Process Management

Prabhjeet Singh

Process Management

1. **PS command –** print all the current process on the machine

NAME

ps - report a snapshot of the current processes.

SYNOPSIS

ps [options]

DESCRIPTION

ps displays information about a selection of the active processes. If you want a repetitive update of the selection and the displayed information, use top instead.

This version of ps accepts several kinds of options:

1 UNIX options, which may be grouped and must be preceded by a dash.

2 BSD options, which may be grouped and must not be used with a dash.

3 GNU long options, which are preceded by two dashes.

Options of different types may be freely mixed, but conflicts can appear. There are some synonymous options, which are functionally identical, due to the many standards and ps implementations that this ps is compatible with.

Note that ps -aux is distinct from ps aux. The POSIX and UNIX standards require that ps -aux print all processes owned by a user named x, as well as printing all processes that would be selected by the -a option. If the user named x does not exist, this ps may interpret the command as ps aux instead and print a warning. This behavior is intended to aid in transitioning old scripts and habits. It is fragile, subject to change, and thus should not be relied upon.

By default, ps selects all processes with the same effective user ID (euid=EUID) as the current user and associated with the same terminal as the invoker. It displays the process ID (pid=PID), the terminal associated with the process (tname=TTY), the cumulated CPU time in [DD-]hh:mm:ss format (time=TIME), and the executable name (ucmd=CMD). Output is unsorted by default.

The use of BSD-style options will add process state (stat=STAT) to the default display and show the command args (args=COMMAND) instead of the executable name. You can override this with the PS_FORMAT environment variable. The use of BSD-style options will also change the process selection to include processes on other terminals (TTYs) that are owned by you; alternately, this may be described as setting the selection to be the set of all processes filtered to exclude processes owned by other users or not on a terminal. These effects are not considered when options are described as being "identical" below, so -M will be considered identical to Z and so on.

Except as described below, process selection options are additive. The default selection is discarded, and then the selected processes are added to the set of processes to be displayed. A process will thus be shown if it meets any of the given selection criteria.

EXAMPLES

To see every process on the system using standard syntax:

```
ps -e
ps -ef
ps -eF
ps -ely
```

To see every process on the system using BSD syntax:

```
ps ax
ps axu
```

To print a process tree:

```
ps -ejH
ps axjf
```

To get info about threads:

```
ps -eLf
ps axms
```

To get security info:

```
ps -eo euser,ruser,suser,fuser,f,comm,label
ps axZ
ps -eM
```

To see every process running as root (real & effective ID) in user format:

```
ps -U root -u root u
```

To see every process with a user-defined format:

```
ps -eo pid,tid,class,rtprio,ni,pri,psr,pcpu,stat,wchan:14,comm
ps axo stat,euid,ruid,tty,tpgid,sess,pgrp,ppid,pid,pcpu,comm
ps -Ao pid,tt,user,fname,tmout,f,wchan
```

Print only the process IDs of syslogd:

```
ps -C syslogd -o pid=
```

Print only the name of PID 42:

```
ps -q 42 -o comm=
```

$ ps → it is for the current user only

```
  PID TTY          TIME CMD
 1347 pts/0    00:00:28 zsh
33791 pts/0    00:00:00 zsh
33808 pts/0    00:00:00 bash
39634 pts/0    00:00:00 ps
```

└─$ ps ax | head -> BSD format

```
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:16 /sbin/init splash
    2 ?        S      0:00 [kthreadd]
    3 ?        I<     0:00 [rcu_gp]
    4 ?        I<     0:00 [rcu_par_gp]
    6 ?        I<     0:00 [kworker/0:0H-events_highpri]
    8 ?        I<     0:00 [mm_percpu_wq]
    9 ?        S      0:00 [rcu_tasks_rude_]
   10 ?        S      0:00 [rcu_tasks_trace]
   11 ?        S      0:00 [ksoftirqd/0]
```

$ ps -e | head →

```
  PID TTY          TIME CMD
    1 ?        00:00:16 systemd
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 rcu_gp
    4 ?        00:00:00 rcu_par_gp
    6 ?        00:00:00 kworker/0:0H-events_highpri
    8 ?        00:00:00 mm_percpu_wq
    9 ?        00:00:00 rcu_tasks_rude_
```

```
   10 ?        00:00:00 rcu_tasks_trace
   11 ?        00:00:00 ksoftirqd/0
```

