 ?        I<     0:00 [rcu_par_gp]
```

- Process State Codes

```
D    uninterruptible sleep (usually IO)
I    Idle kernel thread
R    running or runnable (on run queue)
S    interruptible sleep (waiting for an event to complete)
T    stopped by job control signal
t    stopped by debugger during the tracing
W    paging (not valid since the 2.6.xx kernel)
X    dead (should never be seen)
Z    defunct ("zombie") process, terminated but not reaped by
     its parent
```

- **Additional** keywords

```
<    high-priority (not nice to other users)
N    low-priority (nice to other users)
L    has pages locked into memory (for real-time and custom IO)
s    is a session leader
l    is multi-threaded (using CLONE_THREAD, like NPTL pthreads
     do)
+    is in the foreground process group
```

Required reading : man ps

Process Commands & Utilities

- To view every process on the system using standard syntax: **ps –ef**
- To view process of an user, **-u** or **-U** or **U** option is used, **ps -u** *username*

  Note: additionally process STATE is given with **U** *option*
- To view process for group **–G** option is used: **ps -G** *groupname*
- To find process using PID: **ps -p** *PID*
- To find the process id's (pids) of the named programs : **pidof** *process_name*
- Command **pgrep** looks through the currently running processes and lists the process IDs which match the selection criteria to stdout.
- Command **pstree** – shows running process as a tree

Process Commands & Utilities

HUMBER
Faculty of Applied Sciences & Technology

Examples with **ps**, **pidof**, **pgrep** and some options. Use man and help to find more on these commands.

```
[unixuser@rhcsa ~]$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  1 23:49 ?        00:00:03 /usr/lib/systemd/systemd --swi
root           2       0  0 23:49 ?        00:00:00 [kthreadd]
root           3       2  0 23:49 ?        00:00:00 [ksoftirqd/0]
root           4       2  0 23:49 ?        00:00:00 [kworker/0:0]
root           5       2  0 23:49 ?        00:00:00 [kworker/0:0H]
root           6       2  0 23:49 ?        00:00:00 [kworker/u256:0]
```

```
[unixuser@rhcsa ~]$ ps -a
   PID TTY          TIME CMD
  3210 pts/0    00:00:00 ps
[unixuser@rhcsa ~]$ ps -ax
   PID TTY      STAT   TIME COMMAND
     1 ?        Ss     0:03 /usr/lib/systemd/systemd --switched-root --system --deserial
     2 ?        S      0:00 [kthreadd]
     3 ?        S      0:00 [ksoftirqd/0]
     4 ?        S      0:00 [kworker/0:0]
     5 ?        S<     0:00 [kworker/0:0H]
```

```
[unixuser@rhcsa ~]$ ps -aX
   PID    STACKP      ESP      EIP TMOUT ALARM STAT TTY      TIME COMMAND
  1892 00000000 00000000 00000000     -     - Ssl+ tty1     0:11 /usr/bin/X :0 -backgr
  3055 436410c0 43640b78 a09ce10c     -     - Ss   pts/0    0:00 bash
  3312 9afb65d0 9afb62b8 10748c00     -     - R+   pts/0    0:00 ps -aX
```

```
[unixuser@rhcsa ~]$ ps -e
   PID TTY          TIME CMD
     1 ?        00:00:03 systemd
     2 ?        00:00:00 kthreadd
     3 ?        00:00:00 ksoftirqd/0
     4 ?        00:00:00 kworker/0:0
     5 ?        00:00:00 kworker/0:0H
```

```
[unixuser@rhcsa ~]$ pidof crond
1330
[unixuser@rhcsa ~]$ pgrep crond
1330
[unixuser@rhcsa ~]$ pidof NetworkManager
966
[unixuser@rhcsa ~]$ pgrep NetworkManager
966
```

Finding process of users and group

```
[unixuser@rhcsa ~]$ ps -u root
   PID TTY          TIME CMD
     1 ?        00:00:03 systemd
     2 ?        00:00:00 kthreadd
     3 ?        00:00:00 ksoftirqd/0
```

```
[unixuser@rhcsa ~]$ ps -U root
   PID TTY          TIME CMD
     1 ?        00:00:03 systemd
     2 ?        00:00:00 kthreadd
     3 ?        00:00:00 ksoftirqd/0
```

