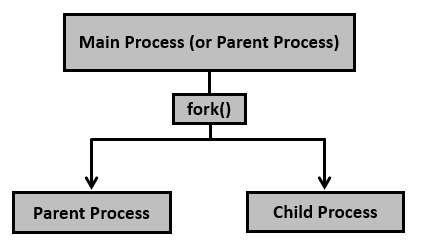
**System Calls**

By – Ankit kumar sharma

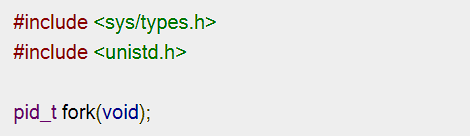
2023UCD2169

TOPIC : Process creation and termination for the operating system ( fork, wait, signal etc.)

#Process creation is achieved through the fork() system call. The newly created process is called the child process and the process that initiated it (or the process when execution is started) is called the parent process. After the fork() system call, now we have two processes - parent and child processes. How to differentiate them? Very simple, it is through their return values.



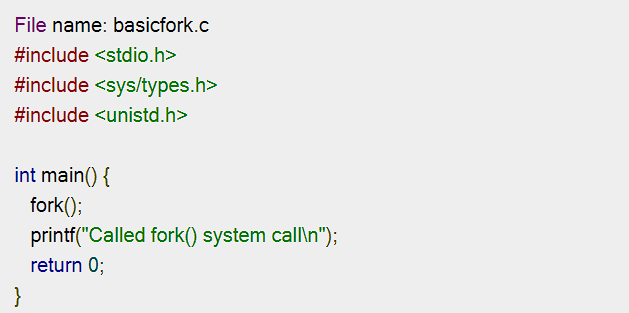
After creation of the child process, let us see the fork() system call details.

