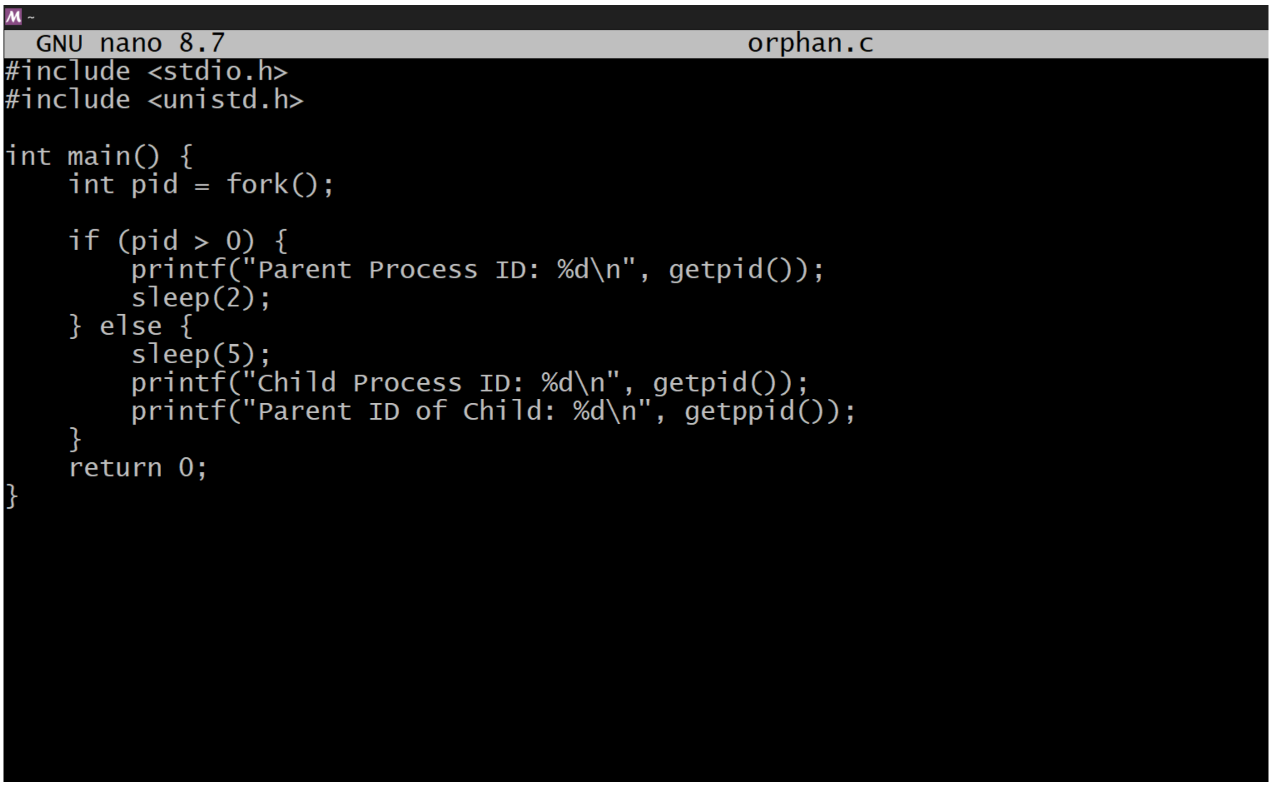
