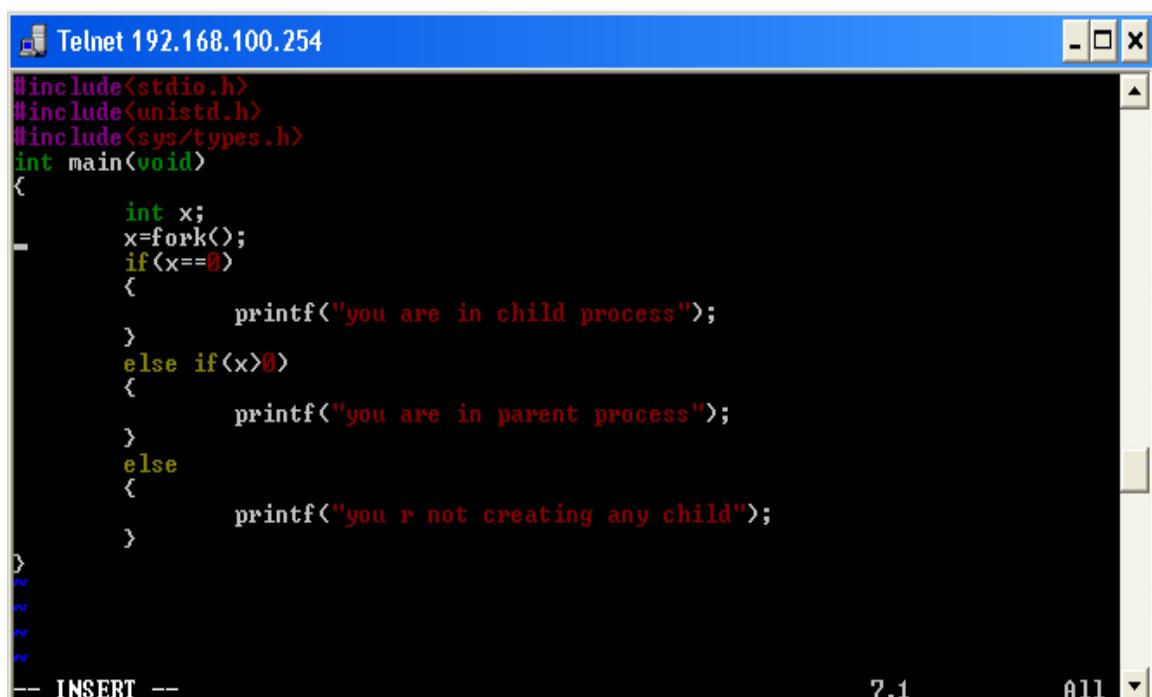


Practical :- 10

AIM: WAP to study various commands related to processes.

A simple program using fork command

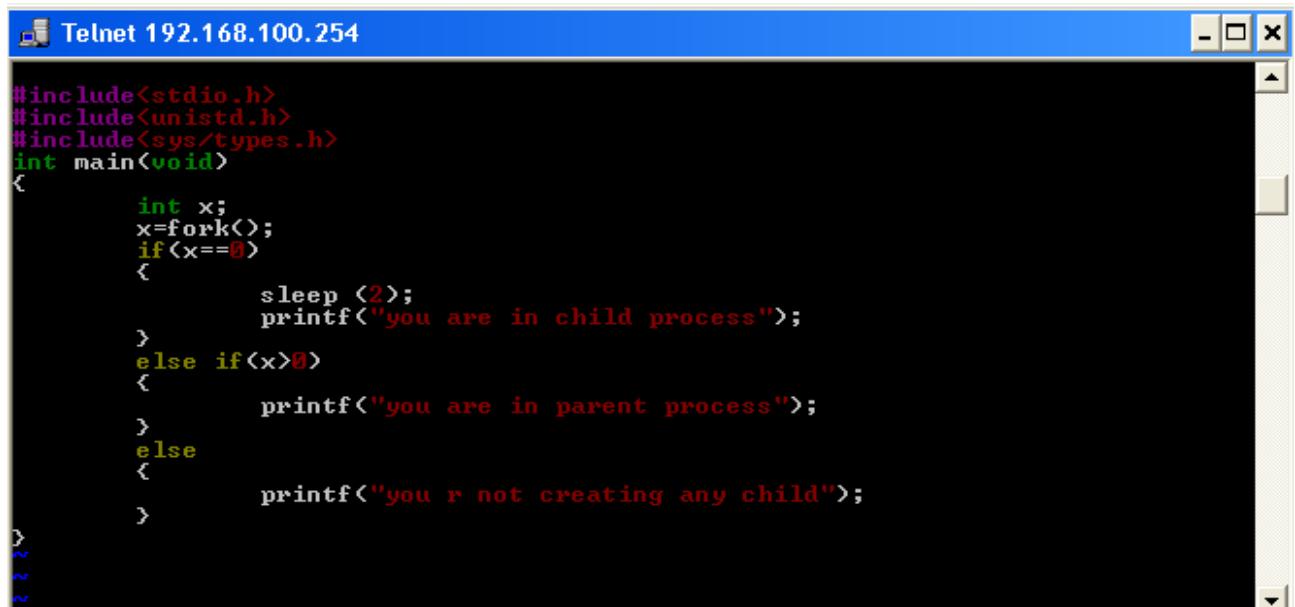


```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
int main(void)
{
    int x;
    x=fork();
    if(x==0)
    {
        printf("you are in child process");
    }
    else if(x>0)
    {
        printf("you are in parent process");
    }
    else
    {
        printf("you r not creating any child");
    }
}
~
~
~
~
-- INSERT --
```

7,1

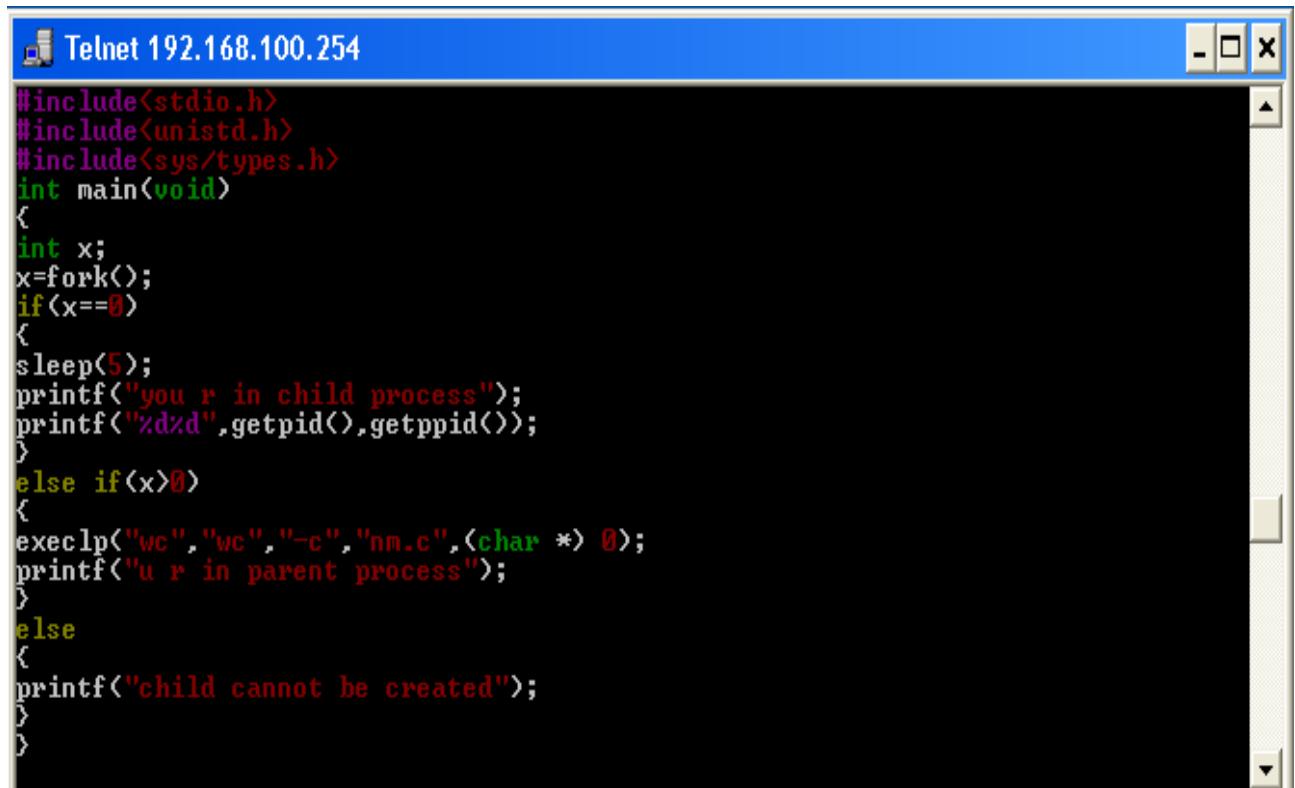
All

A simple program using fork and sleep command



```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
int main(void)
{
    int x;
    x=fork();
    if(x==0)
    {
        sleep(2);
        printf("you are in child process");
    }
    else if(x>0)
    {
        printf("you are in parent process");
    }
    else
    {
        printf("you r not creating any child");
    }
}
~
~
~
```

A simple program printing the id of process using various commands

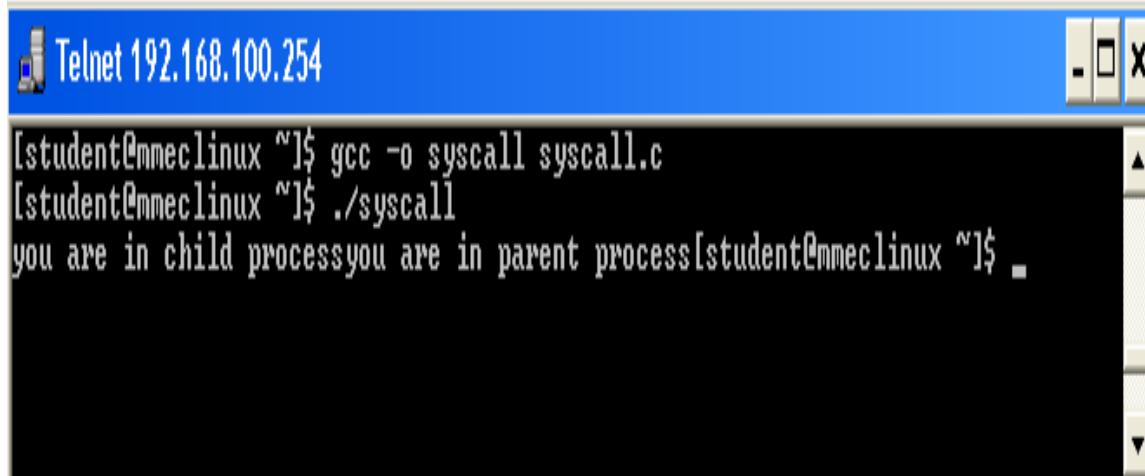


The screenshot shows a Telnet session titled "Telnet 192.168.100.254". The code displayed is a C program that includes stdio.h, unistd.h, and sys/types.h. It defines a main function that forks. If the fork fails (x == -1), it prints "child cannot be created". Otherwise, if x is 0 (parent), it execs wc -c nm.c. If x is > 0 (child), it sleeps for 5 seconds and then prints "you r in child process" followed by its pid and ppid.

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
int main(void)
{
    int x;
    x=fork();
    if(x==0)
    {
        sleep(5);
        printf("you r in child process");
        printf("%d%d",getpid(),getppid());
    }
    else if(x>0)
    {
        execvp("wc","wc",-c,"nm.c",char * 0);
        printf("u r in parent process");
    }
    else
    {
        printf("child cannot be created");
    }
}
```

SCREENSHOTS:

When only fork command was used



The screenshot shows a terminal window titled "[student@mmeclinux ~]\$". The user runs gcc -o syscall syscall.c and then ./syscall. The output shows two lines: "you are in child process" and "you are in parent process".

```
[student@mmeclinux ~]$ gcc -o syscall syscall.c
[student@mmeclinux ~]$ ./syscall
you are in child processyou are in parent process[student@mmeclinux ~]$ _
```

When fork and sleep were used

Telnet 192.168.100.254

```
"syscall.c" 21L, 277C written
[student@mmeclinux ~]$ gcc -o syscall syscall.c
[student@mmeclinux ~]$ ./syscall
you are in parent process[student@mmeclinux ~]$ you are in child process
```

Displaying the id's process

Telnet 192.168.100.254

```
"km.c" 25L, 325C written
[student@mmeclinux ~]$ gcc -o km km.c
[student@mmeclinux ~]$ ./km
107 nm.c
[student@mmeclinux ~]$ you r in child process27051
```