UNIX FORKS

Fork is a system call that creates a new process under the UNIX operating system. When the fork( ) system call is executed, another process is created. The original process is called the parent process and the second process is called the child process. The child process is an almost exact copy of the parent process. Both processes continue executing from the point where the fork( ) calls returns execution to the main program. Since UNIX is a time-shared operating system, the two processes can execute concurrently.

Some differences between the child and parent process are:

- different pids
- in the parent, fork( ) returns the pid of the child process if a child process is created
- in the child, fork( ) always returns 0
- separate copies of all data, including variables with their current values and the stack
- separate program counter (PC) indicating where to execute next; originally both have the same value but they are thereafter separate
- after fork, the two processes do **not** share variables

fork returns:

- the pid of the new child process: to the parent process; this is equivalent to telling the parent the name of its child.
- 0: to the child process
- -1: 1 if there is an error; i.e., fork( ) failed because a new process could not be created

Example: Calculate number of times welcome is printed:

```
#include <stdio.h>
#include <sys/types.h>
int main()
{
   fork();
   fork();
   fork();
   printf("welcome\n");
   return 0;
}
```

Output:

```
welcome
welcome
 welcome
```


By: Pragya Singh Tomar
ICS, Vikram University, Ujjain

welcome
welcome
welcome
welcome
welcome

The number of times ‘hello’ is printed is equal to number of process created. Total Number of Processes = 2^n, where n is number of fork system calls. So here n = 3, $2^3 = 8$

Let us put some label names for the three lines:

```
fork ();   // Line 1
fork ();   // Line 2
fork ();   // Line 3
```

```
      L1          // There will be 1 child process created by line 1.
   /     \
  L2      L2       // There will be 2 child processes created by line 2
 /  \    /  \
L3  L3  L3  L3     // There will be 4 child processes created by line 3
```

So there are total eight processes (new child processes and one original process).

If we want to represent the relationship between the processes as a tree hierarchy it would be the following:

The main process: P0
Processes created by the 1st fork: P1
Processes created by the 2nd fork: P2, P3
Processes created by the 3rd fork: P4, P5, P6, P7

```
          P0
       /  |  \
     P1   P4   P2
    /  \        \
   P3   P6       P5
  /
 P7
```


By: Pragya Singh Tomar
ICS, Vikram University, Ujjain

NOTE: Parent process and child process are running the same program, but it does not mean they are identical. OS allocate different data and states for these two processes, and the control flow of these processes can be different.

By: Pragya Singh Tomar
ICS, Vikram University, Ujjain