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?

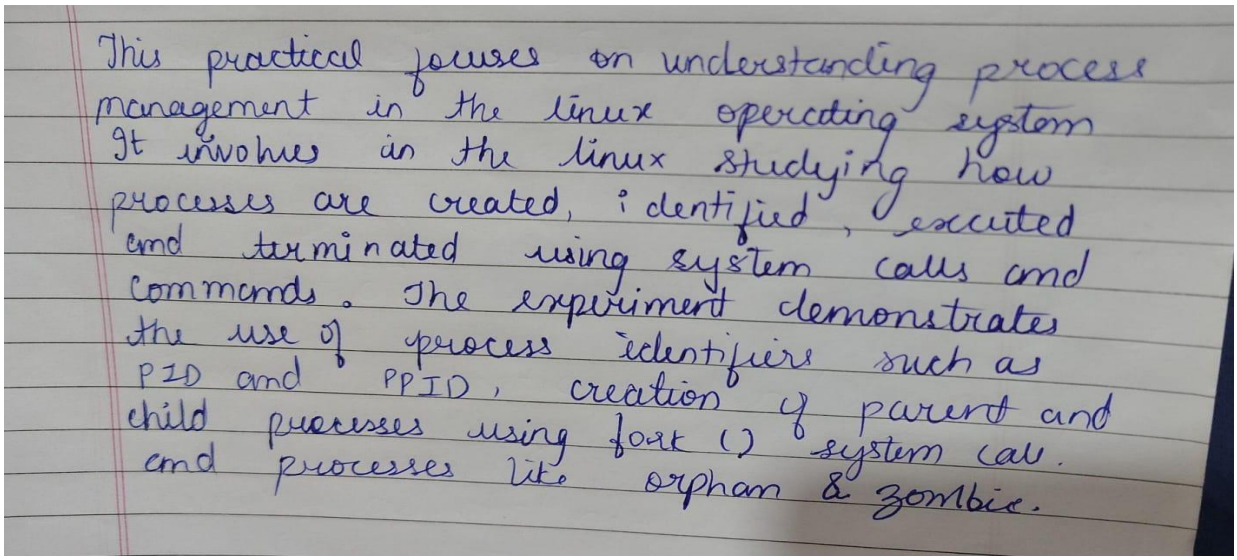
- Kill Processes by name
- Kill a process based on the process name
- Kill a single process at a time with the given process lD

