

## **Assignment 1**

### *Assignment 1(a):-*

Write a C program to create a child process using the system call `fork( )` and from the child process, display the PID and PPID and also display the same from parent process. Demonstrate the use of `exit(0)` and `exit(1)`.

Hints:

- \* There is a system call available in the LINUX to create a child process called `fork( )`. Again the `fork( )` system call returns the PID of the child process to the parent process and returns the value 0 to the child process. Moreover the return value type of the `fork( )` system call is `pid_t`.

- \* For `fork( )` system call , you need a header file `#include<sys/types.h>`

- \* To get the PID and PPID , we have two more system call `getpid( )` and `getppid( )` and the return types of both are `int`.

### *Assignment 1(b):-*

Write a C program like the assignment 1(a). But here use the system call `wait()` system to synchronize the execution of parent program in your program until child process finishes.

Hints:

- \* For `wait( )` system call , you need a header file `#include<sys/wait.h>`

### *Assignment 1(c):-*

Write a C program like the assignment 1(b) and over a user defined program into the address space of the child process using `execv()` system call. Again use `wait()` system call to synchronize the execution of parent program in your program until child process finishes. Here use the macro `WIFEXITED` to capture the returned status of the child in parent process. Also demonstrate the use of argument vector to print the program name by the child process.