# 1a: Processing Environment

**Name : Hruhsikesh Vijaykumar Bhosale**
**PRN : 2020BTEIT00047**

## a. Write the application or program to open applications of Linux by creating new processes using fork system call. Comment on how various application's/command's process get created in linux.

**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.

**Code :**

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

int main() {

   pid_t pid;

   pid = fork();

   if (pid == 0) {

      // child process
      char *args[] = {"/usr/bin/gedit", NULL};
      execvp(args[0], args);
      exit(0);

   } else if (pid > 0) {

      // parent process
      printf("Child process created with PID %d\n", pid);

   } else {

      // fork failed
      printf("Fork failed\n");
```

```
    }

    return 0;
}
```

**Theory:**

In this example, the fork() system call creates a new child process, which then uses the execvp() function to open the "gedit" text editor application. The parent process, which is the original program, prints a message indicating the child process's PID.

In Linux, new processes are typically created in one of three ways:

fork() system call, as shown in the example above, creates a new child process that is an exact copy of the parent process.
exec() family of functions, such as execvp(), replaces the current process with a new process.
system() library function creates a new process and waits for it to complete.
Another common way to create new process is through spawn commands like systemd-run, dbus-launch and start-stop-daemon. They all use the above mentioned system calls to create new process and also helps in managing and monitoring those processes.