2.Write a program for process creation using C

- Orphan Process
- Zombie Process

3. Create the process using fork () system call

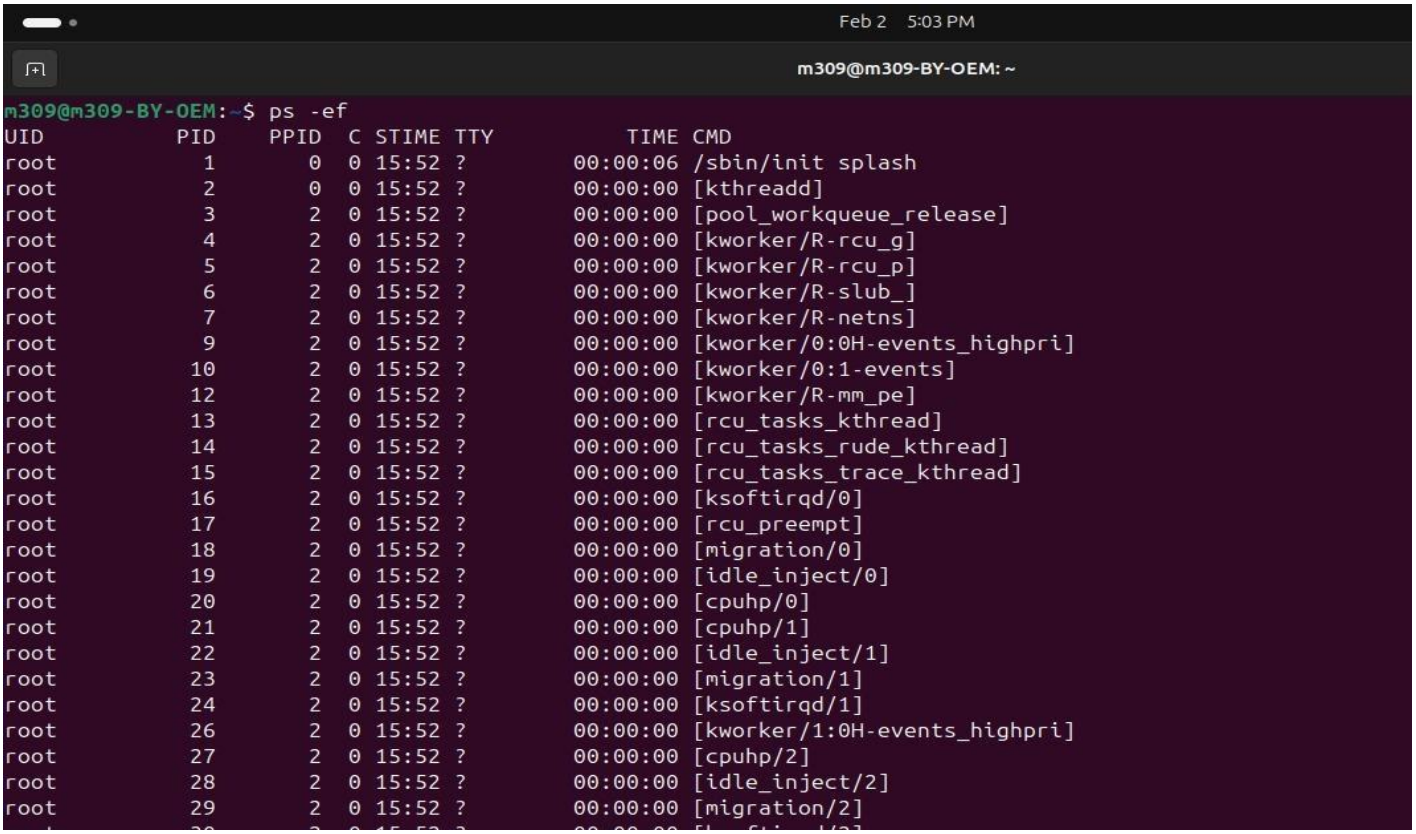
- Child Process creation
- Parent Process creation
- PPID and PID THEORY:

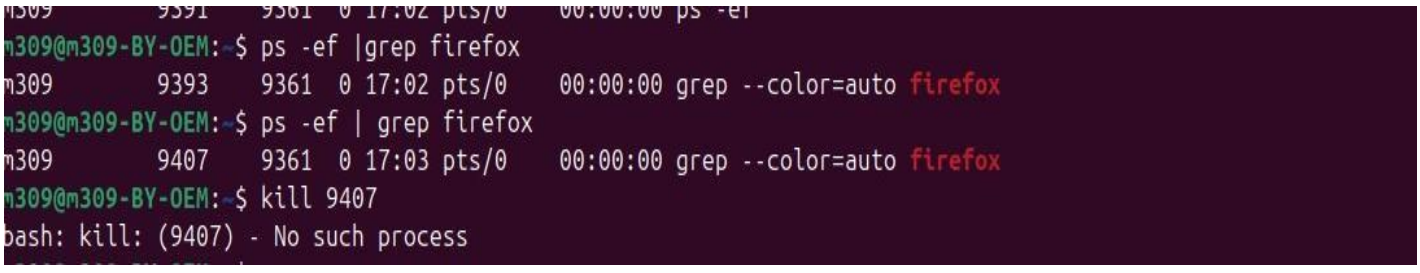


PERFORMANCE

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

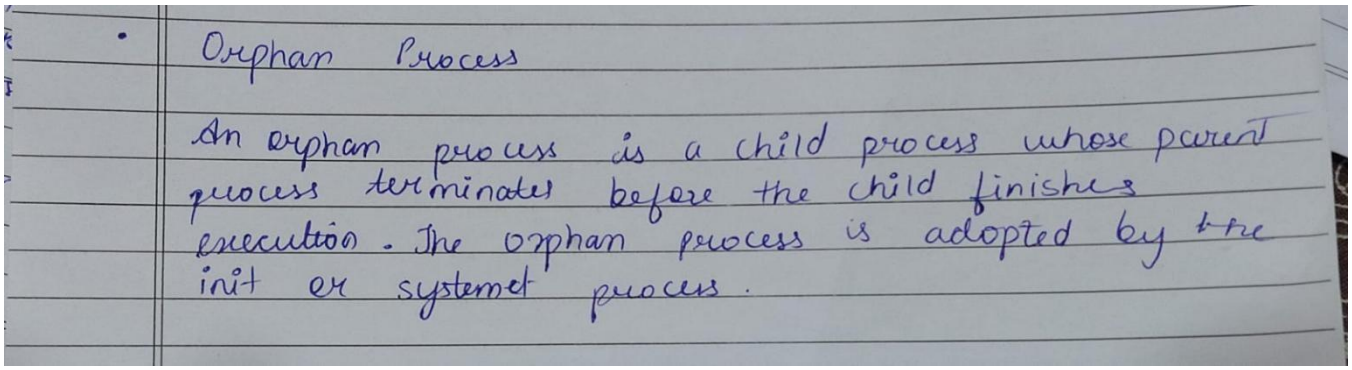
COMMAND:





2.Write a program for process creation using C

- Orphan Process:



- orphan.c :

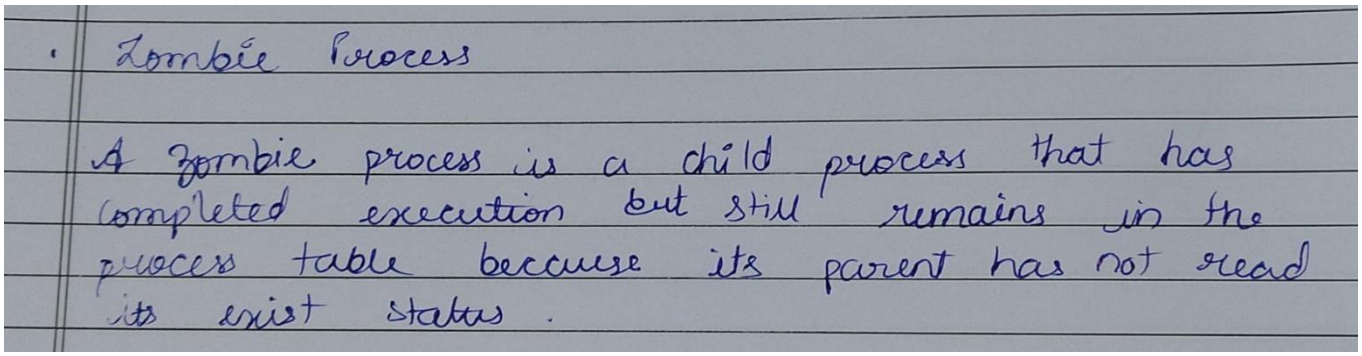
```
m309@m309-BY-OEM:~$ nano orphan.c
```

```
 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
```

