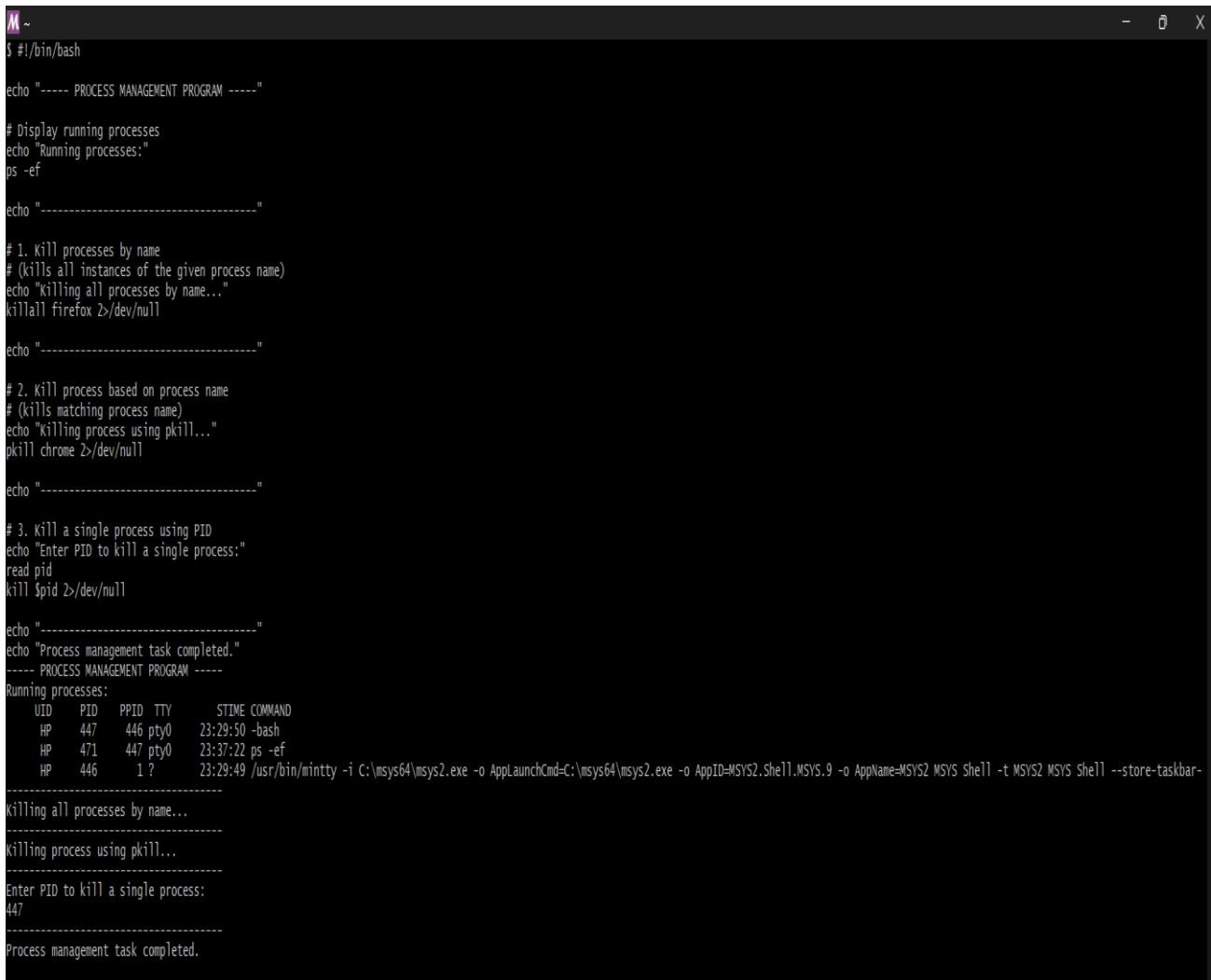


1. Adam is working in an IT company. He has been given a task to reduce the load of a system by killing some of the processes running in the LINUX operating system. Which commands will he use to complete the given task with the help of the following operation?

- 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, echo statements, and command executions. It handles three cases: killing all processes by name, killing processes based on name using pkill, and killing a single process by PID. The output shows the execution of killall firefox, pkill chrome, and kill 447, followed by a confirmation message.

```
$#!/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	S TIME	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:

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

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

    if (pid > 0)
    {
        printf("Parent Process ID: %d\n", getpid());
        printf("Parent exiting...\n");
    }
    else if (pid == 0)
    {
        sleep(5);
        printf("Child Process ID: %d\n", getpid());
        printf("New Parent Process ID (init): %d\n", getppid());
    }
    else
    {
        printf("Fork failed\n");
    }
    return 0;
}
```

OUTPUT:

```
M ~
HP@Hiteshri MSYS ~
$ nano orphan.c

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

HP@Hiteshri MSYS ~
$ ./orphan
Parent Process ID: 368
Parent exiting...

HP@Hiteshri MSYS ~
$ Child Process ID: 369
New Parent Process ID (init): 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 :

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

int main()
{
    pid_t pid;

    pid = fork();

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

    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 ~
$ |
```