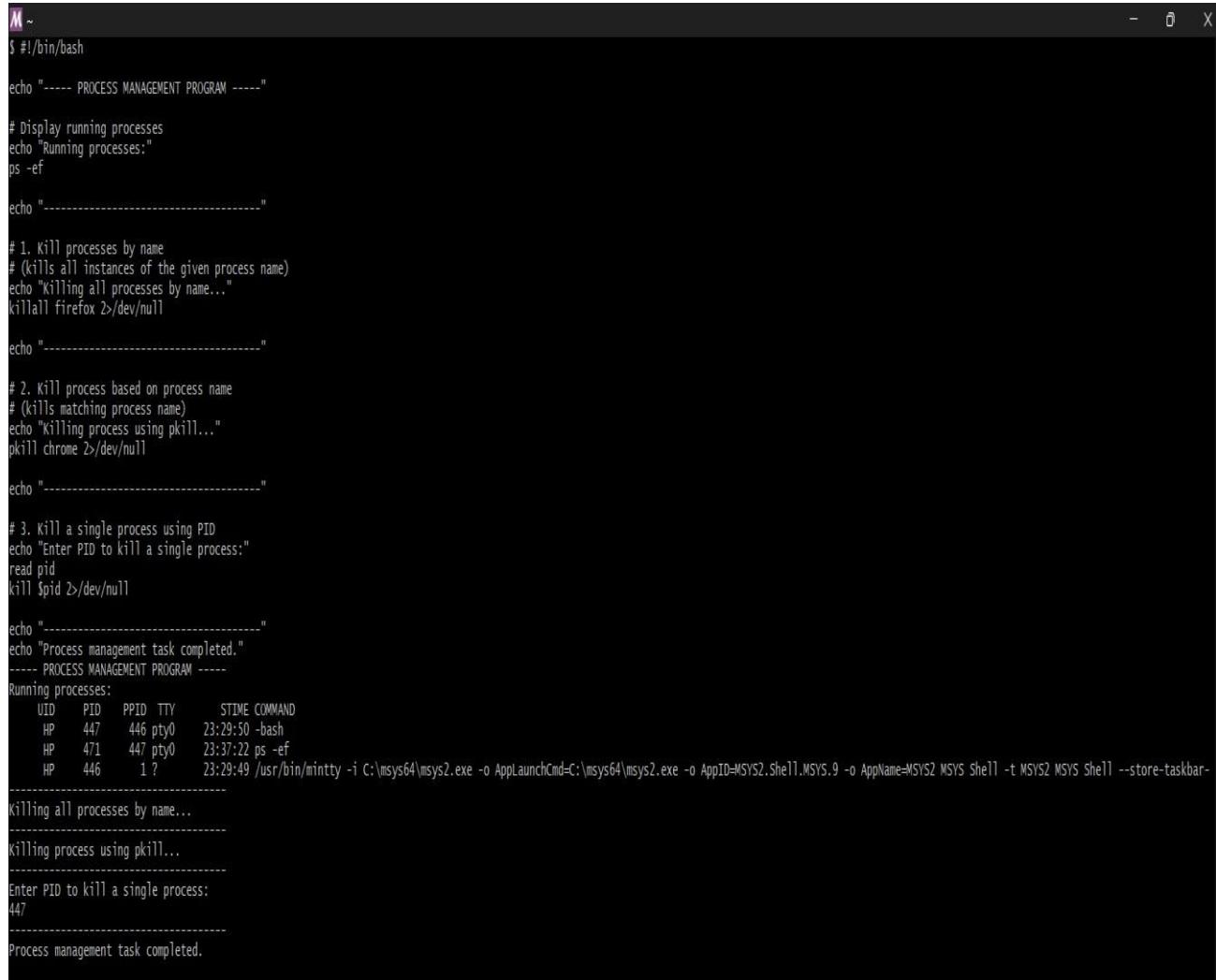


1. Adam is working in an IT company. He has been given a task to reduce the load of a system by killing some of the processes running in the LINUX operating system. Which commands will he use to complete the given task with the help of the following operation?

