EX.NO: 4	**DEMONSTRATE THE CREATION OF ZOMBIE AND ORPHAN PROCESSES**
DATE :	

AIM

To demonstrate the creation of Zombie and Orphan process using C program.

Zombie Process

- A zombie is a process which has terminated but its entry still exists in the process table until the parent terminates normally or calls wait().
- Suppose you create a child process and the child finishes before the parent process does.
- Running the ps command before the parent gets terminated, the output of ps will show the entry of a zombie process (denoted by defunct).

ALGORITHM:

1. Create a new child process using fork ()
2. If pid is equal to zero executer child process and terminate immediately using exit() system call
3. If pid is not equal to zero execute parent process and sleeps for 50 seconds
4. To see already terminated process i.e., zombie process, run the ps command which shows entry as defunct

PROGRAM:

```
//zombie.c #include<stdio.h> #include<unistd.h>
int main()
{
pid_t t; t=fork(); if(t==0)
{
printf("Child having id %d\n",getpid()); exit(0);
}
else
{
printf("Parent having id %d\n",getpid());
sleep(50); // Parent sleeps. Run the ps command during this time
}
}
```

OUTPUT:

```
$ gcc zombie.c
./a.out & (execute the program in the background) Parent having id 3687
Child having id 3688
```

```
ps

PID       TTY    TIME        CMD
3033      pts/0  00.00.00    bash
3687      pts/0  00.00.01    a.out
3688      pts/0  00.00.01    a.out <defunct>
3689      pts/0  00.00.01    ps
```

Orphan Process

• A process whose parent process no more exists i.e., either finished or terminated without waiting for its child process to terminate.

• An orphan process is a process whose parent has finished.

• Suppose P1 and P2 are two process such that P1 is the parent process and P2 is the child process of P1. Now, if P1 finishes before P2 finishes, then P2 becomes an orphan process.

• The init process is assigned as the new parent to the orphan process

• The init process periodically invokes wait(), thereby allowing the exit status of any orphaned process to be collected and releasing orphan's process

ALGORITHM

1. Create a new child process using fork()
2. If pid is not equal to zero execute parent process block of code and terminates
3. If pid is equal to zero executes child process and block process for 5 seconds in mean time parent process terminates.
4. To see child process has became orphan process open new command prompt and type ps -a it shows status O means that process is orphan process.

PROGRAM:

```
//orphan.c
#include<stdio.h
#include<unistd.h>
#include<sys/types.h> int main()
{
pid_t p; p=fork(); if(p==0)
{
sleep(5); //child goes to sleep and in the mean time parent terminates printf("I am child having PID
%d\n",getpid());
printf("My parent PID is %d\n",getppid());
}
else
{
printf("I am parent having PID %d\n",getpid()); printf("My child PID is %d\n",p);
}

}
```

OUTPUT:

```
$ gcc orphan.c
./a.out
I am parent having PID 138 My child PID is 139

$ps -a
I am child having PID 139 My parent PID is 1
```

RESULT:

Thus C Program is created to demonstrate Zombie and Orphan process successfully.