```
[unixuser@rhcsa ~]$ ps U root
   PID TTY      STAT   TIME COMMAND
     1 ?        Ss     0:03 /usr/lib/systemd/systemd --switched-root --system --deserialize 22
     2 ?        S      0:00 [kthreadd]
     3 ?        S      0:00 [ksoftirqd/0]
```

Finding process name using PID

```
[user1@rhel ~]$ ps -p 1
    PID TTY          TIME CMD
      1 ?        00:00:07 systemd
```

```
[user1@rhel ~]$ ps -p 2
    PID TTY          TIME CMD
      2 ?        00:00:00 kthreadd
```

Finding parent process with **pstree** command

```
[user1@rhel ~]$ pidof sshd
1617
[user1@rhel ~]$ pstree -s 1617
systemd---sshd
```

*Need to find PID of the process and then use **pstree -s PID** to find the parent*

Using options -ao with ps and specifying the headers, will display only those headers

```
[unixuser@rhcsa ~]$ ps -eao pid,ppid,user,tty,cmd
  PID  PPID USER     TT       CMD
    1     0 root     ?        /usr/lib/systemd/systemd --switched-root --system --dese
    2     0 root     ?        [kthreadd]
    3     2 root     ?        [ksoftirqd/0]
    5     2 root     ?        [kworker/0:0H]
    7     2 root     ?        [migration/0]
    8     2 root     ?        [rcu_bh]
```

kill

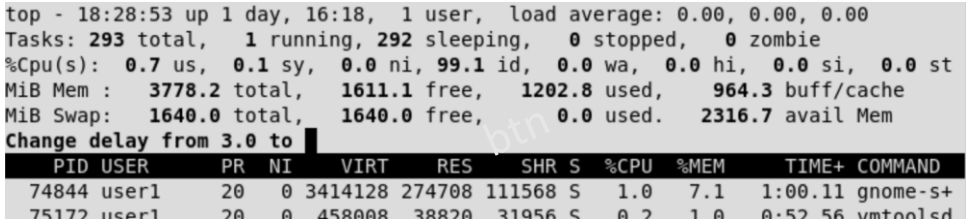
- Users can stop (**kill**) their own processes and sudo will be needed to kill other processes
- The command **kill** command sends a signal to processes, causing them to terminate or otherwise act upon receiving the signal in some way.
- To list all signals **kill -l** or **kill -L** is used
- To stop a process with its process id (PID) **kill** *pid* is used which sends SIGTERM signal or **kill -15** *pid*
- To stop a process id with signal SIGKILL, **kill -9 pid** is used
- The command **kill** need to be used cautiously, as stopping the wrong process could affect the system performance and functioning.

Process Commands & Utilities

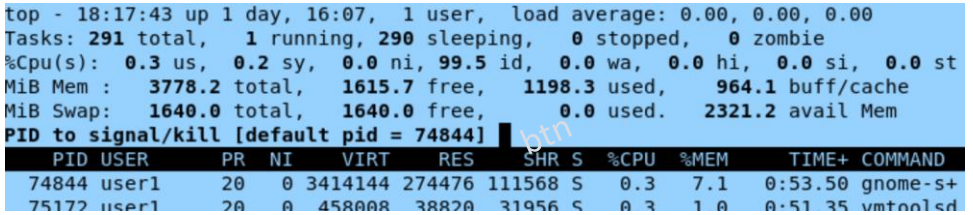
top

- top program provides a dynamic real-time view of a running system.
- It can display
  - system summary information
  - as well as a list of processes or threads currently being managed by the Linux kernel
- By default, 3 secs is the delay in refreshing the output
  - which can be changed **top –d** *delay_in_secs*
  - example to change the refresh delay to 10 secs is **top –d 10**
- to display only specific process by PID,
  - **top –p** *pid no*, to display only pocess with pid 2 is **top –p 2**
- to display process by users
  - **top –u** *username*, to display processes of user1 is **top –u user1**

When top is running some commands be run
For example when **d** is pressed, the delay can be modified



Likewise, to stop a process, **k** can be pressed to kill the process. Need to enter **PID** and then **signal** and press enter



Refer to man top for more options available

Required reading: man top

top


