Practical : 03

Aim: Implement process creation using the fork() system call in Linux to generate parent and child processes, retrieve their PID and PPID, and analyse system behaviour by creating orphan and zombie processes.

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

```
Nandini Kasare@LAPTOP-NIHKGM4O MINGW64 ~/Desktop (main)
$ taskkill //IM notepad.exe
SUCCESS: Sent termination signal to the process "Notepad.exe" with PID 21432.

Nandini Kasare@LAPTOP-NIHKGM4O MINGW64 ~/Desktop (main)
$
```

```
Nandini Kasare@LAPTOP-NIHKGM4O MINGW64 ~/Desktop (main)
$ taskkill //IM notepad.exe
SUCCESS: Sent termination signal to the process "Notepad.exe" with PID 21432.

Nandini Kasare@LAPTOP-NIHKGM4O MINGW64 ~/Desktop (main)
$ taskkill //IM chrome.exe //F
SUCCESS: The process "chrome.exe" with PID 29752 has been terminated.
SUCCESS: The process "chrome.exe" with PID 3424 has been terminated.
SUCCESS: The process "chrome.exe" with PID 15308 has been terminated.
SUCCESS: The process "chrome.exe" with PID 25812 has been terminated.
```

2. Write a program for process creation using C

▪ Orphan Process

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

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

    if (pid > 0) {
        // Parent process
        printf("Parent process exiting...\n");
        printf("Parent PID: %d\n", getpid());
        sleep(2);
    }
    else if (pid == 0) {
        // Child process
        sleep(5);  // parent ke exit hone ka wait
        printf("Child process running...\n");
        printf("Child PID: %d\n", getpid());
        printf("New Parent PID (init/systemd): %d\n", getppid());
    }
    else {
        printf("Fork failed!\n");
    }

    return 0;
}
                          [ Read 26 lines ]
^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
```

OUTPUT:

```
Nandini Kasare@LAPTOP-NIHKGM4O MSYS ~
$ nano Orphan.c

Nandini Kasare@LAPTOP-NIHKGM4O MSYS ~
$ gcc Orphan.c -o Orphan

Nandini Kasare@LAPTOP-NIHKGM4O MSYS ~
$ ./Orphan
Parent process exiting...
Parent PID: 1183

Nandini Kasare@LAPTOP-NIHKGM4O MSYS ~
$ Child process running...
Child PID: 1184
New Parent PID (init/systemd): 1
```

▪ Zombie Process

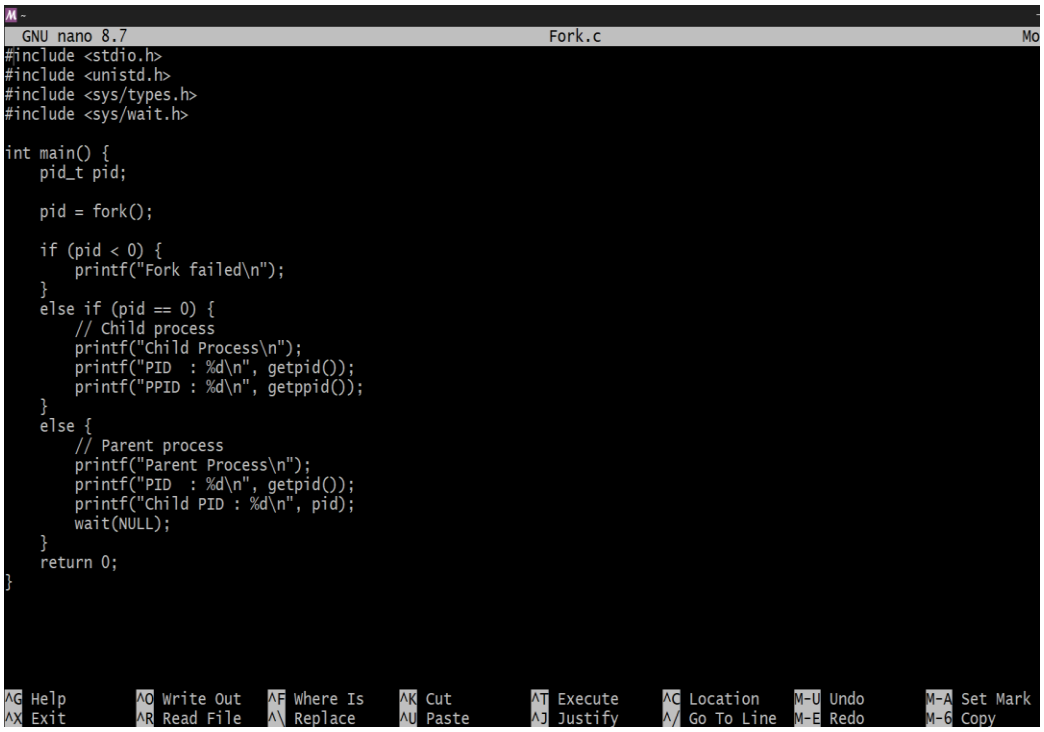
```
~
  GNU nano 8.7                                  Zomibe.c
#include <stdio.h>
#include <unistd.h>

