Processing Environment

Subject:- Unix Operating System

Class :- TYIT

Name **PRN**

Dipankar Dubey **2020BTEIT00057**

Assignment No - 1c

Title-Write the program to use fork/vfork system call. Justify the difference by using suitable application of fork/vfork system calls.

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.

Theory-

fork():

The fork() is a system call use to create a new process. The new process created by the fork() call is the child process, of the process that invoked the fork() system call. The code of child process is identical to the code of its parent process. After the creation of child process, both process i.e. parent and child process start their execution from the next statement after fork() and both the processes get executed simultaneously.

vfork():

The modified version of fork() is vfork(). The vfork() system call is also used to create a new process. Similar to the fork(), here also the new process created is the child process, of the process that invoked vfork(). The child process code is also identical to the parent process code. Here,the child process suspends the execution of parent process till it completes its execution as both the process share the same address space to use.

Basis for comparision	Fork()	VFork()
Basic	Child process and parent process has separate address spaces.	Child process and parent process shares the same address space.
Execution	Parent and child process execute simultaneously.	Parent process remains suspended till child process completes its execution
Modification	If the child process alters any page in the address space, it is invisible to the parent process	If child process alters any page in the address space, it is visible to the parent

	as the address space are separate.	process as they share the same address space.
Copy-on-write	fork() uses copy-on-write as an alternative where the parent and child shares same pages until any one of them modifies the shared page	vfork() does not use copy-on-write

Program-

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

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
void forkexample()
{

        if (fork() == 0)
                printf("Hello from Child!\n");
        else
                printf("Hello from Parent!\n");
}
int main()
{
        forkexample();
        return 0;
}
```

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
int main()
{
   int n =10;
   pid_t pid = vfork();
   if (pid == 0)
   {
      printf("Child process started\n");
   }
   else
```

```
    {
       printf("Now i am coming back to parent process\n");
    }
    printf("value of n: %d \n",n);
    return 0;
}
```

Output-

```
 it@it-OptiPlex-3050:~/Desktop/57/UOS$ gcc 1c.c
it@it-OptiPlex-3050:~/Desktop/57/UOS$ ./a.out
Hello world!
Hello world!
it@it-OptiPlex-3050:~/Desktop/57/UOS$ gedit 1c.c
it@it-OptiPlex-3050:~/Desktop/57/UOS$ gcc 1c.c
it@it-OptiPlex-3050:~/Desktop/57/UOS$ ./a.out
Hello from Parent!
Hello from Child!
it@it-OptiPlex-3050:~/Desktop/57/UOS$ gedit 1c.c
it@it-OptiPlex-3050:~/Desktop/57/UOS$ gcc 1c.c
it@it-OptiPlex-3050:~/Desktop/57/UOS$ ./a.out
Child process started
value of n: 10
Now i am coming back to parent process
value of n: 1424741904
```

Conclusion:

fork() and vfork() system calls have some differences which allows different type of execution of child processes.

References:

www.tutorialspoint.com/unix_system_calls/