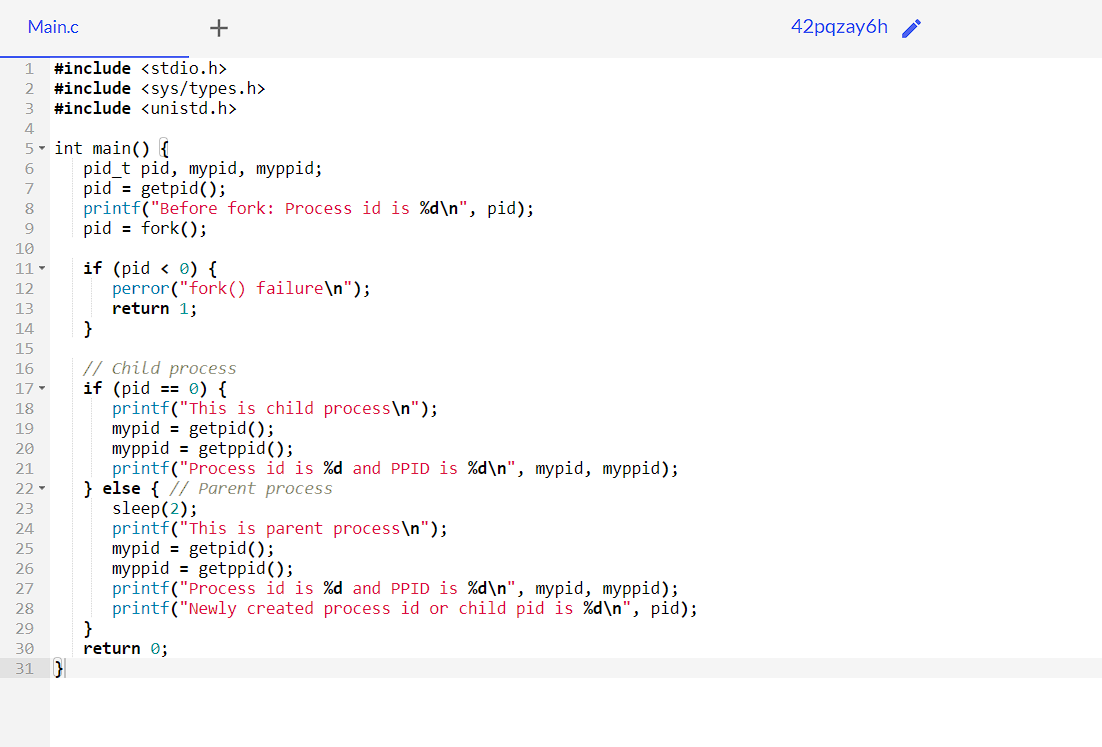


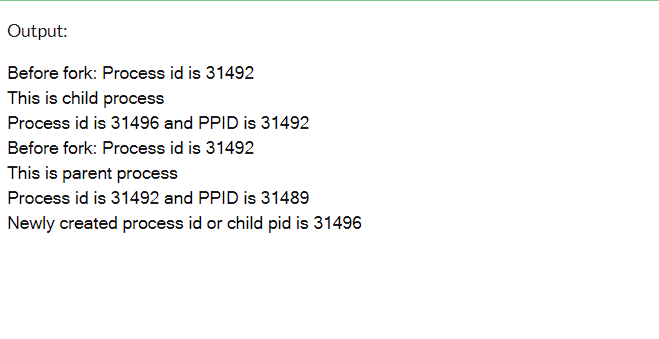
Creates the child process. After this call, there are two processes, the existing one is called the parent process and the newly created one is called the child process.

The fork() system call returns either of the three values −

* 1.Negative value to indicate an error, i.e., unsuccessful in creating the child process.
* 2.Returns a zero for child process.
* 3.Returns a positive value for the parent process. This value is the process ID of the newly created child process.

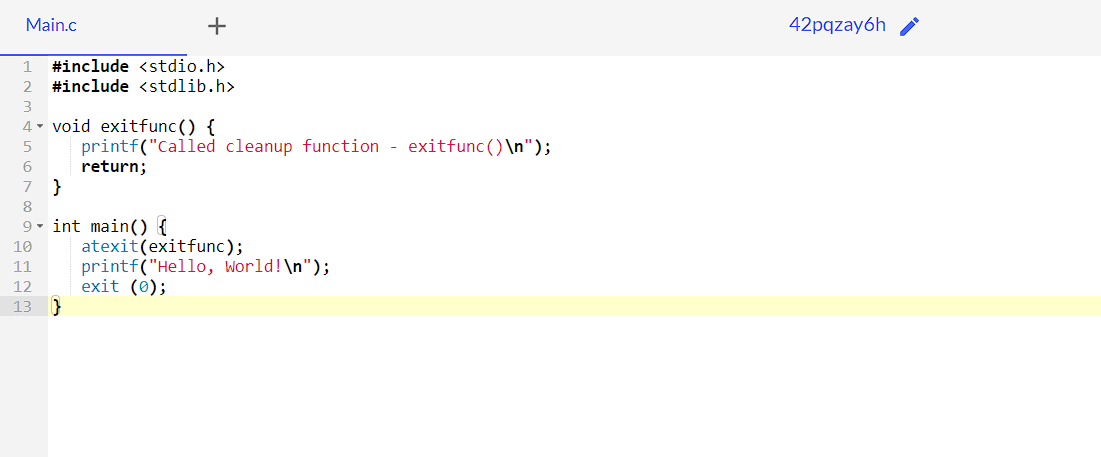
For Example : 

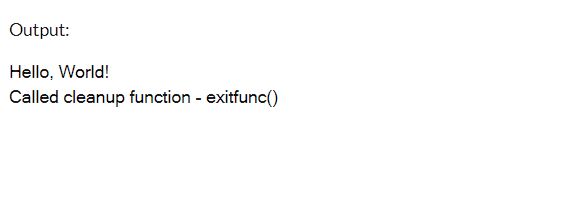
#Now new program to see details of parent and child processes



#Sleep (system call): The sleep system call is used to pause the execution of a program for a specified amount of time.

#exit() does some cleanup before returning the control back to the kernel, while the \_exit() (or \_Exit()) would return the control back to the kernel immediately.



Output:

Signal : It takes two arguments: a reference to a signal handler code and a signal number (one of those SIGsomethings). A single integer input representing a sent signal number is taken by the signal handler function, which returns void. In this manner, you can apply the same signal handler function to numerous signals.

Syntax:

signal(SIGINT, sig\_handler);