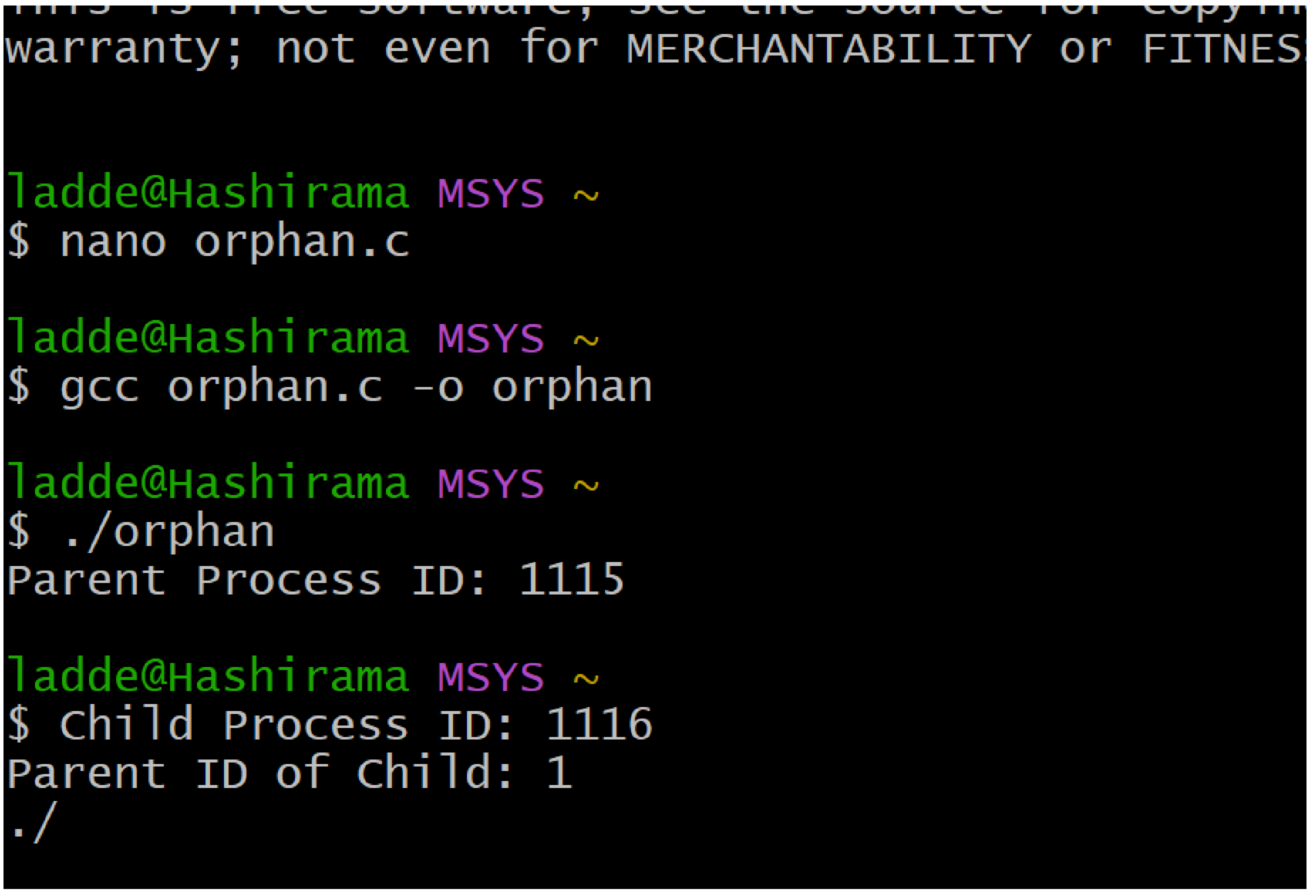
Practical NO.3

