

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

```
M ~

np@DESKTOP-CA6S8T8:~$ 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.

np@DESKTOP-CA6S8T8:~$ sleep 1000 &
[1] 1727

np@DESKTOP-CA6S8T8:~$ ps
  PID  PPID  PGID   WINPID   TTY      UID  STIME  COMMAND
 1708  1707  1708   16376  pty0    197609 21:50:51 /usr/bin/bash
 1727  1708  1727   10472  pty0    197609 21:51:17 /usr/bin/sleep
 1728  1708  1728   15996  pty0    197609 21:51:22 /usr/bin/ps
 1707          1  1707   16212 ?     197609 21:50:51 /usr/bin/mintty

np@DESKTOP-CA6S8T8:~$ pkill sleep
-bash: pkill: command not found

np@DESKTOP-CA6S8T8:~$ ps
  PID  PPID  PGID   WINPID   TTY      UID  STIME  COMMAND
 1708  1707  1708   16376  pty0    197609 21:50:51 /usr/bin/bash
 1727  1708  1727   10472  pty0    197609 21:51:17 /usr/bin/sleep
 1707          1  1707   16212 ?     197609 21:50:51 /usr/bin/mintty
 1730  1708  1730   12356  pty0    197609 21:51:43 /usr/bin/ps

np@DESKTOP-CA6S8T8:~$ kill 1238
-bash: kill: (1238) - No such process

np@DESKTOP-CA6S8T8:~$ kill 1727

np@DESKTOP-CA6S8T8:~$ ps | grep sleep
[1]+  Terminated                  sleep 1000

np@DESKTOP-CA6S8T8:~$
```

```
M ~

hp@DESKTOP-CA6S8T8:MSYS ~
$ sleep 1000 &
[1] 562

hp@DESKTOP-CA6S8T8:MSYS ~
$ ps -e | grep sleep
 562      544      562      18308  pty0      197609 21:54:42 /usr/bin/sleep

hp@DESKTOP-CA6S8T8:MSYS ~
$ kill 562

hp@DESKTOP-CA6S8T8:MSYS ~
$ ps -e | grep sleep
[1]+  Terminated                 sleep 1000
sleep 1000

hp@DESKTOP-CA6S8T8:MSYS ~
$ |
```

```
M ~
hp@DESKTOP-CA6S8T8 MSYS ~
$ sleep 1000 &
[1] 435

hp@DESKTOP-CA6S8T8 MSYS ~
$ ps -e
  PID  PPID  PGID    WINPID   TTY      UID      STIME COMMAND
  435    417    435     17560  pty0    197609 21:57:58 /usr/bin/sleep
  417    416    417     5008  pty0    197609 21:57:30 /usr/bin/bash
  416      1    416     5808    ?    197609 21:57:30 /usr/bin/mintty
  436    417    436    16316  pty0    197609 21:58:05 /usr/bin/ps

hp@DESKTOP-CA6S8T8 MSYS ~
$ kill 435

hp@DESKTOP-CA6S8T8 MSYS ~
$ ps -e | grep sleep
[1]+  Terminated                 sleep 1000
hp@DESKTOP-CA6S8T8 MSYS ~
$
```

2. Write a program for process creation using C

- Orphan Process

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

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

    if (pid > 0) {
        printf("Parent PID: %d\n", getpid());
        sleep(2);
        printf("Parent exiting...\n");
    } else {
        sleep(5);
        printf("child PID: %d\n", getpid());
        printf("child PPID: %d\n", getppid());
    }
    return 0;
}
```

