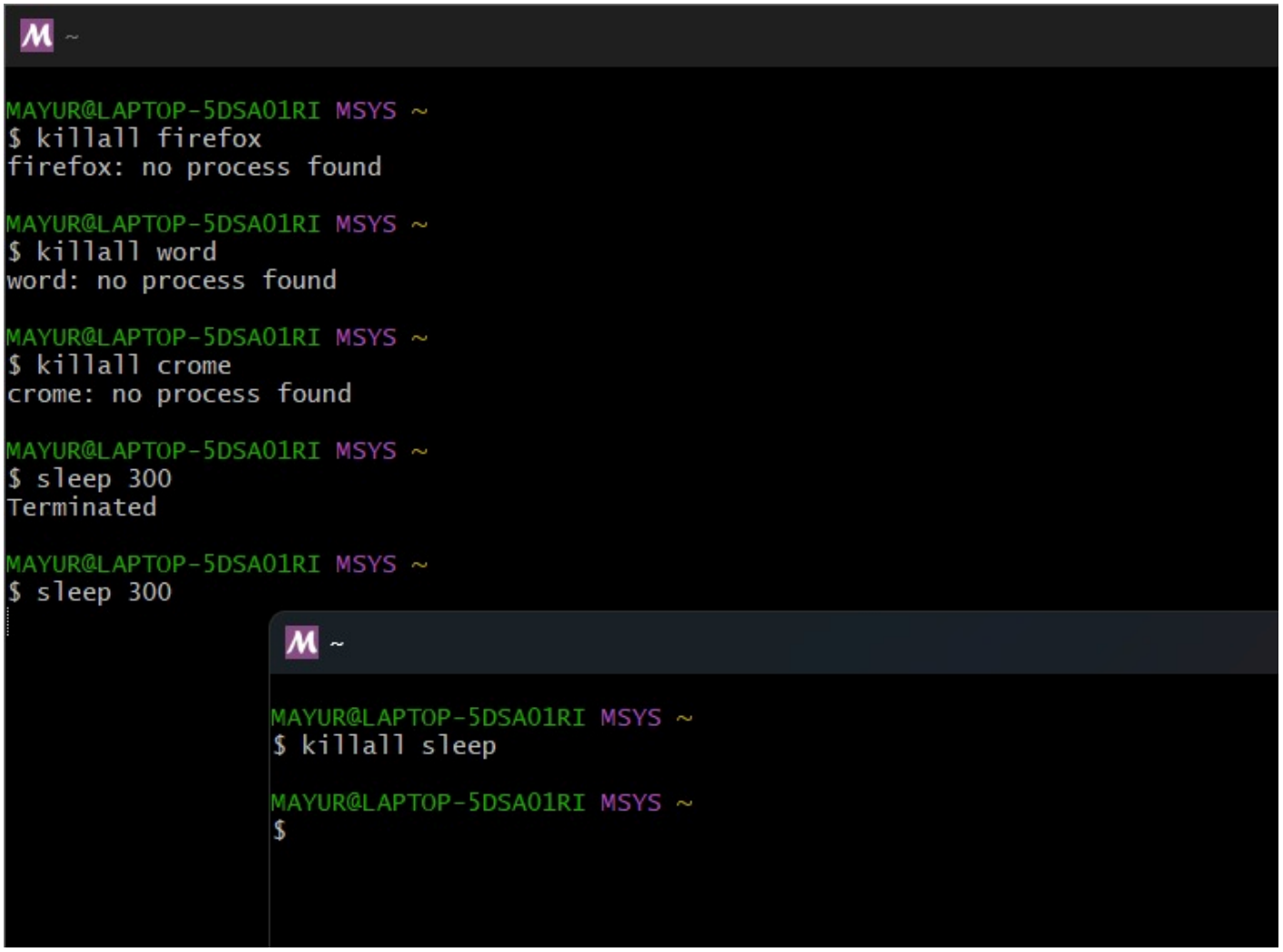
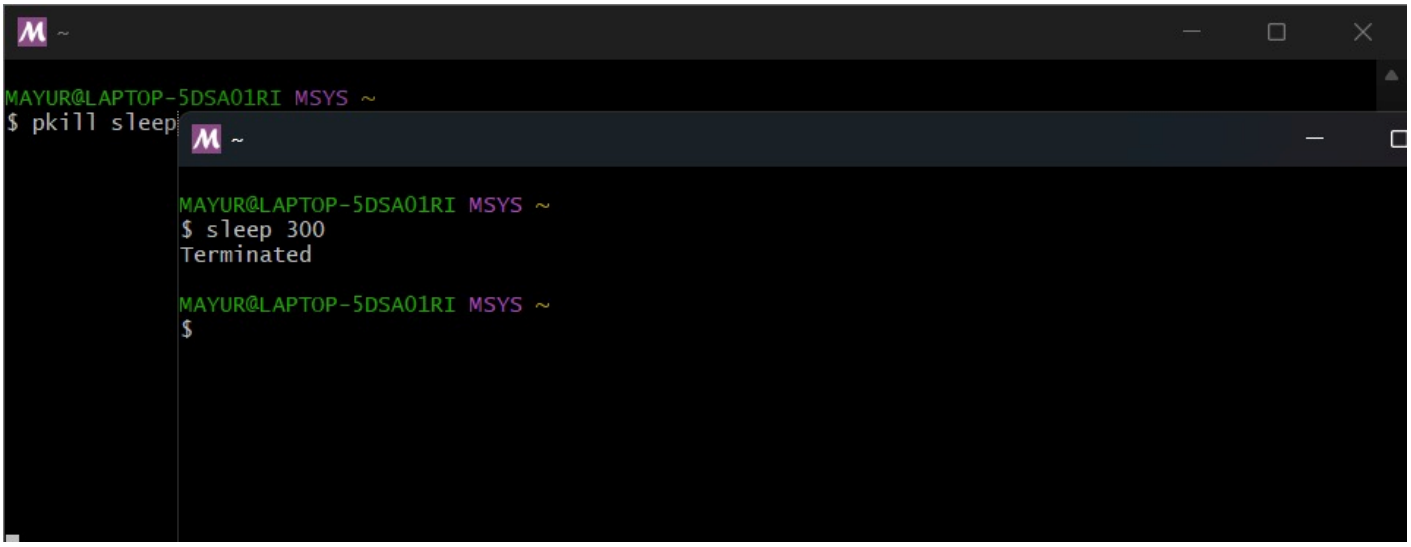
1. LINUX operating system. Which commands will he use to complete the given task with the help of the following operation?

