## PRACTICAL NO: 4

**AIM:**

1. Adam is working in an IT company. He has been given a task to reduce the load of a system by killing some of the processes running in the LINUX operating system. Which commands will he use to complete the given task with the help of the following operation?

- Kill Processes by name
- Kill a process based on the process name
- Kill a single process at a time with the given process lD

2.Write a program for process creation using C

- Orphan Process
- Zombie Process

3. Create the process using fork () system call

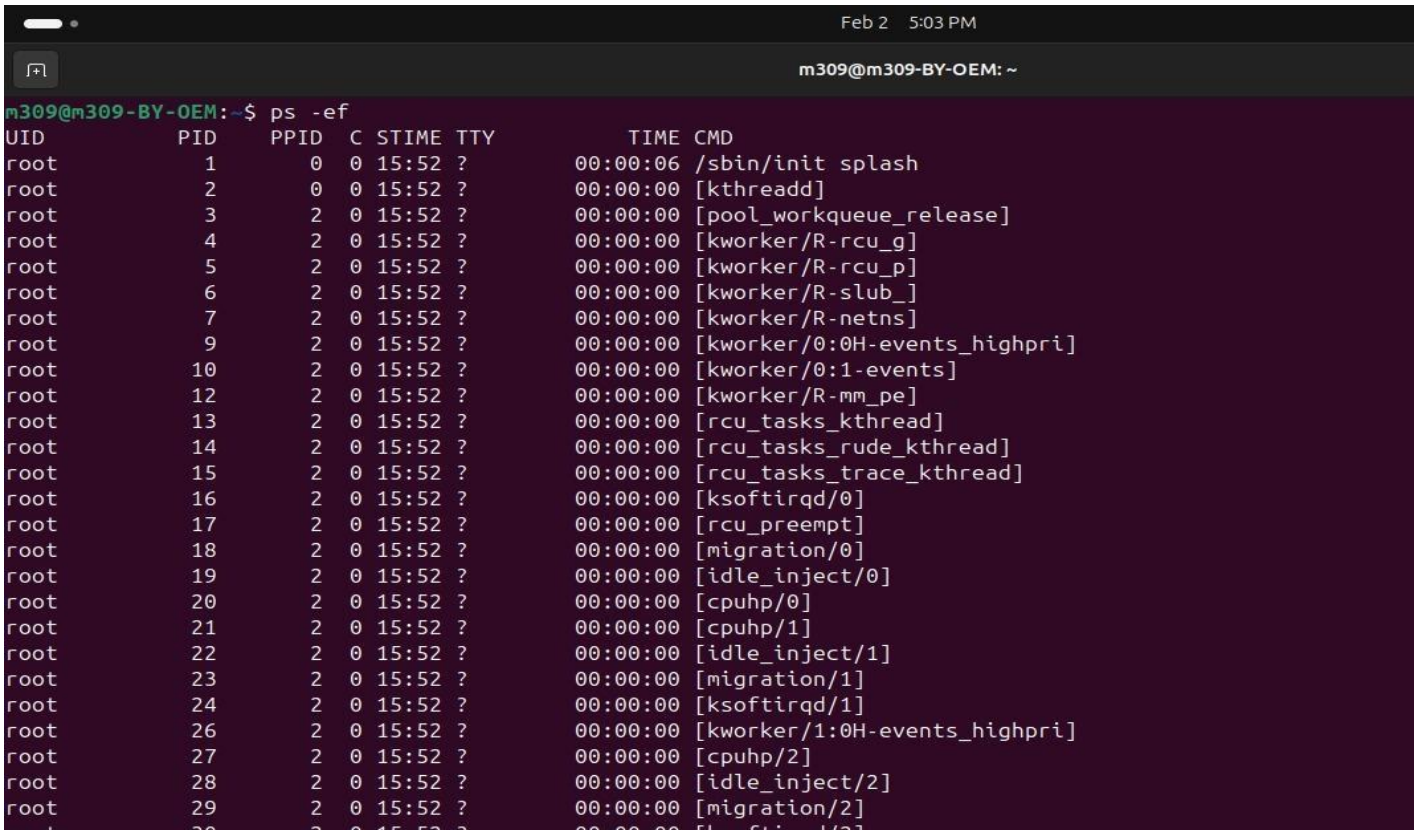
- Child Process creation
- Parent Process creation
- PPID and PID

**PERFORMANCE**

**1.**

- Kill Processes by name
- Kill a process based on the process name
- Kill a single process at a time with the given process ID