```
top - 00:06:52 up 16 min,  2 users,  load average: 0.00, 0.01, 0.05
Tasks: 304 total,   1 running, 302 sleeping,   1 stopped,   0 zombie
%Cpu(s):  0.1 us,  0.1 sy,  0.0 ni, 99.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1865308 total,   259304 free,  1047456 used,   558548 buff/cache
KiB Swap:  2097148 total,  2097148 free,        0 used.   585272 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
  531 root      20   0       0      0      0 S   0.3  0.0   0:00.73 xfsaild/dm-0
  695 root      20   0       0      0      0 S   0.3  0.0   0:04.07 kworker/1:3
  883 root      20   0   21672   1296    972 S   0.3  0.1   0:00.22 irqbalance
  899 root      20   0  320060   6532   5076 S   0.3  0.4   0:03.97 vmtoolsd
 1892 root      20   0  401740  42348  13656 S   0.3  2.3   0:14.30 X
 2457 unixuser  20   0 3827472 129936  46316 S   0.3  7.0   0:11.05 gnome-shell
```

First Line, containing program or window name, depending on display mode current time and length of time since last boot, total number of users, system load avg over the last 1, 5 and 15 minutes

TASK and CPU States

This portion consists of a minimum of two lines. In an SMP environ-ment, additional lines can reflect individual CPU state percentages.

Tasks Line shows total tasks or threads, depending on the state of the Threads-mode toggle. That total is further classified as: running; sleeping; stopped; zombie

%CPU Line shows CPU state percentages based on the interval since the last refresh.

us - user	: time running un-niced user processes
sy - system	: time running kernel processes
ni - nice	: time running niced user processes
wa - IO-wait	: time waiting for I/O completion
hi	: time spent servicing hardware interrupts
si	: time spent servicing software interrupts
st	: time stolen from this vm by the hypervisor

MEMORY Usage

This portion consists of two lines which may express values in kibibytes (KiB) through exbibytes (EiB) depending on the scaling factor enforced with the 'E' interactive command.

Line 1 reflects physical memory, classified as:
total, free, used, and buffers

Line 2 reflects mostly virtual memory, classified as:
total, free, used, and cached (which is physical memory)

PID – PROCESS ID
USER – USER STARTED THE PROCESS
PR – PRIORITY
NI – NICE
VIRT – VIRTUAL MEMORY SIZE
RES – RESIDENT MEMORY SIZE
SHR – SHARED MEMORY SIZE
S – PROCESS STATE

Process Command & Utilities

vmstat

- reports on processes, memory, I/O and CPU, typically providing an average since the last reboot; or you can make it report usage for a current period by telling the time interval in seconds and number of iterations you desire,

  vmstat 5 10 ***(****runs vmstat every 5 secs for 10 iterations)*

```
[unixuser@rhcsa ~]$ vmstat 5 10
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0 197632  95080      0 370876    4    9    42    10   82   72  2  0 97  0  0
 0  0 197632  94696      0 370876    0    0     0    13  630  631  2  1 97  0  0
 1  0 197632  95296      0 370824   13    0    13     0 2020 2032 12  1 87  0  0
 0  0 197632  95320      0 370824    0    0     0     0  525  551  2  0 97  0  0
 0  0 197632  95372      0 370824    0    0     0     0  478  522  2  0 98  0  0
 0  0 197632  95000      0 370824    0    0     0     0  257  259  1  0 99  0  0
 0  0 197632  94532      0 370824    0    0     0     1  496  524  2  0 98  0  0
 0  0 197632  94612      0 370824    0    0     0     7  503  541  2  0 97  0  0
 0  0 197632  94680      0 370824    0    0     0     0  244  244  1  0 99  0  0
 0  0 197632  94840      0 370824    0    0     0     1  507  509  2  0 97  0  0
```

```
FIELD DESCRIPTION FOR VM MODE
   Procs
       r: The number of runnable processes (running or waiting for run time).
       b: The number of processes in uninterruptible sleep.

   Memory
       swpd: the amount of virtual memory used.
       free: the amount of idle memory.
       buff: the amount of memory used as buffers.
       cache: the amount of memory used as cache.
       inact: the amount of inactive memory.  (-a option)
       active: the amount of active memory.  (-a option)

   Swap
       si: Amount of memory swapped in from disk (/s).
       so: Amount of memory swapped to disk (/s).

   IO
       bi: Blocks received from a block device (blocks/s).
       bo: Blocks sent to a block device (blocks/s).

   System
       in: The number of interrupts per second, including the clock.
       cs: The number of context switches per second.