$ ps aux | head

```
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.5 164356 10692 ?        Ss   Aug31   0:16 /sbin/init splash
root           2  0.0  0.0      0     0 ?        S    Aug31   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   Aug31   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   Aug31   0:00 [rcu_par_gp]
root           6  0.0  0.0      0     0 ?        I<   Aug31   0:00 [kworker/0:0H-events_highpri]
root           8  0.0  0.0      0     0 ?        I<   Aug31   0:00 [mm_percpu_wq]
root           9  0.0  0.0      0     0 ?        S    Aug31   0:00 [rcu_tasks_rude_]
root          10  0.0  0.0      0     0 ?        S    Aug31   0:00 [rcu_tasks_trace]
root          11  0.0  0.0      0     0 ?        S    Aug31   0:00 [ksoftirqd/0]
```

$ ps -ef | head → this is for unix

```
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 Aug31 ?        00:00:16 /sbin/init splash
root           2       0  0 Aug31 ?        00:00:00 [kthreadd]
root           3       2  0 Aug31 ?        00:00:00 [rcu_gp]
root           4       2  0 Aug31 ?        00:00:00 [rcu_par_gp]
root           6       2  0 Aug31 ?        00:00:00 [kworker/0:0H-events_highpri]
root           8       2  0 Aug31 ?        00:00:00 [mm_percpu_wq]
root           9       2  0 Aug31 ?        00:00:00 [rcu_tasks_rude_]
root          10       2  0 Aug31 ?        00:00:00 [rcu_tasks_trace]
root          11       2  0 Aug31 ?        00:00:00 [ksoftirqd/0]
```

TTY – terminal used to control the process

PPID – parent process ID

PID – process ID

UID – User who initiated process

C – CPU usage

STIME – start time of the process

TIME – time of CPU used.

CMD – command that started the process.

$ ps -U root -u root u → To see every process running as root (real & effective ID) in user Format.

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.5 164356 10692 ?        Ss   Aug31   0:16 /sbin/init splash
root         2  0.0  0.0      0     0 ?        S    Aug31   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        I<   Aug31   0:00 [rcu_gp]
root         4  0.0  0.0      0     0 ?        I<   Aug31   0:00 [rcu_par_gp]
root         6  0.0  0.0      0     0 ?        I<   Aug31   0:00 [kworker/0:0H-events_highpri]
root         8  0.0  0.0      0     0 ?        I<   Aug31   0:00 [mm_percpu_wq]
root         9  0.0  0.0      0     0 ?        S    Aug31   0:00 [rcu_tasks_rude_]
root        10  0.0  0.0      0     0 ?        S    Aug31   0:00 [rcu_tasks_trace]
root        11  0.0  0.0      0     0 ?        S    Aug31   0:00 [ksoftirqd/0]
root        12  0.0  0.0      0     0 ?        I    Aug31   0:09 [rcu_sched]
root        13  0.0  0.0      0     0 ?        S    Aug31   0:01 [migration/0]
root        15  0.0  0.0      0     0 ?        S    Aug31   0:00 [cpuhp/0]
```

$ ps -eH | head → -e to show processes for the all the user., H -> To show processes hierarchically.

```
  PID TTY          TIME CMD
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00   rcu_gp
    4 ?        00:00:00   rcu_par_gp
    6 ?        00:00:00   kworker/0:0H-events_highpri
    8 ?        00:00:00   mm_percpu_wq
    9 ?        00:00:00   rcu_tasks_rude_
   10 ?        00:00:00   rcu_tasks_trace
   11 ?        00:00:00   ksoftirqd/0
   12 ?        00:00:09   rcu_sched
    1 ?        00:00:16 systemd
  276 ?        00:00:09   systemd-journal
  302 ?        00:00:00   vmware-vmblock-
  306 ?        00:00:01   systemd-udevd
  426 ?        00:00:06   haveged
  429 ?        00:07:19   vmtoolsd
  460 ?        00:00:00   cron
  461 ?        00:00:18   dbus-daemon
  463 ?        00:00:10   NetworkManager
  473 ?        00:00:15   polkitd
  476 ?        00:00:00   rsyslogd
  479 ?        00:00:02   systemd-logind
  568 ?        00:00:00   ModemManager
  574 ?        00:03:45   containerd
```
575 ?   00:00:00   lightdm
```
  593 tty7     00:03:59    Xorg
```
993 ?   00:00:00   lightdm
```
 1029 ?        00:00:00      xfce4-session
 1077 ?        00:00:00       ssh-agent
 1114 ?        00:00:21       xfwm4
 1130 ?        00:00:02       xfsettingsd
 1138 ?        00:00:10       xfce4-panel
 1148 ?        00:00:07         panel-1-whisker
 1152 ?        00:00:01         panel-16-systra
 1153 ?        00:02:14         panel-17-pulsea
```

$ pstree   - To display parent and child processes in a tree.