●Orphan Process



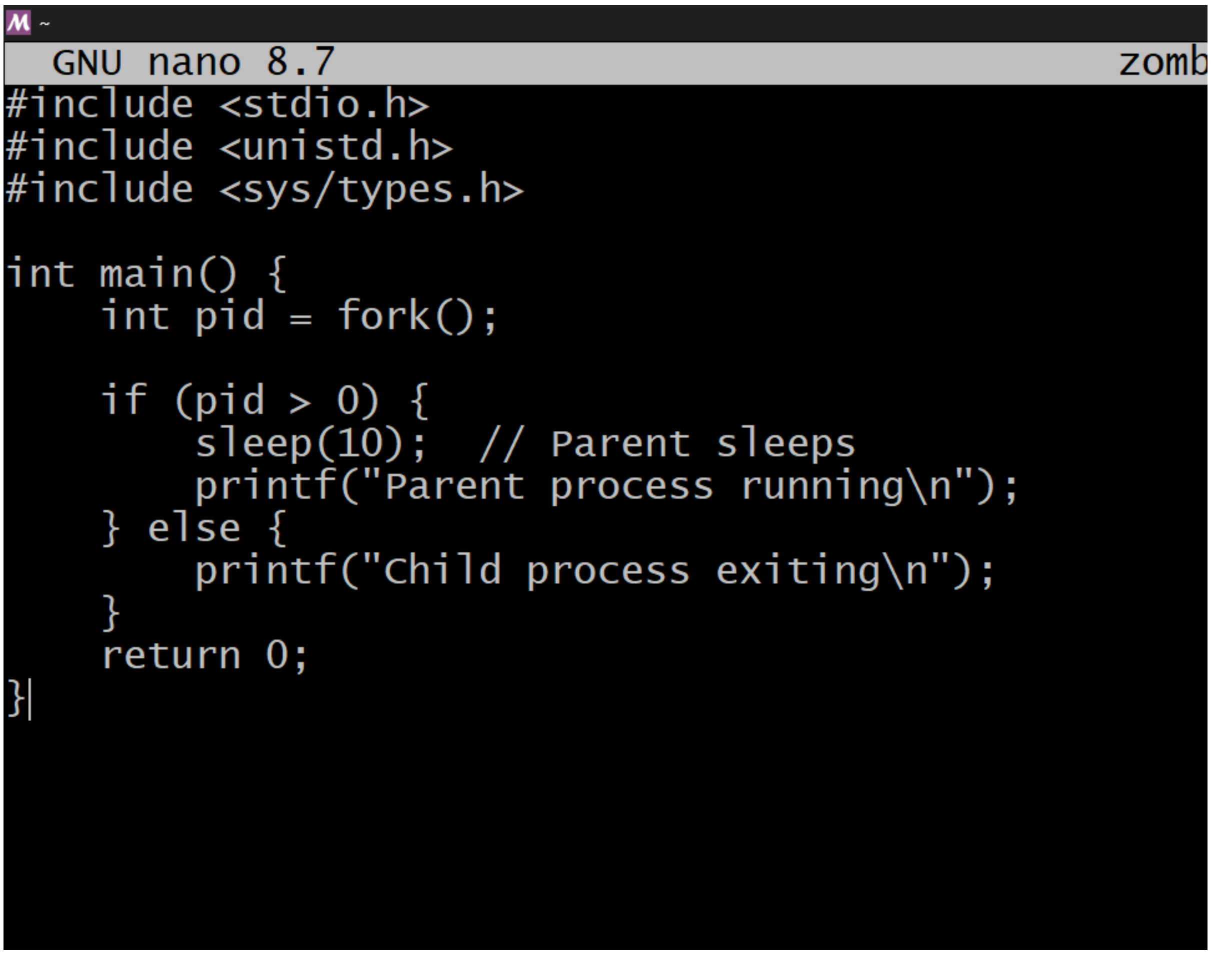
```c
#include <stdio.h>
#include <unistd.h>

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

    if (pid > 0) {
        printf("Parent Process ID: %d\n", getpid());
        sleep(2);
    } else {
        sleep(5);
        printf("Child Process ID: %d\n", getpid());
        printf("Parent ID of Child: %d\n", getppid());
    }
    return 0;
}
```