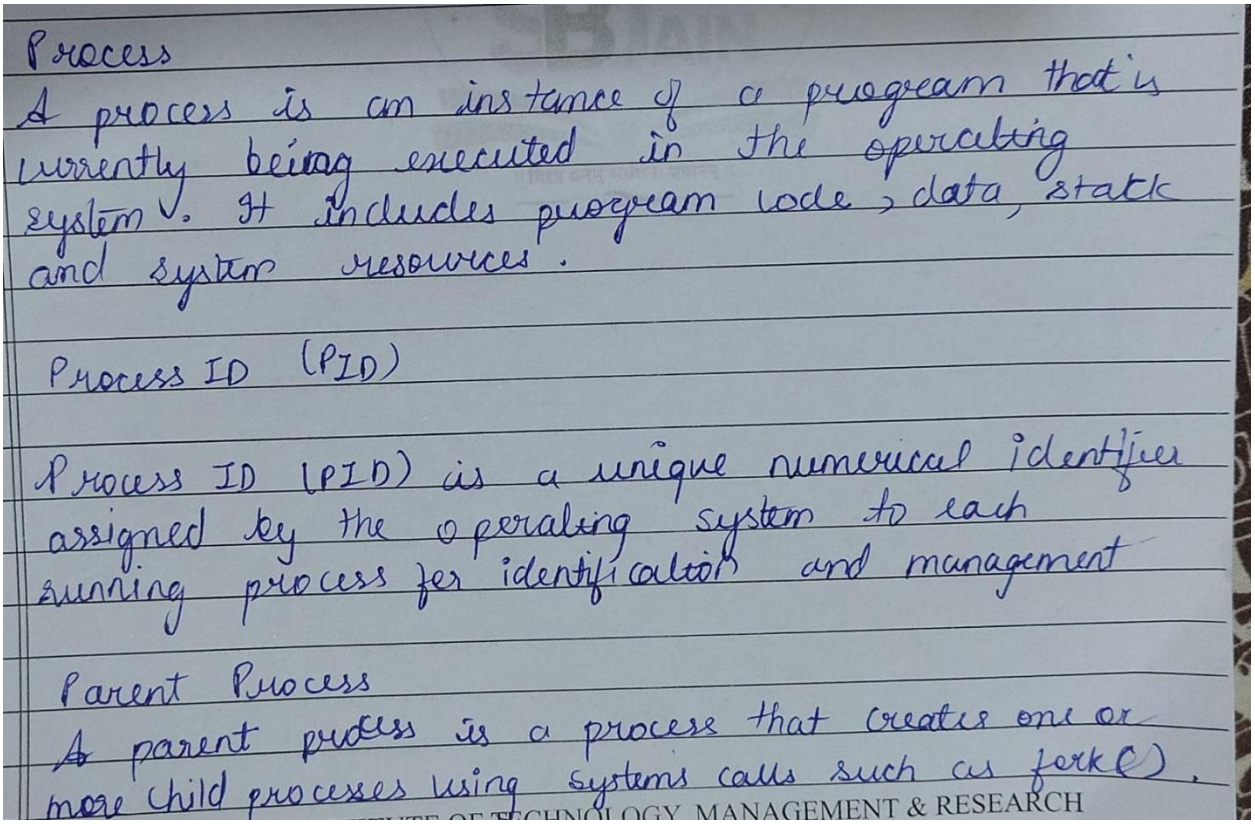
```
 GNU nano 7.2                                  zombie3.c
#include <stdio.h>
#include <unistd.h>

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



Process

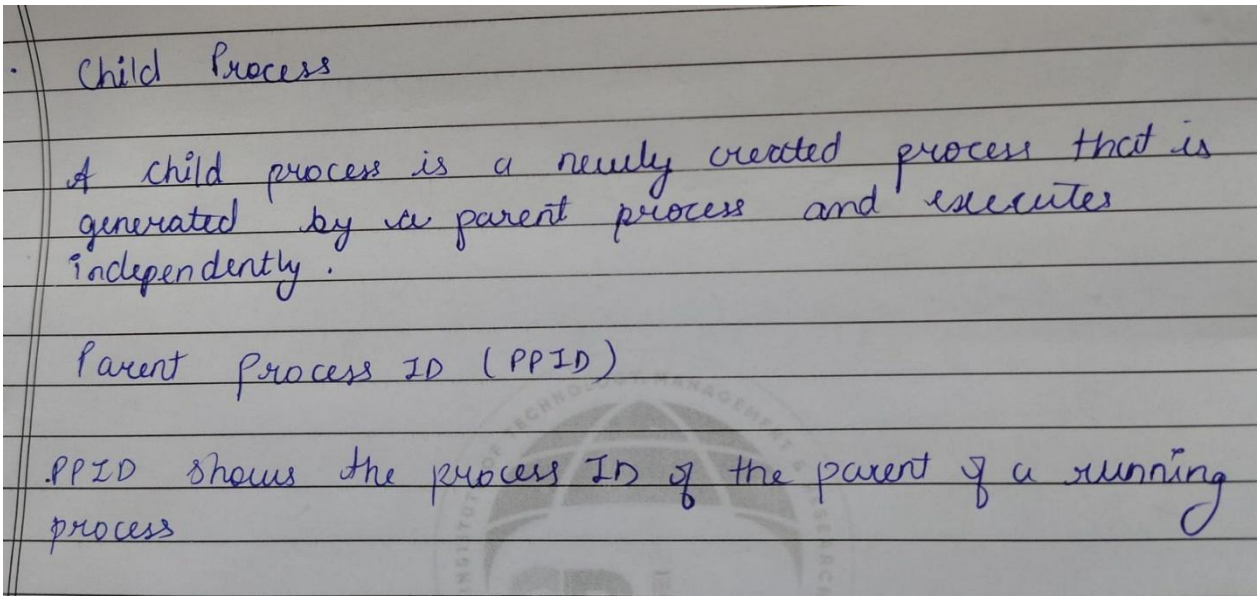
A process is an instance of a program that is currently being executed in the operating system. It includes program code, data, stack and system resources.

Process ID (PID)

Process ID (PID) is a unique numerical identifier assigned by the operating system to each running process for identification and management

