

SAMPLE PROGRAM TO DEMONSTRATE `fork()` SYSTEM CALL OPERATION

```
#include <stdio.h>
#include <sys/types.h>

#define MAX_COUNT 200

void ChildProcess(void);          /* child process prototype */
void ParentProcess(void);        /* parent process prototype */

void main(void)
{
    pid_t pid;

    pid = fork();
    if (pid == 0)
        ChildProcess();
    else
        ParentProcess();
}

void ChildProcess(void)
{
    int i;

    for (i = 1; i <= MAX_COUNT; i++)
        printf("    This line is from child, value = %d\n", i);
    printf("    *** Child process is done ***\n");
}

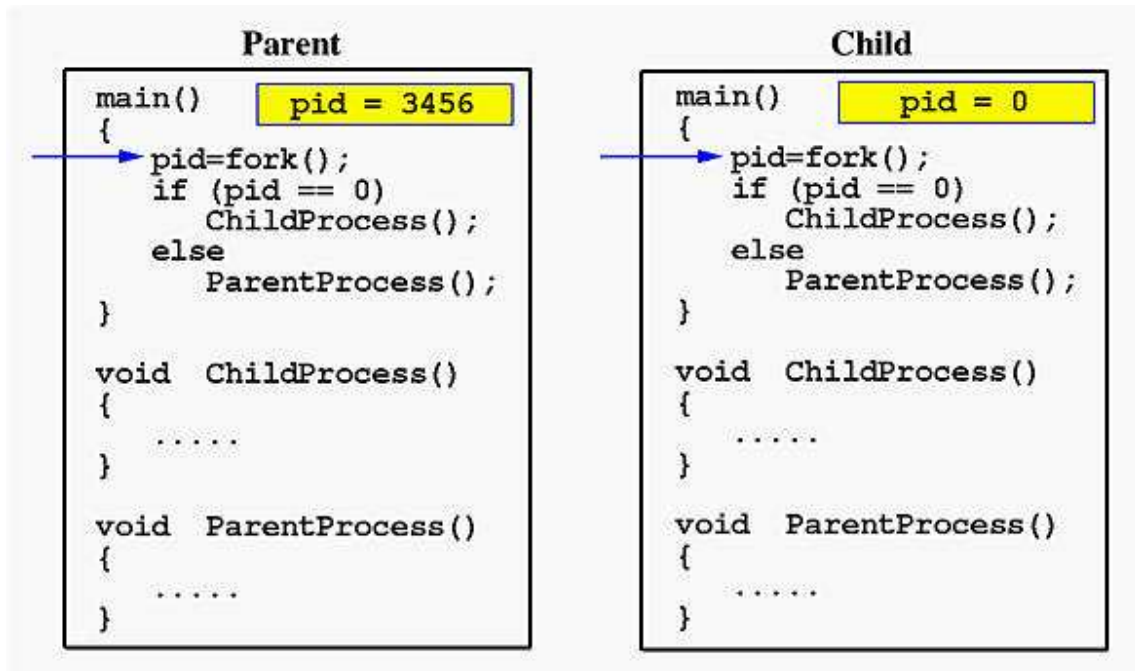
void ParentProcess(void)
{
    int i;

    for (i = 1; i <= MAX_COUNT; i++)
        printf("This line is from parent, value = %d\n", i);
    printf("*** Parent is done ***\n");
}
```

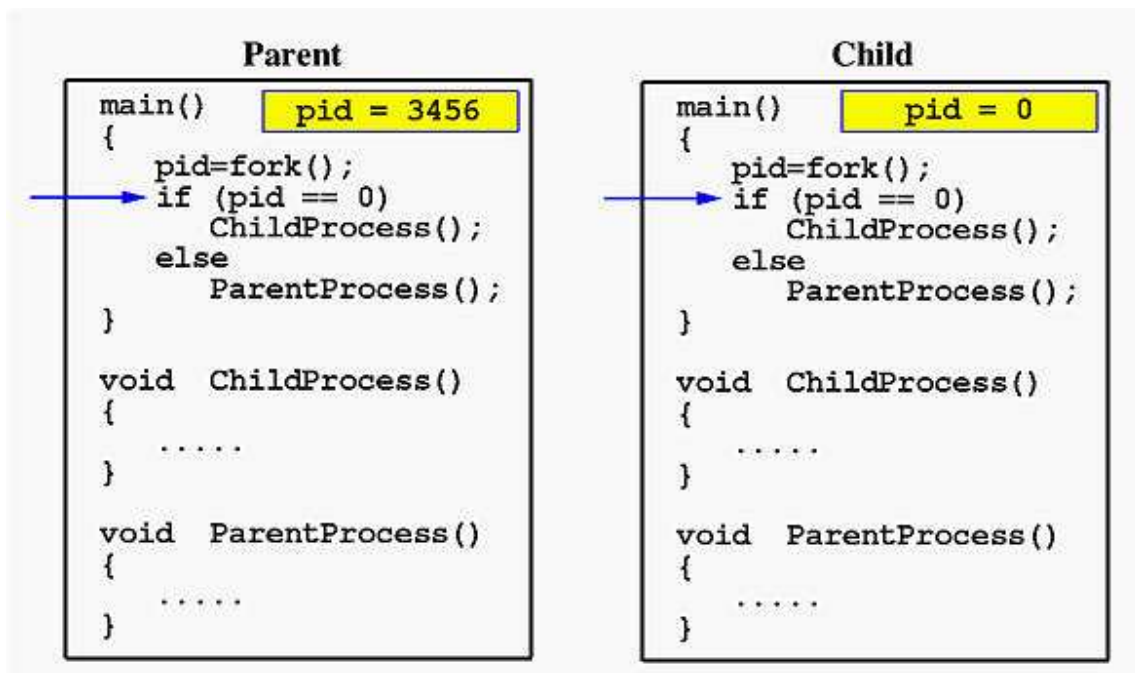
In this program, both processes print lines that indicate