●Kill processes by name

●Kill a process based on the process name

●Kill a single process at a time with the given process ID





```
MAYUR@LAPTOP-5DSA01RI MSYS ~
$ sleep 300

MAYUR@LAPTOP-5DSA01RI MSYS ~
$
```

```
MAYUR@LAPTOP-5DSA01RI MSYS ~
$ sleep 300 &
[2] 2003

MAYUR@LAPTOP-5DSA01RI MSYS ~
$ kill 2003

MAYUR@LAPTOP-5DSA01RI MSYS ~
$ kill %2
-bash: kill: (2003) - No such process
[2]+  Terminated              sleep 300

MAYUR@LAPTOP-5DSA01RI MSYS ~
$
```

2 .Write a program for process creation using C

- **Orphan Process**
- **Zombie Process**

orphan process

```
GNU nano 8.7                                    hello.c
#include <stdio.h>
#include <unistd.h>

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

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

```
MAYUR@LAPTOP-5DSA01RI MSYS :
nano nano hello.c

MAYUR@LAPTOP-5DSA01RI MSYS :
gcc hello.c -o hello

MAYUR@LAPTOP-5DSA01RI MSYS :
Parent process exiting
Child process becomes orphan
PID: 1924, PPID: 1
```

zombie Process

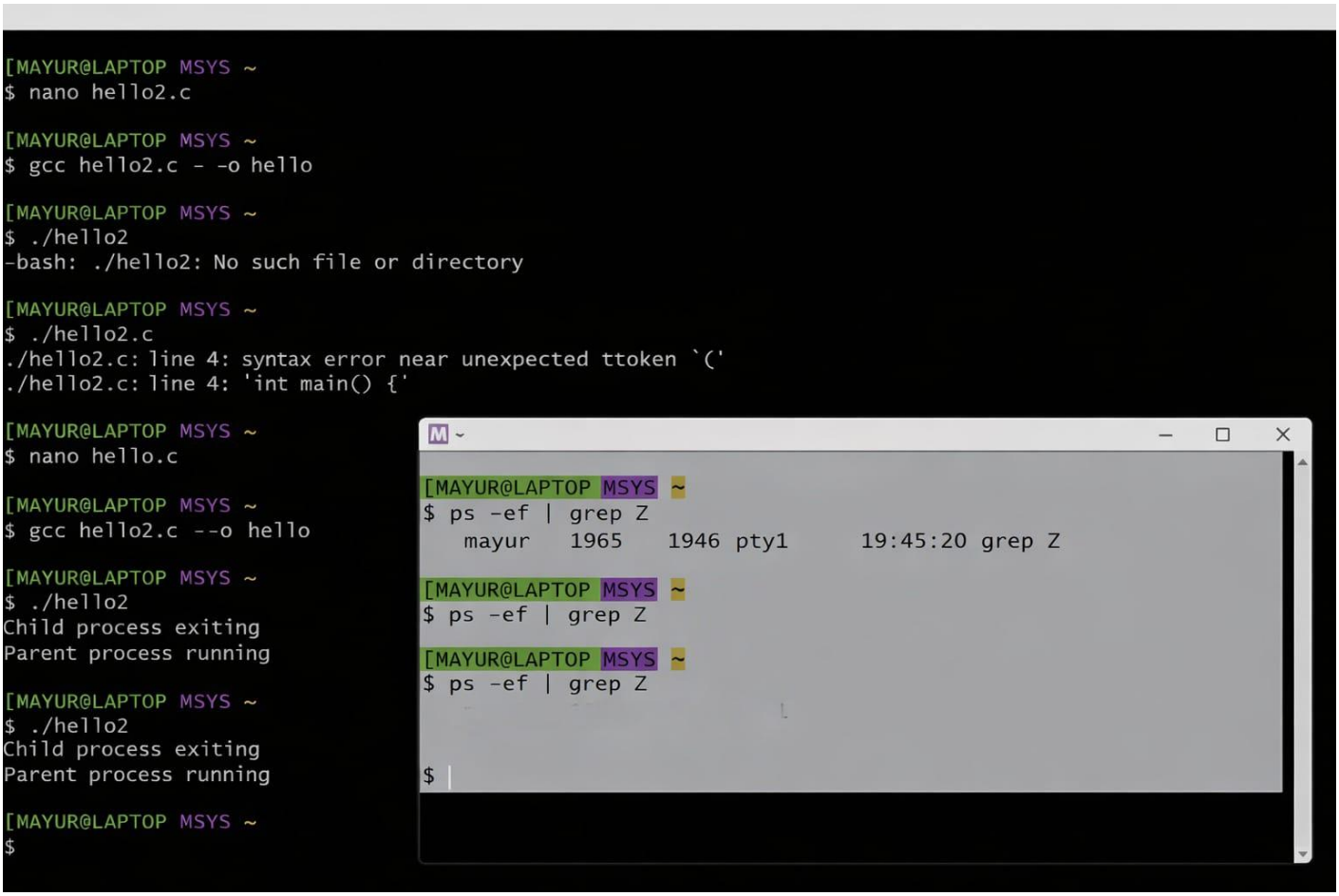
~

```
  GNU nano 8.7                                          hello2.c
#include <stdio.h>
#include <unistd.h>

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

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

    return 0;
}
```

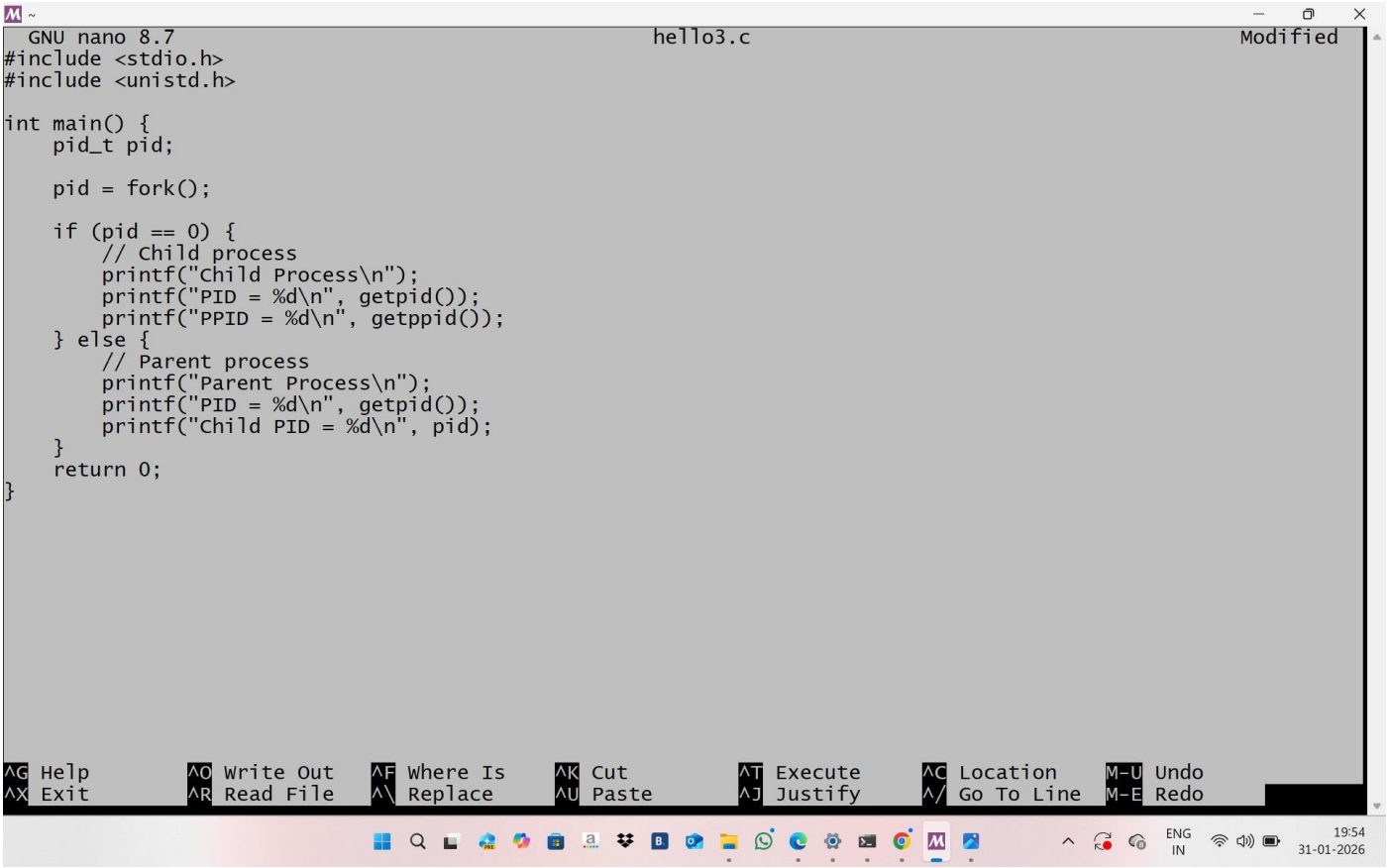


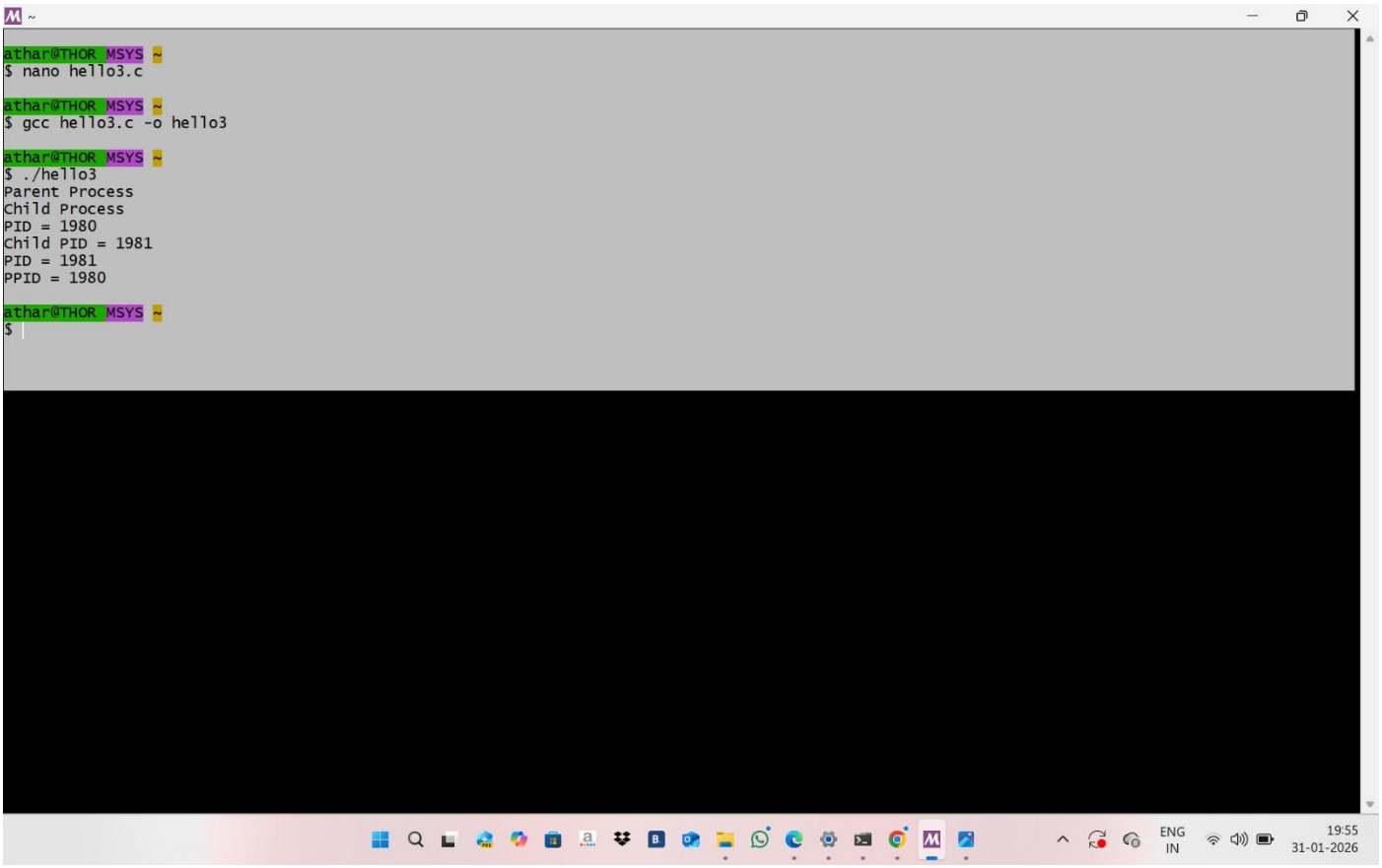
3. Create the process using fork () system call.

●Child Process creation

●Parent process creation

●PPID and PID


~
GNU nano 8.7 hello3.c Modified
#include <stdio.h>
#include <unistd.h>
int main() {
pid_t pid;
pid = fork();
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;
}
^G Help ^O Write Out ^F Where Is ^K Cut ^T Execute ^C Location M-U Undo
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo
ENG IN
19:54
31-01-2026


~
athar@THOR MSYS ~
$ nano hello3.c
athar@THOR MSYS ~
$ gcc hello3.c -o hello3
athar@THOR MSYS ~
$ ./hello3
Parent Process
Child Process
PID = 1980
Child PID = 1981
PID = 1981
PPID = 1980
athar@THOR MSYS ~
$
ENG IN
19:55
31-01-2026