



## Summary

### Session No – 13

- **pgrep** is a command in Linux to check the process ID of the running program

```
[root@localhost ~]# pgrep firefox
[root@localhost ~]# pgrep firefox
2771
[root@localhost ~]#
```

- **ps -aux** command will tell us all the processes running in the system

```
root      2179  0.0  0.0 450028  7152 ?        Ssl  21:03   0:00 /usr/libexec/gvfs-
root      2183  0.0  0.0 448260  6208 ?        Ssl  21:03   0:00 /usr/libexec/gvfs-
root      2186  0.0  0.4 894444 36852 ?        Ssl  21:03   0:00 /usr/libexec/goa-d
root      2196  0.0  0.1 530984  9312 ?        Ssl  21:03   0:00 /usr/libexec/goa-i
root      2202  0.1  0.4 270856 40344 ?        Ss   21:03   0:00 /usr/libexec/sssd/
root      2206  0.2  0.6 1144968 49868 ?        Ssl  21:03   0:00 /usr/libexec/evolu
root      2209  0.0  0.3 2796408 26376 ?        Ssl  21:03   0:00 /usr/bin/gjs /usr/
root      2210  0.0  0.0 161696  7208 ?        Ssl  21:03   0:00 /usr/libexec/at-sp
root      2221  0.0  0.1 522160  8472 ?        Ssl  21:03   0:00 /usr/libexec/gsd-a
root      2222  0.1  0.3 748880 31968 ?        Ssl  21:03   0:00 /usr/libexec/gsd-c
root      2235  0.0  0.2 594360 18652 ?        Ssl  21:03   0:00 /usr/libexec/gsd-d
root      2238  0.0  0.0 524528  7228 ?        Ssl  21:03   0:00 /usr/libexec/gsd-h
root      2254  0.1  0.4 599720 32640 ?        Ssl  21:03   0:00 /usr/libexec/gsd-k
root      2263  0.1  0.4 744160 33428 ?        Ssl  21:03   0:00 /usr/libexec/gsd-m
```

- In one operating system, there are lots of processes running and inside the operating system, there is one main program known as Kernel. The duty of the kernel is to start the process and give the ram
- Process ID at the memory level or physical level is just a directory
- Whenever we run a program it starts the process and behind the scene, the kernel gives every process a unique id which means go to the ram and create one directory, and anything related to the process is stored in that directory
- The entire information of the ram is linked to the **/proc** directory

```
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# cd /proc/
[root@localhost proc]#
```

- In the /proc directory whatever we see in blue colour is a folder or directory and anything in white colour is a file

```
14      21      2267    2767    38      501     735     839      filesystem
15      2102    2273    2771    386     51      737     9        fs
1530    2112    2277    2792    388     519     745     933      interrupt
16      2117    2284    2794    39      52      748     937      iomem
1693    2118    2293    2796    390     520     760     938      ioports
17      2137    23      28      392     521     761     939      irq
18      2143    2311    2808    393     522     763     966      kallsyms
184     2154    232     2809    4       523     764     967      kcore
1854    2161    2321    2812    40      524     765     969      keys
19      2175    2328    2817    404     525     766     970      key-users
1927    2179    2333    2833    405     526     767     acpi     kmsg
1932    2183    2335    29      408     527     768     asound   kpagecgr
1939    2186    2346    2933    41      53      769     bootconf kpagecour
2055    2198    2347    2933    41      53      769     bootconf kpagecour
```

- As soon as we stop the program the process id folder will be deleted from the /proc directory

```
[root@localhost proc]# pgrep firefox
2771
[root@localhost proc]# pgrep firefox
[root@localhost proc]#
```

- This means the PID is just a folder or directory in the ram
- All the information about the process in the RAM we can find inside the process ID folder in the “ / ” proc directory

```
[root@localhost proc]# cd 3442
[root@localhost 3442]#
[root@localhost 3442]# pwd
/proc/3442
[root@localhost 3442]# ls
arch_status      cwd              mem              patch_state      st
attr             environ         mountinfo        personality       st
autogroup        exe             mounts           projid_map       sy
auxv             fd              mountstats       root              sy
cgroup           fdinfo          net              sched             ta
clear_refs       gid_map         ns               schedstat        t:
cmdline          io              numa_maps        sessionid         t:
comm             limits          oom_adj          setgroups         u:
coredump_filter  loginuid        oom_score        smaps             u:
cpu_resctrl_groups map_files       oom_score_adj    smaps_rollup     wo
cpuset           maps            pagemap          stack
```

