1. **Uses a local and global variable to prove that fork creates a copy of the parent process and child has its own copy of variables to work on.**

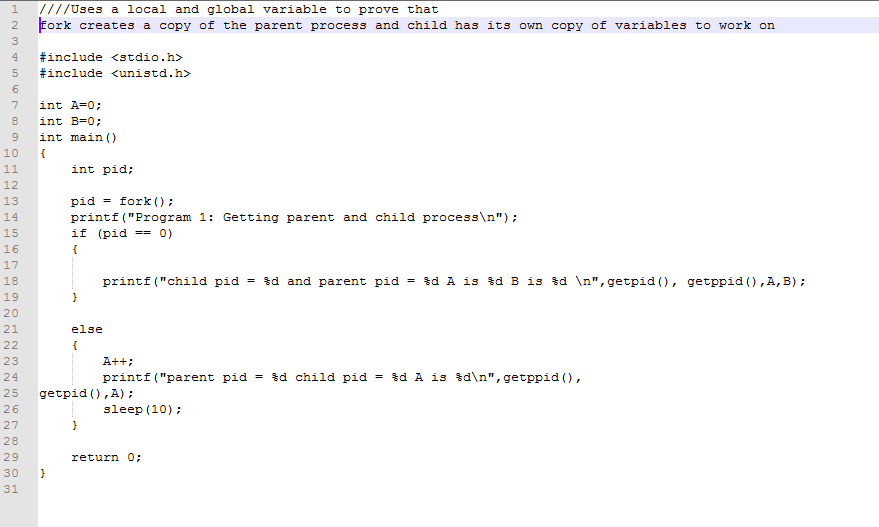
**Create a global variable A and initialize it to 0;**

**Create a child process using fork process and store the value in a variable.**

**Child process will give zero for fork().**

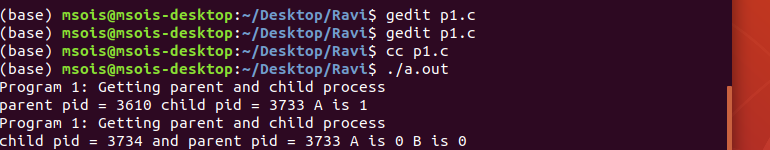
**So, increment the variable A in Parent process and print the same variable value in both parent and child process.**

**And declare one more variable inside the child process and print it in the Child itself.**

****

**OUTPUT:**

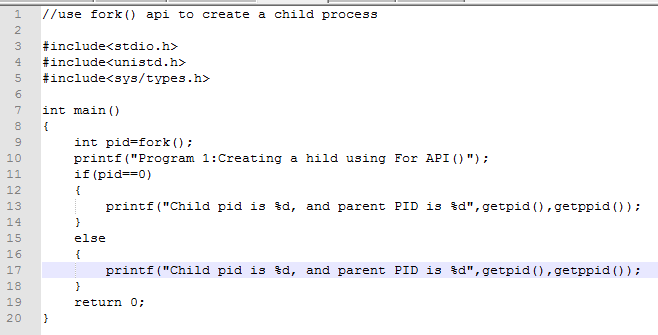
**The variable A is printed as 1 in parent process and 1 in child process as these two are two different processes.**

**The output is attached with both child process creation as well as Global and local variable declaration differences in the output.**

1. **Uses fork() API to create a child process**

**Create a child process using fork process and store the value in a variable.**

**Child process will give zero for fork().**

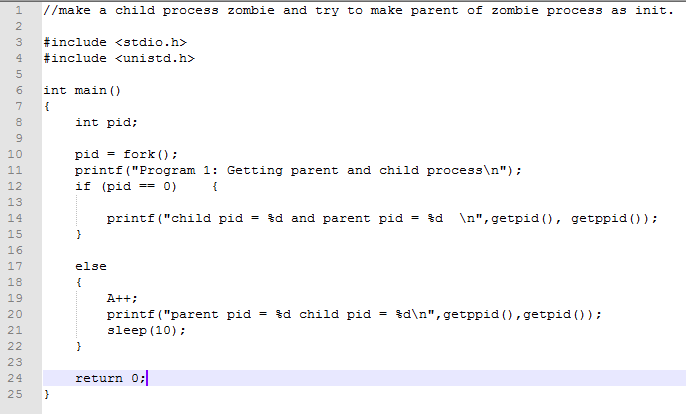
****

1. **make a child process zombie and try to make parent of zombie process as init.**

**Whenever the parent process is killed before its child is killed, In this case the parent of all the process,"init" process becomes the PPID This operation is called re-parenting.**

**Even though technically orphan process has the init process as parent it is called as orphan process since the process that originally created it no**

**longer exists.**

****

**OUTPUT:**