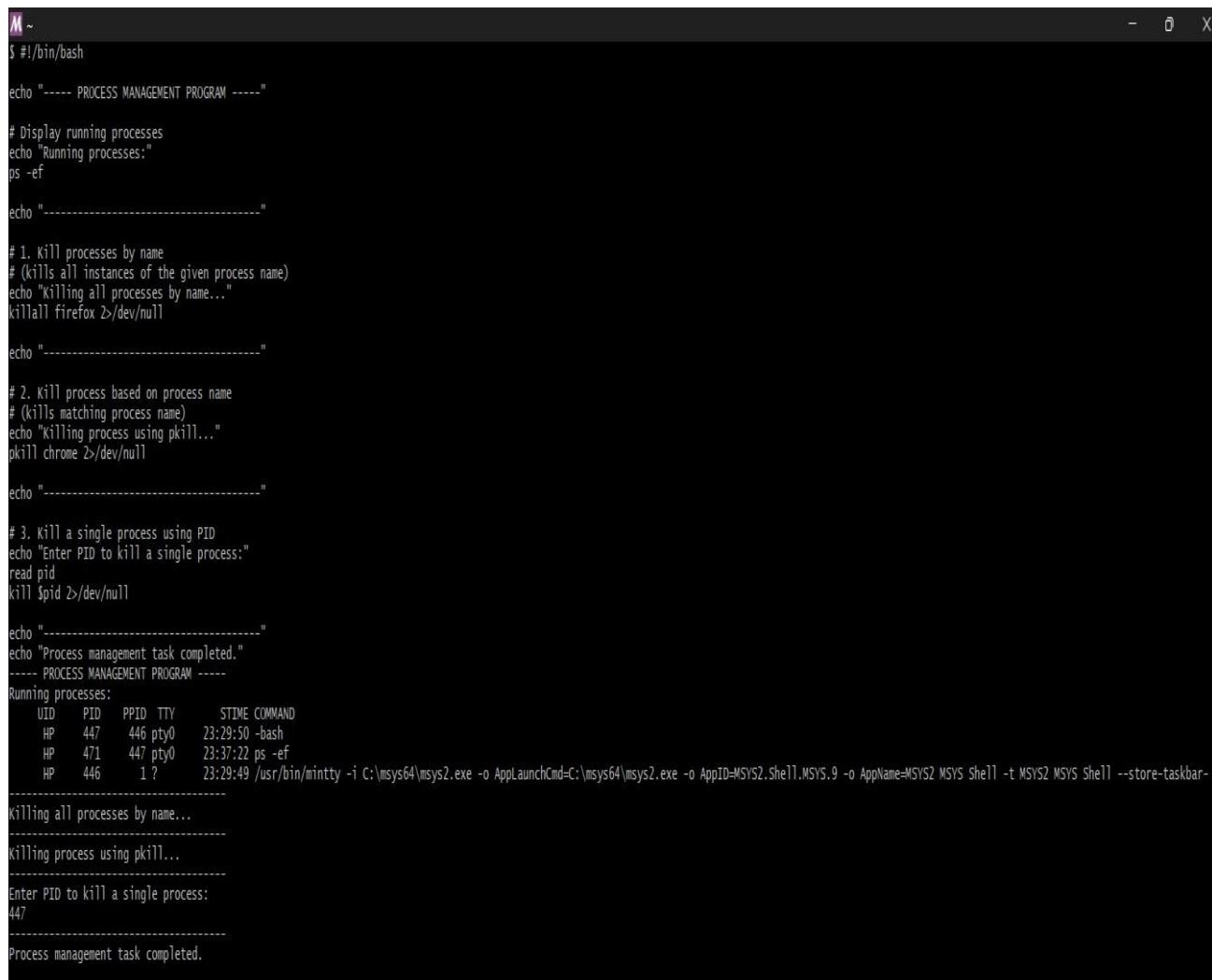


1. Adam is working in an IT company. He has been given a task to reduce the load of a system by killing some of the processes running in the LINUX operating system. Which commands will he use to complete the given task with the help of the following operation?

- 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 explaining the steps: 1. Kill processes by name (kill all instances of a given name), 2. Kill process based on process name (kill matching process name using pkill), and 3. Kill a single process using PID (enter PID to kill a single process). The script also handles user input for the third step. The output shows the running processes, the execution of killall and pkill commands, the user entering a PID, and finally a message indicating the task is completed.

```
$ #!/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:

```
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 ~
chanc@Chanchal MSYS ~
$ nano orphan.c

chanc@Chanchal MSYS ~
$ gcc orphan.c -o orphan

chanc@Chanchal MSYS ~
$ ./orphan
Parent exiting

chanc@Chanchal MSYS ~
$ Child Process
PID : 2393
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 :

```
M ~
chanc@Chanchal MSYS ~
$ nano zombie.c

chanc@Chanchal MSYS ~
$ gcc zombie.c -o zombie

chanc@Chanchal MSYS ~
$ ./zombie
Child exiting
Parent running

chanc@Chanchal MSYS ~
$ ps -el
   PID   PPID   PGID   WINPID   TTY      UID   STIME COMMAND
 1959     1   1959    19544   ? 197610 14:49:23 /usr/bin/mintty
 1960   1959   1960    18476  pty0 197610 14:49:23 /usr/bin/bash
 2410   1960   2410    19408  pty0 197610 15:07:27 /usr/bin/ps

chanc@Chanchal 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 :

```
chanc@Chanchal MSYS ~
$ gcc --version
gcc (GCC) 15.2.0
Copyright (C) 2025 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

chanc@Chanchal MSYS ~
$ nano fork.c

chanc@Chanchal MSYS ~
$ gcc fork.c -o fork

chanc@Chanchal MSYS ~
$ ./fork
Child Process
PID : 2384
PPID : 2383
Parent Process
PID : 2383
Child PID : 2384

chanc@Chanchal MSYS ~
$
```