Name : Hrushikesh Vijaykumar Bhosale
PRN : 2020BTEIT00047

Objectives:

1.To learn about Processing Environment.

2. To know the difference between fork/vfork and various execs variations.

3. Use of system call to write effective programs.

    1. Write the program to use fork/ vfork system call. Justify the difference by using suitable application of fork/vfork system calls. (I)

Code :

```
#include <stdio.h>
#include <unistd.h>

int main() {
   int pid;

   // Use fork system call to create a new process
   pid = fork();

   if (pid == 0) {
      // Child process
      printf("I am the child process (pid: %d)\n", getpid());
   } else {
      // Parent process
      printf("I am the parent process (pid: %d)\n", getpid());
   }

   // Use vfork system call to create a new process
   pid = vfork();

   if (pid == 0) {
      // Child process
      printf("I am the child process (pid: %d)\n", getpid());
   } else {
      // Parent process
      printf("I am the parent process (pid: %d)\n", getpid());
   }

   return 0;
}
```

Theory:

The fork system call creates a new process by duplicating the calling process. The new process, called the child process, is an exact copy of the calling process, called the parent process. Both the parent and child process continue execution from the point where fork was called.The fork system call creates a new process by duplicating the calling process. The new process, called the child process, is an exact copy of the calling process, called the parent process. Both the parent and child process continue execution from the point where fork was called.

The vfork system call also creates a new process, but it has some important differences from fork. The main difference is that the child process created by vfork shares the same memory space as the parent process, while the child process created by fork has its own memory space. This means that any changes made by the child process to its memory space will be visible to the parent process, and vice versa. Because of this, the child process created by vfork should not return or call exec system call family function.

In general, vfork is less resource-intensive than fork, since the child process shares
In general, vfork is less resource-intensive than fork, since the child process shares the memory space of the parent process. But it's not recommended to use vfork in general scenarios. vfork is mainly used in situations where the child process will shortly call exec to replace itself with a new process image