   CPU
       These are percentages of total CPU time.
       us: Time spent running non-kernel code.  (user time, including nice time)
       sy: Time spent running kernel code.  (system time)
       id: Time spent idle.  Prior to Linux 2.5.41, this includes IO-wait time.
       wa: Time spent waiting for IO.  Prior to Linux 2.5.41, included in idle.
       st: Time stolen from a virtual machine.  Prior to Linux 2.6.11, unknown.
```

iostat

- Command **iostat** reports on
  - Central Processing Unit (CPU) statistics and
  - Input/output statistics for devices and partitions.
- Used for monitoring system input/output device loading by observing the time the devices are active in relation to their average transfer rates.
- Command **iostat** requires package **sysstat** to be installed
- Refer manual pages for various options could be used with iostat and definition for various columns, For example iostat –x, iostat –c, iostat –d

```
[unixuser@rhcsa ~]$ iostat
Linux 3.10.0-862.el7.x86_64 (rhcsa.rha.net)     10/22/2018      _x86_64_        (6 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           2.12    0.00    0.42    0.01    0.00   97.45

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda               4.90       221.18        52.70    1120976     267102
sdb               0.04         0.86         0.00       4344          0
dm-0              3.62       195.18         4.14     989230      20982
dm-1             17.11        20.64        48.16     104584     244072
```

mpstat

```
[unixuser@rhcsa ~]$ mpstat
Linux 3.10.0-862.el7.x86_64 (rhcsa.rha.net)     10/22/2018      _x86_64_        (6 CPU)

02:01:08 AM  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
02:01:08 AM  all    2.47    0.00    0.43    0.01    0.00    0.01    0.00    0.00    0.00   97.08
```

- The **mpstat** command writes to standard output activities for each available processor, processor 0 being the first one.
- Global average activities among all processors are also reported.
- It is included in package **sysstat**

CPU
Processor number. The keyword all indicates that statistics are calculated as averages among all processors.

%usr
Show the percentage of CPU utilization that occurred while executing at the user level (application).

%nice
Show the percentage of CPU utilization that occurred while executing at the user level with nice priority.

%sys
Show the percentage of CPU utilization that occurred while executing at the system level (kernel). Note that this does not include time spent servicing hardware and software interrupts.

%iowait
Show the percentage of time that the CPU or CPUs were idle during which the system had an outstanding disk I/O request.

%irq
Show the percentage of time spent by the CPU or CPUs to service hardware interrupts.

%soft
Show the percentage of time spent by the CPU or CPUs to service software interrupts.

%steal
Show the percentage of time spent in involuntary wait by the virtual CPU or CPUs while the hypervisor was servicing another virtual processor.

%guest
Show the percentage of time spent by the CPU or CPUs to run a virtual processor.

%gnice
Show the percentage of time spent by the CPU or CPUs to run a niced guest.

%idle
Show the percentage of time that the CPU or CPUs were idle and the system did not have an outstanding disk I/O request.

File name with date of creation

- To create a file with file name being date and time it was created, the file name should be as

  $(date +'%Y%m%d-%H%M%S')

*Refer to **man date** for more information on the options used*

- For example, daily backup file could be named as YYYYMMDD-HHMMSS-dailybackup
- Try command **touch $(date +'%Y%m%d-%H%M%S')**
- Similarly text editors can be used to create files with date and time of creation.

Archiving

- Command **tar** is used to archive and compress file/files/directory
- To archive: **tar -cf** *archivedfile.tar originalfile/directory*
- To archive and compress:

  tar -czf *archivedfile.tgz originalfile/directory*
- To list files in a archive file: **tar -tf** *archivefile.tar*
- To extract files in tar or tgz file: **tar -xf** *archivedfile.tar*
- To extract files in tar or tgz file and extract to a directory :

  tar -xf *archived file.tar* **-C** *directory*
- Other archiving tools: **star**, **gzip**, **bzip2**
- **Required reading : Chapter 3 of Linux Course Book**.

Scheduling

- Tasks, also known as jobs, can be configured to run automatically within a specified period of time, on a specified date, or when the system load average decreases below 0.8.
- Red Hat Enterprise Linux is pre-configured to run important system tasks to keep the system updated.
- For example, the locate database used by the locate command is updated daily.
- A system administrator can use automated tasks to perform periodic backups, monitor the system, run custom scripts, and so on.
- Red Hat Enterprise Linux comes with the following automated task utilities: **cron**, **anacron**, **at**, and **batch**.
- Every utility is intended for scheduling a different job type: while **cron** and **anacron** schedule recurring jobs, **at** and **batch** schedule one-time jobs

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/

at, batch command

- Command **at** run jobs queued for later execution
- at is used with
  - time month day, or
  - with now or now +1 day or now +2 hour or now + 2 min
  - with time followed by today or tomorrow or next day
  - and this enters at prompt where the command to be executed is entered and Press enter
  - to exit need to enter CTL +D
- Command **atq** lists the user's pending jobs, unless the user is the superuser; in that case, everybody's jobs are listed
- Command **atrm** deletes jobs, identified by their job number
- The command **batch** executes commands when system load levels permit; in other words, when the load average drops below 0.8, or the value specified in the invocation of atd.*(atd run jobs queued for later execution)*

```
[user1@rhel ~]$ at 8pm Nov 2
warning: commands will be executed using /bin/sh
at> bash report1
at> <EOT>
job 4 at Mon Nov  2 20:00:00 2020
```

```
[user1@rhel ~]$ at now
warning: commands will be executed using /bin/sh
at> bash weeklyreport
at> <EOT>
job 5 at Wed Oct 28 22:36:00 2020
[user1@rhel ~]$
[user1@rhel ~]$ at 2am tomorrow
warning: commands will be executed using /bin/sh
at> bash report2
at> <EOT>
job 6 at Thu Oct 29 02:00:00 2020
[user1@rhel ~]$ at 11.59pm next day
warning: commands will be executed using /bin/sh
at> bash salesrep
at> <EOT>
job 7 at Thu Oct 29 23:59:00 2020
[user1@rhel ~]$ at 11.59pm today
warning: commands will be executed using /bin/sh
at> bash dlyreport
at> <EOT>
job 8 at Wed Oct 28 23:59:00 2020
```

```
[user1@rhel ~]$ atq
4       Mon Nov  2 20:00:00 2020 a user1
6       Thu Oct 29 02:00:00 2020 a user1
7       Thu Oct 29 23:59:00 2020 a user1
8       Wed Oct 28 23:59:00 2020 a user1
```

```
[user1@rhel ~]$ atrm 4
[user1@rhel ~]$ atq
6       Thu Oct 29 02:00:00 2020 a user1
7       Thu Oct 29 23:59:00 2020 a user1
8       Wed Oct 28 23:59:00 2020 a user1
```

at & *batch* control files

- Default configuration for *at* & *batch* is to allow everyone to use it (which is not always recommended)
- Access control two files /etc/at.allow and /etc/at.deny
- By default at.deny exists and the usernames listed are not allowed to use at.
- If at.deny is empty, which allows everyone to use at and batch
- Alternatively, at.allow file does not exist by default.
- If at.allow file is blank, no one except root is allowed to schedule jobs
- Either at.allow or at.deny should be used as best practice

Cron and Anacron

- Both, **cron** and **anacron**, are daemons that can schedule execution of recurring tasks to a certain point in time defined by the exact time, day of the month, month, day of the week, and week.
- **Cron** jobs can run as often as every minute. However, the utility assumes that the system is running continuously and if the system is not on at the time when a job is scheduled, the job is not executed.
- On the other hand, **Anacron** remembers the scheduled jobs if the system is not running at the time when the job is scheduled. The job is then executed as soon as the system is up. However, Anacron can only run a job once a day.
- To install Cron and Anacron, **cronie** and **cronie-anacron** packages are required
- The cron and anacron jobs are both picked by the **crond** service.
- Command **systemctl** is used to manage the **crond** service
- Logs are stored in **/var/log/cron**

Required reading Chapter 8, man crontab and **man 5 crontab**
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/

Anacron and Cron

- The main configuration file to schedule jobs is the **/etc/anacrontab** file, which can be only accessed by the **root** user

```
[user1@rhel ~]$ cat /etc/anacrontab
# /etc/anacrontab: configuration file for anacron

# See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#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
```

- The main configuration file to schedule jobs is the **/etc/crontab** file, which can be only accessed by the root user
- Any lines that begin with a hash sign (#) are comments and are not processed.
- Users other than root can configure cron tasks with the **crontab** utility.
- The user-defined crontabs are stored in the **/var/spool/cron/** directory and executed as if run by the users that created them.

/etc/crontab

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# 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
```

Required reading Chapter 8, man crontab and **man 5 cron**

Cron configuration

- User cron jobs can be created or edit scheduled cron job using **crontab -e**
- To view an existing scheduled cron job, **contab -l** can be used.
- When using the first time need to select the default text editor to edit crontab

- The schedule has to be given in sequence of minute(m), hour(h), day of month(dom), month(mon), day of week(dow) and then username, command to run as per schedule

- Best practice is to use a script to execute the command
- Using sudo with crontab, is owned by root and not the user who created.
- If path is not specified for input file location or output file location, the home directory of the crontab user will be default path

Required reading Chapter 8, man crontab and **man 5 cronb**

Cron configuration

- Highly recommended to use script in crontab to execute scheduled tasks
- Example of crontab entries, to schedule task
  - daily at 11.59pm

```
59 23 * * 1-5 bash daily_backup
```

  - every 1st day of the month task at 12.01am

```
1 0 1 * * bash month_end_report
```

  - every 1st Sunday of the month at 12.10 am

```
10 0 1-7 * 0 bash first_sunday_report
```

  - end of every quarter report to run at beginning of every quarter at 12.01 am

```
1 0 1 1,4,7,10 * bash quarterly-report
```

Cron configuration

- Some more examples of cronjob being scheduled :

```
59 23 * * 1-5 bash daily_backup
1 0 1 * * bash month_end_report
10 0 1-7 * 0 bash first_sunday_report
1 0 31 3,12 * bash quarterly_report
1 0 30 6,9 * bash quarterly_report
```

- For any of the above values, an asterisk (*) can be used to specify all valid values. For example, define the month value as an asterisk, the job will be executed every month within the constraints of the other values.
- A hyphen (-) between integers specifies a range of integers. For example, 1-4 means the integers 1, 2, 3, and 4.
- A list of values separated by commas (,) specifies a list. For example, 3, 4, 6, 8 indicates exactly these four integers.
- The forward slash (/) can be used to specify step values. The value of an integer will be skipped within a range following the range with /integer. For example, minute value defined as 0-59/2 denotes every other minute in the minute field. Step values can also be used with an asterisk. For instance, if the month value is defined as */3, the task will run every third month.

Required reading man crontab and **man 5 crontab**

cron scheduling control

- If the **/etc/cron.allow** file exists, then you must be listed (one user per line) therein in order to be allowed to use this command.
- If the **/etc/cron.allow** file does not exist but the **/etc/cron.deny** file does exist, then you must not be listed in the /etc/cron.deny file in order to use this command.
- If neither of these files exists, then depending on site-dependent configuration parameters, only the super user will be allowed to use this command, or all users will be able to use this command.
- **If both files exist then /etc/cron.allow takes precedence**. Which means that /etc/cron.deny is not considered and your user must be listed in /etc/cron.allow in order to be able to use the crontab.
- Regardless of the existence of any of these files, the root administrative user is always allowed to setup a crontab.
- Empty cron.deny (default), everyone can set cron jobs
- Empty cron.allow file means that no one (except root) can set jobs

Managing Service Units

- Service unit, starts, stops, restarts or reloads service daemons and the processes they are made up of. It also handles services controlled by scripts in the /etc/rc.d/init.d directory in previous RHEL releases
- To check whether the service unit is set to autostart at the next system reboot: **systemctl is-enabled** *service_unit*

```
[user1@rhel ~]$ systemctl is-enabled crond
enabled
```

- To check whether the service unit is running : **systemctl is-active** *service_unit*

```
[user1@rhel ~]$ systemctl is-active crond
active
```

- To check the current operational status of the service unit and its location : **systemctl status** *service_unit*

```
[user1@rhel ~]$ systemctl status crond
● crond.service - Command Scheduler
   Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2020-08-22 20:33:30 EDT; 2h 7min ago
 Main PID: 1111 (crond)
    Tasks: 1 (limit: 11366)
   Memory: 6.0M
   CGroup: /system.slice/crond.service
           └─1111 /usr/sbin/crond -n
```

Managing Service Units

- To stop the service unit : **systemctl stop** *service_unit*
- To start the service unit : **systemctl start** *service_unit*

```
[user1@rhel ~]$ sudo systemctl stop crond
[user1@rhel ~]$ sudo systemctl is-active crond
inactive
[user1@rhel ~]$ sudo systemctl start crond
[user1@rhel ~]$ sudo systemctl is-active crond
active
```

Required reading Chapter 12 of course book

- To disable a service unit from autostarting at the next system reboot : **systemctl disable** *service_unit*
- To enable a service unit to autostart at the next system reboot : **systemctl enable** *service_unit*

```
[user1@rhel ~]$ sudo systemctl disable crond
Removed /etc/systemd/system/multi-user.target.wants/crond.service.
[user1@rhel ~]$ systemctl is-enabled crond
disabled
[user1@rhel ~]$ sudo systemctl enable crond
Created symlink /etc/systemd/system/multi-user.target.wants/crond.service → /usr/lib/systemd/system/crond.service.
[user1@rhel ~]$ systemctl is-enabled crond
enabled
```

System Logging

- System logging is an essential and one of the most basic elements of an operating system.
- In RHEL, it is performed to **capture messages generated** by the kernel, daemons, commands, user activities, applications, and other events.
- These messages are forwarded to various log files, which **store them for security auditing, service malfunctioning, system troubleshooting, or informational purposes**.
- The daemon that is responsible for system logging is called **rsyslogd.**
- **rsyslogd** daemon's service **rsyslog (syslog)** can be managed with **systemctl** command
- **rsyslogd** daemon reads System Log Configuration file /etc/rsyslog.conf and the files located in the /etc/rsyslog.d directory at startup
- The default port this daemon uses is **514**, which may be configured to use either UDP or TCP protocol.
- The default repository for most system log files is the /var/log directory, as defined in /etc/rsyslog.conf
- /var/log/boot.log file logs generated during system startup
- /var/log/messages logs most of system activities
- Custom logging could be done with **logger** command which appends to /var/log/messages

System Logging

- Typical /var/log directory

```
[user1@rhel ~]$ ls /var/log
anaconda                cups                    lastlog                 secure                  vmware-network.3.log
audit                   dnf.librepo.log         libvirt                 secure-20201013         vmware-network.4.log
boot.log                dnf.librepo.log-20201013 maillog                secure-20201018         vmware-network.5.log
boot.log-20200916       dnf.librepo.log-20201018 maillog-20201013       secure-20201026         vmware-network.6.log
boot.log-20200924       dnf.librepo.log-20201026 maillog-20201018       secure-20201028         vmware-network.7.log
boot.log-20201008       dnf.librepo.log-20201028 maillog-20201026       speech-dispatcher       vmware-network.8.log
boot.log-20201009       dnf.log                 maillog-20201028        spooler                 vmware-network.9.log
boot.log-20201014       dnf.rpm.log             messages                spooler-20201013        vmware-network.log
boot.log-20201018       firewalld               messages-20201013       spooler-20201018        vmware-vgauthsvc.log.0
btmp                    gdm                     messages-20201018       spooler-20201026        vmware-vmsvc-root.log
btmp-20201028           glusterfs               messages-20201026       spooler-20201028        vmware-vmtoolsd-root.log
chrony                  hawkey.log              messages-20201028       sssd                    vmware-vmusr-root.log
cron                    hawkey.log-20201013     private                 swtpm                   wtmp
cron-20201013           hawkey.log-20201018     qemu-ga                 tuned                   Xorg.9.log
cron-20201018           hawkey.log-20201026     rhsm                    vmware-imc
cron-20201026           hawkey.log-20201028     sa                      vmware-network.1.log
cron-20201028           insights-client         samba                   vmware-network.2.log
```

- The **dmesg** command is used to examine or control the kernel ring buffer.
- By design, the /var/log/dmesg file is not generated during boot.
- The kernel ring buffer is captured within the systemd-journal as well as /var/log/messages, via the imjournal rsyslog plugin
- The **wtmp**, **utmp** logs in logins. The utmp file allows one to discover information about who is currently using the system. These files could be referenced by commands **w** and **who**
- The **btmp** and **wtmp** files are used by **last** and **lastb** commands. **ac** command refers **wtmp**
- The **lastlog** files is referenced by **lastlog** command

https://www.redhat.com/sysadmin/monitor-users-linux

/etc/rsyslog.conf

- The rsyslog configuration file in /etc/rsyslog.conf contains three sections : Modules, Global Directives and Rules
- **Modules** section includes two modules : **imuxsock** and **imjournal**
- **Global Directives** contains five active directives. The definitions in this section influence the rsyslogd daemon as a whole.
  - The **first** of the five directives specifies the location for the storage of auxiliary files,
  - the **second** directive instructs the daemon to save captured messages in the traditional way,
  - the **third** directive directs the daemon to read additional configuration files from /etc/rsyslogd.d/ and loads them,
  - the **fourth** directive orders the daemon to retrieve local messages via imjournal rather than the old local log socket, and
  - the **last** directive defines the file name to store the position in the journal.
- **Rules** section, each line entry consists of two fields.
  - The left field is called selector and the right field is referred to as action.
  - The selector field is further divided into two dot-separated sub-fields called facility (left) and priority (right)

Facilities and Levels

- Log messages are entered with specified priority which may be specified as **facility.level** pair
- The **facility** describes the part of the system generating the message, and is one of the following keywords: auth, authpriv, cron, daemon, kern, lpr, mail, news, syslog, user, uucp and local0 through local7.
- The **priority** (**level**) describes the severity of the message, and is a keyword from the following ordered list (higher to lower): emerg, alert, crit, err, warning, notice, info, debug.
- **Required Reading:**

  man logger,

  man 5 rsyslog.conf,

  man 3 syslog

```
FACILITIES AND LEVELS
      Valid facility names are:

            auth
            authpriv   for security information of a sensitive nature
            cron
            daemon

            ftp
            kern       cannot be generated from userspace process, automatically converted to user
            lpr
            mail
            news
            syslog
            user
            uucp
            local0
              to
            local7
            security   deprecated synonym for auth
```

```
Valid level names are:

      emerg
      alert
      crit
      err
      warning
      notice
      info
      debug
      panic     deprecated synonym for emerg
      error     deprecated synonym for err
      warn      deprecated synonym for warning

For the priority order and intended purposes of these facilities and levels, see syslog(3).
```

Log Rotation

- The purpose of log rotation is to archive and compress old logs so that they consume less disk space, but are still available for inspection as needed.
- The **logrotate** command helps in log rotation. Typically, **logrotate** is called from the system-wide cron script **/etc/cron.daily/logrotate**
- /etc/cron.daily/logrotate invokes the logrotate command once every day to rotate log files by sourcing the /etc/logrotate.conf file and the configuration files located in the /etc/logrotate.d directory.
- These files may be modified to include additional tasks such as removing, compressing, and emailing identified log files.
- Log rotation is defined by the configuration file **/etc/logrotate.conf**.
- Individual configuration files can be added into **/etc/logrotate.d** (for example, where the yum(dnf) and cups configurations are stored).

systemd journal

- **systemd-journald** daemon is used to implement systemd based logging service for collection and storage of logging data
- systemd journal stores in binary format in files called journals located in **/run/log/journal** directory
- Configuration file is /etc/systemd/journald.conf
- Command **journalctl** retrieves messages from journal and various option are available for viewing in different formats

```
[user1@rhel ~]$ sudo journalctl
-- Logs begin at Sat 2020-10-17 23:08:00 EDT, end at Tue 2020-10-27 00:21:18 EDT. --
Oct 17 23:08:00 rhel1.nest207.net kernel: Linux version 4.18.0-193.el8.x86_64 (mockbuild
Oct 17 23:08:00 rhel1.nest207.net kernel: Command line: BOOT_IMAGE=(hd0,msdos1)/vmlinuz-
Oct 17 23:08:00 rhel1.nest207.net kernel: Disabled fast string operations
```

- To show alerts generated by a particular service, /usr/sbin/sshd

```
[user1@rhel ~]$ sudo journalctl /usr/sbin/sshd
-- Logs begin at Sat 2020-10-17 23:08:00 EDT, end at Tue 2020-10-27 00:28:15 EDT. --
Oct 17 23:08:21 toronto sshd[1617]: Server listening on 0.0.0.0 port 22.
Oct 17 23:08:21 toronto sshd[1617]: Server listening on :: port 22.
```

- Required reading of **man journalctl** and Chapter 12 of course book

System Tuning

- To monitor storage, networking,processor, audo, video and variety of other connected devices and adjusts their parameters for better performance **tuned** service is used
- service tuned includes nine predefined profiles located in **/usr/lib/tuned** to support a variety of cases and custom profiles can be created and saved **/etc/tuned** for tuned service to recognize it.
- Tuning profiles are in three groups
  - optimized for better performance
  - geared more towards power consumption
  - that offers a balance between other two and maximum performance/power combination
- Command **tuned-adm** is a single profile management command for tuned
- To list available tuning profiles: **tuned-adm list**
- To list active profile: **tuned-adm active**
- To switch to powersafe profile: **tuned-adm profile powersave**
- To turn off tuning: **tuned-adm off**

Required Activity: Exercise 12-1