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



The screenshot shows a terminal window with a black background and white text. The window title is 'M ~'. The terminal displays a bash script for process management. The script includes comments and code for displaying running processes, killing all processes by name (using killall), killing processes by name (using pkill), and killing a single process by PID. It also includes sections for entering a PID and a final message. The output shows the running processes before each action and the results of the actions.

```
$ #!/bin/bash
echo "----- PROCESS MANAGEMENT PROGRAM -----"
# Display running processes
echo "Running processes:"
ps -ef
echo "-----"
# 1. Kill processes by name
# (kills all instances of the given process name)
echo "Killing all processes by name..."
killall firefox >/dev/null
echo "-----"
# 2. Kill process based on process name
# (kills matching process name)
echo "Killing process using pkill..."
pkill chrome >/dev/null
echo "-----"
# 3. Kill a single process using PID
echo "Enter PID to kill a single process:"
read pid
kill $pid >/dev/null
echo "-----"
echo "Process management task completed."
----- PROCESS MANAGEMENT PROGRAM -----
Running processes:
UID PID PPID TTY      STIME COMMAND
HP 447 446 pty0    23:29:50 -bash
HP 471 447 pty0    23:37:22 ps -ef
HP 446 1 ?        23:29:49 /usr/bin/mintty -i C:\msys64\msys2.exe -o AppLaunchCmd=C:\msys64\msys2.exe -o AppID=MSYS2_shell.MSYS.9 -oAppName=MSYS2 MSYS shell -t MSYS2 MSYS shell --store-taskbar-
----- Killing all processes by name...
----- Killing process using pkill...
Enter PID to kill a single process:
447
----- Process management task completed.
```

2. Write a program for process creation using C

- Orphan Process

**INPUT:**

```
GNU nano 8.7
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

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

    if (pid < 0) {
        printf("Fork failed\n");
    }
    else if (pid == 0) {
        sleep(5); // Child waits so parent terminates
        printf("\nChild Process (Orphan)\n");
        printf("PID = %d\n", getpid());
        printf("PPID = %d\n", getppid()); // will change after parent exits
    }
    else {
        printf("Parent Process exiting...\n");
    }

    return 0;
}
```

**OUTPUT :**

```
HP@LAPTOP-9M3IBDT5 MSYS ~
$ ./orphan
Parent Process exiting...

HP@LAPTOP-9M3IBDT5 MSYS ~
$
Child Process (Orphan)
PID = 1131
PPID = 1
```

- Zombie Process

**INPUT :**

```
M ~
GNU nano 8.7                               zombie.c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

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

    if (pid > 0)
    {
        printf("Parent Process ID: %d\n", getpid());
        sleep(10);
        printf("Parent exiting...\n");
    }
    else if (pid == 0)
    {
        printf("Child Process ID: %d\n", getpid());
        printf("Child exiting...\n");
    }
    else
    {
        printf("Fork failed\n");
    }

    return 0;
}
```

## OUTPUT :

```
HP@Hiteshri MSYS ~
$ nano zombie.c

HP@Hiteshri MSYS ~
$ gcc zombie.c -o zombie

HP@Hiteshri MSYS ~
$ ./zombie
Parent Process ID: 378
Child Process ID: 379
Child exiting...
Parent exiting...

HP@Hiteshri MSYS ~
$ |
```

3. Create the process using fork () system call.
  - Child Process creation
  - Parent process creation PPID and PID

## INPUT :

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

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

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

    return 0;
}
```

## OUTPUT :

```
M ~

HP@Hiteshri MSYS ~
$ nano fork.c

HP@Hiteshri MSYS ~
$ gcc fork.c -o fork

HP@Hiteshri MSYS ~
$ ./fork
Parent Process
PID : 388
Child PID : 389
Child Process
PID : 389
Parent PID (PPID) : 388

HP@Hiteshri MSYS ~
$ |
```