```
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─NetworkManager───2*[{NetworkManager}]
        ├─agetty
        ├─blueman-tray───2*[{blueman-tray}]
        ├─colord───2*[{colord}]
        ├─containerd───10*[{containerd}]
        ├─cron
        ├─dbus-daemon
        ├─dockerd───9*[{dockerd}]
        ├─haveged
        ├─lightdm─┬─Xorg───{Xorg}
        │         ├─lightdm─┬─xfce4-session─┬─Thunar───2*[{Thunar}]
        │         │         │               ├─agent───2*[{agent}]
        │         │         │               ├─blueman-applet───3*[{blueman-applet}]
```

```
|       |       |           ├─light-locker───3*[{light-locker}]
|       |       |           ├─nm-applet───3*[{nm-applet}]
|       |       |           ├─polkit-gnome-au───2*[{polkit-gnome-au}]
|       |       |           ├─ssh-agent
|       |       |           ├─xfce4-panel─┬─panel-1-whisker───2*[{panel-1-whisker}]
|       |       |           |             ├─panel-16-systra───2*[{panel-16-systra}]
|       |       |           |             ├─panel-17-pulsea───2*[{panel-17-pulsea}]
|       |       |           |             ├─panel-18-notifi───2*[{panel-18-notifi}]
|       |       |           |             ├─panel-19-power-───2*[{panel-19-power-}]
|       |       |           |             ├─panel-21-action───2*[{panel-21-action}]
|       |       |           |             └─2*[{xfce4-panel}]
|       |       |           ├─xfce4-power-man───2*[{xfce4-power-man}]
|       |       |           ├─xfdesktop───2*[{xfdesktop}]
|       |       |           ├─xfsettingsd───2*[{xfsettingsd}]
|       |       |           ├─xfwm4───2*[{xfwm4}]
|       |       |           ├─xiccd───2*[{xiccd}]
|       |       |           └─2*[{xfce4-session}]
|       |       └─2*[{lightdm}]
|       └─2*[{lightdm}]
├─polkitd───2*[{polkitd}]
├─qterminal─┬─zsh───zsh───bash───pstree
|           └─2*[{qterminal}]
├─rsyslogd───3*[{rsyslogd}]
├─rtkit-daemon───2*[{rtkit-daemon}]
├─systemd─┬─(sd-pam)
|         ├─at-spi-bus-laun─┬─dbus-daemon
|         |                 └─3*[{at-spi-bus-laun}]
|         ├─at-spi2-registr───2*[{at-spi2-registr}]
|         ├─dbus-daemon
|         ├─dconf-service───2*[{dconf-service}]
|         ├─gnome-keyring-d───3*[{gnome-keyring-d}]
|         ├─gpg-agent
|         ├─gvfs-afc-volume───3*[{gvfs-afc-volume}]
|         ├─gvfs-goa-volume───2*[{gvfs-goa-volume}]
|         ├─gvfs-gphoto2-vo───2*[{gvfs-gphoto2-vo}]
|         ├─gvfs-mtp-volume───2*[{gvfs-mtp-volume}]
|        ├─gvfs-udisks2-vo──3*[{gvfs-udisks2-vo}]
|        ├─gvfsd─┬─gvfsd-trash──2*[{gvfsd-trash}]
|        |       └─2*[{gvfsd}]
|        ├─gvfsd-fuse──6*[{gvfsd-fuse}]
|        ├─gvfsd-metadata──2*[{gvfsd-metadata}]
|        ├─obexd
|        ├─pipewire──{pipewire}
|        ├─pipewire-media-──{pipewire-media-}
|        ├─pulseaudio──2*[{pulseaudio}]
|        ├─xfce4-notifyd──2*[{xfce4-notifyd}]
|        └─xfconfd──2*[{xfconfd}]
├─systemd-journal
├─systemd-logind
├─systemd-udevd
├─udisksd──4*[{udisksd}]
├─upowerd──2*[{upowerd}]
├─vmtoolsd──2*[{vmtoolsd}]
├─vmtoolsd──3*[{vmtoolsd}]
├─vmware-vmblock-──2*[{vmware-vmblock-}]
└─xcape──{xcape}
```

$top → to give information about CPU usages by each process dynamically.

