```
/*****************************
First Program: fork1.c
*****************************/

#include <stdio.h>
#include <sys/ipc.h>
main( )
{
   int i;
   if (fork( ) == 0) { /* Child */
         while (1) {   for (i=0; i<100000; i++) ;
                       sleep(4);
                       printf("\t\t\t Child executing\n ");
         }
   }
   else {              /* Parent */
         while (1) {   for (i=0; i<100000; i++) ;
                       sleep(7);
                       printf("Parent executing\n"); }
   }
}


OUTPUT:
-------

Child executing
Parent executing
                         Child executing
                         Child executing
Parent executing
                         Child executing
                         Child executing
.................




/*****************************************************
Second Program: fork2.c
*****************************************************/
#include <stdio.h>
#include <sys/ipc.h>
main()
{
   int i, x = 10, pid1, pid2 ;
   printf("Before forking, the value of x is %d\n", x);
   if ( ( pid1 = fork( ) ) == 0) { /* First child process */
          for (i=0 ; i < 5; i++) {
            printf("At first child: x= %d\n", x);
            x= x+10;         sleep(1) ; /* Sleep for 1 second */
          }
   }
   else {         /* Parent process */
        if ( ( pid2 = fork( ) ) == 0) { /* Second child */
              for (i=0 ; i < 5; i++) {
                  printf("At second child: x= %d\n", x);
                  x= x+20; sleep(1) ; /* Sleep for 1 second */
              }
        }
        else { /* Parent process */
                  waitpid(pid1,NULL,0);
                  waitpid(pid2,NULL,0);
                  printf("Both children terminated\n");
        }
 }
}
```

```
OUTPUT:
--------
Before forking, the value of x is 10
At first child: x= 10
At second child: x= 10
At first child: x= 20
At second child: x= 30
At first child: x= 30
At second child: x= 50
At first child: x= 40
At second child: x= 70
At first child: x= 50
At second child: x= 90
Both children terminated
```