```
M ~
hp@DESKTOP-CA6S8T8 MSYS ~
$ nano orphan.c
hp@DESKTOP-CA6S8T8 MSYS ~
$ gcc orphan.c -o orphan
hp@DESKTOP-CA6S8T8 MSYS ~
$ ./orphan
Parent PID: 1193
Parent exiting...
hp@DESKTOP-CA6S8T8 MSYS ~
$ Child PID: 1194
Child PPID: 1
$ |
```

- Zombie Process

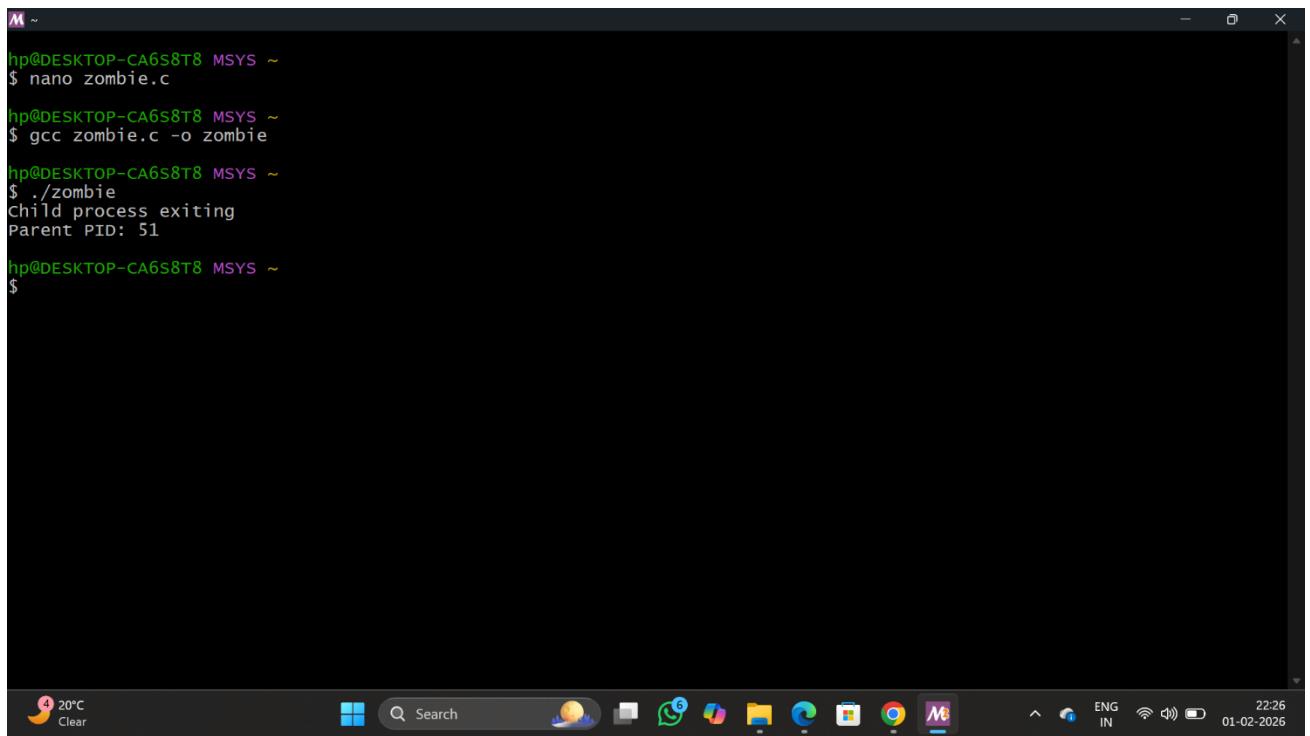
```
M ~
GNU nano 8.7
#include <stdio.h>
#include <unistd.h>