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

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

Write to File: Zomibe.c
^G Help          M-D DOS Format    M-A Append      M-B Backup File   ^T Browse
^C Cancel        M-M Mac Format    M-P Prepend     ^Q Discard buffer
```

OUTPUT:

```
~
Nandini Kasare@LAPTOP-NIHKGM4O MSYS ~
$ nano Zombie.c

Nandini Kasare@LAPTOP-NIHKGM4O MSYS ~
$ gcc Zombie.c -o Zombie

Nandini Kasare@LAPTOP-NIHKGM4O MSYS ~
$ ./Zombie
Child exiting
Parent running

Nandini Kasare@LAPTOP-NIHKGM4O MSYS ~
$
```

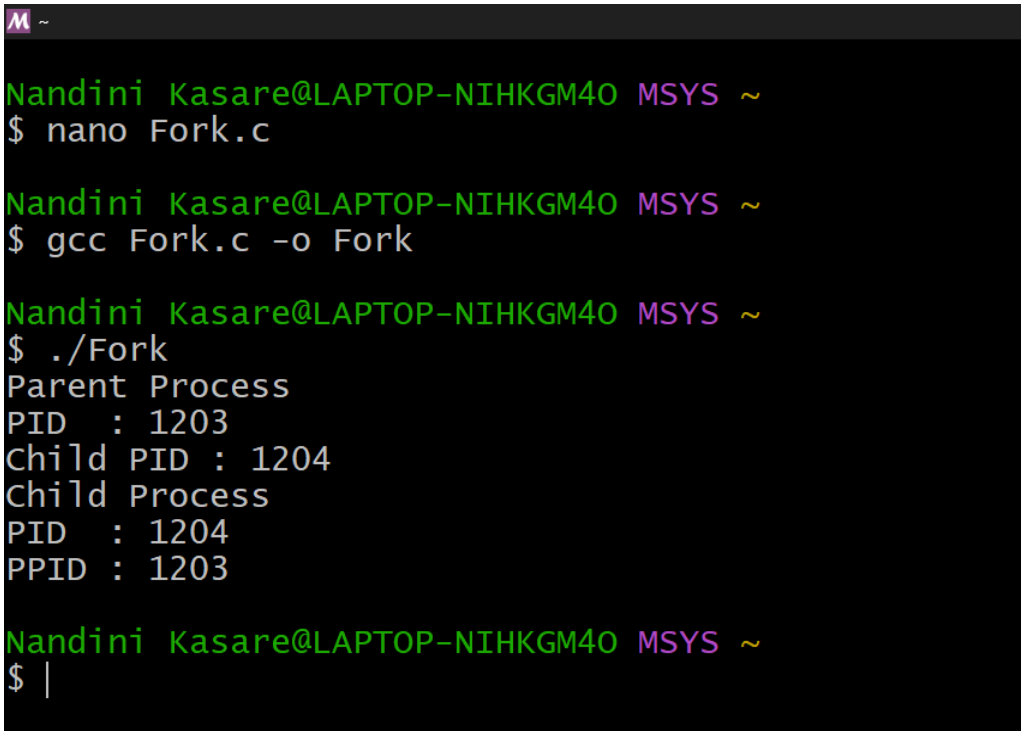
3.Create the process using fork () system call.

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



```
 GNU nano 8.7                                  Fork.c                                  Mo
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main() {
    pid_t pid;

    pid = fork();

    if (pid < 0) {
        printf("Fork failed\n");
    }
    else 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);
        wait(NULL);
    }
    return 0;
}

^G Help      ^O Write Out   ^F Where Is   ^K Cut       ^T Execute   ^C Location   M-U Undo    M-A Set Mark
^X Exit      ^R Read File   ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line M-E Redo    M-6 Copy
```

OUTPUT:



```
~

Nandini Kasare@LAPTOP-NIHKGM4O MSYS ~
$ nano Fork.c

Nandini Kasare@LAPTOP-NIHKGM4O MSYS ~
$ gcc Fork.c -o Fork

Nandini Kasare@LAPTOP-NIHKGM4O MSYS ~
$ ./Fork
Parent Process
PID  : 1203
Child PID : 1204
Child Process
PID  : 1204
PPID : 1203

Nandini Kasare@LAPTOP-NIHKGM4O MSYS ~
$
```