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 wait/ waitpid system call and explain what it do when call in parent and called in child (). Justify the difference by using suitable application . (I)

Code :

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
   int pid, status;

   // 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());
      sleep(5); // simulate some work
      return 0; // exit child process
   } else {
      // Parent process
      printf("I am the parent process (pid: %d)\n", getpid());

      // Use wait system call to wait for child process to exit
      wait(&status);

      printf("Child process has exited with status %d\n", status);
   }

   // 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());
      sleep(5); // simulate some work
      return 0; // exit child process
   } else {
      // Parent process
      printf("I am the parent process (pid: %d)\n", getpid());

      // Use waitpid system call to wait for child process to exit
```

```
        waitpid(pid, &status, 0);

        printf("Child process has exited with status %d\n", status);
    }

    return 0;
}
```

Theory:

In the above program, fork system call is used to create child process from the parent process. The child process runs the code inside the if (pid == 0) block, and the parent process continues to run the code inside the else block.

The parent process uses waitpid system call to wait for the specific child process to exit. The first argument of waitpid is the process ID of the child process to wait for, and the second argument is a pointer to a variable where the exit status of the child process will be stored. Additionally, waitpid accept a third argument which is a set of flags, these flags can be used to change the behavior of the waitpid call.

On the other hand, the child process calls wait system call. This will cause the child process to block and wait for any of its sibling process to exit. Once a sibling process exits, the child process can retrieve the exit status of the sibling process using the status argument passed to wait.

It's worth noting that, in the above program, the child process calling wait() is not a recommended practice because it is not a parent process and it doesn't have any child process to wait for, so it's not necessary to call wait() in child process. But the parent process calling waitpid() is a good practice as it ensures that parent process is waiting for the specific child process to exit. This can be useful in situations where a parent process has multiple child processes and wants to wait for a specific one to exit