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

    if (pid == 0) {
        printf("child process exiting\n");
    } else {
        sleep(10);
        printf("Parent PID: %d\n", getpid());
    }
    return 0;
}
```

[Wrote 14 lines]

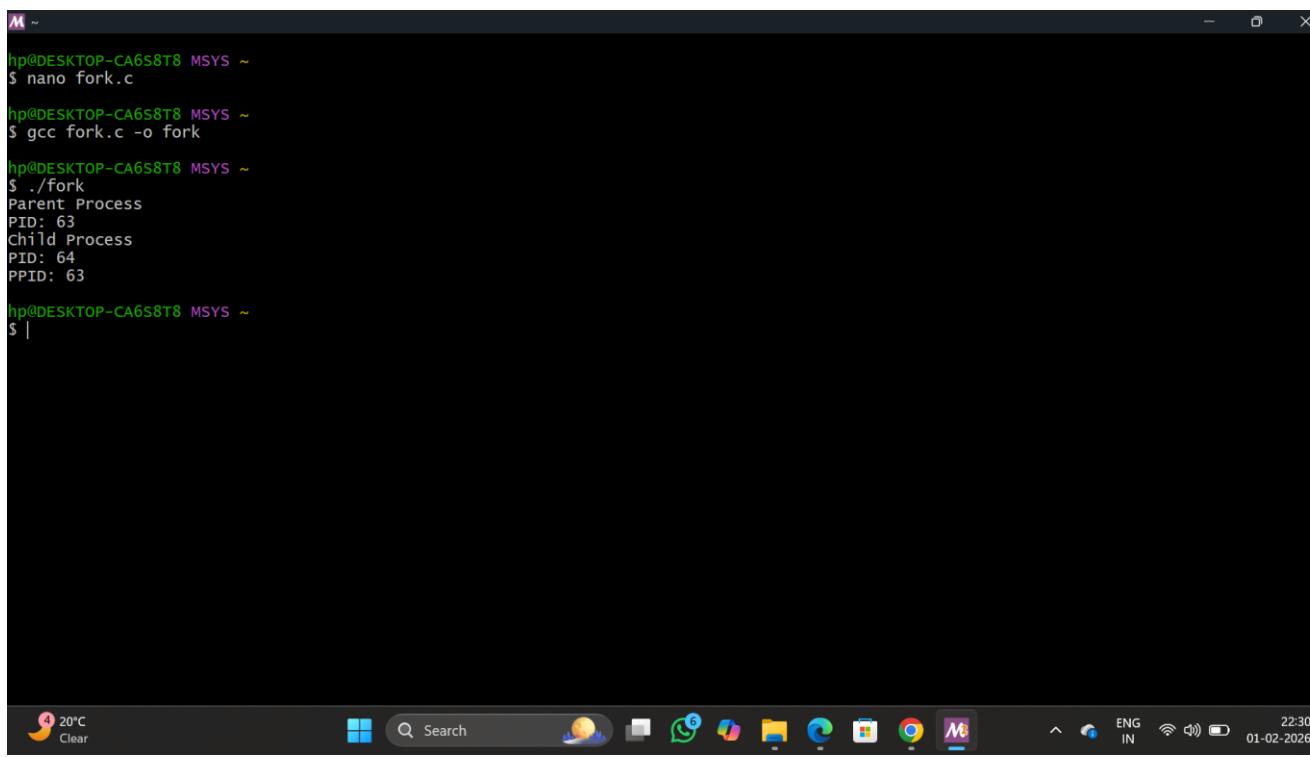
AG Help F0 Write Out AF Where Is AK Cut AU Paste AT Execute AC Location M-U Undo
AX Exit AR Read File AR Replace AJ Justify AG Go To Line M-E Redo M-A Set Mark M-B To Bracket
J Clear ENG IN 22:23 01-02-2026



```
M ~
hp@DESKTOP-CA6S8T8 MSYS ~
$ nano zombie.c
hp@DESKTOP-CA6S8T8 MSYS ~
$ gcc zombie.c -o zombie
hp@DESKTOP-CA6S8T8 MSYS ~
$ ./zombie
child process exiting
Parent PID: 51
hp@DESKTOP-CA6S8T8 MSYS ~
$
```

3.Create the process using fork () system call.

- Child Process creation
- Parent process creation
- PPID and PID



```
M ~
hp@DESKTOP-CA6S8T8 MSYS ~
$ nano fork.c
hp@DESKTOP-CA6S8T8 MSYS ~
$ gcc fork.c -o fork
hp@DESKTOP-CA6S8T8 MSYS ~
$ ./fork
Parent Process
PID: 63
Child Process
PID: 64
PPID: 63
hp@DESKTOP-CA6S8T8 MSYS ~
$ |
```

M ~

GNU nano 8.7 orphan.c

```
#include <stdio.h>
#include <unistd.h>

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

    if (pid > 0) {
        printf("Parent PID: %d\n", getpid());
        sleep(2);
        printf("Parent exiting...\n");
    } else {
        sleep(5);
        printf("Child PID: %d\n", getpid());
        printf("Child PPID: %d\n", getppid());
    }
    return 0;
}
```

[Read 17 lines]

▲G Help ▲O Write Out ▲F Where Is ▲K Cut ▲T Execute ▲C Location M-U Undo
▲X Exit ▲R Read File ▲N Replace ▲U Paste ▲J Justify ▲V Go To Line M-E Redo M-A Set Mark
M-G Copy

20°C Clear Search File Explorer WhatsApp Edge Microsoft Store Google Chrome 22:08
ENG IN 01-02-2026