Output:-



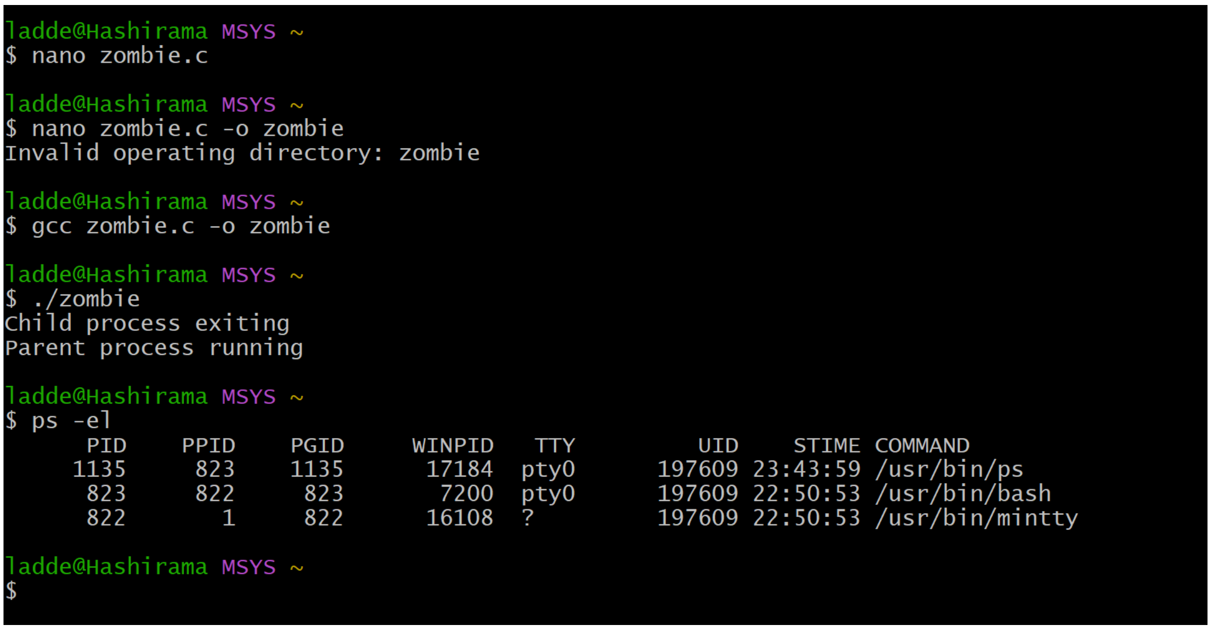
```
This is free software; see the source for copyin
warranty; not even for MERCHANTABILITY or FITNES


ladde@Hashirama MSYS ~
$ nano orphan.c

ladde@Hashirama MSYS ~
$ gcc orphan.c -o orphan

ladde@Hashirama MSYS ~
$ ./orphan
Parent Process ID: 1115

ladde@Hashirama MSYS ~
$ Child Process ID: 1116
Parent ID of Child: 1
./
```

●Zombie Process



```
  GNU nano 8.7                                             zomb
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

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

    if (pid > 0) {
        sleep(10);   // Parent sleeps
        printf("Parent process running\n");
    } else {
        printf("Child process exiting\n");
    }
    return 0;
}
```