- **root** file is given to all the processes and it is linked with the “ / ” drive. It means any process has access to / drive

```
[root@localhost 3442]# cd root/
[root@localhost root]# pwd
/proc/3442/root
[root@localhost root]# cd ..
[root@localhost 3442]# pwd
/proc/3442
[root@localhost 3442]# cd root/
[root@localhost root]# pwd
/proc/3442/root
[root@localhost root]# ls
afs  boot  etc  lib  media  opt  root  sbin  sys  usr  vimal
bin  dev  home  lib64  mnt  proc  run  srv  tmp  var
```

- In the command prompt, we can run the commands because the bash shell or bash program is running

```
[root@localhost ~]# echo $SHELL
/bin/bash
[root@localhost ~]#
[root@localhost ~]# dsgfhgg
bash: dsgfhgg: command not found...
```

- We can see the process ID of the bash & we can run the two different processes ie bash process in the bash

```
[root@localhost ~]# pgrep bash
4561
[root@localhost ~]# ps -aux | grep bash
root      4561    0.0   0.0 224112   5780 pts/0    Ss   21:27   0:00 bash
root      4621    0.0   0.0 221668   2264 pts/0    S+   21:28   0:00 gre
auto bash
[root@localhost ~]# bash
[root@localhost ~]# ps -aux | grep bash
root      4561    0.0   0.0 224112   5780 pts/0    Ss   21:27   0:00 bash
root      4626    0.1   0.0 224120   5724 pts/0    S    21:28   0:00 bash
root      4651    0.0   0.0 221668   2256 pts/0    S+   21:28   0:00 gre
auto bash
[root@localhost ~]#
```

- Users can only interact with the process
- Inside one OS we can run multiple processes & it has the capability to launch the process inside the process
- Docker behind the scene launching the process in that we can launch multiple processes that is the reason docker is superfast

```
[root@ip-172-31-46-75 ~]# docker run -dit --name os1 centos:7
2c2d7de439cd6150f15b99279ac27f60bcla208a19e08eeela24273d92cd168e
[root@ip-172-31-46-75 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS   NAMES
2c2d7de439cd   centos:7 "/bin/bash"             4 seconds ago Up 3 seconds   os1
[root@ip-172-31-46-75 ~]#
[root@ip-172-31-46-75 ~]#
[root@ip-172-31-46-75 ~]# ps -aux | grep bash
ec2-user 31901  0.0  0.3 124740 3928 pts/0    Ss   15:24   0:00 -bash
root     31926  0.0  0.4 124868 4204 pts/0    S    15:24   0:00 -bash
root     32392  0.5  0.2 11844 2844 pts/0    Ss+  16:05   0:00 /bin/bash
root     32435  0.0  0.0 119432 936 pts/0    S+   16:06   0:00 grep --color=auto bash
[root@ip-172-31-46-75 ~]#
```

- If we kill the process the container will stop

```
[root@ip-172-31-46-75 ~]# kill -9 32392
[root@ip-172-31-46-75 ~]# ps -aux | grep bash
ec2-user 31901  0.0  0.3 124740 3928 pts/0    Ss   15:24   0:00 -bash
root     31926  0.0  0.4 124868 4204 pts/0    S    15:24   0:00 -bash
root     32484  0.0  0.1 119432 996 pts/0    S+   16:07   0:00 grep --color=auto bash
[root@ip-172-31-46-75 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS   NAMES
2c2d7de439cd   centos:7 "/bin/bash"             About a minute ago Exited (137) 18 seconds ago os1
[root@ip-172-31-46-75 ~]#
[root@ip-172-31-46-75 ~]# docker ps -
"docker ps" accepts no arguments.
See 'docker ps --help'.

Usage: docker ps [OPTIONS]

List containers
[root@ip-172-31-46-75 ~]# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS   NAMES
2c2d7de439cd   centos:7 "/bin/bash"             About a minute ago Exited (137) 18 seconds ago os1
[root@ip-172-31-46-75 ~]#
```

