

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 ID

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:

This practical focuses on understanding process management in the Linux operating system. A process is a program in execution, and Linux manages multiple processes through multitasking. Processes are created using the fork() system call, which forms parent and child processes. Each process is identified using PID and PPID to maintain hierarchy. The experiment also demonstrates process execution, termination, and the use of system calls like exec() and wait(). It further explains special cases such as orphan and zombie processes in Linux

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:

```
akshad123@LAPTOP-ABSVSLT x + v
akshad123@LAPTOP-ABSVSLTV:~$ ps -ef
UID      PID     PPID  C  STIME TTY          TIME CMD
root         1         0  4  14:32 ?        00:00:00 /sbin/init
root         2         1  0  14:32 ?        00:00:00 /init
root         7         2  0  14:32 ?        00:00:00 plan9 --control-socket 7 --log-level 4 --server-fd 8 --pipe-fd 10 --
root        64         1  1  14:32 ?        00:00:00 /lib/systemd/systemd-journald
root        90         1  1  14:32 ?        00:00:00 /lib/systemd/systemd-udev
systemd+   93         1  0  14:32 ?        00:00:00 /lib/systemd/systemd-resolved
systemd+   94         1  0  14:32 ?        00:00:00 /lib/systemd/systemd-timesyncd
root       154         1  0  14:32 ?        00:00:00 /usr/sbin/cron -f -P
message+  157         1  0  14:32 ?        00:00:00 @dbus-daemon --system --address=systemd: --nofork --nopidfile --syst
root       196         1  1  14:32 ?        00:00:00 /usr/bin/python3 /usr/bin/networkd-dispatcher --run-startup-triggers
syslog    197         1  0  14:32 ?        00:00:00 /usr/sbin/rsyslogd -n -iNONE
root       199         1  1  14:32 ?        00:00:00 /usr/lib/napd/napd
root       200         1  0  14:32 ?        00:00:00 /lib/systemd/systemd-logind
root       225         1  0  14:32 hvc0     00:00:00 /sbin/agetty -o -p -- \u --noclear --keep-baud console 115200,38400,
root       227         1  0  14:32 tty1     00:00:00 /sbin/agetty -o -p -- \u --noclear tty1 linux
root       235         1  0  14:32 ?        00:00:00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-s
root       269         1  0  14:32 ?        00:00:00 /lib/systemd/systemd-timedated
root       283         2  0  14:32 ?        00:00:00 /init
root       285        283  0  14:32 ?        00:00:00 /init
akshad1+  290        285  0  14:32 pts/0    00:00:00 -bash
root       291         2  0  14:32 pts/1    00:00:00 /bin/login -f
akshad1+  350         1  1  14:32 ?        00:00:00 /lib/systemd/systemd --user
akshad1+  351        350  0  14:32 ?        00:00:00 (sd-pam)
akshad1+  365        291  0  14:32 pts/1    00:00:00 -bash
root       394         90  0  14:32 ?        00:00:00 /lib/systemd/systemd-udev
root       395         90  0  14:32 ?        00:00:00 /lib/systemd/systemd-udev
akshad1+  396        290  0  14:32 pts/0    00:00:00 ps -ef
akshad123@LAPTOP-ABSVSLTV:~$
```

```
akshad123@LAPTOP-ABSVSLTV:~$ ps -ef | grep firefox
akshad1+    861    290  0 15:02 pts/0    00:00:00 grep --color=auto firefox
akshad123@LAPTOP-ABSVSLTV:~$ ps -ef | grep firefox
akshad1+    871    290  0 15:03 pts/0    00:00:00 grep --color=auto firefox
```

```
akshad123@LAPTOP-ABSVSLTV:~$ kill 9407
-bash: kill: (9407) - No such process
akshad123@LAPTOP-ABSVSLTV:~$ pkill firefox
```

2. Write a program for process creation using C

Orphan Process:

An orphan process is a child process whose parent process terminates before the child finishes execution. The orphan process is adopted by the init or system process.

➤ orphan.c :

➤ output:

```
akshad123@LAPTOP-ABSVSLT  ×  +  ▾
akshad123@LAPTOP-ABSVSLTV:~$ nano orphan.c
akshad123@LAPTOP-ABSVSLTV:~$ gcc orphan.c -o orphan
akshad123@LAPTOP-ABSVSLTV:~$ ./orphan
Parent exiting...
akshad123@LAPTOP-ABSVSLTV:~$ Child Process
PID  = 1025
PPID = 285 (Parent is init)
```

```
GNU nano 6.2 orphan.c *
#include <stdio.h>
#include <unistd.h>

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

    if (pid == 0) {
        // Child
        sleep(5);
        printf("Child Process\n");
        printf("PID = %d\n", getpid());
        printf("PPID = %d (Parent is init)\n", getppid());
    } else {
        // Parent
        printf("Parent exiting...\n");
    }

    return 0;
}
```

- **Zombie Process**

A zombie process is a child process that has completed execution but still remains in the process table because its parent has not read its exit status.

➤ **Zombie.c :**

➤ **Output:**

```
akshad123@LAPTOP-ABSVSLT  x  +  v
akshad123@LAPTOP-ABSVSLTV:~$ nano zombie3.c
akshad123@LAPTOP-ABSVSLTV:~$ gcc zombie3.c -o zombie3
akshad123@LAPTOP-ABSVSLTV:~$ ./zombie3
Child exiting...
Parent still running...
akshad123@LAPTOP-ABSVSLTV:~$ |
```

```
akshad123@LAPTOP-ABSVSLT  x  +  v
GNU nano 6.2                zombie3.c
#include <stdio.h>
#include <unistd.h>

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

    if (pid == 0) {
        // Child
        printf("Child exiting...\n");
    } else {
        // Parent
        sleep(10); // Parent does not call wait()
        printf("Parent still running...\n");
    }

    return 0;
}
```

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 system 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.

- fork_demo:
- Output:

```
akshad123@LAPTOP-ABSVSLT x + v
akshad123@LAPTOP-ABSVSLTV:~$ nano fork.c
akshad123@LAPTOP-ABSVSLTV:~$ gcc fork.c -o fork
akshad123@LAPTOP-ABSVSLTV:~$ ./fork
Parent Process
PID = 1178
Child PID = 1179
Child Process
PID = 1179
PPID = 285
akshad123@LAPTOP-ABSVSLTV:~$ |
```

```
akshad123@LAPTOP-ABSVSLT x + v
GNU nano 6.2 fork.c
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid;

    pid = fork();

    if (pid < 0) {
        printf("Fork failed\n");
    }
    else if (pid == 0) {
        // Child process
        printf("Child Process\n");
        printf("PID = %d\n", getpid());
        printf("PPID = %d\n", getppid());
    }
    else {
        // Parent process
        printf("Parent Process\n");
        printf("PID = %d\n", getpid());
        printf("Child PID = %d\n", pid);
    }

    return 0;
}
```

3. Infinite loop process:

An infinite loop process is a process that runs continuously without termination until it is manually stopped by the user or system.

```
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:

[illegible]

```
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:~$
```