Output

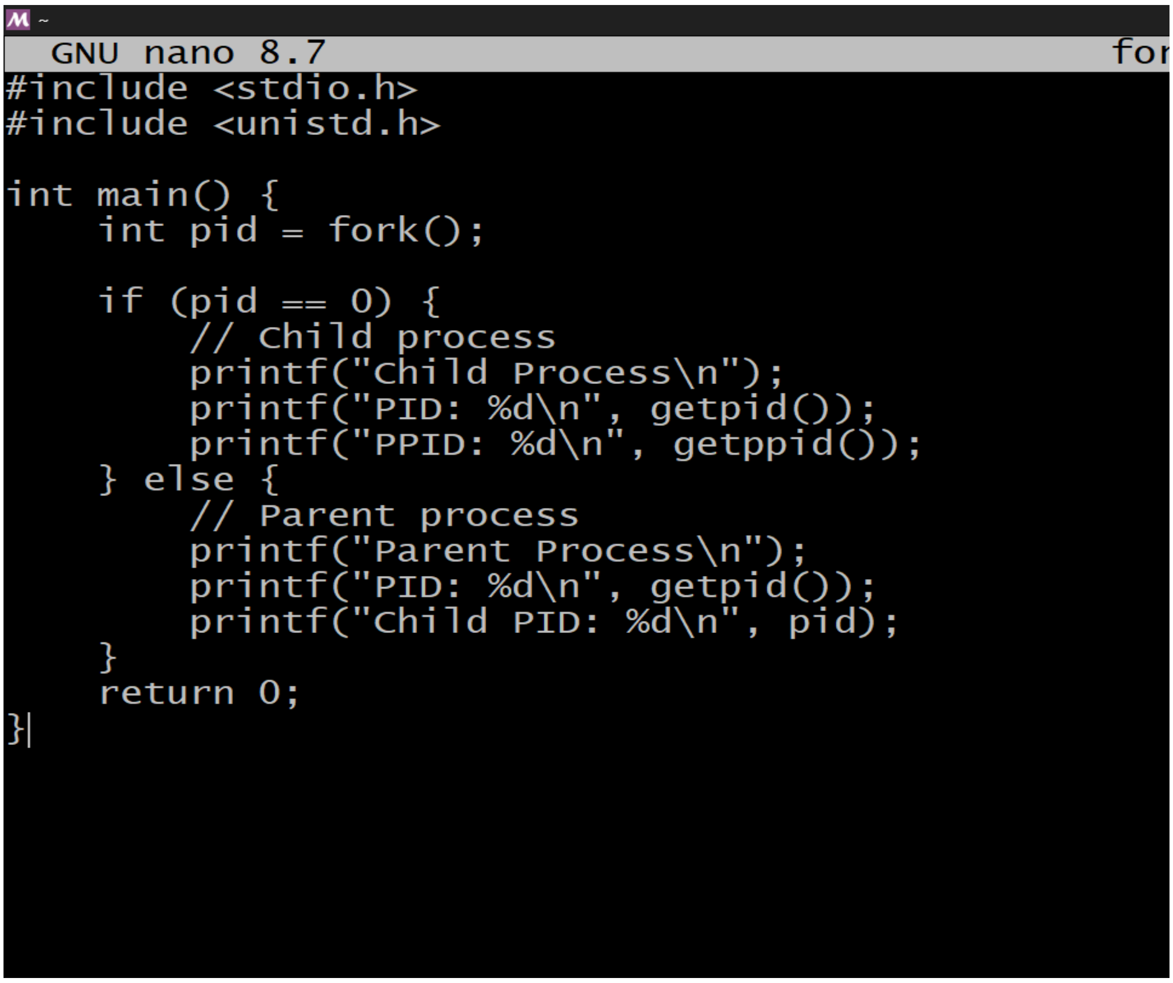


```
ladde@Hashirama MSYS ~
$ nano zombie.c

ladde@Hashirama MSYS ~
$ nano zombie.c -o zombie
Invalid operating directory: zombie

ladde@Hashirama MSYS ~
$ gcc zombie.c -o zombie

ladde@Hashirama MSYS ~
$ ./zombie
Child process exiting
Parent process running

ladde@Hashirama MSYS ~
$ ps -el
      PID    PPID    PGID     WINPID   TTY         UID    STIME COMMAND
     1135     823    1135      17184  pty0      197609 23:43:59 /usr/bin/ps
      823     822     823       7200  pty0      197609 22:50:53 /usr/bin/bash
      822       1     822      16108  ?         197609 22:50:53 /usr/bin/mintty

ladde@Hashirama MSYS ~
$
```

Create the process using fork () system call.