```
top - 07:20:06 up 1 day, 22:29,  1 user,  load average: 0.02, 0.01, 0.00
Tasks: 193 total,   1 running, 192 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.6 us,  0.3 sy,  0.0 ni, 99.0 id,  0.1 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   1975.2 total,    163.4 free,    628.4 used,   1183.4 buff/cache
MiB Swap:    975.0 total,    944.7 free,     30.3 used.   1139.2 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    593 root      20   0  460832 152124  59276 S   3.0   7.5   4:04.44 Xorg
   1344 kali      20   0  412652  73972  61736 S   1.3   3.7   0:49.96 qterminal
   1114 kali      20   0  406576  75056  55200 S   0.7   3.7   0:21.87 xfwm4
    426 root      20   0    8164   5272   1728 S   0.3   0.3   0:06.58 haveged
    429 root      20   0  237864   9092   5800 S   0.3   0.4   7:20.29 vmtoolsd
    574 root      20   0 1489536  49588  28512 S   0.3   2.5   3:46.19 containerd
   1153 kali      20   0  511376  33404  25384 S   0.3   1.7   2:14.30 panel-17-pulsea
   1200 kali      20   0  292380  32344  22984 S   0.3   1.6   7:24.97 vmtoolsd
  39736 kali      20   0   10304   3756   3236 R   0.3   0.2   0:00.03 top
      1 root      20   0  164356  10692   7908 S   0.0   0.5   0:16.90 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.19 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.11 rcu_par_gp
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-events_highpri
      8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
      9 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_rude_
     10 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_trace
     11 root      20   0       0      0      0 S   0.0   0.0   0:00.33 ksoftirqd/0
     12 root      20   0       0      0      0 I   0.0   0.0   0:09.47 rcu_sched
     13 root      rt   0       0      0      0 S   0.0   0.0   0:01.08 migration/0
     15 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
     16 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/1
     17 root      rt   0       0      0      0 S   0.0   0.0   0:01.17 migration/1
     18 root      20   0       0      0      0 S   0.0   0.0   0:00.30 ksoftirqd/1
     20 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/1:0H-events_highpri
     21 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/2
     22 root      rt   0       0      0      0 S   0.0   0.0   0:01.17 migration/2
     23 root      20   0       0      0      0 S   0.0   0.0   0:00.61 ksoftirqd/2
```

Foreground and Background Processes.

In Linux, only one process will be running at a time in foreground. But we can multiple background process running at the same time.

$ xeyes

ls

here xeyes is running now so ‘ls’ won’t give any result.

To run a process in the background use &