- If we stop the container behind the scene, it is stopping the process only

```
[root@ip-172-31-46-75 ~]# docker run -dit --name os2 centos:7
4bf7955c28dda3ce274ca6068dd4184887bdfef8149fc19d4b22ec681cda99
[root@ip-172-31-46-75 ~]#
[root@ip-172-31-46-75 ~]#
[root@ip-172-31-46-75 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS   NAMES
4bf7955c28dd   centos:7  "/bin/bash"             2 seconds ago Up 2 seconds   os2
[root@ip-172-31-46-75 ~]# ps -aux | grep bash
ec2-user 31901  0.0  0.3 124740 3928 pts/0    Ss   15:24   0:00 -bash
root     31926  0.0  0.4 124868 4204 pts/0    S    15:24   0:00 -bash
root     32547  0.9  0.2 11844  2916 pts/0    Ss+  16:07   0:00 /bin/bash
root     32591  0.0  0.0 119432  976 pts/0    S+   16:08   0:00 grep --color=auto bash
[root@ip-172-31-46-75 ~]# docker stop os2

os2
[root@ip-172-31-46-75 ~]#
[root@ip-172-31-46-75 ~]# ps -aux | grep bash
ec2-user 31901  0.0  0.3 124740 3928 pts/0    Ss   15:24   0:00 -bash
root     31926  0.0  0.4 124868 4204 pts/0    S    15:24   0:00 -bash
root     32644  0.0  0.0 119432  940 pts/0    S+   16:08   0:00 grep --color=auto bash
[root@ip-172-31-46-75 ~]#
```

- Whenever we launch the container in interactive mode docker gives us the terminal because behind the scene it is running the bash shell

```
[root@ip-172-31-46-75 ~]# docker run -it --name os2 centos:7
[root@9040f955df11 /]#
[root@9040f955df11 /]#
[root@9040f955df11 /]# echo $SHELL
/bin/bash
[root@9040f955df11 /]# gg
bash: gg: command not found
[root@9040f955df11 /]#
```

- In the image with the help of the CMD or ENTRYPOINT keyword, we can decide which process to run & the entire container will be the process

```
[root@ip-172-31-46-75 ~]# docker history centos:7
IMAGE          CREATED          CREATED BY          SIZE      COMMENT
eeb6ee3f44bd   14 months ago   /bin/sh -c # (nop)  CMD ["/bin/bash"]   0B
<missing>      14 months ago   /bin/sh -c # (nop)  LABEL org.label-schema.sc... 0B
<missing>      14 months ago   /bin/sh -c # (nop)  ADD file:b3ebbe8bd304723d4... 204MB
[root@ip-172-31-46-75 ~]#
```

- Kernel version 2.6 and above supports nested process concept

```
[root@ip-172-31-46-75 ~]# uname -r
5.10.144-127.601.amzn2.x86_64
[root@ip-172-31-46-75 ~]#
```

- Whenever we start the process, every process has a unique id and a unique folder known as root which is linked to the / drive



- In the container, the root directory is linked with the “ / ” drive and the data in the / drive is coming from the image, not from the base system

```
[root@ip-172-31-46-75 /]# cd /proc/667/root/
[root@ip-172-31-46-75 root]# ls
anaconda-post.log  dev  hi.txt  lib  media  opt  root  sbin  sys  usr
bin                etc  home   lib64  mnt  proc  run  srv  var
[root@ip-172-31-46-75 root]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
45a41f9a2f3d   centos:7  "/bin/bash"   3 minutes ago   Up 3 minutes           os1
[root@ip-172-31-46-75 root]# docker attach os1
[root@45a41f9a2f3d /]# cd
[root@45a41f9a2f3d ~]# cd /
[root@45a41f9a2f3d /]# ls
anaconda-post.log  dev  hi.txt  lib  media  opt  root  sbin  sys  usr
bin                etc  home   lib64  mnt  proc  run  srv  var
[root@45a41f9a2f3d /]# cat hi.txt
hiiii
[root@45a41f9a2f3d /]#
```

- Data in / drive is coming from the image that is the reason if a container is running & we tried to remove the container, it gives the error

```
[root@ip-172-31-46-75 root]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
45a41f9a2f3d   centos:7  "/bin/bash"   6 minutes ago   Up 6 minutes           os1
[root@ip-172-31-46-75 root]# docker rmi centos:7
Error response from daemon: conflict: unable to remove repository reference "centos:7" (must force) - container 45a41f9a2f3d is using its referenced image eeb6ee3f44bd
[root@ip-172-31-46-75 root]#
```