- Child Process creation
- Parent process creation

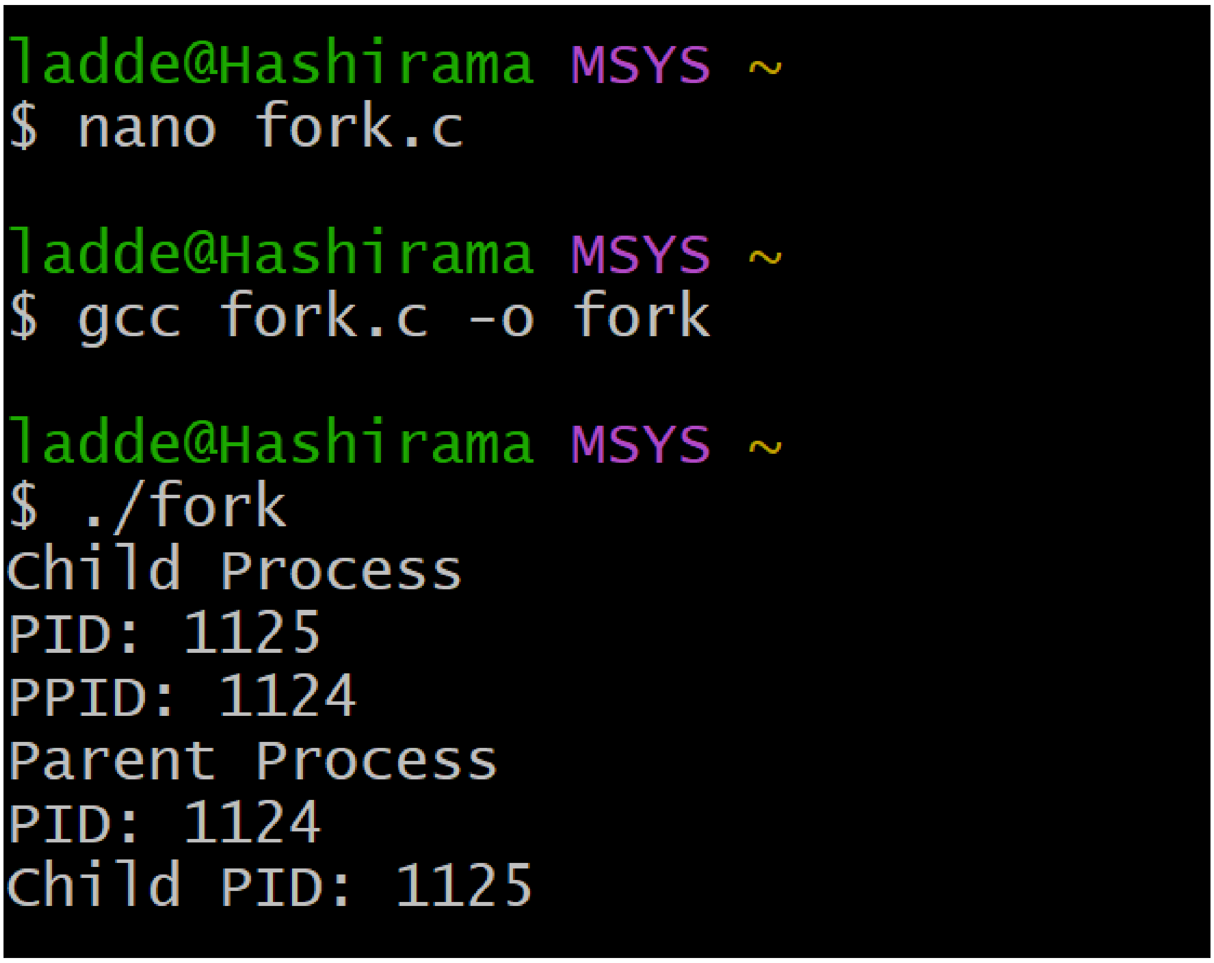
●PPID and PID



GNU nano 8.7 for

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

int main() {
    int 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;
}
```

Output



```
ladde@Hashirama MSYS ~
$ nano fork.c

ladde@Hashirama MSYS ~
$ gcc fork.c -o fork

ladde@Hashirama MSYS ~
$ ./fork
Child Process
PID: 1125
PPID: 1124
Parent Process
PID: 1124
Child PID: 1125
```