```
–$ xeyes &
[1] 39750
└─$ ls
 all.txt       dir_list.txt   file          file1_soft.txt   file2.txt  'file   space.txt'   location.txt   my_dir2    zip__backup.zip
 combine1.txt   errors.txt     file1_hard.txt   file1.txt       file3.txt  'file space.txt'    my_dir1       space.txt
```

Suspend/stop the process using ctrl+z.

$job → To see the suspended process.

$ fg → to show foreground processes.

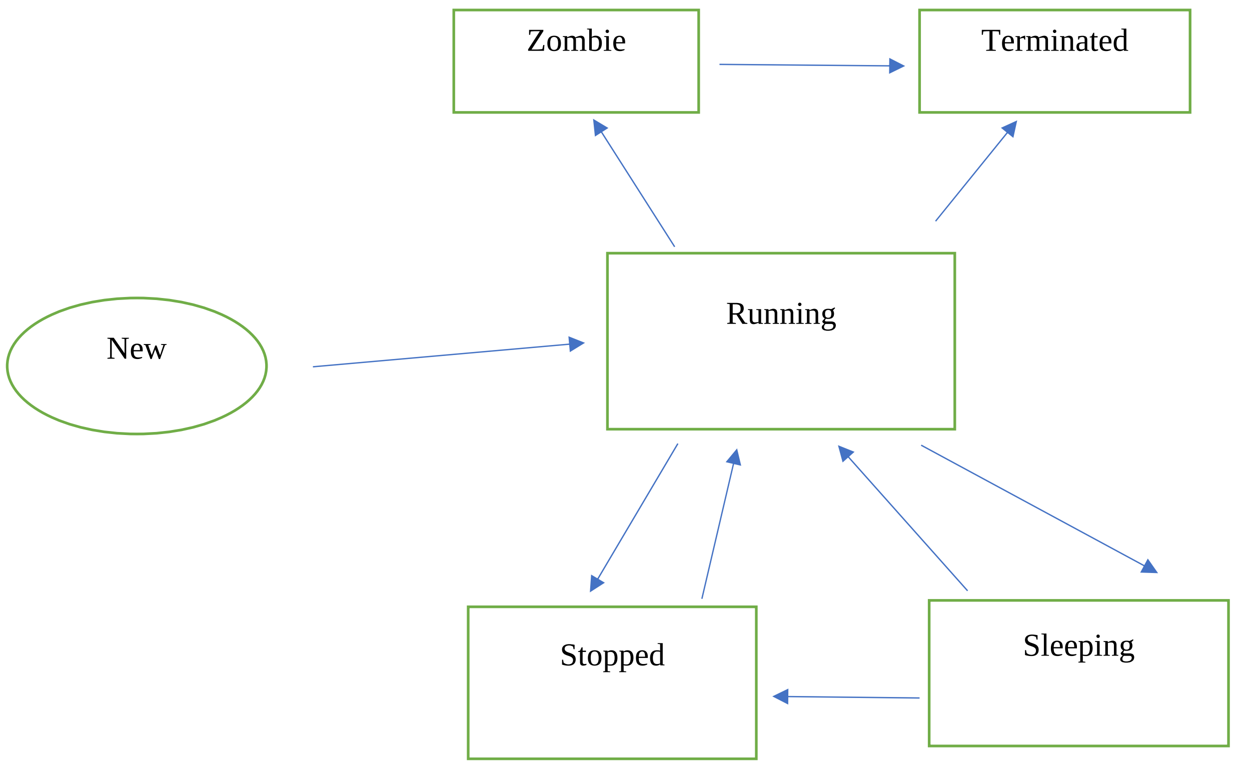
$ bg → to show background processes.

```
$ xeyes
^Z
[3]+  Stopped                 xeyes

└─$ jobs
[1]   Running                 xeyes &
[2]-  Running                 xeyes &
[3]+  Stopped                 xeyes

└─$ fg
xeyes
^Z
[3]+  Stopped                 xeyes
└─$ bg
[3]+ xeyes &
└─$ jobs
[1]   Running                 xeyes &
[2]-  Running                 xeyes &
[3]+  Running                 xeyes &
```

$ xclock → to show clock

Managing Processes.

Processes states



Zombie – processes which are not cleaned properly

Sleeping - processes waiting for the resources.

kill command- send a signal to a process.

kill [options] <pid> [...]

The default signal for kill is TERM. Use -l or -L to list available signals. Particularly useful signals include HUP, INT, KILL, STOP, CONT, and 0. Alternate signals may be specified in three ways: -9, -SIGKILL or -KILL. Negative PID values may be used to choose whole process groups; see the PGID column in ps command output. A PID of -1 is special; it indicates all processes except the kill process itself and init.

–$ kill -l   → different options/signal , kill command can send

```
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

$ kill <psid>

$ kill 39769

$ ps -ef | grep xeyes   → find the pid.

```
kali     39750   33808  0 07:27 pts/0    00:00:03 xeyes
kali     39769   33808  0 07:35 pts/0    00:00:02 xeyes
kali     40010   33808  0 08:12 pts/0    00:00:00 grep --color=auto xeyes
```

└─$ kill -9 39750   → send sigkill signal to the process

└─$ ps -ef | grep xeyes

```
kali     39769   33808  0 07:35 pts/0    00:00:02 xeyes
kali     40013   33808  0 08:14 pts/0    00:00:00 grep --color=auto xeyes
[1]-  Killed                  xeyes
```

<u>$ pkill xeyes</u> -→ to kill all the xeyes processes in one go.