**COMMAND:**


Feb 2 5:03 PM
m309@m309-BY-OEM: ~
m309@m309-BY-OEM:~$ ps -ef
UID PID PPID C STIME TTY TIME CMD
root 1 0 0 15:52 ? 00:00:06 /sbin/init splash
root 2 0 0 15:52 ? 00:00:00 [kthreadd]
root 3 2 0 15:52 ? 00:00:00 [pool_workqueue_release]
root 4 2 0 15:52 ? 00:00:00 [kworker/R-rcu_g]
root 5 2 0 15:52 ? 00:00:00 [kworker/R-rcu_p]
root 6 2 0 15:52 ? 00:00:00 [kworker/R-slub_]
root 7 2 0 15:52 ? 00:00:00 [kworker/R-netns]
root 9 2 0 15:52 ? 00:00:00 [kworker/0:0H-events_highpri]
root 10 2 0 15:52 ? 00:00:00 [kworker/0:1-events]
root 12 2 0 15:52 ? 00:00:00 [kworker/R-mm_pe]
root 13 2 0 15:52 ? 00:00:00 [rcu_tasks_kthread]
root 14 2 0 15:52 ? 00:00:00 [rcu_tasks_rude_kthread]
root 15 2 0 15:52 ? 00:00:00 [rcu_tasks_trace_kthread]
root 16 2 0 15:52 ? 00:00:00 [ksoftirqd/0]
root 17 2 0 15:52 ? 00:00:00 [rcu_preempt]
root 18 2 0 15:52 ? 00:00:00 [migration/0]
root 19 2 0 15:52 ? 00:00:00 [idle_inject/0]
root 20 2 0 15:52 ? 00:00:00 [cpuhp/0]
root 21 2 0 15:52 ? 00:00:00 [cpuhp/1]
root 22 2 0 15:52 ? 00:00:00 [idle_inject/1]
root 23 2 0 15:52 ? 00:00:00 [migration/1]
root 24 2 0 15:52 ? 00:00:00 [ksoftirqd/1]
root 26 2 0 15:52 ? 00:00:00 [kworker/1:0H-events_highpri]
root 27 2 0 15:52 ? 00:00:00 [cpuhp/2]
root 28 2 0 15:52 ? 00:00:00 [idle_inject/2]
root 29 2 0 15:52 ? 00:00:00 [migration/2]


m309@m309-BY-OEM:~$ ps -ef |grep firefox
m309 9393 9361 0 17:02 pts/0 00:00:00 grep --color=auto firefox
m309@m309-BY-OEM:~$ ps -ef | grep firefox
m309 9407 9361 0 17:03 pts/0 00:00:00 grep --color=auto firefox
m309@m309-BY-OEM:~$ kill 9407
bash: kill: (9407) - No such process


bash: kill: (9407) - No such process
m309@m309-BY-OEM:~$ pkill firefox

## 2. Write a program for process creation using C

- Orphan Process:

  - orphan.c :

```
m309@m309-BY-OEM:~$ nano orphan.c
```

```
  GNU nano 7.2                                                  orphan.c
#include <stdio.h>
#include <unistd.h>

int main() {
    if (fork() == 0) {
        sleep(5);
        printf("Child PID: %d\n", getpid());
        printf("New Parent PID: %d\n", getppid());
    } else {
        printf("Parent exiting\n");
    }
    return 0;
}
```

  - Output:

```
m309@m309-BY-OEM:~$ nano orphan.c
m309@m309-BY-OEM:~$ gcc orphan.c -o orphan
m309@m309-BY-OEM:~$ ./orphan
Parent exiting
m309@m309-BY-OEM:~$ Child PID: 9506
New Parent PID: 6892
```

- Zombie Process
  - Zombie.c :

```
m309@m309-BY-OEM:~$ nano zombie3.c
```

```
 GNU nano 7.2                                  zombie3.c
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid = fork();

    if (pid > 0) {
        sleep(10);   // Parent wait nahi kar raha
        printf("Parent process\n");
    } else {
        printf("Child process exiting\n");
    }
    return 0;
}
```

  - Output:

```
m309@m309-BY-OEM:~$ nano zombie3.c
m309@m309-BY-OEM:~$ gcc zombie3.c -o zombie3
m309@m309-BY-OEM:~$ ./zombie3
Child process exiting
Parent process
```

### 3. Create the process using fork () system call

- Child Process creation
- Parent Process creation
- PPID and PID

```
m309@m309-BY-OEM:~$ nano fork_demo.c
```

- fork_demo:

```
 GNU nano 7.2                                  fork_demo.c
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid = fork();

    if (pid == 0) {
        printf("Child Process\n");
        printf("Child PID: %d\n", getpid());
        printf("Parent PID: %d\n", getppid());
    } else {
        printf("Parent Process\n");
        printf("Parent PID: %d\n", getpid());
    }
    return 0;
}
```

- Output:

```
m309@m309-BY-OEM:~$ nano fork_demo.c
m309@m309-BY-OEM:~$ gcc fork_demo.c -o fork_demo
m309@m309-BY-OEM:~$ ./fork_demo
Parent Process
Parent PID: 9465
Child Process
Child PID: 9466
Parent PID: 9465
```

## 4. Infinite loop process:

```
m309@m309-BY-OEM:~$ nano loop.c
m309@m309-BY-OEM:~$ gcc loop.c -o loop
m309@m309-BY-OEM:~$ ./loop
Running...
Running...
Running...
Running...
Running...
Running...
```

```
 GNU nano 7.2                                  loop.c *
#include <stdio.h>
#include <unistd.h>

int main() {
    while (1) {
        printf("Running...\n");
        sleep(1);
    }
    return 0;
}
```

- Stopped the Loop:

```
Running...
Running...
Running...
Running...
Running...
Running...
qRunning...
Running...
Running...
Running...
Running...
Running...
^[tRunning...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
```

```
m309@m309-BY-OEM: ~
m309@m309-BY-OEM:~$ q
Command 'q' not found, but can be installed with:
sudo snap install q                       # version 1.6.3-1, or
sudo apt  install python3-q-text-as-data  # version 3.1.6-3
See 'snap info q' for additional versions.
m309@m309-BY-OEM:~$
```