- (1) whether the line is printed by the child or by the parent process, and
- (2) the value of variable **i**.

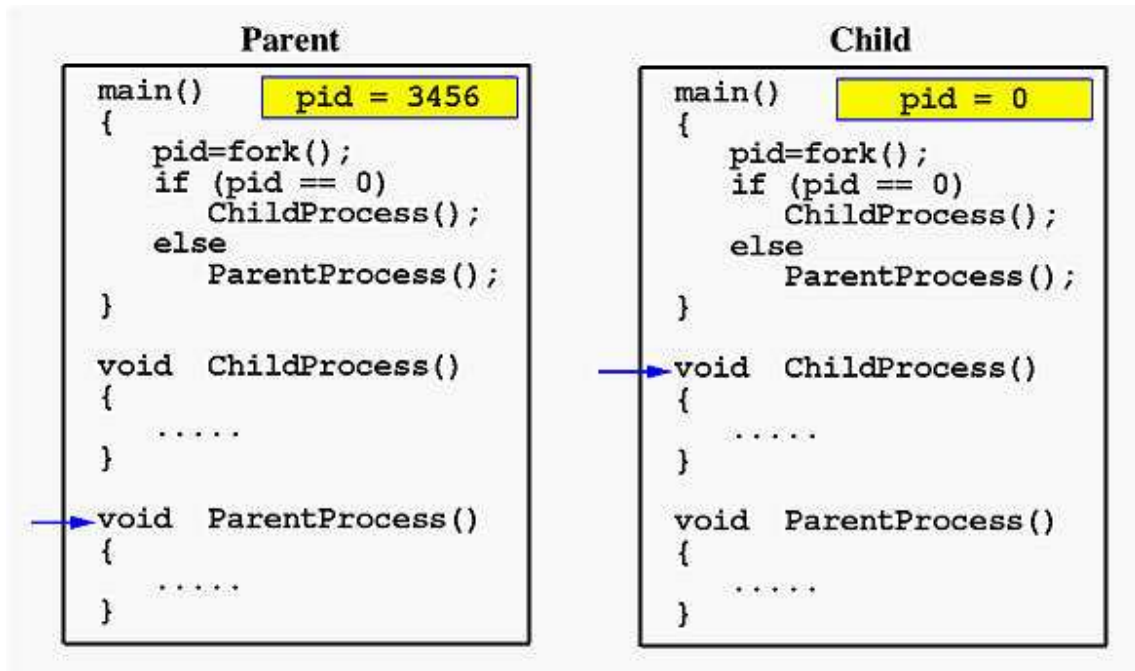
When the main program executes **fork()**, an identical copy of its address space, including the program and all data, is created. System call **fork()** returns the child process ID to the parent and returns 0 to the child process. The following figure shows that in both address spaces there is a variable **pid**. The one in the parent receives the child's process ID 3456 and the one in the child receives 0.



Now both programs (*i.e.*, the parent and child) will execute independent of each other starting at the next statement:



In the parent, since **pid** is non-zero, it calls function **ParentProcess()**. On the other hand, the child has a zero **pid** and calls **ChildProcess()** as shown below:



Due to the fact that the CPU scheduler will assign a time quantum to each process, the parent or the child process will run for some time before the control is switched to the other and the running process will print some lines before you can see any line printed by the other process. Therefore, the value of **MAX_COUNT** should be large enough so that both processes will run for at least two or more time quanta. If the value of **MAX_COUNT** is so small that a process can finish in one time quantum, you will see two groups of lines, each of which contains all lines printed by the same process.