NAME

pgrep, pkill, pidwait - look up, signal, or wait for processes based on name and other attributes

SYNOPSIS

pgrep [options] pattern

pkill [options] pattern

pidwait [options] pattern

DESCRIPTION

pgrep looks through the currently running processes and lists the process IDs which match the selection criteria to stdout. All the criteria have to match. For example,

$ pgrep -u root sshd

will only list the processes called sshd AND owned by root. On the other hand,

$ pgrep -u root,daemon

will list the processes owned by root OR daemon.

pkill will send the specified signal (by default SIGTERM) to each process instead of listing them on stdout.

pidwait will wait for each process instead of listing them on stdout.

$ sleep 5 -> delay the script for 5 seconds or wait for 5 seconds.

Scheduling process with cron tab and init.d

Process can be scheduled to execute at a specific time using cron.

There are 2 types of cron tab file

1. Located in the directory /etc/crontab – system level

   $ less /etc/crontab

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# 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
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
```

$ ll /etc/cron.daily

```
total 40
-rwxr-xr-x 1 root root  539 Aug  8  2020 apache2
-rwxr-xr-x 1 root root 1478 Aug 14  2021 apt-compat
-rwxr-xr-x 1 root root  157 Dec 13  2017 debtags
-rwxr-xr-x 1 root root 1298 May 18  2021 dpkg
-rwxr-xr-x 1 root root  377 Aug 16  2021 logrotate
-rwxr-xr-x 1 root root 1123 Feb 19  2021 man-db
-rwxr-xr-x 1 root root 1403 Sep 23  2020 ntp
-rwxr-xr-x 1 root root  652 Dec  7  2020 plocate
```

```
-rwxr-xr-x 1 root root  383 May  6  2021 samba
-rwxr-xr-x 1 root root  518 Feb  2  2021 sysstat
```

$ crontab -e   → to open own crontab file.

Ask for the type of editor, select nano

Edit will open file with below comments

```
Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
```

Explanation, how to create crontab script.

# m h dom mon dow   command

m - minutes – 0 to 59

h → hours – 0 to 23

dom - day of the month → 1 to 31

mon - month → 1 to 12

dow   - day of the week → 0 to 6 → Monday 0, Sunday 6

command – command which we want to run.

$ 5 1 2 * * touch /home/kali/crontab-run.txt

This command will run every 1 hour and 2nd day of the month and every month and every day of the week.

It will run touch command every second day of the month , day can be any and at 01:05 hrs

$ /5 * * * * touch /home/kali/crontab-run.txt   → this will run every 5 minutes.

$ crontab -r   → Delete cron job

$ crontab -l   → Find if any cron job is running

/etc/init.d file → To execute commands while booting the system, we can add command in the file.

```
–$ cd /etc/init.d

┌──(kali㉿kali)-[/etc/init.d]
└─$ ls
apache2             console-setup.sh  hwclock.sh       miredo        plymouth              rsync        snmpd             x11-common
apache-htcacheclean  cron             inetsim          networking    plymouth-log          rsyslog      speech-dispatcher  xl2tpd
apparmor            cryptdisks       iodined          nfs-common    postgresql            rwhod        ssh
atftpd              cryptdisks-early  ipsec            nginx         procps                samba-ad-dc  sslh
avahi-daemon        dbus             keyboard-setup.sh  nmbd          ptunnel               saned        stunnel4
binfmt-support      dns2tcp          kmod             ntp           pulseaudio-enable-autospawn  screen-cleanup  sudo
bluetooth           docker           lightdm          open-vm-tools  redsocks              smartmontools  sysstat
cgroupfs-mount      haveged          mariadb          openvpn       rpcbind               smbd         udev
```