Parent Process

A parent process is a process that creates one or more child processes using systems calls such as fork().



Child Process

A child process is a newly created process that is generated by a parent process and executes independently.

Parent Process ID (PPID)

PPID shows the process ID of the parent of a running process

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

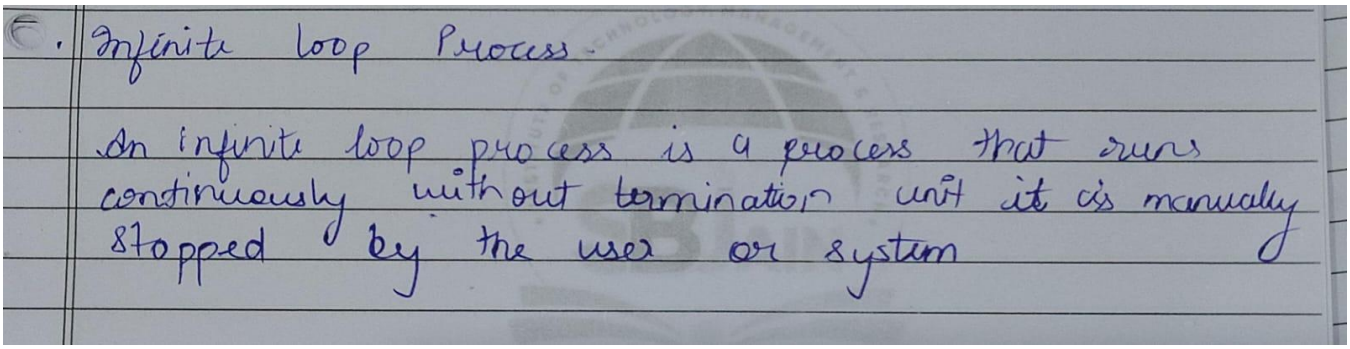
- fork_demo:

```
 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 infinite loop: